Overview
What are we talking about?

input  → Computation  → output
What are we talking about?

Computation

Mapping
What are we talking about?

Computation

Mapping
What are we talking about?

Computation

Mapping
What are we talking about?

Mapping

Computation
What are we talking about?

[Diagram showing a process labeled 'Computation' with an arrow labeled 'emulation' pointing to another process labeled 'Mapping'.]
**What are we talking about?**

- We want to learn this mapping with machine learning techniques
What are we talking about?

• We want to learn this mapping with machine learning techniques
• Different levels of complexity for the mapping are possible
Simplest Mapping: Look-up table
**Simplest Mapping:**
**Look-up table**

**Pro:** Simple, fast, non-smooth mappings are possible
**Simplest Mapping:**
*Look-up table*

**Pro:** Simple, fast, non-smooth mappings are possible

**Con:** You have to know every single point of mapping (no generalization properties), no memory
Artificial Neural Networks
Artificial Neural Networks

hidden layer

input

output
ARTIFICIAL NEURAL NETWORKS

hidden layer

input

... Input node ...

output
Artificial Neural Networks

Input node

Node: summation of input plus nonlinearity
Artificial Neural Networks

- Hidden layer

Input node
- Node: summation of input plus nonlinearity
- Weight of “importance”

Output
Artificial Neural Networks

Input node
Node: summation of input plus nonlinearity
Weight of "importance"
Output node (linear or nonlinear)
Artificial Neural Networks

hidden layer

input

output
Artificial Neural Networks

- Universal function approximator (Hornik et al. 1989)
Artificial Neural Networks

- Universal function approximator (Hornik et al. 1989)
- Various learning algorithms available (back-propagation)
Artificial Neural Networks

- Universal function approximator (Hornik et al. 1989)
- Various learning algorithms available (back-propagation)
- Guaranteed convergence
**Artificial Neural Networks**

hidden layer

input → hidden layer → output
Artificial Neural Networks

- Feedforward structure
Artificial Neural Networks

- Feedforward structure
- Static (no memory) - mapping point to point
Artificial Neural Networks

- Feedforward structure
- Static (no memory) - mapping point to point
- Nonlinear with generalization property
Artificial Neural Networks

- Feedforward structure
- **Static (no memory)** - mapping point to point
- Nonlinear with generalization property
Memory is important in a lot of cases
Memory is important in a lot of cases.
Memory is important in a lot of cases
Memory is important in a lot of cases.
Memory is important in a lot of cases.
Memory is important in a lot of cases

discrete case

delay line

history

memory is needed
Memory is important in a lot of cases.

- Discrete case
- Delay line
- History

Memory is needed.
Memory is important in a lot of cases.

In the continuous case, history is needed for the output.
Memory is important in a lot of cases
Memory is important in a lot of cases

- Emulate dynamical systems (controller)
Memory is important in a lot of cases

- Emulate dynamical systems (controller)
- Audio/Speech processing
Memory is important in a lot of cases

- Emulate dynamical systems (controller)
- Audio/Speech processing
- Time Series (financial data, weather, etc.)
Memory is important in a lot of cases

- Emulate dynamical systems (controller)
- Audio/Speech processing
- Time Series (financial data, weather, etc.)
- Robot control
Memory is important in a lot of cases

- Emulate dynamical systems (controller)
- Audio/Speech processing
- Time Series (financial data, weather, etc.)
- Robot control
- Understanding interaction (context)
Memory is important in a lot of cases

- Emulate dynamical systems (controller)
- Audio/Speech processing
- Time Series (financial data, weather, etc.)
- Robot control
- Understanding interaction (context)
- many other ...
Recurrent Artificial Neural Networks
Recurrent Artificial Neural Networks
Recurrent Artificial Neural Networks
Recurrent Artificial Neural Networks

- Integration of over time (discrete/continuous)

Feedback
Recurrent Artificial Neural Networks

- Integration of over time (discrete/continuous)
- We get a dynamical system
Recurrent Artificial Neural Networks

- Integration of over time (discrete/continuous)
- We get a dynamical system
- What structure should be used?
Different RNNs

- Elman networks

Different RNNs

- Elman networks
- Hopfield

http://commons.wikimedia.org/wiki/File:Hopfield-net.png
Different RNNs

- Elman networks
- Hopfield
- Anything is possible...

http://edizquierdo.wordpress.com/2008/10/02/bacterial-chemotaxis/
Different RNNs

- Elman networks
- Hopfield
- Anything is possible...
- Reflecting the problem

http://edizquierdo.wordpress.com/2008/10/02/bacterial-chemotaxis/
Learning Rules for RNN

Learning is not trivial anymore!
Learning Rules for RNN

- Learning is not trivial anymore!
- Back Propagation Through Time (BPTT)
LEARNING RULES FOR RNN

Learning is not trivial anymore!

- Back Propagation Through Time (BPTT)
- Real Time Recurrent Learning (RTRL)
Learning Rules for RNN

Learning is not trivial anymore!

- Back Propagation Through Time (BPTT)
- Real Time Recurrent Learning (RTRL)
- Global optimization approach like GA and SA
Learning Rules for RNN

Learning is not trivial anymore!

- Back Propagation Trough Time (BPTT)
- Real Time Recurrent Learning (RTRL)
- Global optimization approach like GA and SA
Learning Rules for RNN

Learning is not trivial anymore!

- Back Propagation Through Time (BPTT)
- Real Time Recurrent Learning (RTRL)
- Global optimization approach like GA and SA

- No guaranteed convergence to global minimum
**Learning Rules for RNN**

Learning is not trivial anymore!

- Back Propagation Through Time (BPTT)
- Real Time Recurrent Learning (RTRL)
- Global optimization approach like GA and SA

⚠️

- No guaranteed convergence to global minimum
- Slow learning
Learning Rules for RNN

- Back Propagation Through Time (BPTT)
- Real Time Recurrent Learning (RTRL)
- Global optimization approach like GA and SA

- No guaranteed convergence to global minimum
- Slow learning
- Can get stuck in local minima
Solution Reservoir Computing Approach
Solution Reservoir Computing Approach

- Randomly initialized recurrent network
Solution Reservoir Computing Approach

- Randomly initialized recurrent network
- Random dynamic parameters and connections
Solution Reservoir Computing Approach

- Randomly initialized recurrent network
- Random dynamic parameters and connections
- Complex, high-dimensional, nonlinear dynamic system
Solution Reservoir Computing Approach

- Randomly initialized recurrent network
- Random dynamic parameters and connections
- Complex, high-dimensional, nonlinear dynamic system
Solution Reservoir Computing Approach

- Randomly initialized recurrent network
- Random dynamic parameters and connections
- Complex, high-dimensional, nonlinear dynamic system

reservoir

fixed during learning
Solution Reservoir Computing Approach
Solution Reservoir Computing Approach

- Input excites the dynamical system
Solution Reservoir Computing Approach

- Input excites the dynamical system
- Get complex, high-dimensional, nonlinear response
Solution Reservoir Computing Approach

- Input excites the dynamical system
- Get complex, high-dimensional, nonlinear response
- Has to be exponentially stable (fading memory)
Solution Reservoir Computing Approach
Solution Reservoir Computing Approach

\[ u(t) = b_0 + \sum_{i=1}^{N} w_i \cdot s_i(t) \]
Solution Reservoir Computing Approach

- Linear combination of the (partial) state of the system
- Weights $w_i$ can be learned with learning regression
- Only the weights $w_i$ are adapted during learning
Solution Reservoir Computing Approach

- Linear combination of the (partial) state of the system
- Weights $w_i$ can be learned with learning regression
- Only the weights $w_i$ are adapted during learning

$$u(t) = b_0 + \sum_{i=1}^{N} w_i \cdot s_i(t)$$
Learning Setup
Learning Setup

one or multiple
Learning Setup
Learning Setup

randomly initialized
Learning Setup

randomly initialized
Learning Setup

- Randomly initialized
- Multitasking

\[ \sum \rightarrow u_1 \]
\[ \sum \rightarrow u_2 \]
\[ \sum \rightarrow u_K \]
Learning Setup

randomly initialized
Learning Setup
Learning Setup

mapping that we want to emulate (Black Box)
Learning Setup

mapping that we want to emulate (Black Box)
Learning Setup

(mapping that we want to emulate (Black Box))

target output
Learning Setup

mapping that we want to emulate (Black Box)

emulation

target output
Learning Setup

supervised learning setup

emulation

mapping that we want to emulate (Black Box)

target output
Learning Setup

supervised learning setup
Learning Setup

supervised learning setup

run system with input
Learning Setup

run system with input

supervised learning setup

collect data

run system with input

s

s

N

s

s

... s

s

s

s

s
Learning Setup

supervised learning setup

collect data

run system with
input

N number of signals

matrix

number of time steps
Learning Setup

supervised learning setup

collect data

run system with input

N number of signals

matrix

number of time steps
Learning Setup

supervised learning setup

[Diagram of neural network with input, hidden layers, and target output]
Learning Setup

supervised learning setup

matrix

target output
Learning Setup

supervised learning setup

matrix

linear regression

optimal weights $w_i$

target output

target output
Learning Setup

supervised learning setup

Matrix

Target output

Linear regression

Optimal weights $w_i$

Minimize the quadratic error
Learning Setup for Feedback

- Supervised learning setup
- Computation we want to emulate (Black Box)
- Target output
Learning Setup for Feedback

supervised learning setup

computation we want to emulate (Black Box)

unstable

target output
Learning Setup for Feedback

collect all data points over time

teacher forcing

target output
Learning Setup for Feedback

- Collect all data points over time
- Noise is crucial for robustness!

