Probabilistic Robotics: Planning & Control

METR 4202: **Robotics** & Automation

Dr Surya Singh -- Lecture # 12

October 19, 2016

metr4202@itee.uq.edu.au
http://robotics.itee.uq.edu.au/~metr4202/ [http://**metr4202.com**]

Schedule of Events

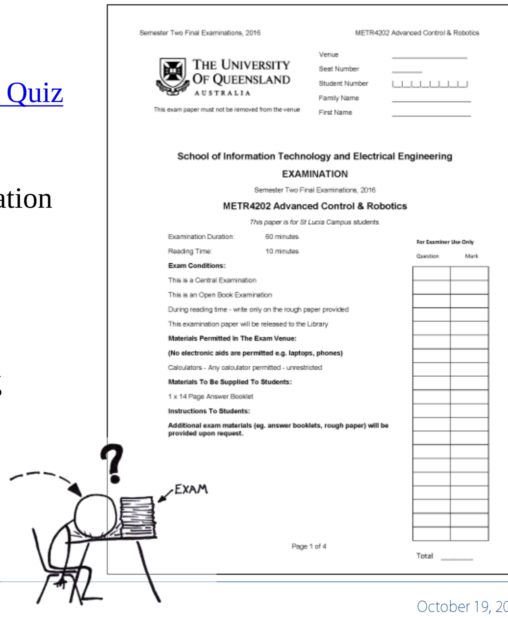
Week	Date	Lecture (W: 12:05-1:50, 50-N202)
1	27-Jul	Introduction
2	3-Aug	Representing Position & Orientation & State (Frames, Transformation Matrices & Affine Transformations)
3	10-Aug	Robot Kinematics Review (& *Ekka Day*)
4	17-Aug	Robot Inverse Kinematics & Kinetics
5	24-Aug	Robot Dynamics (Jacobeans)
6	31-Aug	Robot Sensing: Perception & Linear Observers
7	7-Sep	Robot Sensing: Single View Geometry & Lines
8	14-Sep	Robot Sensing: Feature Detection
9	21-Sep	Robot Sensing: Multiple View Geometry
	28-Sep	*Study break*
10	5-Oct	Motion Planning
11	12-Oct	Probabilistic Robotics: Localization & SLAM
12	**19-Oct**	**Probabilistic Robotics: Planning & Control (State-Space/Shaping the Dynamic Response/LQR)**
13	26-Oct	The Future of Robotics/Automation + Challenges + Course Review

Final Exam!

- 4 Questions | 60 Minutes
- **Open Book**
- Similar in nature to the [2015 Quiz]

Topics:

- Position, orientation and location in space
- Robot analysis (forward/Inverse kinematics, recursive Newton-Euler formulations, etc.)
- Sensing geometry (including camera calibration)
- Multiple-view geometry
- Motion planning and control


Semester Two Final Examinations, 2016 METR4202 Advanced Control & Robotics

THE UNIVERSITY OF QUEENSLAND AUSTRALIA

Venue
Seat Number
Student Number
Family Name
First Name

This exam paper must not be removed from the venue

School of Information Technology and Electrical Engineering

EXAMINATION

Semester Two Final Examinations, 2016

METR4202 Advanced Control & Robotics

This paper is for St Lucia Campus students

Examination Duration: 60 minutes
Reading Time: 10 minutes

Exam Conditions:
This is a Central Examination
This is an Open Book Examination
During reading time - write only on the rough paper provided
This examination paper will be released to the Library

Materials Permitted In The Exam Venue:
(No electronic aids are permitted e.g. laptops, phones)
Calculators - Any calculator permitted - unrestricted

Materials To Be Supplied To Students:
1 x 14 Page Answer Booklet

Instructions To Students:
Additional exam materials (eg. answer booklets, rough paper) will be provided upon request.

For Examiner Use Only

Question	Mark

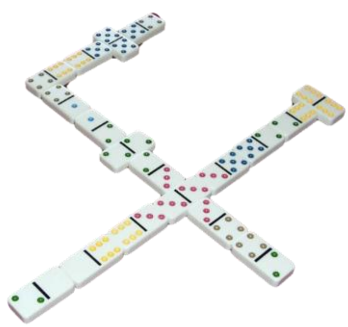
Page 1 of 4

Total

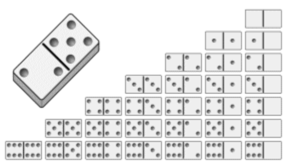
Lab 3: Robotics of a Domino Sort

I. Playing Dominoes of a Sort

- Option 2: Play Dominos



Source https://en.wikipedia.org/wiki/Muggins#/media/File:Muggins.jpg



Cool Robotics Share

[White House AI Symposium]

- Report on the opportunities, considerations, challenges of AI

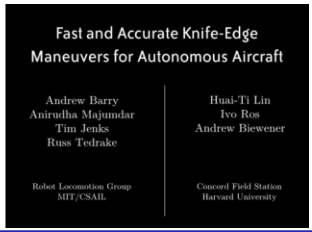
(Oct 12, 2016)

- Related Event (similar thread): ETHZ **Cybathlon**

Source: https://www.whitehouse.gov/blog/2016/10/12/administrations-report-future-artificial-intelligence | http://www.nbcnews.com/health/health-news/brain-chip-helps-paralyzed-man-feel-his-fingers-n665881

Planning & Control Robotics

- **Obstacles** + **Dynamics**



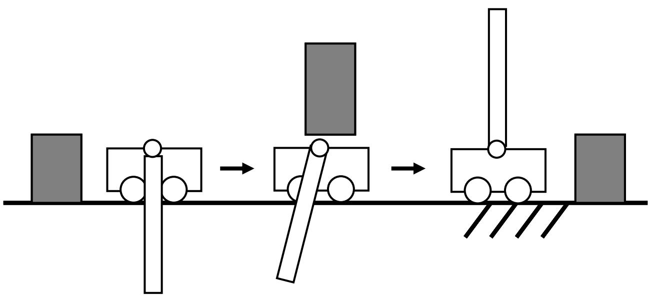
- Partial collisions allowed!

(Oct 14, 2016)

Source: (top) https://youtu.be/voN9CCmzxYk | (bottom) https://youtu.be/LS8sa42xevA

Integrated Planning and Control Methods…

- A motivating problem (for agility)
  – Cart and pole in a cluttered workspace …



Outline

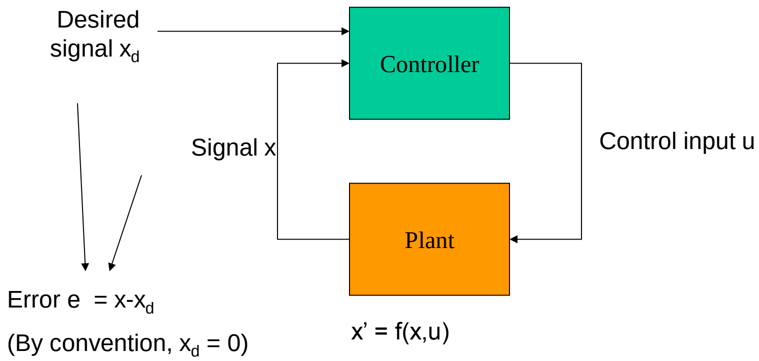
1. Guest Presentation:
   Josh Song, CHARM Planning
2. Probabilistic Robotics
3. Sample-Based Planning
4. Control (State-Space | Shaping Response | LQR)
5. Integrated Planning & Control

Control

(Feedback is our friend!)

Control Theory

- The use of feedback to regulate a signal

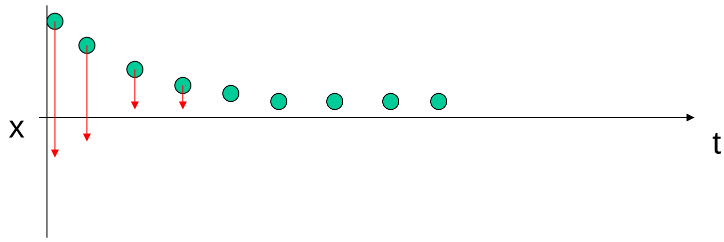


Model-free vs model-based

- Two general philosophies:
  - Model-free: do not require a dynamics model to be provided
  - Model-based: do use a dynamics model during computation
- Model-free methods:
  - Simpler (eg. **PID**)
  - Tend to require much more manual tuning to perform well
- Model-based methods:
  - Can achieve good performance (optimal w.r.t. some cost function)
  - Are more complicated to implement
  - Require reasonably good models (system-specific knowledge)
  - Calibration: build a model using measurements before behaving
  - Adaptive control: “learn” parameters of the model online from sensors

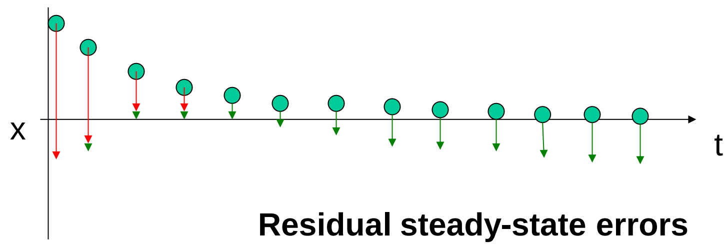
PID control

- Proportional-Integral-Derivative controller
  - A workhorse of 1D control systems
  - Model-free
- Proportional Case:
  - $u(t) \underbrace{=}_{\text{Gain}} -K_p\, x(t)$
  - Negative sign assumes control acts in the same direction as x



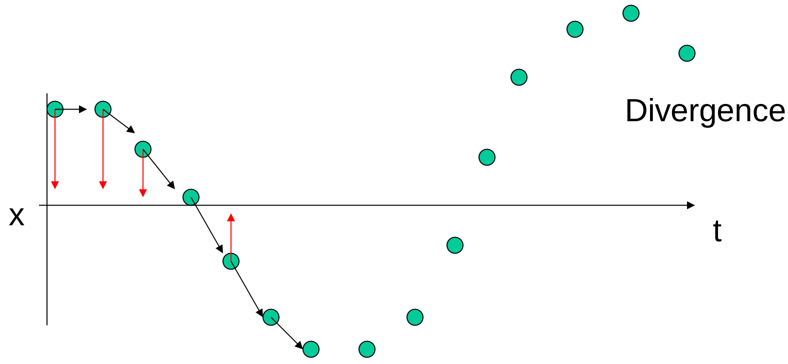
PID control: Integral term

- $u(t) = -K_p\, x(t) - \underbrace{K_i}_{\text{Integral gain}}\, I(t)$
- $I(t) = \int_0^t x(t)dt$ (accumulation of errors)



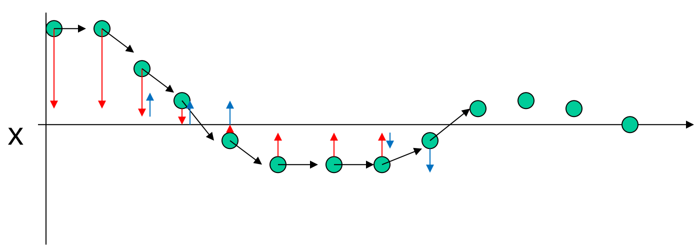
PID control: Integral term: Instability

- I adds a pole
- If not tuned correctly → this adds instability
- Ex: For a 2nd order system (momentum), P control



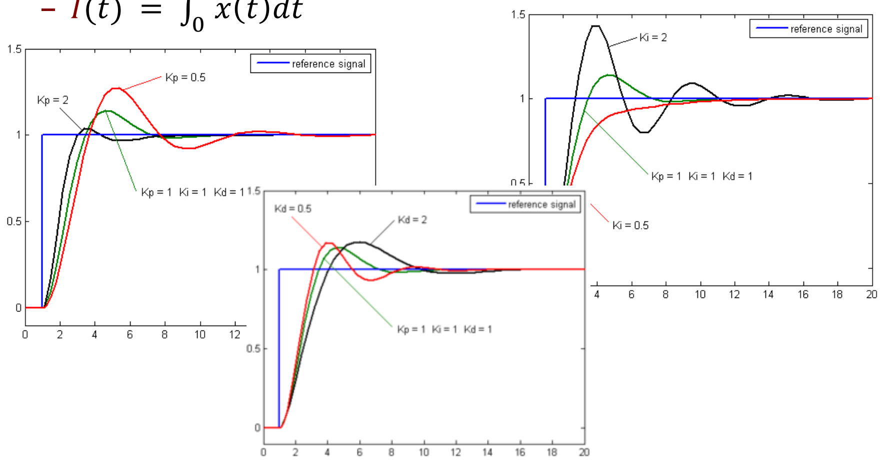
PID control: Derivative term

- $u(t) = -K_p\, x(t) - \underbrace{K_d}_{\text{Derivative gain}}\, x'(t)$



PID control: Together

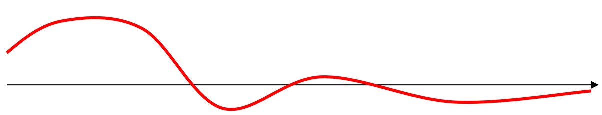
- P+I+D:
  - $u(t) = -K_p\, x(t) - K_i\, I(t) - K_d\, x'(t)$
  - $I(t) = \int_0^t x(t)dt$



Stability and Convergence

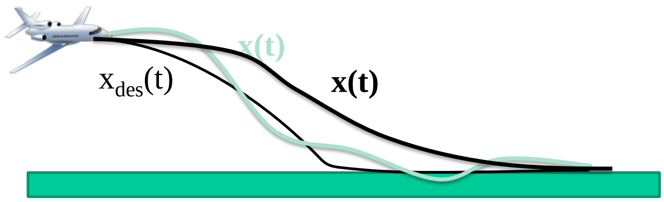
- System is *stable* if errors stay bounded

- System is *convergent* if errors -> 0



Example: Trajectory following

- Say a trajectory $x_{des}(t)$ has been designed
  - E.g., a rocket's ascent, a steering path for a car, a plane's landing
- Apply PID control
  - $u(t) = K_p (x_{des}(t) - x(t)) - K_i I(t) + K_d (x'_{des}(t) - x'(t))$
  - $I(t) = \int_0^t x_{des}(t) - x(t) dt$
- The designer of x_{des} needs to be knowledgeable about the controller's behavior!



Controller Tuning Workflow

- Hypothesize a control policy
- Analysis:
  - Assume a model
  - Assume disturbances to be handled
  - Test performance either through **mathematical analysis**, or through **simulation**
- Go back and redesign control policy
  - Mathematical techniques give you more insight to improve redesign, but require more work

Multivariate Systems

What about more than more interacting aspect?

$$x' = f(x, u)$$
$$x \in \mathbf{X} \subseteq \mathbb{R}^n$$
$$u \in \mathbf{U} \subseteq \mathbb{R}^m$$

- Note: $m \neq n$ *and* variables are <u>**coupled**</u>

→ This is not as easy as setting n PID controllers

➔ Derive a "space" of controllers??

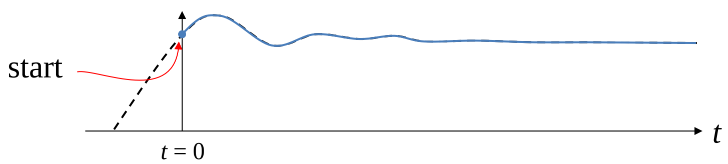
State-Space Modelling (ELEC3004 Super-Summary!)

("Hear Ye! It be stated")

Affairs of state

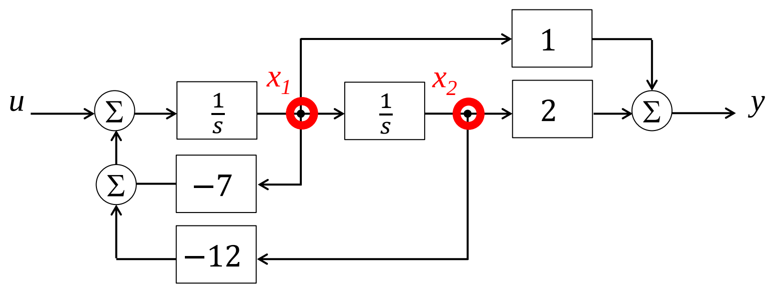
- Introductory brain-teaser:
  - If you have a dynamic system model with history (ie. integration) how do you represent the instantaneous state of the plant?

Eg. how would you setup a simulation of a step response, mid-step?



Introduction to state-space

- We can identify the nodes in the system
  - These nodes contain the integrated time-history values of the system response
  - We call them “states”



Linear system equations

- We can represent the dynamic relationship between the states with a linear system:

$$\begin{aligned} \dot{x_1} &= -7x_1 - 12x_2 + u \\ \dot{x_2} &= x_1 + 0x_2 + 0u \\ y &= x_1 + 2x_2 + 0u \end{aligned}$$

State variable transformation

- Important note!
  - The states of a control canonical form system are not the same as the modal states
  - They represent the same dynamics, and give the same output, but the vector values are different!
- However we can convert between them:
  - Consider state representations, $\boldsymbol{x}$ and $\boldsymbol{q}$ where

$$\boldsymbol{x} = \mathbf{T}\boldsymbol{q}$$

T is a "transformation matrix"

State variable transformation

- Two homologous representations:

$$\dot{\boldsymbol{x}} = \mathbf{A}\boldsymbol{x} + \mathbf{B}u \quad \text{and} \quad \dot{\boldsymbol{q}} = \mathbf{F}\boldsymbol{q} + \mathbf{G}u$$
$$y = \mathbf{C}\boldsymbol{x} + Du \qquad\qquad y = \mathbf{H}\boldsymbol{q} + Ju$$

We can write:

$$\dot{\boldsymbol{x}} = \mathbf{T}\dot{\boldsymbol{q}} = \mathbf{A}\mathbf{T}\boldsymbol{z} + \mathbf{B}u$$
$$\dot{\boldsymbol{q}} = \mathbf{T}^{-1}\mathbf{A}\mathbf{T}\boldsymbol{z} + \mathbf{T}^{-1}\mathbf{B}u$$

Therefore, $\mathbf{F} = \mathbf{T}^{-1}\mathbf{A}\mathbf{T}$ and $\mathbf{G} = \mathbf{T}^{-1}\mathbf{B}$

Similarly, $\mathbf{C} = \mathbf{H}\mathbf{T}$ and $D = J$

Example: (Back To) Robot Arms

Slides 17-27 Source: R. Lindeke, ME 4135, "Introduction to Control"

Remembering the Motion Models:

- Recall from Dynamics, the Required Joint Torque is:

$$\tau_i = D_i\left(q\right)\ddot{q}_i + C_i\left(q, \dot{q}_i\right) + h\left(q\right) + b\left(\dot{q}_i\right)$$

Dynamical Manipulator Inertial Tensor – a function of position and acceleration

Coupled joint effects (centrifugal and coriolis) issues due to multiple moving joints

Gravitational Effects

Frictional Effect due to Joint/Link movement

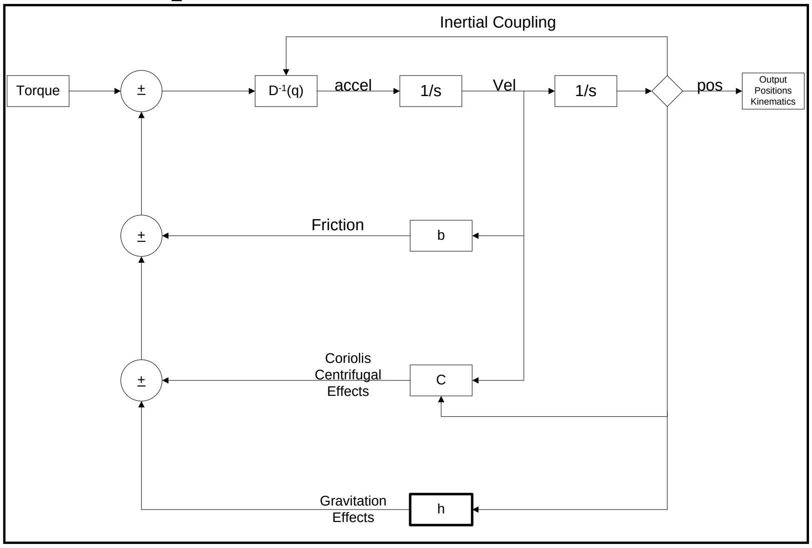
Lets simplify the model

- This torque model is a 2nd order one (in position) lets look at it as a velocity model rather than positional one then it becomes a system of highly coupled 1st order differential equations

- We will then isolate Acceleration terms (acceleration is the 1st derivative of velocity)

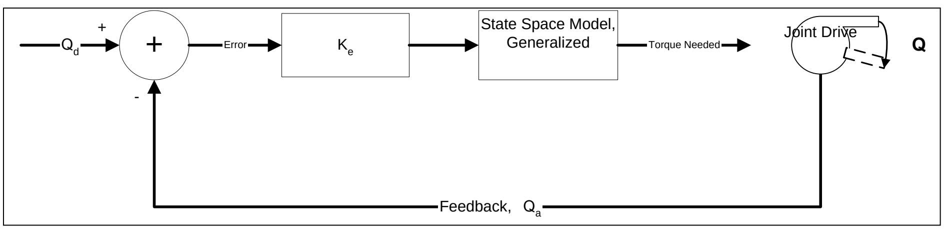
$$a = \dot{v} = \ddot{q} = D_i^{-1}(q)\left(\tau_i - C_i(q, \dot{q}_i) - h(q) - b\left(\dot{q}_1\right)\right)$$

The State-Space Control Model:



Setting up a Real Control

- We will (start) by using positional error to drive our torque devices



- This simple model is called a PE (proportional error) controller

PE Controller:

- To a 1st approximation, $\tau = K_m * I$
  - Torque is proportional to motor current

- And the Torque required is a function of 'Inertial' (Acceleration) and 'Friction' (velocity) effects as suggested by our L-E models

$$\tau_m \simeq J_{eq}\ddot{q} + F_{eq}\dot{q}$$

→ Which can be approximated as:

$$K_m I_m = J_{eq}\ddot{q} + F_{eq}\dot{q}$$

Setting up a "Control Law"

- We will use the positional error (as drawn in the state model) to develop our torque control
- We say then for PE control:

$$\tau \propto k_{pe}(\theta_d - \theta_a)$$

- Here, k_{pe} is a "gain" term that guarantees sufficient current will be generated to develop appropriate torque based on observed positional error

Using this Control Type:

- It is a representation of the physical system of a mass on a spring!
- We say after setting our target as a ‘zero goal’ that:

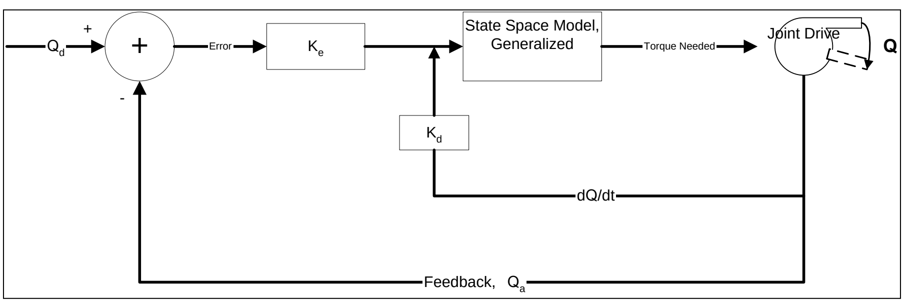
$$-k_{pe} * \theta_a = J\ddot{\theta} + F\dot{\theta}$$

the solution of which is:

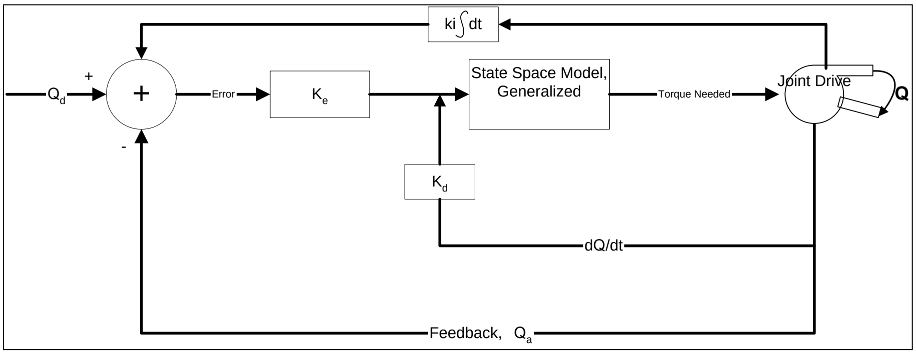
θ_a is a function of the servo feedback as a function of time!

$$\theta_a = e^{-\left(F/2J\right)t}\left[C_1 e^{\left(1/2\right)\omega t} + C_2 e^{-\left(1/2\right)\omega t}\right]$$

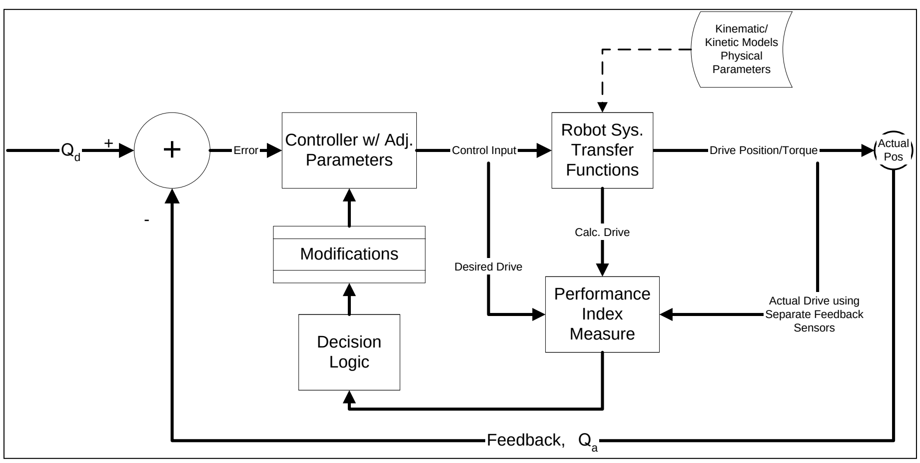
State Space Model of PD:



PID State Space Model:



State Model of Adjustable Controller



State-Space Control Design

AKA, it's all about $\vec{\mathbf{u}}$ ☺

Punchline: it's an Optimization
*(Algebraic Riccati equation → Nonlinear optimization → **Search!** …*
Just like SLAM | Motion Planning!!)

State-space control design

- Design for discrete state-space systems is just like the continuous case.
  - Apply linear state-variable feedback:

$$u = -\mathbf{K}\boldsymbol{x}$$

such that $\det(z\mathbf{I} - \boldsymbol{\Phi} + \boldsymbol{\Gamma}\mathbf{K}) = \alpha_c(z)$

where $\alpha_c(z)$ is the desired control characteristic equation

Predictably, this requires the system controllability matrix $\mathcal{C} = [\boldsymbol{\Gamma} \quad \boldsymbol{\Phi\Gamma} \quad \boldsymbol{\Phi}^2\boldsymbol{\Gamma} \quad \cdots \quad \boldsymbol{\Phi}^{n-1}\boldsymbol{\Gamma}]$ to be full-rank.

Great, so how about control?

- Given $\dot{\boldsymbol{x}} = \mathbf{F}\boldsymbol{x} + \mathbf{G}u$, if we know $\mathbf{F}$ and $\mathbf{G}$, we can design a controller $u = -\mathbf{K}\boldsymbol{x}$ such that

$$\mathrm{eig}(\mathbf{F} - \mathbf{GK}) < 0$$

- In fact, if we have full measurement and control of the states of $\boldsymbol{x}$, we can position the poles of the system in arbitrary locations!

(Of course, that never happens in reality.)

Example: PID control

- Consider a system parameterised by three states:
  - x_1, x_2, x_3
  - where $x_2 = \dot{x}_1$ and $x_3 = \dot{x}_2$

$$\dot{\boldsymbol{x}} = \begin{bmatrix} 1 & & \\ & 1 & \\ & & -2 \end{bmatrix} \boldsymbol{x} - \mathbf{K}u$$

$$y = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \boldsymbol{x} + 0u$$

x_2 is the output state of the system;

x_1 is the value of the integral;

x_3 is the velocity.

- We can choose **K** to move the eigenvalues of the system as desired:

$$\det\begin{bmatrix}1-K_1 & & \\ & 1-K_2 & \\ & & -2-K_3\end{bmatrix} = \mathbf{0}$$

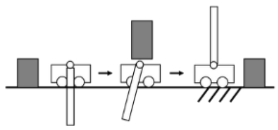
All of these eigenvalues must be positive.

It's straightforward to see how adding derivative gain K_3 can stabilise the system.

Example: Inverted Pendulum

Digital Control

Wikipedia,
Cart and pole



$$L = \frac{1}{2}Mv_1^2 + \frac{1}{2}mv_2^2 - mg\ell\cos\theta$$

where v_1 is the velocity of the cart and v_2 is the velocity of the point mass m. v_1 and v_2 can be expressed in terms of x and θ by writing the velocity as the first derivative of the position;

$$v_1^2 = \dot{x}^2$$
$$v_2^2 = \left(\frac{d}{dt}(x - \ell\sin\theta)\right)^2 + \left(\frac{d}{dt}(\ell\cos\theta)\right)^2$$

Simplifying the expression for v_2 leads to:

$$v_2^2 = \dot{x}^2 - 2\ell\dot{x}\dot{\theta}\cos\theta + \ell^2\dot{\theta}^2$$

The Lagrangian is now given by:

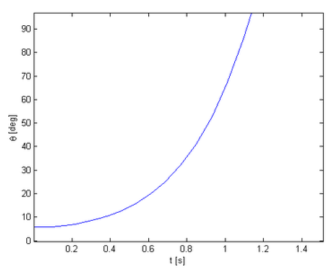
$$L = \frac{1}{2}(M+m)\dot{x}^2 - m\ell\dot{x}\dot{\theta}\cos\theta + \frac{1}{2}m\ell^2\dot{\theta}^2 - mg\ell\cos\theta$$

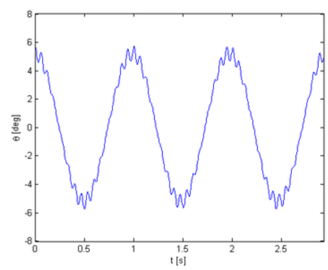
and the equations of motion are:

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{x}} - \frac{\partial L}{\partial x} = F$$
$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{\theta}} - \frac{\partial L}{\partial \theta} = 0$$

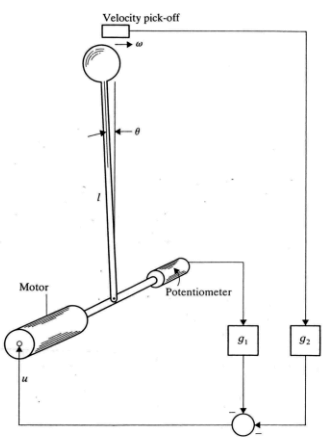
substituting L in these equations and simplifying leads to the equations that describe the motion o

$$(M+m)\ddot{x} - m\ell\ddot{\theta}\cos\theta + m\ell\dot{\theta}^2\sin\theta = F$$
$$\ell\ddot{\theta} - g\sin\theta = \ddot{x}\cos\theta$$





Inverted Pendulum



$$L = \frac{1}{2}Mv_1^2 + \frac{1}{2}mv_2^2 - mg\ell\cos\theta$$

where v_1 is the velocity of the cart and v_2 is the velocity of the point mass m. v_1 and v_2 can be expressed in terms of x and θ by writing the velocity as the first derivative of the position;

$$v_1^2 = \dot{x}^2$$
$$v_2^2 = \left(\frac{d}{dt}(x - \ell\sin\theta)\right)^2 + \left(\frac{d}{dt}(\ell\cos\theta)\right)^2$$

Simplifying the expression for v_2 leads to:

$$v_2^2 = \dot{x}^2 - 2\ell\dot{x}\dot{\theta}\cos\theta + \ell^2\dot{\theta}^2$$

The Lagrangian is now given by:

$$L = \frac{1}{2}(M+m)\dot{x}^2 - m\ell\dot{x}\dot{\theta}\cos\theta + \frac{1}{2}m\ell^2\dot{\theta}^2 - mg\ell\cos\theta$$

and the equations of motion are:

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{x}} - \frac{\partial L}{\partial x} = F$$
$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{\theta}} - \frac{\partial L}{\partial \theta} = 0$$

substituting L in these equations and simplifying leads to the equations that describe the motion o

$$(M+m)\ddot{x} - m\ell\ddot{\theta}\cos\theta + m\ell\dot{\theta}^2\sin\theta = F$$
$$\ell\ddot{\theta} - g\sin\theta = \ddot{x}\cos\theta$$

Inverted Pendulum – Equations of Motion

- The equations of motion of an inverted pendulum (under a small angle approximation) may be linearized as:

$$\dot{\theta} = \omega$$
$$\dot{\omega} = \ddot{\theta} = Q^2\theta + Pu$$

Where:

$$Q^2 = \left(\frac{M+m}{Ml}\right)g$$
$$P = \frac{1}{Ml}.$$

If we further assume unity ***Ml*** ($Ml \approx 1$), then $P \approx 1$

Inverted Pendulum –State Space

- We then select a state-vector as:

$$\boldsymbol{x} = \begin{bmatrix} \theta \\ \omega \end{bmatrix}, \text{ hence } \dot{\boldsymbol{x}} = \begin{bmatrix} \dot{\theta} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} \omega \\ \dot{\omega} \end{bmatrix}$$

- Hence giving a state-space model as:

$$A = \begin{bmatrix} 0 & 1 \\ Q^2 & 0 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

- The resolvent of which is:

$$\Phi(s) = (sI - A)^{-1} = \begin{bmatrix} s & -1 \\ -Q^2 & s \end{bmatrix}^{-1} = \frac{1}{s^2 - Q^2}\begin{bmatrix} s & 1 \\ Q^2 & s \end{bmatrix}$$

- And a state-transition matrix as:

$$\Phi(t) = \begin{bmatrix} \cosh Qt & \dfrac{\sinh Qt}{Q} \\ Q\, sinh\, Qt & \cosh Qt \end{bmatrix}$$

Shaping of Dynamic Responses

ELEC3004 Flashback: Another way to see P I|D

- Derivative

  D provides:
  - High sensitivity
  - Responds to change
  - Adds "damping" & ∴ permits larger K_P
  - Noise sensitive
  - Not used alone

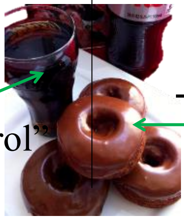
    (∵ its on rate change of error – by itself it wouldn't get there)

→ "**D**iet Coke of control"

- Integral
  - Eliminates offsets (makes regulation ☺)
  - Leads to Oscillatory behaviour
  - Adds an "order" but instability

    (Makes a 2nd order system 3rd order)

→ "**I**nteresting cake of control"



PID control

- Consider a system parameterised by three states:
  - x_1, x_2, x_3
  - where $x_2 = \dot{x}_1$ and $x_3 = \dot{x}_2$

$$\dot{\boldsymbol{x}} = \begin{bmatrix} 1 & & \\ & 1 & \\ & & -2 \end{bmatrix} \boldsymbol{x} - \mathbf{K}u$$

$$y = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \boldsymbol{x} + 0u$$

x_2 is the output state of the system;

x_1 is the value of the integral;

x_3 is the velocity.

PID control [2]

- We can choose **K** to move the eigenvalues of the system as desired:

$$\det \begin{bmatrix} 1 - K_1 & & \\ & 1 - K_2 & \\ & & -2 - K_3 \end{bmatrix} = \mathbf{0}$$

All of these eigenvalues must be positive.

It's straightforward to see how adding derivative gain K_3 can stabilise the system.

Implementation of Digital PID Controllers

We will consider the PID controller with an s-domain transfer function

$$\frac{U(s)}{X(s)} = G_c(s) = K_P + \frac{K_I}{s} + K_D s. \tag{13.54}$$

We can determine a digital implementation of this controller by using a discrete approximation for the derivative and integration. For the time derivative, we use the **backward difference rule**

$$u(kT) = \left.\frac{dx}{dt}\right|_{t=kT} = \frac{1}{T}(x(kT) - x[(k-1)T]). \tag{13.55}$$

The z-transform of Equation (13.55) is then

$$U(z) = \frac{1 - z^{-1}}{T} X(z) = \frac{z - 1}{Tz} X(z).$$

The integration of $x(t)$ can be represented by the **forward-rectangular integration** at $t = kT$ as

$$u(kT) = u[(k-1)T] + Tx(kT), \tag{13.56}$$

Source: Dorf & Bishop, Modern Control Systems, §13.9, pp. 1030-1

Implementation of Digital PID Controllers (2)

where $u(kT)$ is the output of the integrator at $t = kT$. The z-transform of Equation (13.56) is

$$U(z) = z^{-1}U(z) + TX(z),$$

and the transfer function is then

$$\frac{U(z)}{X(z)} = \frac{Tz}{z - 1}.$$

Hence, the z-domain transfer function of the **PID controller** is

$$\boxed{G_c(z) = K_P + \frac{K_I T z}{z - 1} + K_D \frac{z - 1}{Tz}.} \tag{13.57}$$

The complete difference equation algorithm that provides the PID controller is obtained by adding the three terms to obtain [we use $x(kT) = x(k)$]

$$\begin{aligned} u(k) &= K_P x(k) + K_I[u(k-1) + Tx(k)] + (K_D/T)[x(k) - x(k-1)] \\ &= [K_P + K_I T + (K_D/T)]x(k) - K_D T x(k-1) + K_I u(k-1). \end{aligned} \tag{13.58}$$

Equation (13.58) can be implemented using a digital computer or microprocessor. Of course, we can obtain a PI or PD controller by setting an appropriate gain equal to zero.

Source: Dorf & Bishop, Modern Control Systems, §13.9, pp. 1030-1

Let's Generalize This: Shaping the Dynamic Response

- A method of designing a control system for a process in which all the state variables are accessible for Measurement

➔ This method is also known as ***pole-placement***

- Theory:
  - We will find that in a controllable system, with all the state variables accessible for measurement, it is possible to place the closed-loop poles anywhere we wish in the complex *s* plane!
- Practice:
  - Unfortunately, however, what can be attained in principle may not be attainable in practice. Speeding the response of a sluggish system requires the use of large control signals which the actuator (or power supply) may not be capable of delivering. And, control system gains are **very sensitive** to the location of the open-loop poles

Pole Placement

Pole Placement (Following FPW – