Computational Issues in Nonlinear Control and Estimation

Arthur J Krener
Naval Postgraduate School
Monterey, CA 93943

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.

Jack Little, an engineer, was exposed to it during a visit Moler made to Stanford University 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.

Jack Little, an engineer, was exposed to it during a visit Moler made to Stanford University 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.

Jack Little, an engineer, was exposed to it during a visit Moler made to Stanford University 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 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 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 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 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.

The Control Systems Toolbox for linear systems is

- **General Purpose**

Nonlinear Systems Toolbox 2016

The Control Systems Toolbox for linear systems is

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

Nonlinear Systems Toolbox 2016

The Control Systems Toolbox for linear systems is

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

Nonlinear Systems Toolbox 2016

The Control Systems Toolbox for linear systems is

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

Nonlinear Systems Toolbox 2016

The Control Systems Toolbox for linear systems is

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

Nonlinear Systems Toolbox 2016

The Control Systems Toolbox for linear systems is

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

Nonlinear Systems Toolbox 2016

The Control Systems Toolbox for linear systems is

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

Nonlinear Systems Toolbox 2016

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?

Nonlinear Systems Toolbox 2016

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?

In the rest of this talk I would to present a few baby steps in this direction, which can be found in NST 2016.

Nonlinear Systems Toolbox 2016

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?

In the rest of this talk I would to present a few baby steps in this direction, which can be found in NST 2016.

It should be noted that the numerical optimization community has already developed software with these properties for a wide variety of applications like Mixed Intger Linear Programs (i.e., GUROBI and CPLEX) and Nonlinear Programs (i.e., KNITRO and IPOPT). For optimzation solver benchmarks see http://plato.la.asu.edu/bench.html

Smooth Nonlinear Systems

A smooth nonlinear system is of the form

$$\begin{aligned} \dot{x} &= f(x,u) \\ y &= h(x,u) \end{aligned}$$

with state $x \in \mathbb{R}^{n\times 1}$, input $u \in \mathbb{R}^{m\times 1}$ and output $y \in \mathbb{R}^{p\times 1}$

Smooth Nonlinear Systems

A smooth nonlinear system is of the form

$$\begin{aligned} \dot{x} &= f(x,u) \\ y &= h(x,u) \end{aligned}$$

with state $x \in \mathbb{R}^{n\times 1}$ **, input** $u \in \mathbb{R}^{m\times 1}$ **and output** $y \in \mathbb{R}^{p\times 1}$

By smooth we mean that $f(x,u), h(x,u)$ **have as many continuous derivatives as needed by problem and its solution.**

Smooth Nonlinear Systems

A smooth nonlinear system is of the form

$$\begin{aligned} \dot{x} &= f(x,u) \\ y &= h(x,u) \end{aligned}$$

with state $x \in \mathbb{R}^{n\times 1}$, input $u \in \mathbb{R}^{m\times 1}$ and output $y \in \mathbb{R}^{p\times 1}$

By smooth we mean that $f(x,u), h(x,u)$ have as many continuous derivatives as needed by problem and its solution.

Later we shall relax this to piecewise smooth.

Smooth Nonlinear Systems

A smooth nonlinear system is of the form

$$\begin{aligned} \dot{x} &= f(x,u) \\ y &= h(x,u) \end{aligned}$$

with state $x \in I\!R^{n\times 1}$ **, input** $u \in I\!R^{m\times 1}$ **and output** $y \in I\!R^{p\times 1}$

By smooth we mean that $f(x,u), h(x,u)$ **have as many continuous derivatives as needed by problem and its solution.**

Later we shall relax this to piecewise smooth.

There may be state and control constraints of the form

$$g(x,u) \leq 0$$

Any problem with other than linear equality constraints is inherently nonlinear even if $f(x,u), h(x,u)$ **are linear in** x, u **.**

Fundamental Problems

What are the fundamental problems of control?

Fundamental Problems

What are the fundamental problems of control?

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

Fundamental Problems

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

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.**
- **Estimating the state of a system from partial and inexact measurements.**

Fundamental Problems

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.**
- **Estimating the state of a system from partial and inexact measurements.**
- **Finding a feedforward and feedback that tracks a reference trajectory.**

Fundamental Problems

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.**
- **Estimating the state of a system from partial and inexact measurements.**
- **Finding a feedforward and feedback that tracks a reference trajectory.**
- **Your favorite problem.**

Linear Solutions

These are fundamental problems for both linear and nonlinear systems. For linear systems we have theoretical solutions that are easily implemented numerically. Here is a few.

- **The Linear Quadratic Regulator (LQR) can be used for optimally stabilizing a linear plant.**

Linear Solutions

These are fundamental problems for both linear and nonlinear systems. For linear systems we have theoretical solutions that are easily implemented numerically. Here is a few.

- **The Linear Quadratic Regulator (LQR) can be used for optimally stabilizing a linear plant.**
- **Linear controllability and the Controllability Gramian lead to an optimal transfer from one state to another.**

Linear Solutions

These are fundamental problems for both linear and nonlinear systems. For linear systems we have theoretical solutions that are easily implemented numerically. Here is a few.

- **The Linear Quadratic Regulator (LQR) can be used for optimally stabilizing a linear plant.**
- **Linear controllability and the Controllability Gramian lead to an optimal transfer from one state to another.**
- **The Kalman Filter yields an optimal estimate.**

Linear Solutions

These are fundamental problems for both linear and nonlinear systems. For linear systems we have theoretical solutions that are easily implemented numerically. Here is a few.

- The Linear Quadratic Regulator (LQR) can be used for optimally stabilizing a linear plant.
- Linear controllability and the Controllability Gramian lead to an optimal transfer from one state to another.
- The Kalman Filter yields an optimal estimate.
- The Francis theory of regulation can be used to optimally track a reference signal.

Linear Solutions

These are fundamental problems for both linear and nonlinear systems. For linear systems we have theoretical solutions that are easily implemented numerically. Here is a few.

- **The Linear Quadratic Regulator (LQR) can be used for optimally stabilizing a linear plant.**
- **Linear controllability and the Controllability Gramian lead to an optimal transfer from one state to another.**
- **The Kalman Filter yields an optimal estimate.**
- **The Francis theory of regulation can be used to optimally track a reference signal.**

Linear Solutions

These are fundamental problems for both linear and nonlinear systems. For linear systems we have theoretical solutions that are easily implemented numerically. Here is a few.

- **The Linear Quadratic Regulator (LQR) can be used for optimally stabilizing a linear plant.**
- **Linear controllability and the Controllability Gramian lead to an optimal transfer from one state to another.**
- **The Kalman Filter yields an optimal estimate.**
- **The Francis theory of regulation can be used to optimally track a reference signal.**

Notice the repeated use of the word "optimal".

Linear Solutions

These are fundamental problems for both linear and nonlinear systems. For linear systems we have theoretical solutions that are easily implemented numerically. Here is a few.

- **The Linear Quadratic Regulator (LQR) can be used for optimally stabilizing a linear plant.**
- **Linear controllability and the Controllability Gramian lead to an optimal transfer from one state to another.**
- **The Kalman Filter yields an optimal estimate.**
- **The Francis theory of regulation can be used to optimally track a reference signal.**

Notice the repeated use of the word "optimal".

It can be difficult to solve a problem that has many solutions. Introducing an optimality criterion narrows the search.

Optimal Stabilization

An operating point (equilibrium point) is a pair x^e, u^e such that $f(x^e, u^e) = 0$.

Optimal Stabilization

An operating point (equilibrium point) is a pair x^e, u^e such that $f(x^e, u^e) = 0$.

Without loss of generalitity we can assume $x^e = 0, u^e = 0$.

Optimal Stabilization

An operating point (equilibrium point) is a pair x^e, u^e such that $f(x^e, u^e) = 0$.

Without loss of generalitity we can assume $x^e = 0, u^e = 0$.

A classic way to find a feedback $u = \kappa(x)$ that stabilizes a nonlinear system to an operating point is to pose and solve an infinite horizon optimal control problem. Choose a Lagrangian $l(x, u)$ that is nonnegative definite in x, u and positve definite in u and

$$\min_{u(\cdot)} \int_0^\infty l(x, u) \; dt$$

subject to

$$\begin{aligned} \dot{x} &= f(x, u) \\ x(0) &= x^0 \end{aligned}$$

Optimal Stabilization

The reason is the optimal solution is given by a feedback $u(t) = \kappa(x(t))$ and the optimal cost $\pi(x^0) \geq 0$ starting at x^0 is a potential Lyapunov function for the closed loop system

$$\frac{d}{dt}\pi(x(t)) \;\; = \;\; -l(x(t), \kappa(x(t))) \leq 0$$

that can be used to verify stability.

Optimal Stabilization

The reason is the optimal solution is given by a feedback $u(t) = \kappa(x(t))$ and the optimal cost $\pi(x^0) \geq 0$ starting at x^0 is a potential Lyapunov function for the closed loop system

$$\frac{d}{dt}\pi(x(t)) \;\; = \;\; -l(x(t), \kappa(x(t))) \leq 0$$

that can be used to verify stability.

If we only have approximations $\pi(x), \kappa(x)$ to the true solutions then we can verify local asymptotic stability on the largest sublevel set on which the standard Lyapunov equations hold.

Optimal Stabilization

The reason is the optimal solution is given by a feedback $u(t) = \kappa(x(t))$ and the optimal cost $\pi(x^0) \geq 0$ starting at x^0 is a potential Lyapunov function for the closed loop system

$$\frac{d}{dt}\pi(x(t)) \;=\; -l(x(t), \kappa(x(t))) \leq 0$$

that can be used to verify stability.

If we only have approximations $\pi(x), \kappa(x)$ to the true solutions then we can verify local asymptotic stability on the largest sublevel set on which the standard Lyapunov equations hold.

If $x \neq 0$ and $\pi(x) \leq c$

$$\begin{aligned} 0 &< \pi(x) \\ 0 &> \frac{\partial \pi}{\partial x}(x) f(x, \kappa(x)) \end{aligned}$$

HJB Equations

If there exist smooth solutions $\pi(x), \kappa(x)$ to the Hamilton-Jacobi-Bellman equations

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

then these are the optimal cost and optimal feedback.

HJB Equations

If there exist smooth solutions $\pi(x), \kappa(x)$ to the Hamilton-Jacobi-Bellman equations

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

then these are the optimal cost and optimal feedback.

The control Hamiltonian is

$$\mathcal{H}(\lambda, x, u) = \lambda' f(x,u) + l(x,u)$$

HJB Equations

If the control Hamiltonian is strictly convex in u for every λ, x then the HJB equations simplify to

$$
\begin{aligned}
0 &= \frac{\partial \pi}{\partial x}(x) f(x, \kappa(x)) + l(x, \kappa(x)) \\
0 &= \frac{\partial \pi}{\partial x}(x) \frac{\partial f}{\partial u}(x, \kappa(x)) + \frac{\partial l}{\partial u}(x, \kappa(x))
\end{aligned}
$$

HJB Equations

If the control Hamiltonian is strictly convex in u for every λ, x then the HJB equations simplify to

$$\begin{aligned} 0 &= \frac{\partial \pi}{\partial x}(x) f(x, \kappa(x)) + l(x, \kappa(x)) \\ 0 &= \frac{\partial \pi}{\partial x}(x) \frac{\partial f}{\partial u}(x, \kappa(x)) + \frac{\partial l}{\partial u}(x, \kappa(x)) \end{aligned}$$

If $R(x) > 0$ and

$$\begin{aligned} f(x, u) &= f_0(x) + f_1(x) u \\ l(x, u) &= q(x) + \frac{1}{2} u' R(x) u \end{aligned}$$

then $\mathcal{H}(\lambda, x, u)$ is strictly convex in u for every λ, x.