Noise diagram with arrows and text boxes.
Learning Setup for Feedback

- Target output
- Optimal weights $w_i$
- Linear regression
- Minimize the quadratic error
- Matrix
- Noise
Learning Setup for Feedback
Learning Setup for Feedback

- close loop, let system run freely
Theories of Reservoir Computing
Theories of Reservoir Computing

feedforward
Theories of Reservoir Computing

feedforward

feedback
Theories of Reservoir Computing

two different theories

feedforward

feedback
Mathematical Formulation

reservoir

∑
Mathematical Formulation

reservoir

\[ \sum \]
Mathematical Formulation

reservoir $\sum$ mathematical operator
Mathematical Formulation

input functions \rightarrow \text{mapping} \rightarrow \text{output function}

reservoir \rightarrow \sum \rightarrow \text{mathematical operator}
Mathematical Formulation

reservoir \[ \sum \] \rightarrow \text{mathematical operator}

input functions \rightarrow \text{mapping} \rightarrow \text{output function}

input streams \rightarrow \text{mathematical operators} \rightarrow \text{output streams}
**Mathematical Formulation**

Encode our computations

Input functions

Input streams

Mathematical operators

Mapping

Output function

Output streams

Reservoir

Encode our computations
First Theoretical Model

• Based on a result by [Boyd and Chua 1985]

First Theoretical Model

- Based on a result by [Boyd and Chua 1985]
- Arbitrary time invariant operators with fading memory

First Theoretical Model

- Based on a result by [Boyd and Chua 1985]
- Arbitrary time invariant operators with fading memory
- Encoding our computational task
First Theoretical Model

- Based on a result by [Boyd and Chua 1985]
- Arbitrary time invariant operators with fading memory
- Encoding our computational task
- Nonlinear, dynamic operator
First Theoretical Model

- Based on a result by [Boyd and Chua 1985]
- Arbitrary time invariant operators with fading memory

- Encoding our computational task
- Nonlinear, dynamic operator

nonlinear with memory
First Theoretical Model

- Based on a result by [Boyd and Chua 1985]
- Arbitrary time invariant operators with fading memory

- Encoding our computational task
- Nonlinear, dynamic operator
- Any exp. stable nonlinear dynamic system with one point of equilibrium
First Theoretical Model

- Based on a result by [Boyd and Chua 1985]
- Arbitrary time invariant operators with fading memory

- Encoding our computational task
- Nonlinear, dynamic operator
- Any exp. stable nonlinear dynamic system with one point of equilibrium

nonlinear controller

nonlinear with memory
First Theoretical Model

- Based on a result by [Boyd and Chua 1985]
- Arbitrary **time invariant** operators with **fading memory**

---

First Theoretical Model

- Based on a result by [Boyd and Chua 1985]
- Arbitrary time invariant operators with fading memory

First Theoretical Model

- Based on a result by [Boyd and Chua 1985]
- Arbitrary **time invariant** operators with **fading memory**

---

First Theoretical Model

- Based on a result by [Boyd and Chua 1985]
- Arbitrary **time invariant** operators with **fading memory** can be uniformly approximated by computational devices, which consist of two simple stages:
First Theoretical Model

- Based on a result by [Boyd and Chua 1985]

- Arbitrary **time invariant** operators with **fading memory** can be uniformly approximated by computational devices, which consist of two simple stages:

  temporal integration

  ![Diagram](image)
First Theoretical Model

- Based on a result by [Boyd and Chua 1985]

- Arbitrary time invariant operators with fading memory can be uniformly approximated by computational devices, which consist of two simple stages:

1. Linear, dynamic
2. Nonlinear, static
First Theoretical Model

- Based on a result by [Boyd and Chua 1985]

- Arbitrary **time invariant** operators with **fading memory** can be uniformly approximated by computational devices, which consist of two simple stages:

![Diagram showing two stages: temporal integration and nonlinear combination.](image-url)
Arbitrary time invariant operators with fading memory can be uniformly approximated by computational devices, which consist of two simple stages:

First Theoretical Model

Based on a result by [Boyd and Chua 1985]

Stage 1

Has to integrate information over time (fading memory)

Has to separate signals

linear, dynamic  nonlinear, static
Arbitrary time invariant operators with fading memory can be uniformly approximated by computational devices, which consist of two simple stages:

Stage 1

Has to \textbf{integrate} information over time (fading memory)

Has to \textbf{separate} signals

\begin{align*}
B_1 & \quad B_2 \\
\ldots & \quad \ldots \\
B_k &
\end{align*}

\begin{align*}
\text{linear, dynamic} & \quad f & \quad \text{nonlinear, static}
\end{align*}
Arbitrary time invariant operators with fading memory can be uniformly approximated by computational devices, which consist of two simple stages:

Stage 1

Has to integrate information over time (fading memory)

Has to separate signals

Based on a result by [Boyd and Chua 1985] temporal integration nonlinear combination linear, dynamic nonlinear, static
Arbitrary time invariant operators with fading memory can be uniformly approximated by computational devices, which consist of two simple stages:

- **Stage 1**
  - Has to **integrate** information over time (fading memory)
  - Has to **separate** signals

**Based on a result by [Boyd and Chua 1985]**

- Theoretical Model
- **Stage 1**
  - Linear, dynamic
  - Nonlinear, static
Arbitrary time invariant operators with fading memory can be uniformly approximated by computational devices, which consist of two simple stages:

First Theoretical Model

Based on a result by [Boyd and Chua 1985], temporal integration is performed in the first stage. This stage involves integrating information over time (fading memory) and separating signals.

Stage 1

Has to integrate information over time (fading memory)

Has to separate signals

\[ B_1 \]

\[ B_2 \]

\[ \ldots \]

\[ B_k \]

linear, dynamic

\[ f \]

nonlinear, static
First Theoretical Model

- Based on a result by [Boyd and Chua 1985]

Arbitrary time invariant operators with fading memory can be uniformly approximated by computational devices, which consist of two simple stages:

**Stage 1**

- Has to **integrate** information over time (fading memory)
- Has to **separate** signals

- Linear, dynamic
- Nonlinear, static
Arbitrary time invariant operators with fading memory can be uniformly approximated by computational devices, which consist of two simple stages:

First Theoretical Model

- Based on a result by [Boyd and Chua 1985]

Stage 1

- Has to **integrate** information over time (fading memory)
- Has to **separate** signals

Stage 2

- Has to **compute** the final result

Linear, dynamic

Nonlinear, static
First Theoretical Model

- Based on a result by [Boyd and Chua 1985]
- Arbitrary time invariant memory can be uniformly approximated by computational devices, which consist of two simple stages:

  **Stage 2**
  Has to combine integrated information nonlinearly

  - Temporal integration
  - Linear, dynamic
  - Nonlinear, static

  \[ B_1, B_2, \ldots, B_k \]
First Theoretical Model

nonlinear, dynamic operator

temporal integration

linear, dynamic

nonlinear combination

B_1

B_2

B_k

f

nonlinear, static
First Theoretical Model

nonlinear, dynamic operator

temporal integration

linear, dynamic

nonlinear, static
First Theoretical Model

- Nonlinear, dynamic operator
- Temporal integration
- Nonlinear combination
- Linear, dynamic
- Nonlinear, static

B<sub>1</sub>, B<sub>2</sub>, ..., B<sub>k</sub>
First Theoretical Model

nonlinear, dynamic operator

emulation

temporal integration

nonlinear combination

linear, dynamic

nonlinear, static
First Theoretical Model

linear dynamic systems
\[ \dot{x} = A(x) + bu \]
\[ y = C(x) \]

nonlinear, dynamic operator

emulation

temporal integration

nonlinear combination

linear, dynamic

nonlinear, static
First Theoretical Model

nonlinear, dynamic operator

linear, dynamic systems
\[ \dot{x} = A(x) + bu \]
\[ y = C(x) \]

temporal integration

emulation

nonlinear combination

linear dynamic systems
\[ \dot{x} = A_1(x) + b_1 u \]
\[ \dot{x} = A_2(x) + b_2 u \]
\[ \vdots \]
\[ \dot{x} = A_k(x) + b_k u \]

nonlinear, static
First Theoretical Model

- **Nonlinear, Dynamic Operator**
- **Temporal Integration**
- **Nonlinear Combination**
- **Emulation**

Mathematical Equations:

\[
\dot{x} = A_1(x) + b_1u \\
\dot{x} = A_2(x) + b_2u \\
\vdots \\
\dot{x} = A_k(x) + b_ku
\]

**Linear, Dynamic**

**Nonlinear, Static**
First Theoretical Model

Nonlinear, dynamic operator

Temporal integration

Nonlinear combination

Emulation

Linear, dynamic

Nonlinear, static

\[ \dot{x} = A_1(x) + b_1 u \]

\[ \dot{x} = A_2(x) + b_2 u \]

\[ \ldots \]

\[ \dot{x} = A_k(x) + b_k u \]

ANN
NOT A RESERVOIR SETUP YET

temporal integration

\[ \dot{x} = A_1(x) + b_1 u \]
\[ \dot{x} = A_2(x) + b_2 u \]
\[ \dot{x} = A_k(x) + b_k u \]

nonlinear combination

ANN
NOT A RESERVOIR SETUP YET

Nonlinear, static mapping could also be done by "kernel"
Nonlinear, static mapping could also be done by "kernel"

Idea: complex, nonlinear high-dimensional dynamic system could be seen as an implementation of such a finite kernel
• Nonlinear, static mapping could also be done by "kernel"
• Idea: complex, nonlinear high-dimensional dynamic system could be seen as an implementation of such a finite kernel
• Combining both stages - we get a reservoir
Reservoir Setup

- Nonlinear, static mapping could also be done by "kernel"
- Idea: complex, nonlinear high-dimensional dynamic system could be seen as an implementation of such a finite kernel
- Combining both stages - we get a reservoir
Reservoir Setup

temporal integration + nonlinear combination

Σ
Reservoir Setup

temporal integration \quad + \quad \text{nonlinear combination}

nonlinearities, memory are part of the reservoir
Reservoir Setup

nonlinearities, memory are part of the reservoir

simple, static, linear readout - linear regression
Reservoir Setup

- Nonlinear, dynamic operator

- Nonlinearities, memory are part of the reservoir

- Simple, static, linear readout - linear regression
Reservoir Setup

- Nonlinear, dynamic operator
- Nonlinearities, memory are part of the reservoir
- Simple, static, linear readout - linear regression
**Desired Properties**

- temporal integration
- nonlinear combination

Diagram: Network with arrows representing connections and a sum symbol (Σ) indicating the summation process.
Desired Properties

- temporal integration
- nonlinear combination

+ kernel property
The reservoir should be:

**Desired Properties**

- temporal integration
- nonlinear combination

kernel property
Desired Properties

The reservoir should be:

- Nonlinear
Desired Properties

The reservoir should be:

- Nonlinear
- High-dimensional
The reservoir should be:

- Nonlinear
- High-dimensional
- Dynamic (fading memory)
Desired Properties

The reservoir should be:

- Nonlinear
- High-dimensional
- Dynamic (fading memory)

...but otherwise completely generic
Desired Properties

The reservoir should be:

- Nonlinear
- High-dimensional
- Dynamic (fading memory)

...but otherwise completely generic
different flavors of RC
Different Flavors of RC

- temporal integration + nonlinear combination

- Neuron (model)
- Synaptic connection
- Spike trains
- Connection based on biological data
Different Flavors of RC

Temporal integration + nonlinear combination

Neuron (model)
Synaptic connection
Spike trains
Connection based on biological data

Liquid State Machine
Maass et al. 2002
Different Flavors of RC

temporal integration + nonlinear combination

Simple diff. equation (nonlinearity)
weighted connection
Full connectivity
Pure machine learning technique
Different Flavors of RC

- Simple diff. equation (nonlinearity)
- Weighted connection
- Full connectivity
- Pure machine learning technique

Echo State Network
Jaeger 2002
Different Flavors of RC

- Temporal integration
- Nonlinear combination

bucket of water

\[ \sum \]
Different Flavors of RC

- Water is mechanically perturbed (with motors)
**Different Flavors of RC**

- Water is mechanically perturbed (with motors)
- Complex response of the surface
Different Flavors of RC

- Water is mechanically perturbed (with motors)
- Complex response of the surface
- Readout is digitized picture frame + processing (vision)

Fernando and Sojakka, 2003
Different Flavors of RC

temporal integration + nonlinear combination

Nonlinear optical effects of laser

\[ \sum \]
Different Flavors of RC

- Nonlinear optical effects are exploited

\[
\text{temporal integration} + \text{nonlinear combination}
\]

Nonlinear optical effects of laser
Different Flavors of RC

- Nonlinear optical effects are exploited
- Calculating at the speed of light
**Different Flavors of RC**

- Nonlinear optical effects are exploited
- Calculating at the speed of light
- Generic computational device
Different Flavors of RC

- Nonlinear optical effects of laser
- Temporal integration + nonlinear combination
- Photonic RC

- Nonlinear optical effects are exploited
- Calculating at the speed of light
- Generic computational device

Massar, Dambri, Schrauwen
Different Flavors of RC

- e.g., physical body of a robot
- temporal integration + nonlinear combination

Diagram: A network of interconnected nodes with arrows indicating flow, leading to a summation symbol (Σ).
Different Flavors of RC

e.g., physical body of a robot

temporal integration + nonlinear combination

masses (scaling)

nonlinear springs

only local connectivity

Morphological computation setup
Different Flavors of RC

- Physical reservoir computing
- Nonlinear combination
- Temporal integration

- E.g., physical body of a robot

- Masses (scaling)
- Nonlinear springs
- Only local connectivity
- Morphological computation setup

Hauser et al. 2012
Different Flavors of RC

e.g., physical body of a robot

temporal integration + nonlinear combination

masses (scaling)
nonlinear springs
only local connectivity

Morphological computation setup

Hauser et al. 2012
Remarkable Conclusion

nonlinear, dynamic operator

temporal integration  nonlinear combination

Σ
Remarkable Conclusion

nonlinear, dynamic operator

temporal integration

nonlinear combination

Outsourcing big part of the computation to the morphology
Remarkable Conclusion

nonlinear, dynamic operator

Outsourcing big part of the computation to the morphology

resulting task is easier: linear regression
Remarkable Conclusion

- Nonlinear, dynamic operator
- Temporal integration
- Nonlinear combination

Outsourcing big part of the computation to the morphology

Resulting task is easier: linear regression

Concept of Morphological Computation
Remarkable Conclusion

Concept of Morphological Computation
Remarkable Conclusion

Concept of Morphological Computation
Remarkable Conclusion

Concept of Morphological Computation

randomly initialized!

temporal integration

nonlinear combination

high-dimensional
Remarkable Conclusion

randomly initialized!

temporal integration

nonlinear combination

high-dimensional nonlinear

Concept of Morphological Computation
Remarkable Conclusion

Concept of Morphological Computation

randomly initialized!

high-dimensional nonlinear compliant

temporal integration nonlinear combination

...
Remarkable Conclusion

- high-dimensional
- nonlinear
- compliant

Concept of Morphological Computation
Remarkable Conclusion

- high-dimensional
- nonlinear
- compliant

I don’t like that!

Concept of Morphological Computation
Remarkable Conclusion

- high-dimensional
- nonlinear
- compliant

I don’t like that!

Actually, I do like that!

Concept of Morphological Computation
Limitation

temporal integration + nonlinear combination

∑
The theoretical model is limited to time-invariant operators with fading memory.
The theoretical model is limited to time-invariant operators with fading memory. Persistent memory is of interest too, or limit cycles.
The theoretical model is limited to **time-invariant operators with fading memory**

- Persistent memory is of interest too, or limit cycles
- **Another theory is needed!**
Limitation

- The theoretical model is limited to **time-invariant operators with fading memory**
- Persistent memory is of interest too, or limit cycles
- **Another theory is needed!**
- based on feedback linearization from control theory
Second Theoretical Model
Second Theoretical Model

- Based on the concept of “feedback linearization” from control theory
- Maass et al. 2007 applied to generic neural networks, Hauser et al. to physical bodies
Second Theoretical Model

- Based on the concept of “feedback linearization” from control theory
- Maass et al. 2007 applied to generic neural networks, Hauser et al. to physical bodies

Fixed dynamical system
- feedback linearizable
- e.g., nonlinear mass-spring system
Second Theoretical Model

- Based on the concept of “feedback linearization” from control theory
- Maass et al. 2007 applied to generic neural networks, Hauser et al. to physical bodies

\[ x'(t) = f(x(t)) + g(x(t)) \cdot v(t) \]

\[ z^{(n)}(t) = G(z(t), z'(t), \ldots, z^{(n-1)}(t)) + u(t) \]

- Fixed dynamical system
- Feedback linearizable
- E.g., nonlinear mass-spring system

- Desired computation encoded as a nonlinear dynamical system
**Second Theoretical Model**

- Based on the concept of “feedback linearization” from control theory
- Maass et al. 2007 applied to generic neural networks, Hauser et al. to physical bodies

\[ x'(t) = f(x(t)) + g(x(t)) \cdot v(t) \]

\[ z(t)^{(n)} = G(z(t), z(t)', \ldots, z(t)^{(n-1)}) + u(t) \]

- Fixed dynamical system
- Feedback linearizable
- E.g., nonlinear mass-spring system
**Second Theoretical Model**

- Based on the concept of “feedback linearization” from control theory
- Maass et al. 2007 applied to generic neural networks, Hauser et al. to physical bodies

\[ \dot{x}(t) = f(x(t)) + g(x(t)) \cdot v(t) \]

\[ z(t)^{(n)} = G(z(t), z(t)', \ldots, z(t)^{(n-1)}) + u(t) \]

- Fixed dynamical system
- Feedback linearizable
- E.g., nonlinear mass-spring system

Desired computation: very powerful description!
Second Theoretical Model

\[ x'(t) = f(x(t)) + g(x(t)) \cdot v(t) \]

\[ z(t)^{(n)} = G(z(t), z(t)', \ldots, z(t)^{(n-1)}) + u(t) \]
Second Theoretical Model

\[ x'(t) = f(x(t)) + g(x(t)) \cdot v(t) \]

\[ z(t) = G(z(t), z(t)', \ldots, z(t)^{(n-1)}) + u(t) \]

- Static functions \( h \) and \( K \) are static and nonlinear → define behaviour
Second Theoretical Model

\[ x'(t) = f(x(t)) + g(x(t)) \cdot v(t) \]

\[ z(t) = G(z(t), z(t)', \ldots, z(t)^{(n-1)}) + u(t) \]

- Static functions \( h \) and \( K \) are static and nonlinear → define behaviour
- For a given \( G \) and known \( f \) and \( b \) → \( h \) and \( K \) can be calculated
Second Theoretical Model

\[ x'(t) = f(x(t)) + g(x(t)) \cdot v(t) \]

- Static functions \( h \) and \( K \) are static and nonlinear \( \rightarrow \) define behaviour
- For a given \( G \) and known \( f \) and \( b \) \( \rightarrow \) \( h \) and \( K \) can be calculated
- Idea: nonlinearities could be “provided” by a reservoir

\[ z(t)^{(n)} = G(z(t), z(t)', \ldots, z(t)^{(n-1)}) + u(t) \]
Second Theoretical Model

\[ x'(t) = f(x(t)) + g(x(t)) \cdot v(t) \]

\[ z(t) = G(z(t), z'(t), \ldots, z^{(n-1)}(t)) + u(t) \]

- Static functions \( h \) and \( K \) are static and nonlinear \( \rightarrow \) define behaviour
- For a given \( G \) and known \( f \) and \( b \) \( \rightarrow \) \( h \) and \( K \) can be calculated
- Idea: nonlinearities could be ”provided” by a reservoir
- Therefore, linear feedback and linear readouts might be sufficient
**Second Theoretical Model**

- Static functions $h$ and $K$ are static and nonlinear $\rightarrow$ define behaviour
- For a given $G$ and known $f$ and $b \rightarrow h$ and $K$ can be calculated
- Idea: nonlinearities could be "provided" by a reservoir
- Therefore, linear feedback and linear readouts might be sufficient
- Feedback could be even fixed and directly feed the output back
Second Theoretical Model

\[ u(t) \rightarrow \sum \rightarrow z(t) \]
Second Theoretical Model

\[ u(t) \]

\[ z(t) = G(z(t), z(t)', \ldots, z(t)^{(n-1)}) + u(t) \]

\[ z(t) \]
Second Theoretical Model

\[ u(t) \] encodes our computation

\[ z(t)^{(n)} = G(z(t), z(t)', \ldots, z(t)^{(n-1)}) + u(t) \]
Second Theoretical Model

\[ z(t) = G(z(t), z(t)', ..., z(t)^{(n-1)}) + u(t) \]

encodes our computation

emulation
Second Theoretical Model

\[ u(t) \]

\[ z(t) = G(z(t), z(t)', ..., z(t)^{(n-1)}) + u(t) \]
encodes our computation

\[ \sum \]

emulation

\[ z(t) \]
Applications for RC
Applications for RC

- Chaotic time series prediction (Jager and Haass 2004)
Applications for RC

- Chaotic time series prediction (Jager and Haass 2004)
- Speech recognition (Verstraeten et al. 2005 and 2006, Jaeger et al. 2007)
Applications for RC

