Nonlinear Systems Toolbox

Arthur J Krener
Naval Postgraduate School
Monterey, CA 93943

ajkrener@nps.edu

Modern Control Theory

Modern Control Theory dates back to first International Federation for Automatic Control (IFAC) World Congress that took place in Moscow in 1960.

Modern Control Theory

Modern Control Theory dates back to first International Federation for Automatic Control (IFAC) World Congress that took place in Moscow in 1960.

At least three major theoretical accomplishments were reported there.

- **Pontryagin, Bolshanski, Gamkerlidze and Mischenko presented the Maximum Principle.**

Modern Control Theory

Modern Control Theory dates back to first International Federation for Automatic Control (IFAC) World Congress that took place in Moscow in 1960.

At least three major theoretical accomplishments were reported there.

- **Pontryagin, Bolshanski, Gamkerlidze and Mischenko presented the Maximum Principle.**
- **Bellman presented the Dynamic Programming Method and the Hamilton-Jacobi-Bellman (HJB) PDE.**

Modern Control Theory

Modern Control Theory dates back to first International Federation for Automatic Control (IFAC) World Congress that took place in Moscow in 1960.

At least three major theoretical accomplishments were reported there.

- **Pontryagin, Bolshanski, Gamkerlidze and Mischenko presented the Maximum Principle.**
- **Bellman presented the Dynamic Programming Method and the Hamilton-Jacobi-Bellman (HJB) PDE.**
- **Kalman presented the theory of Controllability, Observability and Minimality for Linear Control Systems.**

Modern Control Theory

Modern Control Theory dates back to first International Federation for Automatic Control (IFAC) World Congress that took place in Moscow in 1960.

At least three major theoretical accomplishments were reported there.

- **Pontryagin, Bolshanski, Gamkerlidze and Mischenko presented the Maximum Principle.**
- **Bellman presented the Dynamic Programming Method and the Hamilton-Jacobi-Bellman (HJB) PDE.**
- **Kalman presented the theory of Controllability, Observability and Minimality for Linear Control Systems.**

Modern Control Theory

Modern Control Theory dates back to first International Federation for Automatic Control (IFAC) World Congress that took place in Moscow in 1960.

At least three major theoretical accomplishments were reported there.

- **Pontryagin, Bolshanski, Gamkerlidze and Mischenko presented the Maximum Principle.**
- **Bellman presented the Dynamic Programming Method and the Hamilton-Jacobi-Bellman (HJB) PDE.**
- **Kalman presented the theory of Controllability, Observability and Minimality for Linear Control Systems.**

What did these accomplishments have in common.

Modern Control Theory

Modern Control Theory dates back to first International Federation for Automatic Control (IFAC) World Congress that took place in Moscow in 1960.

At least three major theoretical accomplishments were reported there.

- **Pontryagin, Bolshanski, Gamkerlidze and Mischenko presented the Maximum Principle.**
- **Bellman presented the Dynamic Programming Method and the Hamilton-Jacobi-Bellman (HJB) PDE.**
- **Kalman presented the theory of Controllability, Observability and Minimality for Linear Control Systems.**

What did these accomplishments have in common.

They dealt with control and estimation problems in

State Space Form.

Linear State Space Control Theory

The decade of the 1960 witnessed the explosion of linear state space control theory.

Linear State Space Control Theory

The decade of the 1960 witnessed the explosion of linear state space control theory.

- **Linear Quadratic Regulator (LQR)**

Linear State Space Control Theory

The decade of the 1960 witnessed the explosion of linear state space control theory.

- **Linear Quadratic Regulator (LQR)**
- **Kalman Filtering**

Linear State Space Control Theory

The decade of the 1960 witnessed the explosion of linear state space control theory.

- **Linear Quadratic Regulator (LQR)**
- **Kalman Filtering**
- **Separation Principle, Linear Quadratic Gaussian (LQG)**

Linear State Space Control Theory

The decade of the 1960 witnessed the explosion of linear state space control theory.

- **Linear Quadratic Regulator (LQR)**
- **Kalman Filtering**
- **Separation Principle, Linear Quadratic Gaussian (LQG)**
- **Many other linear results.**

Linear State Space Control Theory

The decade of the 1960 witnessed the explosion of linear state space control theory.

- **Linear Quadratic Regulator (LQR)**
- **Kalman Filtering**
- **Separation Principle, Linear Quadratic Gaussian (LQG)**
- **Many other linear results.**

Linear State Space Control Theory

The decade of the 1960 witnessed the explosion of linear state space control theory.

- **Linear Quadratic Regulator (LQR)**
- **Kalman Filtering**
- **Separation Principle, Linear Quadratic Gaussian (LQG)**
- **Many other linear results.**

It was the time of F, G, H, J if you lived near Stanford and A, B, C, D if you lived anywhere else.

Linear State Space Control Theory

The decade of the 1960 witnessed the explosion of linear state space control theory.

- **Linear Quadratic Regulator (LQR)**
- **Kalman Filtering**
- **Separation Principle, Linear Quadratic Gaussian (LQG)**
- **Many other linear results.**

It was the time of F, G, H, J if you lived near Stanford and A, B, C, D if you lived anywhere else.

And there was much lamenting the gap between the linear state space theory and the linear frequency domain practice.

LINPACK

In the early 1970s something happened to change the conversation.

LINPACK

In the early 1970s something happened to change the conversation.

The Department of Energy funded a project to develop software for linear algebra applications for what were then called "supercomputers".

LINPACK

In the early 1970s something happened to change the conversation.

The Department of Energy funded a project to develop software for linear algebra applications for what were then called "supercomputers".

Four numerical analysts, Dongarra, Bunch, Moler and Stewart wrote LINPACK in Fortran. LINPACK makes use of the BLAS (Basic Linear Algebra Subprograms) libraries for performing basic vector and matrix operations.

LINPACK

In the early 1970s something happened to change the conversation.

The Department of Energy funded a project to develop software for linear algebra applications for what were then called "supercomputers".

Four numerical analysts, Dongarra, Bunch, Moler and Stewart wrote LINPACK in Fortran. LINPACK makes use of the BLAS (Basic Linear Algebra Subprograms) libraries for performing basic vector and matrix operations.

In the later 1970s LINPAK and a related package called EISPAK were replaced and supplanted by LAPACK which also uses BLAS.

MATLAB

Cleve Moler, the chairman of the computer science department at the University of New Mexico, started developing MATLAB in the late 1970s.

MATLAB

Cleve Moler, the chairman of the computer science department at the University of New Mexico, started developing MATLAB in the late 1970s.

He designed it to give his students access to LINPACK and EISPACK without them having to learn Fortran.

MATLAB

Cleve Moler, the chairman of the computer science department at the University of New Mexico, started developing MATLAB in the late 1970s.

He designed it to give his students access to LINPACK and EISPACK without them having to learn Fortran.

The roots of MATLAB are in pedagogy

MATLAB

Cleve Moler, the chairman of the computer science department at the University of New Mexico, started developing MATLAB in the late 1970s.

He designed it to give his students access to LINPACK and EISPACK without them having to learn Fortran.

The roots of MATLAB are in pedagogy

Jack Little, a Stanford engineering graduate student, was exposed to it during a visit Moler made in 1983.

MATLAB

Cleve Moler, the chairman of the computer science department at the University of New Mexico, started developing MATLAB in the late 1970s.

He designed it to give his students access to LINPACK and EISPACK without them having to learn Fortran.

The roots of MATLAB are in pedagogy

Jack Little, a Stanford engineering graduate student, was exposed to it during a visit Moler made in 1983.

Recognizing its commercial potential, he joined with Moler and Steve Bangert. They rewrote MATLAB in C and founded MathWorks in 1984 to continue its development.

MATLAB

Cleve Moler, the chairman of the computer science department at the University of New Mexico, started developing MATLAB in the late 1970s.

He designed it to give his students access to LINPACK and EISPACK without them having to learn Fortran.

The roots of MATLAB are in pedagogy

Jack Little, a Stanford engineering graduate student, was exposed to it during a visit Moler made in 1983.

Recognizing its commercial potential, he joined with Moler and Steve Bangert. They rewrote MATLAB in C and founded MathWorks in 1984 to continue its development.

MATLAB was first adopted by researchers and practitioners in control engineering, Little's specialty, but quickly spread to many other domains.

Competitors

With BLAS readily available, several other companies arose to compete with MathWorks, among them Control-C and Matrix-X. But they gradually faded away.

Competitors

With BLAS readily available, several other companies arose to compete with MathWorks, among them Control-C and Matrix-X. But they gradually faded away.

There continues to arise new competitors to Matlab like Scilab but with Matlab's estabished base and extensive suite of toolboxes it is hard for them to get market share.

Competitors

With BLAS readily available, several other companies arose to compete with MathWorks, among them Control-C and Matrix-X. But they gradually faded away.

There continues to arise new competitors to Matlab like Scilab but with Matlab's estabished base and extensive suite of toolboxes it is hard for them to get market share.

Alan Laub and others wrote the Control Systems Toolbox and Leonard Ljung wrote the System Identification Toolbox. Both deal primarily with linear systems.

Competitors

With BLAS readily available, several other companies arose to compete with MathWorks, among them Control-C and Matrix-X. But they gradually faded away.

There continues to arise new competitors to Matlab like Scilab but with Matlab's estabished base and extensive suite of toolboxes it is hard for them to get market share.

Alan Laub and others wrote the Control Systems Toolbox and Leonard Ljung wrote the System Identification Toolbox. Both deal primarily with linear systems.

Other Controls related toolboxes include the Aerospace Toolbox, the Model Predictive Control Toolbox, the Robotics Toolbox and the Robust Control Toolbox.

Competitors

With BLAS readily available, several other companies arose to compete with MathWorks, among them Control-C and Matrix-X. But they gradually faded away.

There continues to arise new competitors to Matlab like Scilab but with Matlab's estabished base and extensive suite of toolboxes it is hard for them to get market share.

Alan Laub and others wrote the Control Systems Toolbox and Leonard Ljung wrote the System Identification Toolbox. Both deal primarily with linear systems.

Other Controls related toolboxes include the Aerospace Toolbox, the Model Predictive Control Toolbox, the Robotics Toolbox and the Robust Control Toolbox.

So special purpose software is becoming available for some nonlinear systems or perhaps more precisely the Jacobi linearizations of nonlinear system, e.g. the Extended Kalman Filter.

Fundamental Problems, Mathematician's Opinion

What are the fundamental problems of control?

Fundamental Problems, Mathematician's Opinion

What are the fundamental problems of control?

- **Finding a feedback that stabilizes a plant to an operating point.**

Fundamental Problems, Mathematician's Opinion

What are the fundamental problems of control?

- **Finding a feedback that stabilizes a plant to an operating point.**
- **Finding an open loop control trajectory that steers a plant from one state to another.**

Fundamental Problems, Mathematician's Opinion

What are the fundamental problems of control?

- **Finding a feedback that stabilizes a plant to an operating point.**
- **Finding an open loop control trajectory that steers a plant from one state to another.**
- **Finding a feedforward and feedback that tracks a reference trajectory.**

Fundamental Problems, Mathematician's Opinion

What are the fundamental problems of control?

- **Finding a feedback that stabilizes a plant to an operating point.**
- **Finding an open loop control trajectory that steers a plant from one state to another.**
- **Finding a feedforward and feedback that tracks a reference trajectory.**
- **Estimating the state of a system from partial and inexact measurements.**

Fundamental Problems, Mathematician's Opinion

What are the fundamental problems of control?

- **Finding a feedback that stabilizes a plant to an operating point.**
- **Finding an open loop control trajectory that steers a plant from one state to another.**
- **Finding a feedforward and feedback that tracks a reference trajectory.**
- **Estimating the state of a system from partial and inexact measurements.**
- **Identifying a model of a plant from first principles and/or input-output measurments.**

Fundamental Problems, Mathematician's Opinion

What are the fundamental problems of control?

- **Finding a feedback that stabilizes a plant to an operating point.**
- **Finding an open loop control trajectory that steers a plant from one state to another.**
- **Finding a feedforward and feedback that tracks a reference trajectory.**
- **Estimating the state of a system from partial and inexact measurements.**
- **Identifying a model of a plant from first principles and/or input-output measurments.**

Fundamental Problems, Mathematician's Opinion

What are the fundamental problems of control?

- **Finding a feedback that stabilizes a plant to an operating point.**
- **Finding an open loop control trajectory that steers a plant from one state to another.**
- **Finding a feedforward and feedback that tracks a reference trajectory.**
- **Estimating the state of a system from partial and inexact measurements.**
- **Identifying a model of a plant from first principles and/or input-output measurments.**

These are fundamental problems for both linear and nonlinear systems.

Fundamental Problems, Mathematician's Opinion

What are the fundamental problems of control?

- **Finding a feedback that stabilizes a plant to an operating point.**
- **Finding an open loop control trajectory that steers a plant from one state to another.**
- **Finding a feedforward and feedback that tracks a reference trajectory.**
- **Estimating the state of a system from partial and inexact measurements.**
- **Identifying a model of a plant from first principles and/or input-output measurments.**

These are fundamental problems for both linear and nonlinear systems.

For linear systems we have theoretical solutions that are easily implemented numerically.

Fundamental Problems, Mathematician's Opinion

What are the fundamental problems of control?

- **Finding a feedback that stabilizes a plant to an operating point.**
- **Finding an open loop control trajectory that steers a plant from one state to another.**
- **Finding a feedforward and feedback that tracks a reference trajectory.**
- **Estimating the state of a system from partial and inexact measurements.**
- **Identifying a model of a plant from first principles and/or input-output measurments.**

These are fundamental problems for both linear and nonlinear systems.

For linear systems we have theoretical solutions that are easily implemented numerically.

For nonlinear systems we have theoretical solutions that usually cannot be implemented numerically.

Fundamental Problems, Practitioners' Opinion

Table 1: Results of a survey by the IFAC Industry Committee on the current and future impact of PID and advanced control technologies

	Current Impact		Future Impact	
Control Technology %	**High**	**Low/No**	**High**	**Low/No**
PID control	91%	0%	78%	6%
System Identification	65%	5%	72%	5%
Estimation & filtering	64%	11%	63%	3%
Model-predictive control	**62%**	**11%**	**85%**	**2%**
Process data analytics	51%	15%	70%	8%
Fault detection & identification	48%	17%	8%	8%
Decentralized and/or coordinated control	29%	33%	54%	11%
Robust control	26%	35%	42%	23%
Intelligent control	24%	38%	59%	11%
Nonlinear control	21%	44%	42%	15%
Discrete-event systems	24%	45%	39%	27%
Adaptive control	18%	38%	44%	17%
Repetitive control	12%	74%	17%	51%
Other advanced control technology	11%	64%	25%	39%
Hybrid dynamical systems	11%	68%	33%	33%
Game theory	5%	76%	17%	52%

Control Systems Toolbox

The Control Systems Toolbox for linear systems is

- **General Purpose**

Control Systems Toolbox

The Control Systems Toolbox for linear systems is

- **General Purpose**
- **Reliable**

Control Systems Toolbox

The Control Systems Toolbox for linear systems is

- **General Purpose**
- **Reliable**
- **Scalable**

Control Systems Toolbox

The Control Systems Toolbox for linear systems is

- **General Purpose**
- **Reliable**
- **Scalable**
- **Portable**

Control Systems Toolbox

The Control Systems Toolbox for linear systems is

- **General Purpose**
- **Reliable**
- **Scalable**
- **Portable**
- **Easy to Use**

Control Systems Toolbox

The Control Systems Toolbox for linear systems is

- **General Purpose**
- **Reliable**
- **Scalable**
- **Portable**
- **Easy to Use**
- **Benchmarked**

Control Systems Toolbox

The Control Systems Toolbox for linear systems is

- **General Purpose**
- **Reliable**
- **Scalable**
- **Portable**
- **Easy to Use**
- **Benchmarked**

Control Systems Toolbox

The Control Systems Toolbox for linear systems is

- **General Purpose**
- **Reliable**
- **Scalable**
- **Portable**
- **Easy to Use**
- **Benchmarked**

Can we develop a toolbox with similar properties that is applicable to a broad class of nonlinear systems?

Control Systems Toolbox

The Control Systems Toolbox for linear systems is

- **General Purpose**
- **Reliable**
- **Scalable**
- **Portable**
- **Easy to Use**
- **Benchmarked**

Can we develop a toolbox with similar properties that is applicable to a broad class of nonlinear systems?

I call my attempt the "Nonlinear Systems Toolbox"

Polynomial Systems

My Nonlinear Systems Toolbox (nst19) deals with polynomial systems around an operating point $x = 0, u = 0$ in continuous time

$$
\begin{aligned}
\dot{x} &= f(x,u) = Fx + Gu + f^{[2]}(x,u) + \cdots + f^{[d]}(x,u) \\
y &= h(x,u) = Hx + Ju + h^{[2]}(x,u) + \cdots + h^{[d]}(x,u)
\end{aligned}
$$

Polynomial Systems

My Nonlinear Systems Toolbox (nst19) deals with polynomial systems around an operating point $x = 0, u = 0$ in continuous time

$$\begin{aligned} \dot{x} &= f(x,u) = Fx + Gu + f^{[2]}(x,u) + \cdots + f^{[d]}(x,u) \\ y &= h(x,u) = Hx + Ju + h^{[2]}(x,u) + \cdots + h^{[d]}(x,u) \end{aligned}$$

or in discrete time

$$\begin{aligned} x^+ &= f(x,u) = Fx + Gu + f^{[2]}(x,u) + \cdots + f^{[d]}(x,u) \\ y &= h(x,u) = Hx + Ju + h^{[2]}(x,u) + \cdots + h^{[d]}(x,u) \end{aligned}$$

Polynomial Systems

My Nonlinear Systems Toolbox (nst19) deals with polynomial systems around an operating point $x = 0, u = 0$ in continuous time

$$\begin{aligned} \dot{x} &= f(x,u) = Fx + Gu + f^{[2]}(x,u) + \cdots + f^{[d]}(x,u) \\ y &= h(x,u) = Hx + Ju + h^{[2]}(x,u) + \cdots + h^{[d]}(x,u) \end{aligned}$$

or in discrete time

$$\begin{aligned} x^+ &= f(x,u) = Fx + Gu + f^{[2]}(x,u) + \cdots + f^{[d]}(x,u) \\ y &= h(x,u) = Hx + Ju + h^{[2]}(x,u) + \cdots + h^{[d]}(x,u) \end{aligned}$$

where $x^+(t) = x(t = 1)$ and $f^{[j]}(x,u)$ denotes terms of homogeneous degree j in x, u.

Polynomial Systems

My Nonlinear Systems Toolbox (nst19) deals with polynomial systems around an operating point $x = 0, u = 0$ in continuous time

$$\begin{aligned} \dot{x} &= f(x,u) = Fx + Gu + f^{[2]}(x,u) + \cdots + f^{[d]}(x,u) \\ y &= h(x,u) = Hx + Ju + h^{[2]}(x,u) + \cdots + h^{[d]}(x,u) \end{aligned}$$

or in discrete time

$$\begin{aligned} x^{+} &= f(x,u) = Fx + Gu + f^{[2]}(x,u) + \cdots + f^{[d]}(x,u) \\ y &= h(x,u) = Hx + Ju + h^{[2]}(x,u) + \cdots + h^{[d]}(x,u) \end{aligned}$$

where $x^{+}(t) = x(t = 1)$ and $f^{[j]}(x,u)$ denotes terms of homogeneous degree j in x, u.

Polynomial systems arise naturally in various contexts, e.g. chemical reactions, predator prey, or they could be the Taylor polynomials of more general smooth systems, e.g. pendula.

Fundamental Data Type

The fundamental data type is a polynomial vector field which is stored as its matrix of coefficients.

Fundamental Data Type

The fundamental data type is a polynomial vector field which is stored as its matrix of coefficients.

For example if the state x and control u dimensions are n and m respectively then this matrix is of dimension

$$n \times \binom{n+m+d}{d}$$

Fundamental Data Type

The fundamental data type is a polynomial vector field which is stored as its matrix of coefficients.

For example if the state x and control u dimensions are n and m respectively then this matrix is of dimension

$$n \times \binom{n+m+d}{d}$$

The argument and value of a polynomial vector field can be vectors of vectors.

Fundamental Data Type

The fundamental data type is a polynomial vector field which is stored as its matrix of coefficients.

For example if the state x and control u dimensions are n and m respectively then this matrix is of dimension

$$n \times \binom{n+m+d}{d}$$

The argument and value of a polynomial vector field can be vectors of vectors.

The quadratic coefficients of $f^{[2]}(x,u)$ are grouped as follows, first the monomials quadratic in x then the monomials bilinear in x and u and finally the monomials quadratic in u .

Fundamental Data Type

The fundamental data type is a polynomial vector field which is stored as its matrix of coefficients.

For example if the state x and control u dimensions are n and m respectively then this matrix is of dimension

$$n \times \binom{n+m+d}{d}$$

The argument and value of a polynomial vector field can be vectors of vectors.

The quadratic coefficients of $f^{[2]}(x,u)$ are grouped as follows, first the monomials quadratic in x then the monomials bilinear in x and u and finally the monomials quadratic in u .

Within groups the monomials are stored in lexographic order with right indices moving faster than left indices.

Alternatives

On could try to manipulate polynomials symbolicaly but this is only possible in low dimensions and low degrees.

Alternatives

On could try to manipulate polynomials symbolicaly but this is only possible in low dimensions and low degrees.

Morover it is impossible to solve even linear equations in symbolic polynomials.

Alternatives

On could try to manipulate polynomials symbolicaly but this is only possible in low dimensions and low degrees.

Morover it is impossible to solve even linear equations in symbolic polynomials.

Another alternative is to forget that multiplication of variables is commutative and then use Kronecker algebra.

Alternatives

On could try to manipulate polynomials symbolicaly but this is only possible in low dimensions and low degrees.

Morover it is impossible to solve even linear equations in symbolic polynomials.

Another alternative is to forget that multiplication of variables is commutative and then use Kronecker algebra.

But this greatly increases the dimensions of the matrices involved.

Alternatives

On could try to manipulate polynomials symbolicaly but this is only possible in low dimensions and low degrees.

Morover it is impossible to solve even linear equations in symbolic polynomials.

Another alternative is to forget that multiplication of variables is commutative and then use Kronecker algebra.

But this greatly increases the dimensions of the matrices involved.

There are $\mathbf{56}$ monomials of degree $\mathbf{3}$ in $\mathbf{6}$ commuting variables.

Alternatives

On could try to manipulate polynomials symbolicaly but this is only possible in low dimensions and low degrees.

Morover it is impossible to solve even linear equations in symbolic polynomials.

Another alternative is to forget that multiplication of variables is commutative and then use Kronecker algebra.

But this greatly increases the dimensions of the matrices involved.

There are $\mathbf{56}$ monomials of degree $\mathbf{3}$ in $\mathbf{6}$ commuting variables.

There are $\mathbf{216}$ monomials of degree $\mathbf{3}$ in $\mathbf{6}$ noncommuting variables.

Alternatives

On could try to manipulate polynomials symbolicaly but this is only possible in low dimensions and low degrees.

Morover it is impossible to solve even linear equations in symbolic polynomials.

Another alternative is to forget that multiplication of variables is commutative and then use Kronecker algebra.

But this greatly increases the dimensions of the matrices involved.

There are 56 monomials of degree 3 in 6 commuting variables.

There are 216 monomials of degree 3 in 6 noncommuting variables.

There are 336 monomials of degree 3 in 12 commuting variables.

Alternatives

On could try to manipulate polynomials symbolicaly but this is only possible in low dimensions and low degrees.

Morover it is impossible to solve even linear equations in symbolic polynomials.

Another alternative is to forget that multiplication of variables is commutative and then use Kronecker algebra.

But this greatly increases the dimensions of the matrices involved.

There are 56 monomials of degree 3 in 6 commuting variables.

There are 216 monomials of degree 3 in 6 noncommuting variables.

There are 336 monomials of degree 3 in 12 commuting variables.

There are 1728 monomials of degree 3 in 6 noncommuting variables.

Basic Operations of NST

- **Directionally differentiate one polynomial vector field in the direction of another** `dd.mex`

Basic Operations of NST

- **Directionally differentiate one polynomial vector field in the direction of another** `dd.mex`
- **Compose two polynomial vector fields** `cmp.mex`**.**

Basic Operations of NST

- **Directionally differentiate one polynomial vector field in the direction of another** `dd.mex`
- **Compose two polynomial vector fields** `cmp.mex`**.**
- **Compute the Jacobian matrix field of a polynomial vector field** `jcbn.mex`**.**
  (Matrix fields are converted to vector fields by row stacking.)

