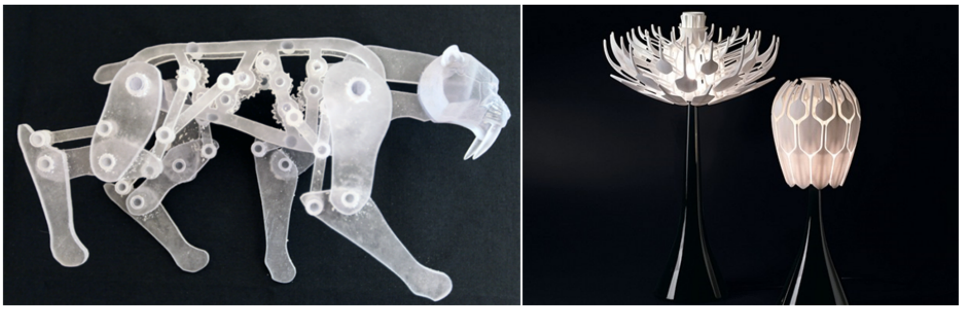
Thesis Title



Jonas Stolle

Bachelor Thesis
May 2021

Supervisors:
Moritz Geilinger
Prof. Dr. Stelian Coros

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Abstract

This thesis addresses the development of a novel sample thesis. We analyze the requirements of a general template, as it can be used with the LaTeX text processing system. (And so on…) The abstract should not exceed half a page in size!

Contents

List of Figures

List of Tables

1

Introduction

also state here that we are dealing with mechanical systems in particular

2

Related Work

Sample references are [?] and [?].

2.1. Appearance Modeling

2.1.1. Taxonomy

at the end of the one paper talk about how the goal is to extend on this by applying it to different and more complex systems.

3

Your Central Work

3.1. Fundamentals and Problem Formulation

In this section describes the relvant physical definitions from which the robustness measure is derived and which will be referred to in the later code implementation. all of which are important for understanding and some of which are directly used for the implementation.

Examples will be given n terms of laikag quadruped robot as most of the testing of the framework was done with that.

3.1.1. Dynamical Systems

Dynamical systems are distinguished by an evolution of their state $\mathbf{x}(t)$ through time. This evolution can be fully described by a set of ordinary differential equations of the form $\dot{\mathbf{x}}(t) = \mathbf{F}(\mathbf{x}(t), t)$, where $\mathbf{F}$ is some nonlinear function. This simplifies to $\dot{\mathbf{x}}(t) = \mathbf{F}(\mathbf{x}(t))$ under the assumption that the dynamical system is autonomous, i.e. is not explicitly dependent on time. When solving for the explicit solution $\mathbf{x}(t)$, an initial condition $\mathbf{x_0} = \mathbf{x}(t_0)$ is required, which is a system state at an initial time. For simplicity and without loss of generality for autonomous systems, we set $t_0 = 0$. With this the Initial Value Problem (IVP) can be formulated:

$$find\ \ \mathbf{x}(t) \tag{3.1}$$

$$s.t.\ \dot{\mathbf{x}}(t) = \mathbf{F}(\mathbf{x}(t)) \tag{3.2}$$

$$and\ \ \mathbf{x}(0) = \mathbf{x}_0 \tag{3.3}$$

We denote the solution to the IVP as $\mathbf{x}(t, \mathbf{x}_0)$. It represents trajectory of the system state through time given an initial condition. The space in which this trajectory lies is spanned by all possible states $\mathbf{x}$ and termed the $state\ space$. Note that any future state of the trajectoy $\mathbf{x}(\tau, \mathbf{x}_0)$ at time τ can be taken as an initial condition of the IVP itself. It turns out that the new solution coincides with the initial one, i.e. $\mathbf{x}(t, \mathbf{x}_0) = \mathbf{x}(t, \mathbf{x}(\tau, \mathbf{x}_0))$, which illustrates that any state $\mathbf{x}(t)$ of a trajectory $\mathbf{x}(t, \mathbf{x}_0)$ is suffitient to represent the trajectory as a whole.

Mechanical systems (on which we will focus on form here on out) tend to be described in terms of so called generalized coordinates:

$$\mathbf{q}(t) = \begin{pmatrix} q_1(t) & q_2(t) & \dots & q_n(t) \end{pmatrix}^\intercal. \tag{3.4}$$

They are the minimal set of coordinates needed to fully describe the position and orientation of all of the systems elements. Their dimension n cooincides with the number of degrees of freedom of the system. The corresponding differential equations are of second order, depending on $\ddot{\mathbf{q}}(t)$ in addition to $\dot{\mathbf{q}}(t)$ and $\mathbf{q}(t)$. Simply put, this is due to their derivation by Newton's second method, where forces acting on the system are related to the second time derivative via $F = ma$. Through an order reduction (Appendix) these differential equations can be cast into the reviously mentioned general form 3.1, where

$$\mathbf{x}(t) = \begin{pmatrix} \mathbf{q}(t) & \dot{\mathbf{q}}(t) \end{pmatrix}^\intercal.$$

This implies that in order to solve the IVP, initial conditions $\mathbf{q}_0$ and $\dot{\mathbf{q}}_0$ are required. We can also state that for a system with n degrees of freedom, $\mathbf{x}(t) \in \mathbb{R}^{2n}$. The particular state space spanned by generalized coordinates and velocities is termed the $phase\ space$. Within it, any states are single points and state trajectories are smooth curves. The phase space is used for generalizable qualitative analysis of the behaviour of nonlinear dynamical systems and plays a pivotal role in the fromulation of the robustness measure. It shall be stressed at this point, that any instantaneous configuration of the isolated system one can think of is a point in the phase space and it is impossible for the system state to leave or exist outside of it without fundamentally changing the system.

When discussing high level concepts we will refer to the system state $\mathbf{x}(t)$ for simplicity, while in the code implementation the generalized coordinates $\mathbf{q}(t)$ and velocities $\dot{\mathbf{q}}(t)$ will be more relevant. Keep in mind that both are equivalent.

3.1.2. Attractors and Convergence

In nonlinear dynamical systems, there may exist sets of states in the phase space which show an attracting behaviour. By "attracting" we mean that once a trajectory reaches an element of such a set, all of its future states will also be part of that set. Define an attractor as a set of states:

$$\mathbf{A} \subset phase\ space, \tag{3.5}$$

$$s.t.\ if\ \mathbf{x}(t_0) = \mathbf{x}_0\ \in \mathbf{A}, \tag{3.6}$$

$$\mathbf{x}(t, \mathbf{x}_0) \in \mathbf{A} \quad \forall\, t > t_0. \tag{3.7}$$

Attractors can be divided into two fundamental variants. If the attracting set consists of only one state, it is called a fixed point. Fixed points are associated with the condition

$$\dot{\mathbf{x}}(t) = \mathbf{0}\ \ \forall\, t \in \mathbb{R}, \tag{3.8}$$

meaning that the state of the fixed point is unchanging and the related trajectory is reduced to a point in the phase space. Whether a state $\mathbf{x}(t)$ is a fixed point can be easily determined by checking $\mathbf{F}(\mathbf{x}(t)) = \mathbf{0}$. A classical example of a fixed point is the stable bottom position of a pendulum, where if given zero initial velocity, it won't leave the stable position. This is not the case for any other position as gravity will act on the mass (except for the inverted position, however this is practically not realizable). In the general case with A containing of more than one state, (reference above eq) is not true. Rather the trajectory is moving through the sets of A, visiting every element at some point in time and returning to it at future times in a periodic fashion. These types of attractors are called limit cycles. Finding them is generally a hard problem, but for simpler cases one can check:

$$If\ for\ \mathbf{x}(t, \mathbf{x}_0),\ t > 0 \quad \exists \quad \mathbf{x}(\tau) = \mathbf{x}(\tau + h),\ \tau > 0,\ h > 0, \tag{3.9}$$

$$\{\ \mathbf{x}(t) \mid t \in [\tau, \tau + h)\ \},\ is\ a\ \textbf{limit cycle}. \tag{3.10}$$

An example for this case are the compliant linkages described in (ref strandbeest compliant version), where the end effector follows a cyclic trajectory, i.e. set of states if undisturbed. In the code implementation section, we provide methods for dealing with both types of attractors and the problem of applying the continuous analysis in a discrete setting.

For any initial condition not part of the attractor, the related trajectory may land and stay on the attractor after some time $t > t_0$. We define this occurance:

$$Given\ an\ attractor\ \mathbf{A},\ for\ any\ \mathbf{x}_0 \notin \mathbf{A} \tag{3.11}$$

$$if\ \lim_{t \to \infty} \mathbf{x}(t, \mathbf{x}_0) \in \mathbf{A}\ \Rightarrow\ \mathbf{x}(t, \mathbf{x}_0)\ \textbf{converges}\ to\ \mathbf{A} \tag{3.12}$$

Should the definition above not hold, we denote the trajectoy as diverging (as in cell mapping methods).

Note that there may exist any number of attractors in the phase space of a system (reference paper with multitude of attractors). Convergence is always defined with respect to a particular attractor $\mathbf{A}$, which needs to be specified. Therefore if the trajectory converges to a different attractor, it is still defined as diverging from the attractor of interest.

The set of all states for which the related trajectories converge to the attractor of interest is called the $Basin\ of\ Attraction$ (BoA) and is defined as:

$$\{\ \mathbf{x}_0 \in \mathbb{R}^{2n} \mid \lim_{t \to \infty} \mathbf{x}(t, \mathbf{x}_0) \in \mathbf{A}\ \} \tag{3.13}$$

where $\mathbf{A}$ is an attractor as defined in 3.8.

From here on we will represent the general attractor of interest with $\mathbf{A}$. Additionally, we define disturbances as any event that acts upon the system, inducing a state change. This new state implies a new trajectory as a solution of the IVP, meaning the future behaviour of the system may differ vastly from the undisturbed case.

3.1.3. Robustness Measure

The robustness measure proposed in REF and implemented in the following is derived using the concepts of nonlinear dynamics outlined in the previous section. In the actual implementation and testing compromises have to be made (rephrase) in order to improve feasibility, which is why we propose a slight reframing of some definitions to accomodate for those variations.

First we formulate a binary classification of trajectories in successful or failed recoveries from disturbances. Given a dynamical system, assume that it shows some stable and desirable behaviour in its undisturbed state. Examples of this might be holding a specific pose or walking with a periodic gait. We can interpret these types of behaviour as attractors in the phase space of the system. Applying disturbances may be move the system state outside of the attracting set. Following the new trajectory, we can detemine whether the system will converge back to the attractor. In such a case successful recovery of the system from the disturbance is achieved; one might say the system is robust to that disturbance.

Note that a this holds for multiple consecutive or distributed disturbances as well, as only eventual convergence of the trajectory is relevant. Here the IVP has to be solved repeatedly after every new disturbance.

Because of the direct relation between disturbances and state changes, one may choose to only analyze the convergence behaviour of states in phase space. This is how REF formulated their robustness measure. The size of the set of states in the phase space which converge back to the attractor is a measure of robustness as it is a representation of the amount of disturbances the system can recover from. Notice that this is precisely the basin of attraction defined in REF. As outlined in REF, finding the size of the BOA is nontrivial, for which REF introduced the conervative measure of the minimal Radius as the shortest distance from the origin to the boundary of the BOA.

While this approach has a strong foundation in nonlinear dynamics theory, it quickly becomes infeasible for more complex applications. In the paper the most complex system had six DOFs and the computational time was already high (numbers??). The system on which most of the test were executed is a simple model of a quadruped which aready has 78 degrees of freedom implying a 156 dimensional phase space. If one wanted to discretize the space with just 3 nodes along each dimension, $2.697x10^74$ initial conditions would need to be evaluated. One simplification would be choosing only small number of dimensions of the phase space, however then the robustness measure is valid only for a part of the system, making it applications less useful, as one could only look at the robustness of a single leg for example. A different way of choosing a subset of disturbacnes is needed.

Here we remind ourselves that the disturbed states in the phase space are results of underlying disturbances. Instead of just sampling initial conditions, we propose applying disturbances directly to the model in the simulation. We define a disturbance space from which we will sample disturbances and in which we will measure robustness. The idea is to apply the same concept of a minimal Radius in this space to approximate the size of the set of disturbances the system can recover from in order to measure robustness. Note that recovery of the system under a disturbance will still be evaluated by checking convergence of the state to the attractor in the phase space.

(mathematical formulation of robustness measure)

"find the minimal distance to boundary of converging set, where converging set is all elements of DS, for which trajectory will go to attractor in limit within the phase space"

The goal of finding a robustness measure can be though of as $RM : \mathbb{R}^p \to \mathbb{R}$, i.e. a mapping from the p-dimensional parameter space to a scalar value.

3.1.4. Parameter space and Disturbance space

In order to apply the concept of the minimal radius as a measure of robustness in the DS, we must impose some conditions on it. In the phase space, the origin is generally chosen such that the attractor of interest lies on or in close viscinity to it. This means sampled initial conditions with a small norm represent no or very small disturbances, guaranteeing recovery if chosen small enough. The minimal Radius approach builds on the fact that the origin lies within the converging set and the it's boundary is reached at some point when moving farther away. For this reason me must ensure that in the DS the origin also represents the state being on the attractor, implying no disturbance. Each element of the disturbance space is just a d dimensional vector of coordinates representing some disturbance. To impose the above condition we must ensure that the 0 vector in the DS represents no disturbance. Taking oscillations as an example we might want to represent them in a 2 dimensional DS with amplitude and frequency as the coordinates. An oscillation with either 0 amplitude or 0 frequency implies no oscillation at all, making this choice valid. An invalid example of a coordinate is the direction of a force applied. If the angle of attack is 0, the disturbance itself is clearly non zero, breaking our condition.

For any set of valid coordinates, their scaling wrt each other turns out to stronly affect the resulting robustness measure. Just choosing different units for any coordinate will stretch the DS along the corresponding dimension, changing the shape of the converging set and in turn changing the minimal Radius to its boundary. This issue is alluded to in the paper REF by allowing the minimal radius to trace an elliptical shape, which corresponds to rescaling one dimension. There seems to be no comprehensive solution to this issue, which is why we suggest finding a well posed DS by trial and error and not changing it as long as possible. Conversely, this problem may also be leveraged for controlling later optimization of robustness. If the boundary of the converging set is a given distance away along a dimensions, shrinking that dimension will move that boundary closer to the origin, making it more likely that the minimal Radius lies in that direction. With this the scaling of the coordinates could be seen as a weighting of how important robustness against that part of the disturbance is. (illustration)

For the aforementioned optimization, we define the parameter space (PS) to perform the optimization in. Each element of the PS is a vector of parameters describing some parts of the underlying system. A different vector of parameters will fundamentally change the system and for each element of the PS, robustness can be evaluated. Eventually we want to find the vector of parameters for which robustness is maximized. We can formulate the optimization problem:

Note that the choice of DS is fundamental for the results.

3.2. Code Implementation

This section details the implementations of the robustness measure and particular challenges that were encountered and overcome. (rephrase)

3.2.1. Framework Overview

differential equations are implicitly represented as simulation objects in dde. Their parameters can be set. The state x as well. Use the solver to compute the trajectory under a disturbance for a given amount of timesteps. check if trajectory converges or diverges. Do this repeatedly with initial conditions sample talk about how knowing the exact shape of the basin of attraction is not necessary for optimization of the robustness, with this and because of the additional work needed to implement the cell mapping algorithms, the method with full trajectories was chosen. For this we need a solver. boom. Great segue!

We choose the method from REF over cell mapping because of it's simplicity in implementation with the given solver. I.e. cell mapping also needs the solver, but in addition also a lot of other structure around it.

specify laikago as an example. Or rather briefly describe what it is and use it to illustrate issue that might arise,

here we want a nice block diagram.

Also here we want

3.2.2. Solver

One of the hardes part of the process is actually finding the state trajectories of the system. Finding an explicit analytical solution to the IVP is possible for simple cases, but very hard if not impossible for more complex systems. Numerical solvers represent a general approach to approximate the trajectories. For this the differential equations of the IVP are integrated over small time steps Δt to approximate future states. Iterating this process gives a discrete approximation of the continuous state trajectory. Note that we represent general discrete points in time with t_n. The state trajectory is therefore just a set of states $\mathbf{x}(t_n)$ at time points $[t_0, t_1, \ldots, t_n]$ with $t_{i+1} - t_i = \Delta t \quad \forall\ i \in [0, n-1]$. A smaller time step will result in a more accurate approximation, however it will take more computational effort to progress through the same amount of time. It is advised to find a time step that is a sufficient compromise between accuracy and cumputational time and keeping it fixed from there on. Here $\Delta t = 0.01$ was found to be appropriate. Still numerical errors and will always be present, fundamentally limiting the precision that can be achieved.

For this project CRL's Differentiable Dynamics Engine (DDE) was provided, doing most of the heavy lifting. It simulates mechanical systems by considering multibody dynamics and contact forces. The latter are modeled via spring dampener elements, the dynamics of which again depend on the time step so it is imperative to keep it constant between different test. Here, it was f..? DDE represents the system states by generalized coorinates and provides the generalized accelerations $\ddot{\mathbf{q}}(t_n)$ in addition to $\mathbf{q}(t_n)$ and $\dot{\mathbf{q}}(t_n)$ for ever iteration of the trajectory. One particularity to keep in mind is the choice y is the vertical axis, while x and z lie in the horizontal plane.

(anything else important to note?)

3.2.3. Detecting Attractors

In order to evaluate the convergence of a trajectory resulting from a disturbance, the attracting set of states must be found. For this we follow the nonlinear dynamics definitions for a general approach.

When setting up a simulation, especially when compliance is present, the undisturbed initial state resulting from the construction is usually not immediatly an attractor. To remedy this it is suggested to run an undisturbed simulation such that the system can settle and a valid attractor can be found. As in (REF seciton), we distinguish between fixed points and limit cycles.

In the case of fixed points, it suffices to check every state along the computed trajectory for the fixed point condition (REF section for fixed points). In the case of DDE checking this is trivial as the generalized accelerations $\ddot{\mathbf{q}}(t_n)$ are provided. The first state at time $t_n = t_{fp}$ for which $\dot{\mathbf{q}}(t_{fp}) = \mathbf{0}$ and $\ddot{\mathbf{q}}(t_{fp}) = \mathbf{0}$ hold, can be saved as an attractor in the form of $\mathbf{x} = \begin{pmatrix} \mathbf{q}(t_{pf}) & \mathbf{0} \end{pmatrix}^\intercal$, for disturbed trajectories to be compared to. If $\ddot{\mathbf{q}}(t_n)$ is not eailsy obtainable, one may check a number of states $\mathbf{x}(t_n),\ t_n > t_{pf}$ for $\dot{\mathbf{q}}(t_n) = \mathbf{0}$. This does not guarantee that state $\mathbf{x}(t_{pf})$ is a fixed point, but it makes it more probable the more additional states are checked. If the state $\mathbf{x}(t_{pf})$ is smaller than some chosen error tolerance.

(Insert example graph of a 1d mass spring system over time.)

Finding limit cycles is a bit more challenging as they may show chaotic behaviour and because of the discretization of time, aliasing effects could arise. For practicality we make the following assumptions. The period T of the limit cycle coincides with the period of the forcing period that causes the periodic behaviour in the first place. So if we have a controller tasked to move a leg in a periodic fashion, we assume the actual end effector trajectory to have that exact same period. Note that this is not true generally and needs to be verified. In addition we choose the forcing period of the system to be an integer multiple of the time step $T = m \cdot \Delta t, m \in \mathbb{N}$ to mitigate aliasing. In this scenario we can apply the conditions for a limit cycle in continuous time to the discrete case directly. If a state $\mathbf{x}(t_n) = \mathbf{x}(t_n + T)$ is found, the set of states $\{\ \mathbf{x}(t_n) \mid t_n \in [t_i, t_{i+1}, \ldots, t_i + T]\ \}$ can be saved as the attractor. Denote t_{lc} as the time where the trajectory first reaches the limit cycle. To rule ot numerical errors one may want to verify the attractor by choosing one period of states after the initial occurance of the attractor and compare each and every state, i.e. $\mathbf{x}(t_{lc} + i\Delta t) = \mathbf{x}(t_{lc} + i\Delta t + T)\ \ \forall\, i \in \{1, 2, \ldots, m-1\}$.

