Ontwerp en computationele aspecten van flexibele tensegrity-robots

Design and Computational Aspects of Compliant Tensegrity Robots

Ken Caluwaerts

Promotor: prof. dr. ir. B. Schrauwen
Proefschrift ingediend tot het behalen van de graad van
Doctor in de Ingenieurswetenschappen: Computerwetenschappen

Vakgroep Elektronica en Informatiesystemen
Voorzitter: prof. dr. ir. J. Van Campenhout
Faculteit Ingenieurswetenschappen en Architectuur
Academiejaar 2013 - 2014

UNIVERSITEIT
GENT

ISBN 978-90-8578-709-9
NUR 950
Wettelijk depot: D/2014/10.500/55

Examencommissie

Prof. Jan Van Campenhout, chair
ELIS Department, Faculty of Engineering and Architecture
Ghent University

Prof. Joni Dambre, secretary
ELIS Department, Faculty of Engineering and Architecture
Ghent University

Prof. Benjamin Schrauwen, supervisor
ELIS Department, Faculty of Engineering and Architecture
Ghent University

Prof. Alain Sarlette
SYSTEMS Department, Faculty of Engineering and Architecture
Ghent University

Prof. Hod Lipson
Creative Machines Lab, School of Mechanical and Aerospace Engineering
Cornell University

M.Sc. Vytas SunSpiral
NASA Ames Intelligent Robotics Group, Intelligent Systems Division
SGT Inc., on-site contractor at NASA Ames Research Center

Summary

This dissertation explores the computational and hardware design aspects of compliant tensegrity robots and structures. I first focus on the control and computational features of robots based on the tensegrity design principle. Afterwards, I present the design of two hardware platforms developed to investigate the practical aspects of compliant tensegrity designs. In addition to this, I provide a general overview of the statics and dynamics of tensegrities.

Tensegrity Structures

Tensegrities (*tensile-integrity*) are structures in which compressive elements (e.g. bars, struts) are held together by tensile elements (e.g. springs or cables). Due to the specific arrangement of the members, it is possible to make highly efficient use of materials as only axial forces are present (no bending or shear forces).

More precisely, one can build free-standing structures consisting of only struts and cables in which no two struts are connected to each other. Forces diffuse throughout the whole system instead of concentrating at the joints, decreasing the risk of failure due to impacts. Additionally, tensegrity structures can be folded for efficient storage in tight spaces such as payload fairings. The combination of these properties makes them ideal candidates for environments requiring robust and capable robot designs.

While tensegrities were invented decades ago by Buckminster Fuller and Kenneth Snelson, only a handful of robotic systems have applied this design principle. One of the main reasons for this is the technical difficulty of constructing and controlling such structures, due to the subtle interplay between tensile and compressive forces which propagate through the structure.

In the first part of this dissertation I provide an in-depth overview of the statics and dynamics of tensegrity structures. I propose new methods

to optimize the shape of a compliant tensegrity structure and to tune the stiffness of structures with a redundant number of controllable tensile elements. Additionally, the two simulation environments used throughout this work are introduced.

Computational Aspects

Afterwards, I redirect my attention to the computational aspects of tensegrity robots. I first cast tensegrities as a specific type of Physical Reservoir Computing and demonstrate how this allows feedback controllers for tensegrity robots to be implemented. Reservoir Computing is originally an efficient technique to use Recurrent Neural Networks. The method works by training only an output layer while leaving the internal network (or Reservoir) intact. This contrasts with more established methods which aim at training the full network. A number of physical implementations of Reservoir Computing have emerged in recent years, particularly in the photonics and electronics fields. It is in the realm of Physical Reservoir Computing that I present simple, yet efficient computational techniques for compliant tensegrity robots.

I then extend this computational approach by introducing learning rules based on Reward Modulated Hebbian plasticity. I demonstrate that similar learning rules can be applied to tensegrity robots and Recurrent Neural Networks. This provides an interesting perspective of the similarity between highly compliant robots and the well established Neural Network research domain.

Design of Tensegrity Robots

A considerable part of my PhD research was dedicated to the design of a compliant tensegrity robot nicknamed ReCTeR, which is short for Reservoir Compliant Tensegrity Robot. ReCTeR is a small (1 m diameter), untethered, lightweight (1.1 kg) and underactuated (6 DC motors) robot. Its design is based on the common tensegrity icosahedron (6 struts, 24 tensile elements). However, the 6 actuated tensile members are not included in the 24 standard tensile members and instead run through the structure. This design allows low power motors to considerably deform the robot.

The robot's two fundamental hardware components are tension springs and carbon fiber tubes. Carbon fiber struts reduce the mass of the robot, without reducing its mechanical strength. ReCTeR has been dropped both on purpose and accidentally from heights up to 1 m. None of these impacts has resulted in mechanical failure, which is remarkable considering the fragility of the carbon fiber struts in bending. This clearly illustrates one of the benefits

of a tensegrity design. The tensile members are off-the-shelf linear springs attached to high-strength wires which connect the ends of the carbon fiber struts. Actuated tensile elements have a rotational DC motor connected to a wire spindle which wraps a wire attached to a spring, thus effectively modifying the rest length of the linear spring.

ReCTeR can fold, deploy and roll and has a battery life of over 30 min with all systems active. This makes it possible to validate the simulators and techniques employed throughout this work based on experimental results obtained with this robot. Despite its low weight, ReCTeR is equipped with a large amount of sensors to enable feedback control.

Tensegrities for Space Exploration

In the last Chapter of this work, I focus on tensegrity robots for space exploration. I have contributed to the NASA Innovative Advanced Concepts Tensegrity project and have had the opportunity to visit the Intelligent Robotics Group at the NASA Ames Research Center in California. During this research period, I worked on SUPERball (an acronym for Spherical Underactuated Planetary Exploration Robot). SUPERball is a next generation tensegrity robotics research platform. The main differences between this new platform and ReCTeR are that SUPERball is larger ($\approx$1.5 m diameter, $\approx$15 kg), more robust and highly modular. Its fundamental objective is to provide an extensible tensegrity robotics research platform rather than a single robot.

The current target hardware design is a tensegrity icosahedron with 12 actuated tensile members. Unlike ReCTeR, the springs are embedded into the compressive members of the new robotic platform. This allows for a safer construction that is capable of handling large tensile forces (which occur during an impact) without the risk of wires wrapping around springs or plasticly deforming. The main hardware design aspects such as the wire spindle design have been inspired by ReCTeR's, but significant improvements have been made. While ReCTeR provides modularity at the strut level, SUPERball's end caps are fully independent. Each end cap hosts a battery, an actuator, springs and all necessary electronics. An original design allows to fix an end cap onto a strut with a single bolt. This makes it possible to quickly experiment with modified actuator and sensor designs.

Currently, one strut has been built and construction of the full robot is expected by mid 2014. Successful transfer of the control methods to the hardware platform is planned within a year.

Samenvatting

Tensegrity-robots en -structuren vormen het hoofdonderwerp van dit doctoraat. In het eerste deel van dit werk focus ik mij op de computationele aspecten van flexibele robots gebaseerd op het tensegrity-principe. Nadien stel ik het ontwerp van twee robots voor die gebouwd werden om de praktische aspecten van flexibele tensegrity's te bestuderen. Bovendien geef ik een overzicht van de statische en dynamische eigenschappen van tensegrity-structuren.

Flexibel betekent meegaand (*compliant* in het Engels) in de context van dit werk. Robots van dit type kunnen soepele interacties met hun omgeving aangaan, in tegenstelling tot stijve en vaak gevaarlijke klassieke industriële robots. Meegaandheid kan op verschillende manieren gerealiseerd worden, bijvoorbeeld door zeer snelle controle in combinatie met krachtsensoren (*active compliance*). Alle robots beschouwd in dit werk bevatten echter flexibele elementen – zoals veren – en zijn dus inherent meegaand (*passive compliance*).

Tensegrity-structuren

Tensegrity's (afkomstig van het Engelse *tensile-integrity*) zijn structuren waarin constructie-elementen in compressie worden samengehouden door een set van elementen die enkel onderhevig zijn aan treklasten. Door de specifieke configuratie van deze elementen is optimaal gebruik van materialen mogelijk, omdat ieder element enkel aan axiale krachten onderhevig is (geen buigingen of schuifkrachten).

Meer bepaald maakt dit mogelijk om vrijstaande structuren te bouwen die enkel bestaan uit (flexibele) kabels en compressie-elementen waarbij er geen twee compressie-elementen met elkaar verbonden zijn. Dit heeft tot resultaat dat inwerkende krachten zich verspreiden doorheen de hele structuur in plaats van zich te concentreren ter hoogte van de gewrichten.

Bovendien kunnen tensegrity-structuren opgeplooid worden, zodat ze efficiënt opgeslagen kunnen worden in een beperkte ruimte. Deze combinatie van eigenschappen maakt tensegrity's uitermate geschikt voor omgevingen waarin een robuust en veelzijdig robotontwerp gewenst is.

Hoewel het tensegrity-principe meerdere decennia geleden werd uitgevonden door Buckminster Fuller en Kenneth Snelson, zijn er tot op heden maar een beperkt aantal tensegrity-robots gebouwd. De voornaamste reden hiervoor is de complexiteit van het ontwerp en de regelaars ten gevolge van het subtiele samenspel van trek- en duwkrachten die zich doorheen de volledige structuur verspreiden.

In het eerste deel van dit doctoraat ga ik dieper in op de statische en dynamische aspecten van tensegrity-structuren. Ik beschrijf nieuwe methoden voor de optimalisatie van de vorm van een tensegrity-structuur en van de stijfheid van structuren met een redundant aantal regelbare kabels. Bovendien stel ik de twee simulators voor die ik heb gebruikt voor dit werk.

Computationele aspecten

Vervolgens richt ik mijn aandacht op de computationele aspecten van tensegrity-robots. In de eerste plaats maak ik gebruik van het Reservoir Computing principe voor fysieke systemen en demonstreer hoe regelaars met terugkoppeling voor tensegrity-robots gerealiseerd kunnen worden. Reservoir Computing is oorspronkelijk een efficiënte leertechniek voor recurrente neurale netwerken. In tegenstelling tot meer courante methodes wordt bij Reservoir Computing enkel een uitleeslaag getraind, het interne netwerk (Reservoir in het Engels) blijft onaangeroerd. Klassieke methodes trachten in de meeste gevallen alle verbindingen te optimaliseren, wat problemen oplevert als de precieze dynamica van het systeem niet geweten is. Recent is men Reservoir Computingtechnieken beginnen te ontwikkelen voor fysieke systemen. Een fysiek systeem, bijvoorbeeld gebaseerd op optische versterkers of elektronische elementen, neemt in dit geval de plaats in van het interne neurale network. Het is in dit kader dat ik Reservoir Computing voor fysieke systemen uitbreid naar tensegrity-robots en eenvoudige, doch efficiënte leeralgoritmes voorstel voor flexibele tensegrity-robots.

Nadien zet ik deze lijn verder door het invoeren van leerregels gebaseerd op beloningsgemoduleerde Hebbiaanse theorie. Ik toon aan dat gelijkaardige leerregels van toepassing zijn op tensegrity-robots en recurrente neurale netwerken. Dit biedt een interessant perspectief voor de gelijkenis tussen zeer flexibele robots en neurale netwerken.

Ontwerp van tensegrity-robots

Een significant deel van mijn doctoraatsonderzoek heb ik gewijd aan het ontwerp van een flexibele tensegrity-robot die ReCTeR (Reservoir Compliant Tensegrity Robot) werd gedoopt. ReCTeR is een vrij kleine (1 m diameter), draadloze, lichte (1.1 kg) en ondergeactueerde (6 gelijkstroommotoren) robot. Het ontwerp van deze robot is gebaseerd op het standaard tensegrity-icosaëder (6 elementen in compressie, 24 in treklast). De 6 aangedreven verbindingen behoren echter niet tot de 24 elementen in treklast uit het standaardontwerp. Ze lopen doorheen de robot, wat toelaat om minder krachtige motoren te gebruiken die toch grote vormveranderingen van de robot teweeg kunnen brengen.

De twee voornaamste structurele elementen van de robot zijn trekveren en koolstofvezelstaven. De koolstofvezelstaven laten toe het gewicht te beperken, zonder aan robuustheid in te boeten. ReCTeR is meerdere keren opzettelijk of per ongeluk gevallen van hoogtes tot 1 m. Geen enkele van deze voorvallen had mechanische problemen tot gevolg. Dit is vrij opmerkelijk, gezien de koolstofvezelstaven slecht bestand zijn tegen buiging en niet-axiale belasting. Het toont duidelijk één van de voordelen van het tensegrity-principe aan. De gebruikte trekveren zijn vrij verkrijgbare lineaire veren die door middel van een kabel worden verbonden met de uiteinden van de koolstofvezelstaven. De aandrijving van de geactueerde verbindingen bestaat uit een gelijkstroommotor gekoppeld aan een kleine draadspoel. Op deze wijze kan de rustlengte van een veerverbinding aangepast worden.

ReCTeR is in staat om zichzelf plat te vouwen, te ontplooien en te rollen. De robot heeft een batterijtijd van meer dan 30 min met alle systemen en motoren geactiveerd. Deze eigenschappen maakten het mogelijk om ReCTeR te gebruiken ter validatie van de simulatoren en de ontwikkelde regeltechnieken. Ondanks het lage gewicht, is ReCTeR toch uitgerust met een groot aantal sensoren om het gebruik van terugkoppelingsregelaars mogelijk te maken.

Tensegrity-structuren voor de ruimtevaart

In het laatste hoofdstuk van dit doctoraat bespreek ik de toepassingen van tensegrity-robots voor de ruimtevaart. Tijdens mijn doctoraat had ik de unieke kans om mee te werken aan het NASA Innovative Advanced Concepts tensegrity-project. In dit kader heb ik de Intelligent Robotics Group van het NASA Ames Research Center in Californië bezocht. Tijdens deze onderzoeksperiode heb ik meegewerkt aan SUPERball (Spherical Underactuated Planetary Exploration Robot). SUPERball is een nieuw tensegrity-onderzoeksplatform dat een aantal ontwerpsaspecten van ReCTeR overneemt. De grote verschillen tussen het nieuwe platform en ReCTeR zijn dat SUPER-

ball groter en zwaarder is (≈1.5 m diameter, ≈15 kg), meer robuust is en vooral zeer modulair opgebouwd is. De fundamentele doelstelling is om een uitbreidbaar tensegrity-onderzoeksplatform aan te bieden in plaats van één robot met een vast ontwerp.

In het huidige ontwerp is SUPERball een tensegrity-icosaëder met 12 geactueerde kabels. In tegenstelling tot ReCTeR zijn alle veren van het nieuwe platform geïntegreerd in de staven. Dit maakt het ontwerp veiliger en laat bovendien grote treklasten op de kabels toe (wat voorvalt bij een val) zonder risico op plastische vervorming of het onderling verstrikt raken. Een aantal belangrijke mechanische componenten, zoals het ontwerp van de kabelaandrijving, zijn gebaseerd op ReCTeR. Desondanks zijn er significante verbeteringen aangebracht ten opzichte van het eerdere ontwerp. De staven van ReCTeR zijn modulair, maar SUPERball gaat een stap verder op dit vlak. Elk uiteinde van een staaf dat een kabel aandrijft, is volledig onafhankelijk. Batterijen, motoren, veren en alle elektronische onderdelen werden samengevoegd en het geheel kan met een enkele bout vastgezet worden aan een staaf. Dit laat toe snel te experimenteren met aangepaste ontwerpen voor de aandrijving en sensoren.

Op dit moment is een staaf van de robot volledig gebouwd. Voltooiing van de volledige robot wordt tegen midden 2014 verwacht. Nadien is de doelstelling om binnen het jaar de in simulatie ontwikkelde regelaars naar het nieuwe robot platform over te hevelen.

Dankwoord

Voor u verder leest, wil ik me alvast verontschuldigen bij iedereen die ik vergeet te vermelden in de onderstaande paragrafen. Tijdens een doctoraat kom je met veel personen in contact en dit dankwoord is slechts een momentopname tijdens de stressvolle laatste dagen van een lange rit. Een groot aantal personen droeg direct of indirect bij aan het uiteindelijke resultaat, al is het maar door een opmerking die me aan het denken zette of een vraag waarop ik niet meteen een duidelijk antwoord had. Dus sta mij toe nu al iedereen te bedanken die van zichzelf weet dat hij of zij heeft meegeholpen aan mijn opleiding of doctoraat.

Het zijn zeker en vast niet enkel deze laatste vier jaren die mijn proefschrift gemaakt hebben tot wat het is. Een doctoraat is een (lijdens)weg die je leert om zelfstandig en origineel wetenschappelijk onderzoek te verrichten. In tegenstelling tot een master- of bacheloropleiding, krijg je de kans zelf te kiezen wat je leest, bestudeert en neerschrijft (zolang je promotor ermee akkoord gaat). Dit is meteen ook de reden waarom ik voor een doctoraat koos. Je bouwt verder op de kennis die je vergaard hebt tijdens je opleidingen en opvoeding. Nu mag je proberen om met al deze blokjes kennis een bijdrage aan de wetenschap in elkaar te knutselen. Naast het feit dat je fier mag zijn dat je mogelijk iets hebt bijgedragen aan de stand van de wetenschap, zijn er natuurlijk de vele conferenties op leuke locaties en de zeer flexibele uren (dagen) die een doctoraatsopleiding interessant maken.

In de allereerste plaats wil ik mijn ouders persoonlijk bedanken. Ongetwijfeld waren zij mijn grootste steun gedurende mijn opleiding en doctoraat. Het is bijna onmogelijk om te overdrijven in de mate waarin ze mij bijgestaan hebben. Ze kwamen mij bezoeken waar ik ook studeerde (zelfs San Francisco is niet te ver voor hen) en ze zorgden ervoor dat ik mij optimaal op mijn studies kon toeleggen.

Spijtig genoeg kunnen niet al mijn grootouders meemaken dat hun kleinzoon *computerdokter* wordt. Hoewel het niet steeds even eenvoudig is om uit te leggen wat het doel is van robots met stokjes en veren, hoop ik dat ze fier kunnen zijn op dit werk. Natuurlijk wil ik mijn overige familieleden niet uit het oog verliezen en hen eveneens van harte bedanken.

Vervolgens richt ik een woordje van dank aan mijn promotor prof. dr. ir. Benjamin Schrauwen, die ik ondertussen ook gewoon Ben mag noemen. Zijn ononderbroken stroom van ideeën hebben zeker in de initiële fase van mijn

doctoraat bijgedragen aan de invulling ervan. Bij deze wens ik hem ook alle succes toe bij zijn toekomstige avonturen in de Verenigde Staten.

Michiel D'Haene heeft sterk bijgedragen tot het ontwerp van de elektronica voor mijn robot en zonder hem zou het eindresultaat heel wat minder elegant geweest zijn. Pieter-Jan Kindermans ken ik al van toen ik negen jaar geleden aan de universiteit begon en hem wil ik bovendien bedanken voor de praktische hulp in Californië. Mijn overige collega's en vrienden wil ik ook zeker en vast persoonlijk bedanken: Philémon Brakel, Michaël Burm, Pieter Buteneers, Juan Pablo Carbajal, Jonas Degrave, Sander Dieleman, Michiel Hermans, Tim Waegeman, Francis wyffels, Aäron van den Oord, Thibault Verhoeven en David Verstraeten. De logistieke ondersteuning van Marnix Vermassen dient natuurlijk ook vermeld te worden.

Martin Fiers en Thomas Van Vaerenbergh zijn de volgende op mijn lijstje. Bedankt om een bijdrage te mogen leveren aan CAPHE. Ik hoop dat Martin erin slaagt om samen met zijn collega's hieromtrent een succesvol bedrijf uit te bouwen.

Mehdi Khamassi, my master's thesis advisor in Paris, provided support at multiple points during my PhD and I hope that we'll find future occasions to collaborate.

Hoewel mijn collega's en andere academici de meest directe impact hebben gehad op dit proefschrift, zijn het mijn vrienden die ervoor zorgden dat ik gemotiveerd bleef en regelmatig wat kon ontspannen. Het is moeilijk om iedereen op te noemen, dus ik vermeld er maar een paar: Ina, Jani, Joris, Marianne, Tia ... en natuurlijk mijn vrienden die ik Parijs leerde kennen.

Vytas SunSpiral was an amazing supervisor during my stay at NASA. I hope everything works out over the next few months and that we can soon turn the basic ideas developed in this dissertation into a space-qualified tensegrity robot! I had an awesome time in the U.S. and I wish to thank my collaborators: Jonathan Bruce, Jérémie Despraz, Jeffrey Friesen, Atil Iscen, George Korbel, Sophie Milam, Kyle Morse, In Won Park, Alexie Pogue, Andrew Sabelhaus, Brian Tietz and the members of the Intelligent Robotics Group. Hopefully, we can continue collaborating in the near future. As they supported my work at the Ames Research Center, I am also indebted to Adrian Agogino and Terry Fong. On a related note, I wish to thank Mostafa Ajallooeian for informing me about the NASA tensegrity project.

Finally, I wish to express my gratitude to prof. dr. ir. Joni Dambre, prof. Hod Lipson, prof. dr. ir. Alain Sarlette and prof. dr. ir. Jan Van Campenhout for their guidance and valuable comments on this manuscript.

Ken Caluwaerts

This work was supported by a PhD Fellowship of the Research Foundation Flanders (FWO).

List of Publications

Journal Articles

Caluwaerts K. and Carbajal J.P.
Energy Conserving Constant Shape Optimization of Tensegrity Structures.
Submitted, 2014.

Caluwaerts K., Despraz J., Iscen A., Bruce J., Sabelhaus A.P., Schrauwen B. and SunSpiral V.
Design and Control of Compliant Tensegrity Robots Through Simulation and Hardware Validation.
Journal of the Royal Society Interface, 2014.

Iscen A., Caluwaerts K., Bruce J., Agogino A.K., SunSpiral V. and Tumer K.
Learning Tensegrity Locomotion Using Open-Loop Control Signals and Coevolutionary Algorithms.
Submitted, 2014.

Buteneers P., Caluwaerts K., Verstraeten D., Dambre J. and Schrauwen B.
Optimized Parameter Search for Large Datasets of the Regularization Parameter and Feature Selection for Ridge Regression.
Neural Processing Letters 38 (3): 403–416, 2013.

Caluwaerts K., D'Haene M., Verstraeten D. and Schrauwen B.
Locomotion Without a Brain: Physical Reservoir Computing in Tensegrity Structures.
Artificial Life 19 (1), 2013.

Fiers M., Van Vaerenbergh T., Caluwaerts K., Vande Ginste D., Schrauwen B., Dambre J. and Bienstman P.
Time-domain and Frequency-domain Modeling of Nonlinear Optical Components at the Circuit-level Using a Node-based Approach.
Journal of the Optical Society of America B-Optical Physics 29 (5): 896–900, 2012.

Caluwaerts K., Staffa M., N'Guyen S., Grand C., Dollé L., Favre-Félix A., Girard B. and Khamassi M.
A Biologically Inspired Meta-control Navigation System for the Psikharpax Rat Robot.
Bioinspiration & Biomimetics 7 (2), 2012.

Conferences

Bruce J., Caluwaerts K., Iscen A., Sabelhaus A.P. and SunSpiral V.
Design and Evolution of a Modular Tensegrity Robot Platform.
IEEE International Conference on Robotics and Automation (ICRA): 3483–3489, 2014.

Bruce J., Sabelhaus A.P., Caluwaerts K., Agogino A.M. and SunSpiral V.
SUPERball: Exploring Tensegrities for Planetary Probes.
International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS), 2014.

Sabelhaus A.P., Caluwaerts K., Bruce J., Agogino A.M. and SunSpiral V.
SUPERball: Modular Hardware for a Mobile Tensegrity Robot.
Sixth World Conference on Structural Control and Monitoring (WCSCM), 2014.

Caluwaerts K., wyffels F., Dieleman S. and Schrauwen B.
The Spectral Radius Remains a Valid Indicator of the Echo State Property for Large Reservoirs.
IEEE International Joint Conference on Neural Networks (IJCNN), 2013.

Caluwaerts K., Favre-Félix A., Staffa M., N'Guyen S., Grand C., Girard B. and Khamassi, M.
Neuro-Inspired Navigation Strategies Shifting for Robots: Integration of a Multiple Landmark Taxon Strategy.
Lecture Notes in Artificial Intelligence 7375: 62–73, 2012.

Caluwaerts K. and Schrauwen B.
The Body as a Reservoir: Locomotion and Sensing with Linear Feedback.
2nd International Conference on Morphological Computation (ICMC), 2011.

Fiers M., Van Vaerenbergh T., Dumon P., Caluwaerts K., Schrauwen B., Dambre J. and Bienstman P.
CAPHE: a Circuit-Level Time-Domain and Frequency-Domain Modeling Tool for Nonlinear Optical Components.
16th Annual Symposium of the IEEE Photonics Benelux Chapter: 277–280, 2011.

wyffels F., D'Haene M., Waegeman T., Caluwaerts K., Nunes C. and Schrauwen B.
Realization of a Passive Compliant Robot Dog.
IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechanics (BIOROB): 882–886, 2010.

Caluwaerts K. and Galayko D.
SystemC-AMS Modeling of an Electromechanical Harvester of Vibra-

tion Energy.
Forum on Specification, Verification and Design languages (FDL): 123–128, 2008.

Book Chapters

Caluwaerts K. and Galayko D.
Heterogeneous and Non-Linear Modeling in SystemC-AMS.
In M. Radetzki (Ed.): Languages for Embedded Systems and their Applications 36 (2): 113–128, Springer, 2009.

Contents

List of Acronyms

BLDC	BrushLess Direct Current
CAD	Computer-Aided Design
CAN	Controller Area Network
CPG	Central Pattern Generator
DC	Direct Current
DOF	Degree Of Freedom
EDL	Entry, Descent and Landing
ESN	Echo State Network
ESP	Echo State Property
GD	Gradient Descent
IK	Inverse Kinematics
IMU	Inertial Measurement Unit
MIPS	Million Instructions Per Second
MSE	Mean Squared Error
NIAC	NASA Innovative Advanced Concepts program
NMSE	Normalized Mean Squared Error
NTRT	NASA Tensegrity Robotics Toolkit
PCB	Printed Circuit Board
POM	PolyOxyMethylene
PRC	Physical Reservoir Computing
PTFE	PolyTetraFluoroEthylene
RC	Reservoir Computing
ReCTeR	Reservoir Compliant Tensegrity Robot
RLS	Recursive Least Squares
RMH	Reward Modulated Hebbian plasticity
ROS	Robot Operating System
SUPERball	Spherical Underactuated Planetary Exploration Robot
UHMWPE	Ultra-High-Molecular-Weight PolyEthylene

List of Symbols

$\boldsymbol{A}$	Equilibrium matrix
$\boldsymbol{C}$	Connectivity matrix
$\boldsymbol{E}$	Stress matrix
$\boldsymbol{f}$	Nodal forces in vector form
$\boldsymbol{F}$	Nodal forces in matrix form
$\boldsymbol{G}$	Expanded stress matrix
$\boldsymbol{K}$	Stiffness matrix
$\boldsymbol{l}$	Member length vector
$\boldsymbol{L}$	Member length diagonal matrix
$\boldsymbol{l}_0$	Member rest length vector
$\boldsymbol{L}_0$	Member rest length diagonal matrix
$\boldsymbol{n}$	Nodal coordinates in vector form
$\boldsymbol{N}$	Nodal coordinates in matrix form
p	Number of nodes of a tensegrity structure
$\boldsymbol{q}$	Force density vector
$\boldsymbol{Q}$	Force density diagonal matrix
s	Number of spring-cable assemblies of a tensegrity structure
$\boldsymbol{S}$	Spring constants diagonal matrix
$\boldsymbol{W}$	A weight matrix

1 Introduction

This work is an in-depth study of the computational and hardware design aspects of compliant tensegrity structures. I present a number of results related to the computational aspects of tensegrity structures and describe the hardware designs of two hardware platforms developed within the context of this thesis. Additionally, I provide background information on most aspects of tensegrity structures to cast this work as a general overview of the current state of tensegrity robotics.

1.1 Tensegrity Structures

On August 31th 1959 Buckminster Fuller filed a patent in which he described tensegrity structures as [57]

> ... it has seemed appropriate to characterize the structure further as comprising compression elements which are like *"islands" of compression in a "sea" of tension elements.*

In a less poetic phrasing tensegrity structures can be defined as a set of compression elements (bars, struts...) held together by a tension net (springs, cables...). Many common structures fit this description: suspension bridges, bicycle wheels... While the tensegrity concept is indeed sometimes used to describe such a broad spectrum of systems, the more common use of the term is restricted to structures which have the following properties:

Pure Compression and Tension Tensegrities are pin-jointed structures. This means that a tensegrity configuration can be described as a set of members interconnected by ideal ball joints. This allows for optimal use of materials as no shear forces or bending moments act on the members of the structure.

Disconnected Compression Members Compressive members *float* in the tension net. The maximum number of interconnected compressive members defines the class of the structure. In this work, I only consider Class 1 tensegrity structure in which compressive members only connect to tensile members. This partition of tensile and compressive members provides a global level of compliance to Class 1 tensegrities.

Prestress Tensegrities can be prestressed. The stability of a tensegrity structure is a global property, which is the result of a subtle energy balance of all the members. This differs from ubiquitous compression designs which can often be built sequentially by stacking components.

There is some discussion on whether Snelson [158] or Fuller [58] invented the tensegrity concept (see [126], p.221 for a discussion or [163]), but I shall not delve into these historical issues.

Figure 1.1 shows an example of a free-standing tensegrity structure in simulation and hardware. The image on the left clearly shows how bars are suspended in a tension net. The configuration shown is called the tensegrity icosahedron and consists of 6 compressive members held together by 24 tensile elements. This structure has 6 additional actuated tensile members which transform it into a *tensegrity robot*.

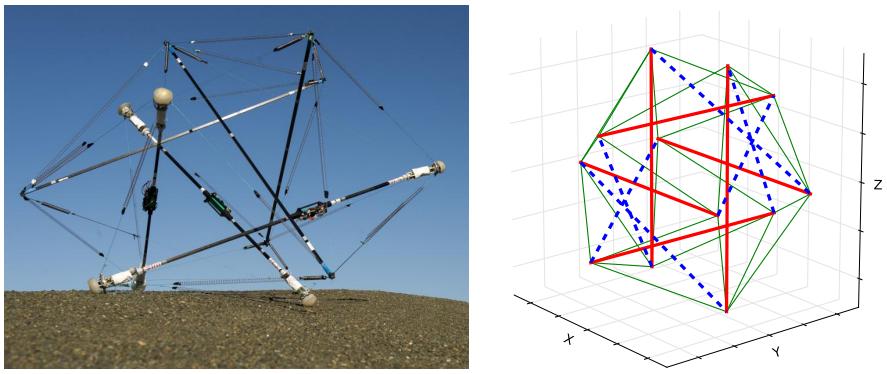


Figure 1.1: The tensegrity robot introduced in Chapter 6 on the left and its simulation model on the right.The tensegrity structures studied in this dissertation consist of fixed length compressive members (bars), passive tensile members (springs in line with a cable) and actuated tensile members (springs in line with an actuated cable). In the illustration on the right, the bars are shown in red (thick full lines), the passive tensile elements in green (thin lines) and the actuated tensile members in blue (thick dashed lines).

1.1.1 Bridging Fields

An intriguing aspect of tensegrity structures is that they appear at several scales. Historically, they were first introduced as an architectural design principle. Later, seminal work by Ingber [84] demonstrated how tensegrities are particularly suited models for the study of the mechanical behavior of cells (Figure 1.2). Tensegrities have also been studied for large space structures (e.g. deployable antennae [172]) and more recently there has been interest in tensegrities as biomechanical models. Finally, the tensegrity robotics field is now emerging.

Such a variety of fields linked by a common principle allows one field to adopt results or methods from another domain. For example, it is possible to use biomechanical measures to study the locomotion properties of a tensegrity robot or to investigate the force distribution of the cytoskeleton by building large scale replicas.

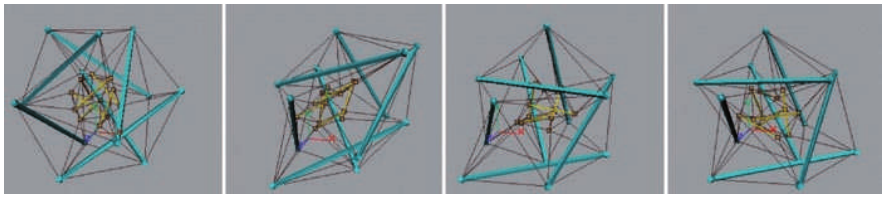


Figure 1.2: Computer simulations of a nucleated tensegrity cell model exhibits mechanical coupling between the cell, the cytoskeleton, and the nucleus. Adapted by permission from Macmillan Publishers Ltd: Nature Reviews Molecular Cell Biology [188], 2009.

1.1.2 Contributions to Robotics

Over the past few decades, there has been a clear shift from stiff robots to platforms that include compliant or flexible elements. The main objectives are to simplify control problems and to increase the safety of robot-human interactions. Often, biological analogies are used to substantiate the choice for flexible elements. Indeed, life on earth appears extremely resilient and robust, while most of the life forms have far less computational capabilities than today's computers. Machines that imitate nature have always been a core objective of scientific research. Currently, it is still not clear if or when we will be able to build a robot with *better* overall autonomous performance than, say, a cat. However, compliant robotics has brought us promising and highly convincing demonstrations of robots with similar or better physical skills than animals or humans for specific situations. Today, compliant robots

begin to appear in the workplace (e.g. Rethink Robotics' Baxter robot) and even space (NASA's Robonaut in the ISS).

Compliant tensegrities are more than merely a type of robot with a large amount of flexible elements. They are highly structured, tunable soft robots with global compliance. It is thus reasonable to say that tensegrities go beyond the now common approach of adding compliance to a stiff mechanism (e.g. a robot arm joint) and instead use compliance and a global distribution of forces as core building blocks.

Investigation of the properties of dynamic compliant tensegrities is opportune. The field of compliant robotics is clearly maturing and this makes it possible to use the lessons learned from this domain and develop fundamentally different design and control approaches for soft structures. It is my hope that the research ideas developed in this dissertation may help establish tensegrity robots as a significant research domain over the next years.

1.2 Research Questions

I will focus on two main research questions in this dissertation:

How can computational methods provide insight into tensegrity robot control problems? This first question is not very detailed on purpose. The goal is to study methods that allow a tensegrity structure with suitable sensor equipment to simplify its proper control problems.

How to design capable tensegrity robots with rich sensor feedback? To be able to validate the methods developed to answer the first research question, it is invaluable to build hardware prototypes. As will be detailed in the relevant chapters, the tensegrity robotics field is not (yet) well established. Therefore, this work should advance the state of the art in tensegrity robotics by presenting results about the design aspects of capable compliant tensegrity robots.

1.3 Related Research & Techniques

The work presented herein touches upon multiple fields and research topics. Presenting every subject here would significantly lengthen this manuscript without adding scientific value. Instead, the topics are introduced throughout this work where relevant, except for a few techniques (Reservoir Computing and tensegrity structures) which merit their proper chapter.

The original tensegrity concept is first and foremost an architectural or mechanical principle. As this thesis is submitted to the Department of Electronics and Information Systems, I provide an extensive overview of the mechanical properties of tensegrities in Chapter 2. In general, the static aspects of tensegrity structures are fairly well studied and a number of review articles and books are available.

To study the computational aspects of tensegrities, I will take inspiration from three main fields: Recurrent Neural Networks (Reservoir Computing in particular), Morphological Computation and Hebbian plasticity. Recurrent Neural Networks and Reservoir Computing have been studied in depth in our lab and Morphological Computation is a broad concept or idea that physical structures and robots can perform computations which can simplify control or sensing problems. Tensegrities provide an ideal setup to study this concept as tensegrities can be used to model physical systems at various scales. However, I take a pragmatic approach to Morphological Computation: When studying computational aspects, I will always focus on solving a task at hand. Hebbian theory has been around for over half a century, but it is still actively studied today. I provide variations on the classic Hebbian plasticity rule to allow for learning in compliant tensegrity robots.

Additionally, I use Central Pattern Generators and Evolutionary Computation at various points. These topics are discussed in Chapters 4 and 7. In recent years, a limited number of control methods and hardware platforms for tensegrity robots have been demonstrated. Chapter 2 reviews tensegrity control methods and Chapter 6 discusses a number of related hardware designs.

1.4 Thesis Outline

This dissertation is structured as follows. I begin with an in-depth overview of tensegrity structures in Chapter 2. In Chapter 3 I provide an introduction to Reservoir Computing. Chapters 4 and 5 present a number of results about the computational aspects of tensegrity structures. I then go on to present the hardware design aspects of two tensegrity robots in Chapters 6 and 7. The former chapter focuses on the design, control and simulation of ReCTeR, a small scale compliant tensegrity robot built in the context of this thesis. The latter discusses tensegrity structures for space exploration applications and the construction of SUPERball, the tensegrity robot platform built at the NASA Ames Research Center. I end with my conclusions and future perspectives in Chapter 8.

2
Tensegrity Structures

The goal of this chapter is to provide a rather general introduction to tensegrity structures. As tensegrities have been studied in various domains, it is imperative to define the types of structural elements and the mathematical and engineering tools used in this work. In addition to this, I also present a number of new results about the statics of tensegrity structures. After reading this chapter, the reader should have an overview of the main tensegrity terminology and research problems.

This chapter is organized as follows. I will first introduce the notation used to describe the connection patterns and spatial arrangement of tensegrities (Section 2.1). After that, I describe stability conditions (Section 2.2) and continue with linear stiffness and vibration analysis (Sections 2.4 & 2.5) of tensegrity structures. The main focus of this thesis are actuated tensegrity structures and I therefore present a short discussion about actuation patterns, which is useful for underactuated systems (Section 2.6). These sections are followed by form-finding methods and a number of new methods for self-stress tuning (Section 2.7). Afterwards, simulation techniques are given for tensegrity structures in Section 2.8. Considerable effort was put into the validation of the simulators using experimental measurements. As hardware platforms for these analyses are presented in a later chapter, the relevant results will be presented there. Finally, I present a literature overview of control strategies for tensegrity robots (Section 2.9).

Some historic information about tensegrity structures was provided in the introduction of this thesis and it will therefore not be discussed here. For a more extensive general introduction to tensegrity kinematics and dynamics, I refer to Skelton's book [157].

2.1 Notation and Terminology

A tensegrity structure consists of a set of compressive members held together by a tensile network. The compressive members are also called bars, rods or struts, while the term spring-cable assembly is used for the tensile elements[1]. In this work the assumption is made that compressive members have fixed lengths, while all tensile members have low stiffness and thus behave like springs. Additionally, all tensile members are considered to be linearly elastic, meaning that the tension on a tensile member is a linear function of the distance between its attachments. Every member connects to an adjacent element at a single point and it is assumed that these contacts behave as ideal ball joints. Alternative terms for the connection between two members are node or end cap. Because tensegrities are pin-jointed structures, all forces act linearly along the members. This allows for efficient use of materials, as the members need not resist shear or bending forces.

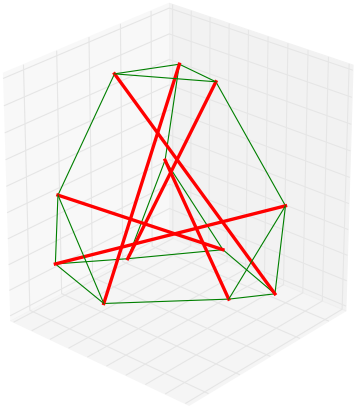


Figure 2.1: A 6-bar Class 1 tensegrity with the minimum (18) number of tensile elements. The structure is a truncated tetrahedron with 4 triangular faces and 4 hexagonal faces. Bars are shown as thick red lines, tensile elements are thin green lines.

In this work, I only consider 3-dimensional Class 1 tensegrities [157], i.e. *pure* tensegrities. This means that no two bars ever share a common node. Hence there are exactly $2b$ nodes with b the number of bars. Calladine provides an extensive discussion on the number of states of prestress and infinitesimal mechanisms by extending Maxwell's rule [20, 118]. The theoretical considerations of these papers are not directly related to the work in this dissertation. Hence, they will not be discussed further. An important remark is that a Class 1 tensegrity with b bars needs at least $3b$ tensile elements, as each node needs to be constrained in 3 dimensions. The tensegrity icosahedron configuration with 24 tensile members and 6 bars, exceeds this requirement and it is in fact possible to construct 6-strut structures with only 18 tensile elements (Figure 2.1).

[1] In a strict sense, a bar is a structural element that can handle both compressive and tensile forces, while a strut is designed for compressive forces. As it is assumed that all compressive elements are always in compression (if not, one could replace it with a tensile member), the terms strut and bars will be used interchangeably.

2.1.1 Elastic Compressive Elements

In a hardware implementation, every bar element will have a finite stiffness coefficient, which can be calculated based on the modulus of elasticity of the material. The two robots discussed in later chapters use carbon fiber and aluminum bars of approximately 1 m and 1.5 m in length respectively. The stiffness coefficient k of these bars can be calculated as:

$$k = \frac{EA}{l_0}, \tag{2.1}$$

where E is the modulus of elasticity (Young's modulus) of the material and A is the cross-sectional area of the element. The aluminum struts used in the hardware setup have a cross-sectional area of $132 \times 10^{-6}\,\mathrm{m}^2$ and the carbon fiber struts have a cross-sectional area of $28 \times 10^{-6}\,\mathrm{m}^2$. Using the Young moduli of aluminum and carbon fiber, one obtains approximate stiffnesses of $k_{alu} = 6 \times 10^6\,\mathrm{N\,m^{-1}}$ and $k_{carbon} = 5.1 \times 10^6\,\mathrm{N\,m^{-1}}$. All tensile elements used in this work have stiffnesses below $1000\,\mathrm{N\,m^{-1}}$ and the fixed length assumption for the compressive members (bars) is thus valid.

As explained in the next section, tensegrities are in static equilibrium when the potential energy of the structure reaches a local minimum. It is easy to show that the rods do not store significant amounts of elastic potential energy and form-finding methods using stiff spring-like compressive members or fixed length rods will thus generate the same results. In a standard icosahedron configuration (shown in Figure 2.4), it can be shown that in equilibrium the compressive member tension equals $\sqrt{6}$ times the tensile member tension. For a common icosahedron configuration with 24 tensile, 6 compressive members, the bar elastic potential energy can be shown to be at least 3000 times lower than the tensile elastic potential energy for the spring constants and prestress levels in this work.

2.1.2 Coordinates

Class 1 structures allow the use of slender elements. These elements are often be cylindrical as there is no preferential direction in the structure. In theory, the angular velocity of any member should remain zero in a pin-jointed structure. Therefore, it is possible to disregard the rotation of the members and use the positions of the nodes or a linear transformation thereof as coordinates. A compressive member can thus be modeled as two point masses with a distance constraint. This description is more elegant and natural than a common rigid body set of coordinates using the position of the center of mass and e.g. Euler angles. Forces act at the nodes, the coordinates of which are directly available. In practice, small amounts of torsion and

bending can occur in a member due to contact forces or offset attachment points. However, even in a hardware implementation the moment of inertia around the longitudinal axis will be at least a few orders of magnitude lower than the moment of inertia around the other axes[2].

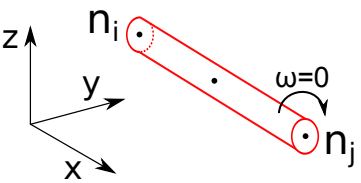


Figure 2.2: Cartesian coordinates for the description of a bar. Structures are simulated by tracking the positions and velocities of the end caps of the compressive elements. Infinitesimally thin (line) elements are modeled and thus zero angular velocity ω is assumed.

For reasons of consistency, Cartesian coordinates (see Figure 2.2) of the nodes are used throughout this thesis to describe the state of a structure. In Skelton's reference work [157] and my prior work [22] a different set of generalized coordinates was used, but the resulting equations are equivalent (up to a linear transform). Skelton's generalized coordinates allow to decouple the rotational and translational equations of motion, while the Cartesian coordinates are more intuitive because the system is directly modeled as a spring-mass net with distance constraints. Note that both these coordinate sets are not minimal. In [22] and [191] it is shown how to replace Skelton's generalized coordinates with a minimal set of generalized coordinates. However, these equations are far less elegant, provide little insight and incur the risk of gimbal lock.

2.1.3 Matrix Description

A number of matrices are commonly used to describe and study tensegrities. This section provides an overview of the most common tensegrity related matrices, their function and properties.

2.1.3.1 Nodal Coordinates Vector and Matrix

The Cartesian coordinates of all nodes are given by:

$$\boldsymbol{N} = \begin{bmatrix} x_1 & y_1 & z_1 \\ & \dots & \\ x_p & y_p & z_p \end{bmatrix} = \begin{bmatrix} \boldsymbol{n}_1^T \\ \boldsymbol{n}_2^T \\ \dots \\ \boldsymbol{n}_p^T \end{bmatrix} = \begin{bmatrix} \boldsymbol{x} \, \boldsymbol{y} \, \boldsymbol{z} \end{bmatrix}. \tag{2.2}$$

[2]For example, consider the extreme case of a solid 1 m rod with a 6 cm radius. The corresponding ratio of the moments of inertia around the longitudinal axis and rotational axes is 185.

It is sometimes convenient to store the nodal coordinates in vector form:

$$\boldsymbol{n} = \begin{bmatrix} x_1 \ \dots \ x_p \, y_1 \ \dots \ y_p \, z_1 \ \dots \ z_p \end{bmatrix}^T . \tag{2.3}$$

2.1.3.2 Connectivity Matrix

The matrix $\boldsymbol{C} \in \{0, 1, -1\}^{s \times p}$ is called a connectivity matrix and contains only ones, zeros and minus ones. Here s is the number of members and p is the number of nodes, which is equal to two times the number of bars[3]. Each row of $\boldsymbol{C}$ contains exactly one -1 and one 1 value, the order of which is arbitrary as all members are symmetric. The positions of the non-zero values indicate connections between nodes (from-to). A connectivity matrix can be split (row-wise) into a spring connectivity matrix and a bar connectivity matrix. Spring connectivity matrices reflect the configurations of tensile members and bar connectivity matrices those of compressive members. A spring connectivity matrix $\boldsymbol{C}_{spring}$ requires $s_{spring} \geq 3\frac{p}{2}$ (each node is connected to at least three springs). Only Class 1 tensegrity structures are considered in this dissertation, which implies that $s_{bar} = \frac{p}{2}$ for a bar connectivity matrix $\boldsymbol{C}_{bar}$.

In graph theory, a connectivity matrix is also called an incidence matrix. This provides some interesting insights. For example, the matrix $\boldsymbol{C}^T\boldsymbol{C}$ is a Laplacian matrix. A Laplacian matrix is always positive semidefinite and its rank deficiency corresponds to the number of connected components (1 in case of a connected structure). Certainly for larger structures, results from graph theory can provide insight into the static properties of a tensegrity. In fact, the Laplacian and in particular the algebraic connectivity are a popular research topic with applications in various fields [35, 51, 124].

2.1.3.3 Nodal Forces Vector and Matrix

The nodal forces matrix is defined as:

$$\boldsymbol{F} = [\boldsymbol{f}_x \, \boldsymbol{f}_y \, \boldsymbol{f}_z] . \tag{2.4}$$

This matrix has the same dimensions as the nodal coordinates matrix $\boldsymbol{N}$, but contains the nodal forces instead of coordinates. Similarly to the nodal coordinates vector, the nodal forces are sometimes used in vector form:

$$\boldsymbol{f} = \begin{bmatrix} f_1^x \ \dots \ f_p^x \, f_1^y \ \dots \ f_p^y \, f_1^z \ \dots \ f_p^z \end{bmatrix}^T . \tag{2.5}$$

[3] These equations can be extended to include fixed nodes. I will not address this here, as free-standing tensegrities are the main focus of this thesis.

2.1.3.4 Force Density Vector and Matrix

A common quantity in tensegrity research is the force density of a member. The force density is defined as the ratio of the tension or force f on a member and its current length l:

$$q = \frac{f}{l}. \tag{2.6}$$

The force densities are used in both vector $\boldsymbol{q}$ and matrix $\boldsymbol{Q} = \mathrm{diag}_v(\boldsymbol{q})$ form[4].

Many equations related to pin-jointed structures and tensegrities in particular, can be written in terms of force densities. If this is the case, one is free to scale the structure by scaling the member forces by the same amount. Additionally, the member stiffness or rest length can be chosen freely.

2.1.3.5 Member Length Vector and Matrix

The elements of the vector $\boldsymbol{l}$ and matrix $\boldsymbol{L} = \mathrm{diag}_v(\boldsymbol{l})$ are the current lengths (distance between adjacent nodes) of the members. For elastic members, $\boldsymbol{l}_0$ and $\boldsymbol{L}_0$ contain their rest lengths, which is the length at which the tension of the member becomes zero.

2.1.3.6 Stiffness Matrix

The stiffness matrix relates small nodal displacements and nodal forces in vector form:

$$\boldsymbol{K}\delta\boldsymbol{n} \approx \delta\boldsymbol{f}. \tag{2.7}$$

As shown later in Section 2.4, it can be computed by taking the partial derivatives of the nodal forces with respect to the nodal coordinates:

$$\boldsymbol{K} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{n}}. \tag{2.8}$$

Note that the stiffness matrix is not constant and depends on the current configuration of the structure as well as the tensions. The variable $\boldsymbol{K}$ on the left-hand side of the above equation should thus be intepreted as the function $\boldsymbol{K}(\boldsymbol{n}, \boldsymbol{q})$.

[4]The diag_v operator transforms a vector into a diagonal matrix.

2.1.3.7 Stress Matrix

The stress matrix is defined as:

$$\boldsymbol{E} = \boldsymbol{C}^T \boldsymbol{Q} \boldsymbol{C}. \tag{2.9}$$

This matrix relates nodal coordinates and nodal forces due to internal forces:

$$\boldsymbol{F} = \boldsymbol{E}\boldsymbol{N}. \tag{2.10}$$

The expanded stress matrix is obtained by:

$$\boldsymbol{G} = \boldsymbol{I_3} \otimes \boldsymbol{E}, \tag{2.11}$$

which defines the equivalent relationship for nodal forces and coordinates in vector form:

$$\boldsymbol{f} = \boldsymbol{G}\boldsymbol{n}. \tag{2.12}$$

Note that the expanded stress matrix is a function of the nodal coordinates and in general does not equal the stiffness matrix (Section 2.4). It can be seen that due to its similarity to the Laplacian matrix $\boldsymbol{C}^T\boldsymbol{C}$, the stress matrix will also have at least 1 zero eigenvalue. In fact, this matrix should have at least 4 zero eigenvalues (12 for the expanded equilibrium matrix), a fact that will be used in the section on form-finding [197].

In the above equations, $\boldsymbol{C}$ is generally constant (the connectivity is fixed) and $\boldsymbol{Q}$ is a non-linear function (Euclidean distances) of the nodal coordinates $\boldsymbol{n}$ and the properties of the tensile elements.

2.1.3.8 Equilibrium Matrix

Eq. 2.12 is a linear function of the nodal coordinates or the force densities, because $\boldsymbol{E}$ is linear in $\boldsymbol{q}$. It is often convenient to decompose this formula in another form:

$$\boldsymbol{A}\boldsymbol{q} = \boldsymbol{f} \tag{2.13}$$

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{C}^T \mathrm{diag}_v(\boldsymbol{C}\boldsymbol{x}) \\ \boldsymbol{C}^T \mathrm{diag}_v(\boldsymbol{C}\boldsymbol{y}) \\ \boldsymbol{C}^T \mathrm{diag}_v(\boldsymbol{C}\boldsymbol{z}) \end{bmatrix}. \tag{2.14}$$

The matrix $\boldsymbol{A}$ is called the equilibrium matrix.

Remark that there is some confusion in the literature about the naming of the equilibrium matrix and stress matrix. In [197] the stress matrix

(according to the definition in this work) is called the equilibrium matrix. Connelly has an equivalent definition for the stress matrix to mine [29], while Tibert uses both the terms stress matrix and force density matrix to refer to the same matrix [171]. Motro uses the same definition for the equilibrium matrix as presented here [126].

2.2 Equilibrium Conditions

The static equilibrium conditions for tensegrity structures and pin-jointed structures have been studied for a long time and various types of stability have been defined, mostly based on properties of the matrices defined in the previous section. I provide a brief overview of the two basic conditions for stability and refer the reader to the extended literature on the subject for additional stability condition. The reason why I limit the scope of this section is that for tensegrity robotics the connectivity and basic shape are most often a driven parameter due to practical constraints.

2.2.1 Balance of Forces

The basic condition for equilibrium is that the net nodal forces balance out:

$$\mathbf{0} = \boldsymbol{G}\boldsymbol{n}, \tag{2.15}$$

or equivalently:

$$\mathbf{0} = \boldsymbol{A}\boldsymbol{q}. \tag{2.16}$$

For a valid tensegrity structure the constraint $\boldsymbol{q}_{spring} > \mathbf{0}$ ensures that springs only carry tension loads. These conditions are trivially extended to include external loads and gravity.

The quadratic form $\boldsymbol{q}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{q} = 0$ of the last equilibrium equation is often convenient to write form-finding or self-stress problems as an optimization problem. For non-degenerate configurations, the fact that $\boldsymbol{C}$ is full rank ensures that the matrix $\boldsymbol{A}^T\boldsymbol{A}$ is non-singular (assuming no parallel springs).

2.2.2 Stability of an Equilibrium

A force balance only means that for a particular shape and set of force densities, the structure does not experience any accelerations. Analogous to an inverted pendulum, which has an unstable equilibrium in the upright position, it is important to investigate the stability of such an equilibrium.

The standard way of doing this is by verifying that the potential energy of the structure is at a local minimum. This ensures that (small) deviations from the equilibrium state will generate restoring force in the direction of the equilibrium. As tensegrities can have multiple equilibrium states, it is only feasible to require a local potential energy minimum.

By definition, the stiffness matrix provides a relationship between small nodal displacements and the nodal forces generated by them. This allows for the common approximation of the change in potential energy due to a nodal displacement:

$$u(\boldsymbol{N} + \delta\boldsymbol{N}) - u(\boldsymbol{N}) \approx \frac{1}{2}\delta\boldsymbol{n}^T \boldsymbol{K} \delta\boldsymbol{n}. \tag{2.17}$$

For a minimum of potential energy, the requirement is $u(\boldsymbol{N}+\delta\boldsymbol{N})-u(\boldsymbol{N}) > 0$ for all $\delta\boldsymbol{N}$ excluding rigid body modes (translations & rotations). This simply means that the non-rigid body mode eigenvalues of the stiffness matrix (which is thus the Hessian of the potential energy) need to be strictly positive.

In the inverted pendulum example, the Hessian has a negative eigenvalue when the pendulum is in the upright position. A similar situation is uncommon for practical tensegrity structures. Instead, zero eigenvalues of the stiffness matrix can occur. These degenerate equilibria correspond to zero stiffness mechanisms. This means that there are (first order) flexes of the structure which do not alter the potential energy of the system. In practice, it can be useful to intentionally create such a mechanism to facilitate active folding of a structure. In this context, Section 2.7.6.2 presents a technique to tune the eigenvalues of the stiffness matrix and Section 2.4.1 provides a situation in which counterintuitive flexes can occur.

Tensegrities with multiple equilibrium states (bistable and multistable solutions) are beyond the scope of this dissertation, but could show potential for tensegrity locomotion by switching between equilibrium states. For example, a tensegrity prism can be pushed inside-out, which results in a mirrored stable configuration. Similarly, there are examples of stacked structures arranged in a torus shape, which can be twisted. This last example could be advantageous for pipe climbing tensegrity robots.

2.2.3 Extended Equilibrium Conditions

The previous paragraphs have provided only very basic equilibrium and stability conditions. However, various other stability criteria are known for tensegrities and pin-jointed structures in general [189]. For example, Connelly [29, 198] introduces the super stability condition; a stronger equilibrium condition which depends only on the properties of the stress matrix.

As equilibrium conditions and form-finding are closely related, the extensive form-finding literature provides various alternative formulations of the equilibrium conditions [171]. Note that an equilibrium condition does in general not ensure stability of a tensegrity structure. Tensegrities in equilibrium can be unstable (e.g. due to a mechanism).

2.3 Actuation and Sensing: Spring-Cable Assemblies

Most experiments in this thesis involve actuated structures with embedded sensors. In this section, I define the setup and terminology used for this.

Actuation and sensing only involve the tensile members of a structure. The hardware robots developed in the context of this thesis have additional sensory equipment embedded in the bars, the properties of which will be discussed in the relevant hardware chapters.

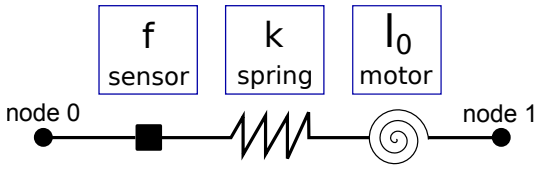


Figure 2.3: Conceptual model of a spring-cable assembly. All tensile members in this thesis have passive compliance. Actuated members have a series motor that changes the length of a cable inline with a spring, thus changing the effective rest length of the assemby. Inline force transducers are used to sense the spring/cable tension.

A tensile member is realized by a *spring-cable assembly*. Spring-cable assemblies have various possible physical realizations, all of which implement the conceptual model presented in Figure 2.3. The model consists of two nodes with 3 series elements in between them. From left to right, one first sees the force transducer to measure the tension f on the assembly. A linear spring with spring constant k sits in the middle of the assembly. In the model, the linear spring has zero equilibrium and acts as a pure linear tension element. The last element is an actuated cable. An actuator changes the length of a cable, which modifies the effective rest length of the spring-cable assembly.

The tension on the spring-cable assembly is given by:

$$f = k \max(l - l_0, 0) \tag{2.18}$$
$$= k \max(\|\boldsymbol{n}_1 - \boldsymbol{n}_0\| - l_0, 0), \tag{2.19}$$

where $\boldsymbol{n}_1$ and $\boldsymbol{n}_0$ are the positions of the two nodes.

The force density of a spring-cable assembly in tension is thus given by:

$$q = \frac{f}{l} \tag{2.20}$$
$$= k\left(1 - \frac{l_0}{l}\right). \tag{2.21}$$

2.4 Stiffness Matrix

In this section, I present a linear static analysis of tensegrity structures. This provides insight into stiffness and oscillatory properties, as well as the differences between actuation patterns. Most of the equations presented here can be derived using linear dynamical systems and finite element theory.

The derivation of the stiffness matrix is based on the descriptions in [64, 152, 197]. To simplify the mathematical formulation, I assume linear elastic tensile and compressive members. Each bar is modeled as two point masses and tensile members attach at these points. This disregards twisting of the bars, which in practice only minimally influences the behavior of a tensegrity robot.

More precise models are necessary to obtain accurate estimates of the behavior during impact. However, these effects are beyond the scope of this work. Pure axial loading is a basic assumption for tensegrity structures, which unfortunately cannot be guaranteed during impact. Impact forces take time to propagate through the tensile and compressive network. As such, bending moments can exist during the brief period that the structure needs to rebalance itself. To the best of my knowledge there has been no significant research in this direction, although the University of Idaho has started performing drop tests to investigate these effects ([1] and personal communication).

The stiffness matrix $\boldsymbol{K}$ is obtained by partial differentiation of the nodal forces with respect to the nodal coordinates $\boldsymbol{N}$. For the elements $\boldsymbol{x}$ of $\boldsymbol{N}$

this gives:

$$\frac{\partial \boldsymbol{f}_x}{\partial \boldsymbol{x}} = \left[\frac{\partial \boldsymbol{E}}{\partial x_1}\boldsymbol{x}\ \frac{\partial \boldsymbol{E}}{\partial x_2}\boldsymbol{x}\ \dots\ \frac{\partial \boldsymbol{E}}{\partial x_p}\boldsymbol{x}\right] + \boldsymbol{E} \tag{2.22}$$

$$\frac{\partial \boldsymbol{f}_y}{\partial \boldsymbol{x}} = \left[\frac{\partial \boldsymbol{E}}{\partial x_1}\boldsymbol{y}\ \frac{\partial \boldsymbol{E}}{\partial x_2}\boldsymbol{y}\ \dots\ \frac{\partial \boldsymbol{E}}{\partial x_p}\boldsymbol{y}\right] \tag{2.23}$$

$$\frac{\partial \boldsymbol{f}_z}{\partial \boldsymbol{x}} = \left[\frac{\partial \boldsymbol{E}}{\partial x_1}\boldsymbol{z}\ \frac{\partial \boldsymbol{E}}{\partial x_2}\boldsymbol{z}\ \dots\ \frac{\partial \boldsymbol{E}}{\partial x_p}\boldsymbol{z}\right]. \tag{2.24}$$

Using the definition of the stress matrix $\boldsymbol{E}$, one obtains:

$$\frac{\partial \boldsymbol{E}}{\partial x_i} = \boldsymbol{C}^T \frac{\partial \boldsymbol{Q}}{\partial x_i}\boldsymbol{C} \tag{2.25}$$

$$\frac{\partial \boldsymbol{Q}}{\partial x_i} = \boldsymbol{L}_0 \boldsymbol{S} \boldsymbol{L}^{-2} \frac{\partial \boldsymbol{L}}{\partial x_i}. \tag{2.26}$$

The diagonal matrix $\boldsymbol{S}$ contains the spring constants k of the members. Note that I use spring constants k instead of Young's modulus (modulus of elasticity) which is common in mechanical engineering and materials science. The reason for this is that for the robots developed in this thesis, the elastic members are tension and compression springs of low axial stiffness. The main characteristic of these elastic elements is the spring constant k as their other properties (e.g. Poisson's ratio) are of less importance in most robotics applications.

Now compute the nodal coordinate differences as:

$$[\boldsymbol{u}\,\boldsymbol{v}\,\boldsymbol{w}] = \boldsymbol{C}\boldsymbol{N}. \tag{2.27}$$

The diagonal matrices $\boldsymbol{U}$, $\boldsymbol{V}$ and $\boldsymbol{W}$ contain the values of $\boldsymbol{u}$, $\boldsymbol{v}$ and $\boldsymbol{w}$ as non-zero elements respectively. Note that the matrix $\boldsymbol{W}$ represents a set of coordinate differences here, whereas $\boldsymbol{W}$ refers to a weight matrix in the remainder of this thesis.

Noting that $\boldsymbol{L}^2 = \boldsymbol{U}^2 + \boldsymbol{V}^2 + \boldsymbol{W}^2$ allows to compute the derivatives of member lengths with respect to the nodal coordinates:

$$\frac{\partial \boldsymbol{L}}{\partial x_i} = \boldsymbol{L}^{-1}\left[\boldsymbol{U}\frac{\partial \boldsymbol{U}}{\partial x_i}\ \boldsymbol{V}\frac{\partial \boldsymbol{V}}{\partial x_i}\ \boldsymbol{W}\frac{\partial \boldsymbol{W}}{\partial x_i}\right], \tag{2.28}$$

with

$$\frac{\partial \boldsymbol{U}}{\partial x_i} = \boldsymbol{C}_i \qquad \frac{\partial \boldsymbol{V}}{\partial x_i} = \boldsymbol{0} \qquad \frac{\partial \boldsymbol{W}}{\partial x_i} = \boldsymbol{0}. \tag{2.29}$$

Here $\boldsymbol{C}_i$ is a diagonal matrix containing column i of $\boldsymbol{C}$ as non-zero elements. Similarly, $\boldsymbol{c}_i$ is a vector containing the same values. Using these definitions,

one obtains:

$$\frac{\partial \boldsymbol{E}}{\partial x_i}\boldsymbol{x} = \boldsymbol{C}^T\boldsymbol{L}_0\boldsymbol{S}\boldsymbol{L}^{-3}\boldsymbol{U}^2\boldsymbol{c}_i, \tag{2.30}$$

where slack spring-cable assemblies are discarded.

Now introduce the variables $\boldsymbol{D}_x = \boldsymbol{C}^T\boldsymbol{U}\boldsymbol{L}^{-1}$, $\boldsymbol{D}_y = \boldsymbol{C}^T\boldsymbol{V}\boldsymbol{L}^{-1}$ and $\boldsymbol{D}_z = \boldsymbol{C}^T\boldsymbol{W}\boldsymbol{L}^{-1}$. The derivatives of the nodal forces with respect to the nodal coordinates can then be written as (again shown for the elements $\boldsymbol{x}$ of $\boldsymbol{N}$):

$$\frac{\partial \boldsymbol{f}_x}{\partial \boldsymbol{x}} = \boldsymbol{D}_x\boldsymbol{L}_0\boldsymbol{S}\boldsymbol{L}^{-1}\boldsymbol{D}_x^T + \boldsymbol{E} \tag{2.31}$$

$$\frac{\partial \boldsymbol{f}_y}{\partial \boldsymbol{x}} = \boldsymbol{D}_y\boldsymbol{L}_0\boldsymbol{S}\boldsymbol{L}^{-1}\boldsymbol{D}_x^T \tag{2.32}$$

$$\frac{\partial \boldsymbol{f}_z}{\partial \boldsymbol{x}} = \boldsymbol{D}_z\boldsymbol{L}_0\boldsymbol{S}\boldsymbol{L}^{-1}\boldsymbol{D}_x^T. \tag{2.33}$$

The complete stiffness matrix is therefore given by:

$$\boldsymbol{K} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{n}} \tag{2.34}$$

$$= \begin{bmatrix} \frac{\partial \boldsymbol{f}_x}{\partial \boldsymbol{x}} & \frac{\partial \boldsymbol{f}_x}{\partial \boldsymbol{y}} & \frac{\partial \boldsymbol{f}_x}{\partial \boldsymbol{z}} \\ \frac{\partial \boldsymbol{f}_y}{\partial \boldsymbol{x}} & \frac{\partial \boldsymbol{f}_y}{\partial \boldsymbol{y}} & \frac{\partial \boldsymbol{f}_y}{\partial \boldsymbol{z}} \\ \frac{\partial \boldsymbol{f}_z}{\partial \boldsymbol{x}} & \frac{\partial \boldsymbol{f}_z}{\partial \boldsymbol{y}} & \frac{\partial \boldsymbol{f}_z}{\partial \boldsymbol{z}} \end{bmatrix} \tag{2.35}$$

$$= \boldsymbol{K}_e + \boldsymbol{K}_g \tag{2.36}$$

$$\boldsymbol{K}_e = \boldsymbol{D}\boldsymbol{L}_0\boldsymbol{S}\boldsymbol{L}^{-1}\boldsymbol{D}^T \tag{2.37}$$

$$= \boldsymbol{D}\left(\boldsymbol{S} - \boldsymbol{Q}\right)\boldsymbol{D}^T \tag{2.38}$$

$$= \boldsymbol{A}\boldsymbol{L}^{-2}\left(\boldsymbol{S} - \boldsymbol{Q}\right)\boldsymbol{A}^T \tag{2.39}$$

$$\boldsymbol{K}_g = \boldsymbol{I} \otimes \boldsymbol{E} = \boldsymbol{G}, \tag{2.40}$$

where I used $\boldsymbol{D}^T = \begin{bmatrix}\boldsymbol{D}_x^T\, \boldsymbol{D}_y^T\, \boldsymbol{D}_z^T\end{bmatrix}$. The form $\boldsymbol{K} = \boldsymbol{A}\boldsymbol{L}^{-2}\left(\boldsymbol{S} - \boldsymbol{Q}\right)\boldsymbol{A}^T + \boldsymbol{G}$ allows to write the stiffness matrix purely in terms of nodal coordinates, member stiffness and force densities. This is often useful, as the force density is a more natural quantity than the rest length for the calculation of the stiffness of the compressive members. The eigenvectors of $\boldsymbol{K}$ are the stiffness modes of the structure. In a free-standing structure, the first 6 modes are rigid-body modes with zero eigenvalues. Figure 2.4 shows an example of a deformation of a tensegrity icosahedron along one of its stiffness modes.

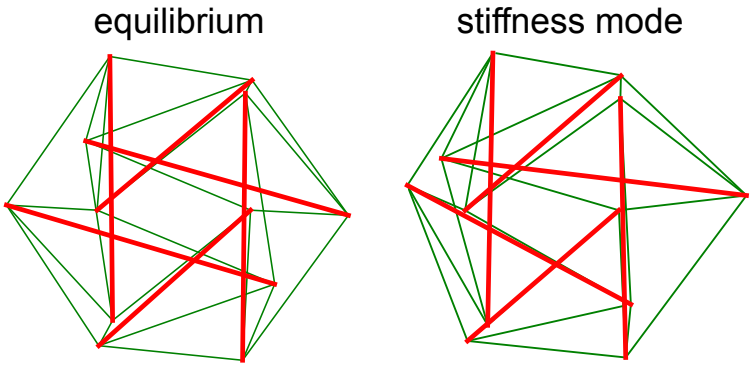


Figure 2.4: Linear static analysis. The common tensegrity icosahedron in equilibrium is shown on the left. The right plot shows a displacement along a stiffness mode of the structure (eigenvector of the stiffness matrix $\boldsymbol{K}$).

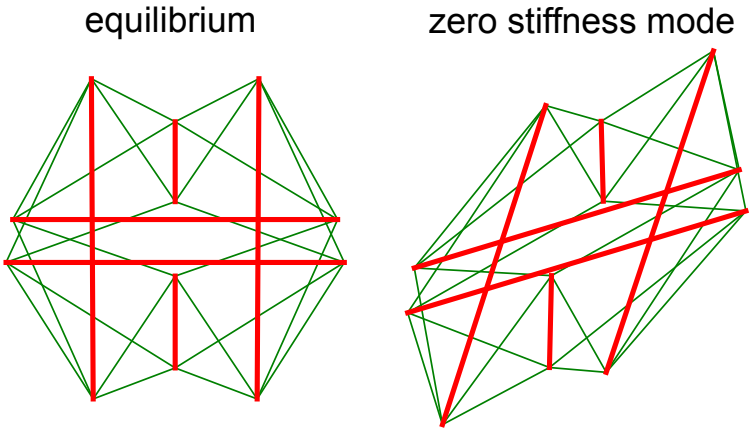


Figure 2.5: A zero stiffness mechanism (shearing) in a tensegrity icosahedron with zero rest length spring-cable assemblies. The length of the bars does not change for finite displacements along this mode and as affine transforms also satisfy the equilibrium conditions, the mechanism is finite.

2.4.1 The Stiffness of Tensegrities

Remark that zero rest length spring-cable assemblies ($l_0 = 0$ or equivalently $q = s$), can reduce the rank of $\boldsymbol{K}_e$. If a displacement is in the null space of $\boldsymbol{K}_e$, only the stress matrix defines the stiffness of the structure with respect to this displacement. As I stated earlier, $\boldsymbol{G}$ will have a nullity of at least 12 for a non-degenerate 3-dimensional tensegrity [197]. This implies that zero stiffness mechanisms can exist in structures with zero rest length spring-cable assemblies.

Figure 2.6: Anglepoise lamp: An example of a balanced arm design with zero stiffness mechanisms with tensioned springs. This mechanism was first patented by George Carwardine in the 1930s [25].

This might be seen as counterintuitive and a disadvantage of rest length actuation for tensegrities at first, as an increase of the pretension and elastic potential energy of the structure can reduce its stiffness. The effective stiffness of individual spring-cable assemblies increases, but the stiffness of some modes drops. However, it potentially allows for interesting control algorithms in the sense that one can create a mechanism by tensioning a cable and stiffen the mechanism by releasing a cable. This can be safer than quickly tensioning a loose cable to stiffen a mechanism.

Zero stiffness tensegrities were studied by Schenk [152]. Schenk showed that zero stiffness mechanisms are not restricted to infinitesimal mechanisms an can thus be moved over finite distances. More precisely, if an affine transform keeps the lengths of the bars constant and the structure has zero rest length spring-cable assemblies, the zero stiffness mechanisms are finite. Figure 2.5 shows an example of such a finite mechanism (shearing) in a tensegrity icosahedron when the rest length of the spring-cable assemblies are reduced to zero.

A similar effect is used in balanced-arm designs, such as the Anglepoise Lamp (Figure 2.6) [56]. In these designs, the structure's gravitational forces are balanced by springs. Typically only a small amount of work (mainly due to friction) is required to change the shape of such designs, as the mechanisms themselves have almost no stiffness, while the springs are in fact tensioned or compressed.

Figure 2.7 shows the eigenvalues of the stiffness matrix of a tensegrity

icosahedron (24 spring-cable assemblies) with 1 m struts and uniform spring constants and rest lengths as a function of the pretension and spring constants. The 6 rigid body modes and the 6 modes corresponding to changing strut lengths were discarded and the remaining 24 modes were averaged per 3 reflecting the symmetry of the structure.

Intuitively one expects a monotonous increase in modal stiffness for increasing spring constants and pretension. For the spring constants, this is indeed the case and the most straightforward way to increase the stiffness of a tensegrity structure is to increase the spring constants.

On the contrary, the pretension tells a more interesting story. While stiffer modes indeed become even stiffer at higher pretensions, the opposite effect is shown for the lowest stiffness mode. At high pretension, when the rest length nears zero, a zero stiffness mechanism is created.

2.5 Modal Analysis

The eigenvectors of the local stiffness matrix $\boldsymbol{K}$ provide information about how a structure deforms under loading, by describing the first order relationship between nodal forces and nodal displacements. The linear dynamic behavior can be modeled by adding inertial forces to the equation:

$$\boldsymbol{M}\ddot{\boldsymbol{n}}(t) + \boldsymbol{R}\dot{\boldsymbol{n}}(t) + \boldsymbol{K}\boldsymbol{n}(t) = \boldsymbol{f}_{ext}(t), \tag{2.41}$$

where $\boldsymbol{f}_{ext}(t)$ are external nodal forces, $\boldsymbol{M}$ is the mass matrix of the structure and $\boldsymbol{R}$ is a damping matrix. The vibration modes decouple for undamped systems ($\boldsymbol{R} = \boldsymbol{0}$). As the compliant Class 1 tensegrities considered in this work have only small amounts of friction due to the lack of joints, the damping matrix is not considered.

The normal modes of the structure can be found by solving the generalized eigenvalue problem for the eigenvectors $\boldsymbol{\phi}_i$ and the eigenvalues ω_i [175]:

$$\boldsymbol{K}\boldsymbol{\phi} = \omega^2 \boldsymbol{M}\boldsymbol{\phi}. \tag{2.42}$$

The eigenvalues ω provide the oscillation frequency of each normal mode.

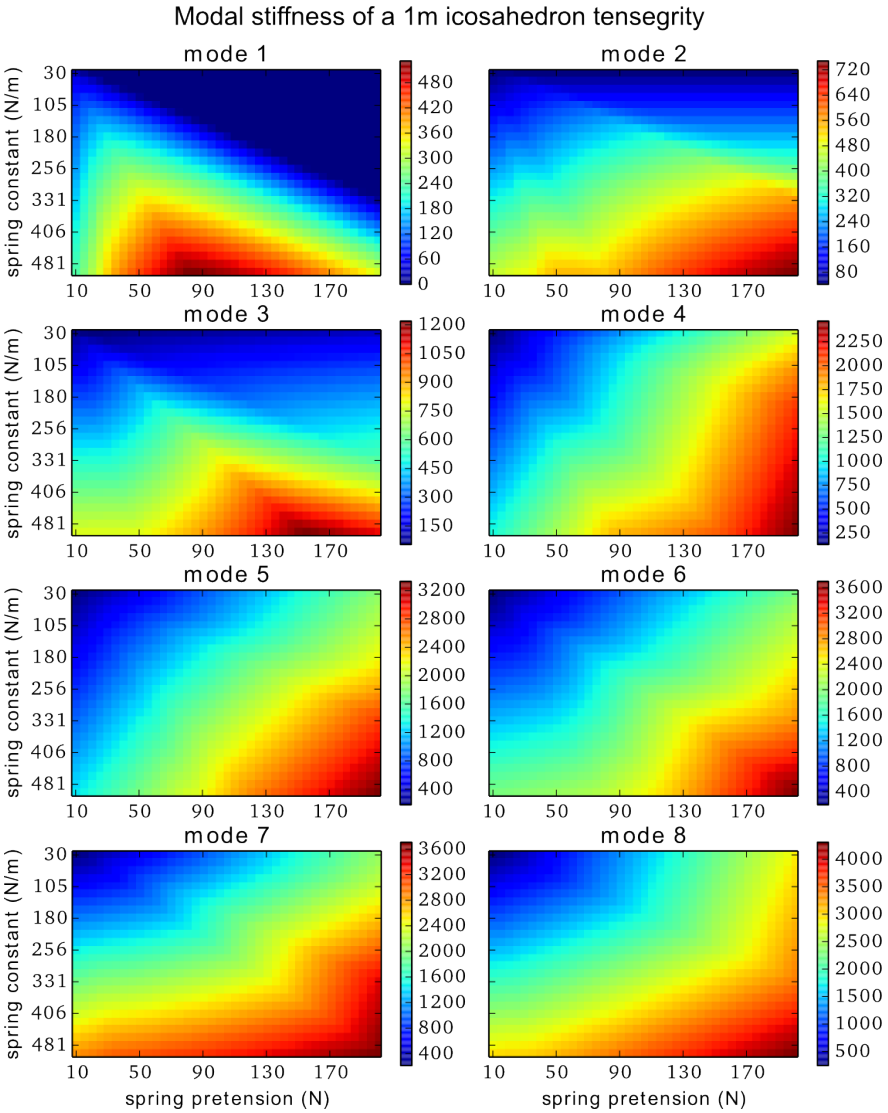


Figure 2.7: Eigenvalues ($\mathrm{N\,m^{-1}}$) corresponding to the modes of the stiffness matrix $\boldsymbol{K}$ of a tensegrity icosahedron (24 spring-cable assemblies) with $1\,\mathrm{m}$ struts and uniform spring constants and rest lengths as a function of the pretension and spring constants. The 6 rigid body modes and the 6 modes corresponding to changing strut lengths were discarded and the remaining 24 modes were averaged per 3 reflecting the symmetry of the structure. Counterintuitively, the stiffness of the lowest mode drops at high pretension (zero rest length).

2.6 Actuation Patterns

In this section, I analyze the behavior of actuated spring-cable assemblies. In particular, one goal is to find which connections are advantageous in an underactuated structure. Using the first order approximation $\boldsymbol{K}\delta\boldsymbol{n} \approx \delta\boldsymbol{f}$, the nodal displacements due to a force applied between any two nodes of the structure can be estimated. Because this is a linear relationship, it is sufficient to study individual node pairs. Additionally, the simulator presented in Section 2.8 is used to compare the non-linear and linear kinematics. This provides useful insights for later chapters in which tensegrities are seen as computational devices. In that context, highly non-linear regimes are often sought after.

The symmetric tensegrity icosahedron with uniform spring constants is used as an example.The reason for this is that this structure is the basis of the two physical realizations built in the context of this work. Figure 2.8 presents the results of various actuation patterns for this structure. It becomes clear that the pattern (0-10) (top right plot of Figure 2.8) which connects perpendicular struts by spring-cable assemblies running through the robot is particularly advantageous for a number of reasons. First, it significantly excites all stiffness modes of the structure, has a large range of motion and causes maximal deformation of the robot. Secondly, an actuator using this pattern does not need to oppose the pretension of the structure and the structure can be powered down without collapsing the robot. Finally, even the kinematics due to this pattern are highly non-linear, which renders the control of a tensegrity robot an interesting and non-trivial problem.

2.7 Form-Finding & Tuning

Form-finding provides answers to the problem of finding valid sets of nodal coordinates $\boldsymbol{N}$ and force densities $\boldsymbol{q}$ for structures in static equilibrium. Most of the literature on this topic is review-like or concerned with a specific new development on tensegrity form-finding. In contrast, this section is structured around specific questions that might arise when one aims to design or optimize a tensegrity structure. The goal is to provide a good solution for each situation, while not aiming at a complete overview of the literature on the topic. In this context, new approaches for shape optimization and for prestress tuning for structures with multiple states of self-stress are presented.

I do not address the question of finding an optimal connection pattern from scratch and instead I assume the connectivity matrix $\boldsymbol{C}$ to be known.

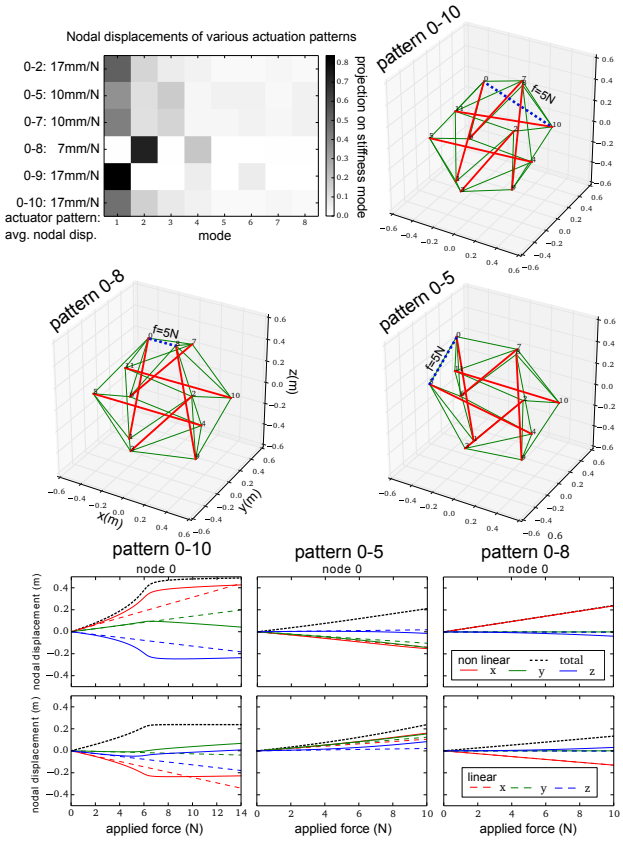


Figure 2.8: Evaluation of actuation patterns for a uniform tensegrity icosahedron with spring-cable assemblies with $30\,\mathrm{N\,m^{-1}}$ springs and a rest length of $0.2\,\mathrm{m}$. The length of the struts is $1\,\mathrm{m}$. The bottom 6 plots show the displacement of 2 nodes as a function of the force applied between various pairs of nodes $(0\text{-}10)$,$(0\text{-}5)$ and $(0\text{-}8)$. The thin dashed lines are the displacements predicted by the linear kinematic model $\boldsymbol{K}\delta\boldsymbol{n} \approx \delta\boldsymbol{f}$, while the full lines show the results of the non-linear model presented in Section 2.8. The top right and center plots show the deformation (based on the non-linear model) of the structure when $5\,\mathrm{N}$ of force is applied between the various actuation patterns. The top left figure presents how different patterns deform the structure along its stiffness mode based on the linear stiffness model. For each pattern, the projections on the stiffness modes are normalized. The rigid body and bar extension modes are discarded and the eigenvectors are averaged per 3 (highly similar due to symmetry). Additionally, the average nodal displacement is shown $(\mathrm{mm\,N^{-1}})$. These results show that the $(0\text{-}10)$ pattern (and its symmetries) allows for large deformations at low tensions with significant non-linearities for even a single actuator. Note that pattern $(0\text{-}5)$ works against the pretension of the structure.

The argument to omit this is that it is almost never an issue in practice as there is a vast set of known tensegrity patterns. For example, Connelly and Terrell [30] provide a catalog of highly symmetric structures at `http://www.math.cornell.edu/~tens/`.

According to my personal experience with designing irregular structures, an efficient method is to add redundancy (additional members) to existing structures and to remove original members by trial-and-error or by following a gradient descent approach to reduce the force on a member to zero. The advantage of this approach is that the resulting irregular structures tend to inherit some of the good properties of the original pattern and shape. In particular, the structures tend to be *open* — meaning that compressive members are spaced far apart — which is important for robotics applications as bar-to-bar collisions must be avoided. This method was used to design the irregular or random structures for some of the experiments in Chapters 4 and 5.

Similarly, Section 2.7.5 optimizes irregular redundant structures which were created by adding a large amount of tensile members to a minimal configuration. For alternative methods, I refer to Rieffel who provides a generative method for irregular structures [143] and Paul who developed a form-finding method using evolutionary algorithms, which is also capable of finding irregular structures [135].

Gravitational and external forces are not considered in the next sections. These types of forces typically enter the equations as linear constraints and as they are of lower or equal order than the other constraints, they can be added easily to the equations presented next.

2.7.1 Unknown Nodal Coordinates and Unknown Force Densities

This is the standard form-finding problem often studied in the literature. Therefore, I will only briefly discuss the issue and instead refer the reader to one of the multiple review articles on the topic [72, 171].

A large number of the form-finding methods for tensegrities are extensions of the force-density method presented in the 1970s by Schek [151]. The original force-density method applies to tension nets, for which the stress matrix is generally positive definite. The typical modus operandi of the extensions of this method to tensegrity structures is to alternate between optimizing the eigenvalues of the stress matrix and finding a valid set of force densities [176, 197].

Relaxation methods are an interesting alternative to the force-density approaches [107, 171]. These methods involve dynamic or kinematic relax-

ation of the forces in a structure to find a stable shape. The advantage is that complex behavior of elements (e.g. cables that become slack, bars that buckle) are usually easier to take into account than with force-density methods.

2.7.2 Unknown or Partially Known Nodal Coordinates and Known Force Densities

This particular problem can occur when a form-finding problem is solved in terms of force densities. If the force densities are known, the equilibrium condition $\boldsymbol{Gn} = \boldsymbol{0}$ can be used to find a suitable set of nodal coordinates. The expanded stress matrix of a non-degenerate three-dimensional tensegrity structure has at least 12 zero eigenvalues and is thus singular. Therefore, the problem has infinitely many solutions and additional information is needed to find an optimal shape. One possibility is to fix the position of 4 nodes. Another interesting choice is to maximize the volume of the structure or minimum inter strut distance.

2.7.3 Known Nodal Coordinates, Unknown Force Densities

The equilibrium condition $\boldsymbol{Aq} = \boldsymbol{0}$, combined with the positive force density constraint for the tensile members, can be used in this case. Section 2.7.5 provides solutions in case the system is underdetermined. If the system is not underdetermined, a simple practical approach is to solve the constrained optimization problem $\text{argmin}_{\boldsymbol{q}}\, \boldsymbol{q}^T \boldsymbol{A}^T \boldsymbol{Aq}$ subject to the constraints $q_i > \rho$ for some lower limit ρ on the force densities of the spring-cable assemblies. If the result satisfies the equilibrium condition, then a valid solution has been found. Otherwise, the nodal coordinates are not a valid equilibrium shape.

2.7.4 Shape Optimization with Respect to a Target Shape

I now consider the problem of finding a set of nodal coordinates and force densities which put a tensegrity structure in a static equilibrium close to a target set of nodal coordinates $\boldsymbol{n}_{target}$. A possible application of this technique is the replacement of a rigid body part of a robot with a tensegrity structure of similar shape (e.g. designing a tensegrity spine for a quadruped, Figure 2.9).

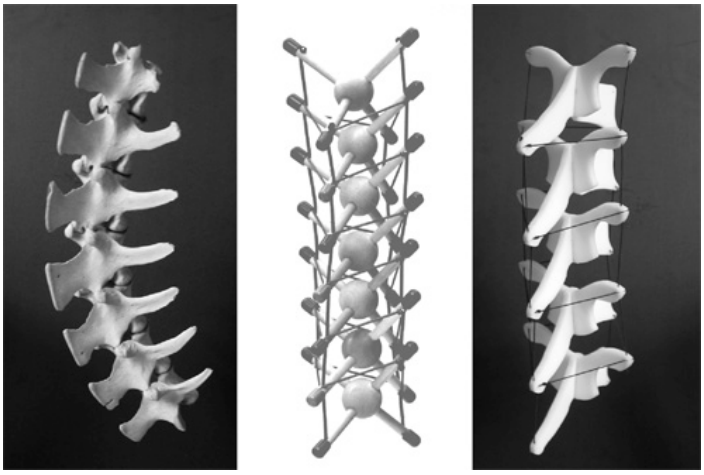


Figure 2.9: Spine and corresponding tensegrity models showing how vertebrae can float without touching. Image courtesy of Tom Flemons (copyright 2006) `http://www.intensiondesigns.com`

A gradient descent approach is used to solve this problem. This method has the added benefit that one can build a structure that morphs from an initial shape into the final one. I assume an initial set of valid force densities to be given, a problem which was addressed in Section 2.7.1.

The idea behind the approach is to alternate between the optimization of the force densities $\boldsymbol{q}$ and the nodal coordinates $\boldsymbol{n}$. However, the static equilibrium condition $\boldsymbol{Gn} = \boldsymbol{0}$ needs to be fulfilled at all times. An alternative method to the gradient approach developed here is presented by Micheletti [121].

Therefore, it is appealing to define a loss function only in terms of the force densities:

$$L(\boldsymbol{q}) = \min_{\boldsymbol{n}} \|\boldsymbol{n} - \boldsymbol{n}_{target}\| \tag{2.43}$$

subject to

$$\boldsymbol{G_q n} = \boldsymbol{0}, \tag{2.44}$$

where $\boldsymbol{G_q}$ is the expanded stress matrix for $\boldsymbol{q}$. The optimization of the nodal coordinates for fixed force densities turns out to have a closed form solution. This makes it possible to compute the gradient of $\boldsymbol{q}$ through the optimal solution of $\boldsymbol{n}$.

To show this, the minimization of $\|\boldsymbol{n} - \boldsymbol{n}_{target}\|$ is first rewritten in the

standard form for quadratic programming:

$$\text{argmin}_{\tilde{\mathbf{n}}} \tilde{\mathbf{n}}^T \tilde{\mathbf{n}} \tag{2.45}$$

with

$$\tilde{\mathbf{n}} = \boldsymbol{n} - \boldsymbol{n}_{target} \tag{2.46}$$

subject to

$$\boldsymbol{G_q} \tilde{\mathbf{n}} = -\boldsymbol{G_q} \boldsymbol{n}_{target}. \tag{2.47}$$

It is well known that the solution to this problem can be found by solving the following set of equations:

$$\begin{bmatrix} \boldsymbol{I} & \boldsymbol{G_q^T} \\ \boldsymbol{G_q} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{n}} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} \\ -\boldsymbol{G_q} \boldsymbol{n}_{target} \end{bmatrix}, \tag{2.48}$$

where $\boldsymbol{\lambda}$ are Lagrange multipliers. As the goal is to compute the partial derivatives of $\tilde{\mathbf{n}}$ with respect to $\boldsymbol{q}$, the derivatives need to be computed through the solution of this equation. The matrix on the left is singular, so one could use the pseudo-inverse of this matrix and compute the derivatives through the pseudo-inverse [61].

However, the unregularized pseudo-inverse tends to be unstable in this case and the resulting derivatives will be as well. I therefore developed a solution based on the regularized pseudo-inverse. First, the regularized pseudo-inverse solution is computed as:

$$\boldsymbol{\Omega} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{G_q^T} \\ \boldsymbol{G_q} & \boldsymbol{0} \end{bmatrix} \tag{2.49}$$

$$\begin{bmatrix} \tilde{\mathbf{n}} \\ \boldsymbol{\lambda} \end{bmatrix} = \left[\boldsymbol{\Omega}^T \boldsymbol{\Omega} + r\boldsymbol{I}\right]^{-1} \boldsymbol{\Omega}^T \begin{bmatrix} \boldsymbol{0} \\ -\boldsymbol{G_q} \boldsymbol{n}_{target} \end{bmatrix}. \tag{2.50}$$

Note that $\boldsymbol{\Omega} = \boldsymbol{\Omega}^T$ and thus $\left[\boldsymbol{\Omega}^T \boldsymbol{\Omega} + r\boldsymbol{I}\right]^{-1} = \left[\boldsymbol{\Omega}^2 + r\boldsymbol{I}\right]^{-1}$. Writing out this last equation in full gives:

$$\left[\boldsymbol{\Omega}^2 + r\boldsymbol{I}\right]^{-1} = \begin{bmatrix} \boldsymbol{I}(1+r) + \boldsymbol{G_q^2} & \boldsymbol{G_q} \\ \boldsymbol{G_q} & \boldsymbol{G_q^2} + \boldsymbol{I}r \end{bmatrix}^{-1}. \tag{2.51}$$

For non-zero regularization terms r, this last matrix is block invertible. This

is a useful fact, because:

$$\boldsymbol{\Omega}^T \begin{bmatrix} \mathbf{0} \\ -\boldsymbol{G_q} \boldsymbol{n}_{target} \end{bmatrix} = \begin{bmatrix} -\boldsymbol{G_q^2} \boldsymbol{n}_{target} \\ \mathbf{0} \end{bmatrix} \tag{2.52}$$

shows that only the top left block of the inverse of the right hand side of Eq. 2.51 needs to be known.

The formula for the blockwise matrix inverse gives (only the relevant block is computed):

$$\left[\boldsymbol{\Omega}^2 + r\boldsymbol{I}\right]^{-1} = \begin{bmatrix} \left[\boldsymbol{I}(1+r) + \boldsymbol{G_q^2} - \boldsymbol{G_q}(\boldsymbol{G_q^2} + \boldsymbol{I}r)^{-1}\boldsymbol{G_q}\right]^{-1} & \cdot \\ \cdot & \cdot \end{bmatrix}. \tag{2.53}$$

Finally, the optimal coordinates are obtained in closed form:

$$\tilde{\mathbf{n}} = -\left[\boldsymbol{I}(1+r) + \boldsymbol{G_q^2} - \boldsymbol{G_q}(\boldsymbol{G_q^2} + \boldsymbol{I}r)^{-1}\boldsymbol{G_q}\right]^{-1} \boldsymbol{G_q^2} \boldsymbol{n}_{target}. \tag{2.54}$$

The main merit of this formula is that it gives a closed form solution for a stable (w.r.t. small changes in $\boldsymbol{q}$) set of coordinates as close as possible to a target shape.

The quadratic loss equals:

$$\tilde{\mathbf{n}}^T \tilde{\mathbf{n}} = \boldsymbol{n}_{target}^T \boldsymbol{G_q^2} \left[\boldsymbol{I}(1+r) + \boldsymbol{G_q^2} - \boldsymbol{G_q}(\boldsymbol{G_q^2} + \boldsymbol{I}r)^{-1}\boldsymbol{G_q}\right]^{-2} \boldsymbol{G_q^2} \boldsymbol{n}_{target} \tag{2.55}$$

It is now straightforward to apply gradient descent on $\boldsymbol{q}$ by evaluating the gradient of $\tilde{\mathbf{n}}^T\tilde{\mathbf{n}}$ for the current best value of $\boldsymbol{q}$. In practice, one only needs to lower limit the force densities of the springs during the gradient descent updates.

Figure 2.10 shows the result of an optimization run. The left plot shows the initial shape (minimal 6-bar structure), the center plot shows the target shape and the gradient descent solution is presented on the right. The optimization was halted when the quadratic loss dropped to $0.1\,\mathrm{m}^2$. In this case the target shape was an invalid shape in the sense that no valid set of force densities exists to keep the target structure in static equilibrium. The result of the optimization is thus a feasible tensegrity structure that approximates the desired invalid shape as well as possible (L_2 norm).

Malerba introduced a gradient descent method for form-finding under general (differentiable) constraints for cable nets [114]. The core idea of their method is to iteratively solve for a modified gradient that obeys the constraints (project the gradient onto the constraint). They solve for the minimum norm change of the force density vector ($\Delta\boldsymbol{q}$) that satisfies the

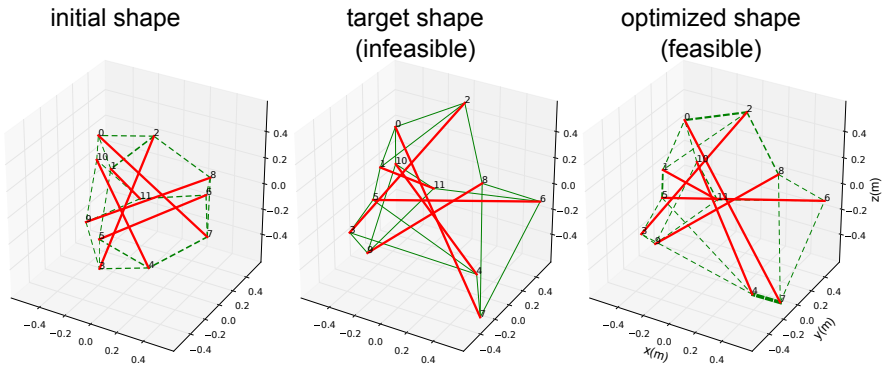


Figure 2.10: Shape optimization of a tensegrity structure to approximate a target shape. The left plot shows the initial shape (minimal 6-bar structure), the center plot shows the target shape and the gradient descent solution is presented on the right. In this case the target shape was an invalid shape in the sense that no valid set of force densities exists to keep the target structure in static equilibrium. The result of the optimization is thus a feasible tensegrity structure that approximates the desired invalid shape as well as possible.

constraints. Their method lends itself easily to find a modified gradient close to the desired gradient of my method $\frac{\partial \tilde{\mathbf{n}}^T \tilde{\mathbf{n}}}{\partial \boldsymbol{q}}$. This can be implemented by a simple change of coordinates in Eq. 16 and 17 of [114]. Replace $\boldsymbol{r}$ with $\Delta \boldsymbol{r} + \boldsymbol{G}^T \frac{\partial \tilde{\mathbf{n}}^T \tilde{\mathbf{n}}}{\partial \boldsymbol{q}}$ and add $\frac{\partial \tilde{\mathbf{n}}^T \tilde{\mathbf{n}}}{\partial \boldsymbol{q}}$ to the solution of their algorithm to obtain the modified gradient. If this method is used, the constraints will be satisfied at each step during the optimization. Note that $\boldsymbol{G}$ here refers to the Jacobian of the constraints as defined in Eq. 14 of [114]. Eq. 22 of Malerba is problematic for tensegrity structures as the stress matrix is not positive definite for free-standing structures. Fortunately, this is of little practical concern, as one can compute the derivative of the constraints directly.

2.7.5 Optimizing Force Densities in a Multiple Self-Stress Configuration

Section 2.2 showed that $\boldsymbol{A}\boldsymbol{q} = \boldsymbol{0}$ is a necessary condition for a tensegrity in static equilibrium. If this equation has multiple linearly independent solutions for $\boldsymbol{q}$, then any linear combination of these will also satisfy the condition. The solution space corresponds to the null space of $\boldsymbol{A}$. If the dimension of the null space is v and if the columns of the matrix $\boldsymbol{V}$ are a

set of v linearly independent vectors in the null space of $\boldsymbol{A}$, then one can rewrite the original condition as:

$$\boldsymbol{AVc} = \boldsymbol{0}, \tag{2.56}$$

where $\boldsymbol{c}$ is a v-dimensional vector.

This simple trick reduces the search space for a good set of force densities, as all $\boldsymbol{q}$ satisfying $\boldsymbol{Aq} = \boldsymbol{0}$ can be produced by evaluating $\boldsymbol{Vc}$. However, additional inequality constraints are needed to ensure that the force densities of spring-cable assemblies are positive. Note that for all practical purposes, the rows of $\boldsymbol{V}$ and $\boldsymbol{q}$ corresponding to compressive members can be omitted. To keep the notation simple it is assumed that $\boldsymbol{V}$ and $\boldsymbol{q}$ have their compressive member rows removed in this and the next section.

Now consider the goal of minimizing the elastic potential energy of a structure under the constraint that the spring-cable force densities are larger than some quantity:

$$\begin{aligned} \text{argmin}_{\boldsymbol{c}} u &= \boldsymbol{c}^T \boldsymbol{V}^T \text{diag}_v(\boldsymbol{\alpha}) \boldsymbol{Vc} && (2.57) \\ &= \boldsymbol{c}^T \boldsymbol{Bc} && (2.58) \end{aligned}$$

subject to

$$-\boldsymbol{Vc} < -\boldsymbol{q}_{min} \tag{2.59}$$

with

$$\alpha_i = \frac{l_i^2}{2k_i}, \tag{2.60}$$

where the fact was used that the elastic potential energy of a member can be written as $u = \alpha q^2 = \frac{l^2}{2k}\left[\frac{k(l-l_0)}{l}\right]^2 = \frac{k(l-l_0)^2}{2}$. The optimization assumes that the potential energy is changed by modifying the rest length of the spring-cable assemblies, therefore $\boldsymbol{\alpha}$ will be constant throughout the optimization. This is a quadratic programming (QP) problem with linear constraints, which can be handled by common QP solvers.

Note that the constant elastic potential energy constraint implies that the Euclidean norm of the eigenvalues of the stiffness matrix $\boldsymbol{K}$ (which depends on the rest lengths) remains constant. This means that $\dfrac{\partial \operatorname{Tr}\left(\boldsymbol{K}^2\right)}{\partial \boldsymbol{c}} = 0$. It can be seen that $\operatorname{Tr}\left(\boldsymbol{K}^2\right)$ is the square of the Frobenius norm of $\boldsymbol{K}$, which can also be defined as the sum of the squares of the elements of the stiffness matrix. This also constrains the resonance frequencies, which depend on the mass distribution of the structure and the stiffness matrix. I therefore

argue that the elastic potential energy constraint is natural. It allows to transfer energy between vibration modes of the structure, which can be useful in structures or robots in which certain modes are damped more than others. Active vibration control is an important topic in tensegrity research for structural engineering [8, 27, 139].

How can the dimension of the null space be changed? A straightforward way to increase the dimension of the null space of a structure, is to add redundant spring-cable assemblies [20]. A parallel spring trivially increases the dimension of the null space by one. One application of this method is the homogenization of the tensions in the structure. This can be useful in an active structure, as highly pretensioned members typically only allow for small changes in length before plastic deformation. Tran et al. also considered the form-finding problem of structures with multiple self-stress states in [177].

Note that only the balance of forces was enforced throughout the optimization. This means that the optimization can potentially yield results with additional mechanisms (deformations that require no mechanical work). However, because this technique primarily applies to redundant structures, this is unlikely to happen in practice (additional members tend to remove mechanisms [64]).

2.7.6 Energy Preserving Self-Stress Control

I now present a method to control the self-stress of a tensegrity structure with multiple self-stress states — redundant structures with nullity $(\boldsymbol{A}) > 1$. The goal is to optimize an objective function under the constraint that the structure maintains its original static equilibrium shape (i.e. there is no movement of the bars) and that there is no change in the total elastic potential energy of the structure. If the structure is in static equilibrium, the exchange of potential energy between the spring-cable assemblies entails that no net mechanical work is needed to *move* within this constrained self-stress space. Two objectives are considered here: Tuning of the elastic potential energy in a target shape $\boldsymbol{n}_{target}$ and stiffening of the eigenmodes of the structure.

A related set of methods in structural engineering are topological optimization methods [9, 112, 128]. Topological optimization allows to tune a structure's frequency response, to minimize its mass and to solve similar problems. The methods I present here are thus in a sense specific applications of topological optimization. The benefits over the generic methods are the increased computational efficiency for this type of structure (tensegrities) and the simpler mathematical formulation.

There are various possible applications of the self-stress control technique. Consider for example the tensegrity spine example from Section 2.7.4. In this case, one might want to vary the stiffness of a particular movement or stiffness mode to handle loading, impact or to design turning strategies in which the controls remain fixed.

In addition to this, variable stiffness is a common principle in biology and human locomotion [43, 44, 47]. In this regard, stiffness tuning plays a central role in locomotion as a mechanism to use energy efficiently. In aquatic locomotion, trouts (living and euthanized) have been observed to exploit the energy of the flow they inhabit to reduce their swimming efforts [see 104, for a review] and variable stiffness is also believed to be a common principle in land locomotion for the adaptation to surfaces with changing mechanical properties [see 147, and references therein]. The phenomenon extends to other behaviors, such as breathing and thermal control. There is evidence that entrainment phenomena are present in the control of airflow in humans, guinea pigs, panting in dogs and in pigeons [186], and this would accomplish the dual aim of maximizing flow and minimizing the work required from the respiratory muscles.

In principle, the presented method can be extended to include dynamics, which would be equivalent to impedance control of the full robot [75]. However, this requires significantly more state information or estimation, which is a hard problem for a free-standing tensegrity.

As redundant tensegrities have a large number of spring-cable assemblies by design, full actuation is expected to be problematic in hardware. Moored presented an interesting variation on the common tensegrity principle by investigating clustered actuation [125]. Smart coupling (e.g. through pulleys) of tensile members could allow for a design that can switch between self-stress states at very low energetic cost with a reduced number of actuators.

The core concept of the method is a constrained gradient descent/ascent algorithm. The objective function L is written in terms of the null space coordinates:

$$\text{argmax}_{\boldsymbol{c}} L(\boldsymbol{c}) \tag{2.61}$$

subject to

$$u_{eq}(\boldsymbol{c}_t) = u_{eq}(\boldsymbol{c}_0), \tag{2.62}$$

where $\boldsymbol{c}_t$ is the solution at iteration t of the algorithm and u_{eq} is the elastic potential energy function in the equilibrium shape. It was shown in Section 2.7.5 that $u_{eq}(\boldsymbol{c})$ can be written as $u_{eq} = \boldsymbol{c}^T \boldsymbol{B} \boldsymbol{c}$. Two interesting choices for $L(\boldsymbol{c})$ are discussed at the end of this section. The constraint ensures

that the structure is at constant elastic potential energy throughout the full optimization run, which implies that an actuation pattern along the (smooth) trajectory $\boldsymbol{c}_0 \dots \boldsymbol{c}_t$ will not cause any movement of the structure.

Now write the gradient descent/ascent step explicitely:

$$\boldsymbol{c}_{t+1} = \text{argmin}_{\boldsymbol{c}} \|\boldsymbol{c} - \boldsymbol{c}_{prop}\|^2 \tag{2.63}$$

subject to

$$\boldsymbol{c}_{t+1}^T \boldsymbol{B} \boldsymbol{c}_{t+1} = u_{eq}(\boldsymbol{c}_0) \tag{2.64}$$

with

$$\boldsymbol{c}_{prop} = \boldsymbol{c}_t + \mu \frac{\partial L(\boldsymbol{c})}{\partial \boldsymbol{c}}(\boldsymbol{c}_t), \tag{2.65}$$

where μ is the step size and $\boldsymbol{c}_{prop}$ are the proposed coordinates by the gradient step. The proposed coordinates are then mapped onto the nearest $\boldsymbol{c}_{t+1}$ satisfying the constraint.

The result is a quadratic problem with quadratic constraints [62]. It is well known that such a problem can be cast as a generalized eigenvalue problem [54, 175]. However, I will follow a standard Langrange multiplier approach here, as this results in a particularly efficient implementation. The time complexity to find the Lagrange multiplier is of the order $O(\log(b)b)$, where b is the dimension of the null space. This is due to the complexity of Newton's method (assuming this method is used to solve the constraint).

The Lagrangian of this problem is given by:

$$\begin{aligned} \mathcal{L}\left(\boldsymbol{c}_{t+1}, \lambda\right) = &\left(\boldsymbol{c}_{t+1} - \boldsymbol{c}_{prop}\right)^T \left(\boldsymbol{c}_{t+1} - \boldsymbol{c}_{prop}\right) \\ &+ \lambda \left(\boldsymbol{c}_{t+1}^T \boldsymbol{B} \boldsymbol{c}_{t+1} - u_{eq}(\boldsymbol{c}_0)\right), \end{aligned} \tag{2.66}$$

where λ is a Lagrange multiplier.

Setting the partial derivatives of $\mathcal{L}\left(\boldsymbol{c}_{t+1}, \lambda\right)$ to zero and solving for $\boldsymbol{c}_{t+1}$ gives:

$$\boldsymbol{c}_{t+1} = \left(\boldsymbol{I} + \lambda \boldsymbol{B}\right)^{-1} \boldsymbol{c}_{prop} \tag{2.67}$$

$$u_{eq}(\boldsymbol{c}_0) = \boldsymbol{c}_{prop}^T \left(\boldsymbol{I} + \lambda \boldsymbol{B}\right)^{-1} \boldsymbol{B} \left(\boldsymbol{I} + \lambda \boldsymbol{B}\right)^{-1} \boldsymbol{c}_{prop}. \tag{2.68}$$

Finding the optimal λ can be simplified by using the eigendecomposition $\boldsymbol{B} = \boldsymbol{P} \text{diag}_v\left(\boldsymbol{d}\right) \boldsymbol{P}^T$ [18]:

$$\left(\boldsymbol{I} + \lambda \boldsymbol{B}\right)^{-1} = \boldsymbol{P} \left[\boldsymbol{I} + \lambda \text{diag}_v\left(\boldsymbol{d}\right)\right]^{-1} \boldsymbol{P}^T. \tag{2.69}$$

Now use the notation $\boldsymbol{m} = \boldsymbol{P}^T \boldsymbol{c}_{prop}$ (here $\boldsymbol{m}$ is a temporary variable to

simplify the notation) to reduce the problem to:

$$0 = \left(\| \left(\boldsymbol{I} + \lambda \operatorname{diag}_v \left(\boldsymbol{d} \right) \right)^{-1} \sqrt{\operatorname{diag}_v \left(\boldsymbol{d} \right)} \boldsymbol{m} \| - \sqrt{u_{eq}(\boldsymbol{c}_0)} \right)^2 \tag{2.70}$$

The gradient of the right hand side is given by:

$$\frac{\partial \left(\| \left(\boldsymbol{I} + \lambda \operatorname{diag}_v (\boldsymbol{d}) \right)^{-1} \sqrt{\operatorname{diag}_v \left(\boldsymbol{d} \right)} \boldsymbol{m} \| - \sqrt{u_{eq}(\boldsymbol{c}_0)} \right)^2}{\partial \lambda} = \tag{2.71}$$

$$-2 \left(1 - \sqrt{u_{eq}(\boldsymbol{c}_0)} \left\| \left(\boldsymbol{I} + \lambda \operatorname{diag}_v \left(\boldsymbol{d} \right) \right)^{-1} \sqrt{\operatorname{diag}_v \left(\boldsymbol{d} \right)} \boldsymbol{m} \right\|^{-1} \right)$$

$$\boldsymbol{m}^{\boldsymbol{T}} \operatorname{diag}_v \left(\boldsymbol{d} \right)^2 \left(\boldsymbol{I} + \lambda \operatorname{diag}_v \left(\boldsymbol{d} \right) \right)^{-3} \boldsymbol{m}. \tag{2.72}$$

As before, the diag_v operator transforms a column vector into a diagonal matrix.

The Lagrange multiplier λ can now efficiently be found using the last two equations. In practice the truncated Newton method was found to perform well in this case with approximately 10 iterations per gradient step for a 6-bar structure with 54 springs.

Two things need clarification at this point. First, how are positive force densities maintained during optimization and secondly, why is gradient descent/ascent used while in theory any $\boldsymbol{c}_{prop}$ can be used? While the first issue could be solved by adding inequality constraints (additional Lagrange multipliers) to the optimization problem, I instead opted for a soft lower limit on $\boldsymbol{q}_{prop} = \boldsymbol{V}\boldsymbol{c}_{prop}$. The elements of $\boldsymbol{q}_{prop}$ below the lower limit are set to the lower limit and a least-squares solution is used to find the modified proposed $\boldsymbol{c}_{prop}$. The corrected new force densities $\boldsymbol{q}_{t+1} = \boldsymbol{V}\boldsymbol{c}_{t+1}$ can (slightly) violate the constraint, but this case will be corrected during the next iteration.

The answer to the second remark is that the outcome of the gradient ascent/descent algorithm is a continuous set of valid force densities. Each element of the sequence $\boldsymbol{c}_0 \ldots \boldsymbol{c}_t$ is a valid set of force densities and it is simple to ensure additional constraints such as positive spring-cable force densities. One could minimize the objective function (possibly in closed form) independent of the constraint and then find the nearest valid solution using the Lagrange multiplier. However, in this case it is harder to maintain valid force densities and depending on the objective function no continuous path of force densities might exist between the original state and the final solution. This assertion was verified by applying the intermediate steps found by the algorithm in a dynamic simulation. More precisely, the results of the example in the next section (with $\mu = 5$) were subsampled (1 in 10 gradient steps) and the next sample was applied 200 ms after the previous. In between two control time steps, the spring-cable rest lengths were linearly interpolated.

This indeed did not generate any measurable movement of the structure.

2.7.6.1 Tuning the Elastic Energy of a Target Shape

The first objective considered here is tuning of the elastic potential energy in a target shape $\boldsymbol{n}_{target}$. The rationale behind this objective is that maximizing the elastic potential energy in a target shape will cause deformation of the robot or structure in the direction of the target to become more difficult and to require more mechanical work. This can be particularly advantageous to anticipate an impact force as it allows to maximize the energy absorption of the structure. A similar effect is seen in bipedal walking: Stretching your leg allows more energy to be stored and tightening your leg muscles allows you to handle higher drops [3]. *Tuning* here means that I wish to be able to both maximize and minimize the required work, i.e. I wish to know the range over which the elastic potential energy u_{target} of a target shape can be varied. This is illustrated in Figure 2.11.

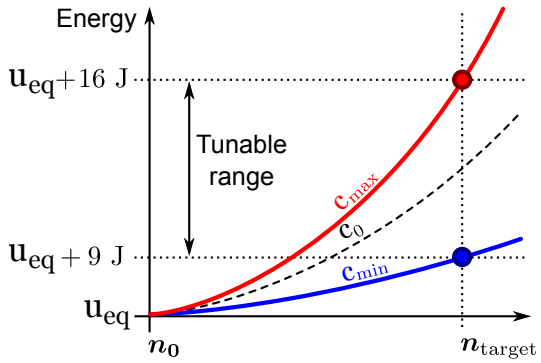


Figure 2.11: Illustration of target energy tuning. The objective is to enable maximization and minimization of the required mechanical work (i.e. the elastic energy u_{target}) to deform the structure from the equilibrium shape $\boldsymbol{n}_0$ to the shape $\boldsymbol{n}_{target}$, while keeping the energy u_{eq} of the equilibrium shape constant. The algorithm optimizes the null space coordinates $\boldsymbol{v}$, assuming that actuation is performed by changing the rest lengths $\boldsymbol{l}_0$ of the spring-cable assemblies.

The objective function $L(\boldsymbol{c})$ is in this case given by:

$$L(\boldsymbol{c}) = \pm u_{target}(\boldsymbol{c}), \tag{2.73}$$

where the sign allows to choose between maximization and minimization of the elastic energy in the target shape.

Using the assumption that the actuation is performed by changing the rest length of the spring-cable assemblies, the gradient step can be written as:

$$\boldsymbol{c}_{prop} = \boldsymbol{c}_t + \mu \frac{\partial L(\boldsymbol{c})}{\partial \boldsymbol{c}}(\boldsymbol{c}_t) \tag{2.74}$$

$$= (\boldsymbol{I} \pm 2\mu \boldsymbol{B})\boldsymbol{c}_t \pm \mu \boldsymbol{V}^T \mathrm{diag}_v(\boldsymbol{\beta}), \tag{2.75}$$

where $\beta_i = l_i(l_{i,target} - l_i)$.

Figure 2.12 shows an example of this algorithm. In this case 36 spring-cable assemblies were added to the minimal 6-bar structure shown in Figure 2.1 for a total of 54 tensile members. The total elastic potential energy in the system in equilibrium was 384 J. Even for the small displacement shown in this example (the average nodal displacement was 0.06 m), the elastic potential energy in the target shape could be varied significantly between 9 J a 16 J above the equilibrium state. During minimization the average spring tension tends to drop, while the total elastic potential energy is constant. This is possible because the spring-cable assemblies have different stiffness coefficients and rest lengths.

Figure 2.13 shows the connectivity of the structure from the example in Figure 2.12 and how the tension on the spring-cable assemblies changes due to the maximization of the energy of the target shape. The tension of spring-cable assemblies which seem to disappear remained constant. This representation allows the identification of the spring-cable assemblies that are more relevant for the change in elastic energy between the two shapes.

2.7.6.2 Stiffness Tuning

The second objective I consider is tuning the eigenvalues of the stiffness matrix $\boldsymbol{K}$. Similar to the previous section, this has a number of practical purposes. In practice it can often be advantageous to stiffen the most flexible eigenmode. An alternative purpose which has yet to be studied is the tuning of a tensegrity structure for optimal energy recuperation. Stiffness tuning is an important aspect of energy harvester design [26, 41].

Let the eigenvalues of $\boldsymbol{K}$ be ω_i with corresponding eigenvectors $\boldsymbol{v}_i$. The objective is then simply given by (the rigid body eigenmodes are implicitly discarded):

$$L(\boldsymbol{c}) = \omega_i. \tag{2.76}$$

In practice, the goal will be to maximize the objective function or move the eigenvalue closer to a desired value. Minimization of an eigenvalue typically results in a zero eigenvalue and thus in the creation of a mechanism.

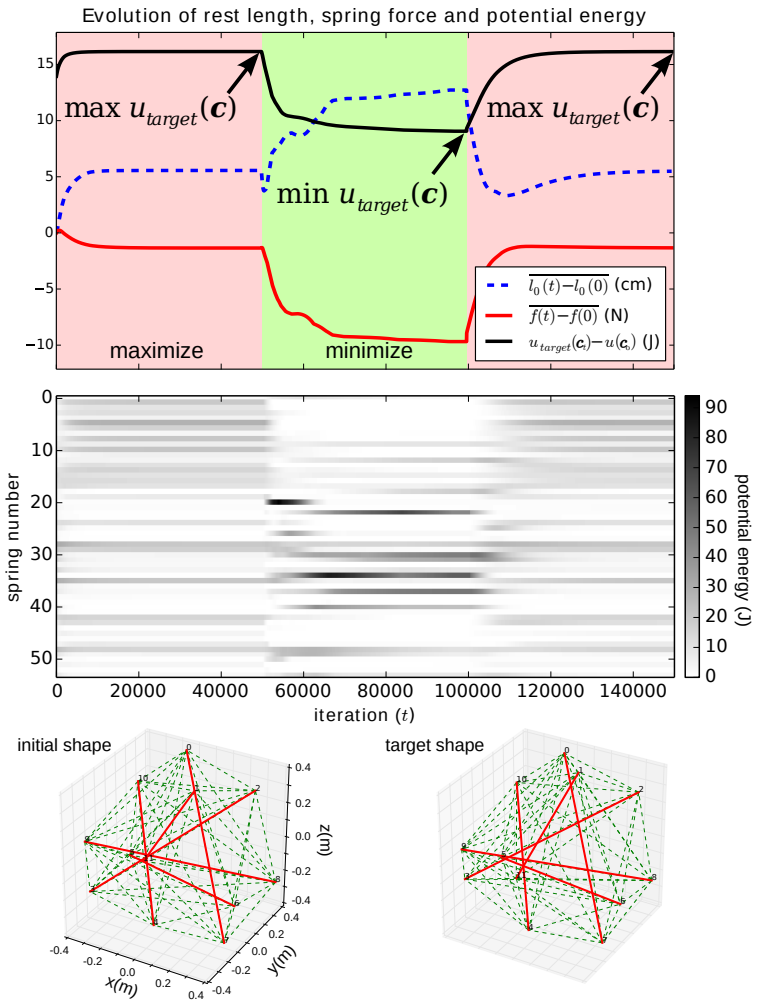


Figure 2.12: Example of the optimization of the elastic potential energy in a target shape. Bottom left: equilibrium shape. Bottom right: target shape (avg. nodal displacement $0.06\,\mathrm{m}$). Starting from a random initial set of self-stress parameters $\boldsymbol{c}_0$, the elastic potential energy in the target shape was first maximized (red region in the top plot), then minimized (green region) and then maximized again to show how the method allows for continuous modification of the elastic potential energy. In the top plot, the black line shows the elastic potential energy in the target shape $\boldsymbol{n}_{target}$, the red line is the average change in spring tension and the dashed blue line is the average change in rest length of the spring-cable assemblies. Even for the small displacement shown in this example the elastic potential energy in the target shape could be varied significantly between $9\,\mathrm{J}$ and $16\,\mathrm{J}$ above the equilibrium state.

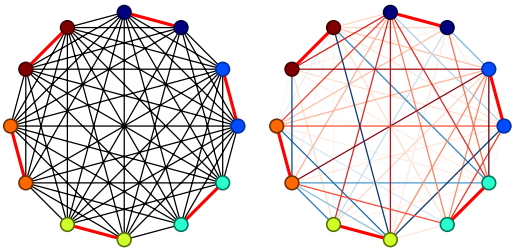


Figure 2.13: Force density change. The figure depicts the structure as a graph, where the nodes of the graph are the extrema of the bars (thicker lines) and the edges (thinner lines) represent the tensile members. Nodes of the same bar are colored accordingly. The left panel shows the connectivity of the structure. In the right panel the springs are colored according to the change in tension relative to the initial state. Darker indicates bigger changes, springs that seem to disappear experienced smaller changes. The color shows if the change is an increment (red) or decrement (blue) of tension.

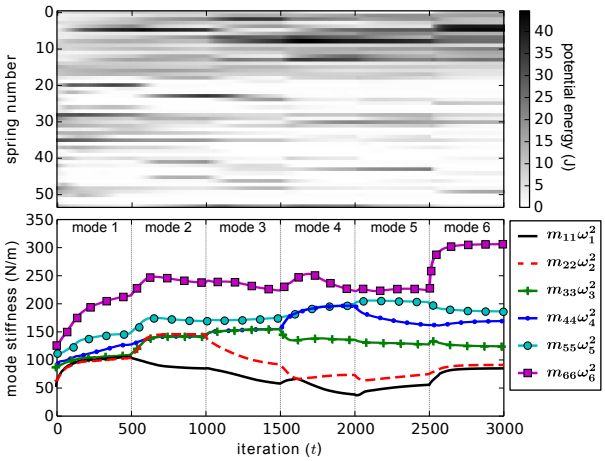


Figure 2.14: Maximization of the stiffness of different eigenmodes. During the first 500 iterations, the eigenvalue of the first non-rigid body mode is maximized. During the next 500 steps the second mode is stiffened and so on. Significant changes in modal stiffness can be obtained by this method. The top plot shows the elastic potential energy in each spring and the bottom plot shows the modal stiffness for the first 6 non-rigid body modes. The constant elastic potential energy constraint implies that the norm of the vector of modal stiffnesses is constant. The structure from Figure 2.12 was used. In the legend m_{ii} refers to the i-th entry of the structure's diagonalized mass matrix.

Recall the definition of the stiffness matrix:

$$\boldsymbol{K} = \boldsymbol{K}_g + \boldsymbol{K}_e. \tag{2.77}$$

Therefore, the partial derivatives of the objective function with respect to $\boldsymbol{c}$ can be written as:

$$\frac{\partial L(\boldsymbol{c})}{\partial \boldsymbol{c}} = \boldsymbol{v}^T \frac{\partial \boldsymbol{K}_g}{\partial \boldsymbol{c}} \boldsymbol{v} + \boldsymbol{v}^T \frac{\partial \boldsymbol{K}_e}{\partial \boldsymbol{c}} \boldsymbol{v} \tag{2.78}$$

$$= \boldsymbol{v}^T \frac{\partial \boldsymbol{K}_g}{\partial \boldsymbol{q}} \frac{\partial \boldsymbol{q}}{\partial \boldsymbol{c}} \boldsymbol{v} + \boldsymbol{v}^T \frac{\partial \boldsymbol{K}_e}{\partial \boldsymbol{q}} \frac{\partial \boldsymbol{q}}{\partial \boldsymbol{c}} \boldsymbol{v} \tag{2.79}$$

$$= \boldsymbol{v}^T \frac{\partial \boldsymbol{K}_g}{\partial \boldsymbol{q}} \boldsymbol{V}\boldsymbol{v} + \boldsymbol{v}^T \frac{\partial \boldsymbol{K}_e}{\partial \boldsymbol{q}} \boldsymbol{V}\boldsymbol{v} \tag{2.80}$$

$$\frac{\partial \boldsymbol{K}_g}{\partial q_i} = \boldsymbol{I}_3 \otimes \boldsymbol{C}_i^T \boldsymbol{C}_i \tag{2.81}$$

$$\frac{\partial \boldsymbol{K}_e}{\partial q_i} = -l_i \boldsymbol{A}_{.,i} \boldsymbol{A}_{.,i}^T \tag{2.82}$$

with $\boldsymbol{C}_i$ the ith row of the connectivity matrix $\boldsymbol{C}$ and $\boldsymbol{A}_{.,i}$ the i-th column of $\boldsymbol{A}$. The last equation is easily obtained using the equality $\boldsymbol{K}_e = \boldsymbol{A}\boldsymbol{L}^{-2}(\boldsymbol{S} - \boldsymbol{Q})\boldsymbol{A}^T$ and the assumption that actuation is performed by modifying the rest lengths (constant $\boldsymbol{S}$). Note that $\boldsymbol{v}$ (the eigenvector corresponding to the eigenvalue ω) is a function of $\boldsymbol{c}$ (the eigenvectors are a function of the force densities). This result is then substituted in Eq. 2.65.

This approach can be trivially extended to the tuning of the natural frequencies, by noting that the partial derivatives of an eigenvalue of the generalized eigenvalue problem $\boldsymbol{K}\boldsymbol{v} = \omega \boldsymbol{M}\boldsymbol{v}$ are given by $\partial\omega = \boldsymbol{v}^T \left(\partial \boldsymbol{K} - \omega \partial \boldsymbol{M}\right) \boldsymbol{v} = \boldsymbol{v}^T \partial \boldsymbol{K} \boldsymbol{v}$ [138].

In practice the optimization of this objective is slightly slower than the elastic potential energy optimization from the previous section, due to the repeated eigendecomposition at each gradient step. If needed, one can assume the eigenvectors to be constant during a few gradient steps.

Figure 2.14 demonstrates the stiffening of different modes of the structure. Initially the first (non-rigid body) mode is maximally stiffened. After 500 iterations, the next eigenvalue is maximized and so forth. It becomes clear that significant changes of the stiffness or resonance frequencies can be obtained by this method.

2.8 Dynamic Simulation

This section presents two simulation environments. To foster interest into tensegrity research, both simulation environments are open source and freely available at `http://ti.arc.nasa.gov/tech/asr/intelligent-robotics/tensegrity/`. The experimental results obtained with these simulators are presented in later chapters.

The first simulator directly implements the equations of motion described in Section 2.8.1. It is an extension of the dynamics formulation described by Skelton [157]. Below I reformulate Skelton's equations in a Cartesian coordinate system as this is more intuitive. My implementation adds support for ground contacts and other practical features. The second simulator is the NASA Tensegrity Robotics Tookit (Section 2.8.2). This simulator is based on the Bullet Physics engine and allows for complex environmental interactions, at the cost of a much simpler integration method. My main contribution to this simulator is a set of validation experiments using motion capture with the robot described in Chapter 6.

2.8.1 Constrained Spring-Mass Net

As much work has been done on the computational aspects of spring-mass nets, the equations of motion in this section are cast as the dynamics of a constrained spring-mass net. This means that the system is modeled as a set of point masses with coordinates $\boldsymbol{n}_i$ with a set of distance constraints for the fixed length struts. The result is a simple set of equations that can efficiently be implemented in matrix form.

As stated in Eq. 2.10 the nodal forces due to internal forces (spring-cable tensions) are given by $\boldsymbol{F}_{internal} = \boldsymbol{EN}$. In case external forces $\boldsymbol{F}_{ext}$ are present in the system, the nodal forces are given by:

$$\boldsymbol{F} = \boldsymbol{EN} + \boldsymbol{F}_{ext} + \boldsymbol{R}\dot{\boldsymbol{N}}, \tag{2.83}$$

where $\boldsymbol{R}\dot{\boldsymbol{N}}$ can be used to model linear damping (e.g. Rayleigh damping).

Now consider a single strut between nodes i and j. The acceleration of node i and j can be written as:

$$\ddot{\boldsymbol{n}}_i = m_i^{-1}\boldsymbol{f}_i + \lambda_{i,j}\frac{\partial\left(\boldsymbol{n}_i - \boldsymbol{n}_j\right)^T\left(\boldsymbol{n}_i - \boldsymbol{n}_j\right) - l_{i,j}^2}{\partial \boldsymbol{n}_i} \tag{2.84}$$

$$\ddot{\boldsymbol{n}}_i = m_i^{-1}\boldsymbol{f}_i + 2\lambda_{i,j}\left(\boldsymbol{n}_i - \boldsymbol{n}_j\right) \tag{2.85}$$

$$\ddot{\boldsymbol{n}}_j = m_j^{-1}\boldsymbol{f}_j + 2\lambda_{i,j}\left(\boldsymbol{n}_j - \boldsymbol{n}_i\right), \tag{2.86}$$

where the Lagrange multiplier $\lambda_{i,j}$ needs to be found to satisfy the constraint $(\boldsymbol{n}_i - \boldsymbol{n}_j)^T (\boldsymbol{n}_i - \boldsymbol{n}_j) = l_{i,j}^2$. Analogous to Skelton's equations of motion, this value can be found by computing the second time derivative of the constraint[5] [157]:

$$\frac{d^2 l_{i,j}^2}{dt^2} = \frac{d\, \boldsymbol{n}_i^T \boldsymbol{n}_j - 2\boldsymbol{n}_i^T \boldsymbol{n}_j + \boldsymbol{n}_i^T \boldsymbol{n}_j}{dt} \tag{2.87}$$

$$0 = (\dot{\boldsymbol{n}}_i - \dot{\boldsymbol{n}}_j)^T (\dot{\boldsymbol{n}}_i - \dot{\boldsymbol{n}}_j) + (\boldsymbol{n}_i - \boldsymbol{n}_j)^T (\ddot{\boldsymbol{n}}_i - \ddot{\boldsymbol{n}}_j)\,. \tag{2.88}$$

Subtracting $\boldsymbol{n}_i$ and $\boldsymbol{n}_j$ and substituting this in the last equation gives:

$$\ddot{\boldsymbol{n}}_i - \ddot{\boldsymbol{n}}_j = m_i^{-1} \boldsymbol{f}_i - m_j^{-1} \boldsymbol{f}_j + 4\lambda_{i,j} (\boldsymbol{n}_j - \boldsymbol{n}_i) \tag{2.89}$$

$$\begin{aligned} 0 = {} & \|\dot{\boldsymbol{n}}_i - \dot{\boldsymbol{n}}_j\|^2 \\ & + (\boldsymbol{n}_i - \boldsymbol{n}_j)^T \left(m_i^{-1} \boldsymbol{f}_i - m_j^{-1} \boldsymbol{f}_j + 4\lambda_{i,j} (\boldsymbol{n}_j - \boldsymbol{n}_i) \right). \end{aligned} \tag{2.90}$$

Using which the Lagrange multiplier can be found:

$$\lambda_{i,j} = -\frac{\|\dot{\boldsymbol{n}}_i - \dot{\boldsymbol{n}}_j\|^2 + (\boldsymbol{n}_i - \boldsymbol{n}_j)^T \left(m_i^{-1} \boldsymbol{f}_i - m_j^{-1} \boldsymbol{f}_j \right)}{4\|\boldsymbol{n}_i - \boldsymbol{n}_j\|^2} \tag{2.91}$$

The implementation of these equations is straightforward, as one can simply compute the (unconstrained) nodal forces followed by the evaluation of the Lagrange multipliers to find the constraint forces. Ground contacts were modeled using the penalty based method described in [195].

As each strut is modeled as two point masses[6] with a constraint, the effective number of degrees of freedom of a strut is 5. The main advantage of this non-minimal coordinate set, is the lack of direction cosines to describe the orientation of each strut. Wroldsen [191] demonstrates how to use a minimal set of generalized coordinates based on the coordinate system used by Skelton [157]. The resulting equations are however far less elucidative than the simple description presented here.

2.8.2 NASA Tensegrity Robotics Toolkit

The second simulator used in this work is the NASA Tensegrity Robotics Toolkit (NTRT), developed by the Intelligent Robotics Group at NASA Ames Research Center. This simulator is an extension of the Bullet Physics engine. Bullet is primarily aimed at performance while providing results that

[5] Thus replacing the fixed length constraint with the constraint that there is no acceleration along a bar.

[6] General mass distributions can be used by replacing the individual nodal masses m_i with a (block) mass matrix.

appear credible. Therefore, its integration method has limited accuracy (e.g. it is not energy conserving), which is cause for concern as the stability of tensegrity structures is defined by a subtle equilibrium of forces in the whole structure.

However, Bullet has an efficient and extensive collision detection algorithm, which is the main reason to focus on this simulator for future work on tensegrity locomotion. As presented in Chapter 6, significant effort was put into validation experiments of (both) simulators.

2.9 Control of Tensegrity Structures

The previous sections have mainly focused on the static and kinematic properties of tensegrity structures. However, the core focus of this thesis are free-standing structures that can move and deform in an environment. This section therefore provides a brief overview of related work on actuated structures, a domain which has only begun to develop in recent years.

The Applied Computing and Mechanics Laboratory (IMAC) at EPFL was one of the first laboratories to construct large-scale prototypes of actuated tensegrities [49, 48]. Their designs are room-sized structures targeting the structural engineering community, with a focus on smart load-bearing structures. Their main actuator design is based on telescopic struts. In recent years, research into actuated tensegrities for structural engineering has resulted in the design of deployable tensegrity bridges [99, 141, 142]. Also in the structural engineering realm, related research has focused on active vibration control [8, 27, 139]. These works are significant because of the large-scale hardware deployments and it becomes particularly exciting when one considers that at the other end of the spectrum tensegrity structures have been built using DNA as structural elements [105].

A few years before the IMAC prototypes, Sulton showed how actuated tensegrities can be used for telescope positioning [164]. Not long after that, Sultan and Skelton developed the mathematics for a tensegrity flight simulator (a 6 degrees of freedom actuated platform to simulate flight conditions) [165].

Early demonstrations were mainly focused on kinematic controls or targeted a very specific design. Skelton and Wroldsen presented results for feedback non-linear dynamic control of general tensegrity structures [157, 191]. However, this type of controls has been largely limited to constrained setups in which accurate state information is available [63, 123].

Around 2005, Paul and Lipson at Cornell University approached the control problem from a computer science perspective. They evolved controllers

in simulation capable of dynamic locomotion without precise state estimation or models of the dynamics of the systems [136]. While also at Cornell, Rieffel presented an original control approach in simulation based on spiking Neural Networks coupled through body dynamics [144].

Recently, Rieffel started focusing on very simple control methods for small, low-cost tensegrity robots using small vibration motors [97]. Similar work was also presented in 2013 by Bohm [15].

The BIER lab at the University of Virginia is researching Central Pattern Generator based controls for tensegrity based flapping wings (e.g. fish tails), which provides another interesting link between tensegrities and biology [12, 13, 14].

While the previous hardware implementations used electric motors, Koizumi and Shibata have developed rolling tensegrity structures based on pneumatic actuators [98, 156]. This is an interesting development, as typical electric motors are optimized for rotational motion, while tensegrity robotics would significantly benefit from efficient linear actuators.

What this literature overview shows is that tensegrities are reasonably well studied from a mechanical and structural point of view, but the number of practical demonstrations of actuated tensegrities is still limited. This last fact in particular applies to free-standing structures in real-world environments. Most of the algorithms developed in the next chapters require no detailed knowledge of the system dynamics or state and thus apply to tensegrity robots which can be deployed in unknown environments. This is fully in line with the goals of the NASA Innovative Advanced Concepts project — to which the work presented in this dissertation has significantly contributed [1, 87, 88] — and the European Community's FP7 AMARSi project. The compliant robotics technologies developed during the last decade, make it an ideal time for a leap forward in dynamically actuated tensegrity research.

3
Reservoir Computing

Reservoir Computing (RC) is a recurring theme throughout this thesis. As various excellent dissertations at the Reservoir Lab [183, 192] have been fully devoted to the study of RC systems, I limit the scope of this chapter to an overview of the main concepts and an introduction to the most relevant techniques. In this context, I will present two original contributions to the Echo State Network type of Reservoir Computing. In collaboration with Francis wyffels, I have experimentally verified the Echo State property of large Reservoirs [24]. Pieter Buteneers and I have developed an efficient technique for regularization parameter selection in feedforward Echo State Networks [18]. Considering this thesis' focus on physical systems, I discuss how the RC concept has been extended beyond the scope of artificial Neural Networks.

This chapter is organized as follows. Section 3.1 first reviews the concepts of Reservoir Computing in hardware and software architectures. This is followed by an in-depth discussion of the properties of Echo State Networks, a common software Reservoir Computing implementation, in Section 3.2. Before presenting my conclusions in Section 3.4, I discuss methods to solve tasks using Reservoir Computing (Section 3.3).

3.1 Overview of Reservoir Computing

3.1.1 Concept

The general concept of Reservoir Computing is to use a dynamical system as a computational black box. Information is fed into the system, which provides random projections of the input and short term memory. Reservoir Computing allows to efficiently use a large range of dynamical systems to

approximate a desired filter. In RC the dynamical system is left untouched, instead only an observation or output layer is trained. Extensions of the basic method allow to design systems which autonomously generate signals and thus have long term memory.

The clear advantage of RC is that only limited knowledge or control of the dynamical system is needed. This is at the same time the main disadvantage of these methods. Better performance could sometimes be obtained by training the computational substrate. However, Reservoir Computing is an attractive method as it tends to have good performance on various problems and can in general be implemented faster than competing algorithms. Because of these features, it is also an excellent baseline for more involved techniques.

3.1.2 In Silico Reservoirs

The most common implementations of Reservoir Computing are software based. Three types of software RC are well known: Liquid State Machines, Echo State Networks and Backpropagation-Decorrelation. Similar techniques have appeared in the literature prior to the introduction of RC, but they were not widely embraced. Echo State Networks were introduced by Jaeger in 2001 [90, 94] and this implementation is discussed in depth in the next sections. Liquid State Machines were developed by Maass [113] around the same time and provide a more biologically plausible perspective. This technique is typically implemented as a set of (continuous time) differential equations. Finally, Steil presented an efficient single step backpropagation algorithm which uses a Reservoir at its core [162]. As much of the knowledge and intuition transfers between these approaches, the general principle of these methods later became known as Reservoir Computing [185].

3.1.3 Physical Reservoirs

While artificial Reservoirs have now been studied for well over a decade, there is a recent trend to study the computational or Reservoir properties of physical systems. In fact, the possibility of physical Reservoirs was realized early during the onset of RC systems, with demonstrations such as a bucket of water used as an RC system [45]. The recent developments in physical implementations of RC are focused on applications that benefit from the fact that RC allows to exploit computational capabilities without precise control of all the aspects of the physical system. One domain that has seen a significant influence from RC techniques is optoelectrical and all-optical computing [40, 100, 133]. Much of this work was initially simulation based [52],

but recently optical RC has been demonstrated in a silicon photonics chip with passive waveguides [182]. In the next chapters, I focus on the Physical RC properties of compliant tensegrity robots. Section 4.4 will revisit the Physical RC concept and explains how it can be adapted to robotics in practice.

It was recently shown that a large class of dynamical systems (artificial or physical) inherently have an equal amount of *information processing capacity* [32]. These are not computations in the Turing sense [180]. Instead, the result from [32] essentially shows how difficult it is to linearly approximate (w.r.t. the quadratic norm) desired transformations of an input stream based on observations of the state of a dynamical system. One caveat is that a desired type of processing might be present in a physical system, but it can be unfeasible to extract it due to sensor limitations. On the contrary, a system might appear unsuitable for a computational task, while an intelligent encoding can dramatically increase the performance. By this last statement I mean that a dynamical system might for example fail at encoding a desired input transformation as pulses, while it performs optimally when encoding it in the frequency spectrum. In this case, the desired information processing is available in the system, but it is hard to effectively use it.

When making claims about computational properties of a system it is therefore ultimately useful to focus on the problems one is interested in solving:

> Beware of the Turing tar-pit in which everything is possible but nothing of interest is easy. [137]

For example, there is no point in using a bucket of water as a general computer beyond the sake of research. Similarly, there is little value in utilizing a compliant robot to add numbers. But how is Physical RC then any different from centuries-old analog or mechanical computers [80, 37]? The answer is that Physical RC systems need not be designed to solve the task at hand. The method allows to exploit computations inherently performed by the dynamical system.

It is my belief that the real issue is to maximally exploit *useful* computations available in a dynamical system. Reservoir Computing provides a means to use or interpret a physical substrate as a computational tool, even if not all its details are fully known or understood. In the context of compliant robotics, one interesting goal is to use Physical RC for locomotion control, as information from the interaction of the robot with its environment is inherently processed. The control of a robot is interpreted as a computational problem.

3.2 Echo State Networks

I now return to the artificial Neural Network RC implementation called Echo State Networks.

3.2.1 Mathematical Formulation

Echo State Networks (ESN) are a common software implementation of Reservoir Computing [90]. While their performance can often be surpassed by a properly trained Recurrent Neural Network with a similar number of neurons, the key features of ESNs are their simple implementation, limited number of parameters and wide applicability. A complete ESN implementation typically only takes a few lines of code in a language with support for matrix operations.

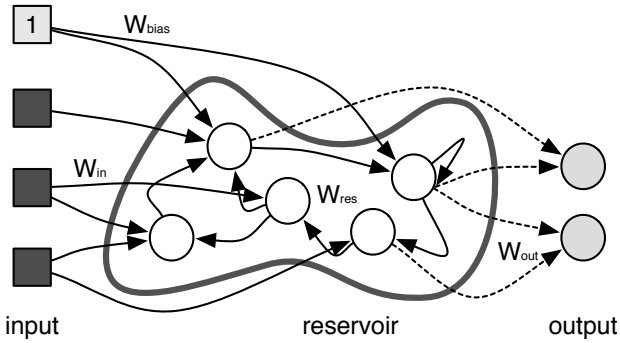


Figure 3.1: Schematic overview of an Echo State Network RC system with 3 inputs (on the left) and 2 outputs (on the right). The Reservoir is the random Recurrent Neural Network in the center of the figure. The key concept is to only train the output connections $\boldsymbol{W}_{out}$, while the other connections are fixed.

Figure 3.1 shows the architecture of a standard ESN with multiple inputs and outputs. At its core, an ESN has a — typically large — discrete time non-linear Recurrent Neural Network, called the Reservoir. The Reservoir is fixed, meaning that the weights of the internal connections remain constant. The hyperbolic tangent function is the most common non-linearity used for ESNs. This squashing function ensures that the values of the neurons in the Reservoir are bounded. Unlike continuous time Reservoirs (e.g. Liquid State Machines), the topology of ESN Reservoirs does not appear to have a significant impact on their performance. In short, the Reservoir section of an ESN is an arbitrary Recurrent Neural Network with a sigmoid non-linearity.

Two sources feed into the Reservoir: the inputs of the system and a bias. Both the bias and the inputs are connected to the Reservoir through random fixed weights. Often, the bias is not explicitly mentioned, as it can be modeled as a fixed constant input.

Formally, for a Reservoir with n neurons, the weights of the connections within the Reservoir are represented by a matrix $\boldsymbol{W}_{res}$ of size $n \times n$. Additionally, matrix $\boldsymbol{W}_{in}$ and vector $\boldsymbol{W}_{bias}$ represent the connection weights from the input to the Reservoir and from a bias to the Reservoir respectively. Typically, $\boldsymbol{W}_{bias}$ has dimensions $n \times 1$ and $\boldsymbol{W}_{in}$ has dimensions $n \times u$, where u is equal to the number of inputs to the Reservoir. After sampling these weights from a random distribution, e.g. a standard normal distribution, the update of the system's state is written as:

$$\boldsymbol{x}[d+1] = \tanh(\boldsymbol{W}_{res}\boldsymbol{x}[d] + \boldsymbol{W}_{in}\boldsymbol{u}[d+1] + \boldsymbol{w}_{bias}), \tag{3.1}$$

where $\boldsymbol{u}$ is the input of the system. The output $\boldsymbol{y}$ of an ESN is defined by[1]:

$$\boldsymbol{y}[d+1] = \boldsymbol{W}_{out}\boldsymbol{x}[d+1], \tag{3.2}$$

where $\boldsymbol{W}_{out}$ are the connection weights from the Reservoir to the output. The dimensions of this weight matrix are $n \times o$, where o equals the number of outputs. The goal of ESN training algorithms is to adapt the only non-fixed matrix $\boldsymbol{W}_{out}$ to approximate the desired output $\boldsymbol{y}$.

3.2.2 Parameters

Echo State Networks are unique in that their properties are to a large extent defined by random sets of weights. However, it turns out that the obtainable performance by optimal training of the output layer primarily depends on a few scalar parameters. More precisely, it is crucial to choose a good scaling of the input matrix $\boldsymbol{W}_{in}$, bias vector $\boldsymbol{w}_{bias}$ and spectral radius. These parameters depend on the problem at hand, but some intuition exists. The bias scaling pushes the neurons of the Reservoir into the non-linear regime of the hyperbolic tangent, which tends to be useful for non-linear computations. For too large input scalings, the memory of the Reservoir tends to be destroyed, as the inputs force the state of the network. The last variable, the spectral radius, is studied in more detail in the following sections.

One of the key principles behind RC is the *Echo State Property* (ESP) introduced by [90]. A Reservoir system exhibits the ESP if it forgets all previous input after a limited time, i.e. it cannot have infinitely long memory

[1] Other types of readouts (e.g. winner take all or soft-max) also appear in the literature, but are not considered here.

(similar to the concept of fading memory introduced in [113]). In other words, without any external input, the system's state should converge to a single fixed point. In order to tune the dynamics of the Reservoir, many researchers use the spectral radius ρ, which is defined as the largest absolute eigenvalue of the Reservoir weight matrix $\boldsymbol{W}_{res}$.

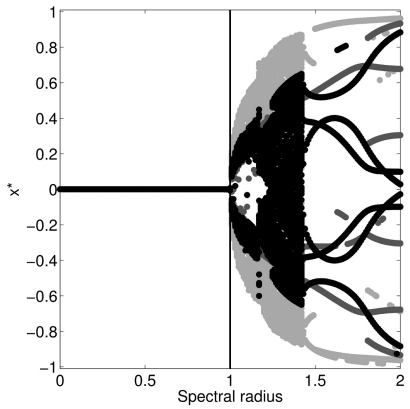


Figure 3.2: Bifurcation diagram of a 128-dimensional Reservoir with zero input and zero bias. The equilibrium points for three randomly selected neurons are visualised. By increasing the spectral radius, the system bifurcates from a single fixed point to spontaneous activity.

The effect of the spectral radius on the dynamics of the Reservoir system becomes clear when the bifurcation diagrams of the Reservoirs are traced. In Figure 3.2 the bifurcation diagram is shown for a 128-dimensional network with neither input, nor bias. The bifurcation diagram shows the different equilibrium points (i.e. local extrema) of three randomly selected neurons of a simulated Reservoir after many different initializations. For $\rho < 1.0$, one observes that the system's state converges to a fixed point at the origin. At $\rho = 1$ the system undergoes a bifurcation which makes the Reservoir dependent on its initial condition. Consequently, the ESP does not hold anymore. This bifurcation point does not always occur for $\rho = 1$. When the system is fed an input signal or a constant bias, this bifurcation point can be observed for a spectral radius slightly larger than 1, see Figure 3.3. Due to the nonlinearity of the system — which has maximum gain at zero input — this will be the case in almost all practical situations in which the Reservoir is excited with one or more input signals.

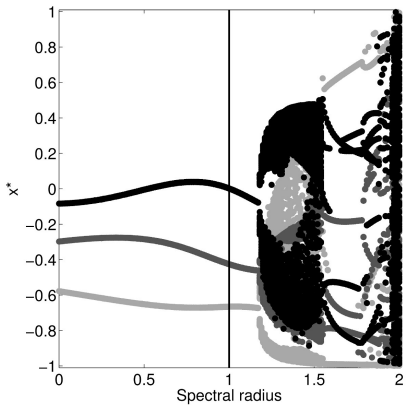


Figure 3.3: Bifurcation diagram of 128-dimensional network fed a constant bias. The equilibrium points for three randomly selected neurons are visualised. By increasing the spectral radius, the system bifurcates from a singled fixed point to spontaneous activity. The constant input (e.g. bias) postpones this bifurcation point.

3.2.3 Echo State Property Revisited

The Echo State Property (ESP) is a key concept in Reservoir Computing which can be formally stated as [90]:

Definition 1. *A network $F : X \times U \to X$ (with the compactness condition) has the Echo State Property with respect to U, if for any left infinite input sequence $u^{-\infty} \in U^{-\infty}$ and any two state vector sequences $x^{-\infty}$, $y^{-\infty} \in X^{-\infty}$ compatible with $u^{-\infty}$, it holds that $x_0 = y_0$.*

Consequences of the ESP are that the current network state only depends on a certain number of previous inputs and is not influenced by the initial state after a certain period of time (often called warm-up period). Reservoirs with the ESP can be used as nonlinear finite impulse response filters. Instead of crafting the Neural Network such that it performs a certain task (i.e. emulates some desired filter), one typically combines the available nonlinear projections of the Reservoir in a linear fashion, assuming that the desired nonlinear computations are available in the system. This will be explained in detail in Section 3.3.

Therefore, it is customary to study the global properties of Reservoirs with respect to a few parameters, such as the spectral radius, input bias and leak rate. In this spirit, the linear memory capacity [70] has been studied as

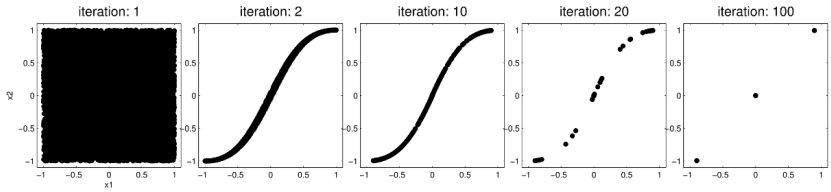


Figure 3.4: A low spectral radius ($\rho = 0.39$) two neuron network without the Echo State Property. 1×10^5 initial states were sampled (uniform $\in [-1, 1]^2$) and the states were recorded for multiple time steps (increasing from left to right). All initial states converge to the origin or continue to oscillate between $\pm[0.8975\, 0.9946]^{\mathrm{T}}$. The weight matrix is given by $\boldsymbol{W}_{res} = [-3\, 1.24; -5.968\, 2.416]$. This is not a degenerate case, as small variations of the weights also result in a non-ESP network (e.g. $\boldsymbol{W}_{res} = [-3\, 1.2; -6\, 2.4]$, $\rho = 0.6$).

well as the apparent tradeoff between linear memory and nonlinearity [184]. Rules of thumb are that high bias networks are useful for highly nonlinear computations (because of the nonlinear behavior of the hyperbolic tangent) and a high spectral radius results in longer memory. See [111] for a complete overview of design strategies.

One complication of the ESP is that it is input dependent. There has been some research into the input dependent ESP [115], but in practice it is often impossible to know all statistics of the input sequences beforehand. The ESP is thus mostly studied independently of the input sequence.

Different methods exist to verify if a given Reservoir exhibits the ESP. The most commonly used method (for hyperbolic tangent networks) is to compute the spectral radius of the weight matrix ($\rho = \max_i |\lambda_i|$). If the spectral radius is below unity, one assumes that the ESP is fulfilled. The rationale for this approach is the fact that the hyperbolic tangent has the highest gain at the origin and one thus regards the linear system with the same weight matrix as an upper bound for the stability (eigenvalues within the unit disk).

Unfortunately the spectral radius method is not sufficient for the ESP (e.g. [196]) and it is possible to construct low spectral radius ($\rho << 1$) counterexamples. Consider Figure 3.4, which shows the state progression for a 2-dimensional network with $\rho = 0.39$ and neither input, nor bias. The system was initialized in 1×10^5 random states and the update equation (Eq. 3.1) was applied. After a few iterations, all initial states contract into the origin or begin oscillating between two states. One intuitive explanation for such behavior is that the nonlinear network weakens negative feedback.

It is therefore possible to construct networks with very large weights, which have low spectral radius and do not exhibit the ESP.

Multiple sufficient conditions or tests for the ESP have been proposed. In his original work, Jaeger [90] proved that having the largest singular value of the weight matrix below unity is sufficient for the ESP. It is easy to show that $\max_i(\sigma_i) \geq \rho$, because any consistent matrix norm has a higher value than the spectral radius. Only for normal weight matrices ($\boldsymbol{W}_{res}^* \boldsymbol{W}_{res} = \boldsymbol{W}_{res} \boldsymbol{W}_{res}^*$), both norms coincide and the SVD condition is thus more restrictive than the spectral radius condition.

More recently, the ESP has been studied in terms of Lyapunov exponents [185], operator norms [17] and Schur stability [196]. Nevertheless, there are multiple reasons to study the usefulness of the spectral radius method. First of all, the proposed methods are generally more complex to verify. Secondly, it will be shown that the spectral radius method is often a tight bound for the ESP.

The Schur stability method by Yildiz et al., the largest singular value test and the spectral radius method are now considered in more detail for small, zero input and zero bias networks.

The Schur stability method is stated as a linear matrix inequality condition [196]:

Definition 2. *A zero bias hyperbolic tangent Reservoir has the Echo State Property for any input if its weight matrix $\boldsymbol{W}_{res}$ is diagonally Schur stable, i.e. there exists a diagonal matrix $\boldsymbol{P} > 0$ such that $\boldsymbol{W}_{res}^T \boldsymbol{P} \boldsymbol{W}_{res} - \boldsymbol{P}$ is negative definite.*

Figure 3.5 and Figure 3.6 show the fraction of rejected weight matrices for the different methods for 2 and 8 neuron Reservoirs respectively with i.i.d. normally distributed weights as a function of the spectral radius averaged over 1×10^4 random Reservoirs. To check whether a network has the ESP, each network was initialized in 1×10^3 random (uniform $\in [-1, 1]^2$ or $\in [-1, 1]^8$) states and updated for 1×10^3 iterations. If the norm of any final state was above 1×10^{-7}, the network was considered to not have the ESP as not all states contracted to the origin. This is the baseline, indicated as *non-ESP*.

It becomes clear that the singular value test and the Schur method reject many networks for high spectral radii, while the fraction of Reservoirs that effectively do not have the ESP for $\rho < 1$ is many times lower. Furthermore, the bounds tend to become weaker for larger networks, while the fraction of non-ESP networks becomes smaller.

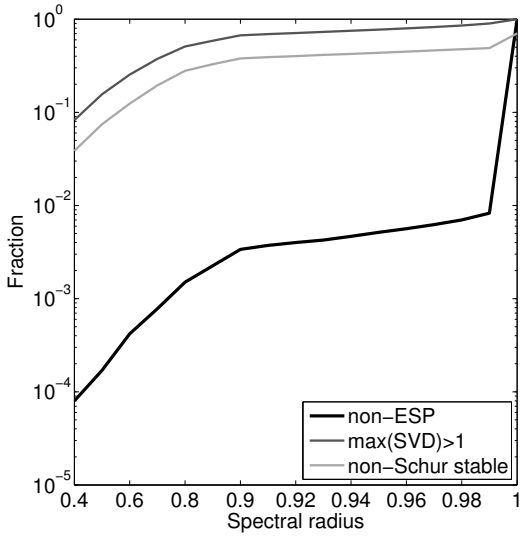


Figure 3.5: Fraction of the 2-dimensional networks (zero input, zero bias) that do not have the experimentally verified ESP in function of ρ, fraction of the 2-dimensional networks with $\max(\mathrm{SVD}) > 1$ and fraction of the 2-dimensional networks which are not Schur stable.

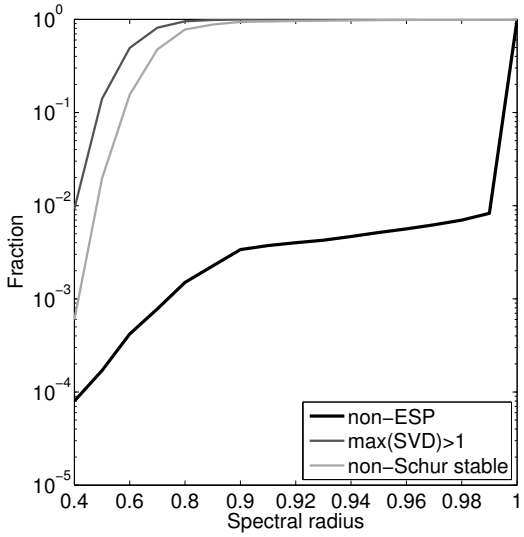


Figure 3.6: Fraction of the 8-dimensional networks (zero input, zero bias) that do not have the experimentally verified ESP in function of ρ, fraction of the 8-dimensional networks with $\max(\mathrm{SVD}) > 1$ and fraction of the 8-dimensional networks which are not Schur stable.

3.2.4 Echo State Property in Large Reservoirs

To study the ESP in large Reservoirs, an extensive numerical verification was performed. These experiments are designed to show the influence of the Reservoir size and connectivity on the ESP.

3.2.4.1 Experimental Setup

In a large-scale numerical experiment the number of neurons n, the connectivity (the fraction of weights that is non-zero) c of the Reservoir weight matrix and the spectral radius ρ were varied:

$$\begin{aligned} n &\in \{2, 4, 8, 16, 32, 64, 128, 256\} \\ c &\in \{0.01, 0.0167, 0.0278, \\ &\quad 0.0464, 0.0774, 0.1292, \\ &\quad 0.2154, 0.3594, 0.5995, 1.0\} \\ \rho &\in \{0.4, 0.5, 0.6, 0.7, 0.8, 0.85, 0.9, \\ &\quad 0.91, 0.92, 0.93, 0.94, 0.95, 0.96, \\ &\quad 0.97, 0.98, 0.99, 1, 1.01, 1.05, 1.1, 1.2\}. \end{aligned}$$

Each parameter combination was tested for 1×10^5 randomly generated networks with weights $\boldsymbol{W}_{res}$ sampled from a standard normal distribution and 1×10^3 random (uniform $\in [-1, 1]^N$) initial states. To test the ESP for a (zero-input) network, each network was updated by applying Eq. 3.1 with zero bias and zero input ($\boldsymbol{u}$ and $\boldsymbol{w}_{bias}$ equal to $\mathbf{0}$) for 1000 iterations for each initial state. The largest norm of the final states ($\|\boldsymbol{x}[1001]\|$) was then stored. Results are presented for fully dense weight matrices with varying network size and for networks with 128 neurons with varying connectivity.

3.2.4.2 Results

Figures 3.7 and 3.8 show the fraction of networks for which the ESP does not hold ($\|\boldsymbol{x}[1001]\| > 10^{-7}$) as a function of the spectral radius with respect to network size and connectivity respectively. One can observe in Figure 3.7 that for relatively large (fully connected) networks ($n > 32$) the probability of finding a network without the ESP is below 1×10^{-3}. This is interesting because most Reservoir systems of practical use are quite large ($n > 50$) and, consequently they are not affected by the fact that the spectral radius method is not a sufficient condition. Additionally, from Figure 3.8 one learns that the connectivity greatly influences the ESP. The sparser the network, the less likely it is to exhibit the ESP. For very sparse networks, with a connectivity

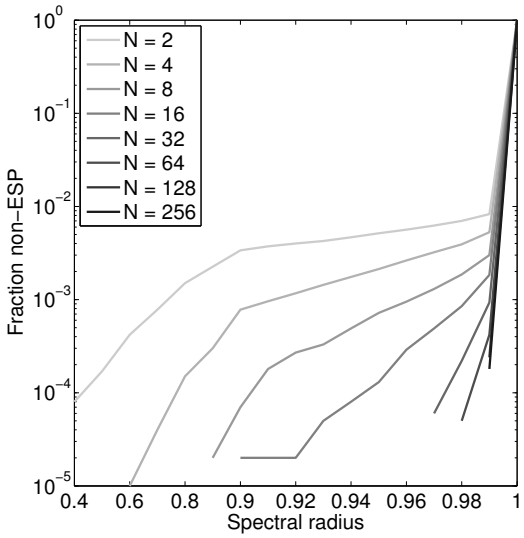


Figure 3.7: Fraction of the fully connected networks with varying size for which the ESP does not hold in function of the spectral radius. The larger the network, the less likely that it does not exhibit the ESP for $\rho < 1.0$.

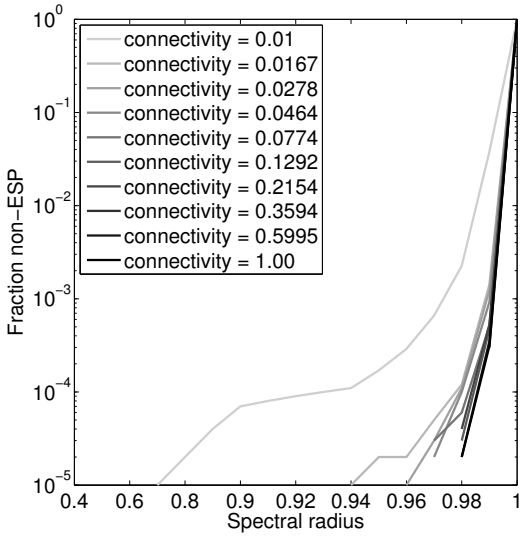


Figure 3.8: Fraction of the 128-dimensional networks with varying connectivity for which the ESP does not hold in function of the spectral radius. Larger or denser networks with $\rho < 1.0$ are less likely not to exhibit the ESP.

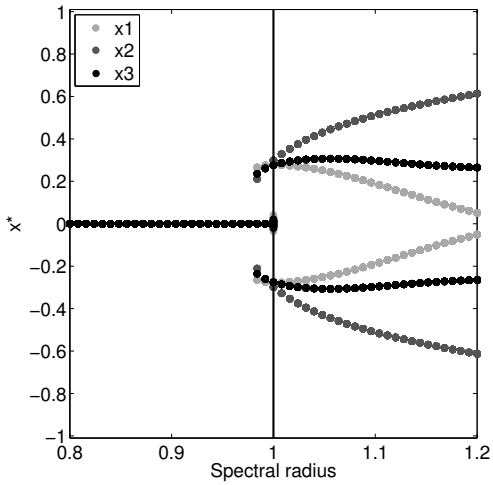


Figure 3.9: Bifurcation diagram for a fully connected 128-dimensional network. As can be observed in the bifurcation diagram, the ESP does *also* not hold for $\rho < 1.0$.

of 1%, the fraction increases to 3.8 % of the networks. In comparison, only 0.031 % of the fully connected 128-dimensional networks did not exhibit the ESP. This observation corresponds with the intuition that it is very likely to find oscillating sub-networks, e.g. the small networks given in [196], in sparse networks.

In [196] a method is given for constructing large Reservoir systems for which the ESP does not hold for spectral radii below 1. By following that procedure, the obtained networks will be sparse. A logical question is whether large dense networks can be found for which the ESP does not hold with $\rho < 1$. The answer is positive, as can be understood from Figures 3.7 and 3.8. A bifurcation plot of such a system is given in Figure 3.9. For $\rho < 0.98$ this system exhibits the ESP. At $\rho \approx 0.98$ the system bifurcates and starts to oscillate for some of the initial conditions. This contrasts with the behavior of a *normal* Reservoir as depicted in Figure 3.2.

3.2.4.3 Discussion

The spectral radius is the most commonly used indicator for the dynamics of a Reservoir. As a rule of thumb, it is assumed that a Reservoir will exhibit the Echo State Property for spectral radii below unity. The ESP indicates that a Reservoir has fading memory and thus that the network state will eventually become independent of the initial state and past inputs. One

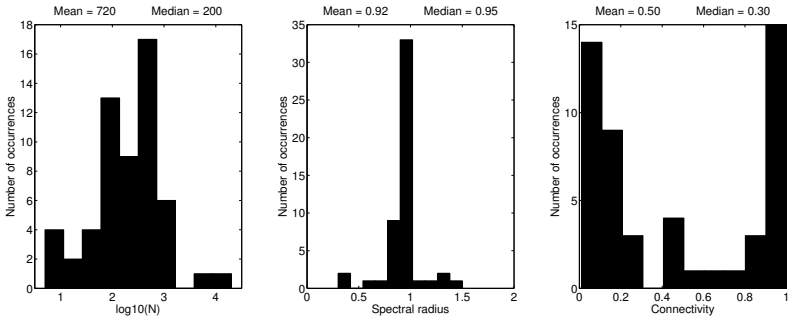


Figure 3.10: Commonly used parameters for Reservoir Computing: the Reservoir size n, spectral radius ρ and the connectivity c. All accessible studies citing [185] were consulted. Researchers tend to use relatively large networks. The majority of the researchers use a spectral radius slightly below unity. Typically very sparsely or very densely connected networks are preferred. This indicates that most studies used parameter ranges for which the spectral radius is a good indicator for the ESP.

consequence of this is that a zero input, zero bias network has to converge to the origin (see Figure 3.2).

However, as has been indicated in the past by a number of researchers (e.g. [111]) and pointed out explicitly in [196], this simple rule does not always hold. Low-dimensional examples in [196] illustrate that in some cases a network oscillates despite the fact that $\rho < 1$. These low-dimensional examples can be extended to higher-dimensional networks, however these are always sparse by construction. By large-scale numerical experimentation also explicit examples of large networks ($n \geq 128$) were found for which the ESP does not hold for $\rho < 1$ (see for example Figure 3.9).

Fortunately, it was observed that the fraction of such networks rapidly decreases with increasing network size. Apart from Reservoir size, the connectivity of the network also influences the ESP. In particular, it was shown that sparser networks with $\rho < 1$ are more likely not to exhibit the ESP compared to dense Reservoirs with equal spectral radius.

These findings do not suggest that the spectral radius should always be below 1. As the bifurcation plot in Figure 3.3 indicates, inputs also greatly influence the ESP. One should consider the exploration of larger ρ, like [196] and see ρ as a task dependent global parameter for optimization.

To get an overview of how Reservoirs are typically tuned, all accessible papers citing [185] that use hyperbolic tangent neurons were analyzed. It can be seen that the majority of the researchers use $0.9 < \rho < 1.0$ (see Figure 3.10). However, practical experience shows that the spectral radius

should be much larger or much smaller for some tasks and consequently that this distribution should be more bell-shaped.

The meta-analysis also shows that most researchers are using large networks. This is positive, since the ESP is more likely to hold for $\rho < 1$ in such networks. More surprising is the connectivity used in many Reservoirs. There seem to be two factions; one preferring fully dense networks for which the spectral radius seems to be a valid indicator for the ESP in general and the other preferring very sparse networks. Although the latter group comprises a significant portion of the studies, this does not necessarily indicate a problem as the networks were typically large.

In [90] and [196] different metrics for the ESP were given. By large-scale experimentation on 2- and 8-dimensional networks, it was shown that both the largest singular value method and the Schur stability are too restrictive conditions for practical use. Further experiments on larger networks indicated that the probability of a network with spectral radius below unity not exhibiting the ESP quickly drops as a function of the network size for zero input, zero bias networks. In conclusion, the spectral radius remains a good indicator of the ESP, especially in large Reservoir systems.

3.3 Solving Tasks

3.3.1 Feedforward Tasks

The properties of ESNs have been discussed in depth, but the training methods have yet to be explained. Recall that the output of a standard ESN is a linear combination of the state of the network's neurons:

$$\boldsymbol{y}[d+1] = \boldsymbol{W}_{out}\boldsymbol{x}[d+1]. \tag{3.3}$$

The most common and simplest application of ESNs are batch regression tasks. In this case, a sequence of inputs $\boldsymbol{u}[0], \boldsymbol{u}[1] \ldots$ is fed into the network and the resulting state sequence $\boldsymbol{x}[1], \boldsymbol{x}[2] \ldots$ can be observed[2]. The goal is to approximate the target sequence $\boldsymbol{z}[1], \boldsymbol{z}[2] \ldots$ as a linear combination of the state sequence. Linear regression solves this problem by minimizing the mean-squared error (MSE). Write $\boldsymbol{X}^T = [\boldsymbol{x}[1]\, \boldsymbol{x}[2]\, \ldots]$ and $\boldsymbol{Z}^T = [\boldsymbol{z}[1]\, \boldsymbol{z}[2]\, \ldots]$. Minimization of the MSE can now be written as:

$$\arg\min_{\boldsymbol{W}_{out}} = \|\boldsymbol{X}\boldsymbol{W}_{out}^T - \boldsymbol{Z}\|^2, \tag{3.4}$$

[2]Note the one time step shift between the inputs and outputs.

where the Frobenius norm was used. A closed form solution to this problem is given by the well known equation:

$$\boldsymbol{W}_{out}^{T} = \left(\boldsymbol{X}^{T}\boldsymbol{X}\right)^{-1}\boldsymbol{X}^{T}\boldsymbol{Z}. \tag{3.5}$$

This simple formula combined with the state update equations of the ESN are sufficient to build a working RC system. The basic recipe for a feedforward regression task using an ESN is thus:

- Initialize a random Recurrent Neural Network, input matrix and bias vector ($\boldsymbol{W}_{res}$, $\boldsymbol{W}_{in}$ and $\boldsymbol{w}_{bias}$).
- Rescale the Reservoir weight matrix to obtain a suitable spectral radius for the task at hand (typically $0.5 < \rho < 1.2$).
- Compute and store the network states by applying Eq. 3.1 to $\boldsymbol{u}[0], \boldsymbol{u}[1] \ldots$
- Apply Eq. 3.5 to compute the output weight matrix $\boldsymbol{W}_{out}$.

Once $\boldsymbol{W}_{out}$ is computed, the system can be used to predict new outputs based on a different input sequence $\boldsymbol{u}_{test}[0], \boldsymbol{u}_{test}[1] \ldots$. The general flow is to train the system using an input sequence $\boldsymbol{u}[0], \boldsymbol{u}[1] \ldots$ and matching target output sequence $\boldsymbol{z}[1], \boldsymbol{z}[2] \ldots$ and then use the trained system to predict the output of a test input sequence $\boldsymbol{u}_{test}[0], \boldsymbol{u}_{test}[1] \ldots$. The testing phase is the most important part of the whole process: A well-trained system should generalize to new input data. A learning system that simply stores and reproduces the training samples it has been fed will achieve perfect performance on the training set, but it will utterly fail to generalize to unseen input data.

The initialization is one minor aspect of ESNs has not yet been properly addressed. Section 3.2.3 discussed the short term memory of ESNs. The state of networks with the Echo State Property becomes independent of past inputs. An ESN can thus be initialized in an arbitrary state and after a *warm-up* sequence the state sequence of the network will be independent of the initial state. Initialization is thus not an issue for ESNs. However, it is important to not consider the state sequence corresponding to the warm-up samples in the learning algorithm. This means that the indices in $\boldsymbol{x}[1]\,\boldsymbol{x}[2] \ldots$ are counted start from the first sample after the warm-up sequence. The length of the warm-up period is typically on the order the network size.

In practice, optimizing the behavior of an ESN for a feedforward regression task primarily involves tuning of 4 scalar parameters: the spectral radius, the input scaling, the bias scaling and the regularization parameter of the output layer. This last parameter is explained in the next section.

3.3.1.1 Efficient Ridge Regression

A well-trained system should generalize to new input data. The assumption is that the test data has the same properties as the training data. It is therefore possible to split the training data into a smaller training set and a *testlike* set of which the targets are known. To prevent confusion, this last dataset is called a *validation* set as it is used to validate the performance of the algorithm trained on the (reduced) training set. This approach makes it possible to introduce additional (hyper)parameters that allow to optimize the generalization to unseen data. Testing and final evaluation of the performance is still based on the test set of which the target outputs are not known to the training method. It is imperative to only use the test set for evaluation purposes and not include it in the training phase in any way.

Various techniques exist to split the original training data into training and validation sets such that optimal use can be made of the available data. These techniques are commonly known as cross-validation. In a k-fold cross-validation scheme, the data is split into k subsets. One subset becomes the validation set and the $k-1$ remaining are used to train the system. This process is repeated k times, such that each subset takes on the role of the validation set once. An extreme form of cross-validation is leave-one-out cross-validation in which the validation set consists of a single sample. As RC methods are applied to sequential data, cross-validation for RC should split the data into chunks instead of randomly selected subsets. Leave-one-out cross-validation is thus generally not applicable and 5-fold or 10-fold versions are more common.

Cross-validation makes it possible to evaluate the generalization properties of a trained system. However, the parameters that have been explained so far (spectral radius, input scaling and bias scaling) mainly influence the computational properties of the system. They have no direct contribution to the trained output layer of the network and thus provide no control over the generalization performance.

In contrast, the regularization parameter, introduced in this section, constrains the weights of the output layer. This prevents overfitting on the training samples and thus to improve generalization properties. The common method to constrain the Euclidean norm of the output weights is called ridge regression. Ridge regression minimizes the following loss function [174]:

$$\arg\min_{\boldsymbol{W}_{out}} = \|\boldsymbol{X}\boldsymbol{W}_{out}^T - \boldsymbol{Z}\|^2 + \lambda\|\boldsymbol{W}_{out}\|^2, \tag{3.6}$$

where λ is the regularization parameter. A closed form solution to this

problem is given by:

$$\boldsymbol{W}_{out}^{T} = \left(\boldsymbol{X}^{T}\boldsymbol{X} + \lambda \boldsymbol{I}\right)^{-1} \boldsymbol{X}^{T}\boldsymbol{Z}. \tag{3.7}$$

Note that in linear regression and ridge regression, the output dimensions are independent. One is thus free to select different regularization parameters per dimension.

There is no closed form formula to select the optimal λ. Instead, a line search is needed to reveal the best value. For high-dimensional data it can be computationally demanding to evaluate Eq. 3.7 for a large number of λ values. In [18] Pieter Buteneers and I have shown that by precomputing the eigenvalue decomposition of the cross-validation subset covariance matrices there is no increase in computational cost with respect to the unregularized solution in most cases. More precisely, we have shown that for a limited number of folds ($kn < m$) and regularization parameters to test ($r < n$), the complexity of finding the optimal regularization parameter and calculating the optimal output weights order $O(n^2 m)$. Here k is the number of folds, n the number of neurons in the ESN, m the number of training samples and r the number of regularization parameters to test.

3.3.1.2 Other Loss Functions & Training Algorithms

The squared loss function is appropriate for regression problems. However, it is well known that it is generally not a good fit for other types of problems such as classification [11]. In feedforward applications of RC, i.e. in which there are no connections from the output layer to the input, the output layer is independent of the reservoir dynamics and it is thus straightforward to use a more appropriate training technique for the problem at hand.

In this context RC methods even appear in unsupervised learning setups for which the target output is not known. This was demonstrated by Antonelo [5], who used RC in combination with Slow Feature Analysis for autonomous robot navigation.

3.3.2 Feedback Tasks

While the simple training methods discussed in the previous sections already make RC applicable to a large set of problems, it is hard to advocate a method for Recurrent Neural Networks not capable of emulating a rather basic component such as a flip-flop. It turns out that a simple, yet ingenious trick extends RC to tasks that rely on long-term memory. The basic concept is illustrated in Figure 3.11. Fixed random feedback connections $\boldsymbol{W}_{fb}$ are

added from the output layer to the Reservoir. The connections from the Reservoir to the output are still the only trained weights.

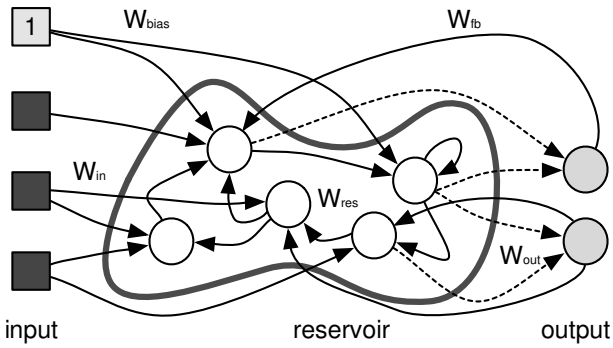


Figure 3.11: Schematic overview of an Echo State Network RC for feedback tasks. The setup differs from Figure 3.1 in that feedback connections have been added from the output neurons to the Reservoir. This makes training methods more complex, but allows for tasks with long-term memory (e.g. a latch) to be implemented.

The question is how the learning strategy for feedforward task can be adapted to this modified architecture? Changes to the $\boldsymbol{W}_{out}$ weights now affect the system in the same way modifications of the internal connections of a Recurrent Neural Network do. However, the essential difference is that the desired outputs are known and only the layer directly connected to the output is trained. This still allows for efficient, simple and robust learning techniques.

3.3.2.1 Teacher Forcing

The simplest way to train an ESN with output feedback is to use the techniques available for the feedforward case and cut the network at the output layer [91]. During training the desired output is fed back into the Reservoir through the feedback connections, while the output layer is trained as before. This technique is known as Teacher Forcing, because the teacher or training data is forced into the Reservoir during training. The method relies on the fact that after training the output approximates the target signals. Thus once the training phase is finished, the actual output can be fed back into the system and the recurrent setup is operational. To increase the robustness and probability of success, noise is commonly added to the training signals or injected into the Reservoir.

3.3.2.2 Recursive Least Squares and FORCE Learning

More elaborate techniques have been developed, which improve upon the original Teacher Forcing method. The most common such methods rely upon the Recursive Least Squares (RLS) algorithm or so-called FORCE learning. The main advantages of the newer methods are increased robustness[3] and online learning capabilities. FORCE learning was introduced by Sussillo and Abbott in 2009 and uses initially chaotic Reservoirs (spectral radius typically significantly above 1) [168]. Learning occurs by adapting the weights to the output neurons in an online fashion with the feedback connections intact. The authors also show how FORCE learning can be adapted to more complex network structures in which the error is not defined locally. The next chapter relies on the RLS algorithm and its details will be explained there.

3.4 Conclusion

Reservoir Computing is a recurring theme throughout this dissertation and this chapter presented the core ideas, properties and training methods. More precisely, I have provided an overview of Reservoir Computing implementations, discussed the underlying principles and presented a review of common training methods for Reservoir Computing systems. Reservoir Computing is in essence a simple approach to exploit the computational resources of a dynamical system. Due to the limited number of assumptions on the underlying substrate, Reservoir Computing has a wide range of possible applications.

The software implementations of Reservoir Computing have been well studied over the last decade. However, it is my opinion that the main innovation in recent years has been the move towards studies of the Reservoir Computing properties of physical systems. The central topic of the next two chapters is Physical Reservoir Computing for tensegrity robots. In this respect, this chapter serves as a basic reference for the terminology and algorithms used there.

[3] The Teacher Forcing method tends to highly depend on an optimal choice of the regularization parameter.

4
Tensegrity Structures as Computational Devices

The goal of this chapter is to demonstrate how tensegrity structures can be seen as computational devices. The methods presented are applied to the lowest level of control, at which sensor data feeds back to the motor drivers through simple static functions. This results in robust controllers for rhythmic motions, which can be integrated with higher level controllers such as Central Pattern Generators or more classic control schemes.

This chapter is organized as follows. First I provide a general overview of the methods and ideas in Section 4.1. This is followed by a discussion of related work in Section 4.2. Afterwards, I introduce Central Pattern Generators (Section 4.3) and define the Physical Reservoir Computing concept in the context of tensegrity robots (Section 4.4). I continue with an extensive experimental Section (4.5) before ending with my conclusions (Section 4.6).

4.1 Overview

The experimental setup is analogous to Reservoir Computing with artifical Neural Networks, which is why I will discuss the concept of Physical Reservoir Computing in the context of tensegrity robotics to link physical systems and artificial Neural Network techniques. The objective is to demonstrate that sufficient computational power can be provided by the nonlinearities and memory in the system dynamics. This statement will be validated by locomotion experiments in simulation.

I believe compliant tensegrity robots to be a natural fit for such simple, low level control methods. My first argument for this is that compliant tensegrity robots are similar to soft robots for which precise control is often not needed or not feasible. The compliance of soft robots allows to cope with small control errors, model inaccuracies or complex environments [178].

For example, a robotic trunk based on a muscular hydrostat can manipulate objects of various shapes without accurate knowledge about the object at hand [120, 187].

However, unlike typical soft robots, tensegrities have well defined and controllable mechanical properties. I showed in Section 2.7.6.2 how the stiffness and related properties of a tensegrity can be tuned. This opens up a straightforward path to match a (rhythmic motion) controller to the (oscillation) properties of a structure and vice versa. For the sake of simplicity, the properties of the structures in this chapter are fixed, except for the actuated spring-cable assemblies.

From a mathematical or control theory perspective, the methods presented in this chapter are extremely simple: I only consider linear feedback methods (for non-linear systems). Additional benefits of the presented approach are that it provides a simple way to use low cost, nonlinear sensors in a useful manner and that it provides insights into how controllers (brains) and structures (bodies) integrate and can work together.

As explained in the previous chapter, Reservoir Computing is originally a very simple approach in which the Neural Networks are fixed during training. The consequence of this is that the results are typically *sub-optimal*, by which I mean that an optimally trained Neural Network could solve a task with fewer computational units. The benefit of the Reservoir Computing inspired approach stems from its simplicity and direct applicability to various problems. One does not need precise knowledge of the computational substrate (robot, Neural Network...) to obtain results.

Throughout this chapter the assumption is made that the target or desired motor signals are known. This means that the often difficult question of finding optimal motor signals for a desired behavior (e.g. through inverse dynamics calculations) should be solved by other means. The main focus here are techniques to emulate or replace an external controller by exploiting the dynamics of a compliant tensegrity robot. In the next chapter, this line of work will be extended to directly learn feedback controllers based on a single reward signal.

4.2 Background and Related Work

Various partner labs have been working on related methods within the European Union FP7 AMARSi project. This both shows the interest in such simple control methods for compliant robots and provides useful validation experiments on various hardware and software platforms. Hauser et al. [68, 67] have shown that spring-mass nets have universal computational power. They

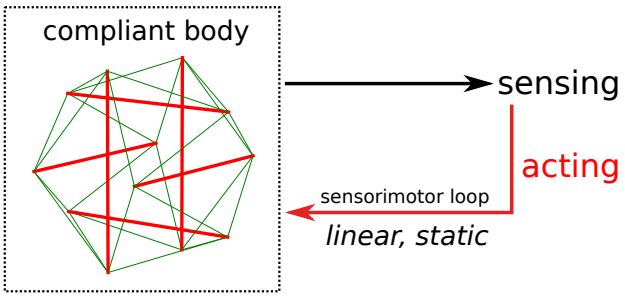


Figure 4.1: Overview of the approach.

have demonstrated that with reasonable assumptions, spring-mass nets can be used to approximate any non-linear filter with fading memory. This provides a theoretical foundation for Morphological Computation by casting it as a form of Reservoir Computing. To be more precise, Reservoir Computing methods allow to simplify the control problem by exploiting computations inherently present in the dynamics of a spring-mass net. Tensegrity robots are highly similar to spring-mass nets, but have the considerable advantage that they are free-standing. The tensegrity approach thus makes it possible to construct physical devices that are highly similar to a widely used theoretical model.

The Artificial Intelligence Lab at the University of Zurich has applied this approach to two platforms: quadruped spines and soft robotic arms. The quadruped spine experiments are interesting, because a compliant spine could be retrofitted into various existing robots (e.g. the Oncilla robot developed within the AMARSi projects). For the quadruped spine experiments, force sensitive resistors were used as sensor devices [199, 200]. The soft robotic arms are based on the muscular hydrostat principle [127]. Muscular hydrostats are continuous soft systems in which stiffness and forces are generated by the constant volume constraint of the system. Common examples of muscular hydrostats are tongues, octopus arms and elephant trunks.

Building on these findings, I show that Morphological Computation can be used to effectively control compliant tensegrity structures. An overview of the set-up can be seen in Figure 4.1. Contrary to the experiments at the University of Zurich, this chapter focuses on online learning. The reason for this is that it is both more natural from a biological perspective and also tends to be more robust. My examples also go significantly further than the now classic demonstrations from Paul, who was the first to conceive simple robots that performed computation through the body [134]. Indeed, Paul also considered tensegrity structures for morphological computation, but the controller in her set up was still external [136].

Note that in this work I only focus on pattern generation through feedback to the body. I do not develop the sensing aspect in this dissertation, although I did some initial work on the topic [23]. More precisely, I showed that it is possible to extract high-level environment information, such as detecting surface properties, directly and linearly from the state of the body. This was possible even while generating locomotion patterns in parallel. I do not continue this line of work here for two reasons. First, this line of work was already explored in a compliant robot by Fuyima Iida [81]. Secondly, sensing is inherently embedded into my closed loop control approach.

To sum up, the main goal of this chapter is three-fold. First, I show that the results of Hauser et al. are not merely a theoretical result and demonstrate that compliant robots indeed have real computational power which can be easily exploited using simple learning algorithms. Secondly, I will study the generation of cyclic motion patterns (similar to the patterns generated by Central Pattern Generators (CPGs)) as an example to demonstrate the available computational power. These rhythmic patters are then used to achieve locomotion. By using the morphology to generate CPG-like signals, the design of the controller can be drastically simplified. Indeed, integrating sensor data into CPGs is not an easy problem, and by integrating the body dynamics in the control structure, the robot intrinsically synchronizes to properties in its environment. Finally, by using tensegrity structures, I provide an implementation of the general principle of Physical Reservoir Computing that is very close to the pure mass-spring nets from Hauser et al.

The structure of this chapter is as follows. I first provide a brief overview of central pattern generators. I then introduce a number of learning rules for simulating CPG-like motor patterns with tensegrity structures. Next, I provide a set of example applications. I show that the gait can be modulated by changing the equilibrium length of a subset of springs, which can prove useful to train robots to adapt their gait depending on the terrain. I optimize gaits using an external controller and then learn the equivalent gait with morphological computation to show that the control can literally be outsourced to the body. In the same spirit, I show an example of the control of an end-effector. Finally I empirically show that the presented methods work over a large parameter space and in non-linear regions when the structures are driven far from their equilibrium state.

4.3 Central Pattern Generators

Central Pattern Generators (CPG) are neural circuits typically found in the spine of vertebrates that generate rhythmic activation patterns without

sensory feedback or higher level control input [82]. My prime goal is to show that a lot of computational power can be exploited in compliant structures. The more computations that can be outsourced to the body, the less effort one needs to put in the construction of CPGs (for robotics applications) and the less external computational power is needed.

Robotic systems are not often as compliant as the ones I study here and the available morphological computational power might be insufficient to allow for the desired behavior with a static linear feedback. I argue that one should however try to keep the body's dynamics as much (and as soon as possible) in the loop to be able to exploit the morphological computational power. Indeed, in the case of compliant tensegrity structures, one can go as far as leaving out the external CPG completely.

4.3.1 Matsuoka Oscillators

The non-linear Matsuoka oscillator is considered here as a model CPG [116, 117]. It is one of the most fundamental oscillator structures, based on a simple integrating neuron with fatigue. Synaptic fatigue is a form of negative feedback in a Neural Network. It regulates the activity of highly excited neurons. The dynamics of this oscillator are given by (dropping the time indices):

$$\dot{\boldsymbol{x}}_{osc} = \frac{-\boldsymbol{x}_{osc} + \boldsymbol{W}_{osc}\boldsymbol{y}_{osc} + \gamma - \iota\boldsymbol{v}_{osc}}{\tau_1} \tag{4.1}$$

$$\dot{\boldsymbol{v}}_{osc} = \frac{\boldsymbol{y}_{osc} - \boldsymbol{v}_{osc}}{\tau_2} \tag{4.2}$$

$$\boldsymbol{y}_{osc} = \max(\boldsymbol{x}_{osc}, \boldsymbol{0}), \tag{4.3}$$

where $\boldsymbol{x}$ represents the internal state of the neuron. Here $\boldsymbol{W}_{osc}$ is the matrix describing how the neurons are connected. It is typically sparse, as Matsuoka mostly analyzed small regular connection patterns. The positive semidefinite weight matrix $\boldsymbol{W}$ was constructed similar to the stress matrix of the tensegrity structure with the diagonal (self feedback) removed. More precisely:

$$\boldsymbol{W}_{osc} = \boldsymbol{C}^T \mathrm{diag}_v(\boldsymbol{h})\boldsymbol{C} - \mathrm{diag}_v(\mathrm{diag}_m(\boldsymbol{C}^T \mathrm{diag}_v(\boldsymbol{h})\boldsymbol{C})) \text{ with } \boldsymbol{h} \in [0,1]^{\not q} \tag{4.4}$$

where $\boldsymbol{C}$ is the connectivity matrix as defined in Section 2.1.3.2. As in Chapter 2, the diag_v operator transforms a column vector into a diagonal matrix. Similarly, the diag_m operator extracts the diagonal of a matrix into a column vector. Hence, the neurons are connected in the same way as the spring-cable assemblies connect the nodes of the tensegrity structures. The choice of this connection pattern was inspired by the symmetric structures

used in [116]. In fact, it is a common approach to reflect the physical structure in the oscillator connectivity [83]. Additionally, random connection patterns tend to generate chaotic trajectories.

The integrating neuron and the fatigue have time constants τ_1 and τ_2 respectively. The steady state firing rate of the neuron is determined by ι and γ is called the impulse rate of the tonic or slowly varying input [116]. These parameters are constant, i.e. the oscillator itself is not modulated. The parameters are: $\iota = 1$,$\tau_1 = 0.5$, $\tau_2 = 5$ and $\gamma = 1$. Figure 4.2 shows an example of CPG signals generated by the above procedure. There were a total of 12 dimensions (5 shown) and the connection pattern was taken from the tensegrity icosahedron (e.g. as shown in Figure 4.3).

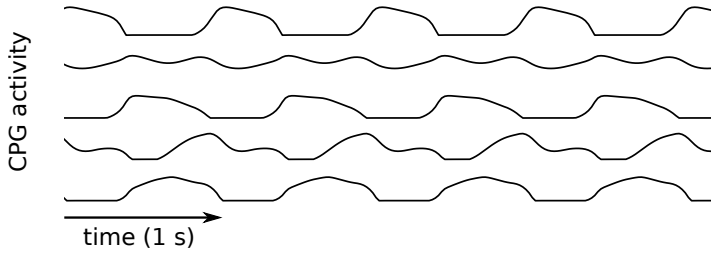


Figure 4.2: Sample Matsuoka oscillator signals. A linear combination of such signals is used as CPG signal for robot locomotion. There were a total of 12 dimensions (5 shown) in CPG of this example and the connection pattern is taken from the tensegrity icosahedron (Figure 4.3).

The vector $\boldsymbol{y}_{osc}$ contains the firing rate of the neurons and $\boldsymbol{v}_{osc}$ models the fatigue. The firing rates $\boldsymbol{y}_{osc}$ are the outputs of the oscillator and these signals are used to construct the target motor signals. The output signals $\boldsymbol{y}_{osc}$ were resampled such that the signals had the correct frequency for the experiment (normally 1 Hz). The desired output signals are random linear combinations of this p-dimensional signal $\boldsymbol{y}_{osc}$.

Based on the signals $\boldsymbol{y}_{osc}$ I construct target motor signals as a simple linear combination:

$$\boldsymbol{y}_{target} = \boldsymbol{W}_{target}\boldsymbol{y}_{osc}. \tag{4.5}$$

In practice, a constant bias variable is added to $\boldsymbol{y}_{osc}$. $\boldsymbol{W}_{target}$ is random (normally distributed values) in most of this work (i.e. I assume the desired CPG signal to be known), except in Section 4.5.2, in which I optimize the CPG signal. The fundamental difference is that $\boldsymbol{y}_{osc}$ is *generated* by a random oscillator. These signals will take values between 0 and 1, while the motor signals will need a correct offset and amplitude. This is solved by $\boldsymbol{W}_{target}$,

which combines the signals from $\boldsymbol{y}_{osc}$ into meaningful motor commands.

I chose the Matsuoka type oscillator because of its simple structure which can be chosen to be similar to the connection pattern of the tensegrity structure itself. While I did not explore this path further, one interesting extension is to integrate the morphological communication idea from Rieffel et al. [144]. In that work, Rieffel et al. use controllers based on spiking Neural Networks that interact only through body dynamics.

4.4 Physical Reservoir Computing

Reservoir Computing was introduced in Chapter 3. The core idea of RC was originally applied only to Neural Networks, but has since been extended to other non-linear dynamical systems, leading to what is called Physical Reservoir Computing (PRC). There have been demonstrations of the Reservoir Computing approach applied to different domains such as photonics [181] and more abstractly electronics [6]. All these implementations share the common idea that a system with complex and rich dynamics is perturbed externally but left untouched otherwise, and a simple readout mechanism is trained to perform the desired computational task. While the idea of PRC originated in the context of Neural Networks, recent theoretical results have extended the applicability of this computational framework [32].

Recall the most common implementation of Reservoir Computing. The discrete time network dynamics are given by:

$$\boldsymbol{x}[d+1] = \tanh(\boldsymbol{W}_{res}\boldsymbol{x}[d] + \boldsymbol{W}_{in}\boldsymbol{u}[d+1] + \boldsymbol{w}_{bias}) \tag{4.6}$$

$$\boldsymbol{y}[d+1] = \boldsymbol{W}_{out}\boldsymbol{x}[d+1]. \tag{4.7}$$

RC can be used to implement functions that do not necessarily have the fading memory property by feeding the output back into the system:

$$\boldsymbol{x}[d+1] = \tanh(\boldsymbol{W}_{res}\boldsymbol{x}[d] + \boldsymbol{W}_{in}\boldsymbol{u}[d+1] + \boldsymbol{w}_{bias} + \boldsymbol{W}_{fb}\boldsymbol{y}[d]) \tag{4.8}$$

$$\boldsymbol{y}[d+1] = \boldsymbol{W}_{out}\boldsymbol{x}[d+1]. \tag{4.9}$$

The feedback weights $\boldsymbol{W}_{fb}$ are typically chosen randomly and, again, only $\boldsymbol{W}_{out}$ is trained. This type of system can be used to autonomously generate desired signals. As persistent oscillations are needed to emulate CPG signals, this last set of equations forms the foundation of the Physical Reservoir Computing approach I present here.

Figure 4.3 shows how tensegrity structures can be used for Physical Reservoir Computing. The equilibrium length of a subset of the spring-cable

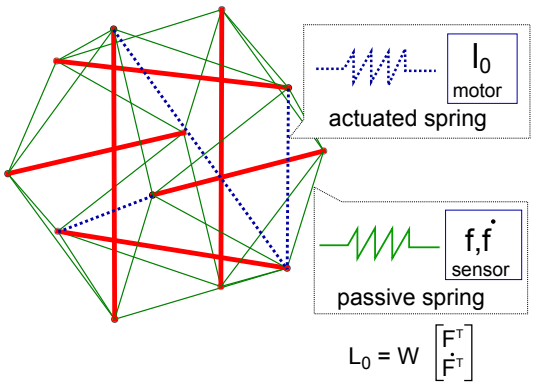


Figure 4.3: Overview of physical Reservoir Computing with compliant tensegrity structures. The thin green lines are passive spring-cable assemblies with a sensor measuring the force and its derivative on the spring. The thick red lines are non-compliant bars. The dotted lines are actuated spring-cable assemblies. The (new) equilibrium length of the actuated spring-cable assemblies is computed as a linear combination of the sensor values.

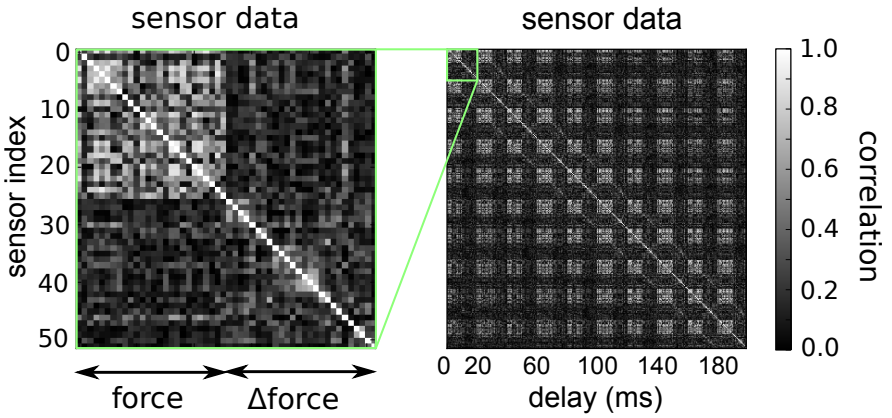


Figure 4.4: Absolute values of the correlation coefficients between the different state/sensor values as defined in Eq. 4.10. The left graph shows the correlation between the state variables at the current time step ($[\boldsymbol{f}^T(t)\ \dot{\boldsymbol{f}}^T(t)]^T$). The image on the right illustrates the correlation between different time steps ($t\ t-\Delta \ldots t-k\Delta$).

assemblies is used as feedback/input to the system. Differently from [68], I use only linear spring-cable assemblies (Eq. 2.19). In my experiments, the non-linearities arising from the changing geometrical configuration and inertia are sufficient for good performance. The time evolution of the tension on each spring-cable assembly and its derivative are used to define the state $\boldsymbol{x}$ of the dynamical system (cf. Eq. 4.8):

$$\boldsymbol{x}(t) = \text{vec} \begin{pmatrix} \boldsymbol{f}(t) & \dot{\boldsymbol{f}}(t) \\ \boldsymbol{f}(t-\Delta) & \dot{\boldsymbol{f}}(t-\Delta) \\ \cdots & \cdots \\ \boldsymbol{f}(t-k\Delta) & \dot{\boldsymbol{f}}(t-k\Delta) \end{pmatrix}, \tag{4.10}$$

where $\boldsymbol{f}(t)$ are the spring forces measured at time t. The parameter $\Delta = 20\,\text{ms}$ is the controller time step and k is the number of delay steps used. The vec operator concatenates all columns of a matrix into a vector. For the tensegrity icosahedron simulations, $k = 9$ was used (maximum delay of 200 ms) and $k = 3$ for the snake robots. The main reason for this is that this allows the feedback to filter out noise due to ground collisions to some degree (by averaging over the delayed inputs). I have also performed a number of simulations with $k = 0$ (see also [23]) to verify that the system does not depend fundamentally on this delayed sensor information. This was indeed not the case, but the ground collisions tend to render figures such as Figure 4.7 less intelligible. One can see from Figure 4.4 that the time delayed sensor information is indeed highly correlated. According to 2.19, each element of $\boldsymbol{f}(t)$ can be written as:

$$f_{i,j}(t) = \max(k_{i,j}(\|\boldsymbol{n}_i(t) - \boldsymbol{n}_j(t)\| - l_{0,(i,j)}(t)), 0), \tag{4.11}$$

where (i, j) indicates the spring-cable assembly connecting end caps i and j

I have explicitly used the time index for the rest lengths $l_{0,(i,j)}(t)$, because a subset of the spring-cable assemblies is actuated and thus has a varying rest length. The subset of passive, fixed rest length spring-cable assemblies are now denoted by the subscript *pas* and the subset of actuated spring-cable assemblies by the subscript *act*. $\boldsymbol{l}_{0,pas}$ is a constant vector defined by the equilibrium state of the structure. $\boldsymbol{l}_{0,act}(t)$ is time-varying and is given by:

$$\boldsymbol{l}_{0,act}(t) = l_{max} g(\boldsymbol{y}(t)) + \boldsymbol{l}_{0,act}(0). \tag{4.12}$$

l_{max} is the maximum change in rest length of the spring-cable assemblies (w.r.t. the initial lengths in $\boldsymbol{l}_{0,act}(0)$) allowed by the actuators. For this to work one must have $g : \mathbb{R}^a \to [-1, 1]^a$, with a the number of actuated spring-cable assemblies. Now $\boldsymbol{y}(t)$ will in general be a linear combination of

$\boldsymbol{x}(t)$ and a constant bias input:

$$\boldsymbol{y}(t) = \boldsymbol{W}\boldsymbol{x}(t). \tag{4.13}$$

The goal of most of the algorithms studied here is to find the matrix $\boldsymbol{W}$.

TERMINOLOGY It might appear strange to define the state of the dynamical system (Eq. 4.10) as a set of measurements, while the state of the robot evolves according to the equation of motion presented in Section 2.8. In Reservoir Computing for Neural Networks, it is common to observe the full state of the system. However, this is not a requirement and one is free to observe a subset of the neurons or to apply a (non-)linear, memoryless transformation to the output of the neurons. In this case the dynamical system (the black box) consists of the Reservoir and the transformation. For Echo State Networks, such a transformation preserves the Echo State Property. Assume that $\boldsymbol{x}[d]$ is the vector containing the state of the neurons in an ESN at time step d as defined in Eq. 3.1. Then a non-linear, memoryless transformation is simply a function $f : \mathbb{R}^n \to \mathbb{R}^{n'}$ that maps $\boldsymbol{x}[d] \in \mathbb{R}^n$ onto $\boldsymbol{x}'[d] = f(\boldsymbol{x}[d]) = f(\tanh(\boldsymbol{W}_{res}\boldsymbol{x}[d] + \boldsymbol{W}_{in}\boldsymbol{u}[d+1] + \boldsymbol{w}_{bias})) \in \mathbb{R}^{n'}$. The output is now based on $\boldsymbol{x}'[d]$ instead of $\boldsymbol{x}[d]$, which gives $\boldsymbol{y}[d] = \boldsymbol{W}_{out}\boldsymbol{x}'[d]$. The same applies to Physical Reservoir Computing for tensegrity structures. The computational black box is the robot in its environment, combined with the measurement equipement. Section 2.8.1 explained how the dynamics of the robot are computed and Eq. 4.10 defines the non-linear, memoryless transformation (f). As the internal (full) state of the robot ($\boldsymbol{n}$ and $\dot{\boldsymbol{n}}$) is not used or observed at any point in the learning algorithms, I directly defined the non-linear transformation of the internal state as *the state* of the system $\boldsymbol{x}$. This prevents cluttering the notation by using $\boldsymbol{x}'$ everywhere and makes the equations consistent with the standard RC results.

For the experiments presented in this chapter, I have used $g(\boldsymbol{y}(t)) = \tanh(x)$. It is important to justify the use of a non-linear function[1], as it can provide computational power (as in the RC approach). Therefore, I have also tested the setup with both a hard limit: $g(\boldsymbol{y}(t)) = \min(\max(\boldsymbol{y}, -1), 1)$ and with the identity function (no limit). Both cases yielded quantitatively similar results to those presented in the experimental Section 4.5. The identity function was discarded because it does not guarantee boundedness of the feedback and spurious sensor data can make the structures collapse. In practice I have noticed that with the identity function, the structure would operate correctly for e.g. 30 s after training and then collapse because

[1] Note however that the non-linear function is applied after the linear feedback and could thus be considered part of the motor driver.

of an extreme sensor value during a ground collision. However, a physical implementation will always be limited by the maximum offset of the motor to ensure safe operation, which validates the use of $g(\boldsymbol{y}(t))$.

To conclude this section, I note that for the PRC setup $\boldsymbol{x}[k]$ from Eq. 4.8 is replaced by sensor measurements of the tensegrity structure and the output $\boldsymbol{y}(t)$ is a linear combination of these values. Differently from the classic RC or ESN implementations, the feedback enters the system through a physical modification of the system by modifying the equilibrium lengths of a set of actuated springs. The system itself operates in continuous time, but the spring-cable rest lengths are updated at discrete time steps and smoothly interpolated in between the control time steps — i.e. the spring-cable rest lengths are the set points of a constant velocity motor controller.

4.5 Experiments

The experimental section of this chapter consists of three parts. First, I introduce a set of algorithms to train tensegrity structures to produce rhythmic patterns. Next, I discuss possible applications for locomotion. I end with a comparison of different parameter combinations to study the importance of non-linearities in the system.

All experiments in this chapter and the next one were performed using a simulator based on the Euler-Lagrange formulation introduced in Section 2.8.1.

4.5.1 Outsourcing Motor Pattern Generation

4.5.1.1 Recursive Least-Squares Approach

The first training algorithm I will consider is based on the Recursive Least-Squares (RLS) algorithm [96]. When the same samples are presented to the RLS algorithm, it will compute the same weights as batch linear regression (which is used by [68]). As such, it is in the first place a method for the efficient computation of the least-squares solution for a sequential data problem. The advantage of RLS is that it allows a gradual transition from a completely teacher forced structure (the desired signals are fed into the system) to a system generating its own control signals and to restart training if needed.

I now describe the RLS algorithm in detail. During training the feedback signal is a mixture of the target feedback signal and the actual feedback output signal which is being trained (see Figure 4.5). The influence of the target signal on the feedback signal is gradually reduced until the output

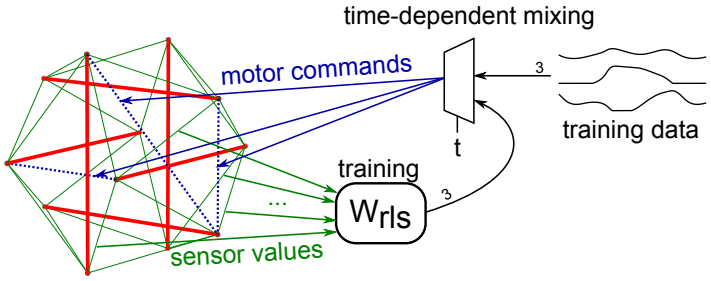


Figure 4.5: Overview of the Recursive Least-Squares (RLS) approach for motor signal generation tasks. Sensor measurements are linearly combined using the matrix $\boldsymbol{W}_{rls}$, which is updated using the RLS algorithm. The objective of this linear combination is to approximate the desired motor signals (training data). During the training phase of the algorithm, the effective motor commands are a mixture of the training signals and the linear combination of sensor measurements. Once the system is trained, the motor commands only depend on the sensor measurements.

signal is only given by the trained signal:

$$\alpha_{rls} = \frac{1}{1+\tau_{rls}t} \text{if } t < \text{training time else } 0 \tag{4.14}$$

$$y_i(t) = \alpha_{rls} y_{i,target}(t) + (1-\alpha_{rls}) \sum_j W_{(i,j),rls}(t) x_j(t). \tag{4.15}$$

At each time step the weights $\boldsymbol{W}_{rls}$ are updated using the RLS equations:

$$\boldsymbol{L}_{rls}(t) = \frac{\boldsymbol{P}_{rls}(t)\boldsymbol{x}(t)}{1+\boldsymbol{x}(t)\boldsymbol{P}_{rls}(t)\boldsymbol{x}(t)} \tag{4.16}$$

$$\boldsymbol{P}_{rls}(t+\Delta t) = \boldsymbol{P}_{rls}(t) - \frac{\boldsymbol{P}_{rls}(t)\boldsymbol{x}(t)\boldsymbol{x}^T(t)\boldsymbol{P}_{rls}(t)}{1+\boldsymbol{x}^T(t)\boldsymbol{P}_{rls}(t)\boldsymbol{x}(t)} \tag{4.17}$$

$$\boldsymbol{e}_{rls}(t) = \boldsymbol{y}_{target}(t) - \boldsymbol{W}_{rls}(t-\Delta t)\boldsymbol{x}(t) \tag{4.18}$$

$$\boldsymbol{W}_{rls}(t) = \boldsymbol{W}_{rls}(t-\Delta t) + \boldsymbol{L}_{rls}(t)\boldsymbol{e}_{rls}(t). \tag{4.19}$$

There is only a single parameter: the teacher forcing decay time constant τ_{rls}. The precision matrix $\boldsymbol{P}_{rls}$ was initialized using the identity matrix. The dimensions of this square matrix are $2k\dim(f) \times 2k\dim(f)$, where $\dim(f)$ indicates the number of spring-cable force transducers. I note the difference w.r.t. FORCE learning [168] in which initially chaotic systems are used. The main reason for this is that tensegrity structures are inherently damped and

to create chaos, one would need a feedback loop to drive the system. From a practical point of view this might be inefficient or even dangerous, as one would need additional actuators which are only used to keep the system active. In this sense, the RLS approach used here is closer to the teacher forcing approach ([194] and Section 3.3.2.1). In this approach the desired output is fed into the system during training and the state of the system $\boldsymbol{x}(t)$ is stored. Then, regression is used to approximate the desired output from the system state. Finally, during testing the approximate output based on the system state is fed back into the system and the system will generate the desired patterns autonomously. The testing phase is also called free run, as the system is no longer forced by the external input.

The gradual change from teacher forcing to free run used in this work allows the structure to take over the control in a smooth way and to restart learning in a straightforward way. I noticed that the RLS algorithm becomes unstable if learning continues with α_{rls} too low (i.e. $\alpha_{rls} \lesssim 0.03$). So I simply switch to free run when α_{rls} drops below the threshold. The most likely explanation for the instability is that this is caused by the phase drift between the output and the teacher signal when the system is unforced. When α_{rls} is low, the feedback loop is trying to improve upon itself. This fails when the target and observed signals go out of phase. To continue learning at this point to build a system that can adapt to small changes over time or to fine-tune the performance, an interesting approach might be to use the reward based techniques presented in the next chapter.

A demonstration of the RLS approach is shown in Figure 4.6 and Figure 4.7. In this case 6 actuators were used (i.e. 6 output dimensions). One can observe that the output signal gets out of phase with respect to the target signal due to collisions with the ground. The ground collisions and the control time step (20 ms) generate noise in the system. Due to these effects, additional regularization is not needed.

Figure 4.7 shows a phase portrait of two output signals from Figure 4.6. The output signals stay in phase w.r.t. each other, which is important for locomotion. The RLS rule can capture the complex details of the target signals through the non-linearities provided by the structure.

In my opinion, there are two disadvantages associated with the RLS approach. First, one needs to update the precision matrix of the observed variables, which does not scale well. The second and more fundamental disadvantage is the dependence on explicit knowledge of the target function, because one needs to know the difference (error) between the optimal motor signal and the current signal generated by the RLS algorithm. In a practical setting the target signals are not always known and often only some global performance measure is available. These two issues are the objects of study

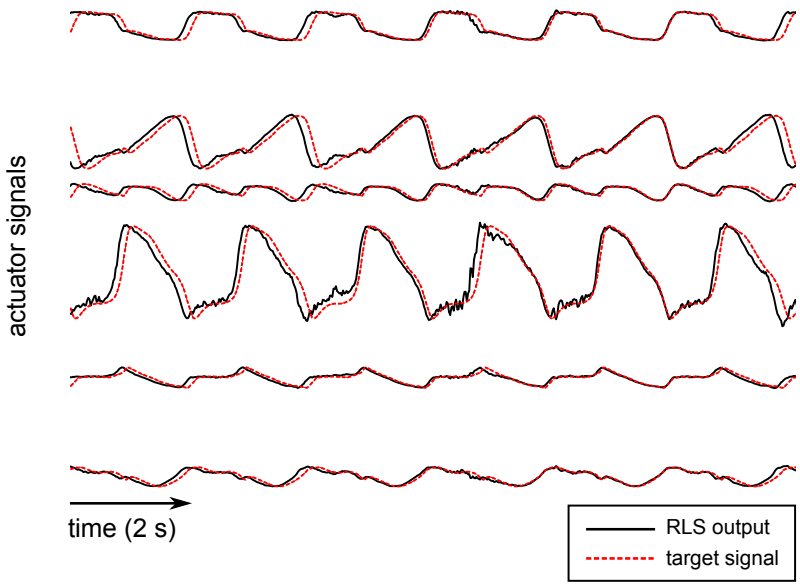


Figure 4.6: Demonstration of the RLS algorithm. 6 outputs were trained for $250\,\mathrm{s}$, followed by $150\,\mathrm{s}$ of testing. Shown is the output at the end of the testing (free run) phase. The dashed line is the target signal, which is generated as in Eq. 4.9. The solid line is the output signal, which is sent to the actuators. Notice that the phase of the target signal is not matched, but that the relative phase of the outputs is fixed. This effect is due to the tensegrity structure synchronizing to ground collisions.

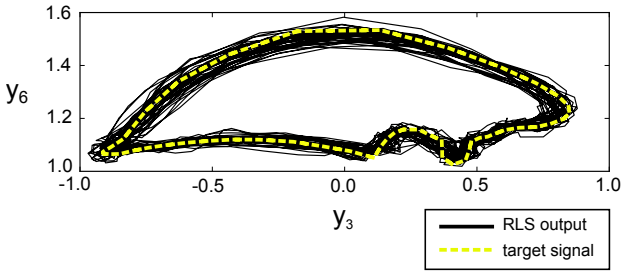


Figure 4.7: Demonstration of the RLS algorithm as in Figure 4.6. Shown are two output dimensions out of the total of 6 during $20\,\mathrm{s}$ of testing. The light line is the target signal, the dark line the output signal. Clearly the system has learned the attractor robustly. The small perturbations are mostly due to ground collisions.

of the next chapter.

4.5.1.2 Gradient Descent Approach

As a means to overcome a disadvantage of the RLS algorithm, namely its complexity vis-à-vis its biological plausibility, I have used stochastic gradient descent on the error signal. The following equation, which can replace the update of $\boldsymbol{W}_{rls}$, is trivially obtained by differentiating the quadratic error at a time step:

$$\boldsymbol{W}_{gd}(t) = \boldsymbol{W}_{gd}(t - \Delta t) - \alpha_{gd}\boldsymbol{e}_{rls}(t)\boldsymbol{x}^T(t). \tag{4.20}$$

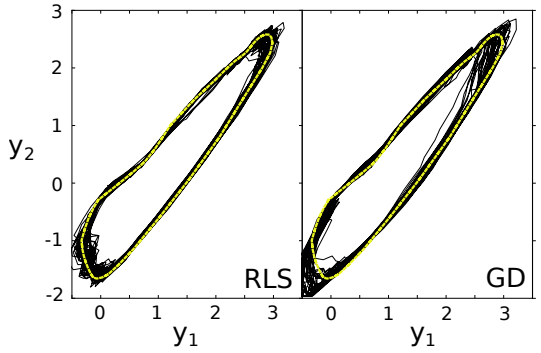


Figure 4.8: A comparison of the performance of the RLS approach compared to the less computationally demanding GD rule. The objective is to emulate the desired two-dimensional attractor (yellow dashed line). While both the GD and RLS rules capture the phase and global shape of the signals, it can be seen that only RLS is capable of reproducing the details by exploiting the covariance structure of the state variables.

Because the learning rate α_{gd} has to be chosen small enough to prevent instability, the GD rule tends to converge more slowly than the RLS rule. Figure 4.8 shows an example of the RLS and GD rules applied to the same problem. A two-dimensional feedback loop is trained on a tensegrity robot to emulate a desired attractor. The GD rule captures less details of the target signals as it cannot exploit the covariance structure of the state variables.

4.5.2 Applications

In this section I present a set of practical applications of morphological computation in tensegrity structures. I first show that the structure can modulate its gait patterns when the equilibrium length of a few spring-cable

assemblies is changed. Next I look at gait optimization. I optimize the gait pattern with an external controller and then outsource the resulting gait to a static, linear feedback controller. Finally, I discuss a basic end-effector control application.

4.5.2.1 Modulating Motor Patterns

An important question is whether the trained tensegrity structures can be controlled by adapting their gait to different configurations of the structure or e.g. the slope of a hill? To test this I have added a single input signal to the system. This signal was fed into the tensegrity structure by modifying the equilibrium length of 2 actuated spring-cable assemblies. The target motor patterns had to be modulated by the structure to linearly interpolate between two CPG patterns with the same frequency.

I have again used the tensegrity icosahedron to show that such modulations are possible even in relatively small systems. Figure 4.9 shows a result from a run of the algorithm. I have trained the system for only 400 s. At each time step, the system switches to another random input (i.e rest length changed) with probability 0.005. Therefore the time between gait changes is variable. This also shows the robustness of the system, because accidental fast switches between input states disturb the system.

The experiment demonstrates a natural way to incorporate input or environmental feedback for controllers based on the Physical Reservoir Computing principle. One of the main advantages of this approach is that the input or feedback is directly available to the feedback loop. No special treatment of the inputs is required nor does the feedback loop increase in complexity. More general modulations can be implemented based on a variable feedback loop instead of a constant weight matrix, but the learning procedure becomes more involved [92, 93].

4.5.2.2 Gait Optimization

Gait optimization in robots is a complex problem, because small changes to e.g. the relative phase of two limbs or the duration of support phases can result in different locomotion patterns or failure in legged robots (see e.g. [3] for reviews of animal gait patterns). For example, an animal typically positions its legs during locomotion to reduce the magnitude of joint moments and as such the required muscle forces [10].

Optimizing all aspects of gait properties is beyond the scope of this chapter. I therefore assume the robot's configuration as well as the CPG frequency to be known. Figure 4.10 gives an overview of the applied training procedure. My goal is to optimize the weights of the matrix $\boldsymbol{W}_{target}$ for

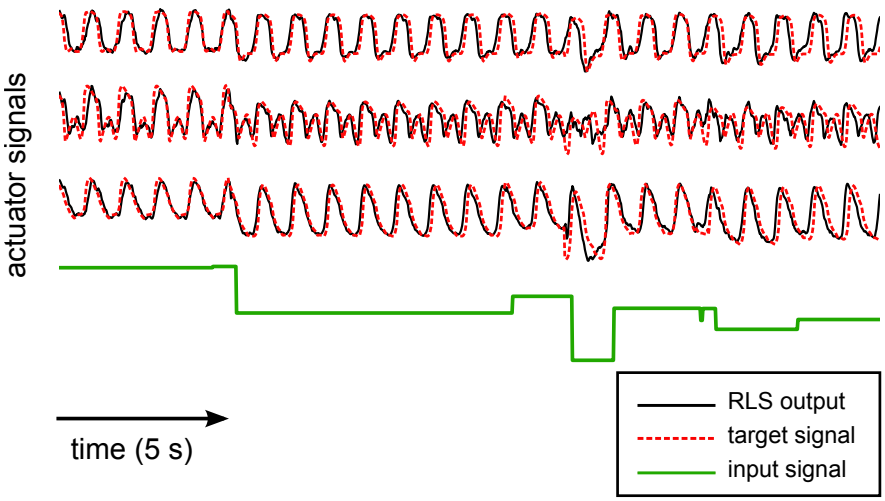


Figure 4.9: Modulating gait patterns through morphological computation. A single input signal was applied to the system by modifying the equilibrium length of two springs. The structure had to linearly interpolate between two CPG signals (3-dimensional) with the same fundamental frequency. The system was trained for $400\,\mathrm{s}$ with random inputs. The phase is not perfectly matched, because fast input changes disturb the system. Note that both the signal offset and shape are changed. A random tensegrity with 6 bars was used. At approximately two thirds of the sample, there is an apparent glitch in the lowest signal. The RLS output signal mismatches the target because of a sudden large drop of the input signal. The reason for this is that RLS output is based on the state of the physical system and thus intrinsically smooth. As such the RLS feedback will naturally interpolate between target signals.

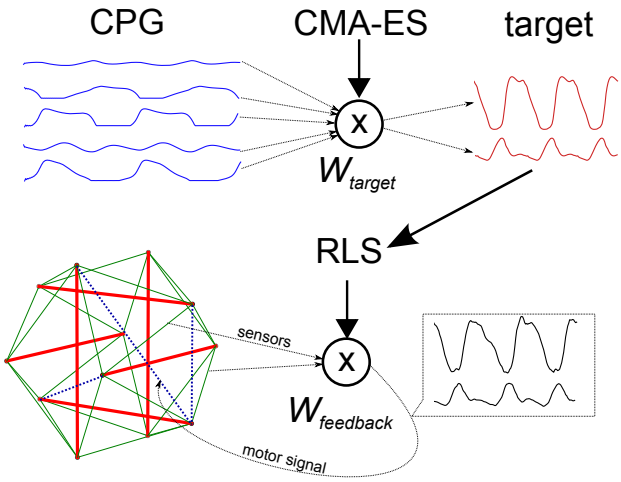


Figure 4.10: Overview of the training principle for gait optimization. The CMA-ES algorithm optimizes the CPG pattern and then RLS is applied to train a feedback to approximate this target pattern using morphological computation. If the robot has rich enough dynamics, the same gait will be obtained using the static feedback loop.

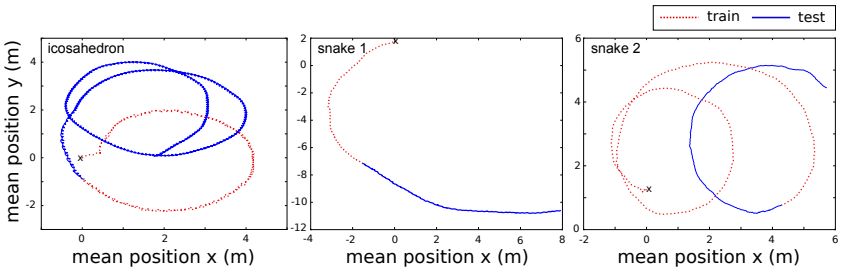


Figure 4.11: Robot trajectories (center of mass) for three runs of the algorithm on different structures (a tensegrity icosahedron and two snake-like structures). In red (dotted) the trajectory during training, in blue the trajectory during testing. Morphological computation is powerful enough to maintain the same gait which was found by optimizing the external CPG using CMA-ES.

a given basic CPG. I will then outsource the optimal gait to the structure with the RLS algorithm. The obtained gaits using only morphological computation match those observed during training. Therefore the structure can approximate the required motor patterns well enough to enable locomotion.

To optimize $\boldsymbol{W}_{target}$ I use the well known CMA-ES (Covariance Matrix Adaptation Evolution Strategy) algorithm [66]. The choice for this algorithm stems from the fact that no explicit teacher is required and that the method is almost parameter-free. Furthermore it has shown excellent performance on a variety of related problems [42, 161]. The fitness function used is simply the distance travelled by the center of mass of the tensegrity. Because of the compliance of the tensegrity structures, I did not need to include penalties for e.g. falling. Later (Sections 7.4.2 and 7.4.3) coevolutionary algorithms are used when distributed control is emphasized [132].

Figure 4.11 shows the trajectory of the center of mass of three different tensegrity structures. On the left, the tensegrity icosahedron with a number of additional springs. Remarkably, the gait was obtained after only 10 iterations of the CMA-ES algorithm. The population size was 50 and there were 4 actuators. The gait was evaluated during 30 s. This means that only 4 h of exploration time would be necessary to obtain this locomotion pattern on a hardware platform.

The two other plots are from snake-like tensegrity structures which were constructed by stacking tensegrity prisms. Figure 4.12 shows the center structure in action, while the third result was a snake-like structure with 5 segments.

To show that the same gait is indeed maintained, I have compared (Figure 4.13) the (vertical) ground reaction forces on the endpoints during training and testing of the large snake-like tensegrity from Figure 4.12. This system has 20 actuators in total. Due to the complexity of the structure, there is some variation in the ground reaction forces, but there is a clear pattern. The relative phase between of the ground contacts is identical during training and testing. The training sample is taken from the beginning of the training (almost completely teacher forced), while the testing sample is from the end of testing (free run).

4.5.2.3 End-Effector Control

To end this applications section, I show that the same technique can also be used to control an end-effector. The objective is now to control the position of the endpoint of a bar with respect to two other endpoints. For this I measure the length along two springs connecting the endpoint of the bar with the endpoints of the other bar. This is similar to controlling the position of the wrist with respect to the shoulder.

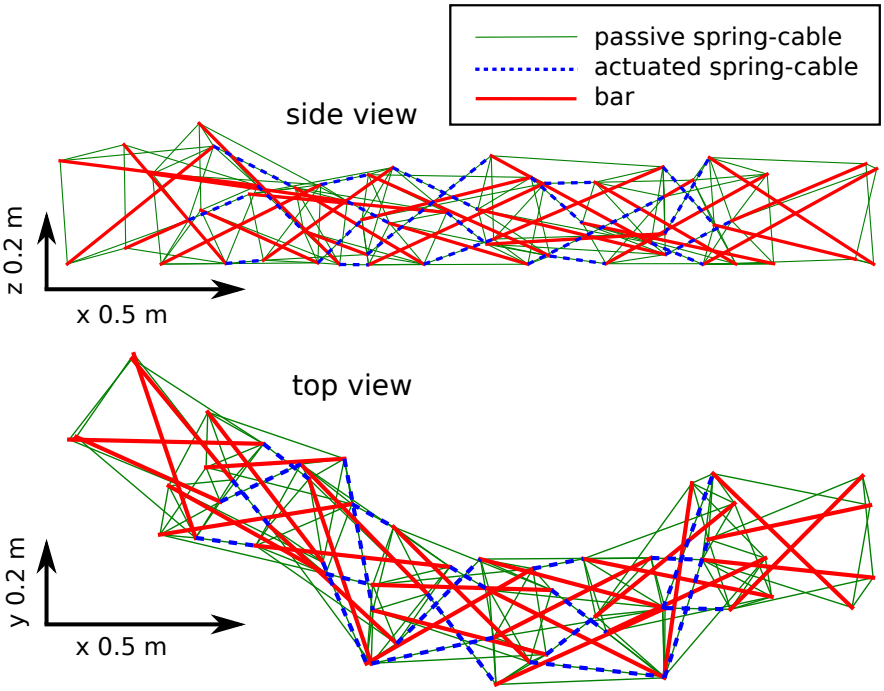


Figure 4.12: A complex snake-like structure controlled using morphological computation. Shown is the structure during locomotion with one of the found gait patterns. There are large shape deformations from the equilibrium state and a total of 20 actuators.

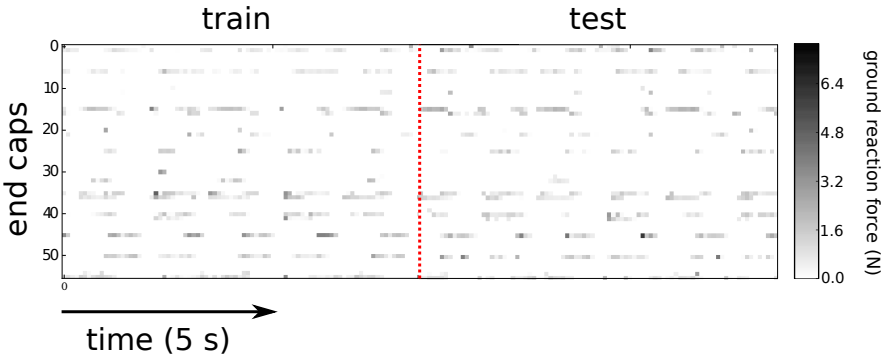


Figure 4.13: Vertical ground reaction forces during training (left) and testing (right) for the structure from Figure 4.12. Note the variation in the signals. The same gait is maintained during testing.

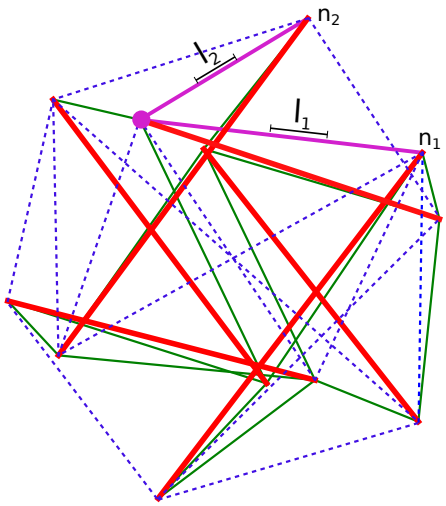


Figure 4.14: End-effector control in a tensegrity robot. In this example I seek to control the lengths l_1 and l_2 of two spring-cable assemblies, i.e. the relative position of the end cap of a bar (large dot) w.r.t two other end caps (n_1, n_2) of the structure.

I do not assume a model of the system to be known and use CMA-ES to optimize $\boldsymbol{W}_{target}$. The CPG has the same frequency as the target movement. Because the CPG only has a limited number of basis signals and the structure is underactuated, it is to be assumed that the target trajectory cannot be perfectly matched. In this example I have used a 30-dimensional CPG, based on a connection pattern from a stacked tensegrity prism.

To compute the fitness, I have simulated the system for 100 s and computed the MSE over the last 80 s. The system was in free fall — gravitational and ground contact effects were not simulated — and the springs along which I have measured the position were not actuated. A tensegrity icosahedron with a total of 13 actuators (Figure 4.14) was used for this example (24 DOF, because the rigid body movements are ignored).

The result is shown in Figure 4.15. While the target trajectory cannot be perfectly matched due to underactuation and the limitations of the CPG, the result is very encouraging. The end-effector is part of the computational system itself and the springs along which the position is measured also influence the system. Only 75 s of training with RLS was needed to transfer the control from the external CPG to morphological computation.

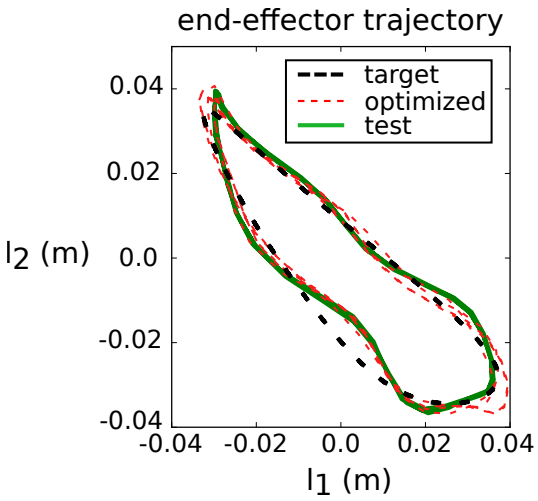


Figure 4.15: Trajectory of the end effector during testing after 75 s of training.

4.5.3 The Importance of Complex Dynamics

To complete this experimental section, I wish to demonstrate that the non-linearities can indeed improve the computational power of the system. Such a statement is of course task-dependent, e.g. to generate sine waves, it is obviously not advantageous to have non-linear dynamics in the system. I will again consider the generation of CPG-like signals based on the Matsuoka-type non-linear oscillator in combination with the tensegrity icosahedron.

Sultan et al. [166] indicated that the linearized dynamics of tensegrity deviate more from the non-linear dynamics of the system at higher (generalized) velocities and lower pretension. While typically, one would restrict the velocities and deformations of the system such that the linearized dynamics are a good model of the system, the PRC technique benefits from the opposite.

Many parameters of the structure can be tuned, and optimizing the configuration of the structure itself is a daunting task. In this section I only consider the importance of two parameters, the oscillator frequency and the maximal change of the actuator equilibrium length. The physically plausible regions of operations for both parameters are defined by the capabilities of the targeted hardware platforms. The objective is to study whether within these regions of operation, there are significant changes of computational performance.

The task is again the simulation of 12-dimensional random Matsuoka-type non-linear oscillators. The tensegrity icosahedron with a random number

of actuators is used, varying from 4 to 8 motors. I have swept the frequency in steps of 0.1 Hz from 0.1 Hz to 3 Hz. The maximum spring-cable assembly rest length offset (l_{max}) was varied in steps of 3.5 cm from 5.5 cm to 37 cm. These approximately match the capabilities of the ReCTeR robot which I introduce in Chapter 6. For each tuple (frequency, distance), I have performed 50 trials, for a total of 15×10^3 trials. I computed the normalized mean squared error, defined as:

$$NMSE = \frac{(\boldsymbol{x} - \boldsymbol{y})^T(\boldsymbol{x} - \boldsymbol{y})}{N\sigma(\boldsymbol{y})}, \tag{4.21}$$

with N the number of samples, $\boldsymbol{x}$ the vectorized output and $\boldsymbol{y}$ the vectorized target signal. For each set of 50 trials, only the 30 best are kept to prevent failures (e.g. collapsing) from influencing the results. The results are shown in Figure 4.16.

So what can be learned from this? First, it can be seen that for the task at hand, it is advantageous to work in a non-linear region by increasing the frequency of the oscillator or the maximum spring equilibrium offset. It is important to note that, although the frequency is a determining factor, the technique is not constrained to the natural frequency of the system. There is a region of frequencies with similar performance. One might consider the bottom right region of operation, with only very small amplitudes. The practical use of this region is however limited, as the movement of the robot will be very limited.

On the other hand, going beyond the 30 cm range, often causes instability (collapsing) and colliding bars. In practice, the performance will be restricted by a diagonal line going down from near the top left to the bottom right (approximately represented by the blue line in Figure 4.16), because of practical limitations such as motor output power. So within this region, better performance can be obtained by increasing the frequency or the maximum spring equilibrium offset.

Interestingly, for the lower frequency range (which might be interesting for energy efficiency reasons) it is advantageous to increase the maximum offset. Larger deformations of the structure — during which even the kinematics are highly non-linear as shown in Figure 2.8 — cause the error to decrease.

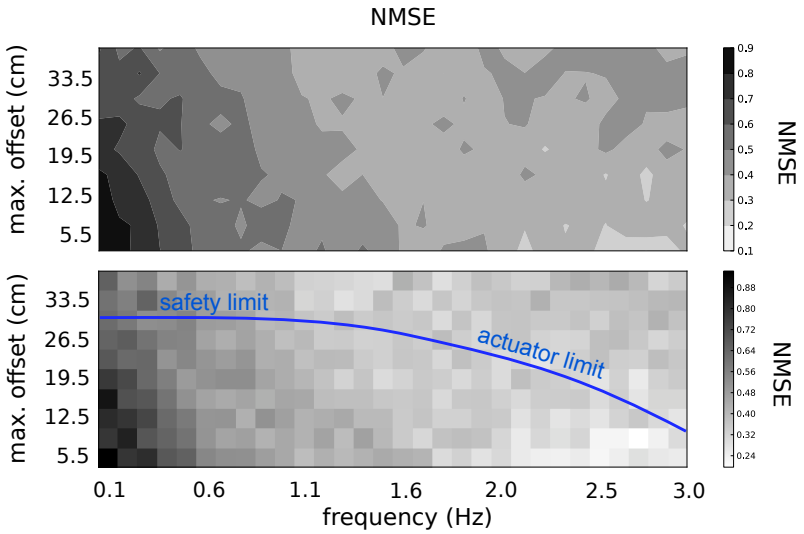


Figure 4.16: Exploiting non-linearity. Plots of the normalized mean squared error of the first $10\,\mathrm{s}$ of testing after training with RLS as a function of the oscillator frequency and the maximum spring equilibrium offset. Top: contour plot showing the different regions. Bottom: result for each combination (frequency, max. offset). The frequency is swept from $0.1\,\mathrm{Hz}$ to $3\,\mathrm{Hz}$ in steps of $0.1\,\mathrm{Hz}$ and the distance from $5.5\,\mathrm{cm}$ to $37\,\mathrm{cm}$ in steps of $3.5\,\mathrm{cm}$. All tests are performed on the tensegrity icosahedron with a random number of actuated spring-cable assemblies (between 4 and 8 motors). For each (frequency,distance) tuple, 50 trials were performed (15×10^3 total), of which the 30 best were retained to reduce the influence of marginal cases. The target was a linear combination of random Matsuoka-type oscillators (12-dimensional). For the task at hand, the system clearly benefits from increasing the frequency of the oscillator and the maximum offset. Very good (computational) results are obtained for a region (bottom right) with only small offset. This region might however not be suited for locomotion applications (limited shape changes). The line in the bottom graph shows the approximate safety (significant risk of a collapsing structure) and actuator limits (based on data from Chapter 6).

4.6 Conclusion

In this chapter, I have introduced an extreme form of embodiment, allowing for so-called Physical Reservoir Computing in a very pronounced sense. This principle was demonstrated by using highly dynamic and actuated tensegrity robots. As the dynamics of these systems provide exploitable computational resources, it becomes possible to use simple learning rules to train complex locomotion patterns or desired end-effector trajectories.

This provides a number of advantages from a robotics standpoint: The control complexity can be highly reduced and the learned control law is robust to perturbations and can easily synchronize to external inputs. The next chapter will extend these results to allow for reward-based learning. In this case, no explicit knowledge of the desired signals is needed and learning can be based solely on observations of the system's (global) behavior.

From a conceptual point of view, the conclusions are more profound. By demonstrating that dynamic "bodies" only require extremely simple "brains" to implement computations, I have effectively opened up a whole spectrum of potential trade-offs between brain-based computation and body-based computation. The powerful computational results from the field of Reservoir Computing [32, 67, 94, 113] can then be used to actually quantify and reason about the computations implemented by the physical dynamical system.

To conclude this chapter it is important to reflect on the meaning of the word "computation" in the context of a physical system. Computation can be defined as the process of transforming input symbols into output symbols, where the word symbol is to interpreted in a broad sense. Symbols are objects (data streams, characters, sensor values...) to which we — as users — attach a meaning. In this sense, physical systems, such as tensegrity robots, perform computations if we define the symbols and their meaning. The tensegrity robots in this chapter convert input data streams through their dynamics into sensor data and thus compute, if we define the meaning of the inputs and outputs. Nevertheless, the actual goal for robotics is to obtain optimal behavior of a robotic system and not to optimize computational performance. The key insight is that Physical Reservoir Computing borrows computational tools and methods from the Neural Network and machine learning fields and interprets control problems as computational tasks.

5
Reward Modulated Hebbian Plasticity

Hebbian theory has been around for over half a century [69], but it still sparks the interest of many researchers today.

First of all, the basic Hebbian plasticity rule is simple and can be efficiently implemented on various platforms in hardware and software. This simplicity also allows for detailed analyses. Secondly, small changes to the basic correlation learning rule result in various well known algorithms, such as principal [130, 150] or independent component [28, 79] extractor networks. Finally, the rule is biologically plausible as are some of its variations [109, 119].

The previous chapter has demonstrated that compliant structures can be used as a computational resource. The feedback weights were learned by applying online learning rules to approximate the target motor signals. However, genetic algorithms were necessary to find those motor signals in the first place.

I now show that simple Hebbian-like learning rules are capable of directly learning controllers for both Recurrent Neural Networks and compliant robots. First, this demonstrates that a biologically plausible learning rule can indeed be used to control a compliant system. Next, I show that it allows for hierarchical control in compliant robotics, by combining a feedforward kinematic controller with a feedback controller to handle the dynamics.

The learning rules I consider fit into the class of Reward Modulated Hebbian plasticity rules [53, 74, 102, 160]. I specifically target this type of learning rule, because of the limited number of assumptions it makes about the substrate to which it is applied. Reward modulation is also an active research field in biology. It is beyond the scope of this work to attempt to explain how the learning rules presented in this chapter can be implemented by biological substrates. However, it is worth mentioning that different reward signals and centers seem to exist in the brain [154], with dopamine

playing an important role in reward signaling [190]. Interestingly, it has been proposed that dopamine signals the reward prediction error or how surprising a reward is [33, 155], which is closely related to the reward estimation on which the learning rules presented herein rely. So while I certainly do not claim that the learning rules presented in this chapter occur in the brain or biological Neural Networks, all major components do have known biological counterparts.

It is vital to define or explain the term *reward*. Throughout this chapter, a reward is a scalar signal R that provides information about the current performance of the system. A reward can be instantaneous, in which case $R(t)$ is a signal that provides a measure of the performance at time step t. Alternatively, a reward can reflect the performance over multiple consecutive time steps, also called a *trial*. In this case, a single reward value R becomes available at the end of the trial. Both cases are studied in this chapter. The reward signal should be a monotonic function of the performance (i.e. an increase in performance should increase the reward signal and vice versa).

In short, this chapter solves a crucial missing aspect of the learning rules presented before. Previously, the problem of finding the desired motor signals was solved by using Evolutionary Algorithms. Here I directly optimize the feedback controller, thus bypassing this problem. Specifically, I will discuss the instantaneous reward case in Section 5.1. Distal (delayed) reward tasks for Neural Networks are addressed in Section 5.2, followed by examples of distal reward learning for tensegrity robots (Section 5.3). I then describe a stabilized version of the basic learning rule used throughout this chapter in Section 5.4. Before presenting my conclusions in Section 5.6, I discuss the results and implications of this chapter and of the previous chapter in Section 5.5.

5.1 Instantaneous Reward

Based on the previous observations, Legenstein et al. [102] introduced a learning rule to train relatively large (compared to my tensegrity structures) Neural Networks. The EH-rule (*Exploratory Hebb*) at the core of Legenstein's work is given by:

$$\boldsymbol{W}_{EH}(t) = \boldsymbol{W}_{EH}(t - \Delta t) - \alpha_{EH}(\boldsymbol{y}(t) - \bar{\boldsymbol{y}}(t))(R(t) - \bar{R}(t))\boldsymbol{x}^T(t). \quad (5.1)$$

In this equation $\bar{R}(t)$ is the short time average (baseline) of the reward signal. Similarly $\bar{\boldsymbol{y}}(t)$ refers to the short time average of the activity $\boldsymbol{y}(t)$ of the neurons which can be observed and controlled (their incoming weight can be

tuned). However, remark that $\boldsymbol{y}(t)$ typically contains all the neurons in the network as there does not need to be a distinct output layer. Unfortunately, the naming can cause some confusion at this point. Previously $\boldsymbol{y}$ typically referred to a set of output neurons, while here $\boldsymbol{y}(t)$ should be interpreted as the *observable* or *controllable* set of neurons. In a neuroscience context, $\boldsymbol{x}(t)$ contains the presynaptic neurons, while $\boldsymbol{y}(t)$ are the postsynaptic neurons. To put it more simply in the setting of this work: $\boldsymbol{y}(t)$ contains the state variables of the system that can be observed and modified to optimize the reward $R(t)$.

One fundamental aspect that is not explicitly mentioned in Eq. 5.5 is that exploration noise — which will be denoted by the variable z — needs to be injected into the system for the learning rule to work. Noise causes variations of the state of the neurons $\boldsymbol{y}(t)$ and $\boldsymbol{y}(t) - \bar{\boldsymbol{y}}(t)$ will thus be non-zero. At the same time the noise can have a beneficial or decremental influence on the reward. The result of this is a non-zero $R(t) - \bar{R}(t)$ factor. What happens next is that the learning rule strengthens or weakens a connection between neurons in $\boldsymbol{x}$ and those in $\boldsymbol{y}$ if variations of $\boldsymbol{y}(t)$ correlate with $\boldsymbol{x}(t)$. The weight updates are *modulated* by the variation of the reward signal. If the reward is above its short term average (or expected value), then positive correlations between $\boldsymbol{x}(t)$ and $\boldsymbol{y}(t) - \bar{\boldsymbol{y}}(t)$ are reinforced and negative ones cause a decrease in the connection weight. The factor $\boldsymbol{y}(t) - \bar{\boldsymbol{y}}(t)$ should really be interpreted as an approximation of the injected noise. Indeed, it is simply the immediate effect of the noise on the behavior of the observed neurons. For moderate noise amplitudes and well-behaved analog neurons (e.g. sigmoidal) the relationship between the noise and $\boldsymbol{y}(t) - \bar{\boldsymbol{y}}(t)$ will be almost linear.

Covariance and noise based rules have a strong biological foundation [109, 108, 159]. For example, it is well known that Neural Networks in biology have intrinsic noise sources [38]. While this type of noise can be measured by external means (e.g. voltage clamps), a plasticity rule within the biological substrate cannot generally observe the noise signals, hence the $\boldsymbol{y}(t) - \bar{\boldsymbol{y}}(t)$ factor in Legenstein's rule. Remark that Legenstein's learning rule extends various earlier techniques with similar mathematical formulations [53, 108, 110]. Note that [110] also provides non-Hebbian variations on this rule.

The Reward Modulated Hebbian rule (RMH) used in this work for PRC applications was inspired by Eq. 5.5 and is given by:

$$\boldsymbol{W}_{rmh}(t) = \boldsymbol{W}_{rmh}(t - \Delta t) - \alpha_{rmh}\boldsymbol{z}(t)(R(t) - \bar{R}(t))\boldsymbol{x}^T(t), \qquad (5.2)$$

where the state $\boldsymbol{x}$ was defined in Eq. 4.10. The short term average reward $\bar{R}(t)$ is computed by taking the average of the rewards during the last 100 ms. Gaussian white noise is used as noise source $\boldsymbol{z}(t)$, with a decreasing standard

deviation as function of time. The noise $\boldsymbol{z}(t)$ is not only used to update the weights, but it is also fed into the structure ($\boldsymbol{y}(t) = \boldsymbol{W}_{rmh}(t-\Delta)\boldsymbol{x}(t) + \boldsymbol{z}(t)$). Indeed, if this were not the case, one would need some critic that provides rewards based on hypothetical motor outputs. The rule depends on exploration noise, which unlike in Legenstein's approach is assumed to be known in all but one experiment[1]. Hence, $\boldsymbol{z}(t)$ replaces $\boldsymbol{y}(t) - \bar{\boldsymbol{y}}(t)$. Motor babbling is the physical realization of the exploration noise and as this type of noise is actively generated, it is straightforward to record it [148]. Furthermore, the number of noise sources corresponds to the number of actuated spring-cable assemblies and is thus limited. Therefore, the PRC framework differs from the large biological Neural Networks considered by Legenstein et al.

The adapted rule will allow to train a set of weights in case an instantaneous reward signal $R(t)$ is available at all times. The reward $R(t)$ should be a monotonic function of the error, i.e. the reward should decrease if the error increases. It is instructive to address two intrinsic limitations of the rule, before delving into the implementation details. First, the instantaneous reward requirement limits the set of problems that can be optimally trained. For example, consider the generation of a pulsating signal. An appropriate reward function for this problem would be based on the frequency spectrum computed over an amount of time. Secondly, in systems with higher order dynamics (e.g. the second order dynamics of a tensegrity structure), there is a time delay before a noise sample at the input/feedback causes a change in the state of the system. In robotic systems such as a tensegrity robot, subsequent time step tend to have highly correlated reward values because of inertial effects. The practical consequence of this is that a learning rule that considers a single time step at a time, can have suboptimal performance for practical PRC tasks. However, in practice the rule performs well for PRC applied to tensegrities if the reward function is based on the input signals. Furthermore, it is instrumental for the introduction of more elaborate Hebbian plasticity rules later in this chapter.

The learning rule from Eq. 5.2 can be used in two ways. First, one can simply use it to replace the learning rules introduced in Chapter 4, when outsourcing the computation to the structure. In this case a teacher is still needed to drive the system during learning, which limits its practical use and it is more or less a replacement for the GD rule. Secondly, it can be used to train feedbacks without knowledge of the target signal at the neural level.

To apply RMH or similar techniques to a Recurrent Neural Network, one typically starts from a chaotic network [74, 168] and the trained instan-

[1] To demonstrate that the rule can indeed be adapted to the unknown noise case.

taneous feedback drives the network toward a cyclic attractor. However, it is reasonable to assume that for robotics applications chaotic movements might be undesired or unsafe. Therefore, I took a slightly different training approach. I first trained the system (using RLS as explained in Chapter 4) to maintain an oscillatory pattern while noise was injected through additional actuators. Hence, I obtained robust but non-chaotic patterns. There are however variations in the oscillations caused by the injected noise. Then, learning through RMH starts on the additional actuators.

One might argue that the use of RLS at this point negates the advantage of RMH. However, RLS is only used to keep the system active during RMH learning and the target signals of RLS and RMH are independent (except for the fundamental frequency). A simple oscillator (e.g. a sine wave or coupled neurons) could also be used instead of a trained feedback controller. In a typical RC setup (with hyperbolic tangent neurons), it is possible to scale up the connection weights to start the learning process in a chaotic regime. In the case of tensegrity structures, I have tried using a random feedback loop which I then scaled to find a chaotic regime. Unfortunately, while doing this the structures often collapsed or did not stay active and I therefore concluded that this method would be cumbersome on a real platform.

The presented approach can be useful in robotic applications in which there is already some oscillatory behavior in the system. This can for example be generated by a very simple CPG signal. The RMH algorithm can then directly be applied to refine the motion. Hence, it is one possible application of the combination of a simplified CPG with my approach. The basic movements can also be provided by a controller based on linearized dynamics, where again RMH can be used to optimize the match between the actual plant and the linearized model.

The three major phases of training using the reward based technique are shown in Figure 5.1. The feedback consisted of 2 trained signals using the RMH rule (out of a total of 8 actuated spring-cable assemblies). RLS was used to train a random motion pattern (with the same frequency) on the 6 first outputs during 200 s (left figure). Then RMH learning starts and initially the target signals are not at all matched. During training (center figure) the outputs begin to match the desired signal more closely, yet there is still some visible error. During testing (right figure), the noise source is disabled and the output almost exactly matches the desired signal. In this example, the tensegrity was in free fall to remove the disturbances from ground collisions to show that the desired signal can be closely matched.

Figure 5.2 shows a phase portrait of the two trained outputs during 40 s of testing compared to the desired output. The target signal is almost perfectly matched. In Figure 5.3 it is shown how the RMH rule is performing

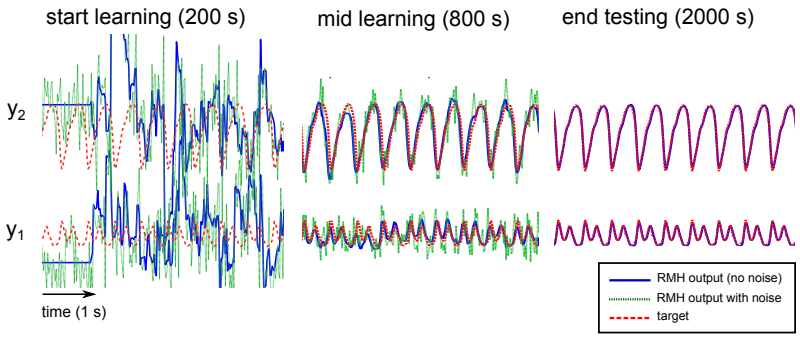


Figure 5.1: The RMH algorithm during training and testing. Training of the two RMH feedbacks starts after 200 s of training with RLS to maintain activity in the system. The tensegrity was in free fall to clearly show the difference between the three phases without influence from ground collisions. A random 6-bar tensegrity was used. The exploration noise decreased linearly as a function of time.

gradient ascent on the reward signal. The signals were smoothed over 2 s to illustrate the evolution of the reward. The figure on the right shows the reward signal with the (estimated) baseline removed. For convergence, the (short-term) mean should approximate zero as otherwise the magnitude of the weights will continue to rise or oscillate. For Figure 5.2 an informative reward was used, namely the sum of absolute errors of both signals:

$$R(t) = -\sum_i |y_i(t) - y_{target,i}(t)| \,. \tag{5.3}$$

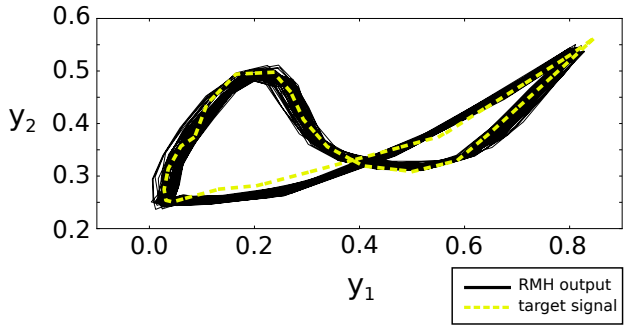


Figure 5.2: Plot of the 2 trained outputs with the RMH algorithm. The system was trained for 2000 s. Yellow: the target signal. Black: the output signal during the final 40 s of testing.

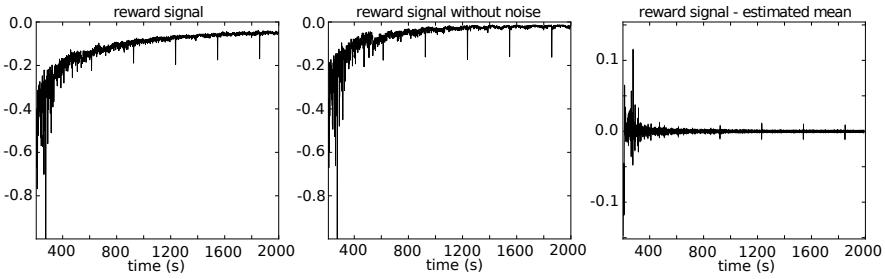


Figure 5.3: The Reward Modulated Hebbian algorithm performing gradient ascent on the reward. The signals were smoothed by averaging over 2 s. From left to right: reward signal, reward signal minus its short term average, reward signal without exploration noise. The reward signal minus the short term average should approach 0 to assure convergence of the weights.

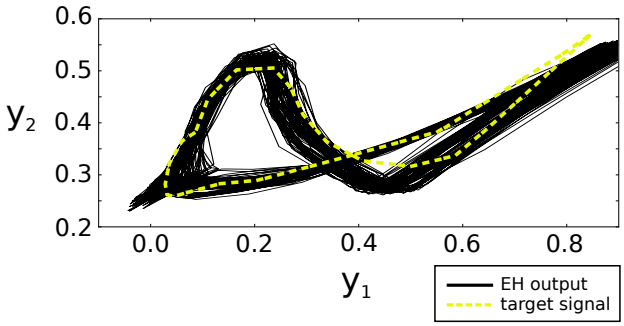


Figure 5.4: Plot of the 2 trained outputs with the EH rule and a less informative reward signal (Eq. 5.6). Hence no knowledge of the target signal, the noise or the precise reward is used. The system was trained for 3000 s. Yellow: the target signal. Black: the output signal during the last 40 s of testing.

Such an informative reward signal as in Eq. 5.3 need not be available for the RMH or EH rule to work. In Figure 5.4, I have applied a binary version of the EH rule with a less informative reward signal (Eq. 5.6). The reward signal only contains information about the observable variable with the worst performance. Additionally, the noise is estimated at the feedback to demonstrate that this does not significantly complicate the problem. The modified learning rule is given by:

$$\boldsymbol{W}_{rmh}(t) = \boldsymbol{W}_{rmh}(t-\Delta t) \tag{5.4}$$

$$-\alpha_{rmh}\left(\boldsymbol{y}(t)-\bar{\boldsymbol{y}}(t)\right)\operatorname{sign}\left(R(t)-\bar{R}(t)\right)\boldsymbol{x}^T(t) \tag{5.5}$$

$$R(t) = -\max_i \left|y_i(t)-y_{target,i}(t)\right|^2. \tag{5.6}$$

The result is clearly less precise compared to the application of the original RMH rule (Figure 5.2) and learning is slower. However, no knowledge of the noise is used in the learning rule and the information contained in the reward signal is limited. The delta version can further reduce the computational/communication power needed as only a single binary signal needs to be exchanged.

This section has demonstrated how the Recursive Least Squares algorithm from the previous chapter can be replaced with a Reward Modulated Hebbian plasticity based learning rule. However, the advantages of the presented learning rule are limited as the reward was instantaneous and although the reward need not contain a lot of information (binary reward), it has to be possible to compute it at the input/feedback for higher order systems. The main contribution of this section was therefore to introduce a reward based version of the learning rules studied in the previous chapter. The next sections overcome these issues by delving into the question of distal reward learning by extending the method presented above.

5.2 Distal Reward Learning for Recurrent Neural Networks

I begin the discussion of the distal reward extension of the Hebbian plasticity rule by applying it to various Recurrent Neural Network topologies. As the learning rules presented in this and the previous chapter only rely upon the existence of a computational black box, it is interesting to verify that similar rules indeed work when applied to both physical substrates (tensegrity robots) and Neural Networks. Originally, the Recurrent Neural Networks implementation was only meant as a verification tool for this. However, as it shows promising results for constrained network topologies, I decided to further investigate this. These results are presented now and although they deviate from the main theme of this dissertation, they do further strengthen the analogy between software (Neural Networks) and hardware (compliant robots) computational substrates.

I start this section by presenting the learning setup shown in Figure 5.5. The central element is the Neural Network to be trained. The network receives input, which I denote $\boldsymbol{U}$ and a readout function provides observations of the network state. Additionally, exploration noise $\boldsymbol{Z}$ is injected into the network.

The term *distal reward* indicates that there is a delay between the action that caused the reward and the actual reward signal [34, 78, 122]. In the

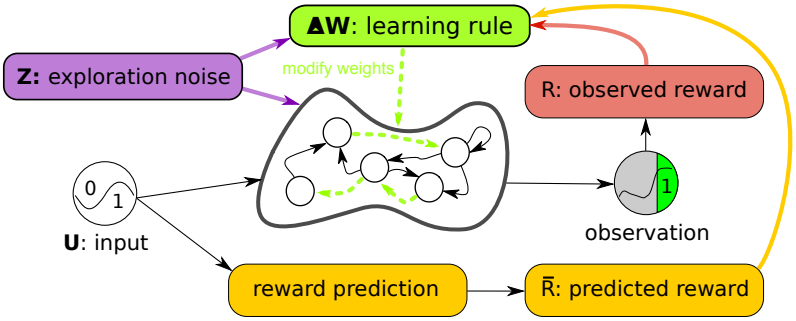


Figure 5.5: Overview of the learning setup for Recurrent Neural Networks. The initially random Recurrent Neural Network receives the inputs $\boldsymbol{U}$ and the exploration noise $\boldsymbol{Z}$. The state of the postsynaptic neurons is computed by applying the hyperbolic tangent function to the sum of the inputs, the noise and the weighted sum of the presynaptic neurons. Observations are made of the state of the network and after every trial (fixed number of time steps), a reward is computed based on the observations made during the last trial. In parallel, a simple reward prediction network predicts the expected reward for the given input. The learning rule then updates the weights between the presynaptic and postsynaptic neurons, by using the reward, the expected reward, the exploration noise and the states of the presynaptic neurons.

context of this section, a reward is provided after a trial, based on the observations of the network (cf. robot). A trial is defined as a number of time steps which correspond to a period of time in which the network tries to perform a task of interest. In parallel to the network, a reward prediction system estimates the expected reward based on the network inputs. The reward prediction system can often be of low complexity. For the instantaneous reward problems, the reward predictor was simply the short time average of the reward signal. Similarly, I will maintain multiple short time average estimators for discrete input spaces. For the continuous input space task, which is presented later in this chapter, a linear reward estimator is trained in parallel with the main system.

The learning rule combines information from the network state, exploration noise, reward and estimated expected reward to compute an update $\boldsymbol{\Delta W}$ of the network weights. Because of the limited duration of a trial and the fixed reward frequency, the scope of this work is limited to distal rewards in which the maximum delay between the reward delivery and the cause of the reward is the length of a trial.

The term *observations* will be used to define a function of the state of the

Neural Network (or robot later on in this chapter). Rewards are based on observations and are therefore not restricted to functions of e.g. the output of the network (in fact there need not be an output).

5.2.1 Neural Networks

The Neural Networks studied here have the same dynamics as the Reservoir Computing networks introduced in Chapter 3, with the addition of a noise source. The network update equation is given by:

$$\boldsymbol{x}[k+1] = \tanh\left(\boldsymbol{W}\boldsymbol{x}[k] + \boldsymbol{W}_{in}\boldsymbol{u}[k+1] + \boldsymbol{z}[k+1]\right). \tag{5.7}$$

Following the Reservoir Computing approach [185], the weights are initialized randomly (i.i.d. standard normally distributed samples) and the weight matrices $\boldsymbol{W}$ are then rescaled to match the desired spectral radius. The same performance was observed for initial spectral radii in [0.95, 1.2]. This differs from related approaches such as [102] and [74] as the networks are not required to be initially chaotic (in case they are input-driven). The input weight matrices $\boldsymbol{W}_{in}$ are sparse (20 % non-zero elements) with i.i.d. normally distributed values with standard deviation 0.05. All networks contain 100 neurons.

5.2.1.1 Network Architectures

Three Neural Network architectures are studied on three tasks (one task per architecture). The objective of using different network architectures for the different tasks is to demonstrate the versatility of the learning rule. Each architecture follows the setup described in the previous section, but with different constraints. The networks for the three tasks are shown in Figure 5.6. All networks are updated using Eq. 5.7 and only differ by how observations are made and by which weights can be modified by the learning rule. The first two networks are observed in the same way, by computing the sum of two neurons of the network. The third network is observed by multiplying three of its neurons.

In all networks, the weights to and from the neurons generating observations are fixed and recurrent. This prevents the learning algorithm from generating solutions in which the observation neurons become a pure output layer, which does not influence the state of the rest of the network.

The weights to and from the other neurons in networks 1 and 3 are modifiable, except for the input weights, which are fixed in all experiments. Therefore there are 98^2 and 97^2 connections to train in networks 1 and 3 respectively. Network 2 further complicates the problem by fixing the

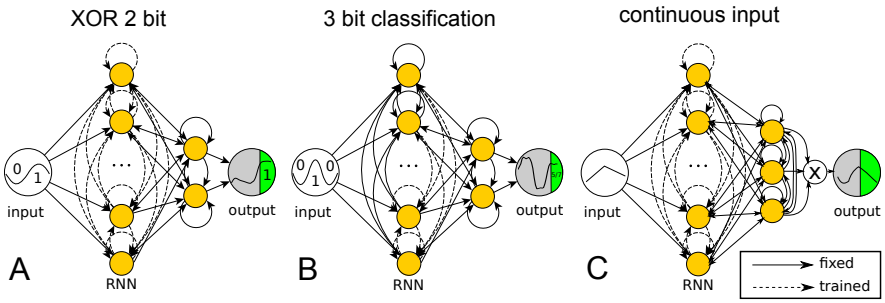


Figure 5.6: Network structures for the Recurrent Neural Network tasks. The networks are simulated in discrete time with hyperbolic tangent neurons (yellow nodes). Full lines are fixed connections, while dashed lines are trained. The reward is evaluated at the output neuron over the green period of time (one reward per trial). **A**) Network structure for the $2\,\mathrm{bit}$ delayed XOR task. The output equals the sum of two neurons (yellow nodes at the right), but only the internal connections can be modified by the algorithm. The RNN consists of 100 neurons and the bit period is 10 time steps. **B**) Network structure for the $3\,\mathrm{bit}$ delayed classification task. The output equals the sum of two neurons as in the XOR task, but now only half of the internal weights can be modified. The RNN consists of 100 neurons and the bit period is 10 time steps. **C**) Network structure for the continuous input task. The network has to reproduce the time reversed input from the first 5 time steps of each trial. A trial consists of 12 time steps in total, of which the last 7 have to be ignored by the network.

weights of half of the network. Of the 98 remaining neurons besides the ones generating observations, 49 have modifiable input weights, while the other 49 have fully fixed weights. This means that about half of the network is a random Recurrent Neural Network (i.e. a Reservoir). Note that having a learning rule that handles delayed rewards is crucial, as the input does not directly, nor immediately define the observations.

It is possible to solve the tasks using any of the three network architectures. In fact, task one and two are highly similar in nature (task two is more complex) and use the same basic network architecture, but with additional constraints for task two. This was done to highlight the aforementioned versatility of the learning rule.

5.2.2 Learning Rule

I now derive the learning rule to solve these distal reward problems. Hebbian plasticity is a biologically plausible learning methodology for Neural Networks. Capital $\boldsymbol{X}$ and $\boldsymbol{Y}$ will be used to indicate that the learning rule can work on multiple time steps at once:

$$\boldsymbol{X} = \Big[\boldsymbol{x}\,[k]\ \boldsymbol{x}\,[k+1]\ \ldots\ \boldsymbol{x}\,[k+i]\Big]^T. \tag{5.8}$$

Recall that a learning rule is called Hebbian if it modifies the weights between a set of presynaptic neurons $\boldsymbol{x}$ and postsynaptic neurons $\boldsymbol{y}$ as a function of their joint activity. Although, Hebb did not provide a precise mathematical formulation of his postulate, a relatively general form can be written as [69]:

$$\boldsymbol{\Delta W}_{\text{Hebb}} = f(\boldsymbol{X}, \boldsymbol{Y}). \tag{5.9}$$

As Recurrent Neural Networks are used, the sets of presynaptic and postsynaptic neurons are the same, shifted by a single time step, i.e. $\boldsymbol{Y}[k] = \boldsymbol{X}[k+1]$.

Similarly to the instantaneous reward case, noise is injected at the postsynaptic neurons for exploration. More precisely, the noise causes variations $\delta\boldsymbol{y}$ of the postsynaptic neurons, and thus variations of the observations and reward. If the exploratory noise causes an improvement in behavior, this will result in a higher reward (and vice versa). A basic learning rule based on this idea is

$$\boldsymbol{\Delta W} = R\boldsymbol{z}\boldsymbol{x}^T, \tag{5.10}$$

where R is the reward. This rule is similar to the RMH rule described in Eq. 5.2.

This rule however suffers from a number of basic flaws to be able to solve distal reward problems. First, a notion of memory of the relationship between exploration noise and the presynaptic neurons is needed. For this, the sample covariance $\boldsymbol{X}^T\boldsymbol{Z}$ can be used to estimate how the exploration noise and the presynaptic neurons correlate. Secondly, note that in its current form, any significant bias of the reward R, will cause unfavorable results. The solution to this issue is to predict the reward and subtract this from the obtained reward, resulting in a learning rule of the form:

$$\boldsymbol{\Delta W} = \alpha(R - \bar{R})\boldsymbol{Z}^T\boldsymbol{X}, \tag{5.11}$$

in which $\bar{R}$ is the predicted reward and a learning rate parameter α was added.

The predicted reward is sometimes ambiguously referred to as the (short term) average reward. More precisely, it is the average (or expected reward) of the trial with noise present in the system. In the RMH learning rule (Eq. 5.2), $\bar{R}$ was indeed simply the short term average reward, because the rule was applied on a sample-by-sample base, under the assumption that noise only influenced the reward of the next time step. As will be demonstrated in Section 5.2.4, the average reward is typically highly dependent on the noise level of the system. The learning rule therefore optimizes the expected reward while noise is present in the system (i.e. $\arg\max_{\boldsymbol{W}} \mathbb{E}[R|\boldsymbol{z}]$), under the assumption that this also optimizes the performance when the exploration noise is removed (i.e. $\arg\max_{\boldsymbol{W}} \mathbb{E}[R|\boldsymbol{z} = 0]$).

In this work, I apply the learning rule on a trial-to-trial basis, by evaluating the sample covariance $\boldsymbol{X}^T\boldsymbol{Z}$ of a number of samples and then applying the learning rule once. Thus, a single reward R is available at the end of a trial and the learning rule performs one update using the recorded state information $\boldsymbol{X}$ and noise $\boldsymbol{Z}$. It is not hard to see that — in case the reward signal can be computed at every time step — one could apply the algorithm at every time step, similar to the Rare Correlation learning rule of [160].

However, there is a distinct difference between the Rare Correlation learning approach (and similar techniques) and the method studied in this dissertation. The basic interpretation I attach to the Reward Modulated Hebbian techniques studied in this dissertation is that if noise induces a correlation between $\boldsymbol{X}$ and $\boldsymbol{Y}$ which leads to an increased performance, then the weights are adapted such that this correlation occurs naturally in future trials. For very long trials this method will tend to fail or become significantly less efficient, as inherent averaging during the computation of the sample covariance $\boldsymbol{X}^T\boldsymbol{Z}$ will cause the useful information to be lost. I have successfully tested the method for trials of up to 60 time steps (Section 5.3.3), but it can be expected that the method will break down at longer trial lengths. Rare

(or spurious) Correlation learning on the other hand is aimed at detecting instances of rare concurrent events in presynaptic and postsynaptic neurons. It is a good fit for spiking Neural Networks as one can look at the spike times to detect unexpected instances of postsynaptic neurons firing within a short time after a presynaptic one. In such a framework, the concurrent activity of presynaptic and postsynaptic neurons can significantly predate the delivery of a reward and the method is thus well adapted for problems in which a particular event is the cause of a reward. Notwithstanding the differences in interpretation and objective, the resulting learning rules are mathematically highly similar.

5.2.3 Discrete Input Space

I now address two problems with discrete input spaces: the 2 bit delayed XOR problem and a 3 bit delayed classification task.

5.2.3.1 Network Input and Reward

For these problems, the Recurrent Neural Network receives a stream of input data that represents a bit stream. A zero bit is encoded as the negative half period of a sine wave, a one bit as the positive half (see the top row of Figure 5.8). This input encoding provides synchronization information to the network as there is a clear distinction between consecutive bits. Thus the resulting networks will be input driven even when the input bit sequence is constant. This can be particularly important when the network has fading memory. Secondly, it is straightforward to change the time scale of the input encoding.

In the 2 bit delayed XOR task, the Recurrent Neural Network has to compute the exclusive OR function applied to the last two bits of its input stream. More precisely, the observations of the network should be as close to plus one or minus one during the last half of the second bit. Similarly, the 3 bit task requires the Neural Network to approximate 8 non-linearly separable values within the range $[-1, 1]$. For each trial, a random sequence is selected out of the 4 (2 bit XOR) or 8 (3 bit classification) input sequences.

Various reward functions are used in this section. The main reason for this is to show that the learning rule is not dependent on a specific reward function, though some reward functions are more appropriate for a specific task. If a less informative or less apropriate reward function is used (cf. Eq. 5.6), the performance of the learning rule will decrease. For the 3 bit task, the reward value R^{3bit} is defined as minus the mean squared error of

the observations during the last five time steps:

$$R_k^{3bit} = \frac{-1}{5} \sum_{i=0}^{4} (t[i] - x_0^o[25k+i] - x_1^o[25k+i])^2, \tag{5.12}$$

where k indicates the number of the current trial and $\boldsymbol{x^o}$ are the neurons that generate the observations (see next section). For the results presented for the 2 bit XOR task, I use minus the mean squared hinge loss, as the hinge loss is a more appropriate reward function for a binary classification task:

$$R_k^{2bit} = \frac{-1}{5} \sum_{i=0}^{4} \max\left(0,\, 1 - t[i]\left(x_0^o[15k+i] - x_1^o[15k+i]\right)\right)^2. \tag{5.13}$$

However, I have also successfully applied the mean squared error reward function for this task. The target values are indicated by red crosses in the bottom row of Figure 5.8.

Estimating the expected reward is trivial in the case of a modest number of different inputs: For the results presented here, I have averaged the last 50 rewards per input sequence.

It is true that these reward functions are rather informative. Nevertheless, they convey only minor information about how the task can be solved (a single performance indication based on the observed neurons per trial). The reward functions do not indicate how the internal weights (the only ones that can be modified) need to change to optimize the expected reward. This is reflected in the state trajectories as shown in Figure 5.8. The state trajectories of the two neurons which generate the observations for the 3 bit classification task are unique and distinct. There is no direct link between the reward function used and the resulting patterns.

5.2.3.2 Discrete Input Space: 2 bit Delayed XOR

The XOR task is a common test or benchmark, because the inputs are not linearly separable. A linear network cannot obtain optimal performance for all inputs. Therefore, this task is a simple test to verify whether the learning rule can exploit the non-linear effects of the network.

Secondly, the task requires the network to remember a specific part of the input, while ignoring inputs over one bit length in the past.

Figure 5.7 shows the evolution of the reward and the spectral radius while the system learns the 2 bit delayed XOR task. The noise level was $\sigma = 0.01$ and the learning rate was $\alpha = 0.05$. The initial spectral radius was set to 0.95 to show how the spectral radius evolves as the average reward increases. Every run of the algorithm for different initial random weights

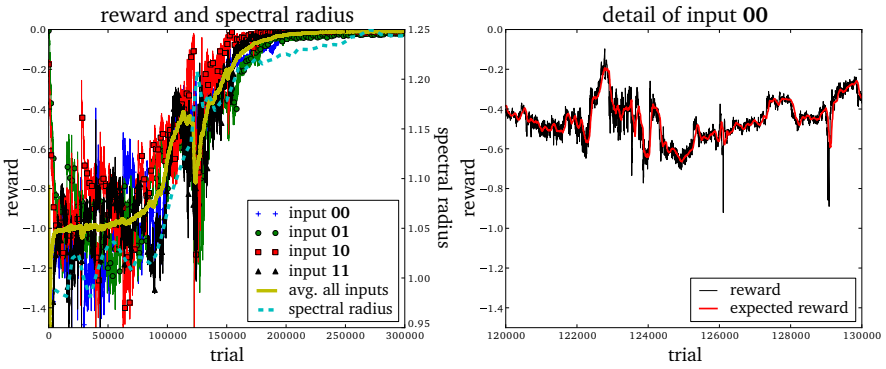


Figure 5.7: Left: Evolution of the reward and the spectral radius for the 2 bit delayed XOR task. Right: Detail of the reward and expected reward for a single input sequence.

resulted in a final spectral radius of $\rho \approx 1.2$, which indicates that the learning rule indeed tunes the memory of the network.

For this particular run, a drop in performance can be seen after about 125 000 trials. This is probably a significant change in network behavior due to only a minor weight change, a typical phenomenon in non-linear Recurrent Neural Networks. However, the algorithm successfully continues to learn afterwards.

In the right column of Figure 5.7, a detailed view of the reward and expected reward for a single input sequence is shown.

5.2.3.3 Discrete Input Space: 3 bit Delayed Classification

I now analyze in more detail the performance and robustness of a more complex task with discrete input space. The Recurrent Neural Network now has to classify 3 bit patterns and the learning rule is only allowed to modify half of the internal weights of the Recurrent Neural Network. A fundamental question is whether the RNN can effectively learn to solve the task by recognizing the input patterns or whether the computations were already present in the network as in a standard Reservoir Computing setup?

Figure 5.8 shows the results of overlaying 50 random orderings of the input sequences. It can be seen that each additional bit reduces the number of possible states of the system by half. Furthermore, the network has learned the correct time window, as the states only depend on the two previous bits and not on older inputs (the random inputs were fed into the system sequentially). Interestingly, the two neurons that generate the observations have different state trajectories, because the learning rule only quantifies the performance based on the observations, without directly enforcing a specific

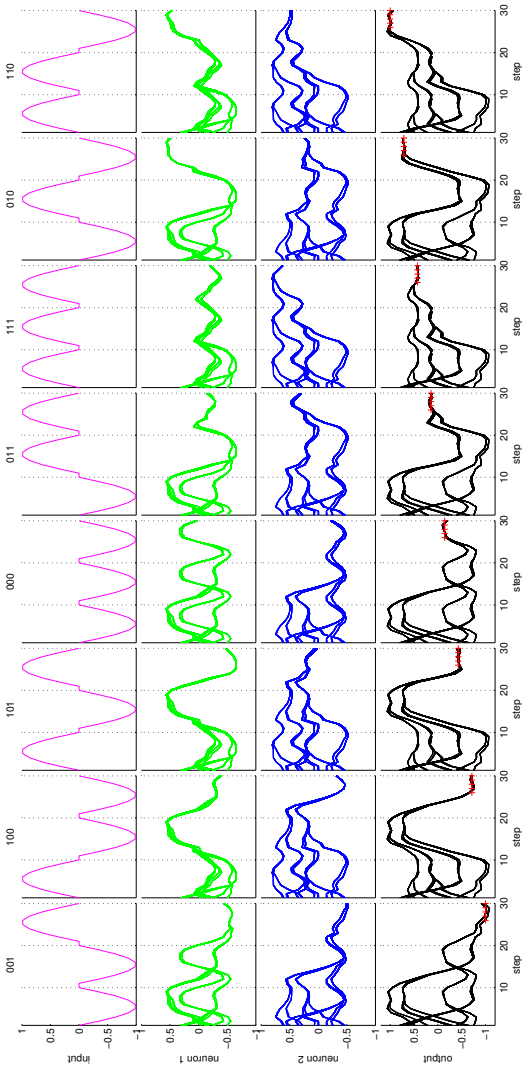


Figure 5.8: State trajectories for the $3\,\mathrm{bit}$ classification task. The top row shows the 8 input sequences. The next two rows show the state trajectories of the 2 neurons which are summed to compute the observations (see Figure 5.6) for the given input sequence of the top row. The bottom row shows the observations (the sum of the two middle rows). The desired observation at the end of the trial is indicated by red crosses. It can be seen that each additional bit reduces the number of possible states by half. Interestingly, the two neurons that generate the observations have different state trajectories, because the learning rule only quantifies the performance based on the observations, without directly enforcing a specific behavior of the neurons responsible for the observations. The plots were generated by overlaying 50 random orderings of the input sequences.

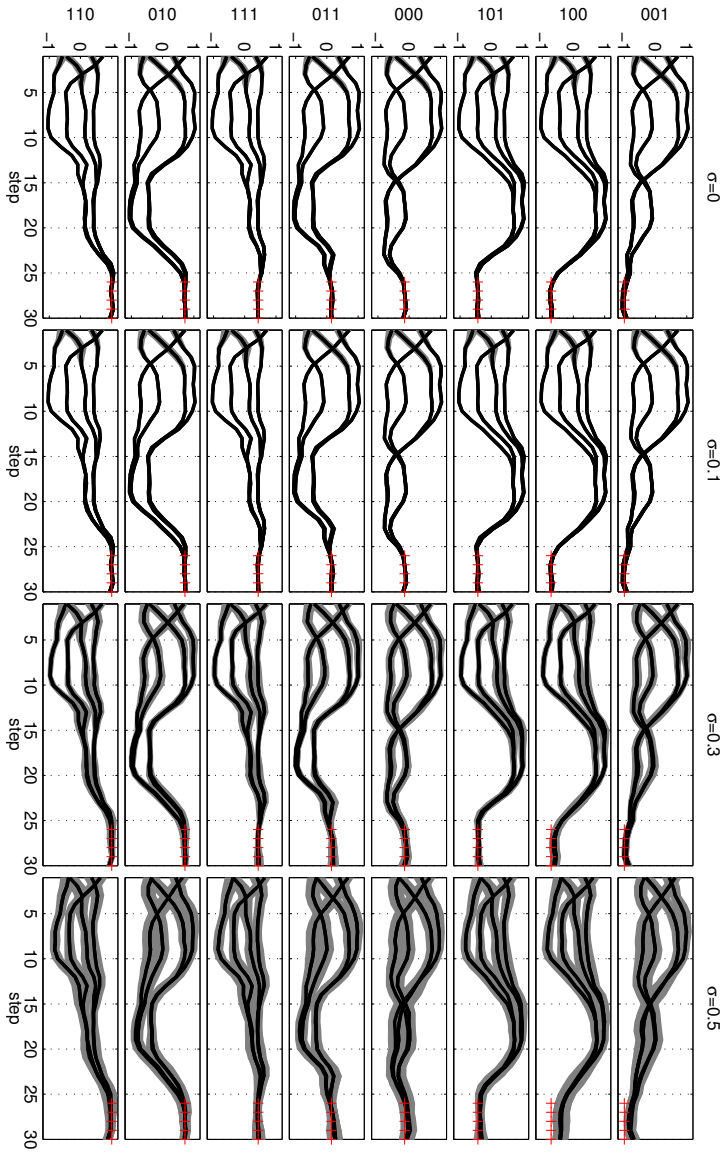


Figure 5.9: Evaluation of a trained network for the $3\,\mathrm{bit}$ classification task under the influence of input noise. As exploration noise drives the learning rule, the trained networks are generally robust against both input and state noise. The noise amplitude increased from left to right. The 8 possible inputs are shown from top to bottom. The gray area indicates one standard deviation around the observations for each of the different state trajectories (thick lines). The red crosses indicate the target observations at the end of the trial.

behavior of the neurons responsible for the observations.

The behavior of the network with various levels of input noise is shown in Figure 5.9. It turns out that even when a large amount of noise is present in the input, the system state still evolves along a fixed number of trajectories/attractors. The network is robust against high amounts of noise on the input data ($\sigma = 0.5$), as the original trajectories are maintained. This plot was generated by applying k-means clustering on the trajectories and then estimating the variance of each centroid. The various centroids and their respective standard deviations are shown. It is remarkable that the network is robust against these levels of input noise as no input noise was present during training. The learning rule appears to significantly regularize the solution and is thus inherently robust.

5.2.4 Continuous Input Space

My third example has a continuous input space and a more complex readout function. The task at hand is to reproduce part of the input in reverse after a delay (see Figure 5.10). To solve this task, the network receives two inputs. The first input contains input sequences of 12 time steps per trial, the first 5 of these steps are straight lines of which the start and end points are uniformly sampled from [0, 1]. The first input is kept constant during the next 2 time steps and during the final 5 time steps this input linearly interpolates between the value at time step 6 and a random value in [0, 1] at time step 11. The network must learn to recall the input during the first 5 steps of each trial and ignore the remaining 7 steps.

The second input is a binary signal that is used to inform the network that it has to generate the desired observation (i.e. a trigger). A second input signal is needed, because the trials are fed into the system, one-by-one. Therefore, it is not clear to the network when a trial starts.

The complete network structure is shown in Figure 5.6. The observations are computed by multiplying the state of three internal neurons with fixed incoming and outgoing connections. The reward function used here is minus the mean absolute error of the observations:

$$R_k = \frac{-1}{5} \sum_{i=0}^{4} |u[12k+i] - \prod_{j=0}^{2} x_j^o[12k+11-i]|, \tag{5.14}$$

where k indicates the number of the current trial and $\boldsymbol{x^o}$ are the neurons that generate the observations.

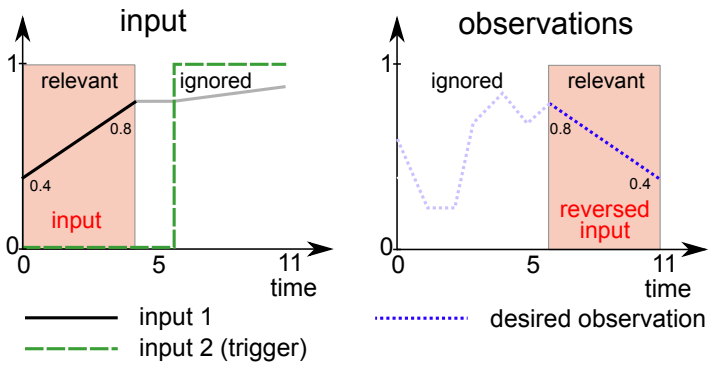


Figure 5.10: Overview of the continuous input space task. The network has to reproduce part (first 5 time steps) of the first input (solid line in the left figure) in reverse at the end of the trial (dotted line in the right figure). The total length of a trial is 12 steps. The task input space is the unit square $[0, 1]^2$ and the first 5 steps of the first input encode this input by applying the first coordinate during the first step and the second coordinate during the last step, with linear interpolation during the intermediate steps. Task input $(0.4,0.8)$ is shown in the above example. A second input (dashed line in the left figure) acts as a trigger and indicates when the network has to start producing the desired observations.

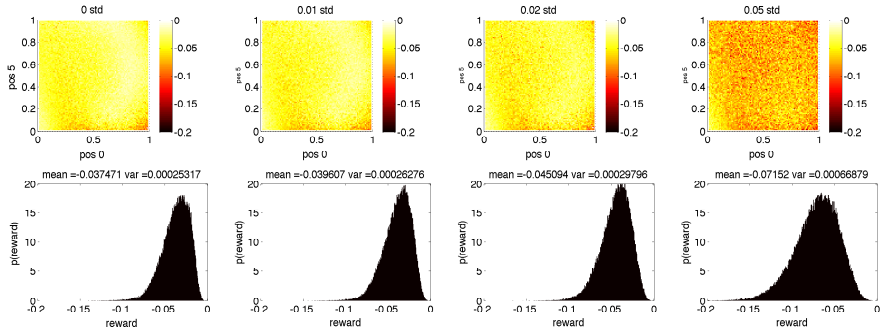


Figure 5.11: Recurrent Neural Network continuous input task results. Top: average reward (negative mean absolute error) for each input combination. Bottom: sample distribution of the rewards for all inputs. The amount of injected state noise increases from left to right. The average reward level shifts to the left (lower rewards).

5.2.4.1 Prediction of the Expected Reward

The expected reward $\bar{R}$ estimates the performance of the system given the noise level σ. Furthermore, the reward is input dependent, therefore $\bar{R}$ estimates the following quantity:

$$\bar{R} = \mathbb{E}[R|\boldsymbol{u}, \boldsymbol{x}, \sigma]. \tag{5.15}$$

Various algorithms can be used to estimate this quantity. I have used the Recursive Least Squares algorithm to learn a simple online estimator of this quantity. More precisely, RLS was used to estimate the reward R based on the input sequences $\boldsymbol{u}$ and the network state $\boldsymbol{x}$ at the end of a trial. The estimator is trained in parallel with the main algorithm as this can be done significantly faster than solving the actual task. In Figure 5.5 only the input was connected to the reward estimator. The additional connections from the observations to the reward estimator are indeed not strictly necessary, but they increase the performance of the system, because the estimator can take the state of the network into account (e.g. the influence of the previous trial).

5.2.4.2 Results and Robustness

Figure 5.11 shows the reward distribution for various state noise levels during testing with 10^5 random inputs per noise level. The internal neurons of the network, except for the 3 neurons that generate the observations, were disturbed. The network was first trained for 10^6 time steps with an exploration noise level $\sigma = 0.05$ and a learning rate of 0.005.

There are two interesting conclusions that can be made based on Figure 5.11. First, one can see that the performance of the network is almost uniform over the input space. Only a small region of diagonal lines near $(1, 0)$ has slightly worse performance.

Second, it becomes clear that $\bar{R}$ is indeed the expected reward under the influence of noise for the given input sequence. The expected reward shifts as a function of the amount of noise in the network. The learning rule evaluates the performance with respect to the expected reward for the current noise level.

5.3 Hebbian Plasticity with Distal Rewards for Tensegrity Robots

I now discuss the adaptation of the distal reward Hebbian algorithm to a tensegrity robot control problem. The general setup is shown in Figure 5.12 and is similar to the Neural Network approach that was presented in Figure 5.5. The Recurrent Neural Network is replaced with a simulated tensegrity robot and the rewards are based on an end-effector trajectory.

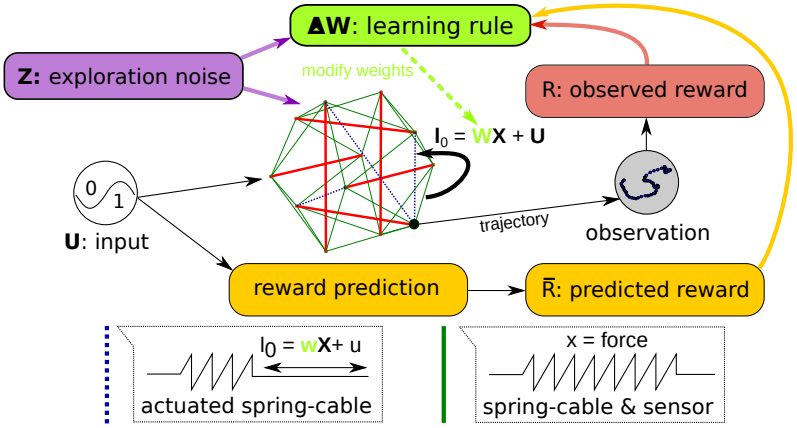


Figure 5.12: Overview of how the learning rule is applied to compliant Tensegrity structures. The setup is fully analogous to the Recurrent Neural Network setup of Figure 5.5. The Neural Network has been replaced with a compliant structure. Force transducers on the spring-cable assemblies act as presynaptic neurons and the actuator signals correspond to the postsynaptic neurons. The learning rule adapts the feedback weights from the force sensors to the motor signals. The observations are based on the trajectories of an end-effector.

As before, spring-cable force transducers provide the readout signals of the computational black box. A minor difference with respect to the previous experiments (Chapter 4) is that no feedback from the derivatives of the spring-cable tensions are used (which gave no significant improvement in performance). Actuators changing the equilibrium lengths of the spring-cable assemblies replace the postsynaptic neurons and motor babbling takes on the role of the exploration noise:

$$\boldsymbol{l}_0 = \boldsymbol{l}_{init} + \boldsymbol{W}\boldsymbol{x} + \boldsymbol{u} + \boldsymbol{z}. \tag{5.16}$$

5.3.1 Tensegrity Structure

The structure used for these experiments has 4 struts and is shown in Figure 5.13. It is based on the standard 3-strut tensegrity prism to which a shorter rod is added that acts as a compliant end-effector. The bottom 3 nodes of the original prism are fixed to the ground through ball-joints (i.e. the robot is standing on the ground and movement of its base is constrained). The resulting structure has 17 $k = 20\,\mathrm{N\,m^{-1}}$ spring-cable assemblies, 14 of which are actuated (the 3 bottom spring-cable assemblies are irrelevant). The controller time step was 50 ms.

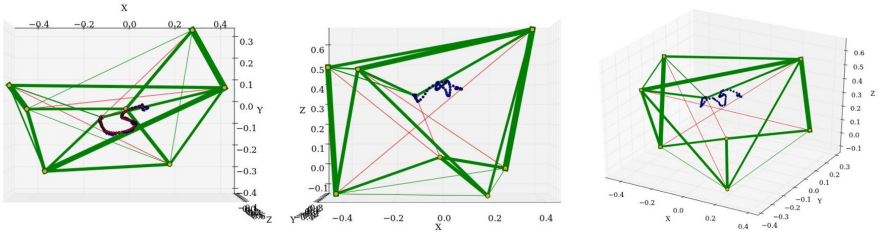


Figure 5.13: Tensegrity structure used for the experiments. The top node of the center rod is used as an end-effector to draw in the XY plane. In this example, the robot draws an "S" as can be seen on the left. The center and right figures show other perspectives to demonstrate that the reward does not depend on the vertical position.

5.3.2 Writing Characters

The task I consider is writing characters with the top node of the rod suspended in the tensegrity structure. More precisely, the node has to trace letters in the XY plane. The characters were taken from the UCI Character Trajectories dataset, integrated (using the velocity values in the dataset) and then subsampled and rescaled.

The reward function used for the next experiments tries to bring the end-effector close to the desired trajectory:

$$R = \frac{-1}{s} \sum_{i=0}^{s-1} \max\left(\|\boldsymbol{n}[i] - \boldsymbol{c}[i]\| - 0.01,\, 0\right), \tag{5.17}$$

where s is the number of steps of the current character, $\boldsymbol{c}[i]$ the vector containing the target position at time i (relative to the beginning of the trial) and $\boldsymbol{n}[i]$ the position in the XY plane of the end-effector at time i. This reward function will cause the learning rule to stop improving a feedback

controller w.r.t. a point on the trajectory in case the end-effector is within 1cm of the target position.

5.3.3 Kinematic and Feedback Controller

The robot is controlled by combining a feedforward kinematic controller and a learned static linear feedback controller. 100 random spring lengths were sampled to create a set of configurations for the kinematic controller. To write a character, the kinematic controller selects a combination of spring lengths that move the end-effector as close as possible to the desired position when the structure is in equilibrium. This was implemented in a straightforward way by sampling 100 random combinations of spring rest lengths and observing the resulting equilibrium position of the end-effector. The kinematic controller then simply chooses the sample which moved the end-effector closest to the target position.

As can be seen from the top row of Figure 5.14, the initial performance of the system with only this basic kinematic controller is low. In fact, the main goal of this controller is not achieving optimal performance, but rather to inject some energy into the structure. When the RMH rule was presented in Section 5.1, I showed an example with instantaneous reward function in which I first trained a feedback controller with known target signals using recursive least squares, and then proceeded to learn additional feedback signals using a Reward Modulated Hebbian rule. The reason why an additional energy source is employed in both cases, is that it is hard to consistently learn pure feedback controllers with simple Hebbian-like learning rules. A small change in a feedback weight can cause the system dynamics to fade out, which often results in instability.

Therefore, an easy and efficient solution is to use an additional input that consistently pumps energy into the system. This can be accomplished using a feedback controller as in Chapter 4, a simple feedforward controller as I use here, or another controller such as a Central Pattern Generator. Furthermore, this allows for a layered control architecture in which the learned feedback controller handles the fine-tuning of the movement. From a practical point of view it is physically unfeasible to start from a chaotic regime as in FORCE learning [168].

5.3.4 Learning Rule

While the robotics setup is fully analogous to the Neural Network setup, the robot has significantly fewer *neurons* or observable variables. Furthermore, the observations tend to be highly correlated, which can slow down learning.

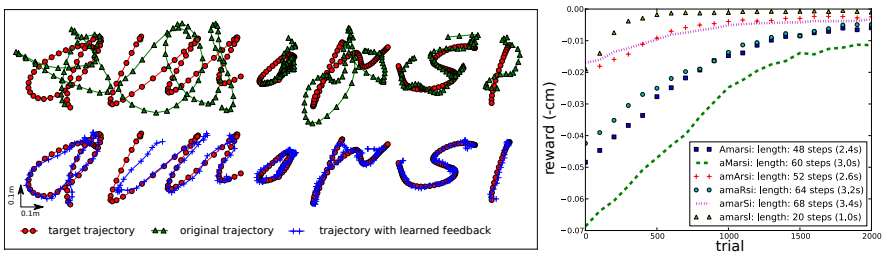


Figure 5.14: Writing characters with a tensegrity end-effector. Top left: characters drawn with only the kinematic feedforward controller active. Bottom left: characters drawn with the kinematic feedforward controller and the learned feedback controller active. Right: learning curves for the different characters. The legend indicates the length of a trial.

For example, stiffening the structure, will typically cause an increase in most sensor values. Decorrelating the observations can overcome this issue. I take a pragmatic approach and decorrelate on a trial-by-trial basis, while a more biologically plausible solution is possible using a decorrelation layer trained using the Generalized Hebbian Algorithm [150].

The resulting learning rule is given by:

$$\boldsymbol{\Delta W} = \alpha H(R - \bar{R})(\boldsymbol{X}^T\boldsymbol{X} + \lambda \boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{Z}, \tag{5.18}$$

where H is the Heaviside step function[2]. The use of the Heaviside step function disregards trials with less than nominal performance. This modification has shown to have slightly better performance for the problem at hand, but is not a requirement.

The learning rule is similar to ridge regression, the difference being that the algorithm will try to reproduce the noise $\boldsymbol{Z}$ (instead of a desired output) proportionally to the relative performance with the injected noise w.r.t. the expected reward. It turns out that unlike the influence of this parameter in normal ridge regression (see Section 3.3), the exact value of λ has little influence on the final result in this case. The regularization parameter was thus manually tuned to find the correct order of magnitude. I found $\lambda = 1$ (or any value in that order of magnitude) to perform well for the robotics tasks discussed here. However, it is imperative to find a good λ in case the method is applied to a different task or another system (it is highly dependent on the scale of the state $\boldsymbol{x}$ and the length of a trial). The reason why the method introduced here has a reduced dependence on the exact value of λ compared to normal ridge regression is that it is an online learning method. In ridge

[2]Note that $H(R - \bar{R})$ is a scalar.

regression, one should optimize the regularization parameter to maximize the performance on a validation set [18]. Here, λ is a parameter of the learning algorithm, similar to the learning rate α. High values of the regularization parameter will tend to increase the learning time (the dimensions are not decorrelated anymore), but can also make the procedure more robust as $\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}$ can be unstable for short samples.

The matrix inverse in Eq. 5.18 makes any claims of biological plausibility unfeasible. However, a biologically plausible decorrelation layer can be used to undo the need for the factor $(\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}$. This was discussed in [22][Appendix I].

5.3.5 Results and Robustness

Figure 5.14 shows how the learning rule allows the tensegrity end-effector to draw various characters between 20 and 68 time steps in duration (1 s to 3.4 s). A different set of feedback weights was learned for each character, therefore it is easy to predict the expected reward. To clarify, I have estimated the expected reward by averaging the rewards obtained during the previous 30 trials.

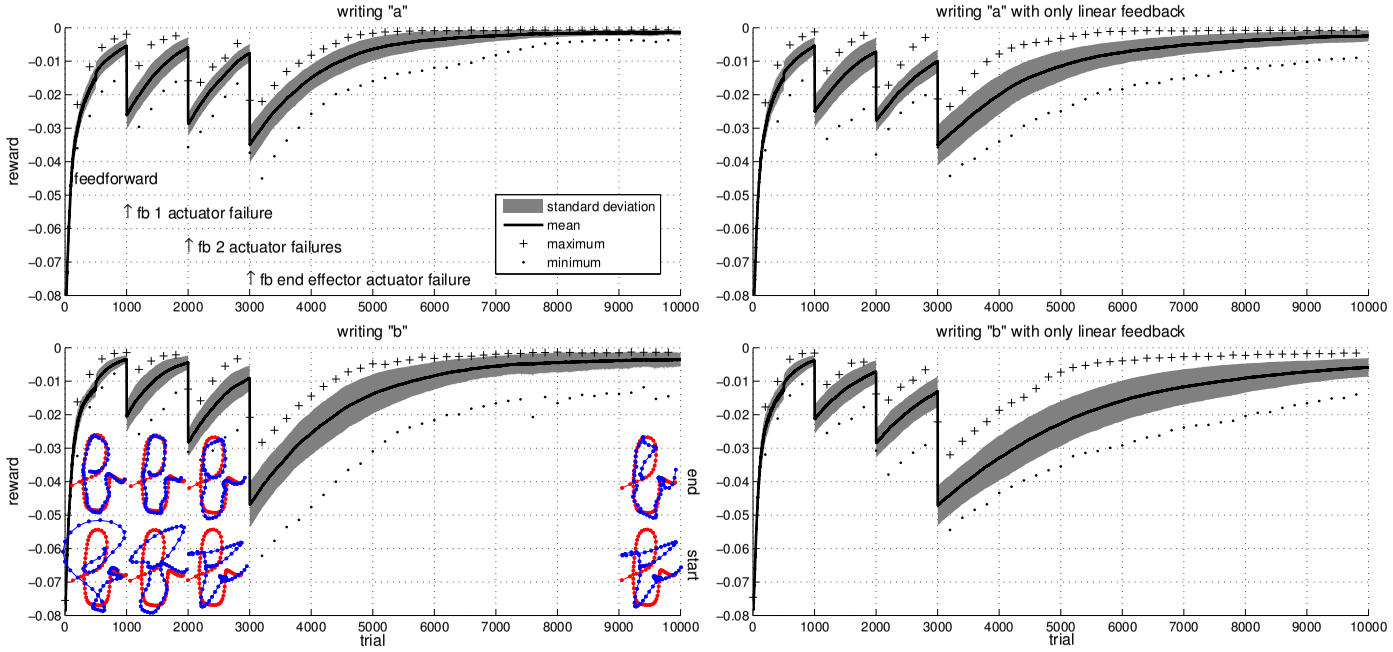
The plot on the right of Figure 5.14 shows the learning curves for the various characters, indicating that for most characters good results were obtained after 1000 to 1500 trials (less than one hour real-time for most characters).

It is possible to accelerate learning by further tuning the learning parameters. A conservative level of exploration noise ($\sigma = 5\,\mathrm{mm}$) was used, while the learning rate was $\alpha = 1$. This consistently resulted in stable feedback controllers.

The learning rule did not achieve the same final reward for all characters (e.g. the "m"). This is due to physical limitations enforced on the motor commands (maximum velocity).

To demonstrate the robustness of the controllers and a more practical application of the learning rule, I have simulated various actuator failures. Figure 5.15 presents the re-

Figure 5.15: Robustness of the learning rule for the writing task. Initially a feedforward controller is optimized using gradient ascent. A failure is then simulated by making a single actuator follow its initial trajectory from trial 1000 onwards. At the same time, the learning rule starts learning a set of feedback weights to compensate for the actuator failure. Similarly, I simulate a failure of 2 actuators after trial 2000. At trial 3000, a failure is simulated of an actuator directly attached to the end-effector. The top row shows the results for writing an "a" character, while the bottom row shows the results for a "b". The left column shows the results when the feedback includes the spring forces and the square of the spring forces, while the right column only includes the spring forces.


writing "a"
writing "a" with only linear feedback
writing "b"
writing "b" with only linear feedback
reward
trial
feedforward
fb 1 actuator failure
fb 2 actuator failures
fb end effector actuator failure
standard deviation
mean
maximum
minimum
end
start

sults of these experiments. Initially, a feedforward controller is optimized using a gradient ascent approach applied to the spring lengths at each time step, again starting from a simple kinematic controller.

Next, communication/actuator failures are simulated, by having one or two actuators follow their initial trajectories instead of the optimized ones. As can be expected, the performance immediately drops significantly. By applying the learning rule, the system is able to recover from the various failures. To investigate the stability of the learning rule, each experiment was performed 30 times.

5.4 Extending Oja's Rule: a Stabilized/Regularized Reward Modulated Hebbian Rule

Oja's rule [130] and its extension — the Generalized Hebbian Algorithm or Sanger's rule [150] — provide a single layer Neural Network implementation to compute principal components. Contrary to pure Hebbian plasticity, these learning rules are stable, because they force the norm of the weight vectors to unity. In this section, I provide a similar derivation for the learning rules studied in this chapter. Unlike in the unsupervised learning case, Reward Modulated rules tend to be stable in practice (i.e. the trained weights remain bounded). However, it can still be useful to control the norm of the weights as this can have practical implications. For example, in a robotics application, this would allow to limit the required feedback gain and thus the required motor power. From a theoretical point of view it is also instructive to see how the learning rules from the previous sections resemble the now classic rule discovered by Sanger more than 20 years ago.

To simplify the notation, I start by defining a number of variables ($\boldsymbol{E}$ is not related to the stress matrix from Chapter 2):

$$\boldsymbol{E} = \boldsymbol{Z}\boldsymbol{X}^T \tag{5.19}$$

$$R' = R - \bar{R}. \tag{5.20}$$

The basic learning rule used for distal reward learning can now be written in element-wise form as:

$$W_i^j[n+1] = W_i^j[n] + \alpha R' E_i^j. \tag{5.21}$$

Oja's rule is a first order approximation to the normalization of the

weights at every update step:

$$W_i^j[n+1] = b^j \frac{(W_i^j[n] + \alpha R' E_i^j)}{||\boldsymbol{W}^j[n] + \alpha R' E^j||}, \tag{5.22}$$

where $\boldsymbol{b}$ contains the desired L_2 norms of the weight vectors.

Now consider the linearization of this rule for small learning rates α. To further simplify the notation, I drop the time index and focus on a single output dimension:

$$w_i \approx b \frac{w_i + \alpha R E_i}{||\boldsymbol{w} + \alpha R E||}|_{\alpha=0} + \alpha b \left(\frac{\delta}{\delta \alpha} \frac{w_i + \alpha R E_i}{||\boldsymbol{w} + \alpha R E||} \right) |_{\alpha=0} \tag{5.23}$$

A straightforward calculation shows that the part inside the parentheses of the second term can be written as:

$$\frac{\delta}{\delta \alpha} \frac{w_i + \alpha R E_i}{||\boldsymbol{w} + \alpha R E||}|_{\alpha=0, ||\boldsymbol{w}||=b} = \frac{1}{b} R E_i - \left(\frac{\sum_k R w_k E_k}{b^3} \right) w_i \tag{5.24}$$

assuming $||\boldsymbol{w}|| = b$ and $\alpha = 0$.

The complete learning rule can therefore be written as:

$$w_i = b \frac{w_i}{b} + \alpha b \left(\frac{1}{b} R E_i - \left(\frac{\sum_k R w_k E_k}{b^3} \right) w_i \right) \tag{5.25}$$

$$= w_i + \alpha R \left(E_i - \frac{w_i}{b^2} \sum_k w_k E_k \right). \tag{5.26}$$

The matrix form of the rule for multiple outputs is given by:

$$\boldsymbol{\Delta W} = \alpha R' (\boldsymbol{E} - \text{diag}_v(\boldsymbol{b})^{-2} \text{diag}_v(\text{diag}_m(\boldsymbol{E W}^T)) \boldsymbol{W}), \tag{5.27}$$

which is of the order $O(mn)$, with m the number of outputs and n the number of inputs. The diag_m operator extracts the diagonal of a matrix into a vector and the diag_v operator converts a vector into a diagonal matrix. Therefore $\text{diag}_v(\text{diag}_m(\boldsymbol{E W}^T)$ is a diagonal matrix containing the diagonal elements of $\boldsymbol{E W}^T$. The stabilized learning rule therefore has a similar form as the Generalized Hebbian Algorithm.