Linear Quadratic Regulator

About the only time the HJB equations can be solved in closed form is the so-called Linear Quadratic Regulator (LQR)

$$\min_{u(\cdot)} \frac{1}{2} \int_0^\infty x'Qx + 2x'Su + u'Ru \; dt$$

subject to

$$\dot{x} = Fx + Gu$$

Linear Quadratic Regulator

About the only time the HJB equations can be solved in closed form is the so-called Linear Quadratic Regulator (LQR)

$$\min_{u(\cdot)} \frac{1}{2}\int_0^\infty x'Qx + 2x'Su + u'Ru \; dt$$

subject to

$$\dot{x} \;=\; Fx + Gu$$

The optimal cost is $\pi(x) = \frac{1}{2}x'Px$ and optimal feedback is $u = Kx$ where P, K satisfy

$$\begin{aligned} 0 &= F'P + PF + Q - (PG+S)R^{-1}(PG+S)' \\ K &= -R^{-1}(PG+S)' \end{aligned}$$

Jacobian Linearization

Standard engineering practice is to approximate the nonlinear dynamics

$$\dot{x} \;=\; f(x,u)$$

by its Jacobian linearization

$$\dot{x} \;=\; Fx + Gu$$

where

$$F = \frac{\partial f}{\partial x}(0,0), \qquad G = \frac{\partial f}{\partial u}(0,0)$$

Jacobian Linearization

Standard engineering practice is to approximate the nonlinear dynamics

$$\dot{x} = f(x,u)$$

by its Jacobian linearization

$$\dot{x} = Fx + Gu$$

where

$$F = \frac{\partial f}{\partial x}(0,0), \qquad G = \frac{\partial f}{\partial u}(0,0)$$

Then $\frac{1}{2}x'Px$ and $u = Kx$ are approximations to the true optimal cost $\pi(x)$ and optimal feedback $u = \kappa(x)$.

Lyapunov Argument

If

$$f(x,u) = Fx + Gu + O(x,u)^2$$

then

$$\begin{aligned}\frac{d}{dt}\left(x'(t)Px(t)\right) &= -x'(t)\left(Q + (PG+S)R^{-1}(PG+S)'\right)x(t) \\ &\quad + O(x(t))^3\end{aligned}$$

so on some neighborhood of $x = 0$ the feedback $u = Kx$ stabilizes the nonlinear system.

Lyapunov Argument

If

$$f(x,u) \quad = \quad Fx + Gu + O(x,u)^2$$

then

$$\begin{aligned} \frac{d}{dt}\left(x'(t)Px(t)\right) \quad &= \quad -x'(t)\left(Q + (PG+S)R^{-1}(PG+S)'\right)x(t) \\ &\quad +O(x(t))^3 \end{aligned}$$

so on some neighborhood of $x = 0$ the feedback $u = Kx$ stabilizes the nonlinear system.

Kicker: How big is the neighborhood?

Al'brekht's Method

In 1961 Ernst Al'brekht noticed that first HJB equation

$$0 \;=\; \frac{\partial \pi}{\partial x}(x) f(x, \kappa(x)) + l(x, \kappa(x))$$

is singular at the operating point $x = 0, u = 0$ because $f(0, 0) = 0$.

Al'brekht's Method

In 1961 Ernst Al'brekht noticed that first HJB equation

$$0 = \frac{\partial \pi}{\partial x}(x) f(x, \kappa(x)) + l(x, \kappa(x))$$

is singular at the operating point $x = 0, u = 0$ because $f(0, 0) = 0$.

This means that it is amenable to power series methods. Let $\pi^{[k]}(x)$ denote a homogeneous polynomial of degree k and consider the linear operator

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

Al'brekht's Method

In 1961 Ernst Al'brekht noticed that first HJB equation

$$0 \;=\; \frac{\partial \pi}{\partial x}(x) f(x, \kappa(x)) + l(x, \kappa(x))$$

is singular at the operating point $x = 0, u = 0$ because $f(0,0) = 0$.

This means that it is amenable to power series methods. Let $\pi^{[k]}(x)$ denote a homogeneous polynomial of degree k and consider the linear operator

$$\pi^{[k]}(x) \;\mapsto\; \frac{\partial \pi^{[k]}}{\partial x}(x)(F + GK)x$$

The eigenvalues of this operator are the sums of k eigenvalues of $F + GK$ so if the latter are all in the open left half plane than so are the former.

Al'brekht's Method

Al'brekht plugged the Taylor polynomial expansions

$$
\begin{aligned}
f(x,u) &\approx Fx + Gu + f^{[2]}(x,u) + \ldots + f^{[d]}(x,u) \\
l(x,u) &\approx \frac{1}{2}\left(x'Qx + 2x'Su + u'Ru\right) + \ldots + l^{[d+1]}(x,u) \\
\pi(x) &\approx \frac{1}{2}x'Px + \pi^{[3]}(x) + \ldots + \pi^{[d+1]}(x) \\
\kappa(x) &\approx Kx + \kappa^{[2]}(x) + \ldots + \kappa^{[d]}(x,u)
\end{aligned}
$$

into the HJB equations and collected terms of like degrees.

Al'brekht's Method

At the lowest level he obtained the familiar LQR equations

$$
\begin{aligned}
\mathbf{0} &= F'P + PF + Q - (PG+S)R^{-1}(PG+S)' \\
K &= -R^{-1}(PG+S)'
\end{aligned}
$$

Al'brekht's Method

At the lowest level he obtained the familiar LQR equations

$$\begin{aligned} 0 &= F'P + PF + Q - (PG+S)R^{-1}(PG+S)' \\ K &= -R^{-1}(PG+S)' \end{aligned}$$

At the next level he obtained linear equations for the coefficients of $\pi^{[3]}(x)$ and $\kappa^{[2]}(x)$.

Al'brekht's Method

At the lowest level he obtained the familiar LQR equations

$$\begin{aligned} 0 &= F'P + PF + Q - (PG+S)R^{-1}(PG+S)' \\ K &= -R^{-1}(PG+S)' \end{aligned}$$

At the next level he obtained linear equations for the coefficients of $\pi^{[3]}(x)$ and $\kappa^{[2]}(x)$.

These linear equations are solvable if the standard LQR conditions hold.

Al'brekht's Method

At the lowest level he obtained the familiar LQR equations

$$\begin{aligned} 0 &= F'P + PF + Q - (PG+S)R^{-1}(PG+S)' \\ K &= -R^{-1}(PG+S)' \end{aligned}$$

At the next level he obtained linear equations for the coefficients of $\pi^{[3]}(x)$ and $\kappa^{[2]}(x)$.