(insert example graph for forced 1d pendulum)

Another approach to detecting limit cycles is reforming the problem to be finding a fixed point. For systems with a dominant oscillation of a particular coordinate, this can be done using Poincare Sections. The ides here is to reduce the trajectory by only picking out states at which the oscillating coordinate is at a specific value and it's derivative has the same sign. This essentially removes the oscillations and the fixed point condition can be applied to the reduced trajectory. This approach is well-founded in the theory of dynamical systems, however it is infeasible once there exist oscillations along multiple coordinates. In addition because of the reduction, much longer trajectories are needed, ultimately increasing computational effort, which is why this approach is not recommended.

Note that when comparing two states $\mathbf{x}$ and $\mathbf{y}$ numerically, checking for equality can give misleading results because of rounding errors. It is rather suggested to check $|| \mathbf{x} - \mathbf{y} || < \epsilon$, for some small positive ϵ and some norm $|| \cdot ||$. For the norm, the root mean square error

$$|| \mathbf{x} - \mathbf{y} ||_{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2}, \quad \mathbf{x}, \mathbf{y} \in \mathbb{R}^n \tag{3.14}$$

was used. This approach is reflected in Figures (the two above, that have yet to be created) by error margins.

It is also important to inspect the resulting attractor. As generally there exist multiple attractors it has to be verified that the found attractor does represent the correct undisturbed behaviour. In initial testing a square cloth model (see pic.) made up of mass spring elements was made dynamic by applying oscillations at the top two corners. When computing the limit cycle for this system, the detected attrator seemed to be shifted from its expected location (oscillations about the "hanging down" position). Visual inspection using a graphical interface for dde showed, that for some initial condition, the entire cloth stabilized in the inverted position, which was reflected in the detected attractor. This effect is known from inverted pendulums, where applying oscillations stabilizes the inverted position (REF). This exemplifies the need for verification.

(add figs of cloth in hanging and inverted position)

For certain systems and application, fully determining all coordinates of the attractor might turn out to be difficult or simply excessively precise. These cases are further detaile in the following section. (don't like this last part)

3.2.4. Evaluating Convergence

Given an attractor, evaluation of convergence simply follows definition 3.12. Clearly we cannot be letting time go to infinity, however by the definition 3.7, we can deduce that if a state $\mathbf{x}(t_{conv})$ lies on the attractor, all future states $\mathbf{x}(t_n),\ t_n > t_{conv}$ will as well, fullfilling the definition of convergence. Evaluating convergence therefore reduces to finding a state on the trajectory that coincides with an element of the attractor at any point in time. If undisturbed after t_{conv}, the state should stay on the attractor for all future times.

In many cases, one can and might even want to loosen the requirements for convergence.

sometimes we can loosen the requirements for convergence. For the laikago experiments, where the goal is just for the quadruped to not tip over, we only check height of the core above ground and it's orientation.

Issue that this is one specific state that does not accomodate for any deviation. With laikago standing up, we might accept translations of the robot in the xy-plane or rotations about the

especially in multibody dynamics it might suffice to only track the generalized coordinates of one body, the core for example. This needs to be decided on a case by case basis.

In real world applications, it is often not enough for a system to return to the desired states

Would be enough to check the state if it is similar to an element of the attractor. But as we might still have disturbances being applied, we should check future states as well. For a twist

Any way of checking and guaranteeing for non convergence (divergence) may be simpler. it actually really useful, as when continually apllied at every timestep, the simulation can be stopped if divergence is detected and computational time be saved. Example Laikgao, if we want it to stand upright and we detect it tipping over, that run has clearly failed and can be stopped early. quite similar to

The nonlinear dynamics view is uesful in detecting attractors and evaluating, but sometimes when the needed information can be acquired more easily, we always chose that route.

3.2.5. minRad algorithm

maybe not go into too much detail on how it works (look at ref) No DO go into detail. This is the part where we actually want to explain how we find the minimal radius. Because of the random sampling of disturbances, the convergence of the minRad algorithm and therefore the robustness value is inherently stochastic and might be dominated by noise. The extent of theis noise can be reduced by larger amounts of samples. An compromize between noise and computational time must be found. Number of iterations

tuning

visualizations plot resultion as a function of iterations. NO, write down the formula $resolution = (1/2)^n$ with n being the number of iterations.

3.2.6. boundarys PS DS

elephant in the room changing even the unit changes the scaling and therefore the robustness value.

explain and give examples on how boundaries were chosen.

was noted in REF that it can be an ellipse as well, but they didn't know how to choose it's parameters. We do this by analysis and finding reasonable boundaries in the DS.

order of complexity depending on PS and DS

3.2.7. Multithreading

computational time can be reduced by

pseudocode or block diagram.

3.2.8. Application to specific systems

need to decide on PS and DS plus boundaries. Need to find or define attractor.

analysis needed to find parameter space and disturbance space boundaries and good initial guesses. analysis of high dof motion tricky (bad 3d image), rather plot every coordinate over time (image).

applyParameters

simAndEval

3.2.9. Optimization and Complexity reduction

find out how the order of computational effort when making discretization smaller vs how much one bisection algorithm iteration can do. cmaes given robustness value, any optimizer that does not depend on derivatives can be used. Of course one can also just explore the entire parameter space, howeve that is quite costly. Useful for debugging though (rough estimate if optimizer converges).

complexity reductions should probably be noted where they are implemented (i.e. not in this section)

3.3. Tests

Results of detecting attractors

Results of detecting convergence

Introduction to laikago high dof rather long computational time. how it's implemented (no walking yet) laikago, that it doesn't do anything but try to keep its limbs in the predefined state. optimization of robustness examples

Laikago Droptest Laikago Swingtest note that here we kind of broke the mathematical framework as technically the underlying diff eq were changed. But we just ignored this, NO effects of this need to be investigated further.

Maybe we could do one high precision, high resolution DS swingtest to see if we can spot resonance

This is kind of a twist on the concept as we are starting at a fixed point and continually applying disturbances to see for what disturbances the trajectory conveges, which in this case is expressed by the system staying at the initial state.

4

Conclusion and Outlook

physical explanation why there can’t be that much optimization for swing and droptest

Implication that the system must be dynamical in nature (i.e. it must evolve). So a rudimentary control strategy must be implemented or outside forced must be applied. Don’t quite know where to put this.

further explore effects of combining disturbences of differnent sorts and the effects of the choice of bounds.

applying to systems of high dof with full phase space. Is it even possible? How much can the code be optimized?

apply to physical dimensions of systems (intuitively more drastic effects)

using the phase space as the disturbance space but heavily restricting it (actuator limits, improbable constellations etc)

Optimizing with limit cycles as attractors.

solver (or rather finding the trajectories in general) will always be the largest bottleneck and finding methods to reduce number of trajectories to be evaluated or the computational time per trajectory would be very benefitial

Seems unlikely that this will ever be plug and play. Lots of tuning, lots of fiddling around. But there could definitely be applications, especially in novel systems where rigorous analysis is lacking.

High complexity is still an issue.

A

Information For The Few (Appendix)

dot = f(q,qdot) qddot = g(q,qdot)

order reduction:

x = [q,qdot] => xdot = [qdot, qddot] = F(x) = [f(x), g(x)]

q,qdot,qddot

A.1. Foo Bar Baz

A.2. Barontes

A.3. A Long Table with Booktabs

Table A.1.: *A sample list of words.*

ID	Word	Word Length	WD	ETL	PTL	WDplus
1	Eis	3	4	0.42	1.83	0.19
2	Mai	3	5	0.49	1.92	0.19
3	Art	3	5	0.27	1.67	0.14
4	Uhr	3	5	0.57	1.87	0.36

continued on next page

Table A.1.: *(Continued)*

ID	Word	Word Length	WD	ETL	PTL	WDplus
5	Rat	3	5	0.36	1.71	0.14
6	weit	4	6	0.21	1.65	0.25
7	eins	4	6	0.38	1.79	0.26
8	Wort	4	6	0.30	1.62	0.20
9	Wolf	4	6	0.18	1.54	0.19
10	Wald	4	6	0.31	1.63	0.19
11	Amt	3	6	0.30	1.67	0.14
12	Wahl	4	7	0.36	1.77	0.42
13	Volk	4	7	0.45	1.81	0.20
14	Ziel	4	7	0.48	1.78	0.42
15	vier	4	7	0.38	1.81	0.42
16	Kreis	5	7	0.26	1.62	0.33
17	Preis	5	7	0.28	1.51	0.33
18	Re-de	4	7	0.22	1.56	0.33
19	Saal	4	7	0.75	2.10	0.43
20	voll	4	7	0.48	1.82	0.24
21	weiss	5	7	0.21	1.59	0.36
22	-ger	5	7	1.16	2.69	0.59
23	bald	4	7	0.18	1.56	0.19
24	hier	4	7	0.40	1.70	0.43
25	neun	4	7	0.17	1.52	0.26
26	sehr	4	7	0.36	1.85	0.43
27	Jahr	4	7	0.50	1.82	0.43
28	Gold	4	7	0.04	1.35	0.20
29	Ter	5	8	0.15	1.39	0.59
30	Tei-le	5	8	0.30	1.71	0.46
31	Na-tur	5	8	0.18	1.59	0.41
32	Feu-er	5	8	0.30	1.71	0.45
33	Rol-le	5	8	0.15	1.46	0.45
34	Rock	4	8	0.29	1.68	0.25
35	Spass	5	8	0.28	1.64	0.32
36	Gte	5	8	0.49	1.75	0.66
37	En-de	4	8	0.36	1.72	0.33
38	Kunst	5	8	0.26	1.59	0.35
39	Li-nie	5	8	0.45	1.88	0.63
40	Bme	5	8	0.48	1.92	0.45
41	Bh-ne	5	9	0.94	2.48	0.62
42	Bahn	4	9	0.21	1.62	0.42
43	Br-ger	6	9	0.38	1.70	0.65
44	Druck	5	9	0.60	2.03	0.31
45	zehn	4	9	0.41	1.84	0.42

continued on next page

Table A.1.: *(Continued)*

ID	Word	Word Length	WD	ETL	PTL	WDplus
46	Va-ter	5	9	0.36	1.78	0.40
47	Angst	5	9	0.29	1.56	0.35
48	lei-der	6	9	0.13	1.47	0.52
49	hfig	6	9	0.82	2.31	0.52
50	le-ben	5	9	0.38	1.85	0.40
51	aus-ser	6	9	1.20	2.26	0.57
52	be-vor	5	9	1.28	2.75	0.39
53	Kai-ser	6	9	0.92	2.37	0.53
54	Markt	5	9	0.23	1.58	0.28
55	Os-ten	5	9	0.21	1.54	0.48
56	Krieg	5	9	0.33	1.67	0.50
57	Mann	4	9	0.31	1.47	0.25
58	Hal-le	5	9	0.24	1.65	0.45
59	heu-te	5	9	0.44	1.87	0.46
60	in-nen	5	10	0.36	1.80	0.45
61	Na-men	5	10	0.28	1.72	0.41
62	jetzt	5	10	0.70	2.07	0.32
63	kei-ner	6	10	0.28	1.62	0.53
64	Schu-le	6	10	1.02	2.12	0.48
65	Ar-beit	6	10	0.34	1.70	0.52
66	An-teil	6	10	0.27	1.63	0.53
67	di-rekt	6	10	0.67	2.04	0.47
68	vor-her	6	10	0.78	2.25	0.47
69	wol-len	6	10	0.44	1.85	0.51
70	Kampf	5	10	0.70	1.96	0.27
71	dern	6	10	1.18	2.62	0.65
72	lau-fen	6	10	0.21	1.64	0.52
73	Eu-ro-pa	6	10	0.23	1.53	0.66
74	statt	5	10	1.61	2.86	0.39
75	Wes-ten	6	10	0.29	1.60	0.54