Figure 5.16 provides a toy example of the stabilized RMH rule derived above. In this example, the objective is to train a single layer linear feedforward network for a regression task with 10 inputs and a single output. The reward function used was minus the MSE and the trial length was 1000 samples (the full train set). The figure shows the short term average reward (with noise) as a function of the norm of the weight vector. The thin black line shows the performance during unbounded training (the basic RMH rule from

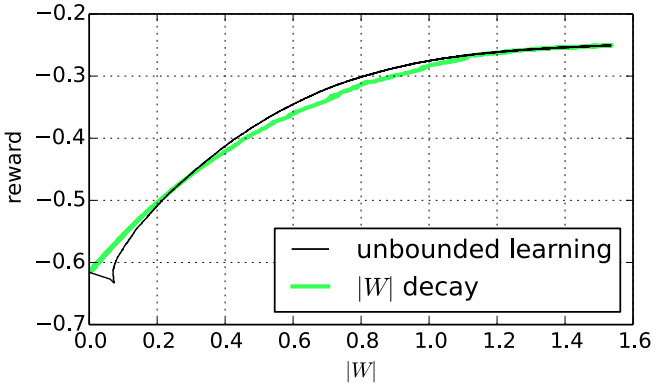


Figure 5.16: Example of the stabilized/regularized Reward Modulated Hebbian Rule from Eq. 5.27. The learning rule is applied to implement weight decay for Reward Modulated learning. The problem at hand is a feedforward regression task with 10 inputs and a single output. The initial unbounded learning phase is shown in thin black line and the thick line traces the performance when the norm is constrained. More precisely, b is gradually reduced starting from the norm at optimal performance during unbounded training.

e.g. Eq. 5.21) which starts from an all-zero weight vector and reaches optimal performance when the norm is approximately 1.55. Afterwards the learning rule from Eq. 5.27 is used to implement weight decay. Beginning from the optimal weight vector obtained during unbounded learning ($b \approx 1.55$), the target norm b of the weight vector is gradually decreased. This entails a decrease in performance (thick green line). This change in performance is clearly non-linear and near-optimal performance can be maintained until the norm b is forced below 1.

5.5 Discussion

Compliant robots have been of interest to the robotics community for over a decade. Many exciting examples have appeared in the literature of very simple control laws leading to complex behavior and of robustness against external perturbation. Compliance offers multiple advantages over classic, stiff robotics: it can allow for safer robot-human interactions by simplifying control laws, increased energy efficiency, and inherent robustness against external perturbations.

Notable examples of compliant robots that have very simple control laws are Puppy [81], Reservoir Dog [193], Wanda [201] and a recent elastic beam robot [140]. The control of the Reservoir Dog in irregular and unknown terrain was simply based on a sine wave with different phase and offset for each leg, while a similar stiff robot would need complex sensory equipment and an elaborate controller [19].

In this and the previous chapter, I have used tensegrity structures to model compliant systems. However, two important remarks need to be made. First, the exact dynamics of the system need not be explicitly known to the learning algorithm. This is the underlying idea of Reservoir Computing: A dynamic system can be used as a computational black box, encoding a non-linearly expanded history of environment interactions in the instantaneous state of the system. Such an abstraction has many advantages, as one can change substrates or construct hybrid systems, while still using the same read-out learning algorithms. It also does not define how the read-out mechanism is actually implemented, and would allow e.g. a neural substrate, electrical wiring or mechanical connections.

Secondly, one can exploit the fact that, historically, tensegrity structures have been used to model a plethora of complex systems from the micro to the macro scale. Even though tensegrity structures were initially only used in art and architecture, they have now also been successfully applied as a model for cellular cytoskeleton structures [84, 85, 188]. At the micro scale, the equations of motion are different and their exact form is often unknown, but one still finds compressive elements (e.g. microtubules) and tensile elements (e.g. microfilaments). Inside a single celled organism, there is no central nervous system, but chemical and mechanical interactions define the cell's behavior, and flagella or cilia allow locomotion [103, 146, 169, 170]. Micro-organisms such as nematodes are often capable of rich movement patterns and interaction with the environment while only possessing very simple nervous systems [36, 46]. Based on this, I hypothesize that the results of my work could provide insight into the fundamental mechanisms underlying the locomotion of simple organisms by interpreting their behavior as a computational problem.

When taking a higher-level viewpoint on the nature of certain aspects of cognition and computation, my results can offer additional, empirically validated arguments in the quest for understanding cognition in biological organisms. Indeed, I have given several examples of systems in which the computational aspects of locomotion are for the most part physical in nature, making the structures discussed here prime examples of the idea of embodied cognition. Moreover, my analyses allow to quantify the nature of the computation occurring in the substructures, most notably the controller and

the physical system. This viewpoint is in my opinion applicable to many of the interactions between the body, sensory inputs and early cognitive layers, but will probably not suffice to fully explain the complete array of cognitive capabilities of human-level intelligence.

When considering cognition as performing computation in the broadest sense, it is clear from my results that this computation is very much divided across the explicit linear control and the implicit non-linear transformations of interactions with the environment, mediated by the physical properties of the structures. Indeed, the idea underlying the principle of Physical Reservoir Computing is precisely that the range of possible dynamical systems which can be used for computation is extremely broad, as are their properties regarding non-linearity or memory.

The significance of the combined feedforward and feedback controller presented in Section 5.3.3 is yet to be addressed. Two questions come to mind: Is there a biological analog of this architecture and are there practical applications of this method?

The answer to the first question is that there are examples of layered control systems in human motor control. Indeed, the cerebellum fine-tunes motions based on sensory input and control signals from higher level brain regions. This process is architecturally similar to the combined kinematic (high-level) and feedback (sensor feedback) approach in Section 5.3.3. Furthermore, it is known that the cerebellum processes reward signals and also communicates with the basal ganglia [16, 77, 153].

This leads to the second question. It turns out that various domains may benefit from the proposed layered control architecture. One obvious domain is robotics, as the dynamics of a robotic platform are often only known approximately. Therefore, one possible use of the learning rules is mitigation of model accuracy issues. However, the most appealing application seems to be that the method can be used to design systems that can adapt to minor changes of their dynamics through time. Such effects appear in multiple domains due to temperature fluctuations (e.g. in Photonic Reservoir Computing), material fatigue.... The presented learning rules are highly suited for this type of problems, as they can be implemented at a very low level with little computational requirements.

5.6 Conclusion

Hebbian theory is a well-established theory to explain synaptic plasticity between neurons. Over the years, many variations of the basic learning rule have been developed. Each of these had a specific application, ranging from

unsupervised feature extraction to reinforcement learning. In this chapter, I have presented Hebbian-like learning rules in which the synaptic plasticity is based on the correlation between the presynaptic neuron states and an exploratory noise signal. The plasticity is modulated by a reward signal, resulting in a learning method that maximizes the expected reward of a trial.

After discussing how to extend an existing rule to instantaneous reward based learning in input-driven Recurrent Neural Networks, I focused on distal reward learning in Recurrent Neural Networks and compliant robots. The main conclusion is that Reward Modulated Hebbian plasticity provides a simple, yet effective tool for bridging learning in Recurrent Neural Networks and compliant robotics, thus strengthening the belief that highly compliant robots can benefit from a computational approach to control problems.

6
The ReCTeR Robot

This chapter describes the design of a compliant tensegrity robot, nicknamed ReCTeR (Reservoir Compliant Tensegrity Robot). ReCTeR is an untethered, compliant tensegrity robot constructed in the context of this dissertation to study the practical aspects of tensegrity robot design and control.

ReCTeR was designed and built at Ghent University. Michiel D'Haene provided significant input on the electronic designs of the robot. Hardware tests were performed at Ghent University and the NASA Ames Research Center. More precisely, the robot was constructed at the Reservoir Lab (Electronics and Information Systems department) and the motion capture experiments at Ghent University were performed using the equipment of the Institute for Psychoacoustics and Electronic Music (IPEM). The hardware was tested at the NASA Ames Intelligent Robotics Group (IRG, Code TI). For motion capture, the equipment and lab space of the Human Systems Integration division (Code TH) was used.

This chapter is organized as follows. I first discuss the motivation and objectives of ReCTeR in Section 6.1. Next, I review a number of related robot designs to illustrate the original aspects of my robot design in Section 6.2. Section 6.3 then presents the hardware design of ReCTeR, followed by an overview of its features in Section 6.4. This leads to a series of experiments in Section 6.5. Finally, I present my conclusions in Section 6.6.

6.1 Motivation & Objectives

After a number of initial experiments in simulation, it quickly became clear that a physical prototype of an actuated tensegrity robot would be desirable. As only a few complex tensegrity robots have been built to date, this platform should provide valuable information about the physical limitations and issues

of compliant actuated tensegrities. ReCTeR should also serve as a validation of the Physical Reservoir Computing principle.

As such, I began the design and construction of ReCTeR in 2011. The following set of design goals were put forward at that time:

Cost
The cost of materials should be less than 5000 EUR.

Untethered
ReCTeR should be fully untethered. Wires would impede locomotion experiments and would have a significant influence on the dynamics of the structure. The wireless interface should provide low-latency access to sensor data and accept motor commands at a rate of at least 50 Hz.

Sensors
The simulation experiments relied on spring-cable assembly tension measurements, although the Physical Reservoir Computing approach can handle various types of sensor data. However, to minimize the reality gap between the simulations and the hardware, it is desirable to have similar sensors available on the robot.

Actuation
The robot should implement the spring-cable assembly model described in Section 2.3. As each motor has a global effect, underactuation (in the sense that not each spring-cable assembly is actuated by a motor on a bar[1]) is permitted. The motors should be able to significantly deform the robot in a dynamic regime, ideally allowing it to actively fold.

Distributed
Each rod of ReCTeR should be electrically independent, such that the platform can be extended to other configurations or provide a fallback mode in case of failure.

Compliance
ReCTeR should be *as compliant as possible*. This is a rather ambiguous statement which merits some clarification. The design should maximize flexibility, while guaranteeing that the robot does not collapse under its own gravitational load.

[1] An actuated spring-cable is a spring-cable assembly attached to a motor on a bar. Similarly, an actuated bar contains end caps with a motor that can change the rest length of spring-cable assembly. See Section 2.3 for the general actuated spring-cable model.

Robust
The robot should safely handle extreme or experimental motor commands and should be robust against moderate drops and impacts. In practice, this means that any motor command should not result in mechanical or electrical failure. Moderate drops are drops in the order of the size of the robot. For example, the robot should survive accidental falls when picked up.

Lightweight
A person should be able to carry the robot.

Safety
The actuated spring-cable assemblies should not pose a safety risk.

Long Battery Life
It should be feasible to validate learning approaches. In practice, an individual active battery life of more than 30 min is required as the variance between the battery supplies of the individual bars reduces the global runtime. In addition to this, live battery switching is desirable for longer experiments.

I will recall these objectives in Section 6.4, which discusses the features of the physical platform in its current state.

While the design requirements (distributed & untethered) allow ReCTeR components to be rearranged in various topologies, I specifically targeted the tensegrity icosahedron configuration from the start. The reason for this is that this topology is symmetric, can be folded into a flat star shape and has a relatively small number of members. The compressive elements are also spaced far apart, which allows for large shape deformations without the risk of colliding bars [71]. More spherical structures exist (Figure 6.1), but they are generally shell-like (similar to a geodesic dome) with all members closely knit together tangent near its surface. In Chapter 2, I have discussed a minimal 6-strut tensegrity structure (also called a tensegrity tetrahedron). This topology (Figure 2.1) has fewer members than the tensegrity icosahedron, but has a smaller internal volume and its large faces make it unfit for locomotion.

6.2 Related Platforms

Two tensegrity robots have had a particular influence on ReCTeR's design and a brief overview of these platforms allows to situate ReCTeR's development.

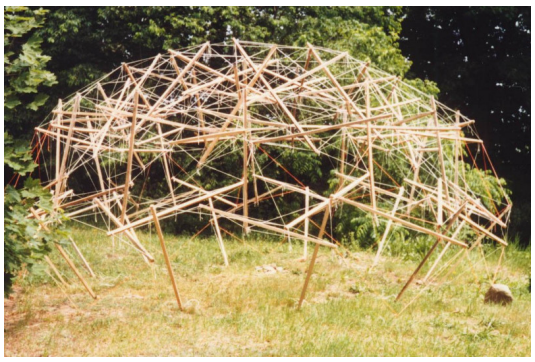


Figure 6.1: A tensegrity dome structure. Note that all the members are tightly packed in the shell of the structure. Shell-like tensegrities tend to be easy to build, because one can generally build such structures in tension. However, such a topology is not ideal for tensegrity robotics based on actuated compressive members, due to the relatively small range of motion (without risking bar-to-bar collisions) of individual members. (Credit: Bob Burkhardt)

6.2.1 Cornell IcoTens

Simon Fivat at The Creative Machines Lab at Cornell University has developed the IcoTens robot [55]. As its name suggests, the IcoTens is an icosahedron tensegrity robot. More precisely, it has a total of 24 members, 12 of which are actuated. This is similar to the proposed initial configuration of the SUPERball robot introduced in the next chapter.

The end caps consist of small, off-the-shelf electrical motors combined with a small lithium battery and electronics. These elements are held together by custom 3D-printed parts and the robot had a diameter of 0.62 m. Some images of the IcoTens are available at `http://creativemachines.cornell.edu/icotens`. Unfortunately, the development of this platform seems to have ceased and only limited information about its capabilities is available. It is reported to achieve a forward velocity of $29\,\mathrm{mm\,s^{-1}}$.

Due to its similar configuration, the IcoTens provided initial inspiration for ReCTeR. However, as will become clear in the next sections, ReCTeR's design represents a significant improvement over the IcoTens platform. For example, the actuated wire routing mechanism on the IcoTens is problematic. I have tested 3D-printed enclosed end caps similar to the IcoTens's during the initial design phases of ReCTeR. However, such designs have proven unreliable as it is hard to precisely guide thin wires when they go slack. ReCTeR therefore only uses a minimal number of 3D-printed parts and instead engi-

neering plastics (POM/PTFE) were machined for smooth surfaces in contact with actuated wires. Moreover, ReCTeR's different actuation pattern allows for efficient use of low power electric motors, without working against the prestress in the structure.

6.2.2 Pneumatic & SMA Rolling Tensegrity

Shibata, Shinichi and Koizumi first presented a simple icosahedron tensegrity robot with shape memory alloy (SMA) actuators [156]. This structure was able to move by slowly changing its shape. In 2012, they presented a new design using 24 pneumatic actuators as members [73, 98]. This tethered robot was not equipped with sensors, but did show interesting locomotion results (rolling). Due to the different nature of the actuators and the omission of sensors, ReCTeR does not imitate any design feature of these robots. However, their locomotion results provided insights about which kinds of deformations ReCTeR should be able to handle and what could be expected of its capabilities in practice.

6.3 Hardware Design

The hardware description of ReCTeR is split into two subsections: the mechanical design and the electronics. The schematics of the electronic designs are provided in Appendix A.

6.3.1 Mechanical Design

6.3.1.1 Structure

ReCTeR is based on the tensegrity icosahedron structure with 6 compressive members and 24 tensile elements. In addition to this, the robot has 6 actuated spring-cable assemblies. The robot is thus underactuated, as only 6 degrees of freedom can be controlled. Disregarding the rigid body modes, twisting of the compressive members and slack spring-cable assemblies, the robot has 24 degrees of freedom[2]. This considerable level of underactuation is a deliberate choice to keep the robot's cost, mechanical complexity and mass down.

The actuated members run through the robot as demonstrated in Figure 6.2. This allows the robot to retain its shape while powered down and to optimally use the available actuation power, as discussed in Section 2.6.

[2]Each bar has 5 effective DOF, for a total of 30 in a 6-bar robot. Subtracting rotation and translation of the whole robot results in 24 DOF.

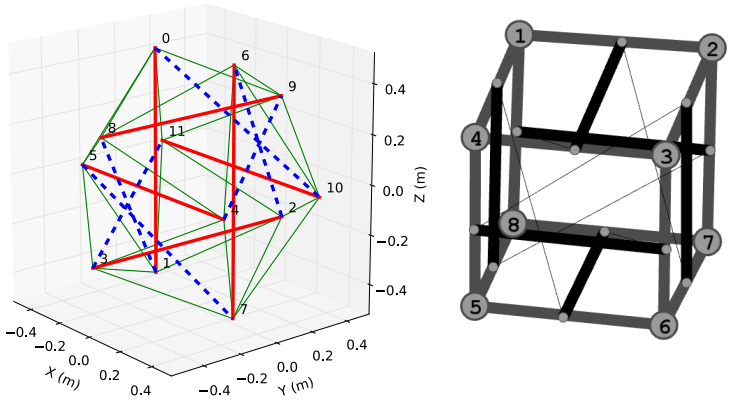


Figure 6.2: ReCTeR connection pattern. Left: The thin green lines are passive spring-cable assemblies (outer shell), the thick full red lines are struts and the dashed blue lines are actuated spring-cable assemblies. Right: A different representation of the connectivity, showing how the actuated members are dual to the struts. The large circles are (approximately) equilateral triangular faces connected by edges representing the end caps (small circles) over which the robot has to roll to reach an adjacent face. The thick black lines represent the struts and the thin lines are the actuated springs, which form the same spatial structure as the struts in the representation on the left. For example, assign the number 5 on the right representation to the equilateral triangular face defined by end caps (1,3,5) on the left image. Analogously, assign the number 6 to the face (3,4,7) on the left. Rolling from face 5 to 6 (right representation) thus occurs over end cap number 3 (left representation).

ReCTeR has a total mass of 1.1 kg (batteries included), which is achieved by using carbon fiber struts (8 mm outer diameter). The tensegrity principle allows to make effective use of the axial strength of the carbon fiber.

Three of ReCTeR's struts are actuated (two actuators each), while the other three are fully passive and sensorless. The total masses of the struts are 0.05 kg and 0.270 kg, for the passive and active struts respectively. The six actuated springs are selected such that each end cap has exactly one actuated spring attached to it. By further requiring the pattern to be symmetric and preventing parallel struts from being connected, exactly one pattern was found (up to a mirror symmetry). It can be seen that this connection pattern allows for large shape deformations, as the actuated spring-cable

assemblies have a larger workspace compared to shell actuation. This was shown in Figure 2.8 (Section 2.6) for which I compared the effect of various actuator patterns. Internal connections are significantly longer than shell elements and a single motor can thus operate over a wider range. ReCTeR is slightly asymmetric due to design revisions. The passive bars are all of equal length (1 m), while the actuated bars have lengths of 0.92 m, 1.01 m and 1.06 m.

Figure 6.2 also shows a different representation of the connection pattern. More precisely, it connects the centers of each equilateral triangle in the tensegrity icosahedron with its adjacent equilateral triangular faces. It can easily be seen that the centers of the equilateral triangles form a cube. The edges of the cube correspond to the end cap over which the robot has to roll to move to an adjacent equilateral triangle. In this representation it is easy to see that the actuated pattern is a dual to the strut connectivity pattern and is therefore an effective way to deform the structure with low power actuators.

Following the rules set out in Section 6.1, ReCTeR should be as flexible as possible while not collapsing under its own gravitational load. This is achieved by using inline, low stiffness tension springs. More precisely, the passive and active cables have inline springs with low spring constants at $28.4\,\mathrm{N\,m^{-1}}$ and $81\,\mathrm{N\,m^{-1}}$, respectively. As a result, the natural frequencies of oscillatory modes for the structure are on the order of a few Hz. While it is not necessary to add springs to the actuated cables, I found that removing the stiffer springs of these cables results in a significant reduction in compliance of the structure, which can be problematic during impact.

Using the analysis techniques presented in Chapter 2, the modal stiffness of ReCTeR can be studied. Figure 6.3 shows the modal stiffness of the robot as a function of the spring constants of the outer shell elements. The actuated tensile elements running through the robot were not taken into account, as they go slack when the robot is powered down. In this figure the pretension is kept constant at the levels used for the experiments described in this chapter ($\approx 10\,\mathrm{N}$). Note that for very low stiffnesses, some modal stiffnesses become zero, due to negative rest lengths of spring-cable assemblies. However, these flexes are not finite due to the irregularities of the structure. The figure illustrates that to stiffen the most flexible modes of the structure, significantly stiffer springs are needed. To confirm this, ReCTeR has been reassembled with stiffer springs ($\approx 80\,\mathrm{N\,m^{-1}}$). As can be expected, the result is that the modified robot stands more upright under gravitational loading. However, this modification impedes folding as this plasticly deforms the new springs and requires excessive motor power. The low $28.4\,\mathrm{N\,m^{-1}}$ spring stiffness is thus a choice that optimizes the flexibility of the robot to maximize the effect

of the actuators, while allowing the robot to be free-standing.

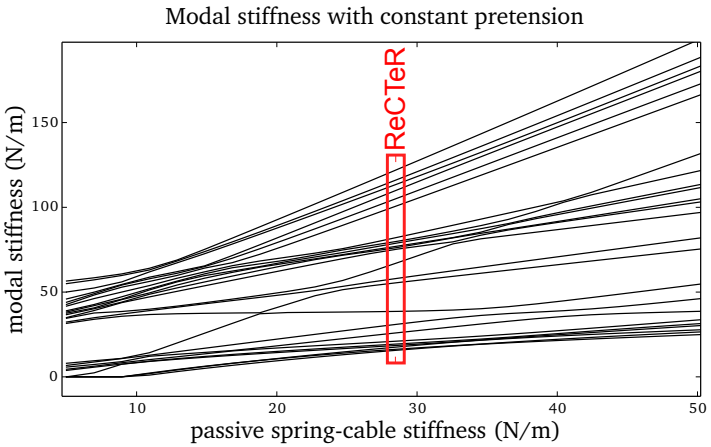


Figure 6.3: The modal stiffness of ReCTeR as a function of the spring constants of the outer shell tensile members (the actuated spring-cable assemblies are not taken into account) for a constant level of pretension. This figure shows that to stiffen the most flexible modes of the structure, significantly stiffer springs are needed. For the experiments described in this chapter, $28.4\,\mathrm{N\,m^{-1}}$ springs were used for the passive spring-cable assemblies (indicated by the box).

I have used 200 N UHMWPE[3] wires for the passive outer shell cables. The actuated spring-cable assemblies are 70 N UHMWPE wires (0.13 mm diameter).

6.3.1.2 End Cap

ReCTeR is equipped with low power DC motors (4.5 W brushed DC, Maxon 216 000) with a single stage plastic gearbox (4.4:1, Maxon 112 862). It is crucial to prevent the tensile forces on the actuated springs from exerting an excessive radial load on the motor axis. Therefore, two miniature ball bearings secure the motor axis (one is mounted inside the bottom of the spindle, one is mounted at the end of the spindle). The current design can shorten the actuated cables at a rate of $0.3\,\mathrm{m\,s^{-1}}$ (under 10 N of tension) and I have observed active unwinding speeds of over $0.6\,\mathrm{m\,s^{-1}}$. The estimated effective (gears and bearings) nominal motor output is 3.5 W. Stated differently, the power-to-weight ratio of ReCTeR is relatively low (approx. $20\,\mathrm{W\,kg^{-1}}$).

[3]UHMWPE: Ultra-high-molecular-weight polyethylene. The fibers are commercialized under the Dyneema and Spectra brand names. While this type of wire can resist high tensile forces, it is subject to significant creep.

I have opted for an open node design to prevent wires getting caught. To this end, a PTFE cap fits tightly around the top and bottom of the spindle, separating the ball bearing and axis from the cable and providing a low friction surface for the wires when the robot is highly deformed.

Figure 6.4 shows an exploded view drawing of a ReCTeR end cap. In this figure, the components on the right fit on top of those on the left. Consider the parts starting from top (top right) to bottom (bottom left).

A protective silicone cap protects the tip of each bar and provides additional friction (Shore 36). The top of the end cap also contains a custom force sensitive resistor to sense ground reaction forces. Next, a 24 mm diameter PCB contains the microcontroller (Microchip 24F), a 24 bit ADC (Analog Devices AD7192) for the strain gages and a magnetic motor position encoder (AMS AS5050). Another circuit board contains the voltage references, solder terminals for the strain gages and a miniature ball bearing for the wire spindle. The strain gages are glued to a custom aluminum part which snaps around this circuit board (see Figure 6.5[D]). The wire spindle is machined out of POM[4] and as is secured with two ball bearings, the bottom one of which is press fitted into a custom aluminum part on top of the gear box. As discussed before, this double bearing design is crucial to protect the actuator against impact forces (no radial loading of the motor axis) and allows the use of light weight plastic gears.

The wire (0.13 mm UHMWPE) is attached to the spindle and can slide over two PTFE surfaces, one above the gear box and one underneath the strain gages. The wire spindle has a 5.5 mm diameter, which results in an actuated spring-cable stall force of approximately 25 N. The custom motor driver PCB is soldered directly to the bottom of the actuator and has the same diameter as the DC motor (17 mm). A small PCB directly underneath the motor driver connects the end cap electronics to the center PCB. The whole end cap assembly slides onto a hollow, square carbon bar (8 mm outer, 6 mm inner width) and the wires between the center module and the end cap electronics run through this bar. The PCB underneath the motor driver provides a single connection point between the center module and the end cap. Note that a press fit is sufficient to join the end caps to the main bar as this whole assembly is in compression.

6.3.2 Electronics

Each of the robot's actuated bars contains 11 small circuit boards to fit all the required electronic components. The center module contains the most complex PCB (Figure 6.5[B]). Two batteries connect to this board: the main

[4] Polyoxymethylene

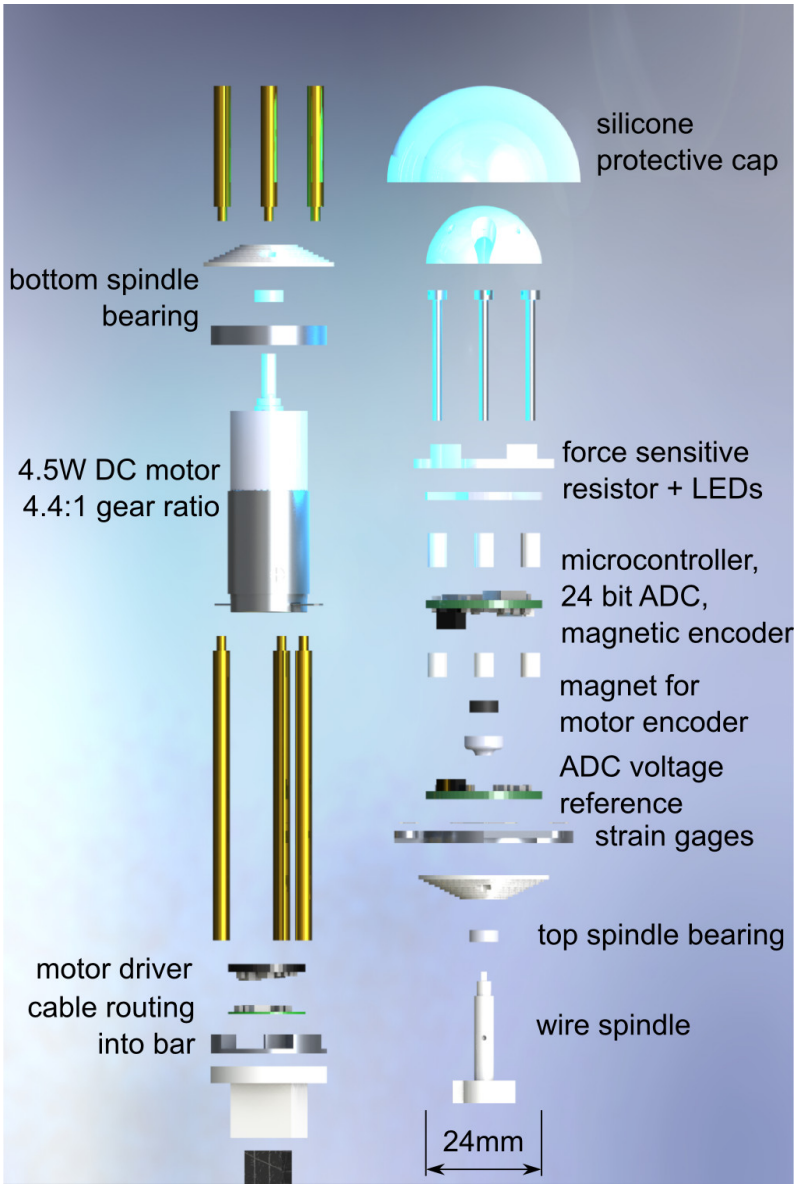


Figure 6.4: Exploded view drawing of a ReCTeR end cap. The components on the right go on top of those on the left. A detailed description of the assembly is given in the main text (Section 6.3.1.2).

battery (a single cell 3.7 V lithium-ion) and a small backup battery (138mAh single cell lithium-ion). This configuration allows live battery switching without loss of calibration. In addition to this, the center also allows the robot to be powered externally and can charge both batteries.

Processing power is provided by a microntroller from Microchip (PIC 24F). This device distributes the messages received over the wireless interface to the end caps, handles the IMU data (also located on this module) and controls the DC/DC converters and power switches. More precisely, the board boosts the battery voltage to 12V for the actuators in the end caps and provides, stable 5.5 V and 3.3 V lines for the electronics and strain gages.

The end caps each host 5 boards. One of these boards only serves to connect the wires coming from the center modules from within the bar to the 4 other boards. The next board contains the motor driver and fits directly onto the bottom of the actuator. The next PCB fits into the aluminum strain gage mount (Figure 6.5[D]) and hosts the precision voltage references for the force transducers. On top of this board, the second to last PCB contains the magnetic motor encoder, the 24 bit strain gage ADC and microcontroller. The last PCB contains the force sensitive resistor used as a ground reaction force sensor and a pair of bright LEDs. The end caps transmit their state information (motor position, strain gages...) at 200 Hz to the center module over a serial connection. The center module transmits the main state variables of the full bar in a single 32 B message.

6.4 Features

I now recall the objectives set out in Section 6.1 and discuss to what extent ReCTeR achieves these goals.

Cost

The cost of the passive bars is negligible. On the other hand, the cost of the actuated struts is defined by the motors, the electronics and the materials. The price of the Maxon motors is approximately 100 EUR/motor (with gearbox). It is hard to accurately estimate the material cost, as most materials were bought in bulk. However, it is safe to assume that this cost is under 100 EUR/bar. The machining was done in-house and this cost is therefore not taken into account. The production cost of the circuit boards was 44 EUR/bar, while components account for approximately 100 EUR. Therefore, the final total cost per bar excluding manufacturing is below 500 EUR.

Untethered

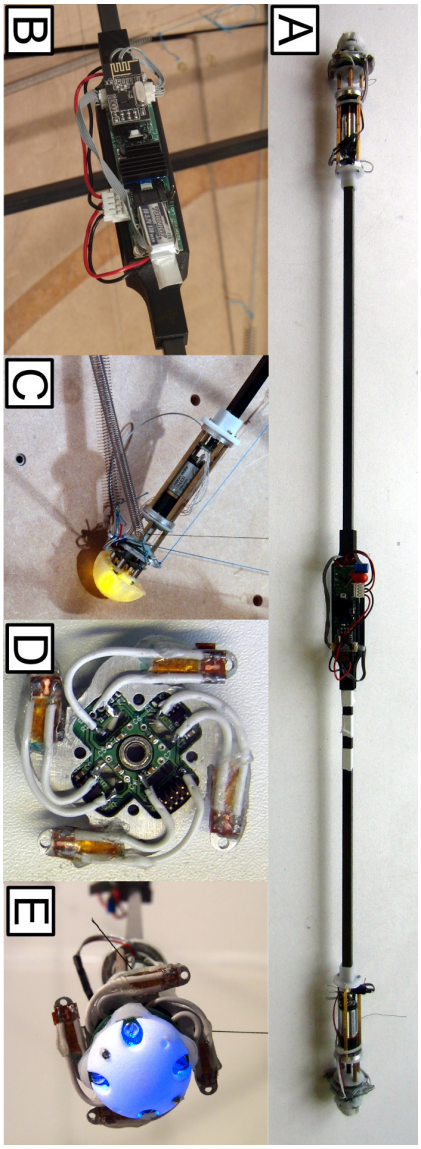


Figure 6.5: **A)** Bar design with all wires embedded into the carbon fiber bar. The center module hosts the battery, wireless interface and IMU. **B)** Close-up of the center module, with the RF module on the left and the backup battery on the right. The main battery sits in line with the carbon tubes connecting the center module and the end caps. This keeps the center of mass of the bar collinear with the carbon tubes and end caps. **C)** Deployed end cap. **D)** Close-up of the strain gage force transducers. The wire spindle fits into the ball bearing. **E)** Top view of the end cap with the protective silicone cap removed.

The robot is wireless and an experimental control/measurement rate of 100 Hz was achieved.

Sensors
ReCTeR has 24 spring-cable force transducers (4 per actuated end cap) and each bar has a 6 DOF IMU. A custom force sensitive resistor in the end caps allows to detect impact forces.

Actuation
A smart connectivity pattern with 6 actuated spring-cable assemblies allows for efficient use of the available motor power. As the actuators to not need to work against the main tension connections, the robot maintains its original icosahedron shape when powered down and the full actuation power can be used to deform the structure.

Distributed
The programming of each bar and end cap is identical and the bars are fully independent (mechanically and electrically). Additionally, it is possibly to reprogram the parameters (ID, sampling rates...) wirelessly. A generic ROS[5] interface was developed to efficiently control up to 16 bars.

Compliance
This was discussed in detail in Section 6.3. The chosen spring stiffnesses result in a high level of passive compliance and efficient use of the actuators. At the same time the structure is stiff enough to resist collapsing under its own gravitational load.

Robust
The experimental section will provide more details on this issue. ReCTeR has been dropped multiple times from up to 1 m without mechanical failure. The robot has been disassembled for transport at various occasions, again without causing mechanical failure. The low power DC motors are capable of plasticly deforming the springs, but cannot break the wires or end caps.

Lightweight
ReCTeRs total mass is 1.1 kg. The batteries account for 0.15 kg. An actuated bar has a mass between 0.25 kg and 0.3 kg.

Safety
The low power actuators do not pose a safety risk and the amount of elastic potential energy stored in the springs is too small to cause harm.

[5]Robot Operating System: A common open source robot middleware developed by Willow Garage and available at `http://ros.org`.

Long Battery Life
An effective battery life of 1 h was achieved using Panasonic NCR-18650A lithium-ion cells (3100 mA h). In standby mode (all sensors active) the battery life increases up to multiple hours.

6.5 Experiments

After presenting the hardware design and features of the ReCTeR robot, I now discuss the main experimental results obtained with this platform. As will become clear, the main experimental focus was on validating the simulators, verification of the hardware capabilities (e.g. folding) and proof-of-concept of the feedback controllers studied in the previous chapters. This implies less attention to sustained locomotion. The reason for this is that after the experiments discussed here, I redirected my focus toward the development of the SUPERball robot (studied in depth in the next chapter). SUPERball is specifically aimed at dynamic locomotion and it therefore makes sense to use ReCTeR as a validation and testing platform.

6.5.1 Experimental Setup

For the experiments which report position information, an active marker[6] motion capture setup was used. The hardware setup was based on the PhaseSpace Impulse X2 system with 15 active markers and 11 cameras. More precisely, each passive strut was fitted with 2 markers, while each actuated strut received 3 markers.

6.5.2 Demonstration of Capabilities

Figure 6.6 demonstrates the capabilities of ReCTeR. The top images (A & B) show the robot in deployed and actively folded state. The next sequence (C) demonstrates the rolling (open loop) from right to left, while mostly maintaining its spherical shape. In contrast with this, sequence (D) illustrates open loop rolling through large shape deformations. Finally, the last series of snapshots (E) shows the robot being dropped to demonstrate its robustness.

[6] In a typical passive marker system, light (typically infrared) is emitted by an external source and reflected by the markers. The motion capture system detects and tracks the highly reflective markers. An active marker system directly tracks light sources (LEDs) located on the subject.

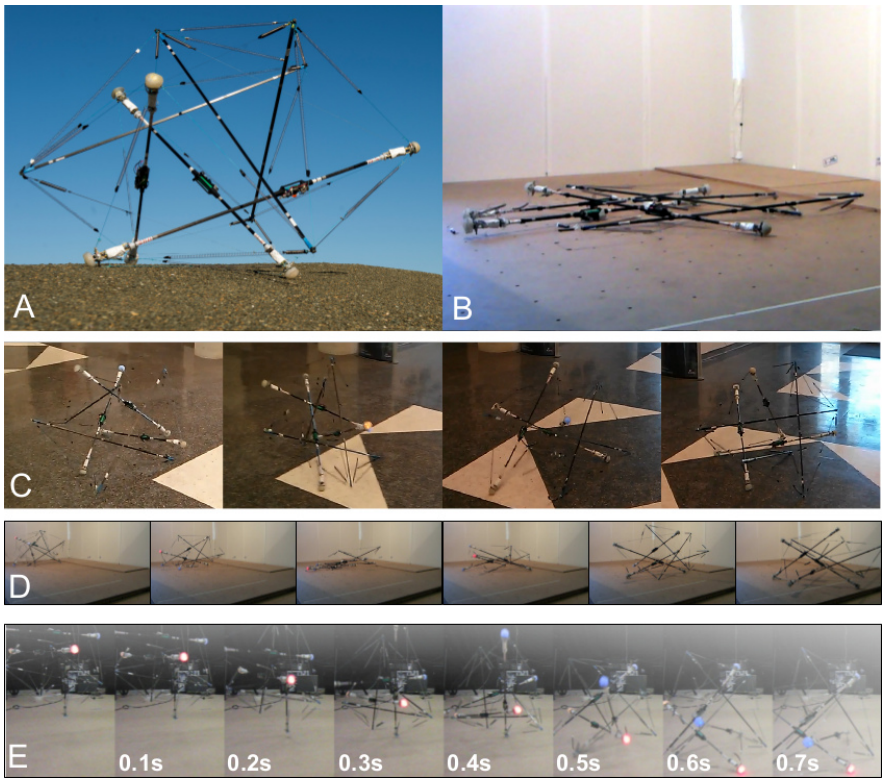


Figure 6.6: Overview of ReCTeR's capabilities. **A)** Robot fully deployed on the NASA IRG RoverScape (Credit: NASA). **B)** Actuated folding of the robot. **C)** Rolling (right to left). **D)** Rolling with large shape deformations (left to right). **E)** Drop test of the structure. The robot has proven robust against repeated drops of up to $1\,\mathrm{m}$ ($\approx 4.4\,\mathrm{m\,s^{-1}}$). For reference, a tensegrity probe for Titan would sustain an estimated terminal velocity of $15\,\mathrm{m\,s^{-1}}$ which is equivalent to a $10\,\mathrm{m}$ drop on Earth (see [167] and Chapter 7).

6.5.3 ReCTeR Kinematics

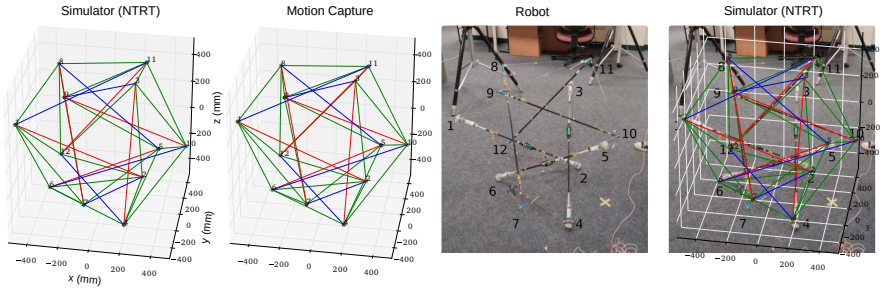


Figure 6.7: Verifying the equilibrium state of the hardware prototype and the NTRT simulator. A maximum error (Euclidean distance) of $0.04\,\mathrm{m}$ was observed over all the end cap positions. Note that the error appears larger in the image on the right due to camera distortion.

After an initial test of the equilibrium state of ReCTeR in simulation and hardware (Figure 6.7), I focused on the first validation experiment: A comparison of the forward kinematics of the Euler-Lagrange (Section 2.8.1) and NTRT simulators (Section 2.8.2) and motion capture data from ReCTeR.

The six-strut ReCTeR robot was placed on one of its triangular faces and two of the top spring-cable assemblies were actuated, as shown in Figure 6.8. The vertical displacement of an end cap not directly actuated by one these two members was tracked. The incident strut was suspended in the air by a total of 10 spring-cable assemblies.

The lengths of the two actuated spring assemblies were varied from the point of no tension in the given configuration (slack) to $0.32\,\mathrm{m}$ beyond this length. Each range was sampled at 10 equally spaced lengths, resulting in 100 measurement positions in total. These ranges were manually tuned to maximally deform the robot, without causing it to roll. This experiment was repeated three times, with no significant difference in the observed displacements.

The average observed difference between the motion capture data and the Euler-Lagrange simulator was $6.5\,\mathrm{mm}$. For NTRT, an average error of $15\,\mathrm{mm}$ was obtained ($0.5\,\%$ and $1.3\,\%$ of the robot's diameter respectively).

6.5.4 Dynamics

Next, the dynamics of the NTRT simulator and the ReCTeR hardware were compared. This experiment was designed for two purposes: To verify that the simulator can replicate ground interactions, and second, that it can

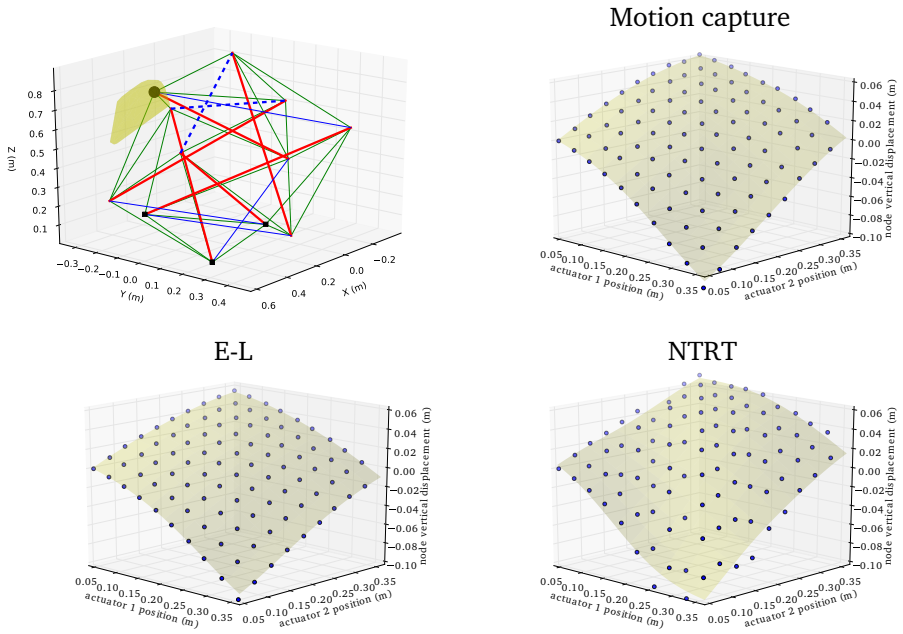


Figure 6.8: Kinematic comparison of the Euler-Lagrange and NASA Tensegrity Robotics Toolkit simulators and ReCTeR motion capture data. The top left plot shows the experimental setup. The rest length of two actuated spring-cable assemblies (dashed lines) is modified. The full range of motion of the tracked end cap during the experiment is shown in light yellow (convex hull). The end caps indicated by small black squares are on the ground. The 3 other plots show the vertical displacement of the end cap indicated by the large black dot in the top left plot as a function of the lengths of the two actuated cables. The end cap of which the displacement is traced, is not directly actuated and is floating. The nodal displacement as a function of the actuator position is non-linear, even for modest displacements. Note that the leftmost point $(0.05,0.05,0)$ is the reference point, as the displacements are relative to this initial state.

simulate the conversion of potential energy into kinematic energy when a spring is released. The experimental setup is shown in Figure 6.9. The robot initially has a non-minimal ground contact (four end caps on the ground instead of just three), and three of the actuated spring-cable assemblies are tensioned. Next, one of the tensioned, actuated springs is loosened by its actuator, which causes the robot to roll over.

As the experiment also depends on the initial state of the robot, the observed initial state from the motion capture data was copied to the NTRT simulator. The ReCTeR model in the NTRT simulator was then released from this initial configuration, allowing it to reach the simulated, predicted equilibrium. Therefore, the simulator first had to match the kinematics of the hardware platform. The recorded motor positions from the physical test were then applied into the simulator, causing a similar rolling-over motion. A time averaged error of the end caps' vertical positions of less than 5 % of the robot's diameter was observed for all end caps.

In addition to this experiment, I performed dynamic and static measurements of a single actuated ReCTeR strut suspended by four spring-cable assemblies — two passive ones attached to the ground and the two actuated ones to the ceiling. The results of these experiments indicated a close match between NTRT, the Euler-Lagrange simulator and the hardware for the lowest resonance frequencies and the kinematics. The simulators could not accurately reproduce the very low amounts of damping of the physical system. However, this does not appear to be a significant issue for the assembled robot, due to the increased damping due to friction between the robot and the environment.

6.5.5 Physical Reservoir Computing

This section presents an implementation of the Physical Reservoir Computing (PRC) principle on the ReCTeR hardware (see Section 4.4). Closed-loop feedback control is used in which the motor signals are generated by a Matsuoka oscillator. This demonstrates a successful adaptation of the simulation results from Chapter 4 to a physical platform (ReCTeR), with similar learning times and robustness.

A static linear feedback controller is designed, which robustly generates a set of desired oscillatory motor signals after a short learning phase. For this experiment, the target spring-cable rest lengths (ℓ_i) are generated using a random Matsuoka oscillator [117] (see Section 4.3.1 for oscillator parameters and motivation). These signals represent the desired actuator signals. I manually scaled the target signals such that the resulting behavior corresponds to a motion pattern with large shape deformations, while keeping the

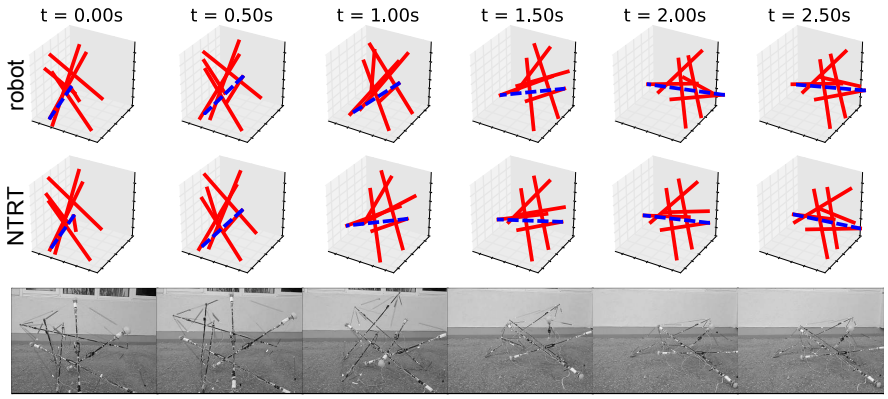


Figure 6.9: Comparing robot and NTRT dynamics. The tensioned spring-cable assembly indicated by the dashed line is released ($0.32\,\mathrm{m}$ to $0.535\,\mathrm{m}$ at $0.6\,\mathrm{m\,s^{-1}}$), causing the robot to topple. Two other actuated members are also tensioned, while the other three actuated springs are at their initial lengths, resulting in two slack springs. A time averaged error of the end caps' vertical positions of less than $5\,\%$ of the robot's diameter was observed for all end caps.

physical structure from moving too fast (as this impeded motion tracking). The resulting gait was a slow crawling motion, which allowed motion-capture tracking of the full experiment. Note, that the robot's behavior was evaluated using the motion capture setup described in Section 6.5.1, but that no motion capture measurements were used in the control loop.

The algorithm proceeds by first applying the target rest lengths to the actuators in an open loop setup, which causes the robot to start moving. Next, online learning is applied to approximate the desired signals based on the sensor readings. These approximations are the feedback signals. The ratio of open loop versus feedback signals is gradually decreased until the signals are generated by the feedback loop alone. At this point, the robot will robustly maintain the oscillatory patterns. The precise equations and parameters used in the experiment are provided in Chapter 4.

The controller feedback signals are obtained from ReCTeR's 24 force transducers. As these sensors are mounted perpendicularly to the robot's struts, the output values depend on the angle of attack and the tension of the attached spring-cable assembly. Thus, the sensors provide a readout of the robot's state similar to to the state observations in RC, albeit non-linear.

Figure 6.10 shows the result of an experiment in which I first outsource the motor signal generation to the feedback loop by online learning of the feedback weights. After training, I disturb the system (lifting the robot

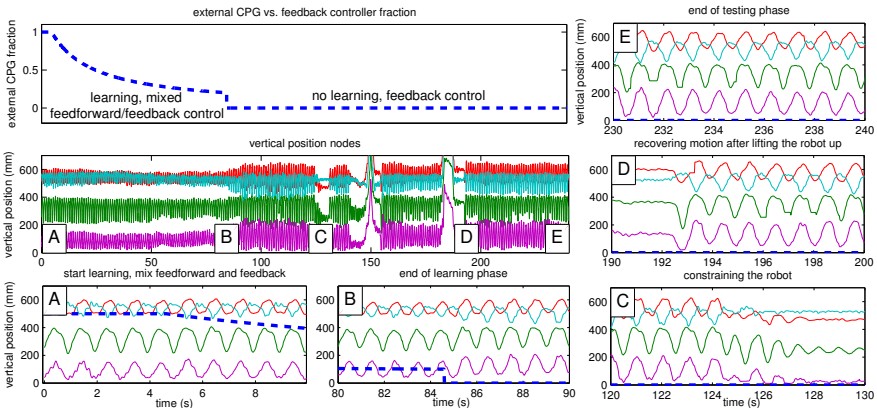


Figure 6.10: Fast online learning of a static feedback controller for a Matsuoka oscillator on the ReCTeR robot based on uncalibrated strain gauge sensors. The top left plot shows the fraction of feedforward vs. feedback control. During learning, both the feedback and feedforward controllers (training signals) are active. The influence of the open loop feedforward controller decreases and when its fraction is below 0.2, learning stops and only the trained feedback controller is active. The left plot on the second row shows the vertical coordinates (in mm) of the four end caps with the largest vertical displacements as a function of time. The five surrounding plots are details of this plot, showing the different training and testing phases. **A)** Fully open loop control. **B)** Switch from partially open loop and feedback control to full feedback control, learning stops. **C)** The robot is perturbed by pushing it down, preventing all movements. **D)** The feedback controller recovers after the robot was lifted up from the ground. **E)** The behavior of the robot after $250\,\mathrm{s}$ ($170\,\mathrm{s}$ closed loop).

up and constraining it). The robot stops moving in this case and switches back to its original mode of oscillation when it is released, demonstrating the robustness of the learned feedback controller, corresponding to my simulation results.

This experiment is a first demonstration of a simple and robust feedback control strategy implemented in both hardware and simulation for this class of untethered tensegrity robots. Additionally, this result shows the usefulness of tension sensors for tensegrity robot control. These PRC experiments are part of a continuous effort to develop low-level controllers for compliant robots that maximally exploit the robots' proper dynamics, and which allow to mitigate stringent sensor requirements. Many variations and extensions of the hardware experiment presented were discussed in Chapters 4 and 5. Only the fundamental result from Chapter 4 was reproduced as priority was given to the simulator validation experiments. The simulator validation results allow control strategies for the new robot platform presented in the next chapter to be developed in parallel to the hardware.Note that none of the experiments detailed in Chapters 4 or 5 make more assumptions about the hardware capabilities of ReCTeR than those needed for the experiment successfully reproduced here, except for the duration of the experiments.

6.6 Conclusion

ReCTeR, a compliant and untethered tensegrity robot, was introduced in this chapter. The design goals made it clear that this robot was designed to be a unique platform that significantly advances upon prior actuated tensegrity designs. As shown in the experimental section, the simulators were successfully validated by comparing the motion capture data of ReCTeR with simulation results. In addition to this, the basic control setup described in Chapter 4 was transferred to hardware to verify the statements made about Physical Reservoir Computing in simulation.

It might have taken the reader by surprise that only minimal attention was given to locomotion results. This was a deliberate choice as ReCTeR's capabilities have been sufficiently demonstrated and I decided to focus on the hardware and software development of SUPERball. SUPERball is a tensegrity robot under development at the NASA Ames Research Center and its design has to a large extent been inspired by ReCTeR. The next chapter revolves around this work.

7

SUPERball: Tensegrity for Space Exploration

In this last chapter, I present the work performed in collaboration with the Intelligent Robotics Group at the NASA Ames Research Center. I had the opportunity to visit the Intelligent Robotics Group between June 2013 and January 2014 to collaborate on the design of the SUPERball platform. As such, most of the work discussed in this chapter is the result of the joint efforts of the tensegrity team at NASA.

In particular, the mechanical designs and electronics were developed in collaboration with Jonathan Bruce (University of California, Santa Cruz/Universities Space Research Association) and Andrew Sabelhaus (University of California, Berkeley). For the controls and simulations, Atil Iscen (Oregon State University) and Jérémie Despraz (Ecole Polytechnique Fédérale de Lausanne) were my prime co-workers. Significant input also came from In Won Park (NASA Ames Research Center/Oak Ridge Associated Universities) and the team leads Vytas SunSpiral (NASA Ames Research Center/SGT Inc.) and Adrian Agogino (NASA Ames Research Center/University of California, Santa Cruz). To reflect that this chapter is the result of a collaborative effort, I will use the first person plural.

SUPERball (an acronym for Spherical Underactuacted Planetary Exploration Robot) is a next generation tensegrity robotics research platform. It is a terrestrial research prototype for concept mission development on Earth. While it incorporates various design aspects of ReCTeR, the main focus is different. ReCTeR is a prototype and proof of concept for locomotion and sensing in compliant tensegrities. SUPERball is a robust, modular and high powered platform aimed at dynamic locomotion and extensibility. This different focus is seen at all levels of the design. Significantly more importance is given to ease of assembly, safety factor, material selection...

I will begin this chapter with an overview of the anticipated advantages of tensegrity robotics for space exploration (Section 7.1). Next, I go over

some of the lessons learned from ReCTeR for robust tensegrity robot designs (Section 7.2). This leads to the mechanical and electronic design in Section 7.3. The mechanical design section starts with an overview of the design goals. Before presenting the conclusions (Section 7.5), I provide an overview of the various control strategies for SUPERball (Section 7.4). As the full robot is being manufactured at the time of writing, the control strategies are restricted to simulation results.

7.1 Tensegrities for Space Exploration

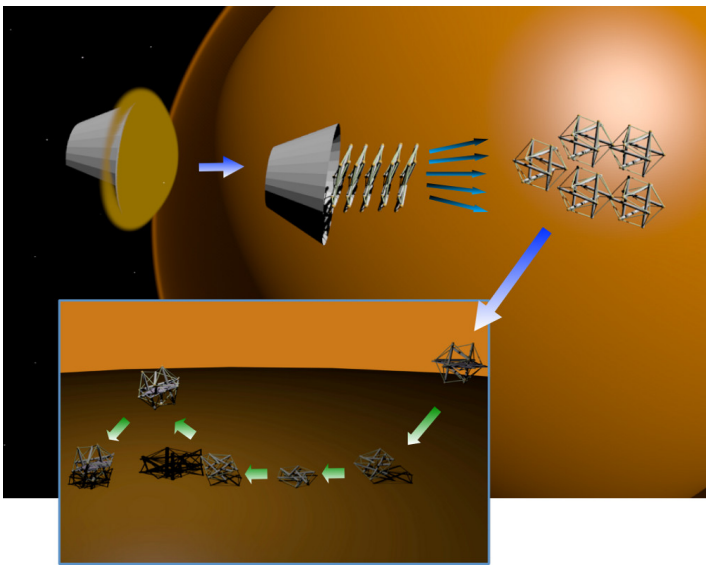


Figure 7.1: Mission scenario – Tightly packed set of tensegrities, expand, spread out, fall to surface of moon, then safely bounce on impact. The same tensegrity structure which cushioned the landing is then used for mobility to explore planetary destinations.

NASA mainly sponsors research into tensegrity robotics through funding from the NASA Innovative Advanced Concepts program (NIAC). Initial work within the context of this NIAC project [1], has shown that the controllable compliance and force distribution properties make for reliable and robust environmental interactions, such as those involved in landing and surface mobility during NASA missions. The initial financial backing for this research was provided in 2012 in the form of NIAC phase 1 funding awarded to Adrian

Agogino, Vytas Sunspiral and David Atkinson. In 2013, the funding was renewed for an additional two years in the form of a NIAC phase 2 award. The project was one out of only six projects obtained this renewed support[1].

A key goal for this work at NASA is to develop a tensegrity probe in which the tensile network can be actively controlled, enabling compact stowage for launch followed by deployment in preparation for landing. Due to their natural compliance and structural force distribution properties, compliant tensegrity probes can safely absorb significant impact forces, enabling high speed Entry, Descent, and Landing (EDL) scenarios where the probe itself acts much like an airbag [167]. However, unlike an airbag which must be discarded after a single use, the tensegrity also provides rolling mobility (Figure 7.1). This motion maintains the structure's original ability to safely absorb impact shocks, such as those that might occur during exploration of a planetary surface. This enables compact and lightweight planetary exploration missions with the capabilities of traditional wheeled rovers, but with a mass and cost similar to a stationary probe. As a result of the dual use of the structure, a tensegrity mission can have a high mass fraction between science payload and overall weight (as measured at atmospheric entry.) This leads to less expensive missions, or new forms of surface exploration that utilize the tensegrity's natural tolerance to impacts [167].

These two aspects — compact stowage and airbags — have been crucial to earlier missions and mission concepts. For example, the Apollo 15 Lunar Roving Vehicle had an intricate stowage mechanism (Figure 7.2), which had to be deployed by the astronauts on the lunar surface.

Airbags were used to facilitate the landings of the Mars Pathfinder and Mars Exploration Rover (Figure 7.3). Note that these airbags were made out of Vectran, the same material used for the spring-cable assemblies of SUPERball (Section 7.3) [76]. In this spirit, various soft robot prototypes for planetary exploration have been developed, most of which are either wind-driven designs inspired by tumbleweed or airbag-like [4, 7, 65, 95]. Figure 7.4 illustrates various designs considered by a NASA Langley Research Center [4]. If we turn to biology, it can be seen that nature provides relatively few examples of animals or plants using rolling as a main means of locomotion. Notable instances are tumbleweed, the wheel spider and some caterpillars. Some of these examples have led to capable robot designs [106].

Tensegrities do not exactly fit in any of these categories. Spherical tensegrities behave similarly to airbags at impact and will roll like a ball when made sufficiently stiff. They can also crawl at lower stiffness, which is of interest for rough terrain locomotion. At moderate and low stiffnesses, they

[1]The list of NIAC projects is available at `http://www.nasa.gov/content/niac-2013-phase-i-and-phase-ii-selections/`.

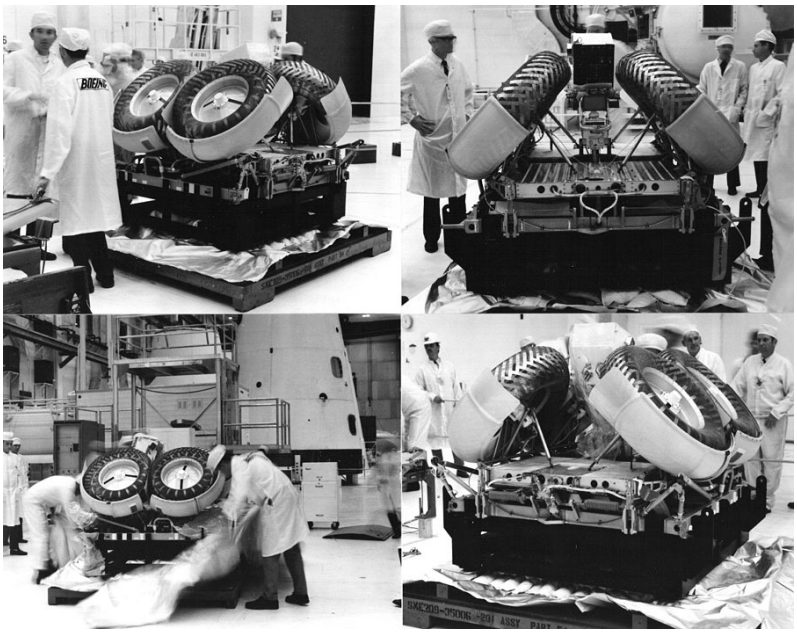


Figure 7.2: The Apollo 15 Lunar Roving Vehicle in its folded stated (Credit: NASA).

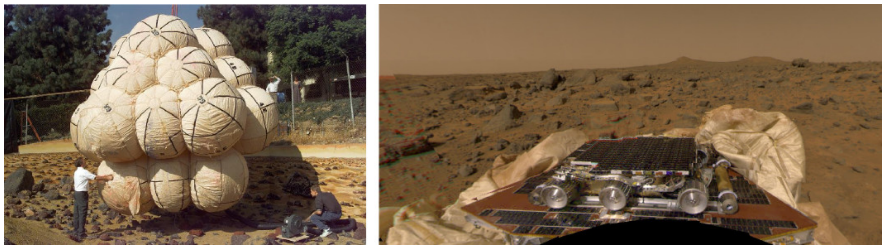


Figure 7.3: Left: Testing of the airbags for the Mars Pathfinder. Right: Deflated airbags of the Mars Pathfinder after landing in Ares Vallis in 1997 (Credit: JPL/NASA).

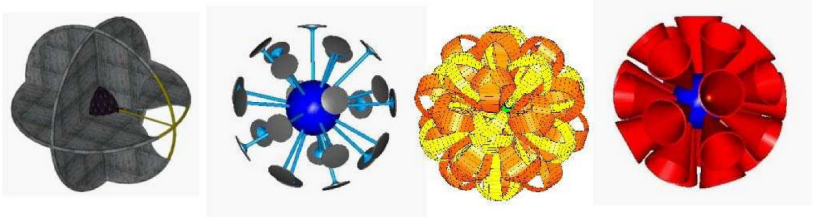


Figure 7.4: Prototypes of tumbleweed like structures for space exploration by NASA Langley [4] (Credit: NASA).

can conform to and grip the terrain, which differs from the rolling locomotion at higher stiffness. In addition to this, tensegrities are open structures, which allows the scientific instruments to directly interact with the environment. This shows that tensegrities are versatile systems, which can combine the advantageous properties of various types of structures.

7.2 Lessons Learned from ReCTeR

While ReCTeR exceeds its design goals — sensor feedback, locomotion and folding — it has a number of limitations, which prevent it from being used as a general modular tensegrity robot platform.

First, the lightweight design cannot transport any significant scientific payload, which is a major feature for any planetary exploration mission. For example, initial calculations showed that the scientific payload could represent up to 50 % of the atmospheric entry mass for a tensegrity mission [167]. In practice, this means that a considerably heavier robot is required to transport a meaningful payload.

The Mars Science Laboratory has a scientific payload (rover mass without mobility equipment) of 723.4 kg [101]. This is an extreme example and beyond the scope of the NASA Innovative Advanced Concepts project. However, significantly smaller scientific payloads have carried out fruitful projects. For example, the CubeSat picosatellites have a mass of up to 1.33 kg and have provided researchers the possibility to quickly design and deploy (in under 2 years) scientific equipment in space [31, 129]. Our current aim is therefore to design a robot capable of transporting a payload of this order of magnitude as it demonstrates the ability to carry a useful scientific payload, while allowing for testing in a normal laboratory setting. Furthermore, the possibility to fold tensegrities for long-term storage allows to launch multiple devices each carrying a reduced scientific payload.

In addition to this, the internal volume (the void between the bars) of ReCTeR is small (approx. a cube with 0.5 m sides) and it therefore makes sense to construct a larger robot to allow for reasonably sized payload. In fact, icosahedron tensegrities have advantageous scaling properties as increasing the length of the bars does not significantly influence the total mass of the structure (because the mass is concentrated in the end caps).

The exposed spring design of ReCTeR becomes a safety issue when increasing the robot's mass (and thus spring tension). It is also difficult to mechanically limit the maximum spring extension to prevent plastic deformation (e.g. in case of a heavy external load). SUPERball will therefore feature an encapsulated spring design, which overcomes both problems and

simplifies assembly.

Robustness was not a primary design goal, but the prototype turned out to be more robust than expected. Extensive experiments (drop tests, rolling, reassembly and folding) have not resulted in any major mechanical or electrical failure. In future designs, we will aim for even more modularity and decentralization, as a failure of a central module will now result in the failure of two actuators. We also aim to implement part of the control algorithms (e.g. CPG generation) locally, as to enable robust locomotion even in case of temporary communication failure. ReCTeR is modular in the sense that the bars can be rearranged and the electronic and mechanical designs allow for differently sized robots. However, the modularity is predominantly a construction feature. It is not trivial to exchange a failing end cap or replace a sensor.

ReCTeR is capable of dynamic locomotion, but has to achieve this by making use of the elastic potential energy stored in the springs. As our calculations show that almost all parts of a tensegrity robot scale favorably in terms of specific strength, our goal is to obtain a final power-to-weight ratio about four times higher in SUPERball ($\pm 100\,\mathrm{W\,kg^{-1}}$ vs. $25\,\mathrm{W\,kg^{-1}}$ for ReCTeR). This allows for locomotion and manipulation experiments in any situation and state (i.e. outside an energy efficient regime).

7.3 Design

In order to develop SUPERball from ReCTeR's design limitations as well as our need for rapid experimentation of various tensegrity configurations and morphologies, we came up with a modular tensegrity platform to research large scale robotic tasks; e.g. a tensegrity planetary probe to explore Saturn's moon Titan.

7.3.1 Design Requirements

We obtained design requirements through an iterative approach involving NTRT simulator (Section 2.8.2) and the ReCTeR robot. As we recently validated our NTRT simulator by experimental validation with ReCTeR [21], we can now quickly evaluate various tensegrity configurations in simulation to find optimal mechanical design goals. Next to the NTRT simulator, we also incorporated results obtained with the Euler Lagrange solver based on Skelton's work [157].

The design requirements obtained from the NTRT simulations are given in Table 7.1. We are confident that a tensegrity robot achieving the following

conditions will be capable of dynamic locomotion, as shown by our evaluation of control policies in Section 7.4.

Table 7.1: SUPERball Design Requirements

robot	l_{strut}	Δl	$k_{passive}$	Ctrl. freq.	$\max \tau$
ReCTeR	1 m	$0.3\,\mathrm{m\,s^{-1}}$	$28.4\,\mathrm{N\,m^{-1}}$	40 Hz	0.03 N m
SUPERball	1.5 m	$\geq 0.3\,\mathrm{m\,s^{-1}}$	$\geq 500\,\mathrm{N\,m^{-1}}$	100 Hz	≥ 3 N m

In Table 7.1, l_{strut} is the length of a strut, Δl the rate of of length change (the change in rest length of the inline spring) of an actuated cable and $\max \tau$ the maximum spindle output torque.

7.3.2 Mechanical Design

SUPERball is an icosahedron tensegrity structure comprised of 12 motors at the end of the robot's 6 rods. Each rod is comprised of three main elements, 2 modular end cap assemblies containing all the mechanical and electrical systems and a connecting aluminum 2024 tube as a support structure. As shown in Figure 7.7, SUPERball's current target configuration is the standard tensegrity icosahedron pattern.

The main structural elements of the end caps were kept simple and split up into sections. This enables each end cap to be modular, as well as self contained, so that the end cap may be removed from the connecting rod as one whole unit. The end caps are held onto the connecting rods by a tube collar for easy removal with a single bolt. There are 5 sections to the modular end cap which are, a spring holder, battery holder and power distribution, motor and electronics element, cable actuation section, and a ground contact section. These sections as they are designed for SUPERball are shown in Figure 7.5. Each of these 5 sections can be removed from the rod as a full sub-assembly and replaced with a new component, increasing the versatility of each rod.

We will now discuss the main mechanical components (starting from the left in Figure 7.5). An overview of the electrical aspects is given in the next section.

7.3.2.1 Motor Assembly

Figure 7.6 displays the motor assembly, which at its core is similar to ReCTeR's. However, the new design is significantly more robust and versa-

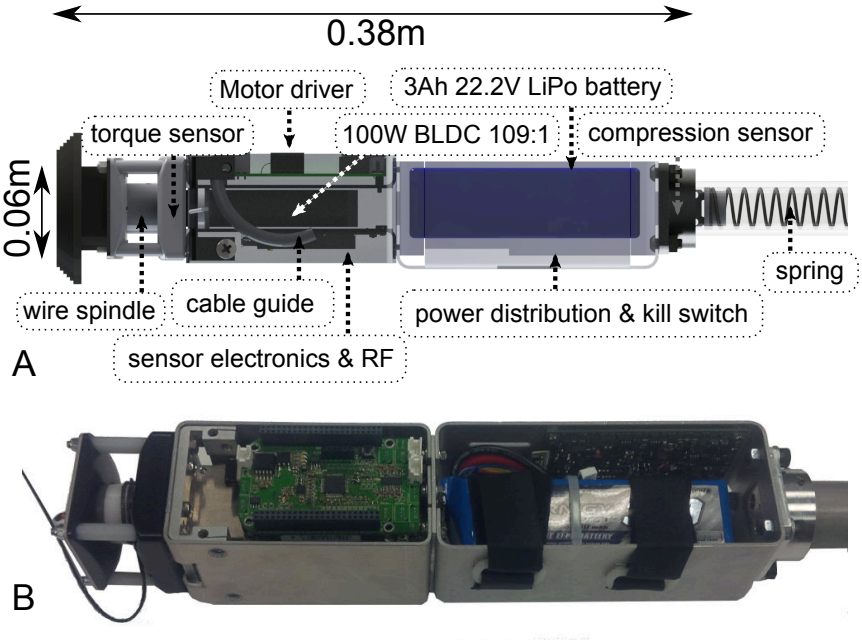


Figure 7.5: SUPERball end cap design. Each end cap is fully independent. **A)** CAD design **B)** Hardware prototype with wiring and protective cap removed. The spring assembly (on the right) slides into an aluminum (2024) tube connecting two end caps. A single bolt is sufficient to secure the end cap on the tube. The main structural components are made out of sheet metal ($2\,\mathrm{mm}$ 6061-T6 aluminum). Some load bearing parts such as the cable guide bracket use 7075-T7 (the top part of the cable guide bracket is clearly visible in **B**). Note that the actual cables are not shown, except for the actuated wire in the hardware prototype. In the current design, two cables run through the cable guide and along the battery and finally into the spring assembly inside the tube. Each has two main wireless interfaces: One provides high bandwidth with variable latency, the other is a low latency network for inter strut communication. A CAN bus provides a robust $1\,\mathrm{Mbit/s}$ communication line between the various circuit boards. To ensure safe operation each end cap has a wireless and physical kill switch.

tile. As on its predecessor, the motor axes are secured by two ball bearings. One is press fitted into the plate on top of the spindle and one integrated into the gear box.

The spindle is machined out of POM and has a diameter of 0.03 m. A 1.4 mm Vectran cable (Cortland 7012 Vectran HT, Type 150) with a breaking strength of 2227 N attaches to the spindle. It is important to prevent tight bends (knots) of the cable, as this reduces its strength. Therefore, the cable can slide into the spindle from the side and come out on top of the assembly (underneath the protective rubber cap). This allows to safely clamp the cable without damaging it. Note that the mass of the cable is only 1.7×10^{-3} kg/m, which implies that the mass of the cables can be neglected in most simulations (it represents around 0.5 % of the total mass of the robot). Vectran has lower creep than Dyneema (used on ReCTeR), but without protective coating it is sensitive to UV light [50, 149]. It has been successfully used in space applications (e.g. Mars Pathfinder airbags) due to its excellent thermal properties.

To further protect the cable and prevent it from getting stuck, the spindle is embedded into two smooth POM surfaces, with a 0.5 mm gap between the spindle and the surfaces. This allows the cable to slide smoothly in almost any direction without excessive friction. A torque sensor is integrated in the bottom sliding surface. Currently, this sensor is mounted on a rigid, cross-shaped motor mount. Each actuated spring-cable assembly is effectively a series elastic actuator, because it attaches to a spring inside another bar. As such, it is in theory not needed to provided an elastic element on the motor mount. However, it is possible to replace this basic torque sensing element with a torsion spring design (e.g. as featured on the Robonaut [39] or the COMAN humanoid robot [179]). Another anticipated extension is a ratcheting mechanism to make it possible to mechanically disconnect the actuators from the spring-cable assemblies. This can be useful for drop tests or long-term stowage.

The assembly is powered by a 100 W brushless DC motor (Maxon 386674, EC 22 mm) reduced by a 109:1 planetary gear box (Maxon 370784). The optical encoder is a Maxon 201940. In combination with the given spindle specifications, the assembly is rated up to 200 N of cable tension. As this is the first hardware iteration of SUPERball, we chose to optimize torque over speed. This gives us significant headroom to explore the robot's behavior with stiffer springs or under loading. More precisely, the nominal output speed (cable retraction) is $0.42\,\mathrm{m\,s^{-1}}$ with a cable tension of approximately 140 N. However, the large output torque poses a significant safety risk and we have taken various measures to allow for safe operation of the device. This is explained in more detail in the electronics Section 7.3.3.

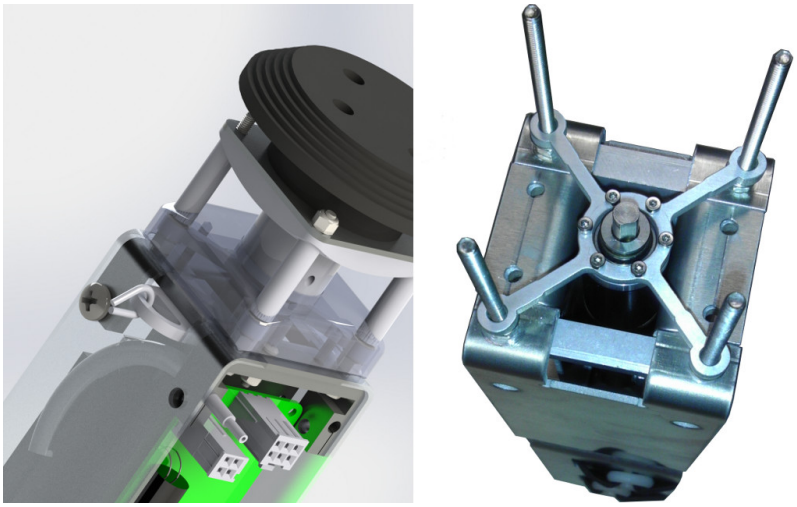


Figure 7.6: Close-up of the motor assembly. Left: CAD design showing the additional attachments, the cable guide, the spindle etc. Right: Top view of a motor assembly with the spindle and the cable surfaces removed. Strain gages are glued onto the arms of the cross-shaped motor mount for torque measurements.

A total of three circuit boards fit onto the motor bracket: the motor driver, a BeagleBone Black and the sensor PCB.

7.3.2.2 Battery Holder & Shaft Collar

The battery holder sits underneath the motor bracket and as it name implies it hosts the batteries of the end cap. It is a simple design, with one open side to allow easy main battery replacement. The battery holder also hosts the power distribution PCB. This circuit board provides battery protection and failover as well as wireless and wired kill switches. The main battery is a 3 A h 22.2 V lithium polymer battery and it is the heaviest component in the end cap (approximately 0.5 kg). The backup battery is a small, single cell (3.7 V) lithium polymer battery. This battery allows for hot swapping of the main battery. The expected battery life is between 30 min and 1 h.

7.3.2.3 Cable Guides & Attachments

While one cable attaches to the motor spindle, the remaining cables need to be attached to a spring inside the bar (explained in next section). Vectran is used for the part of each cable running in between two bars and on each spindle. However, Bowden cable connects the Vectran cable to the springs.

The reason for this is that in between the bars, lightweight, flexible cable is desired, which can be tightly wrapped around a spindle. On the contrary, the cables running inside a bar should be easy to guide and should maintain their shape when slack. In practice, we use common bicycle brake cable, which has an ultimate tensile strength similar to Vectran.

Each motor bracket has two PTFE tubes slightly protruding from each side. These tubes guide the Bowden cables downwards into the assembly. The tubes are in fact two nested tubes, which allows the inner tube to guide the cable, while the outer one protects the inner tube and friction locks onto the motor bracket. The Bowden cables run alongside the motor and then go through (while crossing) the opening between the motor bracket and the battery holder. In the battery holder, the cables run in a protective sleeve along the sheet metal (at opposite sides) and finally cross again while going into the springs assembly through the shaft collar.

It is also possible to attach cables directly to the motor bracket, which is useful to suspend a single bar or the robot for testing. This is shown on the left part of Figure 7.6.

7.3.2.4 Spring Assemblies

A lesson learned from ReCTeR was that externally exposed springs are not ideal for a robotic system. The exposed springs get caught on objects and the assumption of massless cables can no longer be applied. On the modular end cap for SUPERball, an enclosed compression spring system was developed to alleviate these issues. Compression springs were chosen so that during an unknown impact, the springs would not plastically deform. For SUPERball, a spring with a spring constant of $613\,\mathrm{N\,m^{-1}}$ is attached to a passive cable element and a $2850\,\mathrm{N\,m^{-1}}$ spring is attached to an actuated cable. The passive spring has a much higher compressive range to allow for pretension to be instated into the passive springs. A working prototype of our spring holder system can be seen in Figure 7.7. Note that this design allows to stack springs with different characteristics, to easily add damping or to increase the number of compliant spring units per end cap.

A spring is attached to a cable coming from another bar running through the end cap (as explained in detail before). In our current design with two spring units per end caps, the brake cable of the bottom one (the short spring in Figure 7.7) runs through the top spring. Each spring assembly is mounted onto a custom compression force sensor, which provides feedback about the current spring force or equivalently the tension on the cables connecting the bars.

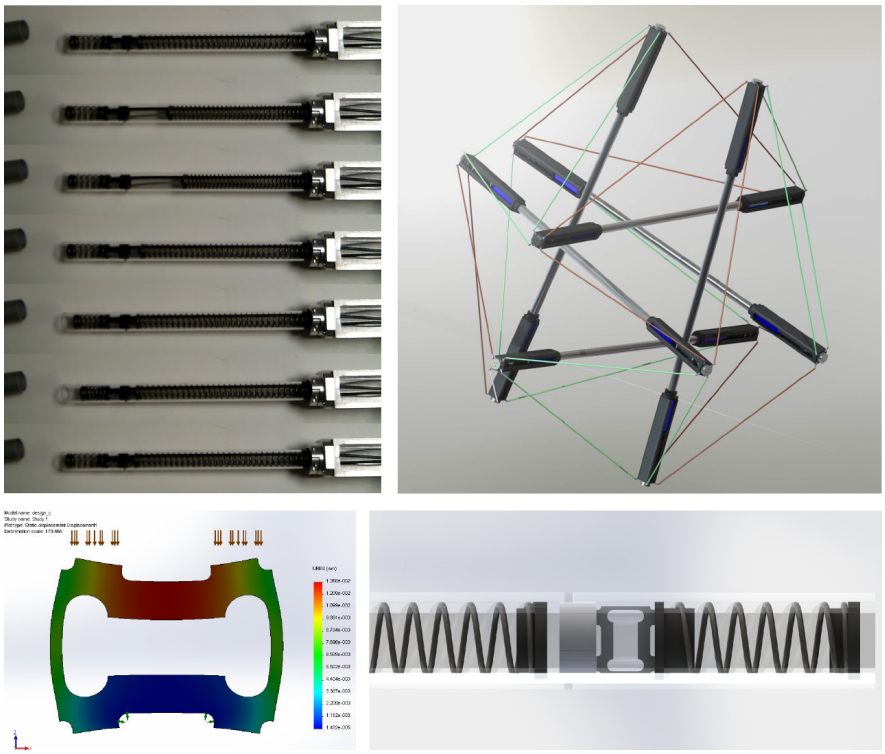


Figure 7.7: SUPERball design characteristics. Top right: Impression of SUPERball in deployed state. The orange/green cables show a connection pattern actuating half of the equilateral triangular faces. Top left: Internal strut design with 2 compression springs for two independent spring-cable assemblies. Brake cables run into the bar and attach to one end of a spring. Pulling the Vectran cable attached to the other end of the brake cable, compresses the spring. This results in a compliant assembly with no exposed springs. Bottom left: Compression force sensor design. Bottom right: Close-up CAD design of a spring assembly (brake cables not shown).

7.3.3 Electrical and Sensor Design

SUPERball was developed with distributed controls in mind. Each rod end cap houses multiple circuit boards: a motor driver, a sensor and communications board, an ARM board providing processing power and a power distribution board. We anticipate further integration of these circuits in future hardware revisions, but favored discrete boards — which allow for partial upgrades — for the first iteration.

The boards are interconnected by a 1 Mbit/s CAN bus, which allows for loose coupling and thus increased flexibility. We opted for this bus as it has a proven industrial and automotive track record and is widely supported on both microcontrollers and microprocessors. Other buses are less robust or don't provide sufficient bandwidth. While newer robust protocols such as EtherCAT provide significantly higher bit rates, they are not yet as widely supported on embedded devices as the CAN bus. Furthermore, the CAN bus bit rate is sufficient for signalling and transport of sensor values within an end cap. We use CANopen as the higher level protocol to transfer state and control commands. This protocol is supported by various manufacturers of high quality motor controllers and sensors (e.g. Maxon EPOS and Elmo controllers). As some newer interfaces can transport CAN bus protocols (e.g. CANopen-over-EtherCAT), the choice is also future proof.

A separate power distribution board was designed to support three main functions. The primary function is to safely switch the connection between the main battery and the motor driver. This is achieved by integrating two hardware kill switches and the ability to enable or disable the power in software (but not override the hardware switches). The wireless kill switch is based on a standard 433 MHz receiver, which is fully independent of the microcontroller or other components on the power board. As pushing a button on a compliant robot can be difficult, the wired kill switch instead uses a lanyard which breaks the circuit when pulled. This design is commonly used on speed boats and motor cycles. The second function is to provide a stable 5.5 V line to the other boards. More precisely, the power distribution board provides up to 3 A per line (4 A total). The choice for 5.5 V allows to use small and low-cost linear regulators on the other boards. The board can independently switch 4 5.5 V outputs. Finally, the PCB provides measurements of the current draw and power consumption of the actuator (using an Allegro ACS711 bidirectional Hall current sensor) and the 5.5 V lines. The board is controlled using a 70 MIPS Microchip microntroller. All communication and power lines use positive-locking connectors (Molex MicroClasp) to prevent faulty connections due to vibrations and shocks. Note that the choice for a single supply voltage, connector type and communication bus significantly simplifies the wiring.

The motor driver is a custom BLDC/PMSM driver board capable of block commutation and sensorless sinusoidal control. It is based on the DRV8303 chip from Texas Instruments which is controlled by a 70 MIPS microcontroller from the Microchip dsPIC33EP series. As can be expected, this microcontroller has significant headroom to implement complex control strategies. In fact, the same microcontroller series is used on all boards to allow code merging in case further integration of the electronics is desired. The board has been tested with brushless DC motors of up to 200 W.

The third PCB on the end caps is the sensor board, which hosts the main sensor equipment and provides a wireless interface. The main sensor equipment consists of a high precision ADC (24 bit Analog AD7193 in combination with the AMS AS1359) for the strain gages and a 9 DOF inertial measurement unit (InvenSense MPU6000 and Freescale MAG3110). The ADC provides the board with up to 8 pseudodifferential analog channels. In addition to this, the board hosts a wireless interface using the XBee pinout. Currently, XBee WiFi modules are used, but we anticipate to replace these with a lower level wireless interface (e.g. based on the nRF24l01+ or TI CC2520 chipset). The goal is to have a high bandwidth, variable latency wireless channel (explained next) in tandem with a lower bandwidth, but fixed, low-latency wireless interface. This allows for periodic transmission of all sensor data to an external observer over the high bandwidth interface, while providing a signalling at a latency similar to the local CAN network.

Finally, the last board is an off-the-shelf BeagleBone Black ARM based single-board computer. This board has a 1 GHz ARM Cortex A8 processor and runs the Ångström Linux distribution. We opted for this board as it low-cost, physically fits the end cap design and has an extensive user community. A USB WiFi stick is plugged into the USB board and acts as the high bandwidth channel to and from the end cap. The sensor board is designed in the shape of a cape (extension board) for the BeagleBone. Both boards thus simply slide onto each other. The sensor board controls the power to the BeagleBone, to allow for remote power cycling. Note that this is one of the reasons why a second wireless interface is available on a different board. The BeagleBone connects to the CAN network using a physical interface hosted on the sensor board.

A standard WiFi network is used to remotely control the robot. The CanFestival CANopen implementation allows to use the same codebase on the microcontrollers and the Linux boards. High bandwidth sensors (e.g. a camera) can directly connect to the BeagleBone to access the WiFi network and the CAN bus is thus primarily used for signalling and transmission of low bandwidth, low latency data. On a higher level, ROS (the Robot Operating System) is used to provide a convenient interface to the robot.

7.3.4 Modularity

SUPERball's current design configuration is the tensegrity icosahedron with 12 actuated spring-cable assemblies. However, one of our future objectives is to experiment with new actuator designs (e.g. two actuators per end cap, ratcheting devices...). This is made possible by a distributed mechanical, electrical and controls design.

On the mechanical side, this is achieved by allowing the end caps to easily slide into the tube connecting to another end cap. An end cap can be released with a single bolt. Electrical separation is obtained by integrating the batteries in the end caps. Furthere, it is straightforward to upgrade different parts of the electronics, as the connections between the circuit boards is limited to a CAN bus and power lines. Finally, the controls can be fully distributed because each end cap has independent wireless communication interfaces and processing power.

7.4 Controls Overview

As the first iteration of the hardware of a single SUPERball bar has only recently been completed, no hardware experiments besides static loading and sensor tests are currently available on the SUPERball platform and this section therefore focuses on simulation results. The main idea is to validate the simulator using experiments on ReCTeR (Chapter 6) and then scale the robot to the size, actuation power and mass of SUPERball. This allowed to estimate the design requirements of SUPERball, by developing and testing multiple control strategies in simulation. Moreover, this makes it possible to explore the most appropriate current and future sensor equipment, as well as the behavior of the structure with a payload. This section provides a qualitative description of the most promising control results we obtained.

The common assumption in these control methods is that the full outer shell can be actuated (24 actuated spring-cable assemblies), while the hardware is currently limited to 12 motors. Furthermore, the first control technique (Section 7.4.1) discusses an actuated payload. ReCTeR demonstrated that underactuation doesn't necessarily limit the locomotion capabilities of a tensegrity robot. The main reason to initially develop control strategies for a fully (i.e. all tensile members) actuated robot is that it simplifies the control problem: The structure is fully symmetric and rolling from any equilateral or isosceles triangular faces can be obtained with the same controls. The main types of knowledge which can be obtained from these fully actuated simulations are: insight into the behavior of different control methods,

bounds on the energy consumption and forces in the system, behavior of the robot under member failure, required control frequency and communication bandwidth. In this context, it is worth mentioning that hardware extensions of the end caps to two actuators are currently being studied at the University of California, Berkeley.

The results presented here are primarily due to Atil Iscen (Oregon State University) and Jérémie Despraz (EPFL). More precisely, I primarily provided input about the physical constraints, available sensor information, control frequencies and interpretations of the results. I therefore refer to Atil Iscen's PhD dissertation and papers, Jérémie Despraz's master's thesis and a recent paper of the NASA tensegrity group for in-depth discussions [21, 86, 87, 88, 89].

7.4.1 Bio-Inspired Controls

The goal of the first three related techniques is to control an icosahedron tensegrity structure with a payload. State feedback will be used to increase the rolling performance of the tensegrity robot, which is simulated with the NASA Tensegrity Robotics Toolkit. The idea behind the control laws is to create a torque by moving the center of mass of the robot with respect to the ground contact surface, in order to cause the robot to roll as illustrated in Figure 7.8. This motion is achieved by a two layer control architecture: The robot's heading and speed is controlled by the displacement of the central payload using the inner spring-cable assemblies, and the motion is simplified by actuating the outer shell. The outer shell refers to the 24 spring-cable assemblies of the standard icosahedron configuration, while the inner or payload members connect the payload to the tensegrity structure.

Three control approaches are tested for the inner spring-cable assemblies: reactive controls, inverse kinematics (IK) based controls, and CPG-based controls. The outer spring-cable assemblies are controlled with a hand-tuned approach. Actuation of the outer shell reduces ground contacts and not directly influences heading or speed. This affects the motion in several ways. First, it allows the creation of greater torques with the same payload displacement. Secondly, it makes the rolling behavior of the structure smoother, by preventing discontinuities due to excessive ground contact.

For each of the control approaches, inputs were taken as functions of the robot state. The height of each strut is computed using simulated omnidirectional distance sensors located at the end of each rods. The height assigned to each spring-cable assembly is computed as the average of the two end caps' height. The control for the outer shell cables was designed to tighten the bottom part of the structure when rolling, changing the lever

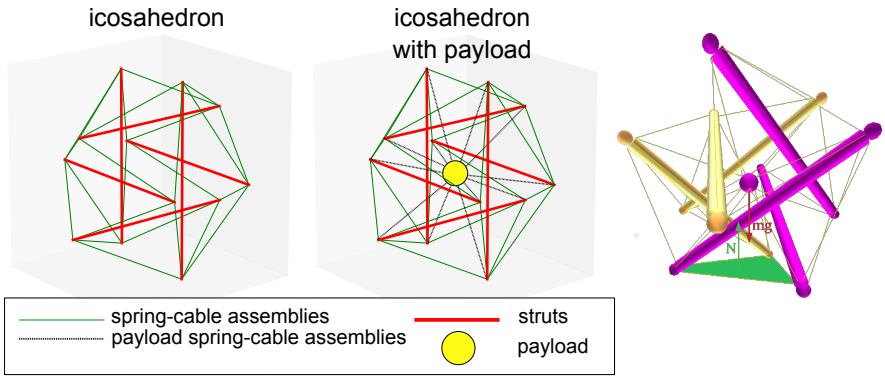


Figure 7.8: Left: Standard icosahedron configuration as used for the current SUPERball hardware. Center: Icosahedron configuration with payload suspended by 12 spring-cable assemblies. Right: Regular icosahedron tensegrity shape with central payload in the NASA Tensegrity Robotics Toolkit (NTRT). The triangular contact surface with the ground, highlighted in green, creates a reaction force N that, at rest, balances the weight of the structure, represented on the figure by the red arrow mg. A torque is created on the whole structure when a displacement of the center of mass from its rest position occurs.

arms of the gravitational force from the robot's center of mass, such that the robot does not require as much force to induce a roll. Typically in the presence of a slope, the reduction of the ground contact surface is sufficient to cause the robot to roll down the slope. In order to take this into account, we added a measure of speed, which is computed as the dot product between the center of mass position and the robot's overall heading direction vector. With this method, the speed is a scalar number and its sign depends on the heading of the tensegrity (positive if heading in the desired direction and negative otherwise). It can thus be used as feedback to influence the spring actuator command. The rest lengths of the *shell* spring-cable assemblies are computed using the following actuation rule:

$$\begin{cases} \dot{l}_{0,i} = w_s \left(l_0^* + \min(h_i^2, h_0^2) - l_{0,i} \right) & , \text{ speed} \geq 0 \\ \dot{l}_{0,i} = w_s \left(\bar{\ell} - l_{0,i} \right) & , \text{ otherwise} \end{cases} \tag{7.1}$$

where h_i is the height of spring-cable assembly i as measured from the distance sensors, $l_{0,i}$ is its current rest length, l_0^*, h_0 and $\bar{\ell}$ are constant parameters and $w_s \in \mathbb{R}_+$ accounts for the time scale at which length corrections occur. The constants l_0^* and h_0 represent the offset rest length of the spring and the maximum height measurement, respectively. The parameter $\bar{\ell}$

represents the default rest lengths of the springs that, if given as a command to all motors, puts the tensegrity in a stable position on the ground. This control law will contract the side of the robot near the ground when the robot is moving in the desired direction. The input and output values of the above control law are updated at each time step of the simulator. Impedance control, which was adapted to tensegrities previously [131, 173], is used to modify the spring-cable rest lengths.

7.4.1.1 Reactive Controls

The first technique for actuation of the inner payload spring-cable assemblies was the use of reactive controllers. We note that the only controllable parameter is the length of a cable. The variables $l_{0,i}$ here are the rest length of the *inner* springs. The global heading direction in a chosen inertial reference frame is defined by the unit vector $\boldsymbol{v}$ and the orientation of each spring in this same reference frame is represented by the vector $\boldsymbol{v}_i$. For each *inner* spring-cable assembly we use the dot product $d_i = \boldsymbol{v} \cdot \boldsymbol{v}_i$ as feedback to control the position of the payload as follows:

$$\dot{l}_{0,i} = (l_0^* + d_i\gamma - \|\boldsymbol{n}_{i,0} - \boldsymbol{n}_{i,1}\|)w_r \tag{7.2}$$

$$l_{0,i}(0) = l_0^* \tag{7.3}$$

where the weight w_r determines the reactivity of the system and $\gamma < 0$ is a fixed parameter. The vectors $\boldsymbol{n}_{i,0}$ and $\boldsymbol{n}_{i,1}$ contain the coordinates of the end caps to which the spring-cable i is attached. Thus, without any external perturbation, the system has a stable equilibrium position at $l_0^* + d_i\gamma$. The rest length of the spring-cable assemblies of which the orientation aligns with the global heading, is reduced. Vice versa, the springs pointing in the opposite direction are elongated. The global result is a displacement of the payload in the direction of the heading vector as shown in Figure 7.9. Note that the heading direction $\boldsymbol{v}$ can be chosen arbitrarily and can be adjusted dynamically. This method resulted in stable and smoothly rolling gaits allowing the tensegrity to roll up to $1\,\mathrm{m\,s^{-1}}$ over flat terrain. The robot could also handle slopes up to 8°, bumpy terrain, obstacles and collisions.

The main disadvantage of the reactive method is the type and amount of sensor feedback required to implement this approach in hardware. This issue is addressed by the control methods presented next, which are based on the same physical principle but require less feedback information.

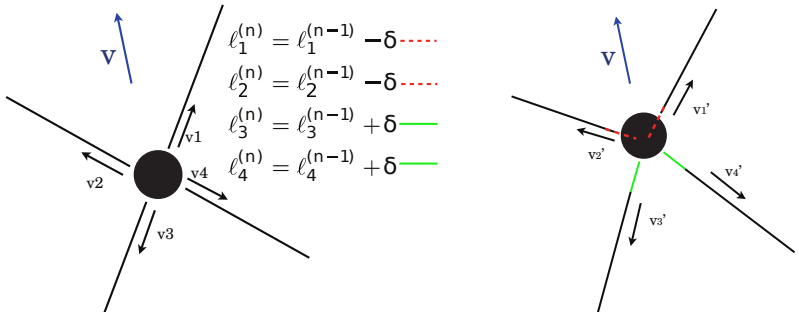


Figure 7.9: computation of the new rest lengths according to the spring-cable assemblies' individual orientations $\vec{v}_i$ (time $t^{(n-1)}$). Length modification is indicated by the colored lines, dashed red if reduced and green if elongated. The resulting effect is a displacement of the central payload in the desired direction $\vec{v}$ (time $t^{(n)} = t^{(n-1)} + dt$).

7.4.1.2 CPG Controls

Central Pattern Generators (CPGs) have been successfully used in past tensegrity systems [12]. Such controls are a feasible alternative to the reactive controllers that enable the generation of regular motion patterns. For this control, full state information is used to generate a smooth motion under the reactive controls, and then the resulting periodic commands were stored as a stable limit cycle of a CPG. Once this process is over, the tensegrity can be driven by the CPG output with much less feedback. We used an adaptive frequency Hopf oscillator [145] during the learning phase where the tensegrity is driven by the reactive controls. This method has the advantage of requiring minimal feedback and thus only a small amount of computations. However, it is important to note that the dynamical system runs on a much larger time scale than the perturbations disturbing the system. A tensegrity driven only by a CPG would then only, in the best case, have a stable rolling gait on a flat, obstacle free terrain. As a result, it is necessary to include also a second control method that can work on this smaller time scale and give an appropriate response to these external perturbations.

7.4.1.3 Hybrid CPG - Inverse Kinematics Controls

The final control method evaluated here for the actuation of the inner spring-cable assemblies is a hybrid technique with inverse kinematics (IK). The position of the central payload $\boldsymbol{p} = (n_x, n_y, n_z)$ is defined as a function of

the inner cable lengths $\boldsymbol{l} = (l_1, ..., l_{12})$.

The outputs of the IK algorithm $\boldsymbol{\xi} = \delta \boldsymbol{l}_{0,\infty}$ represent the length corrections that have to be made to reposition the payload at the desired location. Here $\delta \boldsymbol{l}_{0,\infty}$ is the solution of an iterative algorithm detailed in [21] to generate a target displacement $\Delta \boldsymbol{p}$ of the payload. The outputs $\boldsymbol{\xi}$ can be used together with the adaptive frequency oscillator as presented in Section 7.4.1.2. This approach is inspired by two previous works by Ajallooeian et al. [2] and Gay et al. [59].

While the pure CPG implementation does not allow any steering control of the robot, this implementation enables the guidance of the robot on a desired trajectory on flat terrain (see Figure 7.10). This is realized by modulating the CPG implementation of the previous section by adding the correction $\boldsymbol{\xi}$ to the CPG outputs.

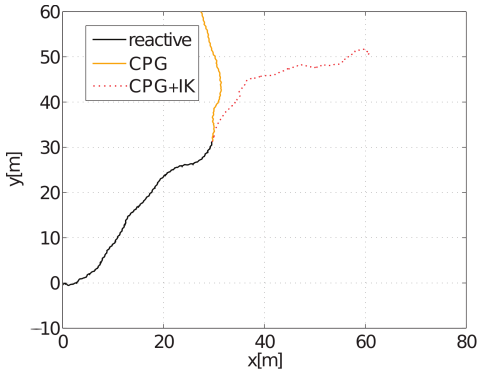


Figure 7.10: Trajectory of the tensegrity (top view). The black curve represents the trajectory while the robot is driven by the reactive control algorithm and the CPG is in the learning mode (50 s). The motion is regular and the heading is maintained throughout the whole period. The yellow and red trajectories represent the path traveled once the CPG controller takes over (40 s). When the CPG is coupled to the height signal and receives inputs from the second order inverse kinematics algorithm (red curve), the resulting trajectory is a long and relatively straight line extending well the reactive control.

Table 7.2 provides a summary of the results obtained with the different control strategies over regular flat terrain.

Note that the results do not take into account the trajectory of the path and, as a consequence, even if the distance traveled using the CPG controller

	reactive	CPG	Hybrid
average speed [$\mathrm{m\,s^{-1}}$]	1.00	0.50	0.38
complex terrain	Yes	No	No

Table 7.2: Bio-inspired control strategies summary

without any trajectory control is larger than with the hybrid control, the *quality* of the path is not as good (see e.g. Figure 7.10). Interestingly, we observe that the stable gait pattern obtained in simulation is a sequence of contacts defined as energetically optimal by Koizumi et al. [98] for a tensegrity icosahedron. With the current implementation, only the reactive controller manages to get the robot to roll in an efficient way over complex terrain and obstacles. To the best of our knowledge, this last result was the first implementation of a tensegrity robot controller demonstrating such capabilities. Figure 7.11 illustrates these results in the NTRT simulator. Closed loop controllers learned using Coevolutionary algorithms now exceed the performance reported here, but currently still require a significant learning phase [86] and Section 7.4.3.

Experimentation showed that the hybrid controller's performance is highly sensitive to the choice of some parameters appearing in the CPG equations. As a result, future work will incorporate other methods to optimize the feedback data and to compute the corrections, in order to more accurately navigate in complex environments. A good example of such an improvement can be found in Gay et al. [60] where sensory information is preprocessed by a Neural Network and trained using particle swarm optimization methods before being fed back to the CPG. Alternatively, the Physical Reservoir Computing principle (Section 6.5.5) can also be a applied for feedback computation in this case.

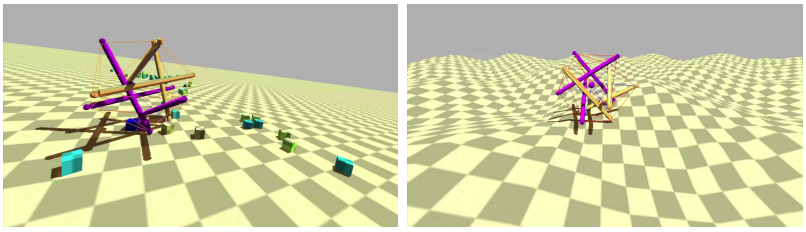


Figure 7.11: Examples of successful locomotion over complex terrains such as slopes, bumps and obstacles

7.4.2 Open Loop Rolling

Iscen demonstrated successful rolling locomotion of a tensegrity icosahedron based on Coevolutionary learning techniques with the NTRT simulator [87]. In his scheme, each spring-cable assembly has a controller that evolves independently from the other controllers (i.e., in independent pools), but cooperation is used to optimize behavior of the entire robot [132]. The objective function for this maximization was set to be the distance traveled during a fixed amount of time. The simplest implementation of this technique is an optimization of open loop control signals that are only a function of time; sinusoidal functions performed well [88].

More recently, the effects of different complexities and frequencies of these open loop signals were analyzed. More precisely, we optimized stepwise functions with varying numbers of via points, which allows for more complex motor signals than the basic sine waves. This enables the study of computational load and scaling properties needed to estimate power consumption of various controllers, as well as to investigate effects of actuator failure.

Figure 7.12 shows the main results of this work. In this case, optimized rest length signals had four via points. The graphs demonstrate that controllers exist for which the tensions generated in the structure are within the mechanical limits. Furthermore, the average power consumption is an order of magnitude lower than the actuator's rated power output. Note, that the power is approximated as the product of the velocity of an actuator multiplied by the tension incident spring-cable assembly (averaged over all actuators). This approximation only estimates the mechanical power, not the electrical power. The electrical power draw will be higher, but the current strut design has significant headroom (100 W vs. ≈6 W). A more detailed analysis of these results will be presented in [89].

While these open loop controllers demonstrated basic rolling behavior, they commonly fail in the presence of external forces or unexpected terrain conditions. This issue is addressed in the next section, which presents a simple rolling algorithm that uses ground contact sensors located on the simulated end caps.

7.4.3 Closed Loop Rolling

We show the results of a new method for learning to roll by exploiting the symmetry of the structure, combined with Coevolutionary algorithms. A rolling icosahedron tensegrity can be considered as consecutive *flops* made one after the other over its faces. For each specific flop, we study the movement of a static structure standing on a base equilateral triangle (i.e. one of the

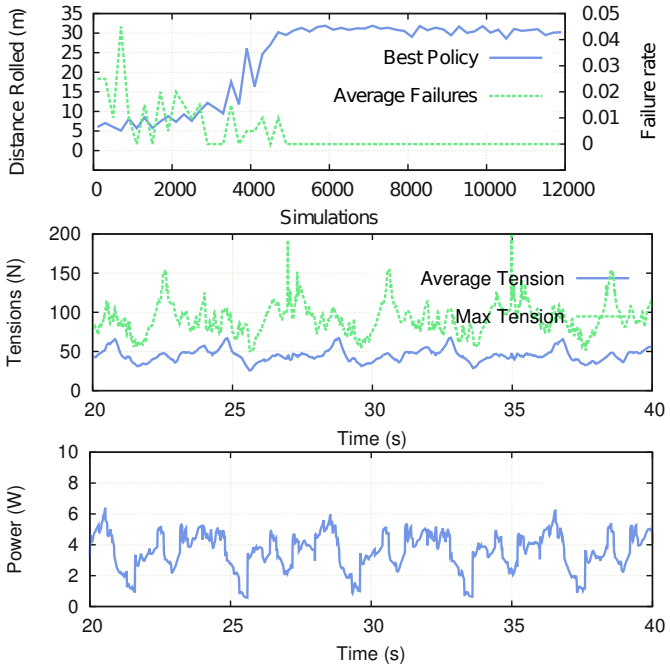


Figure 7.12: Main results of the open loop control method with piecewise linear functions as target spring-cable rest lengths. Co-evolutionary learning is used to optimize the actuation pattern of each of the 24 spring-cable assemblies. The method allows to quickly test the behavior of the robot with arbitrary constraints, without relying on sensor data. The top graph shows the learning curve, which illustrates that after approximately 5000 simulations, almost all controllers in the population successfully achieve locomotion and a reasonable average velocity on flat terrain (distance rolled in $1\,\mathrm{min}$). However, the second and third plots are of more interest as they provide insight into the mechanical requirements of SUPERball. The second plot shows the tensions in the structure to be below the mechanical limits. The third plot presents the mean power output of the actuators. The power is approximated as the product of the velocity of an actuator multiplied by the tension incident spring-cable assembly (averaged over all actuators). This approximation only estimates the mechanical power, not the electrical power (which will be significantly higher). A more detailed discussion will be presented in [89].

eight faces, as shown in Figure 6.2) and rotating itself over one of the sides of this triangle. After this move, the structure ends on another equilateral triangle, so that same move can be repeated to do another roll. This method will enable the structure to move from one static position to another by destabilizing the system to *flop* along one side of the equilateral triangles.

Although rolling is now simplified to one flop, the control problem remains challenging. Every action performed by an actuator propagates through the compliant, non linear system, which makes the problem complicated for classic control methods. Coevolutionary algorithms are a natural fit to solve controlling this compliant, non linear system [132].

As in the open loop method, the fitness function used is the distance that the robot rolls over a fixed period of time. Using this experimental setup, the coevolutionary algorithms optimized the move of a flop to achieve smooth rolling when the policy is applied for a series of flops.

The advantage of a symmetric structure is that once we have learned a controller for rolling in a single direction, the learned policy can then be used for rolling in any direction. The robot can be controlled to go in a specific direction using a series of flops over the closest edge of the base triangle. Figures 7.13 and 7.14 show the result of a controllable learned rolling motion with low tensions. Remark that the average and maximum tensions are similar to the open loop results presented in the previous section.

The main goal of this section was to verify if a tensegrity robot based on the proposed design parameters would be capable of dynamic locomotion. We omit the details of the locomotion algorithm and Coevolutionary learning here, which is discussed in depth in Atil Iscen's PhD dissertation [86]. Iscen also extends these results to different terrain conditions and swarms of tensegrity robots.

7.5 Conclusion

This chapter presented ongoing work on SUPERball, a modular tensegrity robotics platform aimed at planetary exploration. The current version of SUPERball is a robust terrestrial robot, which will be used to study control and payload protection aspects. The ReCTeR robot discussed in the previous chapter, demonstrated untethered rolling, folding, and sensing capabilities in a low-cost, safe, research prototype. SUPERball builds upon these results, but the design approach particularly focuses on robust engineering, electronics and software. As a result, SUPERball is much more a dynamic tensegrity research platform than a single robot.

One of the main premises is that tensegrity robotics research is still at

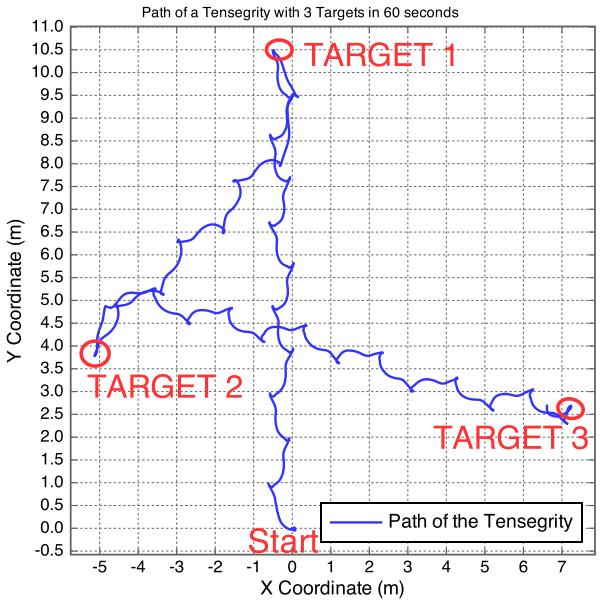


Figure 7.13: Controllable rolling with a learned controller. After learning a control policy, the symmetry of the structure can be used to control the direction. The plot show the robot's center of mass. The zigzag results from the robot rolling over a sequence of equilateral triangular faces.

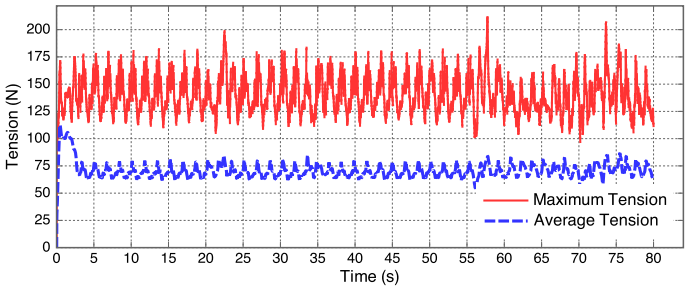


Figure 7.14: Spring tensions for the trajectory in Fig. 7.13. A relatively low average tension of $75\,\mathrm{N}$ keeps the structure in tension, with peak forces up to $200\,\mathrm{N}$ due to the actuators.

an early stage, which justifies and in fact calls for a robust and extensible platform which allows researchers to adapt it to their specific needs. In addition to this the SUPERball designs are made publicly available to the degree allowed. Similar efforts have proven successful in other domains. For example, the e-puck robot and the iCub are open source platforms targeted at mobile/swarm robotics and human cognition respectively. Even though none of these platforms is economical, they have been widely adopted in academia.

Our future outlook for SUPERball is to demonstrate dynamic rolling within a year and then focus on payload transportation and increased drop test robustness. In parallel, we are studying extended end cap designs that allow full actuation of an icosahedron robot.

8

Conclusions and Future Perspectives

I touched upon a wide range of topics related to actuated tensegrity structures. However, the two subjects which stand out are the computational aspects of tensegrities and how to build robots based on this design principle. At the start of my PhD research, I was unaware of the existence of tensegrity structures and my original goal was to investigate design principles for morphological computation in a more general context. When I stumbled upon a few examples of tensegrities in my search for highly compliant free-standing structures, I quickly became intrigued by their at times counterintuitive properties.

Most publications on tensegrity structures either focus purely on mechanical properties or instead disregard their mechanical design and just consider them as a simulation object without addressing practical constraints. Throughout this thesis, I have tried to find a balance between these perspectives. This approach is suitable for robotics as future tensegrity robot designs have to be accessible to both mechanical and computer science engineers. An integrated approach of electronics, mechanics and controls is needed with a common language. Therefore, I provided background information on the mechanical aspects (stiffness, kinematics, form-finding...), computational aspects (e.g. Reservoir Computing) and control methods. This makes the work accessible to readers from various fields and it is my hope that this work may also indirectly advance the tensegrity robotics field by attracting new researchers.

More than halfway through my PhD study, I received the great opportunity to contribute to a NASA project. This was a rather unexpected application of tensegrity robotics, but the goals and methods quickly showed to align with the objectives set out for my thesis. I discussed my contributions to this project in the previous chapter, which thus shows how the sometimes abstract methods from the earlier chapters map onto practical applications in a new domain.

In the next sections I first review the main results of this work, followed by my conclusions. I then end my dissertation with a discussion of future research directions.

8.1 Summary of Results

Tensegrity Statics and Dynamics I provided an overview of the statics and dynamics of tensegrities in Chapter 2. A number of original methods were introduced to:

- Optimize the shape of a tensegrity structure, to match a desired shape as closely as possible;
- Tune the stiffness and behavior under deformation of a redundant (multiple states of self-stresss) tensegrity.

Computational Aspects Chapters 4 and 5 discuss the computational aspects of compliant tensegrity robots. I show how simple, yet robust feedback can be designed which exploit the computations available in the dynamics of the structures. More precisely, Chapter 4 uses the Reservoir Computing paradigm to use tensegrity robots as computational black boxes. The following chapter focuses on Hebbian plasticity rules to find a suitable feedback controller for tensegrity structures. Examples are provided for Recurrent Neural Networks and tensegrity robots to illustrate their similarity from a computational perspective. The key result is that Physical Reservoir Computing can solve control problems by interpreting them as computational tasks.

Hardware Designs Chapter 6 and 7 present two new tensegrity robot designs. ReCTeR is an untethered, lightweight tensegrity icosahedron robot with 6 low power DC motors. SUPERball is a tensegrity robot platform developed at the NASA Ames Research Center. It is inspired by some of the design features of ReCTeR, but SUPERball is larger, highly modular and more robust. To facilitate dynamic locomotion the actuation power has also significantly been increased.

Validation of Simulations Chapter 6 also presents a comparison of simulation results and hardware measurements to validate the simulators used throughout this work. To the best of my knowledge, this is the first time that such a study has been performed. This allows to develop control methods in simulation with reasonable confidence that they can be reproduced by robotic

hardware. Though care needs to be taken to prevent learning algorithms from exploiting unverified properties of the simulated environment.

Control Methods Control methods are discussed at multiple locations. First, the computational aspects were presented as closed loop control method in Chapters 4, 5 and 6. In the final research Chapter 7, I also presented alternative control methods for tensegrity robots.

8.2 Main Conclusions

Computational Tools Although classic control theory approaches have resulted in powerful analytical tools, there is a certain appeal to black box and computational techniques such as those studied in this work. The view that a robot is but a hardware extension that allows a controller or software agent to act in the physical world, is often problematic. Indeed, it is the combination of the controller (mind) and hardware (body) that defines the characteristics and behavior of an agent (embodied cognition).

Today we are faced with new types of soft robots and even hybrid systems, such as artificial limbs. This creates a demand for new control approaches that bridge the gap between hardware and controls. How can we build and control robots that purposefully interact with their environment in intrinsically safe ways, without requiring excessive computational resources? The additional degrees of freedom of compliant and soft robots do not rule out the use of well-established control methods. However, their use can be cumbersome, when for example a large number of state variables and stiff equations are needed to model and control the interactions of such robots with their environment.

There are many aspects that are not relevant to high-level controllers and that would ideally be handled by local, distributed controllers or by the hardware itself. It is known that this is possible, as biological systems do not possess the computational and communication resources to act at this timescale. Some dynamics are intrinsically handled by the musculoskeletal system without interaction with the central nervous system, or at least with only local feedback (e.g. CPGs in the spine).

I think that one of the main future challenges is to combine (classic) control theory methods and computational approaches. In this thesis, I have demonstrated very simple, yet robust low-level control techniques and also a basic layered approach — feedforward kinematics combined with a learned feedback controller. One of the main values of these contributions is that they can manage the lowest levels of control with minimal computational re-

sources. While these are promising results, control theory generally provides deep, often theoretical, insights. I am convinced that a viable and fruitful path to bridge these fields is to begin work on more theoretical analyses for computational control methods and to loosen some of the strict theoretical requirements in the (classic) control theory field.

One important point of self-criticism is that it is non-trivial to obtain useful theoretical results for the types of control discussed in this thesis. In practice, one often still uses (classic) control theory methods (e.g. local linearization) when studying the behavior of the controllers studied herein. I think that the tensegrity and spring-mass models, due to their elegant algebraic descriptions, will allow for more in depth theoretical studies of the computational properties of compliant robots than general soft robots (e.g. blob-like robots).

Finally, there are significant amounts of research being performed in other branches of Physical Reservoir Computing. In particular, I see Photonic Reservoir Computing as one domain to which the learning rules discussed in Chapter 5 could be of great use. There are noise sources in photonic networks and the properties of these systems fluctuate due to temperature changes and other factors. The reward based rules from Chapter 5 appear particularly suited to compensate for such fluctuations, similar to adaptive control.

Tensegrity Hardware is Complex The robot designs presented in this thesis have been described as "a rolling tangle of rods that can take a beating" (Wired) and "jumble of tent poles" (IEEE Spectrum). While this might make it appear that tensegrity robot design is a simple endeavour, this appearance is highly deceiving. Passive tensegrity structures are indeed fairly straightforward to construct. Some geometric insight or trial-and-error is required to assemble a prestressed structure, but the overall process is clear.

On the contrary, actuated tensegrity designs quickly become complex for various reasons. At first, tensegrity robots might simply appear as motors on a stick. However, there are many hard questions to be solved to turn these motors on a stick into a useful robot: How to power the motors, how to efficiently use their output power, how to provide sensing capabilities, how to provide a level of compliance that provides robustness to significant drops...

I provided answers to these questions in the form of two robot designs. While I acknowledge that other solutions exist to certain problems I was faced with (e.g. the use of pneumatic or linear actuators), the designs presented herein provide a working solution to each issue. The result is a recipe and baseline design for distributed, untethered compliant tensegrity robots with rich sensor capabilities. Major parts of these designs have been made publicly

available, to the degree allowed, to accelerate future tensegrity research.

Tensegrities Scale Well ReCTeR is a relatively small tensegrity robot with a diameter of about 1 m and a total mass of 1.1 kg, while SUPERball is a much larger and heavier robot with a 1.5 m diameter and a mass of ≈15 kg. One remarkable feature is that all structural elements of tensegrities tend to scale well (i.e. most scale approximately linearly as a function of the robot's mass). Indeed, springs, cables, bars and actuators are all available in a wide and almost continuous range. During the design phase of SUPERball, one of the first goals was to allow a small payload to be transported by the robot. This implied scaling up ReCTeR's design both in size and weight. Interestingly this did not incur any major difficulties in the materials selection process. The carbon fiber bars were exchanged for aluminum tubes, but the main reason for this was not mechanical. Rather, workplace health and safety restrictions for the machining of carbon fiber and the ease of handling aluminum determined this choice. Furthermore, changing the level of passive compliance of the robot involves little less than replacing a set of springs. Similarly, there do not appear to be any major obstacles to scale down a tensegrity robot design. However, the development of capable tensegrity robots smaller than ReCTeR (say an order of magnitude) may become hard due to manufacturing issues. In practice, building a larger tensegrity robot makes it easier to incorporate features such as modularity as less time and effort needs to be spent on the miniaturization of cabling and electronics and because actuators tend to become more efficient. However, safety becomes an important design aspect for large tensegrities due to the inherently distributed and untethered motor control and the significant loading of thin tensile elements.

Tensegrities are Robust in Practice During the design of ReCTeR robustness to impacts was only a minor concern. The main objective was to build a compliant and actuated structure. However, the robot has been disassembled and reassembled multiple times and underwent drop tests up to 1 m. This is remarkable because the hardware platform is not a perfect tensegrity, in the sense that cable do not attach exactly at the center of the compressive elements. This is an indication that the tensegrity principle is more than a theoretical gimmick.

Tensegrities Handle Imprecise Control The control methods developed and tested in this work were all noisy or exploratory to some extent. For example, the Reward Modulated Hebbian plasticity rules rely on motor babbling to tune the feedback controller. In stiff robots and many compliant robots, such controllers would risk breaking the hardware. In a compliant

tensegrity, especially a spherical one, forces dissipate through the structure. Each actuator causes global reconfiguration. This cuts both ways: A single actuator error tends to have a small global effect, but coordination is needed between to obtain a desired global motion.

Structured Soft Robots Tensegrities are made out of discrete elements, but their behavior is much like that of soft continuous structure. Indeed, a spherical tensegrity tends to act like a squishy ball. However, tensegrity structures allow the behavior of a (soft) structure to be controlled. This became apparent in Chapter 2, where I showed that tensegrities can be reconfigured to stiffen or flex certain movements or modes. While it is possible to do this in a continuous structure (topological optimization methods), it is typically more complex. Common engineering materials can be used to construct tensegrities. Indeed, the tensegrity principle allows the use of engineering materials with well known properties in optimal axial loading conditions. This is a significant benefit over soft robots made out of e.g. rubber. It is my opinion that tensegrities are an exceptional tool to study soft structures or organisms (e.g. caterpillars) without delving into the intricacies of continuous soft body studies or simulations. For example, one can build a tensegrity structure that approximates the behavior of the soft body to study the main load paths. Note that I do not state that soft biological structures are indeed tensegrities, but only that tensegrities can be a good model for it, because of their structure and discrete set of parameters.

Distributed Controls I stated that tensegrities require a coordinated approach to generate a desired global movement. In contrast, the distributed nature of Class 1 tensegrity robots calls for distributed controls. This might seem a fundamental problem at first, but it is not in practice. First, the passive compliance in studied robots allows for relatively slow controllers. ReCTeR's wireless interfaces operated at 40 Hz (limited at 100 Hz) and SUPERball is targeting wireless communication at 100 Hz. These are low numbers for today's robotic hardware. Beyond this wireless communication approach, there are other ways to achieve coordination between distributed controllers in a tensegrity. For example Rieffel et al. used body dynamics to transfer information between controllers and Central Pattern Generators with inputs tend to synchronize.

Tensegrities Bridge Fields As becomes clear throughout this work, tensegrities inspire and take inspiration from multiple disciplines. The principle scales from the cellular level up to the architectural level. Therefore, it is an ideal approach to build compliant robots which bring together experts from different fields.

8.3 Future Perspectives

Better Tensegrity Tools I contributed to the development of the NASA Tensegrity Robotics Toolkit (NTRT) by comparing motion capture measurement data with simulation results. While this was successful, it did not capture the behavior of the actuators, nor did it provide accurate simulations of the control architecture or power electronics. Therefore, the developed tools need to be extended to allow for hardware-in-the-loop simulations which provide a uniform interface to the hardware and simulated robots. Due to the intricate dynamics of compliant tensegrities in an unknown environment, direct transfer of a simulated controller to the hardware with similar performance is not expected. However, by providing a uniform interface/API to the hardware and simulator(s), testing a simulated controller in hardware should ideally involve little less than tuning of a set of parameters.

Extensions of the SUPERball Modular Tensegrity Platform Modularity was a key design feature of SUPERball and it will allow future design improvements and variations. One of the main future goals is to add additional motors (i.e. a fully actuated icosahedron tensegrity) and to extend the sensor equipment. In particular, we plan to embed a ground reaction force sensor in the tip of the end caps.

Payload Transportation Transport of a payload by suspending it in the center of a tensegrity robot has only been studied to a limited extent in simulation. However, it is one of the main goals to push the application of tensegrity robots beyond locomotion capabilities. Therefore, further research is needed to define the additional sensor and actuator equipment needed to transport and protect a payload. While locomotion studies have up to now mainly focused on moving a structure as fast as possible or the robustness of the control methods, control approaches for payload transportation should also take the resulting behavior of the payload into account.

Hardware Implementations of Robust Controls The Physical Reservoir Computing method was tested on the ReCTeR robot, but the other control strategies discussed in or referenced by this thesis have only been implemented in silico. It is clear that comparisons of hardware implementations are needed to provide definitive answers about the advantages and disadvantages of the various approaches.

Ideal Platform for Artificial Muscle Research The most common method to actuate tensegrity robots is by using an active tensile network.

However, this typically involves changing the configuration (stiffness/rest length) of linear elements connecting the compressive members. As such, it is clear that tensegrity robotics would significantly benefit from linear actuation techniques. Unfortunately, linear actuation methods still lag behind the omnipresent rotational motors.

In recent years there have been tremendous efforts in the development of artificial muscles and linear actuators in general. Unfortunately, none of the emerging techniques currently offers the capabilities exhibited by rotational actuators. Therefore, tensegrity robots — and in particular SUPERball — are an ideal test platform for linear actuation techniques. Indeed, tensegrity robots for space exploration require motors which are power efficient, robust against temperature variations and have a large stroke distance.

In this context, the Intelligent Robotics Group at NASA Ames Research Center was awarded a Center Innovation Fund to research carbon nanotube yarn based multi-function tendons in collaboration with the NASA Glenn Research Center. This type of tendon could potentially deliver power, data, and actuation.

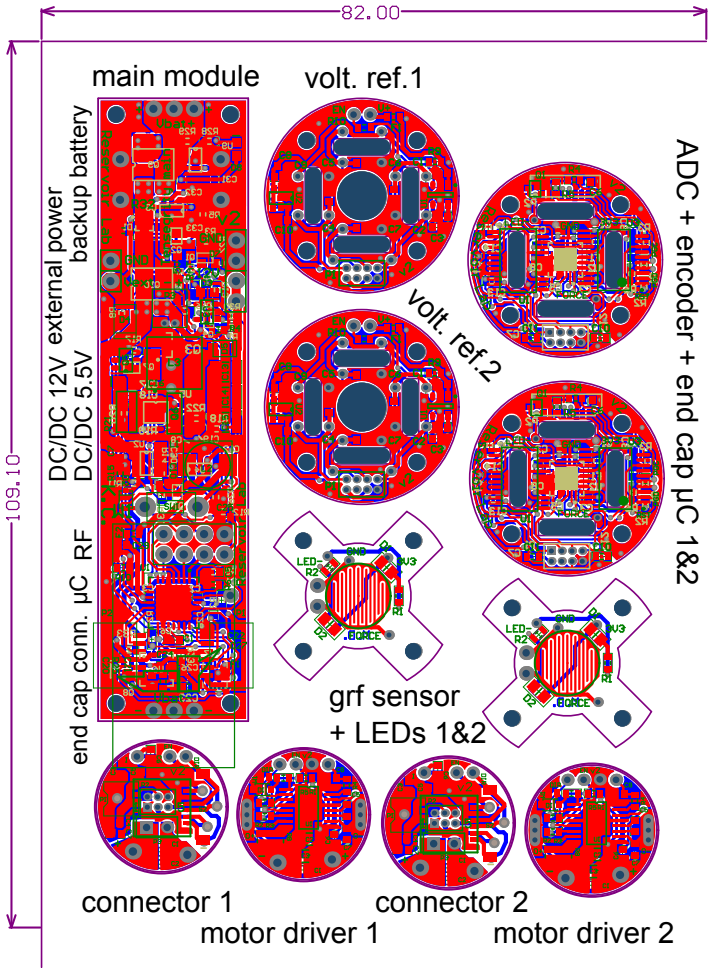
Tensegrities in Extreme (Drop Tests) Conditions The ultimate goal for tensegrity based space exploration is to send a folded robot into space, which is then deployed, lands from orbit without assistance (i.e. a drop) and then provides rover capabilities for a scientific payload. While successful drop tests have been demonstrated with passive structures and ReCTeR survives moderate drops, the goal of a single device integrating all these functions is not yet available. Some of the assumptions (e.g. no bending of the members) may become invalid during a major impact and it remains to be studied what the parameters, configurations and materials of a tensegrity robot combining these features should be.

One of the primary goals of the SUPERball project is to build a robot with locomotion capabilities, which can be deployed in a harsh environment on Earth. This is clearly a first major step towards a versatile space-qualified compliant tensegrity robot with these capabilities.

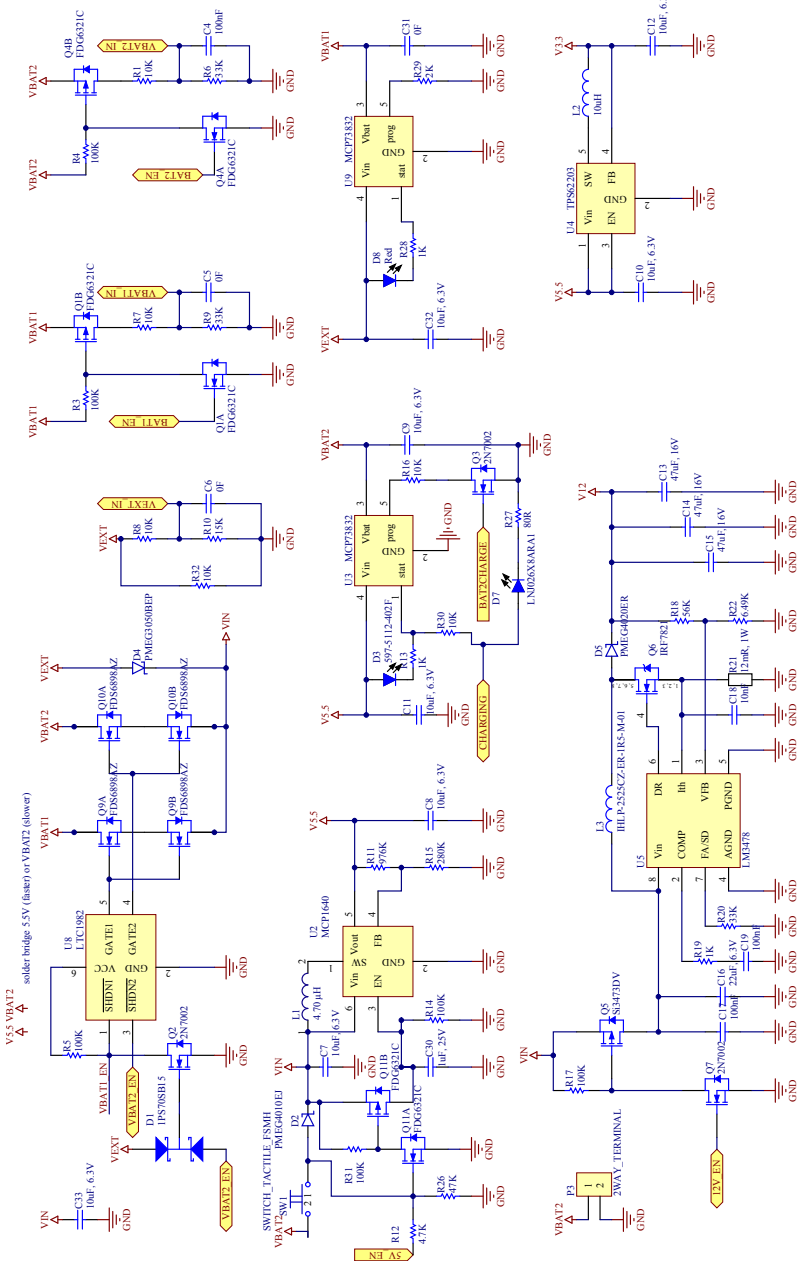
A

ReCTeR Electronics

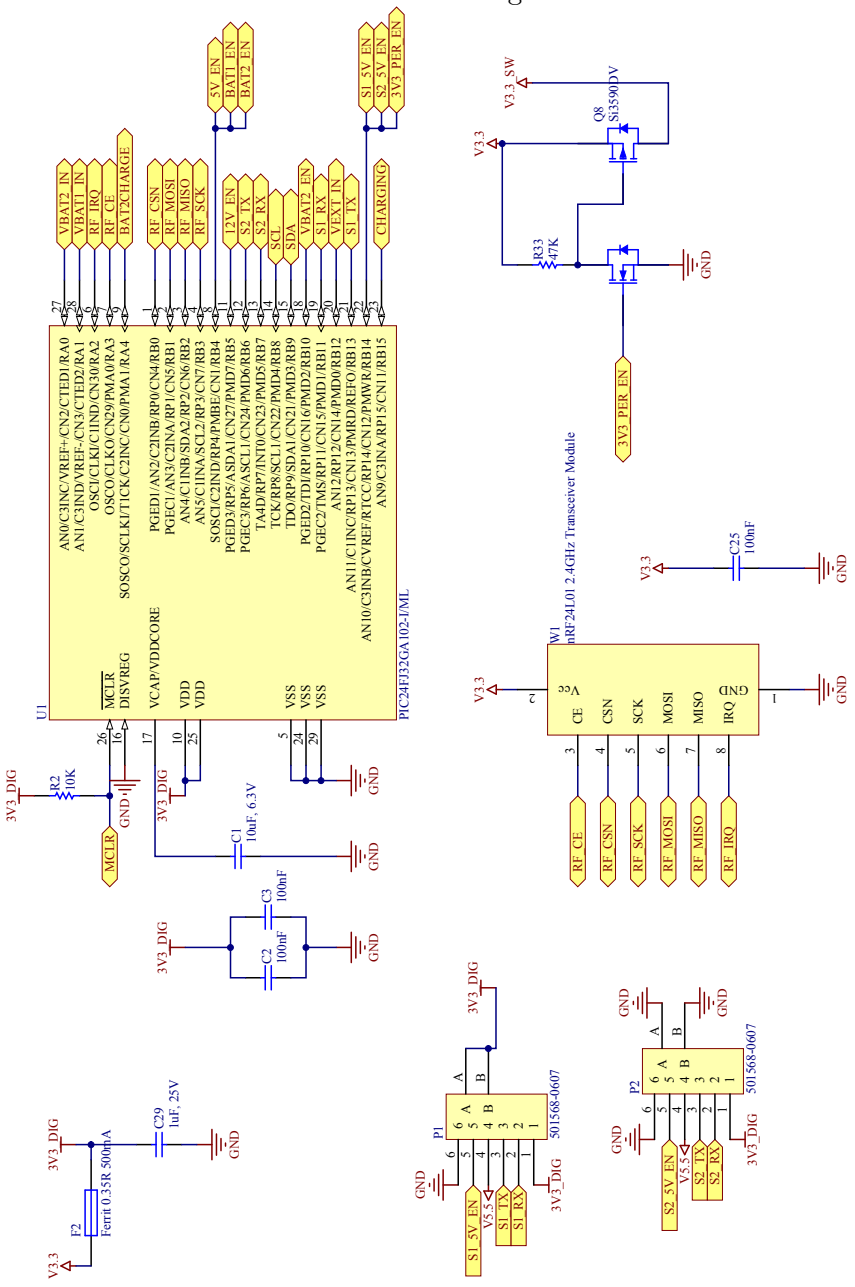
The ReCTeR schematics and PCB layouts are available at `http://ti.arc.nasa.gov/tech/asr/intelligent-robotics/tensegrity/ntrt/`.



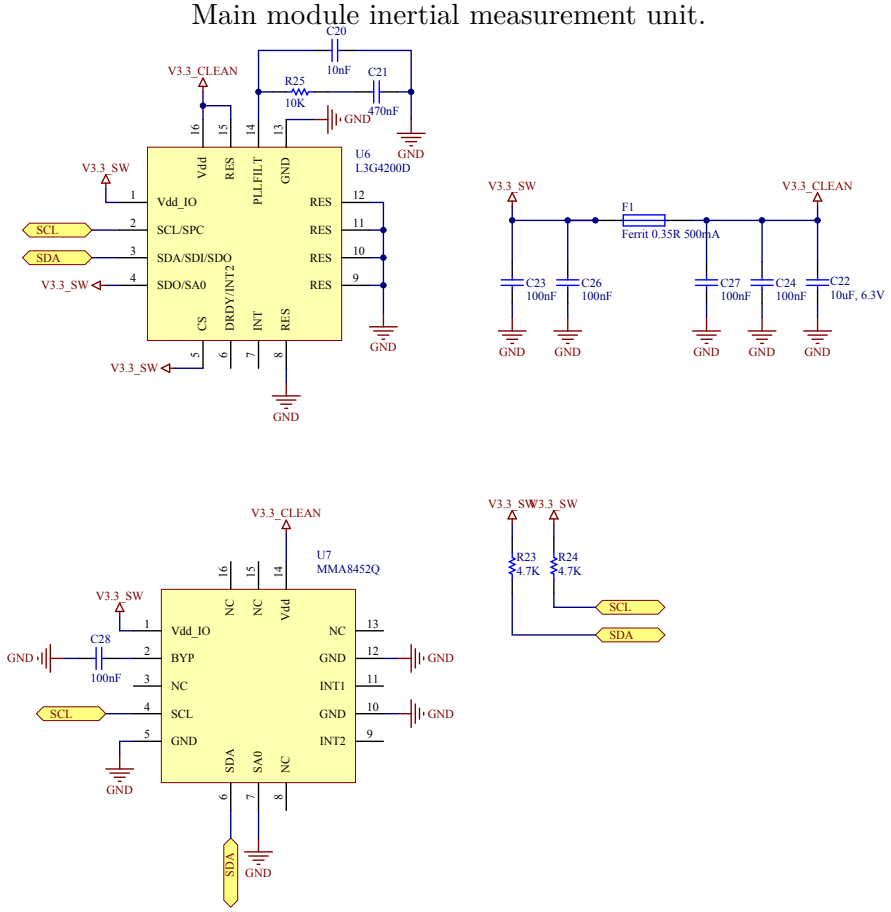
Main module power distribution.



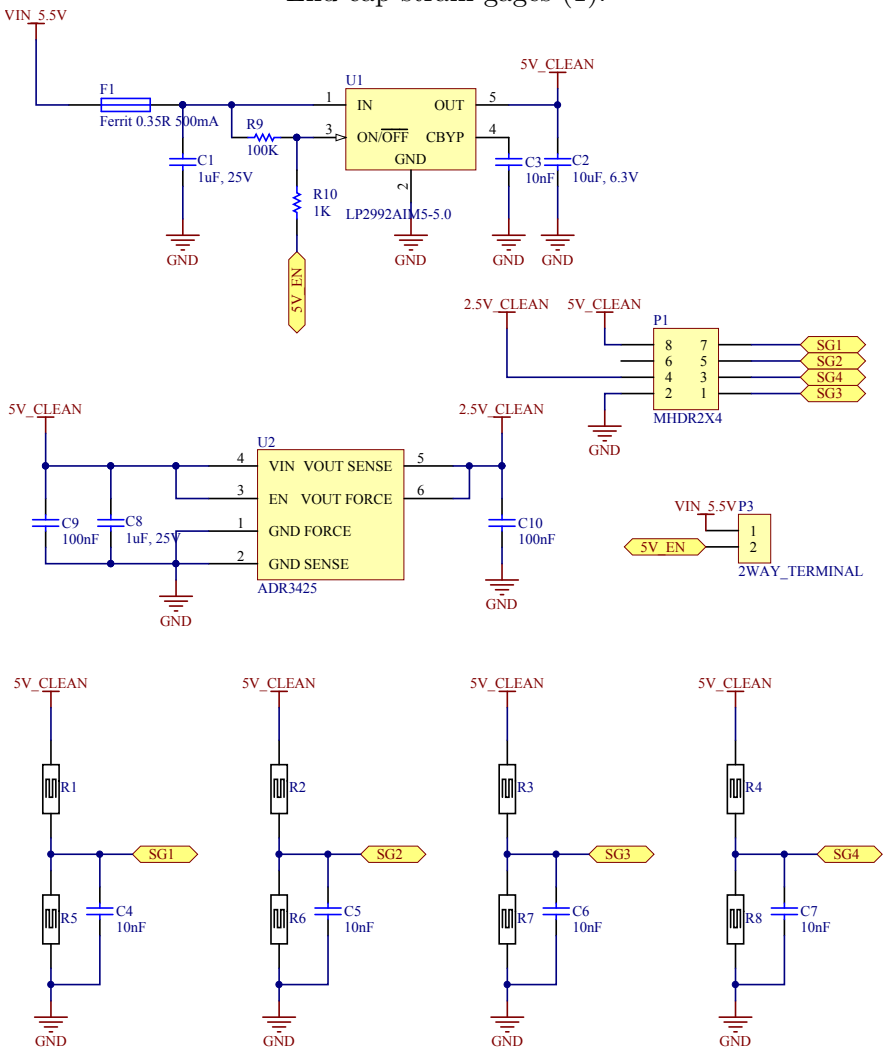
Main module logic.



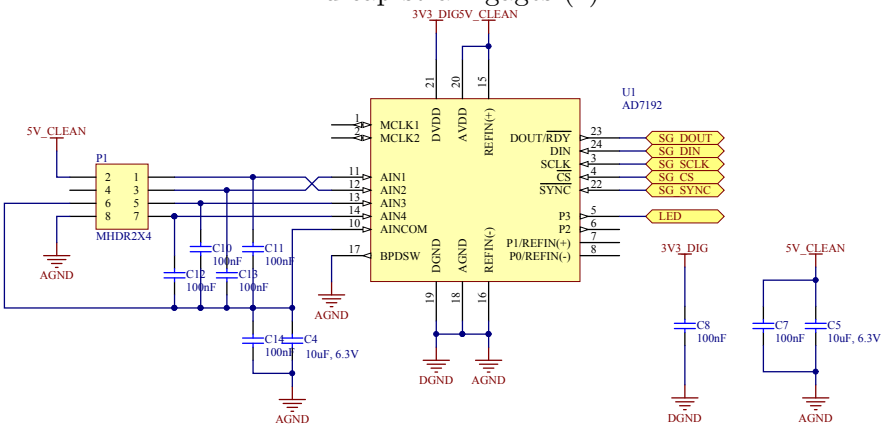
Main module inertial measurement unit.



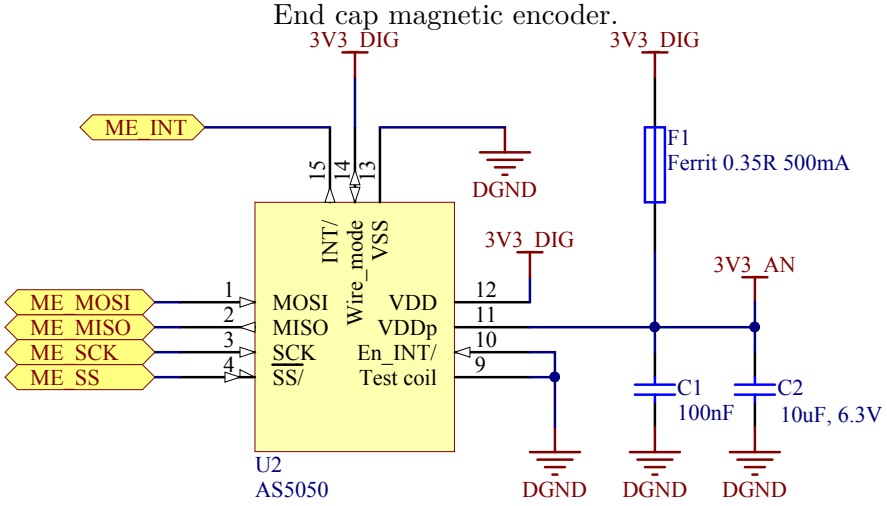
End cap strain gages (1).



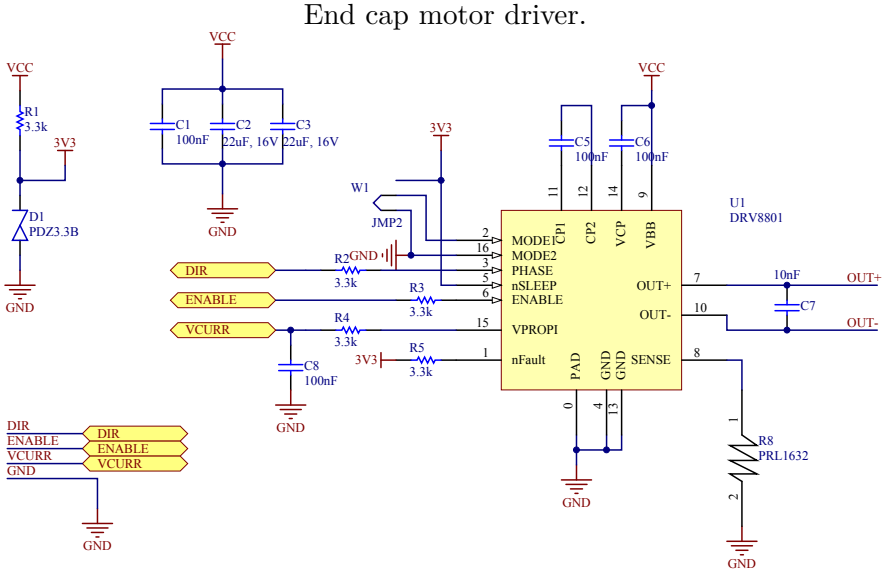
End cap strain gages (2)



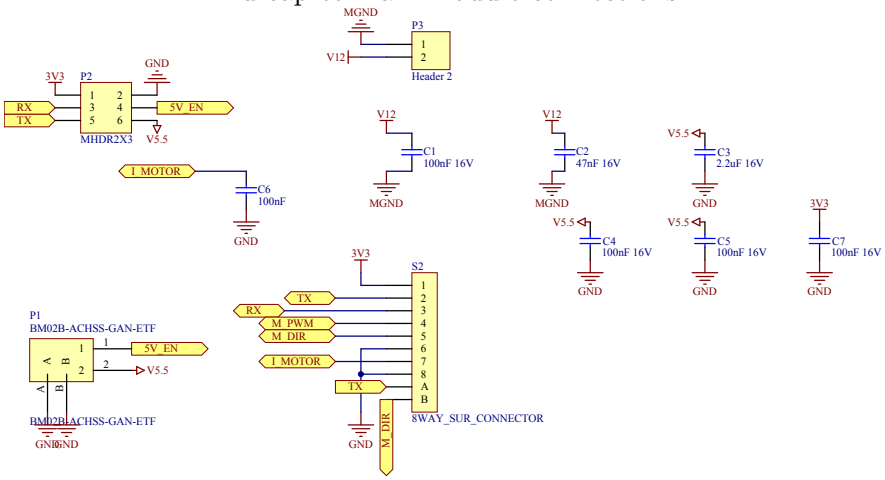
End cap magnetic encoder.



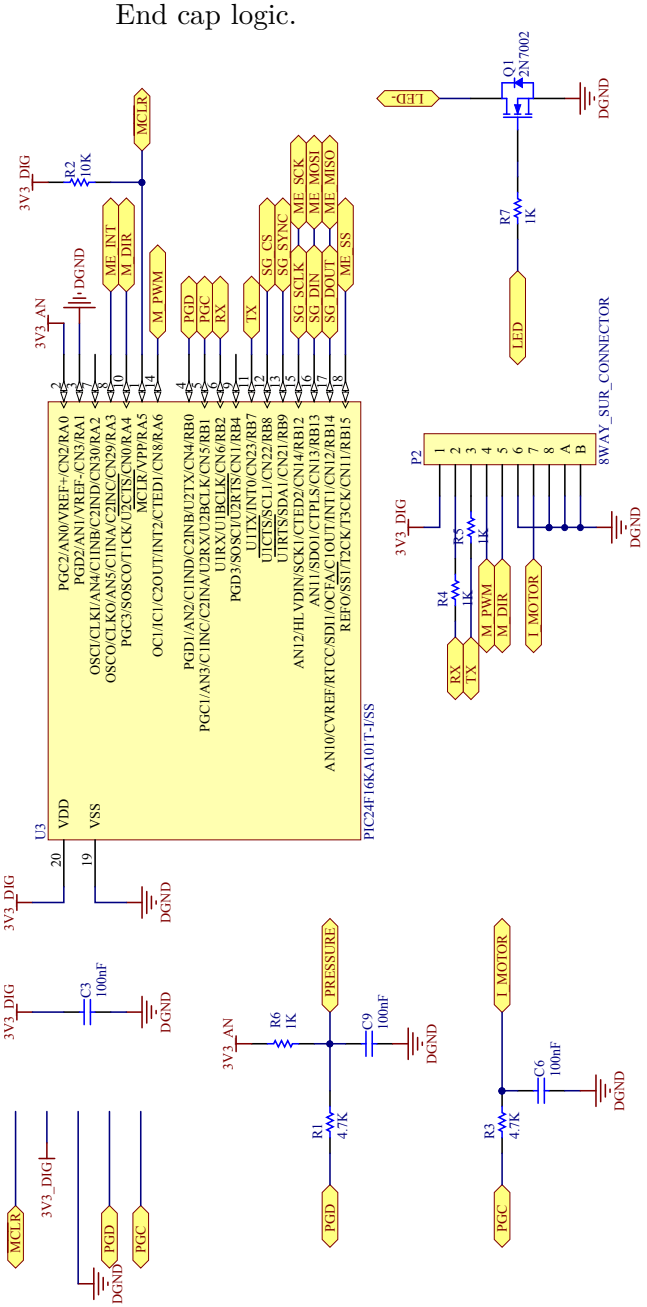
End cap motor driver.



End cap to main module connections.



End cap logic.



Bibliography

[1] A. Agogino, V. SunSpiral, and D. Atkinson. Super ball bot - structures for planetary landing and exploration - phase 1 final report. Technical report, NASA Innovative Advanced Concepts (NIAC) program, 2013.

[2] M. Ajallooeian, S. Gay, A. Tuleu, A. Sproewitz, and A. J. Ijspeert. Modular control of limit cycle locomotion over unperceived rough terrain. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3390–3397, 2013.

[3] R. M. Alexander. *Principles of animal locomotion*. Princeton University Press, 2003.

[4] J. Antol, R. L. Chattin, B. M. Copeland, and S. A. Krizan. The NASA Langley Mars tumbleweed rover prototype. In *Proceedings of the AIAA Aerospace Sciences Meeting and Exhibit*, volume 64, 2006.

[5] E. Antonelo and B. Schrauwen. Learning slow features with reservoir computing for biologically-inspired robot localization. *Neural Networks*, 25:178–190, 2012.

[6] L. Appeltant, M. C. Soriano, G. Van Der Sande, J. Danckaert, S. Massar, J. Dambre, B. Schrauwen, C. R. Mirasso, and I. Fischer. Information processing using a single dynamical node as complex system. *Nature Communications*, 2(13 September 2011):468, 2011.

[7] A. Behar, J. Matthews, F. Carsey, and J. Jones. NASA/JPL tumbleweed polar rover. In *IEEE Aerospace Conference*, volume 1, 2004.

[8] N. Bel Hadj Ali and I. Smith. Dynamic behavior and vibration control of a tensegrity structure. *International Journal of Solids and Structures*, 47(9):1285–1296, 2010.

[9] M. P. Bendsoe and O. Sigmund. *Topology optimization: theory, methods and applications*. Springer, 2003.

[10] A. A. Biewener. *Animal locomotion*. Oxford University Press, 2003.

[11] C. M. Bishop et al. *Pattern recognition and machine learning*, volume 1. Springer New York, 2006.

[12] T. Bliss, T. Iwasaki, and H. Bart-Smith. Resonance entrainment of tensegrity structures via CPG control. *Automatica*, 48(11):2791–2800, 2012.

[13] T. Bliss, T. Iwasaki, and H. Bart-Smith. Central pattern generator control of a tensegrity swimmer. *IEEE/ASME Transactions on Mechatronics*, 18(2):586–597, 2013.

[14] T. Bliss, J. Werly, T. Iwasaki, and H. Bart-Smith. Experimental validation of robust resonance entrainment for CPG-controlled tensegrity structures. *IEEE Transactions on Control Systems Technology*, 21(3):666–678, 2013.

[15] V. Bohm and K. Zimmermann. Vibration-driven mobile robots based on single actuated tensegrity structures. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5475–5480, 2013.

[16] A. C. Bostan, R. P. Dum, and P. L. Strick. The basal ganglia communicate with the cerebellum. *Proceedings of the National Academy of Sciences of the United States of America*, 107(18):8452–8456, 2010.

[17] M. Buehner and P. Young. A tighter bound for the Echo State Property. *IEEE Transactions on Neural Networks*, 17:820–4, May 2006.

[18] P. Buteneers, K. Caluwaerts, D. Verstraeten, J. Dambre, and B. Schrauwen. Optimized parameter search for large datasets of the regularization parameter and feature selection for ridge regression. *Neural Processing Letters*, 38(3):403–416, 2013.

[19] K. Byl, A. Shkolnik, S. Prentice, N. Roy, and R. Tedrake. Reliable dynamic motions for a stiff quadruped. In O. Khatib, V. Kumar, and G. Pappas, editors, *Experimental Robotics*, pages 319–328. Springer Berlin / Heidelberg, 2009.

[20] C. Calladine. Buckminster Fuller's "tensegrity" structures and Clerk Maxwell's rules for the construction of stiff frames. *International Journal of Solids and Structures*, 14(2):161–172, 1978.

[21] K. Caluwaerts, J. Despraz, A. Iscen, J. Bruce, A. P. Sabelhaus, B. Schrauwen, and V. SunSpiral. Design and control of compliant tensegrity robots through simulation and hardware validation. *Journal of the Royal Society Interface*, 2014.

[22] K. Caluwaerts, M. D'Haene, D. Verstraeten, and B. Schrauwen. Locomotion without a brain: physical reservoir computing in tensegrity structures. *Artificial Life*, 19(1):35–66, 2013.

[23] K. Caluwaerts and B. Schrauwen. The body as a reservoir: locomotion and sensing with linear feedback. In *2nd International Conference on Morphological Computation*, pages 45–47, Venice, 2011.

[24] K. Caluwaerts, F. wyffels, S. Dieleman, and B. Schrauwen. The spectral radius remains a valid indicator of the echo state property for large reservoirs. In *IEEE International Joint Conference on Neural Networks (IJCNN)*, page 6, 2013.

[25] G. Carwardine. Improvements in elastic equipoising mechanisms, January 1934. UK patent 404615.

[26] V. R. Challa, M. Prasad, Y. Shi, and F. T. Fisher. A vibration energy harvesting device with bidirectional resonance frequency tunability. *Smart Materials and Structures*, 17(1):015035, 2008.

[27] W. L. Chan, D. Arbelaez, F. Bossens, and R. Skelton. Active vibration control of a three-stage tensegrity structure. In *Smart Structures and Materials*, pages 340–346. International Society for Optics and Photonics, 2004.

[28] C. Clopath, A. Longtin, and W. Gerstner. An online hebbian learning rule that performs independent component analysis. *BMC Neuroscience*, 9(Suppl 1):O13, 2008.

[29] R. Connelly. Tensegrity Structures: Why are they stable. In *Rigidity Theory and Applications*, pages 47–54. Plenum Press, New York, 1999.

[30] R. Connelly and A. Back. Mathematics and tensegrity. *American Scientist*, 86(2):142–151, 1998.

[31] CubeSat Project. Revision 13, CubeSat design specification, provisional release. Technical report, 2013.

[32] J. Dambre, D. Verstraeten, B. Schrauwen, and S. Massar. Information processing capacity of dynamical systems. *Scientific Reports*, 2(514), 2012.

[33] N. D. Daw, S. Kakade, and P. Dayan. Opponent interactions between serotonin and dopamine. *Neural Networks*, 15(4):603–616, 2002.

[34] P. Dayan and L. F. Abbott. *Theoretical neuroscience*, volume 31. MIT press Cambridge, MA, 2001.

[35] N. M. M. de Abreu. Old and new results on algebraic connectivity of graphs. *Linear Algebra and Its Applications*, 423(1):53–73, 2007.

[36] M. De Bono and A. V. Maricq. Neuronal substrates of complex behaviors in C. elegans. *Annual Review of Neuroscience*, 28(28):451–501, 2005.

[37] D. de Solla Price. A history of calculating machines. *IEEE Micro*, 4(1):22–52, 1984.

[38] A. Destexhe and M. Rudolph-Lilith. *Neuronal noise*, volume 8. Springer, 2012.

[39] M. A. Diftler, J. Mehling, M. E. Abdallah, N. A. Radford, L. B. Bridgwater, A. M. Sanders, R. S. Askew, D. M. Linn, J. D. Yamokoski, F. Permenter, et al. Robonaut 2-the first humanoid robot in space. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2178–2183, 2011.

[40] F. Duport, B. Schneider, A. Smerieri, M. Haelterman, and S. Massar. All-optical reservoir computing. *Optics Express*, 20(20):22783–22795, 2012.

[41] A. Erturk and D. J. Inman. An experimentally validated bimorph cantilever model for piezoelectric energy harvesting from base excitations. *Smart Materials and Structures*, 18(2):025009, 2009.

[42] A. Farchy, S. Barrett, P. MacAlpine, and P. Stone. Humanoid robots learning to walk faster: From the real world to simulation and back. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 39–46, 2013.

[43] C. T. Farley and O. Gonzalez. Leg stiffness and stride frequency in human running. *Journal of Biomechanics*, 29(2):181–186, 1996.

[44] C. T. Farley, H. H. Houdijk, C. Van Strien, and M. Louie. Mechanism of leg stiffness adjustment for hopping on surfaces of different stiffnesses. *Journal of Applied Physiology*, 85(3):1044–1055, 1998.

[45] C. Fernando and S. Sojakka. Pattern recognition in a bucket. In *Advances in Artificial Life*, pages 588–597. Springer, 2003.

[46] T. C. Ferrée and S. R. Lockery. Computational rules for chemotaxis in the nematode C. Elegans. *Journal of Computational Neuroscience*, 6(3):263–277, 1999.

[47] D. P. Ferris, M. Louie, and C. T. Farley. Running in the real world: adjusting leg stiffness for different surfaces. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 265(1400):989–994, 1998.

[48] E. Fest, K. Shea, B. Domer, and I. F. Smith. Adjustable tensegrity structures. *Journal of Structural Engineering*, 129(4):515–526, 2003.

[49] E. Fest, K. Shea, and I. F. Smith. Active tensegrity structure. *Journal of Structural Engineering*, 130(10):1454–1465, 2004.

[50] R. B. Fette and M. F. Sovinski. Vectran fiber time dependant behavior and additional static loading properties. Technical report, DTIC Document, 2004.

[51] M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23(2):298–305, 1973.

[52] M. Fiers, T. Van Vaerenbergh, K. Caluwaerts, D. Vande Ginste, B. Schrauwen, J. Dambre, and P. Bienstman. Time-domain and frequency-domain modeling of nonlinear optical components on circuit-level using a node-based approach. *Journal of the Optical Society of America B*, 2012.

[53] I. R. Fiete and H. S. Seung. Gradient learning in spiking neural networks by dynamic perturbation of conductances. *Physical Review Letters*, 97(4):5, 2006.

[54] A. Fitzgibbon, M. Pilu, and R. B. Fisher. Direct least square fitting of ellipses. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):476–480, 1999.

[55] S. Fivat. *A new kind of tensegrity robots.* Master's thesis, Cornell University, 2009.

[56] M. French and M. Widden. The spring-and-lever balancing mechanism, george carwardine and the anglepoise lamp. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 214(3):501–508, 2000.

[57] R. B. Fuller. Tensile-integrity structures, November 1962. US Patent 3,063,521.

[58] R. B. Fuller. *Synergetics: Explorations in the Geometry of Thinking*. Scribner, Jan. 1975.

[59] S. Gay, S. Dégallier, U. Pattacini, A. J. Ijspeert, and J. S. Victor. Integration of vision and central pattern generator based locomotion for path planning of a non-holonomic crawling humanoid robot. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 183–189, 2010.

[60] S. Gay, J. Santos-Victor, and A. J. Ijspeert. Learning robot gait stability using neural networks as sensory feedback function for central pattern generators. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 192–201, 2013.

[61] G. H. Golub and V. Pereyra. The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate. *SIAM Journal on Numerical Analysis*, 10(2):413–432, 1973.

[62] N. I. Gould, M. E. Hribar, and J. Nocedal. On the solution of equality constrained quadratic programming problems arising in optimization. *SIAM Journal on Scientific Computing*, 23(4):1376–1395, 2001.

[63] A. Graells Rovira and J. M. Mirats Tur. Control and simulation of a tensegrity-based mobile robot. *Robotics and Autonomous Systems*, 57(5):526–535, 2009.

[64] S. D. Guest. The stiffness of tensegrity structures. *IMA Journal of Applied Mathematics*, 76(1):57–66, 2010.

[65] G. Hajos, J. Jones, A. Behar, and M. Dodd. An overview of wind-driven rovers for planetary exploration. In *43rd AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV*, 2005.

[66] N. Hansen. The CMA evolution strategy: A comparing review. In *Towards a New Evolutionary Computation*, volume 102 of *Studies in Fuzziness and Soft Computing*, chapter 4, pages 75–102. Springer, 2006.

[67] H. Hauser, A. J. Ijspeert, R. M. Füchslin, R. Pfeifer, and W. Maass. Towards a theoretical foundation for morphological computation with compliant bodies. *Biological Cybernetics*, 105:355–370, 2011.

[68] H. Hauser, A. J. Ijspeert, R. M. Füchslin, R. Pfeifer, and W. Maass. The role of feedback in morphological computation with compliant bodies. *Biological Cybernetics*, 106(10):595–613, 2012.

[69] D. O. Hebb. *The organization of behavior.* Wiley, 1949.

[70] M. Hermans and B. Schrauwen. Memory in reservoirs for high dimensional input. In *IEEE International Joint Conference on Neural Networks (IJCNN)*, 2010.

[71] S. Hernandez Juan and J. M. Mirats Tur. A method to generate stable, collision free configurations for tensegrity based robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3769–3774, 2008.

[72] S. Hernandez Juan and J. M. Mirats Tur. Tensegrity frameworks: Static analysis review. *Mechanism and Machine Theory*, 43(7):859 – 881, 2008.

[73] S. Hirai, Y. Koizumi, M. Shibata, M. Wang, and L. Bin. Active shaping of a tensegrity robot via pre-pressure. In *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 19–25, 2013.

[74] G. M. Hoerzer, R. Legenstein, and W. Maass. Emergence of complex computational structures from chaotic neural networks through reward-modulated hebbian learning. *Cerebral Cortex*, 24:677–690, 2012.

[75] N. Hogan. Impedance control: An approach to manipulation: Part ii implementation. *Journal of Dynamic Systems, Measurement, and Control*, 107(1):8–16, 1985.

[76] T. Hongu, M. Takigami, and G. Phillips. *New millennium fibers.* Elsevier, 2005.

[77] E. Hoshi, L. Tremblay, J. Féger, P. L. Carras, and P. L. Strick. The cerebellum communicates with the basal ganglia. *Nature Neuroscience*, 8(11):1491–1493, 2005.

[78] C. L. Hull. *Principles of behavior: An introduction to behavior theory.* Appleton-Century, 1943.

[79] A. Hyvärinen and E. Oja. Independent component analysis: algorithms and applications. *Neural Networks*, 13(4-5):411–430, 2000.

[80] G. Ifrah. *The Universal History of Computing: From the Abacus to Quantum Computing*. John Wiley & Sons, Inc., 2000.

[81] F. Iida and R. Pfeifer. Sensing through body dynamics. *Robotics and Autonomous Systems*, 54(8):631–640, 2006.

[82] A. J. Ijspeert. Central pattern generators for locomotion control in animals and robots: A review. *Neural Networks*, 21(4):642–653, 2008.

[83] A. J. Ijspeert, A. Crespi, D. Ryczko, and J.-M. Cabelguen. From swimming to walking with a salamander robot driven by a spinal cord model. *Science*, 315(5817):1416–1420, 2007.

[84] D. E. Ingber. Tensegrity: The architectural basis of cellular mechanotransduction. *Annual Review of Physiology*, 59(1):575–599, 1997.

[85] D. E. Ingber. Tensegrity I. Cell structure and hierarchical systems biology. *Journal of Cell Science*, 116(7):1157–1173, 2003.

[86] A. Iscen. *Multiagent Learning for Locomotion and Coordination in Tensegrity Robotics*. PhD thesis, Oregon State University, 2014.

[87] A. Iscen, A. Agogino, V. SunSpiral, and K. Tumer. Controlling tensegrity robots through evolution. In *Genetic and Evolutionary Computation Conference (GECCO)*, 2013.

[88] A. Iscen, A. Agogino, V. SunSpiral, and K. Tumer. Learning to control complex tensegrity robots. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1193–1194, 2013.

[89] A. Iscen, K. Caluwaerts, J. Bruce, A. Agogino, V. SunSpiral, and K. Tumer. Learning tensegrity locomotion using open-loop control signals and coevolutionary algorithms. *submitted*, 2014.

[90] H. Jaeger. The "Echo State" approach to analysing and training recurrent neural networks. Technical Report GMD report 148, German National Research Center for Information Technology, 2001.

[91] H. Jaeger. Tutorial on training recurrent neural networks, covering BPTT, RTRL, EKF and the "Echo State Network" approach. Technical Report GMD Report 159, German National Research Center for Information Technology, 2002.

[92] H. Jaeger. Controlling recurrent neural networks by conceptors. *arXiv preprint arXiv:1403.3369*, 2014.

[93] H. Jaeger. The conceptual logic of neural dynamics. *submitted*, 2014.

[94] H. Jaeger and H. Haas. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science*, 304(5667):78–80, 2004.

[95] J. A. Jones. Inflatable robotics for planetary applications. In *6th International Symposium on Artificial Intelligence, Robotics & Automation in Space: A New Space Odyssey, Montreal, Canada, June 18, 2001.* Pasadena, CA: Jet Propulsion Laboratory, National Aeronautics and Space Administration, 2001., 2001.

[96] T. Kailath, A. H. Sayed, and B. Hassibi. *Linear estimation*. Number 3 in Information and System Sciences. Prentice Hall, 2000.

[97] M. Khazanov, B. Humphreys, W. Keat, and J. Rieffel. Exploiting dynamical complexity in a physical tensegrity robot to achieve locomotion. In *Advances in Artificial Life, ECAL*, volume 12, pages 965–972, 2013.

[98] Y. Koizumi, M. Shibata, and S. Hirai. Rolling tensegrity driven by pneumatic soft actuators. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1988–1993, 2012.

[99] S. Korkmaz, N. B. H. Ali, and I. F. Smith. Configuration of control system for damage tolerance of a tensegrity bridge. *Advanced Engineering Informatics*, 26(1):145–155, 2012.

[100] L. Larger, M. Soriano, D. Brunner, L. Appeltant, J. M. Gutiérrez, L. Pesquera, C. R. Mirasso, and I. Fischer. Photonic information processing beyond turing: an optoelectronic implementation of reservoir computing. *Optics Express*, 20(3):3241–3249, 2012.

[101] D. Lee. The mission loads environment and structural design of the mars science laboratory. Technical report, Jet Propulsion Laboratory, National Aeronautics and Space Administration, 2012.

[102] R. Legenstein, S. M. Chase, A. B. Schwartz, and W. Maass. A reward-modulated Hebbian learning rule can explain experimentally observed network reorganization in a brain control task. *Journal of Neuroscience*, 30(25):8400–8410, 2010.

[103] S. C. Lenaghan, C. A. Davis, W. R. Henson, Z. Zhang, and M. Zhang. High-speed microscopic imaging of flagella motility and swimming in Giardia lamblia trophozoites. *Proceedings of the National Academy of Sciences of the United States of America*, 108(34):E550–E558, 2011.

[104] J. C. Liao. A review of fish swimming mechanics and behaviour in altered flows. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, 362(1487):1973–93, 2007.

[105] T. Liedl, B. Högberg, J. Tytell, D. E. Ingber, and W. M. Shih. Self-assembly of three-dimensional prestressed tensegrity structures from dna. *Nature Nanotechnology*, 5(7):520–524, 2010.

[106] H.-T. Lin, G. G. Leisk, and B. Trimmer. GoQBot: a caterpillar-inspired soft-bodied rolling robot. *Bioinspiration & Biomimetics*, 6(2):026007, 2011.

[107] H. Lipson. A relaxation method for simulating the kinematics of compound nonlinear mechanisms. *Journal of Mechanical Design*, 128(4):719–728, 2006.

[108] Y. Loewenstein. Robustness of learning that is based on covariance-driven synaptic plasticity. *PLoS Computational Biology*, 4(3):10, 2008.

[109] Y. Loewenstein and H. S. Seung. Operant matching is a generic outcome of synaptic plasticity based on the covariance between reward and neural activity. *Proceedings of the National Academy of Sciences of the United States of America*, 103(41):15224–15229, 2006.

[110] Y. Loewenstein and H. S. Seung. Operant matching is a generic outcome of synaptic plasticity based on the covariance between reward and neural activity. *Proceedings of the National Academy of Sciences of the United States of America*, 103(41):15224–15229, 2006.

[111] M. Lukoševičius and H. Jaeger. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3:127–149, 2009.

[112] Z.-D. Ma, N. Kikuchi, and H.-C. Cheng. Topological design for vibrating structures. *Computer Methods in Applied Mechanics and Engineering*, 121(1):259–280, 1995.

[113] W. Maass, T. Natschlager, and H. Markram. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*, 14(11):2531–2560, 2002.

[114] P. Malerba, M. Patelli, and M. Quagliaroli. An extended force density method for the form finding of cable systems with new forms. *Structural Engineering and Mechanics*, 42(2):191–210, 2012.

[115] G. Manjunath and H. Jaeger. Echo State Property linked to an input: Exploring a fundamental characteristic of recurrent neural networks. *Neural Computation*, 25(3):671–696, 2013.

[116] K. Matsuoka. Sustained oscillations generated by mutually inhibiting neurons with adaptation. *Biological Cybernetics*, 52(6):367–376, 1985.

[117] K. Matsuoka. Mechanisms of frequency and pattern control in the neural rhythm generators. *Biological Cybernetics*, 56(5-6):345–353, 1987.

[118] J. C. Maxwell. L. on the calculation of the equilibrium and stiffness of frames. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 27(182):294–299, 1864.

[119] P. Mazzoni, R. A. Andersen, and M. I. Jordan. A More Biologically Plausible Learning Rule for Neural Networks. *Proceedings of the National Academy of Sciences of the United States of America*, 88(10):4433–4437, 1991.

[120] W. McMahan, V. Chitrakaran, M. Csencsits, D. Dawson, I. D. Walker, B. A. Jones, M. Pritts, D. Dienno, M. Grissom, and C. D. Rahn. Field trials and testing of the octarm continuum manipulator. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2336–2341, 2006.

[121] A. Micheletti and W. O. Williams. A marching procedure for form-finding for tensegrity structures. *Journal of Mechanics of Materials and Structures*, 2(5):101–26, 2007.

[122] M. Minsky. Steps toward artificial intelligence. *Proceedings of the IRE*, 49(1):8–30, 1961.

[123] J. M. Mirats Tur. On the Movement of Tensegrity Structures. *International Journal of Space Structures*, 25(1):1–14, 2010.

[124] B. Mohar and Y. Alavi. The laplacian spectrum of graphs. *Graph Theory, Combinatorics, and Applications*, 2:871–898, 1991.

[125] K. Moored and H. Bart-Smith. Investigation of clustered actuation in tensegrity structures. *International Journal of Solids and Structures*, 46(17):3272–3281, 2009.

[126] R. Motro. *Tensegrity: Structural systems for the future*. Butterworth-Heinemann, 2003.

[127] K. Nakajima, H. Hauser, R. Kang, E. Guglielmino, D. Caldwell, and R. Pfeifer. Computing with a muscular-hydrostat system. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1504–1511, May 2013.

[128] S. Nishiwaki, M. I. Frecker, S. Min, and N. Kikuchi. Topology optimization of compliant mechanisms using the homogenization method. *International Journal for Numerical Methods in Engineering*, 42:535–559, 1998.

[129] R. Nugent, R. Munakata, A. Chin, R. Coelho, and J. Puig-Suari. The CubeSat: The picosatellite standard for research and education. *Aerospace Engineering*, 805:756–5087, 2008.

[130] E. Oja. A simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, 15(3):267–273, 1982.

[131] O. Orki, A. Ayali, O. Shai, and U. Ben-Hanan. Modeling of caterpillar crawl using novel tensegrity structures. *Bioinspiration & Biomimetics*, 7(4):046006, 2012.

[132] L. Panait. Theoretical convergence guarantees for cooperative coevolutionary algorithms. *Evolutionary Computation*, 18:581–615, 2010.

[133] Y. Paquot, F. Duport, A. Smerieri, J. Dambre, B. Schrauwen, M. Haelterman, and S. Massar. Optoelectronic reservoir computing. *Scientific Reports*, 2, 2012.

[134] C. Paul. Morphological computation: A basis for the analysis of morphology and control requirements. *Robotics and Autonomous Systems*, 54(8):619–630, 2006.

[135] C. Paul, H. Lipson, and F. J. V. Cuevas. Evolutionary form-finding of tensegrity structures. *Genetic and Evolutionary Computation Conference (GECCO)*, page 3, 2005.

[136] C. Paul, J. W. Roberts, H. Lipson, and F. J. Valero-Cuevas. Gait production in a tensegrity based robot. *International Conference on Advanced Robotics (ICAR)*, pages 216–222, 2005.

[137] A. Pedis. Epigrams on programming. *ACM Sigplan Notices*, pages 7–13, 1981.

[138] K. B. Petersen and M. S. Pedersen. The matrix cookbook. *Technical University of Denmark*, 2008.

[139] M. G. Raja and S. Narayanan. Active control of tensegrity structures under random excitation. *Smart Materials and Structures*, 16(3):809, 2007.

[140] M. Reis, X. Yu, N. Maheshwari, and F. Iida. Morphological computation of multi-gaited robot locomotion based on free vibration. *Artificial life*, 19(1):97–114, 2013.

[141] L. Rhode-Barbarigos, N. B. Hadj Ali, R. Motro, and I. F. Smith. Designing tensegrity modules for pedestrian bridges. *Engineering Structures*, 32(4):1158–1167, 2010.

[142] L. Rhode-Barbarigos, C. Schulin, N. B. Hadj Ali, R. Motro, and I. F. Smith. Mechanism-based approach for the deployment of a tensegrity-ring module. *Journal of Structural Engineering*, 138(4):539–548, 2012.

[143] J. Rieffel, F. J. Valero-Cuevas, and H. Lipson. Automated discovery and optimization of large irregular tensegrity structures. *Computers & Structures*, 87(5):368–379, 2009.

[144] J. Rieffel, F. J. Valero-Cuevas, and H. Lipson. Morphological communication: Exploiting coupled dynamics in a complex mechanical structure to achieve locomotion. *Journal of the Royal Society Interface the Royal Society*, 7(45):613–621, 2010.

[145] L. Righetti, J. Buchli, and A. J. Ijspeert. From dynamic hebbian learning for oscillators to adaptive central pattern generators. In *International Symposium on Adaptive Motion of Animals and Machines (AMAM)*, page 45, 2005.

[146] D. L. Ringo. Flagellar motion and fine structure of the flagellar apparatus in Chlamydomonas. *The Journal of Cell Biology*, 33(3):543–571, 1967.

[147] T. J. Roberts and E. Azizi. Flexible mechanisms: the diverse roles of biological springs in vertebrate movement. *The Journal of Experimental Biology*, 214(3):353–61, 2011.

[148] R. Saegusa, G. Metta, G. Sandini, and S. Sakka. Active motor babbling for sensorimotor learning. In *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 794–799, 2009.

[149] M. Said, B. Dingwall, A. Gupta, A. Seyam, G. Mock, and T. Theyson. Investigation of ultra violet (uv) resistance for high strength fibers. *Advances in Space Research*, 37(11):2052–2058, 2006.

[150] T. Sanger. Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural Networks*, 2(6):459–473, 1989.

[151] H.-J. Schek. The force density method for form-finding and computation of general networks. *Computer Methods in Applied Mechanics and Engineering*, 3(1):115–134, 1974.

[152] M. Schenk, S. D. Guest, and J. Herder. Zero stiffness tensegrity structures. *International Journal of Solids and Structures*, 44(20):6569–6583, 2007.

[153] W. Schultz. Predictive reward signal of dopamine neurons. *Journal of Neurophysiology*, 80(1):1–27, 1998.

[154] W. Schultz. Multiple reward signals in the brain. *Nature Reviews Neuroscience*, 1(3):199–207, 2000.

[155] W. Schultz. Getting formal with dopamine and reward. *Neuron*, 36(2):241–263, 2002.

[156] M. Shibata, F. Saijyo, and S. Hirai. Crawling by body deformation of tensegrity structure robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4375–4380, 2009.

[157] R. Skelton and M. C. de Oliveira. *Tensegrity systems*. Springer, 2009.

[158] K. D. Snelson. Continuous tension, Feb. 16 1965. US Patent 3,169,611.

[159] A. Soltani and X.-J. Wang. A biophysically based neural model of matching law behavior: melioration by stochastic synapses. *Journal of Neuroscience*, 26(14):3731–3744, 2006.

[160] A. Soltoggio and J. J. Steil. Solving the distal reward problem with rare correlations. *Neural computation*, 25(4):940–978, 2013.

[161] S. Song and H. Geyer. Regulating speed and generating large speed transitions in a neuromuscular human walking model. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 511–516, 2012.

[162] J. J. Steil. Backpropagation-Decorrelation: Online recurrent learning with O(N) complexity. In *IEEE International Joint Conference on Neural Networks (IJCNN)*, pages 843–848, 2004.

[163] C. Sultan. Tensegrity: 60 years of art, science, and engineering. In E. van der Giessen and H. Aref, editors, *Advances in Applied Mechanics*, chapter 2, pages 69–145. Elsevier, 43 edition, 2009.

[164] C. Sultan, M. Corless, and R. Skelton. Peak-to-peak control of an adaptive tensegrity space telescope. In *1999 Symposium on Smart Structures and Materials*, pages 190–201. International Society for Optics and Photonics, 1999.

[165] C. Sultan, M. Corless, and R. Skelton. Tensegrity flight simulator. *Journal of Guidance, Control, and Dynamics*, 23(6):1055–1064, 2000.

[166] C. Sultan, M. Corless, and R. Skelton. Linear dynamics of tensegrity structures. *Engineering Structures*, 24(6):671–685, 2002.

[167] V. SunSpiral, G. Gorospe, J. Bruce, A. Iscen, G. Korbel, S. Milam, A. Agogino, and D. Atkinson. Tensegrity based probes for planetary exploration: Entry, descent and landing (EDL) and surface mobility analysis. In *10th International Planetary Probe Workshop (IPPW)*, July 2013.

[168] D. Sussillo and L. F. Abbott. Generating coherent patterns of activity from chaotic neural networks. *Neuron*, 63(4):544–557, 2009.

[169] H. Terashima, S. Kojima, and M. Homma. Flagellar motility in bacteria structure and function of flagellar motor. *International Review of Cell and Molecular Biology*, 270(08):39–85, 2008.

[170] K. M. Thormann and A. Paulick. Tuning the flagellar motor. *Microbiology*, 156(Pt 5):1275–1283, 2010.

[171] A. Tibert and S. Pellegrino. Review of form-finding methods for tensegrity structures. *International Journal of Space Structures*, 18(4):209–223, 2003.

[172] G. Tibert. *Deployable Tensegrity Structures for Space Applications*. PhD thesis, Royal Institute of Technology, 2002.

[173] B. R. Tietz, R. W. Carnahan, R. J. Bachmann, R. D. Quinn, and V. SunSpiral. Tetraspine: Robust terrain handling on a tensegrity robot using central pattern generators. In *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 261–267, 2013.

[174] A. N. Tikhonov and V. I. Arsenin. *Solutions of ill-posed problems*. Scripta series in mathematics. Winston/Halsted Press, 1977.

[175] F. Tisseur and K. Meerbergen. The quadratic eigenvalue problem. *SIAM Review*, 43(2):235–286, 2001.

[176] H. C. Tran and J. Lee. Advanced form-finding of tensegrity structures. *Computers & Structures*, 88(3):237–246, 2010.

[177] H. C. Tran and J. Lee. Form-finding of tensegrity structures with multiple states of self-stress. *Acta Mechanica*, 222(1-2):131–147, 2011.

[178] D. Trivedi, C. D. Rahn, W. M. Kier, and I. D. Walker. Soft robotics: Biological inspiration, state of the art, and future research. *Applied Bionics and Biomechanics*, 5(3):99–117, 2008.

[179] N. G. Tsagarakis, M. Laffranchi, B. Vanderborght, and D. G. Caldwell. A compact soft actuator unit for small scale human friendly robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4356–4362, 2009.

[180] A. M. Turing. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42:230–265, 1936.

[181] K. Vandoorne, J. Dambre, D. Verstraeten, B. Schrauwen, and P. Bienstman. Parallel reservoir computing using optical amplifiers. *IEEE Transactions on Neural Networks*, 22(9):1469–1481, 2011.

[182] K. Vandoorne, P. Mechet, T. Van Vaerenbergh, M. Fiers, G. Morthier, D. Verstraeten, B. Schrauwen, J. Dambre, and P. Bienstman. Experimental demonstration of reservoir computing on a silicon photonics chip. *Nature Communications*, 5, 2014.

[183] D. Verstraeten. *Reservoir Computing: computation with dynamical systems*. PhD thesis, Ghent University, 2009.

[184] D. Verstraeten, J. Dambre, X. Dutoit, and B. Schrauwen. Memory versus non-linearity in reservoirs. In *IEEE International Joint Conference on Neural Networks (IJCNN)*, 2010.

[185] D. Verstraeten, B. Schrauwen, M. D'Haene, and D. Stroobandt. 2007 special issue: An experimental unification of reservoir computing methods. *Neural Networks*, 20(3):391–403, 2007.

[186] S. Vogel. *Glimpses of creatures in their physical worlds*. Princeton University Press, 2009.

[187] I. D. Walker, D. M. Dawson, T. Flash, F. W. Grasso, R. T. Hanlon, B. Hochner, W. M. Kier, C. C. Pagano, C. D. Rahn, and Q. M. Zhang.

Continuum robot arms inspired by cephalopods. In *Defense and Security*, pages 303–314. International Society for Optics and Photonics, 2005.

[188] N. Wang, J. D. Tytell, and D. E. Ingber. Mechanotransduction at a distance: Mechanically coupling the extracellular matrix with the nucleus. *Nature Reviews Molecular Cell Biology*, 10(1):75–82, 2009.

[189] D. Williamson, R. Skelton, and J. Han. Equilibrium conditions of a tensegrity structure. *International Journal of Solids and structures*, 40(23):6347–6367, 2003.

[190] R. A. Wise and P.-P. Rompré. Brain dopamine and reward. *Annual Review of Psychology*, 40(1):191–225, 1989.

[191] A. S. Wroldsen. *Modelling and control of tensegrity structures.* PhD thesis, Norwegian University of Science and Technology, 2007.

[192] F. wyffels. *Sequence generation with reservoir computing systems.* PhD thesis, Ghent University, 2013.

[193] F. wyffels, M. D'Haene, T. Waegeman, K. Caluwaerts, C. Nunes, and B. Schrauwen. Realization of a passive compliant robot dog. In *IEEE RAS & EMBS International conference on Biomedical Robotics and Biomechanics*, pages 882–886, Tokyo, 2010. IEEE.

[194] F. wyffels, B. Schrauwen, and D. Stroobandt. Stable output feedback in reservoir computing using ridge regression. *International Conference on Artificial Neural Networks (ICANN)*, pages 808–817, 2008.

[195] K. Yamane and Y. Nakamura. Stable penalty-based model of frictional contacts. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1904–1909. IEEE, 2006.

[196] I. Yildiz, H. Jaeger, and S. Kiebel. Re-visting the Echo State Property. *Neural Networks*, 35:1–9, 2012.

[197] J. Zhang and M. Ohsaki. Adaptive force density method for form-finding problem of tensegrity structures. *International Journal of Solids and Structures*, 43(18-19):5658–5673, 2006.

[198] L.-Y. Zhang, Y. Li, Y.-P. Cao, X.-Q. Feng, and H. Gao. Self-equilibrium and super-stability of truncated regular polyhedral tensegrity structures: a unified analytical solution. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science*, 468(2147):3323–3347, 2012.

[199] Q. Zhao, K. Nakajima, H. Sumioka, H. Hauser, and R. Pfeifer. Spine dynamics as a computational resource in spine-driven quadruped locomotion. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1445–1451, Nov 2013.

[200] Q. Zhao, K. Nakajima, H. Sumioka, X. Yu, and R. Pfeifer. Embodiment enables the spinal engine in quadruped robot locomotion. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2449–2456, Oct 2012.

[201] M. Ziegler, F. Iida, and R. Pfeifer. "Cheap" underwater locomotion: Roles of morphological properties and behavioural diversity. In *International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines (CLAWAR)*, 2006.