These linear equations are solvable if the standard LQR conditions hold.

There are similar linear equations for the higher degree terms, $\pi^{[k+1]}(x)$ and $\kappa^{[k]}(x)$.

The routine `hjb.m` in my Nonlinear Systems Toolbox can implement Al'brekht's method in any state and control dimensions to any degree subject to machine limitations.

The routine `hjb.m` in my Nonlinear Systems Toolbox can implement Al'brekht's method in any state and control dimensions to any degree subject to machine limitations.

The routine `hjb_set_up.m` converts symbolic expressions for $f(x,u)$ and $l(x,u)$ into the matrices of coefficients that `hjb.m` needs.

The routine `hjb.m` in my Nonlinear Systems Toolbox can implement Al'brekht's method in any state and control dimensions to any degree subject to machine limitations.

The routine `hjb_set_up.m` converts symbolic expressions for $f(x,u)$ and $l(x,u)$ into the matrices of coefficients that `hjb.m` needs.

Here is an example.

Optimal Stabilization of a Rigid Body

The example was fairly simple with $n = 4$ states and $m = 2$.

Optimal Stabilization of a Rigid Body

The example was fairly simple with $n = 4$ states and $m = 2$.

What about a more challenging problem?

Optimal Stabilization of a Rigid Body

The example was fairly simple with $n = 4$ states and $m = 2$.

What about a more challenging problem?

Consider rigid body with six degrees of freedom, a boat, a plane a satellite. There are three translational degrees of freedom and three angular degrees of freedom.

Optimal Stabilization of a Rigid Body

The example was fairly simple with $n = 4$ states and $m = 2$.

What about a more challenging problem?

Consider rigid body with six degrees of freedom, a boat, a plane a satellite. There are three translational degrees of freedom and three angular degrees of freedom.

A state space model has $n = 12$ dimensions with associated velocities.

Optimal Stabilization of a Rigid Body

The example was fairly simple with $n = 4$ states and $m = 2$.

What about a more challenging problem?

Consider rigid body with six degrees of freedom, a boat, a plane a satellite. There are three translational degrees of freedom and three angular degrees of freedom.

A state space model has $n = 12$ dimensions with associated velocities.

Suppose the body is fully actuated, then the control dimension is $m = 6$.

Optimal Stabilization of a Rigid Body

The example was fairly simple with $n = 4$ states and $m = 2$.

What about a more challenging problem?

Consider rigid body with six degrees of freedom, a boat, a plane a satellite. There are three translational degrees of freedom and three angular degrees of freedom.

A state space model has $n = 12$ dimensions with associated velocities.

Suppose the body is fully actuated, then the control dimension is $m = 6$.

Our goal is to optimally stabilize the body to a desired constant velocity and desired attitude.

Runtimes

First we did it to degree 4 in the cost and degree 3 in the feedback.

Runtimes

First we did it to degree 4 in the cost and degree 3 in the feedback.

hjb_set_up_time $= 89.7025$ seconds

Runtimes

First we did it to degree 4 in the cost and degree 3 in the feedback.

hjb_set_up_time $= 89.7025$ seconds

Solving the HJB equations of degree 1
Solving the HJB equations of degree 2
Solving the HJB equations of degree 3

Runtimes

First we did it to degree 4 in the cost and degree 3 in the feedback.

hjb_set_up_time = 89.7025 seconds

Solving the HJB equations of degree 1
Solving the HJB equations of degree 2
Solving the HJB equations of degree 3

hjb_time_3 = 4.0260 seconds

Runtimes

Next we did it to degree $\mathbf{6}$ in the cost and degree $\mathbf{4}$ in the feedback.

Runtimes

Next we did it to degree 6 in the cost and degree 4 in the feedback.

Unfortunately Matlab's symbolic differentiation routine was not up to the task of computing Taylor polynomials of degree $5, 6$.

Runtimes

Next we did it to degree $\mathbf{6}$ in the cost and degree $\mathbf{4}$ in the feedback.

Unfortunately Matlab's symbolic differentiation routine was not up to the task of computing Taylor polynomials of degree $\mathbf{5, 6}$.

So `hjb_set_up.m` **did not run.**

Runtimes

Next we did it to degree 6 in the cost and degree 4 in the feedback.

Unfortunately Matlab's symbolic differentiation routine was not up to the task of computing Taylor polynomials of degree $5, 6$.

So `hjb_set_up.m` **did not run.**

By padding f and l with random coefficients we can test `hjb.m`

Runtimes

Next we did it to degree 6 in the cost and degree 4 in the feedback.

Unfortunately Matlab's symbolic differentiation routine was not up to the task of computing Taylor polynomials of degree $5, 6$.

So `hjb_set_up.m` **did not run.**

By padding f and l with random coefficients we can test `hjb.m`

Solving the HJB equations of degree 1
Solving the HJB equations of degree 2
Solving the HJB equations of degree 3
Solving the HJB equations of degree 4
Solving the HJB equations of degree 5

Runtimes

Next we did it to degree 6 in the cost and degree 4 in the feedback.

Unfortunately Matlab's symbolic differentiation routine was not up to the task of computing Taylor polynomials of degree $5, 6$.

So `hjb_set_up.m` did not run.

By padding f and l with random coefficients we can test `hjb.m`

Solving the HJB equations of degree 1
Solving the HJB equations of degree 2
Solving the HJB equations of degree 3
Solving the HJB equations of degree 4
Solving the HJB equations of degree 5

hjb_time_5 $= 1.6048e+03$

Runtimes

Next we did it to degree 6 in the cost and degree 4 in the feedback.

Unfortunately Matlab's symbolic differentiation routine was not up to the task of computing Taylor polynomials of degree $5, 6$.

So `hjb_set_up.m` **did not run.**

By padding f and l with random coefficients we can test `hjb.m`

Solving the HJB equations of degree 1
Solving the HJB equations of degree 2
Solving the HJB equations of degree 3
Solving the HJB equations of degree 4
Solving the HJB equations of degree 5

hjb_time_5 $= 1.6048e + 03$

This is 26.7466 minutes. The number of monomials of degrees one through five in $n + m = 18$ variables is 33648.