- Chaotic time series prediction (Jager and Haass 2004)
- Speech recognition (Verstraeten et al. 2005 and 2006, Jaeger et al. 2007)
- International financial time series forecasting
Applications for RC

- Chaotic time series prediction (Jager and Haass 2004)
- Speech recognition (Verstraeten et al. 2005 and 2006, Jaeger et al. 2007)
- International financial time series forecasting
- Nonlinear audio processing (Holzmann and Hauser 2010)
Applications for RC

- Chaotic time series prediction (Jager and Haass 2004)
- Speech recognition (Verstraeten et al. 2005 and 2006, Jaeger et al. 2007)
- International financial time series forecasting
- Nonlinear audio processing (Holzmann and Hauser 2010)
- Epileptic seizure detection
Applications for RC

- Chaotic time series prediction (Jager and Haass 2004)
- Speech recognition (Verstraeten et al. 2005 and 2006, Jaeger et al. 2007)
- International financial time series forecasting
- Nonlinear audio processing (Holzmann and Hauser 2010)
- Epileptic seizure detection
- Robot localization (Antonelo and Schrauwen 2011)
Applications for RC

- Chaotic time series prediction (Jager and Haass 2004)
- Speech recognition (Verstraeten et al. 2005 and 2006, Jaeger et al. 2007)
- International financial time series forecasting
- Nonlinear audio processing (Holzmann and Hauser 2010)
- Epileptic seizure detection
- Robot localization (Antonelo and Schrauwen 2011)
- Processing Grammatical Structure (ORGANIC project)
Applications for RC

- Chaotic time series prediction (Jager and Haass 2004)
- Speech recognition (Verstraeten et al. 2005 and 2006, Jaeger et al. 2007)
- International financial time series forecasting
- Nonlinear audio processing (Holzmann and Hauser 2010)
- Epileptic seizure detection
- Robot localization (Antonelo and Schrauwen 2011)
- Processing Grammatical Structure (ORGANIC project)
- Morphological Computation (robot bodies as a reservoir)
The body as a Reservoir
The body as a Reservoir

Limit cycle for locomotion
Results

Generating Stable Nonlinear Limit Cycles
Results

Generating Stable Nonlinear Limit Cycles

\[ x'_1 = x_1 + x_2 - x_1(x_1^2 + x_2^2) \]
\[ x'_2 = -2x_1 + x_2 - x_2(x_1^2 + x_2^2) \]
Results

Generating Stable Nonlinear Limit Cycles

\[ x'_1 = x_1 + x_2 - x_1(x_1^2 + x_2^2) \]
\[ x'_2 = -2x_1 + x_2 - x_2(x_1^2 + x_2^2) \]
Generating Stable Nonlinear Limit Cycles

\[ x_1' = x_1 + x_2 - x_1(x_1^2 + x_2^2) \]
\[ x_2' = -2x_1 + x_2 - x_2(x_1^2 + x_2^2) \]
Results

Generating Stable Nonlinear Limit Cycles

\[ x'_1 = x_1 + x_2 - x_1(x_1^2 + x_2^2) \]
\[ x'_2 = -2x_1 + x_2 - x_2(x_1^2 + x_2^2) \]
Results

Generating Stable Nonlinear Limit Cycles

\[ x_1' = x_1 + x_2 - x_1(x_1^2 + x_2^2) \]
\[ x_2' = -2x_1 + x_2 - x_2(x_1^2 + x_2^2) \]
Generating Stable Nonlinear Limit Cycles

\[ x'_1 = x_1 + x_2 - x_1 (x_1^2 + x_2^2) \]
\[ x'_2 = -2x_1 + x_2 - x_2 (x_1^2 + x_2^2) \]

#masses = 10
#springs = 22
Generating Stable Nonlinear Limit Cycles

\[ x_1' = x_1 + x_2 - x_1 (x_1^2 + x_2^2) \]
\[ x_2' = -2x_1 + x_2 - x_2 (x_1^2 + x_2^2) \]

#masses = 10
#springs = 22
Generating Stable Nonlinear Limit Cycles

\[ x'_1 = x_1 + x_2 - x_1(x_1^2 + x_2^2) \]

\[ x'_2 = -2x_1 + x_2 - x_2(x_1^2 + x_2^2) \]

#masses = 10
#springs = 22
Generating Stable Nonlinear Limit Cycles

\[ x_1' = x_1 + x_2 - x_1(x_1^2 + x_2^2) \]
\[ x_2' = -2x_1 + x_2 - x_2(x_1^2 + x_2^2) \]

#masses = 10
#springs = 22
Results

Testing Robustness of Learned Nonlinear Limit Cycles
Results

Testing Robustness of Learned Nonlinear Limit Cycles

Applied constant disturbance forces at random points
Results

Testing Robustness of Learned Nonlinear Limit Cycles

Applied constant disturbance forces at random points
Results

Testing Robustness of Learned Nonlinear Limit Cycles

Applied constant disturbance forces at random points
Testing Robustness of Learned Nonlinear Limit Cycles

Applied constant disturbance forces at random points
Results

Input Dependent Limit Cycle

\[ x'_1 = x_1 + x_2 - \varepsilon x_1 (x_1^2 + x_2^2) \]
\[ x'_2 = -2x_1 + x_2 - x_2 (x_1^2 + x_2^2) \]
Input Dependent Limit Cycle

\[
x'_1 = x_1 + x_2 - \varepsilon x_1 (x_1^2 + x_2^2)
\]
\[
x'_2 = -2x_1 + x_2 - x_2(x_1^2 + x_2^2)
\]

![Diagram showing limit cycle for different values of \(\varepsilon\).](image-url)
Results

Input Dependent Limit Cycle

\[ x'_1 = x_1 + x_2 - \varepsilon x_1 (x_1^2 + x_2^2) \]
\[ x'_2 = -2x_1 + x_2 - x_2 (x_1^2 + x_2^2) \]
**Results**

Input Dependent Limit Cycle

\[
x'_1 = x_1 + x_2 - \varepsilon x_1 (x_1^2 + x_2^2)
\]
\[
x'_2 = -2x_1 + x_2 - x_2(x_1^2 + x_2^2)
\]
Results

Input Dependent Limit Cycle

\[ x'_1 = x_1 + x_2 - \varepsilon x_1 (x_1^2 + x_2^2) \]
\[ x'_2 = -2x_1 + x_2 - x_2 (x_1^2 + x_2^2) \]
Results

Input Dependent Limit Cycle

\[ x_1' = x_1 + x_2 - \varepsilon x_1 (x_1^2 + x_2^2) \]
\[ x_2' = -2x_1 + x_2 - x_2 (x_1^2 + x_2^2) \]
Input Dependent Limit Cycle

\[ x'_1 = x_1 + x_2 - \varepsilon x_1(x_1^2 + x_2^2) \]
\[ x'_2 = -2x_1 + x_2 - x_2(x_1^2 + x_2^2) \]
Input Dependent Limit Cycle

\[ x'_1 = x_1 + x_2 - \varepsilon x_1 (x_1^2 + x_2^2) \]

\[ + x_2 - x_2 (x_1^2 + x_2^2) \]

one set of weights for all three \( \varepsilon \)
Input Dependent Limit Cycle

\[
\begin{align*}
x_1 &= 0.2 \\
\epsilon &= 5
\end{align*}
\]
Results

Input Dependent Limit Cycle
Results

Input Dependent Limit Cycle

- $x_1$
- $x_2$

- $\varepsilon = 0.2$

- $\varepsilon = 1$
- $\varepsilon = 5$

- from 0 to 20 s
- from 20 to 40 s
- from 40 to 60 s
- from 60 to 80 s
- from 80 to 100 s
Discussion
Discussion

\[ \sum \]

constant
Discussion

squeezing

constant
Discussion

squeezing

constant
Discussion

squeezing

constant
Discussion

squeezing

constant
Discussion

System is able to sense through its morphology!
Discussion

System is able to sense through its morphology!
System is able to sense through its morphology!
Discussion

System is able to sense through its morphology!
Application in Soft Robotics
Application in Soft Robotics

Kohei Nakajima

Tao Li
Application in Soft Robotics

soft (passive) silicone structure

Kohei Nakajima
Tao Li

soft (passive) silicone structure
Application in Soft Robotics

water tank

soft (passive) silicone structure

Kohei Nakajima
Tao Li
Application in Soft Robotics

10 bending sensors

water tank

soft (passive) silicone structure

Kohei Nakajima
Tao Li
Application in Soft Robotics

- rotational motor
- water tank
- soft (passive) silicone structure
- 10 bending sensors

Kohei Nakajima
Tao Li
Application in Soft Robotics
Application in Soft Robotics
Application in Soft Robotics
Application in Soft Robotics

motor signal

\[ \sum \]
Application in Soft Robotics

- exploitation of the body

motor signal
Application in Soft Robotics

- exploitation of the body
- nonlinear and memory

motor signal
**Application in Soft Robotics**

- exploitation of the body
- nonlinear and memory
- noise comes from the sensors
Application in Soft Robotics

- exploitation of the body
- nonlinear and memory
- noise comes from the sensors
- sensors, water and even motor is part of the reservoir
Application in Soft Robotics

- exploitation of the body
- nonlinear and memory
- noise comes from the sensors
- sensors, water and even motor is part of the reservoir
- able to produce robust limit cycle
APPLICATION IN SOFT ROBOTICS
Summary
Summary
Supervised Machine Learning technique
Reservoir: Randomly initialized complex, nonlinear, dynamic system
Summary

Reservoir: Randomly initialized complex, nonlinear, dynamic system

output weights learned with linear regression

Supervised Machine Learning technique
Thank you very much for your Attention!