Basic Operations of NST

- **Directionally differentiate one polynomial vector field in the direction of another `dd.mex`**
- **Compose two polynomial vector fields `cmp.mex`.**
- **Compute the Jacobian matrix field of a polynomial vector field `jcbn.mex`.**
  (Matrix fields are converted to vector fields by row stacking.)
- **Multiply a matrix field times another vector or matrix field `mply.mex`.**

Basic Operations of NST

- **Directionally differentiate one polynomial vector field in the direction of another** `dd.mex`
- **Compose two polynomial vector fields** `cmp.mex`**.**
- **Compute the Jacobian matrix field of a polynomial vector field** `jcbn.mex`**.**
  (Matrix fields are converted to vector fields by row stacking.)
- **Multiply a matrix field times another vector or matrix field** `mply.mex`**.**
- **Given numerical values for arguments of a polynomial vector field compute the numerical values of all the monomials in the arguments from a given lower degree to a given upper degree** `mon.mex`**.**

Basic Operations of NST

- **Directionally differentiate one polynomial vector field in the direction of another** `dd.mex`
- **Compose two polynomial vector fields** `cmp.mex`**.**
- **Compute the Jacobian matrix field of a polynomial vector field** `jcbn.mex`**.**
  (Matrix fields are converted to vector fields by row stacking.)
- **Multiply a matrix field times another vector or matrix field** `mply.mex`**.**
- **Given numerical values for arguments of a polynomial vector field compute the numerical values of all the monomials in the arguments from a given lower degree to a given upper degree** `mon.mex`**.**
- **Of course compatiblity conditions must be satisfied**

Basic Operations of NST

- **Directionally differentiate one polynomial vector field in the direction of another** `dd.mex`
- **Compose two polynomial vector fields** `cmp.mex`**.**
- **Compute the Jacobian matrix field of a polynomial vector field** `jcbn.mex`**.**
  (Matrix fields are converted to vector fields by row stacking.)
- **Multiply a matrix field times another vector or matrix field** `mply.mex`**.**
- **Given numerical values for arguments of a polynomial vector field compute the numerical values of all the monomials in the arguments from a given lower degree to a given upper degree** `mon.mex`**.**
- **Of course compatiblity conditions must be satisfied**

Basic Operations of NST

- **Directionally differentiate one polynomial vector field in the direction of another** `dd.mex`
- **Compose two polynomial vector fields** `cmp.mex`**.**
- **Compute the Jacobian matrix field of a polynomial vector field** `jcbn.mex`**.**
  (Matrix fields are converted to vector fields by row stacking.)
- **Multiply a matrix field times another vector or matrix field** `mply.mex`**.**
- **Given numerical values for arguments of a polynomial vector field compute the numerical values of all the monomials in the arguments from a given lower degree to a given upper degree** `mon.mex`**.**
- **Of course compatiblity conditions must be satisfied**

Some Fundamental PDEs of Continuous Time Nonlinear Control

Hamilton-Jacobi-Bellman PDE, unkowns $c = \pi(x), u = \kappa(x)$

$$
\begin{aligned}
0 &= \min_u \left\{ \frac{\partial \pi}{\partial x}(x) f(x,u) + l(x,u) \right\} \\
\kappa(x) &= \operatorname{argmin}_u \left\{ \frac{\partial \pi}{\partial x}(x) f(x,u) + l(x,u) \right\}
\end{aligned}
$$

Some Fundamental PDEs of Continuous Time Nonlinear Control

Hamilton-Jacobi-Bellman PDE, unkowns $c = \pi(x), u = \kappa(x)$

$$\begin{aligned} 0 &= \min_u \left\{ \frac{\partial \pi}{\partial x}(x) f(x,u) + l(x,u) \right\} \\ \kappa(x) &= \operatorname{argmin}_u \left\{ \frac{\partial \pi}{\partial x}(x) f(x,u) + l(x,u) \right\} \end{aligned}$$

Francis-Byrnes-Isidori PDE of Nonlinear Regulation, unkowns $x = \phi(w), u = \lambda(w)$

$$\begin{aligned} f(\phi(w), \lambda(w)) &= \frac{\partial \phi}{\partial w}(w) a(w) \\ h(\phi(w), \lambda(w)) &= 0 \end{aligned}$$

Some Fundamental PDEs of Continuous Time Nonlinear Control

Feedback Linearization PDE, unkowns $x = \phi(z), v = \alpha(z, u)$

$$f(x, u) = \frac{\partial \phi}{\partial z}(z)\left(Fz + G\alpha(z, u)\right)$$

Some Fundamental PDEs of Continuous Time Nonlinear Control

Feedback Linearization PDE, unkowns $x = \phi(z), v = \alpha(z, u)$

$$f(x, u) = \frac{\partial \phi}{\partial z}(z)\left(Fz + G\alpha(z, u)\right)$$

Kazantzis-Kravaris PDE for an observer with linear error dynamics, unknowns $x = \phi(z), \beta(h(x))$

$$f(x) \;\; = \;\; \frac{\partial \phi}{\partial z}(z)\left(F\phi^{-1}(x) + \beta(h(x))\right)$$

Regular Singular Point

What all these PDEs have in common is that we would to solve them in a neighborhood of a regular singular point.

Regular Singular Point

What all these PDEs have in common is that we would to solve them in a neighborhood of a regular singular point.

A first order PDE has a regular singular point at a point where the coefficient of the first derivative term vanishes.

Regular Singular Point

What all these PDEs have in common is that we would to solve them in a neighborhood of a regular singular point.

A first order PDE has a regular singular point at a point where the coefficient of the first derivative term vanishes.

This implies that the Taylor polynomials of the desired solution can be computed degree by degree.

Regular Singular Point

What all these PDEs have in common is that we would to solve them in a neighborhood of a regular singular point.

A first order PDE has a regular singular point at a point where the coefficient of the first derivative term vanishes.

This implies that the Taylor polynomials of the desired solution can be computed degree by degree.

The lowest terms of the Taylor polynomials might satisfy a nonlinear equation but all the higher degree terms will satisfy linear equations.

Regular Singular Point

What all these PDEs have in common is that we would to solve them in a neighborhood of a regular singular point.

A first order PDE has a regular singular point at a point where the coefficient of the first derivative term vanishes.

This implies that the Taylor polynomials of the desired solution can be computed degree by degree.

The lowest terms of the Taylor polynomials might satisfy a nonlinear equation but all the higher degree terms will satisfy linear equations.

Al'brekht did it for HJB PDEs in 1961 and the others were done later but the basic technique goes back Poincare and even Euler.

Regular Singular Point

Suppose $\pi^{[d]}(x)$ is a polynomial homogeneous of degree d.

Regular Singular Point

Suppose $\pi^{[d]}(x)$ is a polynomial homogeneous of degree d.

Then

$$\pi^{[d]}(x) \quad \mapsto \quad \frac{\partial \pi^{[d]}}{\partial x}(x)\,(F + GK)\,x$$

is a linear operator whose eigenvalues are the sums of d eigenvalues of $F + GK$.

Regular Singular Point

Suppose $\pi^{[d]}(x)$ is a polynomial homogeneous of degree d.

Then

$$\pi^{[d]}(x) \quad \mapsto \quad \frac{\partial \pi^{[d]}}{\partial x}(x)\,(F+GK)\,x$$

is a linear operator whose eigenvalues are the sums of d eigenvalues of $F+GK$.

Functional Equations

Bellman's Dynamic Programming Equation is a functional equation in the unknown functions $\pi(x), \kappa(x)$.

$$
\begin{aligned}
\pi(x) &= \min_u \{\pi(f(x,u)) + l(x,u)\} \\
\kappa(x) &= \operatorname{argmin}_u \{\pi(f(x,u)) + l(x,u)\}
\end{aligned}
$$

Functional Equations

Bellman's Dynamic Programming Equation is a functional equation in the unknown functions $\pi(x), \kappa(x)$.

$$
\begin{aligned}
\pi(x) &= \min_u \{\pi(f(x,u)) + l(x,u)\} \\
\kappa(x) &= \operatorname{argmin}_u \{\pi(f(x,u)) + l(x,u)\}
\end{aligned}
$$

The corresponding linear operator is

$$
\pi^{[d]}(x) \mapsto \pi^{[d]}(x) - \pi^{[d]}((F+GK)\,x)
$$

Functional Equations

Bellman's Dynamic Programming Equation is a functional equation in the unknown functions $\pi(x), \kappa(x)$.

$$
\begin{aligned}
\pi(x) &= \min_u \{\pi(f(x,u)) + l(x,u)\} \\
\kappa(x) &= \operatorname{argmin}_u \{\pi(f(x,u)) + l(x,u)\}
\end{aligned}
$$

The corresponding linear operator is

$$
\pi^{[d]}(x) \mapsto \pi^{[d]}(x) - \pi^{[d]}((F+GK)x)
$$

Its eigenvalues are 1 minus the product of d eigenvalues of $F+GK$.

Stochastic Optimal Control

The stochastic version of HJB equation is a second order nonlinear PDE.

Stochastic Optimal Control

The stochastic version of HJB equation is a second order nonlinear PDE.

It has a regular singular point at the operating point if the coefficients of the white noise in the Ito equation vanishes at the operating point.

Stochastic Optimal Control

The stochastic version of HJB equation is a second order nonlinear PDE.

It has a regular singular point at the operating point if the coefficients of the white noise in the Ito equation vanishes at the operating point.

Hence the Taylor polynomials of the expected optimal cost and optimal feedback can be computed degree by degree.

Stochastic Optimal Control

The stochastic version of HJB equation is a second order nonlinear PDE.

It has a regular singular point at the operating point if the coefficients of the white noise in the Ito equation vanishes at the operating point.

Hence the Taylor polynomials of the expected optimal cost and optimal feedback can be computed degree by degree.

In discrete time the stochastic Dynamic Programming equation for the expected optimal cost and optimal feedback can also be computed degree by degree if the coefficients of the noise in the dynamics vanishes at the operating point.

Nonlinear System Toolbox 2019

The Nonlinear Systems Toolbox is a package of MATLAB .m, .mlx and .mex files for the analysis and synthesis of nonlinear control systems described by polynomials. Some of these .m files are implemented by .mex files.

Nonlinear System Toolbox 2019

The Nonlinear Systems Toolbox is a package of MATLAB .m, .mlx and .mex files for the analysis and synthesis of nonlinear control systems described by polynomials. Some of these .m files are implemented by .mex files.

The basic data type is a vector field which is polynomial of arbitrary degree in a vector which is composed of an arbitrary number of subvectors.

Nonlinear System Toolbox 2019

The Nonlinear Systems Toolbox is a package of MATLAB .m, .mlx and .mex files for the analysis and synthesis of nonlinear control systems described by polynomials. Some of these .m files are implemented by .mex files.

The basic data type is a vector field which is polynomial of arbitrary degree in a vector which is composed of an arbitrary number of subvectors.

Of course, memory and speed limitations will implicitly restrict the degrees of the vector fields and dimensions of vectors.

Nonlinear System Toolbox 2019

The Nonlinear Systems Toolbox is a package of MATLAB .m, .mlx and .mex files for the analysis and synthesis of nonlinear control systems described by polynomials. Some of these .m files are implemented by .mex files.

The basic data type is a vector field which is polynomial of arbitrary degree in a vector which is composed of an arbitrary number of subvectors.

Of course, memory and speed limitations will implicitly restrict the degrees of the vector fields and dimensions of vectors.

NST also works with polynomial matrix fields, polynomial higher order tensor fields and complex fields.

Nonlinear System Toolbox 2019

`hjb.m` **Computes degree by degree the solution to the HJB equations for a continuous time, infinite horizon optimal control problem.**

Nonlinear System Toolbox 2019

`hjb.m` **Computes degree by degree the solution to the HJB equations for a continuous time, infinite horizon optimal control problem.**

`dpe.m` **Computes degree by degree the solution to the Dynamic Programming equation for a discrete time, infinite horizon optimal control problem.**

Nonlinear System Toolbox 2019

`hjb.m` **Computes degree by degree the solution to the HJB equations for a continuous time, infinite horizon optimal control problem.**

`dpe.m` **Computes degree by degree the solution to the Dynamic Programming equation for a discrete time, infinite horizon optimal control problem.**

`tay_poly.m` **Compute the Taylor polynomial of a symbolic vector field.**

Nonlinear System Toolbox 2019

`hjb.m` **Computes degree by degree the solution to the HJB equations for a continuous time, infinite horizon optimal control problem.**

`dpe.m` **Computes degree by degree the solution to the Dynamic Programming equation for a discrete time, infinite horizon optimal control problem.**

`tay_poly.m` **Compute the Taylor polynomial of a symbolic vector field.**

`hjb_set_up.m` **Takes a symbolic infinite horizon optimal control problem and computes the Taylor polynomials needed by** `hjb.m` **or** `dpe.m` **.**

Nonlinear System Toolbox 2019

`hjb.m` Computes degree by degree the solution to the HJB equations for a continuous time, infinite horizon optimal control problem.

`dpe.m` Computes degree by degree the solution to the Dynamic Programming equation for a discrete time, infinite horizon optimal control problem.

`tay_poly.m` Compute the Taylor polynomial of a symbolic vector field.

`hjb_set_up.m` Takes a symbolic infinite horizon optimal control problem and computes the Taylor polynomials needed by `hjb.m` or `dpe.m` .

`fbi.m` Computes degree by degree the solution of the Francis-Byrnes-Isidori PDE of continuous time nonlinear regulation.

Nonlinear System Toolbox 2019

`hjb.m` **Computes degree by degree the solution to the HJB equations for a continuous time, infinite horizon optimal control problem.**

`dpe.m` **Computes degree by degree the solution to the Dynamic Programming equation for a discrete time, infinite horizon optimal control problem.**

`tay_poly.m` **Compute the Taylor polynomial of a symbolic vector field.**

`hjb_set_up.m` **Takes a symbolic infinite horizon optimal control problem and computes the Taylor polynomials needed by** `hjb.m` **or** `dpe.m` **.**

`fbi.m` **Computes degree by degree the solution of the Francis-Byrnes-Isidori PDE of continuous time nonlinear regulation.**

`d_fbi.m` **Computes degree by degree the solution of the Huang-Lin functional equation of discrete time nonlinear regulation.**

Nonlinear System Toolbox 2019

`hjb.m` **Computes degree by degree the solution to the HJB equations for a continuous time, infinite horizon optimal control problem.**

`dpe.m` **Computes degree by degree the solution to the Dynamic Programming equation for a discrete time, infinite horizon optimal control problem.**

`tay_poly.m` **Compute the Taylor polynomial of a symbolic vector field.**

`hjb_set_up.m` **Takes a symbolic infinite horizon optimal control problem and computes the Taylor polynomials needed by** `hjb.m` **or** `dpe.m` **.**

`fbi.m` **Computes degree by degree the solution of the Francis-Byrnes-Isidori PDE of continuous time nonlinear regulation.**

`d_fbi.m` **Computes degree by degree the solution of the Huang-Lin functional equation of discrete time nonlinear regulation.**

`csq.m` **Extends a polynomial of degrees 2 through d+1 whose quadratic part is positive definite to a polynomial of degrees 2 through 2d which is a sum of squares.**

ch_crds.m

Nonlinear System Toolbox 2019

`ch_crds.m`

`d_ch_crds.m`

Nonlinear System Toolbox 2019

`ch_crds.m`

`d_ch_crds.m`

`dsp.m`

Nonlinear System Toolbox 2019

`ch_crds.m`

`d_ch_crds.m`

`dsp.m`

`fbk_lin.m`

Nonlinear System Toolbox 2019

`ch_crds.m`

`d_ch_crds.m`

`dsp.m`

`fbk_lin.m`

`hji.m`

Nonlinear System Toolbox 2019

ch_crds.m

d_ch_crds.m

dsp.m

fbk_lin.m

hji.m

inv_mfd.m

Nonlinear System Toolbox 2019

ch_crds.m

d_ch_crds.m

dsp.m

fbk_lin.m

hji.m

inv_mfd.m

obskk.m

Nonlinear System Toolbox 2019

ch_crds.m

d_ch_crds.m

dsp.m

fbk_lin.m

hji.m

inv_mfd.m

obskk.m

zbv.m

Example Optimal Stabilization

Problems with Al'brekht's Method

But Al'brekhts method has its limitations including the following.

Problems with Al'brekht's Method

But Al'brekhts method has its limitations including the following.

- **The Taylor polynomial of the optimal cost to degree $d + 1 > 2$ need not be positive definite.**

Problems with Al'brekht's Method

But Al'brekhts method has its limitations including the following.

- The Taylor polynomial of the optimal cost to degree $d+1>2$ need not be positive definite.
- Al'brekht's Method cannot handle state and/or control constraints.

Problems with Al'brekht's Method

But Al'brekhts method has its limitations including the following.

- **The Taylor polynomial of the optimal cost to degree $d + 1 > 2$ need not be positive definite.**
- **Al'brekht's Method cannot handle state and/or control constraints.**
- **There is no effective way to compute the domain on which the Taylor polynomial of the optimal cost is a valid Lyapunov function for the closed loop dynamics using the Taylor polynomial of the optimal feedback in dimensions $n > 2$.**

Problems with Al'brekht's Method

But Al'brekhts method has its limitations including the following.

- **The Taylor polynomial of the optimal cost to degree $d+1>2$ need not be positive definite.**
- **Al'brekht's Method cannot handle state and/or control constraints.**
- **There is no effective way to compute the domain on which the Taylor polynomial of the optimal cost is a valid Lyapunov function for the closed loop dynamics using the Taylor polynomial of the optimal feedback in dimensions $n>2$.**
- **Increasing the degrees $d+1,\ d$ of the Taylor polynomials does not necesarily increase the size of this domain.**

Problems with Al'brekht's Method

But Al'brekhts method has its limitations including the following.

- **The Taylor polynomial of the optimal cost to degree $d+1>2$ need not be positive definite.**
- **Al'brekht's Method cannot handle state and/or control constraints.**
- **There is no effective way to compute the domain on which the Taylor polynomial of the optimal cost is a valid Lyapunov function for the closed loop dynamics using the Taylor polynomial of the optimal feedback in dimensions $n>2$.**
- **Increasing the degrees $d+1,\ d$ of the Taylor polynomials does not necesarily increase the size of this domain.**

Problems with Al'brekht's Method

But Al'brekhts method has its limitations including the following.

- **The Taylor polynomial of the optimal cost to degree $d+1>2$ need not be positive definite.**
- **Al'brekht's Method cannot handle state and/or control constraints.**
- **There is no effective way to compute the domain on which the Taylor polynomial of the optimal cost is a valid Lyapunov function for the closed loop dynamics using the Taylor polynomial of the optimal feedback in dimensions $n>2$.**
- **Increasing the degrees $d+1,\ d$ of the Taylor polynomials does not necesarily increase the size of this domain.**

Completing the Squares

Problem: The Taylor polynomial of the optimal cost to degree $d+1>2$ need not be positive definite.

Completing the Squares

Problem: The Taylor polynomial of the optimal cost to degree $d+1>2$ need not be positive definite.

Solution: Any polynomial of degrees 2 through $d+1$ whose quadratic part is positive definite can be extended to a polynomial of degrees 2 through $2d$ which is a sum of squares and hence at least nonnegative definite.

Completing the Squares

Problem: The Taylor polynomial of the optimal cost to degree $d+1>2$ need not be positive definite.

Solution: Any polynomial of degrees 2 through $d+1$ whose quadratic part is positive definite can be extended to a polynomial of degrees 2 through $2d$ which is a sum of squares and hence at least nonnegative definite.

We call this technique Completing the Squares.

Model Pedictive Control

Problem: Al'brekht's Method cannot handle state and/or control constraints.

Model Pedictive Control

Problem: Al'brekht's Method cannot handle state and/or control constraints.

Solution: Assuming the constraints are not active at the operating point, complete the squares of a Taylor polynomial of the optimal cost and use it as the terminal cost in a Model Predictive scheme.

Model Pedictive Control

Problem: Al'brekht's Method cannot handle state and/or control constraints.

Solution: Assuming the constraints are not active at the operating point, complete the squares of a Taylor polynomial of the optimal cost and use it as the terminal cost in a Model Predictive scheme.

Presumably the higher the degree of the Taylor polynomial of the optimal cost the larger the domain on which the extended polynomial is a valid control Lyapunov function.

Model Pedictive Control

Problem: Al'brekht's Method cannot handle state and/or control constraints.

Solution: Assuming the constraints are not active at the operating point, complete the squares of a Taylor polynomial of the optimal cost and use it as the terminal cost in a Model Predictive scheme.

Presumably the higher the degree of the Taylor polynomial of the optimal cost the larger the domain on which the extended polynomial is a valid control Lyapunov function.

If this is so then a shorter horizon can be used in the Model Predictive scheme.

Adaptive Horizon Model Pedictive Control

Problem: There is no effective way to compute the domain on which the extended Taylor polynomial of the optimal cost is a valid Lyapunov function for the closed loop dynamics using the Taylor polynomial of the optimal feedback in dimensions $n > 2$.

Adaptive Horizon Model Pedictive Control

Problem: There is no effective way to compute the domain on which the extended Taylor polynomial of the optimal cost is a valid Lyapunov function for the closed loop dynamics using the Taylor polynomial of the optimal feedback in dimensions $n > 2$.

Solution: Don't try to compute this domain. Instead verify in real time that the endpoint of the optimal trajectory computed by the solver is in this domain.

Adaptive Horizon Model Pedictive Control

Problem: There is no effective way to compute the domain on which the extended Taylor polynomial of the optimal cost is a valid Lyapunov function for the closed loop dynamics using the Taylor polynomial of the optimal feedback in dimensions $n > 2$.

Solution: Don't try to compute this domain. Instead verify in real time that the endpoint of the optimal trajectory computed by the solver is in this domain.

We do this by extending the computed optimal trajectory an additional few times steps under the closed loop dynamics using the Taylor polynomial of the optimal feedback.

Adaptive Horizon Model Pedictive Control

Problem: There is no effective way to compute the domain on which the extended Taylor polynomial of the optimal cost is a valid Lyapunov function for the closed loop dynamics using the Taylor polynomial of the optimal feedback in dimensions $n > 2$.

Solution: Don't try to compute this domain. Instead verify in real time that the endpoint of the optimal trajectory computed by the solver is in this domain.

We do this by extending the computed optimal trajectory an additional few times steps under the closed loop dynamics using the Taylor polynomial of the optimal feedback.

We check that Lyapunov and feasibility conditions are satisfied on the extension.

Adaptive Horizon Model Pedictive Control

Problem: There is no effective way to compute the domain on which the extended Taylor polynomial of the optimal cost is a valid Lyapunov function for the closed loop dynamics using the Taylor polynomial of the optimal feedback in dimensions $n > 2$.

Solution: Don't try to compute this domain. Instead verify in real time that the endpoint of the optimal trajectory computed by the solver is in this domain.

We do this by extending the computed optimal trajectory an additional few times steps under the closed loop dynamics using the Taylor polynomial of the optimal feedback.

We check that Lyapunov and feasibility conditions are satisfied on the extension.

If they are we conclude that the current horizon is long enough.

Adaptive Horizon Model Pedictive Control

Problem: There is no effective way to compute the domain on which the extended Taylor polynomial of the optimal cost is a valid Lyapunov function for the closed loop dynamics using the Taylor polynomial of the optimal feedback in dimensions $n > 2$.

Solution: Don't try to compute this domain. Instead verify in real time that the endpoint of the optimal trajectory computed by the solver is in this domain.

We do this by extending the computed optimal trajectory an additional few times steps under the closed loop dynamics using the Taylor polynomial of the optimal feedback.

We check that Lyapunov and feasibility conditions are satisfied on the extension.

If they are we conclude that the current horizon is long enough.

If they are not we conclude that the current horizon is too short and we increase it.

Example

We simulated AHMPC with two different terminal costs and terminal feedbacks. In both cases the Lagrangian was

$$\frac{0.1}{2}\left(|x|^2+|u|^2\right)$$

The first pair $\pi_f^2(x), \kappa_f^1(x)$ was found by solving the infinite horizon LQR problem obtained by taking the linear part of the dynamics around the operating point $x=0$ and the quadratic Lagrangian. Then $d=1$, $\pi_f^2(x)$ is a positive definite quadratic form and $\kappa_f^1(x)$ is linear function.

Example

We simulated AHMPC with two different terminal costs and terminal feedbacks. In both cases the Lagrangian was

$$\frac{0.1}{2}\left(|x|^2+|u|^2\right)$$

The first pair $\pi_f^2(x), \kappa_f^1(x)$ was found by solving the infinite horizon LQR problem obtained by taking the linear part of the dynamics around the operating point $x=0$ and the quadratic Lagrangian. Then $d=1$, $\pi_f^2(x)$ is a positive definite quadratic form and $\kappa_f^1(x)$ is linear function.

The second pair $\pi_f^6(x), \kappa_f^5(x)$ was found using the discrete time version of Al'brekht's method to degree $d=5$. Then $\pi_f^6(x)$ is the Taylor polynomial of the optimal cost to degree 6 and $\kappa_f^5(x)$ is the Taylor polynomial of the optimal feedback to degree 5. But $\pi_f^6(x)$ is not positive definite so we completed the squares to get $\pi_f^{10}(x)$ which is positive definite.

Example

In all the simulations we imposed the control constraint $|u|_\infty \leq 4$ and started at $x(0) = (0.9\pi, 0.9\pi, 0, 0)$ with an initial horizon of $T = 50$ time steps.

Example

In all the simulations we imposed the control constraint $|u|_\infty \leq 4$ and started at $x(0) = (0.9\pi, 0.9\pi, 0, 0)$ with an initial horizon of $T = 50$ time steps.

The extended horizon was kept constant at $S = 5$.

Example

In all the simulations we imposed the control constraint $|u|_\infty \leq 4$ and started at $x(0) = (0.9\pi, 0.9\pi, 0, 0)$ with an initial horizon of $T = 50$ time steps.

The extended horizon was kept constant at $S = 5$.

If the Lyapunov and/or feasibility conditions were violated the horizon T was increased by 5 and the finite horizon nonlinear program was solved again without advancing the system.

Example

In all the simulations we imposed the control constraint $|u|_\infty \le 4$ and started at $x(0) = (0.9\pi, 0.9\pi, 0, 0)$ with an initial horizon of $T = 50$ time steps.

The extended horizon was kept constant at $S = 5$.

If the Lyapunov and/or feasibility conditions were violated the horizon T was increased by 5 and the finite horizon nonlinear program was solved again without advancing the system.

If after three tries the Lyapunov and/or feasibility conditions were still not satisfied then the first value of the control sequence was used, the simulation was advanced one time step and the horizon was increased by 5.

Example

In all the simulations we imposed the control constraint $|u|_\infty \leq 4$ and started at $x(0) = (0.9\pi, 0.9\pi, 0, 0)$ with an initial horizon of $T = 50$ time steps.

The extended horizon was kept constant at $S = 5$.

If the Lyapunov and/or feasibility conditions were violated the horizon T was increased by 5 and the finite horizon nonlinear program was solved again without advancing the system.

If after three tries the Lyapunov and/or feasibility conditions were still not satisfied then the first value of the control sequence was used, the simulation was advanced one time step and the horizon was increased by 5.

If the Lyapunov and feasibility conditions were comfortably satisfied over the extended horizon then the simulation was advanced one time step and the horizon T was decreased by 1 .

Example

The simulations were first run with no noise and the results are shown in the following figures.

Example

The simulations were first run with no noise and the results are shown in the following figures.

Both methods stabilized the links to straight up in about 80 times steps (8 seconds).

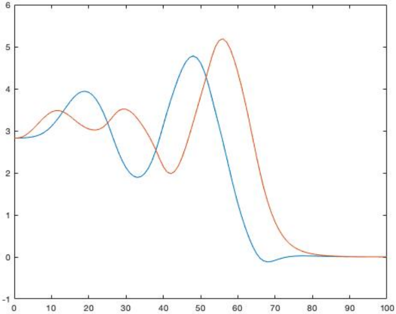
Example

The simulations were first run with no noise and the results are shown in the following figures.

Both methods stabilized the links to straight up in about 80 times steps (8 seconds).

The degree $2d = 10$ terminal cost and the degree $d = 5$ terminal feedback seems to do it a little more smoothly and with shorter maximum horizon $T = 65$ versus $T = 75$ for LQR $d = 1$.

Example



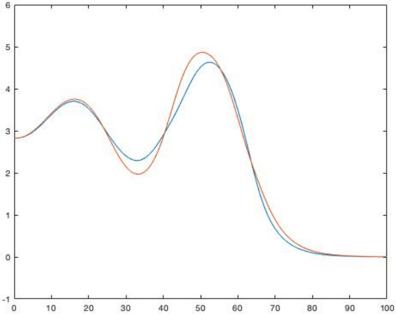


Figure: Angles, $\boldsymbol{d = 1}$ on left, $\boldsymbol{d = 5}$ on right

Example

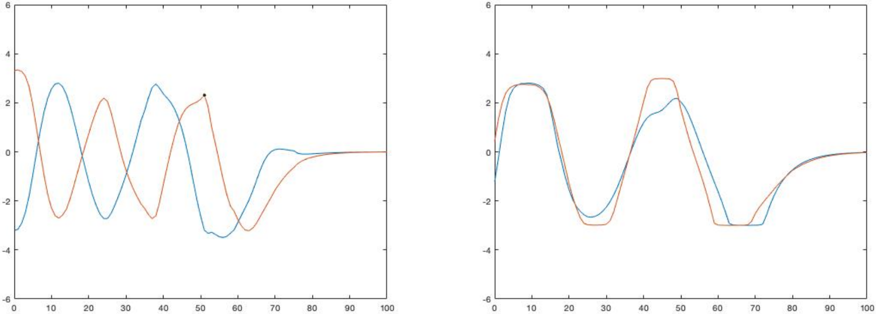


Figure: Controls, $\boldsymbol{d = 1}$ on left, $\boldsymbol{d = 5}$ on right

Example

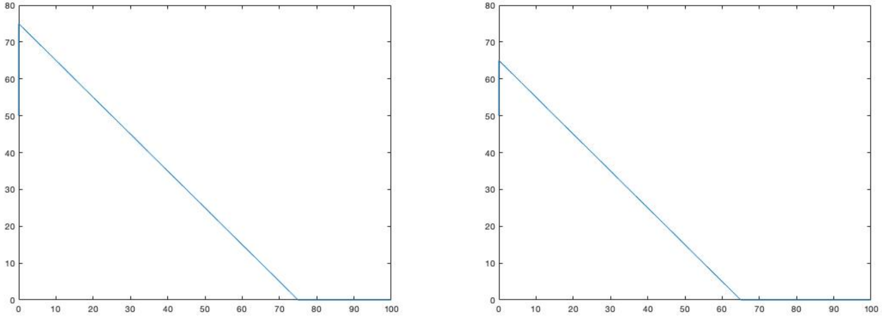


Figure: Horizons, $\boldsymbol{d = 1}$ on left, $\boldsymbol{d = 5}$ on right

Example

The simulations lasted 100 time steps, 10 seconds.

Example

The simulations lasted 100 time steps, 10 seconds.

The simulations were done on this laptop using MATLAB's `fmincon.m` **with its default settings.**

Example

The simulations lasted 100 time steps, 10 seconds.

The simulations were done on this laptop using MATLAB's `fmincon.m` with its default settings.

We did supply `fmincon.m` the gradients of the objective functions but we did not give it the Hessians.

Example

The simulations lasted 100 time steps, 10 seconds.

The simulations were done on this laptop using MATLAB's `fmincon.m` **with its default settings.**

We did supply `fmincon.m` **the gradients of the objective functions but we did not give it the Hessians.**

The cpu time for the degree $2d = 10$ terminal cost and the degree $d = 5$ terminal feedback was 5.01 seconds. So using a faster solver, coding the objective and its gradient in C^+ , compiling it and with a little tweaking of the algorithm, supplying Hessians we probably could control the double pendula in real time.

Example

The simulations lasted 100 time steps, 10 seconds.

The simulations were done on this laptop using MATLAB's `fmincon.m` with its default settings.

We did supply `fmincon.m` the gradients of the objective functions but we did not give it the Hessians.

The cpu time for the degree $2d = 10$ terminal cost and the degree $d = 5$ terminal feedback was 5.01 seconds. So using a faster solver, coding the objective and its gradient in C^+ , compiling it and with a little tweaking of the algorithm, supplying Hessians we probably could control the double pendula in real time.

The cpu time for the LQR terminal cost and terminal feedback was 24.56 seconds so it not clear that it is possible to control the double pendula in real time using LQR.

Example

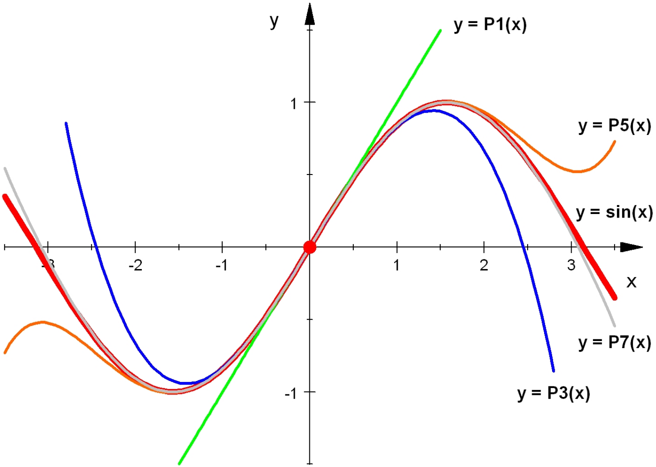


Figure: Taylor Approximations to $y = \sin x$

Rigid Body

We applied Al'brekht's method to the stabilzation of rigid body. The state dimension is $\mathbf{12}$ and the control dimension is $\mathbf{6}$.

Rigid Body

We applied Al'brekht's method to the stabilzation of rigid body. The state dimension is $\mathbf{12}$ and the control dimension is $\mathbf{6}$.

Using MATLAB' symbolic engine, `hjb_set_up.m` computed the Taylor polynomial of the dynamics to degree $\mathbf{3}$ in about $\mathbf{90}$ sec.

Rigid Body

We applied Al'brekht's method to the stabilzation of rigid body. The state dimension is $\mathbf{12}$ and the control dimension is $\mathbf{6}$.

Using MATLAB' symbolic engine, `hjb_set_up.m` computed the Taylor polynomial of the dynamics to degree $\mathbf{3}$ in about $\mathbf{90}$ sec.

`hjb.m` computed the Taylor polynomials of the optimal cost and the optimal feedback to degrees $\mathbf{4}$ and $\mathbf{3}$ in less than $\mathbf{5}$ sec.

Rigid Body

We applied Al'brekht's method to the stabilzation of rigid body. The state dimension is 12 and the control dimension is 6.

Using MATLAB' symbolic engine, `hjb_set_up.m` computed the Taylor polynomial of the dynamics to degree 3 in about 90 sec.

`hjb.m` computed the Taylor polynomials of the optimal cost and the optimal feedback to degrees 4 and 3 in less than 5 sec.

MATLAB couldn't compute the Taylor polynomial to degree 5.

Rigid Body

We applied Al'brekht's method to the stabilzation of rigid body. The state dimension is 12 and the control dimension is 6.

Using MATLAB' symbolic engine, `hjb_set_up.m` computed the Taylor polynomial of the dynamics to degree 3 in about 90 sec.

`hjb.m` computed the Taylor polynomials of the optimal cost and the optimal feedback to degrees 4 and 3 in less than 5 sec.

MATLAB couldn't compute the Taylor polynomial to degree 5.

`hjb.m` computed the Taylor polynomials of the optimal cost and the optimal feedback to degrees 6 and 5 in about 27 minutes.

Rigid Body

We applied Al'brekht's method to the stabilzation of rigid body. The state dimension is 12 and the control dimension is 6.

Using MATLAB' symbolic engine, `hjb_set_up.m` computed the Taylor polynomial of the dynamics to degree 3 in about 90 sec.

`hjb.m` computed the Taylor polynomials of the optimal cost and the optimal feedback to degrees 4 and 3 in less than 5 sec.

MATLAB couldn't compute the Taylor polynomial to degree 5.

`hjb.m` computed the Taylor polynomials of the optimal cost and the optimal feedback to degrees 6 and 5 in about 27 minutes.

There are 33648 monomials of degrees 1 though 5 and 134577 monomials of degrees 2 though 6 in 18 variables.

Rigid Body

We applied Al'brekht's method to the stabilzation of rigid body. The state dimension is 12 and the control dimension is 6.

Using MATLAB' symbolic engine, `hjb_set_up.m` computed the Taylor polynomial of the dynamics to degree 3 in about 90 sec.

`hjb.m` computed the Taylor polynomials of the optimal cost and the optimal feedback to degrees 4 and 3 in less than 5 sec.

MATLAB couldn't compute the Taylor polynomial to degree 5.

`hjb.m` computed the Taylor polynomials of the optimal cost and the optimal feedback to degrees 6 and 5 in about 27 minutes.

There are 33648 monomials of degrees 1 though 5 and 134577 monomials of degrees 2 though 6 in 18 variables.

The infinity norm of the residual of the first HJB equation is $2.2666e-12$

Verification and Benchmarks

One nice thing about polynomial approximations to the optimal cost and optimal feedback is that one can compute infinity norm of the residual of the first HJB equation.

Verification and Benchmarks

One nice thing about polynomial approximations to the optimal cost and optimal feedback is that one can compute infinity norm of the residual of the first HJB equation.

Another approach is to integrate the closed loop dynamics and Lagrangian forward from a random initial condition and check that the losed loop dynamics is feasible and asymptotically stable and that the integral of the Lagrangian approximates the value of the computed optimal cost.

Verification and Benchmarks

One nice thing about polynomial approximations to the optimal cost and optimal feedback is that one can compute infinity norm of the residual of the first HJB equation.

Another approach is to integrate the closed loop dynamics and Lagrangian forward from a random initial condition and check that the losed loop dynamics is feasible and asymptotically stable and that the integral of the Lagrangian approximates the value of the computed optimal cost.

We have very few truly nonlinear optimal control problems where the true solution is known. But if we know the exact solution to LQRs. If we make a nonlinear change of coordinates and nonlinear invertible feedback to an LQR we get a nonlinear problem where the exact solution is known exactly.

Verification and Benchmarks

One nice thing about polynomial approximations to the optimal cost and optimal feedback is that one can compute infinity norm of the residual of the first HJB equation.

Another approach is to integrate the closed loop dynamics and Lagrangian forward from a random initial condition and check that the losed loop dynamics is feasible and asymptotically stable and that the integral of the Lagrangian approximates the value of the computed optimal cost.

We have very few truly nonlinear optimal control problems where the true solution is known. But if we know the exact solution to LQRs. If we make a nonlinear change of coordinates and nonlinear invertible feedback to an LQR we get a nonlinear problem where the exact solution is known exactly.

We need more benchmark problems.

Conclusions

Polynomial systems are natural extensions of linear systems.

Conclusions

Polynomial systems are natural extensions of linear systems.

Many real world plants lend themselves to modeling by polynomial systems. Others arise as the Taylor expansions of smooth systems.

Conclusions

Polynomial systems are natural extensions of linear systems.

Many real world plants lend themselves to modeling by polynomial systems. Others arise as the Taylor expansions of smooth systems.

Many control related PDEs and Functional Equations have a regular singular point at the origin and hence the Taylor polynomials of their solutions can be found degree by degree.

Conclusions

Polynomial systems are natural extensions of linear systems.

Many real world plants lend themselves to modeling by polynomial systems. Others arise as the Taylor expansions of smooth systems.

Many control related PDEs and Functional Equations have a regular singular point at the origin and hence the Taylor polynomials of their solutions can be found degree by degree.

Higher degree Taylor polynomials are more accurate locally around the operating point but they may diverge rapidly away from it.

Conclusions

Polynomial systems are natural extensions of linear systems.

Many real world plants lend themselves to modeling by polynomial systems. Others arise as the Taylor expansions of smooth systems.

Many control related PDEs and Functional Equations have a regular singular point at the origin and hence the Taylor polynomials of their solutions can be found degree by degree.

Higher degree Taylor polynomials are more accurate locally around the operating point but they may diverge rapidly away from it.

Taylor polynomials can be used as a warm start for grid based methods and thereby reduce the number of grid points.

Conclusions

Polynomial systems are natural extensions of linear systems.

Many real world plants lend themselves to modeling by polynomial systems. Others arise as the Taylor expansions of smooth systems.

Many control related PDEs and Functional Equations have a regular singular point at the origin and hence the Taylor polynomials of their solutions can be found degree by degree.

Higher degree Taylor polynomials are more accurate locally around the operating point but they may diverge rapidly away from it.

Taylor polynomials can be used as a warm start for grid based methods and thereby reduce the number of grid points.

Adaptive Horizon Model Predictive Control combines the Taylor polynomials of the optimal cost and feedback with Model Predictive Control to overcome their respective weaknesses.

Think Mathematically

Think Mathematically

Act Computationally

Thank You

Thank You

Questions