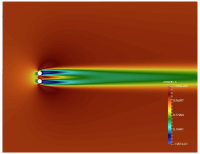


Software for Nonlinear Balancing and Control

Background and Demonstrations

Jeff Borggaard

Virginia Tech

Nonlinear Model Reduction for Control, May 2023

This work was supported in part by the National Science Foundation (DMS-1819110,CMMI-2130695)

Outline

Introduction

Optimal Control Problem

Find a control $\mathbf{u}(\cdot)$ with $\mathbf{u}(t) \in \mathbb{R}^m$ that solves

$$\min_{\mathbf{u}} J(\mathbf{z}, \mathbf{u}) = \int_0^\infty \ell(\mathbf{z}(s), \mathbf{u}(s)) \; ds$$

subject to

$$\dot{\mathbf{z}}(t) = \mathbf{f}(\mathbf{z}(t), \mathbf{u}(t)), \qquad \mathbf{z}(0) = \mathbf{z}_0 \in \mathbb{R}^n.$$

Optimal Control Problem

Find a control $\mathbf{u}(\cdot)$ with $\mathbf{u}(t) \in \mathbb{R}^m$ that solves

$$\min_{\mathbf{u}} J(\mathbf{z}, \mathbf{u}) = \int_0^\infty \ell(\mathbf{z}(s), \mathbf{u}(s))\ ds$$

subject to

$$\dot{\mathbf{z}}(t) = \mathbf{f}(\mathbf{z}(t), \mathbf{u}(t)), \qquad \mathbf{z}(0) = \mathbf{z}_0 \in \mathbb{R}^n.$$

Assume the optimal control is given by $\mathbf{u}_*(t) = \mathcal{K}(\mathbf{z}_*(t))$, and define the value function as

$$v(\mathbf{z}_0) = J(\mathbf{z}_*(\cdot; \mathbf{z}_0), \mathbf{u}_*(\cdot)).$$

Optimal Control Problem

Find a control $\mathbf{u}(\cdot)$ with $\mathbf{u}(t) \in \mathbb{R}^m$ that solves

$$\min_{\mathbf{u}} J(\mathbf{z}, \mathbf{u}) = \int_0^\infty \ell(\mathbf{z}(s), \mathbf{u}(s))\ ds$$

subject to

$$\dot{\mathbf{z}}(t) = \mathbf{f}(\mathbf{z}(t), \mathbf{u}(t)), \qquad \mathbf{z}(0) = \mathbf{z}_0 \in \mathbb{R}^n.$$

Assume the optimal control is given by $\mathbf{u}_*(t) = \mathcal{K}(\mathbf{z}_*(t))$, and define the value function as

$$v(\mathbf{z}_0) = J(\mathbf{z}_*(\cdot; \mathbf{z}_0), \mathbf{u}_*(\cdot)).$$

For $\mathbf{f}$, ℓ, and v smooth enough, the feedback relation satisfies the Hamilton-Jacobi-Bellman partial differential equations

$$\begin{aligned} 0 &= \frac{\partial v}{\partial \mathbf{z}}(\mathbf{z})\mathbf{f}(\mathbf{z}, \mathcal{K}(\mathbf{z})) + \ell(\mathbf{z}, \mathcal{K}(\mathbf{z})) \\ \mathbf{0} &= \frac{\partial v}{\partial \mathbf{z}}(\mathbf{z})\frac{\partial \mathbf{f}}{\partial \mathbf{u}}(\mathbf{z}, \mathcal{K}(\mathbf{z})) + \frac{\partial \ell}{\partial \mathbf{u}}(\mathbf{z}, \mathcal{K}(\mathbf{z})). \end{aligned}$$

Optimal Control Problem

This can be derived using the *dynamic programming principle*. If $\mathbf{u}_*$ is used, then

$$\begin{aligned} v(\mathbf{z}_0) &= \int_0^\infty \ell(\mathbf{z}_*(s), \mathbf{u}_*(s))\ ds \\ &= \int_0^t \ell(\mathbf{z}_*(s), \mathbf{u}_*(s))\ ds + \underbrace{\int_t^\infty \ell(\mathbf{z}_*(s), \mathbf{u}_*(s))\ ds}_{v(\mathbf{z}_*(t;\mathbf{z}_0))}. \end{aligned}$$

Optimal Control Problem

This can be derived using the *dynamic programming principle*. If $\mathbf{u}_*$ is used, then

$$\begin{aligned} v(\mathbf{z}_0) &= \int_0^\infty \ell(\mathbf{z}_*(s), \mathbf{u}_*(s)) \, ds \\ &= \int_0^t \ell(\mathbf{z}_*(s), \mathbf{u}_*(s)) \, ds + \underbrace{\int_t^\infty \ell(\mathbf{z}_*(s), \mathbf{u}_*(s)) \, ds}_{v(\mathbf{z}_*(t;\mathbf{z}_0))} . \end{aligned}$$

Our smoothness assumptions allow us to differentiate with respect to t. The result is

$$0 = \ell(\mathbf{z}_*(t), \mathbf{u}_*(t)) + \frac{\partial v}{\partial \mathbf{z}}(\mathbf{z}_*(t))\dot{\mathbf{z}}_*(t),$$

which is

$$0 = \ell(\mathbf{z}_*(t), \mathbf{u}_*(t)) + \frac{\partial v}{\partial \mathbf{z}}(\mathbf{z}_*(t))\mathbf{f}(\mathbf{z}_*(t), \mathbf{u}_*(t))$$

or

$$0 = \frac{\partial v}{\partial \mathbf{z}}(\mathbf{z})\mathbf{f}(\mathbf{z}, \mathcal{K}(\mathbf{z})) + \ell(\mathbf{z}, \mathcal{K}(\mathbf{z})).$$

Optimal Control Problem

Ideally, one could solve the HJB equations simultaneously for v and $\mathcal{K}$.

The feedback law $\mathbf{u}(t) = \mathcal{K}(\mathbf{z}(t))$ is the quantity of interest.

The value function $v(\mathbf{z})$ can serve as a Lyapunov function, providing information about the stability region around the steady-state solution $\mathbf{z} = \mathbf{0}$.

However, these are notoriously difficult to solve as the HJB equations are nonlinear PDEs to be solved in $\mathbb{R}^n$ (or after model reduction $\mathbb{R}^r$).

Instead, polynomial approximations are constructed of the form:

$$v(\mathbf{z}) \approx v^{[2]}(\mathbf{z}) + v^{[3]}(\mathbf{z}) + \cdots + v^{[d+1]}(\mathbf{z})$$

and

$$\mathcal{K}(\mathbf{z}) \approx \mathbf{k}^{[1]}(\mathbf{z}) + \mathbf{k}^{[2]}(\mathbf{z}) + \cdots + \mathbf{k}^{[d]}(\mathbf{z}).$$

HJB PDEs

ON THE OPTIMAL STABILIZATION OF NONLINEAR SYSTEMS

(OB OPTIMAL'NOI STABILIZATSII NELINEINYKH SISTEM)

PMM Vol.25, No.5, 1961, pp. 836-844

E. G. AL'BREKHT
(Sverdlovsk)

(Received June 26, 1961)

The Nonlinear Systems Toolbox (Krener, 2015) has a routine `hjb.m` to approximate the feedback relation based on an algorithm by Al'brekht (PMM-Journal of Applied Mathematics and Mechanics, **25**:1254-1266, 1961).

$$0 = \frac{\partial v}{\partial \mathbf{z}}(\mathbf{z})\mathbf{f}(\mathbf{z}, \mathcal{K}(\mathbf{z})) + \ell(\mathbf{z}, \mathcal{K}(\mathbf{z})) \tag{1}$$

$$\mathbf{0} = \frac{\partial v}{\partial \mathbf{z}}(\mathbf{z})\frac{\partial \mathbf{f}}{\partial \mathbf{u}}(\mathbf{z}, \mathcal{K}(\mathbf{z})) + \frac{\partial \ell}{\partial \mathbf{u}}(\mathbf{z}, \mathcal{K}(\mathbf{z})). \tag{2}$$

QQR/PQR Software

- Specializes Al'Brekht's method to polynomial systems with control affine inputs.
- The running cost has a quadratic form: $\ell(\mathbf{z}, \mathbf{u}) = \mathbf{z}^\top \mathbf{Q} \mathbf{z} + \mathbf{u}^\top \mathbf{R} \mathbf{u}$.
- Structured linear systems were obtained by using polynomial approximations in Kronecker product form (as we'll detail below).
- This has been used to perform feedback control approximations for systems with hundreds of states.
- A Matlab implementation is available on github (we'll install and test this later).

Using Kronecker products also has a long history in systems theory

KRONECKER PRODUCTS AND THE SECOND METHOD

OF LYAPUNOV

By

Richard Bellman

P-1097

May 29, 1957

Approved for OTS release

Writing polynomials in Kronecker product form

The Kronecker product of two matrices $\mathbf{X} \in \mathbb{R}^{i_x \times j_x}$ and $\mathbf{Y} \in \mathbb{R}^{i_y \times j_y}$, with entries x_{ij} and y_{ij}, is defined as the block matrix $\mathbf{X} \otimes \mathbf{Y} \in \mathbb{R}^{i_x i_y \times j_x j_y}$ with entries

$$\mathbf{X} \otimes \mathbf{Y} \equiv \begin{bmatrix} x_{11}\mathbf{Y} & x_{12}\mathbf{Y} & \cdots & x_{1j_x}\mathbf{Y} \\ x_{21}\mathbf{Y} & x_{22}\mathbf{Y} & \cdots & x_{2j_x}\mathbf{Y} \\ \vdots & & & \vdots \\ x_{i_x1}\mathbf{Y} & x_{i_x2}\mathbf{Y} & \cdots & x_{i_xj_x}\mathbf{Y} \end{bmatrix} \qquad \mathbf{z} \otimes \mathbf{z} = \begin{bmatrix} z_1\mathbf{z} \\ z_2\mathbf{z} \\ \vdots \\ z_n\mathbf{z} \end{bmatrix}.$$

The following properties are useful:

- $(\mathbf{C} \otimes \mathbf{D})(\mathbf{E} \otimes \mathbf{F}) = (\mathbf{CE}) \otimes (\mathbf{DF})$
- $(\mathbf{C} \otimes \mathbf{D})^\top = \mathbf{C}^\top \otimes \mathbf{D}^\top$
- $(\mathbf{a} \otimes \mathbf{b}) \otimes \mathbf{c} = \mathbf{a} \otimes (\mathbf{b} \otimes \mathbf{c})$
- $\mathbf{K} = \mathbf{XVY}^\top \;\leftrightarrow\; \mathrm{vec}(\mathbf{K}) = (\mathbf{Y} \otimes \mathbf{X})\mathrm{vec}(\mathbf{V})$

and the derivative of $c(\mathbf{z}) \equiv \mathbf{c}_2^\top(\mathbf{z} \otimes \mathbf{z})$ in the direction $\mathbf{f}$ is

$$\frac{\partial c}{\partial \mathbf{z}}\mathbf{f} = \mathbf{c}_2^\top(\mathbf{f} \otimes \mathbf{z} + \mathbf{z} \otimes \mathbf{f}).$$

Writing polynomials in Kronecker product form

There are many representations of the same monomial terms when written in Kronecker product form, e.g. for the quadratic monomials

$$[1\ 0\ 2\ 1]\,(\mathbf{z} \otimes \mathbf{z}) = [1\ 1\ 1\ 1]\,(\mathbf{z} \otimes \mathbf{z}) = z_1^2 + 2z_1z_2 + z_2^2.$$

To simplify expressions, we will impose the symmetric form of the coefficients.

If the coefficients are symmetric, then the following properties hold

- $\mathbf{c}^\top (\mathbf{a} \otimes \mathbf{b}) = \mathbf{c}^\top (\mathbf{b} \otimes \mathbf{a})$
- $\mathbf{c}^\top (\mathbf{I} \otimes \mathbf{z}) \ = \mathbf{c}^\top (\mathbf{z} \otimes \mathbf{I})$

(note $\mathbf{c}^\top (\mathbf{M} \otimes \mathbf{N}) \neq \mathbf{c}^\top (\mathbf{N} \otimes \mathbf{M})$)

Quadratic-Quadratic Regulator

Quadratic-Quadratic Regulator (QQR) Problem

Find a control $\mathbf{u}(\cdot)$ with $\mathbf{u}(t) \in \mathbb{R}^m$ that solves

$$\begin{aligned}
\min_{\mathbf{u}} J(\mathbf{z}, \mathbf{u}) &= \int_0^\infty \ell(\mathbf{z}, \mathbf{u}) \; ds \\
&= \int_0^\infty \mathbf{z}(s)^\top \mathbf{Q}_2 \mathbf{z}(s) + \mathbf{u}(s)^\top \mathbf{R}_2 \mathbf{u}(s) \; ds \\
&= \int_0^\infty \mathbf{q}_2^\top (\mathbf{z}(s) \otimes \mathbf{z}(s)) + \mathbf{r}_2^\top (\mathbf{u}(s) \otimes \mathbf{u}(s)) \; ds
\end{aligned}$$

with $\mathbf{q}_2 = \mathrm{vec}(\mathbf{Q}_2)$, $\mathbf{Q}_2 = \mathbf{Q}_2^\top \geqslant 0$ and $\mathbf{r}_2 = \mathrm{vec}(\mathbf{R}_2)$, $\mathbf{R}_2 = \mathbf{R}_2^\top > 0$, subject to

$$\begin{aligned}
\dot{\mathbf{z}}(t) &= \mathbf{f}(\mathbf{z}, \mathbf{u}) \\
&= \mathbf{A}\mathbf{z}(t) + \mathbf{B}\mathbf{u}(t) + \mathbf{N}_2(\mathbf{z}(t) \otimes \mathbf{z}(t)), \qquad \mathbf{z}(0) = \mathbf{z}_0 \in \mathbb{R}^n.
\end{aligned}$$

Quadratic-Quadratic Regulator (QQR) Problem

Recall the Hamilton-Jacobi-Bellman partial differential equations

$$0 = \frac{\partial v}{\partial \mathbf{z}}(\mathbf{z})\mathbf{f}(\mathbf{z}, \mathcal{K}(\mathbf{z})) + \ell(\mathbf{z}, \mathcal{K}(\mathbf{z}))$$

$$\mathbf{0} = \frac{\partial v}{\partial \mathbf{z}}(\mathbf{z})\frac{\partial \mathbf{f}}{\partial \mathbf{u}}(\mathbf{z}, \mathcal{K}(\mathbf{z})) + \frac{\partial \ell}{\partial \mathbf{u}}(\mathbf{z}, \mathcal{K}(\mathbf{z})).$$

We now introduce Kronecker product expansions for v and $\mathcal{K}$

$$v(\mathbf{z}) = \underbrace{\mathbf{v}_2^\top(\mathbf{z}\otimes\mathbf{z})}_{v^{[2]}(\mathbf{z})} + \underbrace{\mathbf{v}_3^\top(\mathbf{z}\otimes\mathbf{z}\otimes\mathbf{z})}_{v^{[3]}(\mathbf{z})} + \cdots \quad \text{and} \quad \mathcal{K}(\mathbf{z}) = \underbrace{\mathbf{k}_1\mathbf{z}}_{\mathbf{k}^{[1]}(\mathbf{z})} + \underbrace{\mathbf{k}_2(\mathbf{z}\otimes\mathbf{z})}_{\mathbf{k}^{[2]}(\mathbf{z})} + \cdots$$

Substitute these expansions into the HJB PDEs and match terms of equal degree.

QQR: Matching Degree 2 and Degree 1 Terms

$$v(\mathbf{z}) = \underbrace{\mathbf{v}_2^\top(\mathbf{z}\otimes\mathbf{z})}_{v^{[2]}(\mathbf{z})} + \underbrace{\mathbf{v}_3^\top(\mathbf{z}\otimes\mathbf{z}\otimes\mathbf{z})}_{v^{[3]}(\mathbf{z})} + \cdots \quad \text{and} \quad \mathcal{K}(\mathbf{z}) = \underbrace{\mathbf{k}_1\mathbf{z}}_{k^{[1]}(\mathbf{z})} + \underbrace{\mathbf{k}_2(\mathbf{z}\otimes\mathbf{z})}_{k^{[2]}(\mathbf{z})} + \cdots$$

$$\mathbf{f}(\mathbf{z},\mathbf{u}) = \mathbf{A}\mathbf{z} + \mathbf{B}\mathbf{u} + \mathbf{N}_2(\mathbf{z}\otimes\mathbf{z}) \qquad \ell(\mathbf{z},\mathbf{u}) = \mathbf{q}_2^\top(\mathbf{z}\otimes\mathbf{z}) + \mathbf{r}_2^\top(\mathbf{u}\otimes\mathbf{u})$$

For example, collecting the degree two from the value function and degree one terms from the feedback equation, leads to

$$\mathbf{v}_2^\top\left((\mathbf{A}+\mathbf{B}\mathbf{k}_1)\otimes\mathbf{I}_n+\mathbf{I}_n\otimes(\mathbf{A}+\mathbf{B}\mathbf{k}_1)\right) + \mathbf{q}_2^\top(\mathbf{I}_n\otimes\mathbf{I}_n) + \mathbf{r}_2^\top(\mathbf{k}_1\otimes\mathbf{k}_1) = \mathbf{0}.$$

and

$$\mathbf{v}_2^\top(\mathbf{B}\otimes\mathbf{I}_n + \mathbf{I}_n\otimes\mathbf{B}) + \mathbf{r}_2^\top(\mathbf{k}_1\otimes\mathbf{I}_m + \mathbf{I}_m\otimes\mathbf{k}_1) = \mathbf{0}.$$

QQR: Matching Degree 2 and Degree 1 Terms

$$\mathbf{v}_2^\top \left((\mathbf{A} + \mathbf{B}\mathbf{k}_1) \otimes \mathbf{I}_n + \mathbf{I}_n \otimes (\mathbf{A} + \mathbf{B}\mathbf{k}_1) \right) + \mathbf{q}_2^\top (\mathbf{I}_n \otimes \mathbf{I}_n) + \mathbf{r}_2^\top (\mathbf{k}_1 \otimes \mathbf{k}_1) = \mathbf{0}.$$

and

$$\mathbf{v}_2^\top (\mathbf{B} \otimes \mathbf{I}_n + \mathbf{I}_n \otimes \mathbf{B}) + \mathbf{r}_2^\top (\mathbf{k}_1 \otimes \mathbf{I}_m + \mathbf{I}_m \otimes \mathbf{k}_1) = \mathbf{0}.$$

can be rearranged as

$$\left.\begin{aligned} (\mathbf{A} + \mathbf{B}\mathbf{k}_1)^\top \mathbf{V}_2 + \mathbf{V}_2 (\mathbf{A} + \mathbf{B}\mathbf{k}_1) + \mathbf{k}_1^\top \mathbf{R}_2 \mathbf{k}_1 + \mathbf{Q}_2 &= \mathbf{0} \\ \mathbf{V}_2 \mathbf{B} + \mathbf{k}_1^\top \mathbf{R}_2 &= \mathbf{0} \end{aligned}\right\} \text{coupled eqns. for } \mathbf{V}_2 \text{ and } \mathbf{k}_1$$

and upon substitution of $\mathbf{k}_1$ into the first equation,

$$\begin{aligned} \mathbf{A}^\top \mathbf{V}_2 + \mathbf{V}_2 \mathbf{A} - \mathbf{V}_2 \mathbf{B} \mathbf{R}_2^{-1} \mathbf{B}^\top \mathbf{V}_2 + \mathbf{Q}_2 = \mathbf{0} \\ \mathbf{k}_1 = -\mathbf{R}_2^{-1} \mathbf{B}^\top \mathbf{V}_2. \end{aligned}$$

Thus $\mathbf{V}_2$ solves the algebraic Riccati equation and $\mathbf{k}_1$ is the familiar solution to the linear-quadratic regulator problem.

QQR: Matching Degree 3 and Degree 2 Terms

Let $\mathbf{A}_c = \mathbf{A} + \mathbf{B}\mathbf{k}_1$. Collecting degree three terms from the value function equation

$$\begin{aligned}\mathbf{v}_3^\top & \left(\mathbf{A}_c \otimes \mathbf{I}_n \otimes \mathbf{I}_n + \mathbf{I}_n \otimes \mathbf{A}_c \otimes \mathbf{I}_n + \mathbf{I}_n \otimes \mathbf{I}_n \otimes \mathbf{A}_c\right) \\ & = -\mathbf{v}_2^\top \left((\mathbf{N}_2 + \mathbf{B}\mathbf{k}_2) \otimes \mathbf{I}_n + \mathbf{I}_n \otimes (\mathbf{N}_2 + \mathbf{B}\mathbf{k}_2)\right) - \mathbf{r}_2^\top \left(\mathbf{k}_1 \otimes \mathbf{k}_2 + \mathbf{k}_2 \otimes \mathbf{k}_1\right).\end{aligned}$$

and the degree two terms from the feedback equation

$$\mathbf{v}_3^\top (\mathbf{B} \otimes \mathbf{I}_{n^2} + \mathbf{I}_n \otimes \mathbf{B} \otimes \mathbf{I}_n + \mathbf{I}_{n^2} \otimes \mathbf{B}) + \mathbf{r}_2^\top (\mathbf{k}_2 \otimes \mathbf{I}_m + \mathbf{I}_m \otimes \mathbf{k}_2) = \mathbf{0}.$$

QQR: Matching Degree 3 and Degree 2 Terms

Let $\mathbf{A}_c = \mathbf{A} + \mathbf{B}\mathbf{k}_1$. Collecting degree three terms from the value function equation

$$\begin{aligned}&\mathbf{v}_3^\top \left(\mathbf{A}_c \otimes \mathbf{I}_n \otimes \mathbf{I}_n + \mathbf{I}_n \otimes \mathbf{A}_c \otimes \mathbf{I}_n + \mathbf{I}_n \otimes \mathbf{I}_n \otimes \mathbf{A}_c\right)\\ &\quad = -\mathbf{v}_2^\top \left((\mathbf{N}_2 + \mathbf{B}\mathbf{k}_2) \otimes \mathbf{I}_n + \mathbf{I}_n \otimes (\mathbf{N}_2 + \mathbf{B}\mathbf{k}_2)\right) - \mathbf{r}_2^\top \left(\mathbf{k}_1 \otimes \mathbf{k}_2 + \mathbf{k}_2 \otimes \mathbf{k}_1\right).\end{aligned}$$

and the degree two terms from the feedback equation

$$\mathbf{v}_3^\top (\mathbf{B} \otimes \mathbf{I}_{n^2} + \mathbf{I}_n \otimes \mathbf{B} \otimes \mathbf{I}_n + \mathbf{I}_{n^2} \otimes \mathbf{B}) + \mathbf{r}_2^\top (\mathbf{k}_2 \otimes \mathbf{I}_m + \mathbf{I}_m \otimes \mathbf{k}_2) = \mathbf{0}.$$

To solve the equations above: identify all of the $\mathbf{k}_2$ terms in the top equation

$$\begin{aligned}&-\mathbf{v}_2^\top (\mathbf{B}\mathbf{k}_2 \otimes \mathbf{I}_n + \mathbf{I}_n \otimes \mathbf{B}\mathbf{k}_2) - \mathbf{r}_2^\top (\mathbf{k}_1 \otimes \mathbf{k}_2 + \mathbf{k}_2 \otimes \mathbf{k}_1)\\ &\quad = -\left(\mathbf{v}_2^\top (\mathbf{B} \otimes \mathbf{I}_n) + \mathbf{r}_2^\top (\mathbf{I}_m \otimes \mathbf{k}_1)\right)(\mathbf{k}_2 \otimes \mathbf{I}_n) - \left(\mathbf{v}_2^\top (\mathbf{I}_n \otimes \mathbf{B}) + \mathbf{r}_2^\top (\mathbf{k}_1 \otimes \mathbf{I}_n)\right)(\mathbf{I}_n \otimes \mathbf{k}_2).\end{aligned}$$

Now, recall the degree one terms from the previous page:

$$\mathbf{v}_2^\top (\mathbf{B} \otimes \mathbf{I}_n) + \mathbf{r}_2^\top (\mathbf{I}_m \otimes \mathbf{k}_1) = \mathbf{0} = \mathbf{v}_2^\top (\mathbf{I}_n \otimes \mathbf{B}) + \mathbf{r}_2^\top (\mathbf{k}_1 \otimes \mathbf{I}_m).$$

QQR: Matching Degree 3 and Degree 2 Terms

Let $\mathbf{A}_c = \mathbf{A} + \mathbf{B}\mathbf{k}_1$. Collecting degree three terms from the value function equation

$$\begin{aligned}\mathbf{v}_3^\top & \left(\mathbf{A}_c \otimes \mathbf{I}_n \otimes \mathbf{I}_n + \mathbf{I}_n \otimes \mathbf{A}_c \otimes \mathbf{I}_n + \mathbf{I}_n \otimes \mathbf{I}_n \otimes \mathbf{A}_c\right) \\ &= -\mathbf{v}_2^\top \left((\mathbf{N}_2 + \mathbf{B}\mathbf{k}_2) \otimes \mathbf{I}_n + \mathbf{I}_n \otimes (\mathbf{N}_2 + \mathbf{B}\mathbf{k}_2)\right) - \mathbf{r}_2^\top \left(\mathbf{k}_1 \otimes \mathbf{k}_2 + \mathbf{k}_2 \otimes \mathbf{k}_1\right).\end{aligned}$$

and the degree two terms from the feedback equation

$$\mathbf{v}_3^\top (\mathbf{B} \otimes \mathbf{I}_{n^2} + \mathbf{I}_n \otimes \mathbf{B} \otimes \mathbf{I}_n + \mathbf{I}_{n^2} \otimes \mathbf{B}) + \mathbf{r}_2^\top (\mathbf{k}_2 \otimes \mathbf{I}_m + \mathbf{I}_m \otimes \mathbf{k}_2) = \mathbf{0}.$$

To solve the equations above: identify all of the $\mathbf{k}_2$ terms in the top equation

$$\begin{aligned}-\mathbf{v}_2^\top & (\mathbf{B}\mathbf{k}_2 \otimes \mathbf{I}_n + \mathbf{I}_n \otimes \mathbf{B}\mathbf{k}_2) - \mathbf{r}_2^\top (\mathbf{k}_1 \otimes \mathbf{k}_2 + \mathbf{k}_2 \otimes \mathbf{k}_1) \\ &= -\left(\mathbf{v}_2^\top (\mathbf{B} \otimes \mathbf{I}_n) + \mathbf{r}_2^\top (\mathbf{I}_m \otimes \mathbf{k}_1)\right) (\mathbf{k}_2 \otimes \mathbf{I}_n) - \left(\mathbf{v}_2^\top (\mathbf{I}_n \otimes \mathbf{B}) + \mathbf{r}_2^\top (\mathbf{k}_1 \otimes \mathbf{I}_n)\right) (\mathbf{I}_n \otimes \mathbf{k}_2).\end{aligned}$$

Now, recall the degree one terms from the previous page:

$$\mathbf{v}_2^\top (\mathbf{B} \otimes \mathbf{I}_n) + \mathbf{r}_2^\top (\mathbf{I}_m \otimes \mathbf{k}_1) = \mathbf{0} = \mathbf{v}_2^\top (\mathbf{I}_n \otimes \mathbf{B}) + \mathbf{r}_2^\top (\mathbf{k}_1 \otimes \mathbf{I}_m).$$

So **all** of the $\mathbf{k}_2$ terms in the top equation vanish, and eqns decouple.

The first equation can be solved for $\mathbf{v}_3$, then inserted into the second equation to compute $\mathbf{k}_2$.

QQR: Simplification of the $\mathbf{v}_3$ Equation

$$\mathbf{v}_3^\top \left(\mathbf{A}_c \otimes \mathbf{I}_n \otimes \mathbf{I}_n + \mathbf{I}_n \otimes \mathbf{A}_c \otimes \mathbf{I}_n + \mathbf{I}_n \otimes \mathbf{I}_n \otimes \mathbf{A}_c\right) = -\mathbf{v}_2^\top \left(\mathbf{N}_2 \otimes \mathbf{I}_n + \mathbf{I}_n \otimes \mathbf{N}_2\right).$$

Taking the transpose of this equation:

$$\left(\mathbf{A}_c^\top \otimes \mathbf{I}_n \otimes \mathbf{I}_n + \mathbf{I}_n \otimes \mathbf{A}_c^\top \otimes \mathbf{I}_n + \mathbf{I}_n \otimes \mathbf{I}_n \otimes \mathbf{A}_c^\top\right) \mathbf{v}_3 = -\left(\mathbf{N}_2^\top \otimes \mathbf{I}_n + \mathbf{I}_n \otimes \mathbf{N}_2^\top\right) \mathbf{v}_2.$$

We define the special Kronecker sum as

$$\mathcal{L}_d(\mathbf{X}) \equiv \underbrace{\mathbf{X} \otimes \mathbf{I}_n \otimes \cdots \otimes \mathbf{I}_n}_{d \text{ terms}} + \cdots + \underbrace{\mathbf{I}_n \otimes \cdots \otimes \mathbf{I}_n \otimes \mathbf{X}}_{d \text{ terms}}.$$

Then we can write equation for $\mathbf{v}_3$ compactly as

$$\mathcal{L}_3(\mathbf{A}_c^\top)\mathbf{v}_3 = -\mathcal{L}_2(\mathbf{N}_2^\top)\mathbf{v}_2.$$

Simplified Description of the Al'brekht Algorithm

Using this special Kronecker sum,

$$\mathcal{L}_d(\mathbf{X}) \equiv \underbrace{\mathbf{X} \otimes \mathbf{I}_n \otimes \cdots \otimes \mathbf{I}_n}_{d \text{ terms}} + \cdots + \underbrace{\mathbf{I}_n \otimes \cdots \otimes \mathbf{I}_n \otimes \mathbf{X}}_{d \text{ terms}}.$$

we can write the higher degree terms in the same way, for example

$$\begin{aligned}
\mathcal{L}_3(\mathbf{A}_c^\top)\mathbf{v}_3 &= -\mathcal{L}_2(\mathbf{N}_2^\top)\mathbf{v}_2 \\
\mathcal{L}_4(\mathbf{A}_c^\top)\mathbf{v}_4 &= -\mathcal{L}_3((\mathbf{B}\mathbf{k}_2 + \mathbf{N}_2)^\top)\mathbf{v}_3 - (\mathbf{k}_2^\top \otimes \mathbf{k}_2^\top)\mathbf{r}_2, \\
\mathcal{L}_5(\mathbf{A}_c^\top)\mathbf{v}_5 &= -\mathcal{L}_4((\mathbf{B}\mathbf{k}_2 + \mathbf{N}_2)^\top)\mathbf{v}_4 - \mathcal{L}_3((\mathbf{B}\mathbf{k}_3)^\top)\mathbf{v}_3 \\
&\quad -(\mathbf{k}_2 \otimes \mathbf{k}_3 + \mathbf{k}_3 \otimes \mathbf{k}_2)^\top \mathbf{r}_2.
\end{aligned}$$

and for all of these...

$$\mathbf{k}_d = -\frac{1}{2}\mathbf{R}_2^{-1}\left(\mathcal{L}_{d+1}(\mathbf{B}^\top)\mathbf{v}_{d+1}\right)^\top.$$

Simplified Description of the Al'brekht Algorithm

Using this special Kronecker sum,

$$\mathcal{L}_d(\mathbf{X}) \equiv \underbrace{\mathbf{X} \otimes \mathbf{I}_n \otimes \cdots \otimes \mathbf{I}_n}_{d \text{ terms}} + \cdots + \underbrace{\mathbf{I}_n \otimes \cdots \otimes \mathbf{I}_n \otimes \mathbf{X}}_{d \text{ terms}}.$$

we can write the higher degree terms in the same way, for example

$$\begin{aligned}
\mathcal{L}_3(\mathbf{A}_c^\top)\mathbf{v}_3 &= -\mathcal{L}_2(\mathbf{N}_2^\top)\mathbf{v}_2 \\
\mathcal{L}_4(\mathbf{A}_c^\top)\mathbf{v}_4 &= -\mathcal{L}_3((\mathbf{B}\mathbf{k}_2 + \mathbf{N}_2)^\top)\mathbf{v}_3 - (\mathbf{k}_2^\top \otimes \mathbf{k}_2^\top)\mathbf{r}_2, \\
\mathcal{L}_5(\mathbf{A}_c^\top)\mathbf{v}_5 &= -\mathcal{L}_4((\mathbf{B}\mathbf{k}_2 + \mathbf{N}_2)^\top)\mathbf{v}_4 - \mathcal{L}_3((\mathbf{B}\mathbf{k}_3)^\top)\mathbf{v}_3 \\
&\quad -(\mathbf{k}_2 \otimes \mathbf{k}_3 + \mathbf{k}_3 \otimes \mathbf{k}_2)^\top \mathbf{r}_2.
\end{aligned}$$

and for all of these...

$$\mathbf{k}_d = -\frac{1}{2}\mathbf{R}_2^{-1}\left(\mathcal{L}_{d+1}(\mathbf{B}^\top)\mathbf{v}_{d+1}\right)^\top.$$

This is consistent with the linear-quadratic regulator problem.

Polynomial-Quadratic Regulator

The Polynomial-Quadratic Regulator Problem

$$\min_{\mathbf{u}} J(\mathbf{z}, \mathbf{u}) = \int_0^\infty \ell(\mathbf{z}(t), \mathbf{u}(t))\ dt,$$

subject to the system dynamics

$$\dot{\mathbf{z}}(t) = \mathbf{f}(\mathbf{z}(t), \mathbf{u}(t)), \quad \mathbf{z}(0) = \mathbf{z}_0,$$

where now

$$\mathbf{f}(\mathbf{z}, \mathbf{u}) \equiv \mathbf{A}\mathbf{z}(t) + \mathbf{B}\mathbf{u}(t) + \mathbf{N}_2(\mathbf{z} \otimes \mathbf{z}) + \cdots + \mathbf{N}_p(\mathbf{z} \otimes \cdots \otimes \mathbf{z}),$$

with $\mathbf{N}_d \in \mathbb{R}^{n \times n^d}$ for $d = 2, \ldots, p$.

The running cost is still quadratic in $\mathbf{z}$ and $\mathbf{u}$:

$$\ell(\mathbf{z}, \mathbf{u}) = \mathbf{q}_2^\top (\mathbf{z} \otimes \mathbf{z}) + \mathbf{r}_2^\top (\mathbf{u} \otimes \mathbf{u}).$$

Equations for $\mathbf{v}_{d+1}$ in the PQR Problem

Following the same process, we see that the systems are similar:

$$\begin{aligned}
\mathcal{L}_3(\mathbf{A}_c^\top)\mathbf{v}_3 &= -\mathcal{L}_2(\mathbf{N}_2^\top)\mathbf{v}_2, \\
\mathcal{L}_4(\mathbf{A}_c^\top)\mathbf{v}_4 &= -\mathcal{L}_3((\mathbf{B}\mathbf{k}_2+\mathbf{N}_2)^\top)\mathbf{v}_3 - (\mathbf{k}_2^\top \otimes \mathbf{k}_2^\top)\mathbf{r}_2, \\
\mathcal{L}_5(\mathbf{A}_c^\top)\mathbf{v}_5 &= -\mathcal{L}_4((\mathbf{B}\mathbf{k}_2+\mathbf{N}_2)^\top)\mathbf{v}_4 - \mathcal{L}_3((\mathbf{B}\mathbf{k}_3+\mathbf{N}_3)^\top)\mathbf{v}_3 \\
&\quad -(\mathbf{k}_2^\top \otimes \mathbf{k}_3^\top + \mathbf{k}_3^\top \otimes \mathbf{k}_2^\top)\mathbf{r}_2.
\end{aligned}$$

We have the same decoupling of the $\mathbf{v}_{d+1}$ and $\mathbf{k}_d$ equations as in the QQR problem. Furthermore,

$$\mathbf{k}_d = -\frac{1}{2}\mathbf{R}^{-1}\left(\mathcal{L}_{d+1}(\mathbf{B}^\top)\mathbf{v}_{d+1}\right)^\top. \tag{3}$$

Simplified Description of the Al'brekht Algorithm

Generally,

$$\begin{aligned}
\mathcal{L}_{d+1}(\mathbf{A}_c^\top)\mathbf{v}_{d+1} \quad &= \quad -\mathcal{L}_2(\mathbf{N}_d^\top)\mathbf{v}_2 \\
&\quad - \sum_{i=3}^{d} \mathcal{L}_i(\mathbf{B}\mathbf{k}_{d+2-i} + \mathbf{N}_{d+2-i})^\top \mathbf{v}_i \\
&\quad - \sum_{\substack{i,j>1 \\ i+j=d+1}} \left(\mathbf{k}_i^\top \otimes \mathbf{k}_j^\top\right) \mathbf{r}_2
\end{aligned}$$

Quadratic Bilinear-Quadratic Regulator

The Quadratic Bilinear-Quadratic Regulator Problem

$$\min_{\mathbf{u}} J(\mathbf{z},\mathbf{u}) = \int_0^\infty \ell(\mathbf{z}(t),\mathbf{u}(t))\ dt,$$

subject to the system dynamics

$$\dot{\mathbf{z}}(t) = \mathbf{f}(\mathbf{z}(t),\mathbf{u}(t)), \quad \mathbf{z}(0) = \mathbf{z}_0,$$

where now

$$\mathbf{f}(\mathbf{z},\mathbf{u}) \equiv \mathbf{A}\mathbf{z}(t) + \mathbf{B}\mathbf{u}(t) + \mathbf{N}_2(\mathbf{z}\otimes\mathbf{z}) + \mathbf{N}_{zu}(\mathbf{z}\otimes\mathbf{u}) + \mathbf{N}_{uu}(\mathbf{u}\otimes\mathbf{u}),$$

with $\mathbf{N}_2 \in \mathbb{R}^{n\times n^2}$, $\mathbf{N}_{zu} \in \mathbb{R}^{n\times nm}$ and $\mathbf{N}_{uu} \in \mathbb{R}^{n\times m^2}$.

The running cost is still quadratic in $\mathbf{z}$ and $\mathbf{u}$:

$$\ell(\mathbf{z},\mathbf{u}) = \mathbf{q}_2^\top(\mathbf{z}\otimes\mathbf{z}) + \mathbf{r}_2^\top(\mathbf{u}\otimes\mathbf{u}).$$

Equations for $\mathbf{v}_{d+1}$ in the QBQR Problem

Following the same process, we see that the systems are similar:

$$
\begin{aligned}
\mathcal{L}_3(\mathbf{A}_c^\top)\mathbf{v}_3 &= -\mathcal{L}_2\left(\mathbf{N}_2 + \mathbf{N}_{zu}(\mathbf{I}_n \otimes \mathbf{k}_1) + \mathbf{N}_{uu}(\mathbf{k}_1 \otimes \mathbf{k}_1)\right)^\top \mathbf{v}_2, \\
\mathcal{L}_4(\mathbf{A}_c^\top)\mathbf{v}_4 &= -\mathcal{L}_3(\mathbf{B}\mathbf{k}_2 + \mathbf{N}_2 + \mathbf{N}_{zu}(\mathbf{I}_n \otimes \mathbf{k}_1) + \mathbf{N}_{uu}(\mathbf{k}_1 \otimes \mathbf{k}_1))^\top \mathbf{v}_3 \\
&\quad -\mathcal{L}_2(\mathbf{N}_{zu}(\mathbf{I}_n \otimes \mathbf{k}_2) + \mathbf{N}_{uu}(\mathbf{k}_1 \otimes \mathbf{k}_2 + \mathbf{k}_2 \otimes \mathbf{k}_1))^\top \mathbf{v}_2 \\
&\quad -(\mathbf{k}_2^\top \otimes \mathbf{k}_2^\top)\mathbf{r}_2,
\end{aligned}
$$

We have the same decoupling of the $\mathbf{v}_{d+1}$ and $\mathbf{k}_d$ equations as in the above problems. However, the equations for $\mathbf{k}_d$ are more complex to extract. For example, gathering $O(z^2)$ terms has the form

$$
\begin{aligned}
\mathbf{v}_3^\top [&(\mathbf{B} \otimes \mathbf{I}_n \otimes \mathbf{I}_n)(\mathbf{I}_m \otimes \mathbf{z} \otimes \mathbf{z}) + (\mathbf{I}_n \otimes \mathbf{B} \otimes \mathbf{I}_n)(\mathbf{z} \otimes \mathbf{I}_m \otimes \mathbf{z}) \\
&(\mathbf{I}_n \otimes \mathbf{I}_n \otimes \mathbf{B})(\mathbf{z} \otimes \mathbf{z} \otimes \mathbf{I}_m)] + \text{many similar terms.}
\end{aligned}
$$

Perfect Shuffle Matrices

Let $\mathbf{S}_{pq}$ be the perfect shuffle matrix:

> *Split a deck (vector) into p piles of q cards each, then recombine the deck (vector) by cyclically taking the first card from each pile.*

Interesting properties:

- if $\mathbf{A} \in \mathbb{R}^{m \times n}$ then $\mathrm{vec}(\mathbf{A}^\top) = \mathbf{S}_{nm}\mathrm{vec}(\mathbf{A})$.
- if $\mathbf{B} \in \mathbb{R}^{m_b \times n_b}$ and $\mathbf{C} \in \mathbb{R}^{m_c \times n_c}$, then

$$\mathbf{C} \otimes \mathbf{B} = \mathbf{S}_{m_b m_c}(\mathbf{B} \otimes \mathbf{C})\mathbf{S}_{n_b n_c}.$$

Perfect Shuffle Matrices

Let $\mathbf{S}_{pq}$ be the perfect shuffle matrix:

Split a deck (vector) into p piles of q cards each, then recombine the deck (vector) by cyclically taking the first card from each pile.

Interesting properties:

- if $\mathbf{A} \in \mathbb{R}^{m \times n}$ then $\mathrm{vec}(\mathbf{A}^\top) = \mathbf{S}_{nm}\mathrm{vec}(\mathbf{A})$.
- if $\mathbf{B} \in \mathbb{R}^{m_b \times n_b}$ and $\mathbf{C} \in \mathbb{R}^{m_c \times n_c}$, then

$$\mathbf{C} \otimes \mathbf{B} = \mathbf{S}_{m_b m_c}(\mathbf{B} \otimes \mathbf{C})\mathbf{S}_{n_b n_c}.$$

Thus,

- $\mathbf{z} \otimes \mathbf{I}_m \otimes \mathbf{z} = (\mathbf{S}_{mn} \otimes \mathbf{I}_n)(\mathbf{I}_m \otimes \mathbf{z} \otimes \mathbf{z})$
- $\mathbf{z} \otimes \mathbf{z} \otimes \mathbf{I}_m = \mathbf{S}_{mn^2}(\mathbf{I}_m \otimes \mathbf{z} \otimes \mathbf{z})$

These are used to write equations for the feedback coefficients $\mathbf{k}_d$ (omitting the details here).

Software Implementation

Software to Download

If you want to follow along in the software, download the following:

- KroneckerTools: `https://github.com/jborggaard/KroneckerTools`
- NLbalancing: `https://github.com/jborggaard/NLbalancing`
- PolynomialSystems: `https://github.com/jborggaard/PolynomialSystems`
- QQR: `https://github.com/jborggaard/QQR`

We'll need this software for the hands-on experiments after the break.

Elements (under the hood)

$$\begin{aligned}\mathcal{L}_{d+1}(\mathbf{A}_c^\top)\mathbf{v}_{d+1} &= -\mathcal{L}_2(\mathbf{N}_d^\top)\mathbf{v}_2 - \sum_{i=3}^{d} \mathcal{L}_i(\mathbf{B}\mathbf{k}_{d+2-i} + \mathbf{N}_{d+2-i})^\top \mathbf{v}_i \\ &\quad - \sum_{\substack{i,j>1 \\ i+j=d+1}} \left(\mathbf{k}_i^\top \otimes \mathbf{k}_j^\top\right) \mathbf{r}_2\end{aligned}$$

- Linear System Solver: `KroneckerTools/src/KroneckerSumSolver.m`

  Solves systems of the form: $\mathcal{L}_{d+1}(\mathbf{A}_c^\top)\mathbf{v}_{d+1} = \mathbf{b}$.

  ```
  [v] = KroneckerSumSolver(Ac.',b,d+1);
  ```

- Forming the RHS: `KroneckerTools/src/LyapProduct.m`

  Performs the products: $\mathcal{L}_i(\mathbf{M}^\top)\mathbf{v}_i$.

  ```
  [b] = LyapProduct(M.',v,i);
  ```

- and $(\mathbf{k}_i^\top \otimes \mathbf{k}_j^\top)\mathbf{r}_2 = \mathtt{vec}(\mathbf{k}_j^\top \mathbf{R}_2 \mathbf{k}_i)$ $(= \mathtt{vec}((\mathbf{k}_i^\top \mathbf{R}_2 \mathbf{k}_j)^\top))$

Note that $\mathbf{A}_c = \mathbf{A} + \mathbf{B}\mathbf{k}_1$ is a stable matrix by construction.

We use an N-way variation of the Bartels-Stewart algorithm.

We perform a Schur decomposition on the matrix $\mathbf{A}_c^\top = \mathbf{U}\mathbf{T}\mathbf{U}^*$.

$$\mathcal{L}_{d+1}(\mathbf{A}_c^\top) = (\mathbf{U} \otimes \cdots \otimes \mathbf{U})\mathcal{L}_{d+1}(\mathbf{T})(\mathbf{U} \otimes \cdots \otimes \mathbf{U})^*.$$

Therefore, using a change of variable $\mathbf{U}$, we can transform our problem to one of the form

$$\mathcal{L}_{d+1}(\mathbf{T})\tilde{\mathbf{v}}_{d+1} = \tilde{\mathbf{b}},$$

where $\tilde{\mathbf{b}} = (\mathbf{U}^* \otimes \mathbf{U}^* \otimes \cdots \otimes \mathbf{U}^*)\mathbf{b}$.

A special Kronecker sum system with triangular $\mathbf{T}$ is an $n^{d+1} \times n^{d+1}$ triangular system.

After solving for $\tilde{\mathbf{v}}_{d+1}$, we compute $\mathbf{v}_{d+1} = (\mathbf{U} \otimes \mathbf{U} \otimes \cdots \otimes \mathbf{U})\tilde{\mathbf{v}}_{d+1}$.

Comments

- If we perform a Schur factorization of $\mathbf{A}_c$, we can make $\mathcal{L}_d(\mathbf{A}_c)$ upper triangular. (but a direct solve would be $\approx O(n^{2d})$ work)
- $\mathbf{A}_c$ is a stable matrix, by the above, the eigenvalues of $\mathcal{L}_d(\mathbf{A}_c)$ are sums of the eigenvalues of $\mathbf{A}_c$. (these systems are solvable for $\mathbf{v}_d$)
- An n-Way version of the Bartels and Stewart algorithm has been implemented to solve these special Kronecker sum systems of equations.
- A block recursive algorithm by Chen and Kressner, which is suitable for more general Kronecker sum systems, is more efficient. (the complexity is just $\approx O(n^{d+1})$ work)
- The assembly of the RHS can also be performed efficiently (products of $\mathcal{L}_d\mathbf{v}_d$)

KroneckerSumSolver.m

For the change of variable:

```
[U,T] = schur(A.','complex');
b = kroneckerLeft(U',b);
```

To simplify the $n \times n$ block-backsubstitution steps:

```
X = zeros(n,n^(degree-1));
B = reshape(b,n,n^(degree-1));
```

The diagonal blocks of $\mathcal{L}_d(\mathbf{T})$ have the form (using `T` for convenience):

```
diagT = 0;
for i=2:degree
  diagT = diagT + T(jIndex(i),jIndex(i));
end
Tt = T + diagT*eye(n);
```

For the current block, we extract the portion of the vector b:

```
rhs = B(:,colIdx);
```

The block-backsubstitution steps have the form:

```
rhs = rhs - X(:,jIdx)*T(jIndex(i),jRange{i}).';
```

Finally, the next n values of $\mathbf{x}$ are solved for

```
X(:,colIdx) = At\rhs;
```

When we are done, we reshape and apply the change of variables:

```
x = X(:);
x = real(kroneckerLeft(U,x));
```

Numerical Examples

A Scalar Example

Minimize

$$J(u) = \frac{1}{2}\int_0^\infty z^2(t) + u^2(t)\ dt$$

subject to

$$\dot{z} = z - z^3 + u,$$

From $z(0) = z_0 = 1$

d	$v^{d+1]}(z_0)$	$\int_0^2 \ell(z,u)dt$
1	1.207110	0.912902
3	0.780330	0.828734
5	0.809793	0.824724
7	0.820841	0.824338

From $z(0) = z_0 = 1.25$

d	$v^{d+1]}(z_0)$	$\int_0^2 \ell(z,u)dt$
1	1.88610	1.27062
3	0.844169	1.15039
5	0.956561	1.05377
7	1.02242	1.04152

Scalar Example Continued

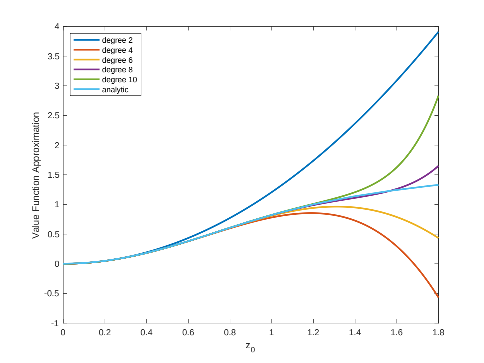


Figure 1: Value Function Approximation

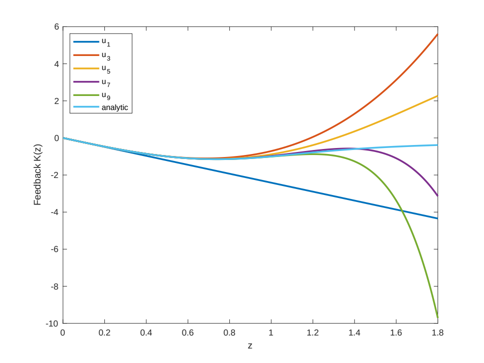
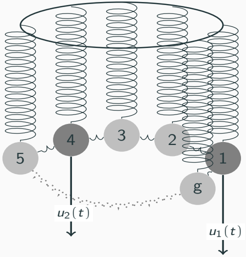


Figure 2: Feedback Approximation

A Scalable Cubic Polynomial System

An example of a *polynomial system* is the ring of van der Pol oscillators



Specify controls $\mathbf{u}(\cdot) \in L_2(0, \infty; \mathbb{R}^m)$ that stabilize

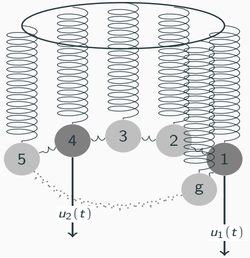
$$\ddot{y}_i + (y_i^2 - 1)\dot{y}_i + y_i = y_{i-1} - 2y_i + y_{i+1} + b_i u_i(t),$$

for $i = 1, \ldots, g$ with $y_i(0) = y^0$ and $\dot{y}_i(0) = 0$.

We identify $y_{g+1} = y_1$ and $y_g = y_0$ to close the ring.

A Scalable Cubic Polynomial System

An example of a *polynomial system* is the ring of van der Pol oscillators



Specify controls $\mathbf{u}(\cdot) \in L_2(0,\infty;\mathbb{R}^m)$ that stabilize

$$\ddot{y}_i + (y_i^2 - 1)\dot{y}_i + y_i = y_{i-1} - 2y_i + y_{i+1} + b_i u_i(t),$$

for $i = 1,\ldots,g$ with $y_i(0) = y^0$ and $\dot{y}_i(0) = 0$.

We identify $y_{g+1} = y_1$ and $y_g = y_0$ to close the ring.

Find a control $\mathbf{u}(\cdot)$ with $\mathbf{u}(t) \in \mathbb{R}^m$ that solves

$$\min_{\mathbf{u}} J(\mathbf{z},\mathbf{u}) = \int_0^\infty \mathbf{z}(s)^\top \mathbf{Q}_2 \mathbf{z}(s) + \mathbf{u}(s)^\top \mathbf{R}_2 \mathbf{u}(s)\; ds \quad (\mathbf{Q}_2 = \mathbf{Q}_2^\top \geqslant 0, \mathbf{R}_2 = \mathbf{R}_2^\top > 0)$$

subject to

$$\dot{\mathbf{z}}(t) = \mathbf{f}(\mathbf{z}(t),\mathbf{u}(t)), \qquad \mathbf{z}(0) = \mathbf{z}_0 \in \mathbb{R}^n \qquad (n = 2g, m < n).$$

Cubic-Quadratic Regulator Problems

- The second-order model

$$\ddot{y}_i + (y_i^2 - 1)\dot{y}_i + y_i = y_{i-1} - 2y_i + y_{i+1} + b_i u_i(t),$$

for $i = 1, \ldots, g$ can be written as

$$\dot{\mathbf{z}} = \mathbf{A}\mathbf{z} + \mathbf{B}\mathbf{u} + \mathbf{N}_3\mathbf{z}^{③}$$

with initial conditions set using $y_i(0) = 0.3$ and $\dot{y}_i(0) = 0$.

- We identify $y_{g+1} = y_1$ and $y_g = y_0$ to close the ring.
- The stability of this system was studied in Nana and Woafo 2006 and a related control problem considered in Barron 2016.
- Choosing different values of g and rewriting as a first-order system of differential equations allows us to study the *cubic-quadratic* regulator problem for problems of size $n = 2g$.
- We set b_i as 0 or 1 with $m = \|\mathbf{b}\|_1$.

Convergence of the Value Function with Increasing Degree

Experiment: $g = 4,\ b_1 = b_2 = 1$.

Table 1: van der Pol: Value Function Approx.

d	$\sum_{i=2}^{d+1} v^{[i]}(\mathbf{z}_0)$	$\int_0^{50} \ell(\mathbf{z}(t), \mathbf{u}(t))dt$
1	4.6380	4.4253
2	4.6380	4.4253
3	4.4125	4.4208
4	4.4125	4.4208
5	4.4246	4.4208
6	4.4246	4.4208
7	4.4242	4.4208

Nonlinear Feedback for the Chafee-Infante Equation

Find $u(\cdot)$ that minimizes

$$J(z,u) = \int_0^\infty \left(\int_0^1 z^2(x,t)\ dx + \|u(t)\|^2 \right)\ dt$$

subject to

$$\dot{z}(x,t) = \nu z_{xx} + \alpha z - z^3 + \sum_{k=1}^{m} \chi_{[(k-1)/m, k/m)}(x) u_k(t)$$

$$z(x,0) = 1.25\cos(3\pi x) \in H^1(0,1), \qquad z_x(0,t) = 0 = z_x(1,t),$$

The zero solution is unstable when $\alpha > 4\pi^2\nu$.

Discretize with 20 linear FE ($n = 21$), $m = 10$, and consider a range of values of α and ν.

Approximate the cubic-quadratic regulator to compute the control.

Source Control, $m = 10$

Table 2: Chafee-Infante Value Function: Source Control

$d+1$	$\sum_{i=2}^{d+1} v^{[i]}(\mathbf{z}_0)$	$\int_0^T \ell(\mathbf{z}(t), \mathbf{u}(t))dt$
$\nu = 1,\ \alpha = 100$		
2	160.3403039	167.6456054
4	167.1904854	167.2009756
6	167.2013401	167.2009701
$\nu = 0.1,\ \alpha = 10$		
2	16.4303086	36.6666702
4	23.1192400	23.2573375
6	23.3103412	23.2549197
$\nu = 0.01,\ \alpha = 1$		
2	3.4886482	diverged
4	7.9363494	10.0721129
6	11.0884438	9.7386884

Closed-loop Simulations, $\nu = 0.1, \alpha = 10$ Case

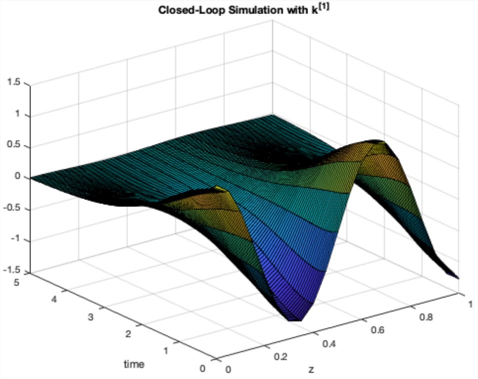


Figure 3: Linear Feedback

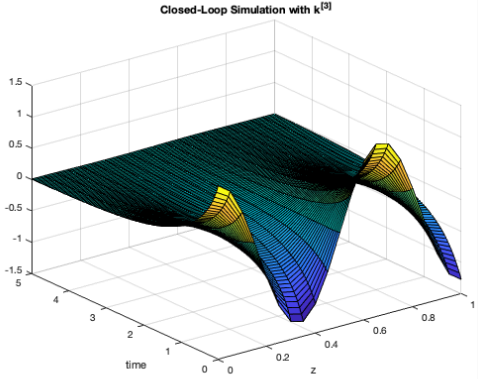


Figure 4: Cubic Feedback

Closed-loop Simulations, $\nu = 0.1, \alpha = 10$ Case

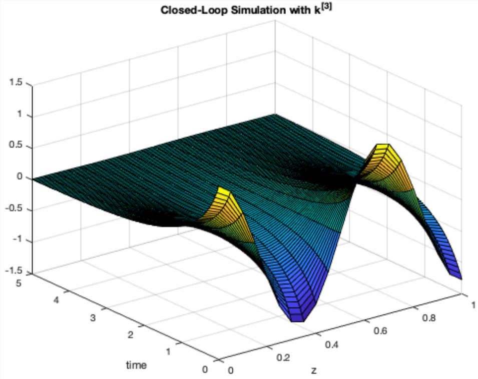


Figure 5: Cubic Feedback, $n = 20$

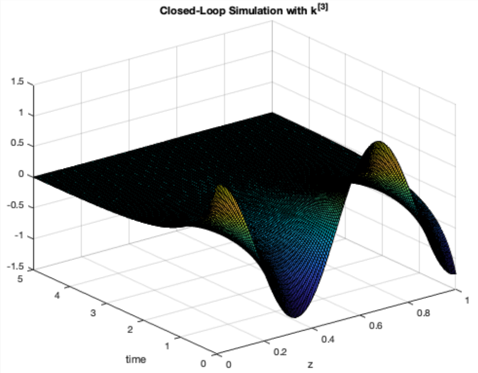


Figure 6: Cubic Feedback, $n = 200$

Neumann Control, $m = 10$

Table 3: Chafee-Infante Value Function: Neumann Control

$d+1$	$\sum_{i=2}^{d+1} v^{[i]}(\mathbf{z}_0)$	$\int_0^T \ell(\mathbf{z}(t), \mathbf{u}(t))dt$
$\nu = 1,\ \alpha = 100$		
2	5.5621284	6.2714968
4	6.1631359	6.2076474
6	6.2054292	6.2073693
$\nu = 0.1,\ \alpha = 10$		
2	1.1756343	diverges
4	1.8783304	2.9177025
6	2.2771905	2.5498542

Closed-loop Simulations, $\nu = 0.1, \alpha = 10$ Case

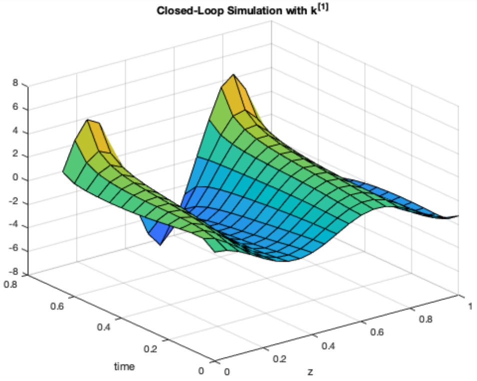


Figure 7: Linear Feedback

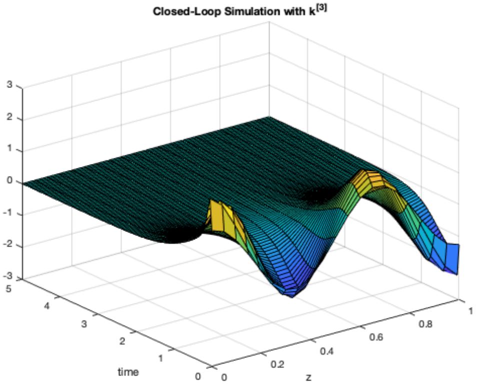


Figure 8: Cubic Feedback

Closed-loop Simulations, $\nu = 0.1, \alpha = 10$ Case

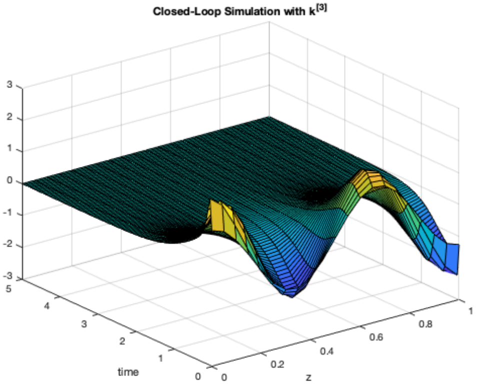


Figure 9: Cubic Feedback

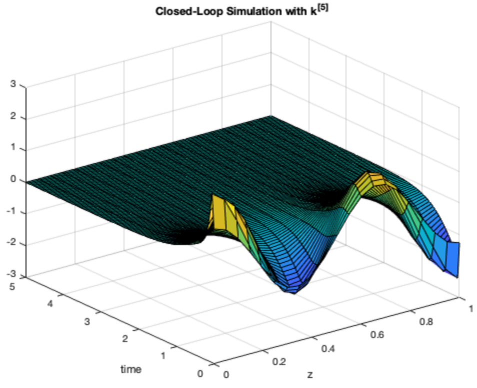


Figure 10: Quintic Feedback

Future Work

- Test on a flow control problem
- DAEs
- Look at other polynomial representations (SoS, tensor).
- Show convergence in PDE setting.

References

Borggaard and Zietsman, *The Quadratic-Quadratic Regulator problem: Approximating feedback controls for quadratic-in-state nonlinear systems*, arχiv: 1910.03396 (2020 American Control Conference).

Borggaard and Zietsman, *On approximating Polynomial-Quadratic Regulator problems*, arχiv: 2009.11068 (IFAC Papers Online, 54(9), 329–334 (2021)).

https://github.com/jborggaard/QQR

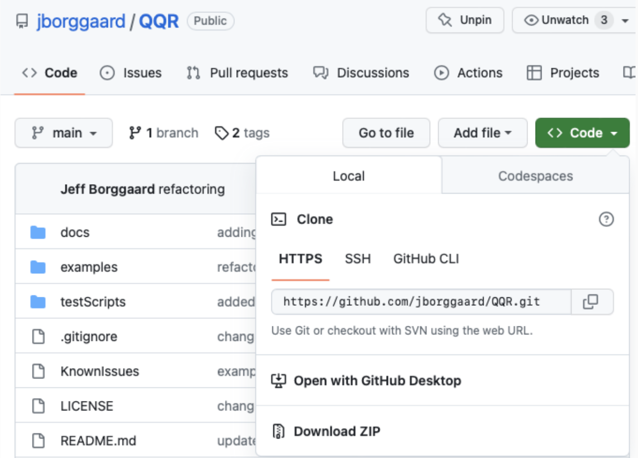
Chen and Kressner, *Recursive blocked algorithms for linear systems with Kronecker product structure*, Numerical Algorithms, v. 84, 1199–1216 (2020), arχiv: 1905.09539.

Lunasin and Titi, *Finite determining parameters feedback control for distributed nonlinear dissipative systems–A computational study*, arχiv: 1506.03709v2.

Hands-on Experiments

Software Installation

At `https://github.com/jborggaard`



Software Installation (continued)

1. Download `KroneckerTools`, `NLbalancing`, `QQR`, and `PolynomialSystems`
2. Open MATLAB
    - Navigate to NLbalancing
        - Edit `setKroneckerToolsPath.m`
    - Navigate to PolynomialSystems
        - Edit `setPaths.m`
    - Navigate to QQR
        - Edit `setKroneckerToolsPath.m`
        - Edit `setPolynomialSystemsPath.m`

Convenience Functions in `KroneckerTools`

- `[Y] = kroneckerLeft(M,B)`
  Applies $Y = (M \otimes M \otimes \cdots \otimes M)B$ and was used to perform the change of variables.
- `[zd] = KroneckerPower(z,d)`
  Computes $z^{ⓓ}$.
- `[c] = kronMonomialSymmetrize(c,n,d);`
  Symmetrizes monomial coefficients that multiply $\mathbf{z}^{ⓓ}$ ($\mathbf{z} \in \mathbb{R}^n$).
- `[Ns] = kronMatrixSymmetrize(N,n,d);`
  Same as above, but for $\mathbf{N}_2$, $\mathbf{N}_3$, etc.
- `[Ns] = kronNxuSymmetrize(Nxu,n,d);`
  Same as above, but when multiplying $\mathbf{z}^{ⓓ} \otimes \mathbf{u}$.
- `[u] = kronPolyEval(k,z,d);`
  evaluates a polynomial in Kronecker form with coefficients k.

```
u = k{1}*z + k{2}*kron(z,z) + ... + k{d}*kron(kron(...z),z),z)
```

Convenience Functions in `KroneckerTools`

- `[S] = perfectShuffle(p,q)`
  Calculates the (sparse) perfect shuffle matrix.

```
A = rand(m1,n1); B = rand(m2,n2);
Smm = perfectShuffle(m1,m2);  Snn = perfectShuffle(n1,n2);
norm(kron(A,B) - Smm.'*kron(B,A)*Snn)         % should be zero

vec = @(x) x(:);                             % create a convenience function
S = perfectShuffle(m1,n1);
norm(S.'*vec(A) - vec(A.'))                   % should be zero
```

Convenience Functions in `KroneckerTools`

- `[A,B,Nzz,Nzu] = kronPolyApprox(f,g,n,m,d);`

Given a (nice) system of the form $\dot{\mathbf{z}} = \mathbf{f}(\mathbf{z}) + \mathbf{g}(\mathbf{z})u$, where $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n$ and $\mathbf{g} : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ then `kronPolyApprox` uses the symbolic toolbox to provide a polynomial approximation in the desired Kronecker product form.

E.g.

```
f = @(z) [z(1)^2; z(1)*z(2)]; g = @(z) [1+z(1) 2+z(2);3+z(2) 4+z(1)];
```

then `[A,B,Nzz,Nzu] = kronPolyApprox(f,g,2,2,2)` gives matrices `A` and `B` and the expected cell arrays, e.g.

```
>> Nzz{2}
  ans =
    1.0000         0         0         0
         0    0.5000    0.5000         0
```

`QQR` Functions

The `QQR` repository has two primary functions: `pqr` and `pqrBilinear`.

The `pqr` function has the following function call.

```
[k,v] = pqr(A,B,Q,R,N,degree)
```

where `N` is a cell array from `N{2}` through `N{d}` of sizes $n \times n^2$ through $n \times n^d$, respectively.

The outputs are cell arrays of the feedback coefficients `k` and coefficients of the value function approximation `v`.

The `pqrBilinear` function has the following function call.

```
[k,v] = pqrBilinear(A,B,Q,R,Nzz,Nzu,Nuu,degree)
```

where `Nzz` is a cell array from `Nzz{2}` through `Nzz{d}` of sizes $n \times n^2$ through $n \times n^d$, `Nzu` contains a cell array from `Nzu{1}` through `Nzu{q-1}` of sizes $n \times nm$ through $n \times n^{q-1}m$, and `Nuu` is a single matrix of size $n \times m^2$.

The outputs are cell arrays of the feedback coefficients `k` and coefficients of the value function approximation `v`.

Currently, we require `degree<6`

Feedback Control of a Bilinear Chemical Reaction Model

This model is described in *An iterative method for the finite-time bilinear-quadratic control problem*, by Hofer and Tibken, J Optimization Theory and Applications, 57(3), 1988.

Their model, on p. 423 includes temperature (z_1) and concentration (z_2) and is

$$\dot{\mathbf{z}} = \begin{bmatrix} 13/6 & 5/12 \\ -50/3 & -8/3 \end{bmatrix} \mathbf{z} + \begin{bmatrix} -1/8 \\ 0 \end{bmatrix} u + \begin{bmatrix} -1 & 0 \\ 0 & 0 \end{bmatrix} \mathbf{z} \otimes u.$$

They perform design to minimize the LQR cost for $\mathbf{Q} = 10\mathbf{I}_2$ and $R = 1$. Simulations from $\mathbf{z}_0 = [0.15, 0]^\top$ are reported in their paper.

Let's reproduce this for the infinite horizon case here.

Feedback Control of a Bilinear Chemical Reaction Model

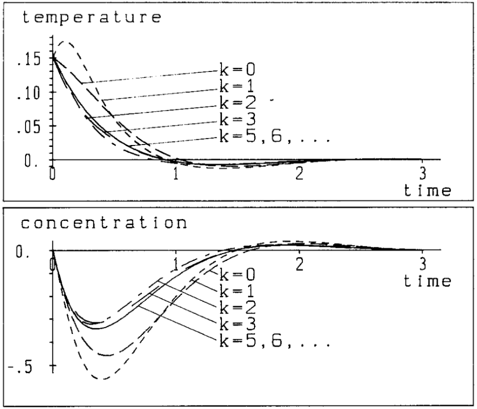
Define the problem

```
>> A = [13/6 5/12;-50/3 -8/3];  B = [-1/8; 0];
>> Nzz{2} = zeros(2,4);         Nzu{1} = [-1 0;0 0];
>> Nuu = zeros(2,1);
>> Q = 10*eye(2);
>> R = 1;
```

Solve the problem

```
>> setPaths
>> [k,v] = pqrBilinear(A,B,Q,R,Nzz,Nzu,Nuu,3);
>>  u = @(z) kronPolyEval(k,z);
>>  rhs = @(t,z) A*z + B*u(z) + Nzu{1}*kron(z,u(z));
>>  [T,Z] = ode23(rhs,[0 3],[.15;0]);
>>  plot(T,Z(:,1))
```

Feedback Control of a Bilinear Chemical Reaction Model



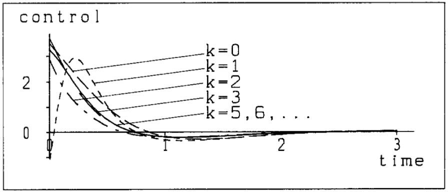
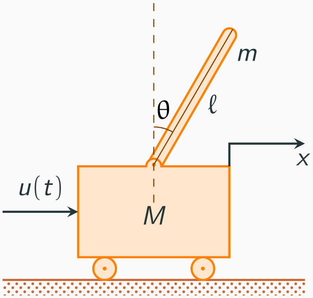


Fig. 4. Profiles for temperature, concentration, and control for $x^0 = (0.15, 0)^T$.

Compare your solution to the final iteration reported in the Hofer and Tibken paper (they are solving the finite horizon problem with large final cost at $t = 3$ and we are solving the infinite horizon problem).

Feedback Control for Cart-Pole System

This is a classic problem in control (see Barto, Sutton, and Anderson, 1983).



modified from `latexdraw.com`

Assuming no friction, the equations of motion can be written as

$$\dot{z} = \underbrace{\begin{bmatrix} z_2 \\ r\ell z_4^2 \sin(z_3) - \frac{r\cos(z_3)(g\sin(z_3) - \ell r z_4^2 \sin(2z_3)/2)}{((4/3) - r\cos^2(z_3))} \\ z_4 \\ \frac{g\sin(z_3) - \ell r z_4^2 \sin(2z_3)/2}{\ell(4/3 - r\cos^2(z_3))} \end{bmatrix}}_{\mathbf{f}(\mathbf{z})} + \mathbf{g}(\mathbf{z})u$$

where $z_1 = x$, $z_3 = \theta$, and $r = \frac{m}{m+M}$.

Feedback Control for Cart-Pole System

We first create an approximate polynomial model for the system

```
[A,B,Nzz,Nzu] = cartpolePolynomial(m,M,L,g);
```

This function has been setup to provide a degree 5 approximation for $\mathbf{f}(\mathbf{z})$ and a degree 2 approximation for $\mathbf{g}(\mathbf{z})$.

We then define the state and control weights $\mathbf{Q}$ and $\mathbf{R}$.

Feedback Control for Cart-Pole System

We first create an approximate polynomial model for the system

```
[A,B,Nzz,Nzu] = cartpolePolynomial(m,M,L,g);
```

This function has been setup to provide a degree 5 approximation for $\mathbf{f}(\mathbf{z})$ and a degree 2 approximation for $\mathbf{g}(\mathbf{z})$.

We then define the state and control weights $\mathbf{Q}$ and $\mathbf{R}$.

The feedback laws are defined using either

```
[K,V] = lqr(A,B,Q,R);
```

or

```
[k,v] = pqrBilinear(A,B,Q,R,Nzz,Nzu,zeros(4,1),3);
```

Feedback Control for Cart-Pole System

We can evaluate the effectiveness of different feedback controls from various initial conditions by defining the control and using one of the built-in Matlab adaptive ODE solvers.

For instance, if functions `f = @(z) ...` and `g = @(z) ...` for the state equation and `u = @(z) ...` for the feedback law are defined, then we could define `rhs = @(t,z) f(z) + g(z)*u(z)` and use

```
[T,Z] = ode45(rhs,[0 tFinal],z0);
```

The ODE solver can be leveraged to estimate the quadratic control cost by adding an additional state:

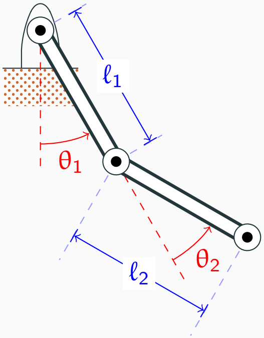
```
rhs = @(t,z) [f(z(1:n)) + g(z(1:n))*u(z(1:n)); ...
              z(1:n).'*Q*z(1:n) + u(z(1:n)).'*R*u(z(1:n))];
```

Then using `[T,Z] = ode45(rhs,[0 tFinal],[z0;0]);` results in `Z(end,end)` being our approximation to the value function.

Feedback Control for Cart-Pole System

Try different feedback laws and different starting points in the script `cartpoleScript.m` located in the `PolynomialSystems` folder.

Another Example: The Acrobot



modified from texample.net

This model is from *Underactuated robotics*, Russ Tedrake, 2023 and has the form

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} = \tau_g(q) + Bu$$

where

$$M(q) = \begin{bmatrix} I_1 + I_2 + m_2\ell_1^2 + m_2\ell_1\ell_2\cos(\theta_2) & I_2 + m_2\ell_1\ell_2\cos(\theta_2)/2 \\ I_2 + m_2\ell_1\ell_2\cos(\theta_2)/2 & I_2 \end{bmatrix},$$

$$C(q,\dot{q}) = \begin{bmatrix} -m_2\ell_1\ell_2\sin(\theta_2)\dot{\theta}_2 & -m_2\ell_1\ell_2\sin(\theta_2)\dot{\theta}_2/2 \\ m_2\ell_1\ell_2\sin(\theta_2)\dot{\theta}_1/2 & 0 \end{bmatrix},$$

$$\tau_g(q) = \begin{bmatrix} -m_1 g\ell_1\sin(\theta_1)/2 - m_2 g(\ell_1\sin(\theta_1) + \ell_2\sin(\theta_1+\theta_2)/2) \\ -m_2 g\ell_2\sin(\theta_1+\theta_2)/2 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

The actuation is a torque applied at the joint between links 1 and 2.

The system has 4 equilibrium points (3 are unstable).

The Acrobot

Similarly, there is a script that builds the polynomial system for this model of the form

$$\dot{\mathbf{z}} = \mathbf{A}\mathbf{z} + \mathbf{B}\mathbf{u} + \mathbf{N}_{zz}\{3\}\mathbf{z}^{③} + \mathbf{N}_{\mathbf{zu}}\{\mathbf{2}\}\left(\mathbf{z}^{②} \otimes \mathbf{u}(\mathbf{t})\right)$$

where the polynomial system can be linearized about any of the equilibrium points:

```
[A,B,Nzz,Nzu] = acrobotPolynomial(ze,parameters);
```

Since the equilibrium point is not necessarily $\mathbf{0}$, when we apply feedback on the full model, the control has the form `u = @(z) kronPolyEval(k,z(1:4)-ze);`

The script `acrobotScript` has suggestions for the initial conditions that are within the stability region for the closed-loop equations.

Test this out for several initial conditions and equilibrium points.

Conclusions of `QQR` software

- In a number of examples, higher degree feedback expanded the radius of convergence for the closed-loop system. Sometime, it only improves the quality of the feedback law very locally.
- This enables higher degree feedback computation in many mathematical models of interest, including flow control problems using ROMs.

Overview of NLbalancing

Functions in the NLbalancing repository are available for polynomial approximations to past and future energy functions, respectively

$$\mathcal{E}_\gamma^-(\mathbf{z}) \approx \mathbf{v}_2^\top \mathbf{z}^{②} + \cdots + \mathbf{v}_\mathbf{d}^\top \mathbf{z}^{ⓓ}$$

and

$$\mathcal{E}_\gamma^+(\mathbf{z}) \approx \mathbf{w}_2^\top \mathbf{z}^{②} + \cdots + \mathbf{w}_\mathbf{d}^\top \mathbf{z}^{ⓓ}.$$

These are used as follows

```
[v] = approxPastEnergy(A,N,B,C,eta,d);
```

and

```
[w] = approxFutureEnergy(A,N,B,C,eta,d);
```

Overview of NLbalancing

A polynomial approximation to the input-normal transformation is computed using

```
[sigma,T] = inputNormalTransformation(v,w,degree);
```

and finally, the approximation to the singular value functions is

```
[c] = approximateSingularValueFunctions(T,w,sigma,maxDegree-1);
```