Kicker

So using Al'brekht's method we can get a local solution to the HJB equations for smooth medium sized systems without constraints.

Kicker

So using Al'brekht's method we can get a local solution to the HJB equations for smooth medium sized systems without constraints.

To my knowledge there is no other method that can do this.

Kicker

So using Al'brekht's method we can get a local solution to the HJB equations for smooth medium sized systems without constraints.

To my knowledge there is no other method that can do this.

But

- **How big is the neighborhood of stabilization?**

Kicker

So using Al'brekht's method we can get a local solution to the HJB equations for smooth medium sized systems without constraints.

To my knowledge there is no other method that can do this.

But

- **How big is the neighborhood of stabilization?**
- **What if there are constraints?**

Kicker

So using Al'brekht's method we can get a local solution to the HJB equations for smooth medium sized systems without constraints.

To my knowledge there is no other method that can do this.

But

- **How big is the neighborhood of stabilization?**
- **What if there are constraints?**
- **What if there are discontinuities?**

Model Predictive Control

A variation on Model Predictive Control (MPC) is an answer to the first question.

Model Predictive Control

A variation on Model Predictive Control (MPC) is an answer to the first question.

In MPC we don't try to solve off-line for the optimal cost $\pi(x)$ and optimal feedback $\kappa(x)$ to an infinite horizon optimal control problem.

Model Predictive Control

A variation on Model Predictive Control (MPC) is an answer to the first question.

In MPC we don't try to solve off-line for the optimal cost $\pi(x)$ and optimal feedback $\kappa(x)$ to an infinite horizon optimal control problem.

Instead we solve on-line a discrete time finite horizon optimal control problem for our current state. Suppose $t = t_0$ and $x(t_0) = x^0$, we choose a horizon length T and a Lagrangian $L(x,u)$ and seek to find the optimal control sequence $\mathbf{u}^*_{t_0} = (u^*(t_0), \dots, u^*(t_0 + T - 1))$ that minimizes

$$\sum_{t=t_0}^{t_0+T-1} L(x(t), u(t)) + \Pi(x(t_0 + T))$$

$$x(t+1) = F(x(t), u(t))$$

Model Predictive Control

There is a discrete time version of Al'brekht for the corresonding infinite horizon optimal control problem implemented in `dpe.m` **that yields Taylor polynomials for the infinite time optimal cost $\mathbf{\Pi}(x)$ and optimal feedback $\mathcal{K}(x)$.**

Model Predictive Control

There is a discrete time version of Al'brekht for the corresonding infinite horizon optimal control problem implemented in `dpe.m` that yields Taylor polynomials for the infinite time optimal cost $\Pi(x)$ and optimal feedback $\mathcal{K}(x)$.

We use $\Pi(x)$ as our terminal cost for the discrete time finite horizon optimal control problem for then the finite horizon and infinite horizon costs are the same.

Model Predictive Control

There is a discrete time version of Al'brekht for the corresonding infinite horizon optimal control problem implemented in `dpe.m` that yields Taylor polynomials for the infinite time optimal cost $\Pi(x)$ and optimal feedback $\mathcal{K}(x)$.

We use $\Pi(x)$ as our terminal cost for the discrete time finite horizon optimal control problem for then the finite horizon and infinite horizon costs are the same.

The discrete time finite horizon optimal control problem is a nonlinear program which we pass to a fast solver like KNITRO or IPOPT. The solver returns $\mathbf{u}^*_{t_0}$, we implement the feedback $u(t_0) = u^*(t_0)$ and move one time step. We increment t_0 by one which slides the horizon forward one time step and repeat the process.

Adaptive Horizon Model Predictive Control

But how do we know that the horizon T is long enough so that the endpoint $x(t_0 + T)$ is in the domain where $\Pi(x)$ is a valid Lyapunov function for closed loop dynamics under the feedback $u = \mathcal{K}(x)$?

Adaptive Horizon Model Predictive Control

But how do we know that the horizon T is long enough so that the endpoint $x(t_0 + T)$ is in the domain where $\Pi(x)$ is a valid Lyapunov function for closed loop dynamics under the feedback $u = \mathcal{K}(x)$?

We compute the extension the state trajectory from $x(t_0 + T)$ an additional S times steps using the closed loop dynamics under the feedback $u = \mathcal{K}(x)$.

Adaptive Horizon Model Predictive Control

But how do we know that the horizon T is long enough so that the endpoint $x(t_0 + T)$ is in the domain where $\Pi(x)$ is a valid Lyapunov function for closed loop dynamics under the feedback $u = \mathcal{K}(x)$?

We compute the extension the state trajectory from $x(t_0 + T)$ an additional S times steps using the closed loop dynamics under the feedback $u = \mathcal{K}(x)$.

As we do so we check that the Lyapunov conditions hold on the extension. If so we infer that we are stabilizing.

Adaptive Horizon Model Predictive Control

But how do we know that the horizon T is long enough so that the endpoint $x(t_0+T)$ is in the domain where $\Pi(x)$ is a valid Lyapunov function for closed loop dynamics under the feedback $u = \mathcal{K}(x)$?

We compute the extension the state trajectory from $x(t_0+T)$ an additional S times steps using the closed loop dynamics under the feedback $u = \mathcal{K}(x)$.

As we do so we check that the Lyapunov conditions hold on the extension. If so we infer that we are stabilizing.

If they are comfortably satisfied we might decrease the horizon.

Adaptive Horizon Model Predictive Control

But how do we know that the horizon T is long enough so that the endpoint $x(t_0+T)$ is in the domain where $\Pi(x)$ is a valid Lyapunov function for closed loop dynamics under the feedback $u=\mathcal{K}(x)$?

We compute the extension the state trajectory from $x(t_0+T)$ an additional S times steps using the closed loop dynamics under the feedback $u=\mathcal{K}(x)$.

As we do so we check that the Lyapunov conditions hold on the extension. If so we infer that we are stabilizing.

If they are comfortably satisfied we might decrease the horizon.

If the are not satisfied we increase the horizon.

Adaptive Horizon Model Predictive Control

If there are constraints $G(x, u) \leq 0$ we can take the same approach as long as they are not active at the operating point $G(0, 0) < 0$.

Adaptive Horizon Model Predictive Control

If there are constraints $G(x,u) \le 0$ we can take the same approach as long as they are not active at the operating point $G(0,0) < 0$.

We pass these constraints to the solver which hopefully can handle them.

Adaptive Horizon Model Predictive Control

If there are constraints $G(x,u) \leq 0$ we can take the same approach as long as they are not active at the operating point $G(0,0) < 0$.

We pass these constraints to the solver which hopefully can handle them.

And we check that the constraints and the Lyapunov conditions are satisfied on the computed extension.

Examples

We seek to stabilize a double pendulum to the upright position using torques at each of the pivots.

Examples

We seek to stabilize a double pendulum to the upright position using torques at each of the pivots.

x_1 **is the angle of the first leg measured in radians counter-clockwise from straight up and** $x_3 = \dot{x}_1$**.**

Examples

We seek to stabilize a double pendulum to the upright position using torques at each of the pivots.

x_1 **is the angle of the first leg measured in radians counter-clockwise from straight up and** $x_3 = \dot{x}_1$**.**
x_2 **is the angle of the second leg measured in radians counter-clockwise from straight up and** $x_4 = \dot{x}_2$**.**

Examples

We seek to stabilize a double pendulum to the upright position using torques at each of the pivots.

x_1 **is the angle of the first leg measured in radians counter-clockwise from straight up and** $x_3 = \dot{x}_1$**.**
x_2 **is the angle of the second leg measured in radians counter-clockwise from straight up and** $x_4 = \dot{x}_2$**.**
u_1 **is the torque applied at the base of the first leg.**

Examples

We seek to stabilize a double pendulum to the upright position using torques at each of the pivots.

x_1 **is the angle of the first leg measured in radians counter-clockwise from straight up and** $x_3 = \dot{x}_1$.

x_2 **is the angle of the second leg measured in radians counter-clockwise from straight up and** $x_4 = \dot{x}_2$.

u_1 **is the torque applied at the base of the first leg.**

u_2 **is the torque applied at the joint between the legs.**

Examples

We seek to stabilize a double pendulum to the upright position using torques at each of the pivots.

x_1 **is the angle of the first leg measured in radians counter-clockwise from straight up and** $x_3 = \dot{x}_1$.
x_2 **is the angle of the second leg measured in radians counter-clockwise from straight up and** $x_4 = \dot{x}_2$.
u_1 **is the torque applied at the base of the first leg.**
u_2 **is the torque applied at the joint between the legs.**
The length of the first massless leg is 1 **m.**

Examples

We seek to stabilize a double pendulum to the upright position using torques at each of the pivots.

x_1 is the angle of the first leg measured in radians counter-clockwise from straight up and $x_3 = \dot{x}_1$.
x_2 is the angle of the second leg measured in radians counter-clockwise from straight up and $x_4 = \dot{x}_2$.
u_1 is the torque applied at the base of the first leg.
u_2 is the torque applied at the joint between the legs.
The length of the first massless leg is 1 m.
The length of the second massless leg is 2 m.

Examples

We seek to stabilize a double pendulum to the upright position using torques at each of the pivots.

x_1 is the angle of the first leg measured in radians counter-clockwise from straight up and $x_3 = \dot{x}_1$.
x_2 is the angle of the second leg measured in radians counter-clockwise from straight up and $x_4 = \dot{x}_2$.
u_1 is the torque applied at the base of the first leg.
u_2 is the torque applied at the joint between the legs.
The length of the first massless leg is 1 m.
The length of the second massless leg is 2 m.
The mass at the joint between the legs is 2 kg.

Examples

We seek to stabilize a double pendulum to the upright position using torques at each of the pivots.

x_1 is the angle of the first leg measured in radians counter-clockwise from straight up and $x_3 = \dot{x}_1$.
x_2 is the angle of the second leg measured in radians counter-clockwise from straight up and $x_4 = \dot{x}_2$.
u_1 is the torque applied at the base of the first leg.
u_2 is the torque applied at the joint between the legs.
The length of the first massless leg is 1 m.
The length of the second massless leg is 2 m.
The mass at the joint between the legs is 2 kg.
The mass at the tip of the second leg is 1 kg.

Examples

We seek to stabilize a double pendulum to the upright position using torques at each of the pivots.

x_1 **is the angle of the first leg measured in radians counter-clockwise from straight up and** $x_3 = \dot{x}_1$**.**
x_2 **is the angle of the second leg measured in radians counter-clockwise from straight up and** $x_4 = \dot{x}_2$**.**
u_1 **is the torque applied at the base of the first leg.**
u_2 **is the torque applied at the joint between the legs.**
The length of the first massless leg is 1 **m.**
The length of the second massless leg is 2 **m.**
The mass at the joint between the legs is 2 **kg.**
The mass at the tip of the second leg is 1 **kg.**
The damping coefficients at the two joints are both 0.5 **s**$^{-1}$ **.**

Examples

We seek to stabilize a double pendulum to the upright position using torques at each of the pivots.

x_1 is the angle of the first leg measured in radians counter-clockwise from straight up and $x_3 = \dot{x}_1$.
x_2 is the angle of the second leg measured in radians counter-clockwise from straight up and $x_4 = \dot{x}_2$.
u_1 is the torque applied at the base of the first leg.
u_2 is the torque applied at the joint between the legs.
The length of the first massless leg is 1 m.
The length of the second massless leg is 2 m.
The mass at the joint between the legs is 2 kg.
The mass at the tip of the second leg is 1 kg.
The damping coefficients at the two joints are both 0.5 s^{-1} .

The continuous time dynamics is discretized using Euler's method with time step 0.1 s. assuming the control is constant throughout the time step.

Examples

The continuous time Lagrangian is $l_c(x, u) = (|x|^2 + |u|^2)/2$.

Examples

The continuous time Lagrangian is $l_c(x,u) = (|x|^2 + |u|^2)/2.$

Its Euler discretization is $l(x,u) = (|x|^2 + |u|^2)/20.$

Examples

The continuous time Lagrangian is $l_c(x, u) = (|x|^2 + |u|^2)/2$.

Its Euler discretization is $l(x, u) = (|x|^2 + |u|^2)/20$.

The initial state is $x = (\pi/2, -\pi/2, 0, 0)'$.

Examples

The continuous time Lagrangian is $l_c(x, u) = (|x|^2 + |u|^2)/2$.

Its Euler discretization is $l(x, u) = (|x|^2 + |u|^2)/20$.

The initial state is $x = (\pi/2, -\pi/2, 0, 0)'$.

The initial horizon length is $N = 5$.

Examples

The continuous time Lagrangian is $l_c(x, u) = (|x|^2 + |u|^2)/2$.

Its Euler discretization is $l(x, u) = (|x|^2 + |u|^2)/20$.

The initial state is $x = (\pi/2, -\pi/2, 0, 0)'$.

The initial horizon length is $N = 5$.

The terminal cost and feedback $V_f(x),\ \kappa_f(x)$ are the solution of the discrete time, infinite horizon LQR problem using the linear part of the dynamics at the origin and the quadratic Lagrangian.

Examples

The continuous time Lagrangian is $l_c(x,u) = (|x|^2 + |u|^2)/2$.

Its Euler discretization is $l(x,u) = (|x|^2 + |u|^2)/20$.

The initial state is $x = (\pi/2, -\pi/2, 0, 0)'$.

The initial horizon length is $N = 5$.

The terminal cost and feedback $V_f(x)$, $\kappa_f(x)$ are the solution of the discrete time, infinite horizon LQR problem using the linear part of the dynamics at the origin and the quadratic Lagrangian.

The class K_∞ function is $\alpha(|x|) = 0.1|x|^2$.

Examples

The continuous time Lagrangian is $l_c(x,u) = (|x|^2 + |u|^2)/2$.

Its Euler discretization is $l(x,u) = (|x|^2 + |u|^2)/20$.

The initial state is $x = (\pi/2, -\pi/2, 0, 0)'$.

The initial horizon length is $N = 5$.

The terminal cost and feedback $V_f(x)$, $\kappa_f(x)$ are the solution of the discrete time, infinite horizon LQR problem using the linear part of the dynamics at the origin and the quadratic Lagrangian.

The class K_∞ function is $\alpha(|x|) = 0.1|x|^2$.

The extended horizon is kept frozen at $L = 5$.

Examples

The continuous time Lagrangian is $l_c(x,u) = (|x|^2 + |u|^2)/2$.

Its Euler discretization is $l(x,u) = (|x|^2 + |u|^2)/20$.

The initial state is $x = (\pi/2, -\pi/2, 0, 0)'$.

The initial horizon length is $N = 5$.

The terminal cost and feedback $V_f(x)$, $\kappa_f(x)$ are the solution of the discrete time, infinite horizon LQR problem using the linear part of the dynamics at the origin and the quadratic Lagrangian.

The class K_∞ function is $\alpha(|x|) = 0.1|x|^2$.

The extended horizon is kept frozen at $L = 5$.

We do not move one time step forward if the feasibility and Lyapunov conditions do not hold over the extended state trajectory but instead we increased N by 1 and recomputed from the same x.

Example 1

$$x^0 \;=\; (0.5\pi, -0.5\pi, 0, 0)'$$

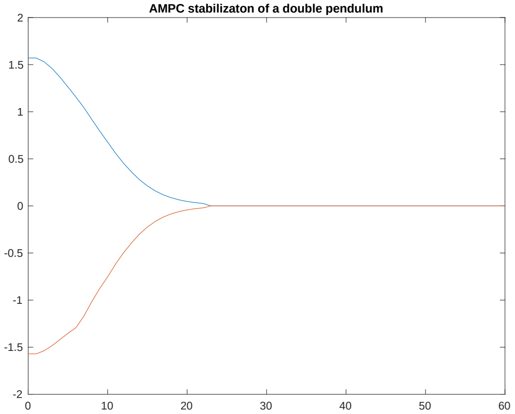


Figure: Angles Converging to the Vertical

Example 1

$$x^0 = (0.5\pi, -0.5\pi, 0, 0)'$$

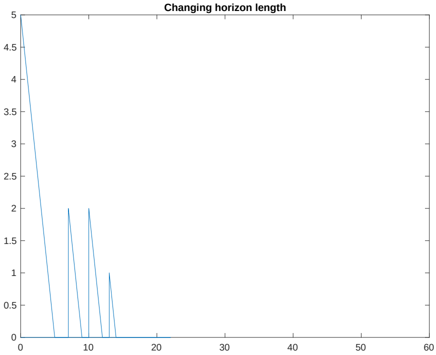


Figure: Adaptively Changing Horizon

Example 1

$$x^0 = (0.5\pi, -0.5\pi, 0, 0)'$$

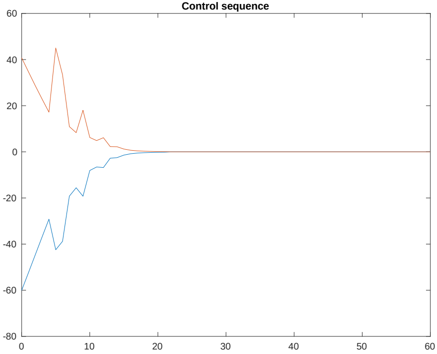


Figure: Control sequence

Example 2

$$x^0 \;=\; (0.9\pi, -0.9\pi, 0, 0)'$$

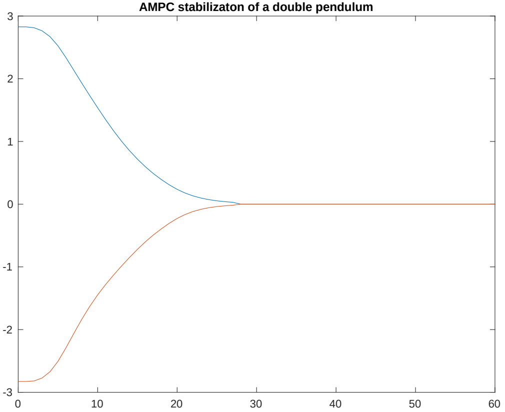


Figure: Angles Converging to the Vertical

Example 2

$$x^0 \;=\; (0.9\pi, -0.9\pi, 0, 0)'$$

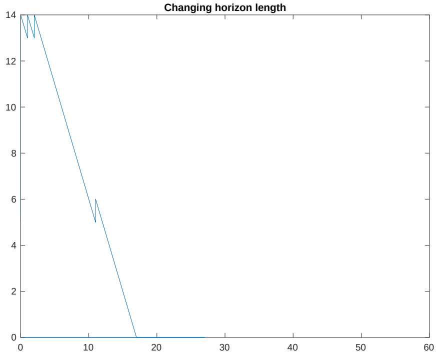


Figure: Adaptively Changing Horizon

Example 2

$$x^0 = (0.9\pi, -0.9\pi, 0, 0)'$$

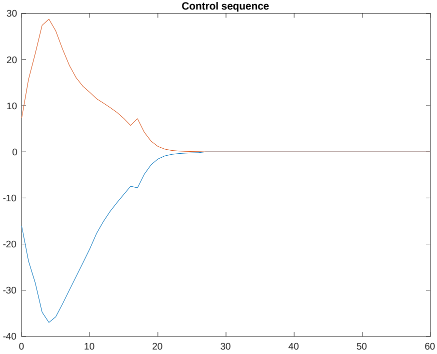


Figure: Control sequence

Example 3

$$x^0 = (0.9\pi, 0.9\pi, 0, 0)'$$

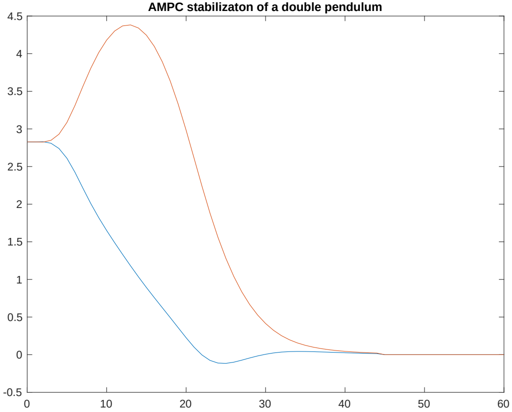


Figure: Angles Converging to the Vertical

Example 3

$$x^0 = (0.9\pi, 0.9\pi, 0, 0)'$$

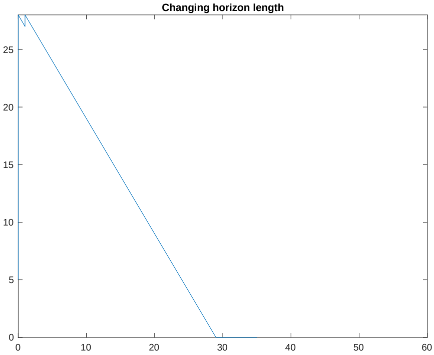


Figure: Adaptively Changing Horizon

Example 3

$$x^0 = (0.9\pi, 0.9\pi, 0, 0)'$$

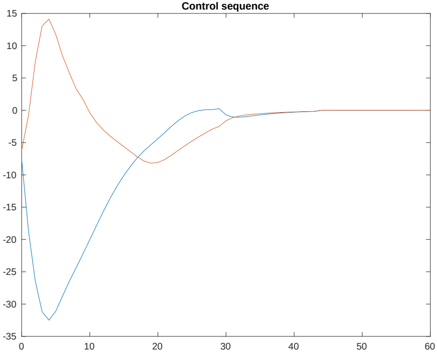
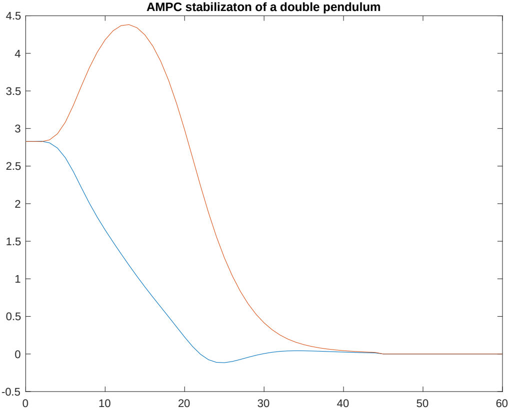


Figure: Control sequence

Example 4

$$
\begin{aligned}
x^0 &= (0.9\pi, 0.9\pi, 0, 0)' \\
|u_1| &\leq 10 \\
|u_2| &\leq 10
\end{aligned}
$$



Example 4

$$
\begin{aligned}
x^0 &= (0.9\pi, 0.9\pi, 0, 0)' \\
|u_1| &\leq 10 \\
|u_2| &< 10
\end{aligned}
$$

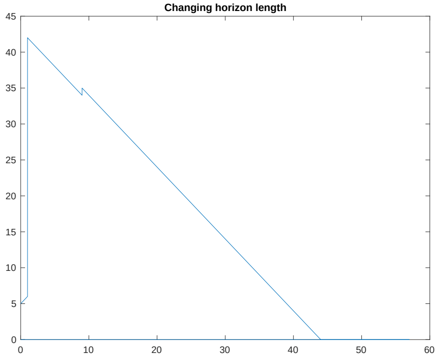


Figure: Adaptively Changing Horizon

Example 4

$$\begin{aligned} x^0 &= (0.9\pi, 0.9\pi, 0, 0)' \\ |u_1| &\leq 10 \\ |u_2| &< 10 \end{aligned}$$

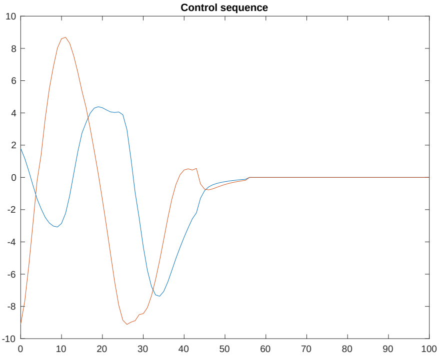


Figure: Control sequence

Our Slogan

Think Mathematically

Our Slogan

Think Mathematically

Act Computationally

Conclusion

Thank You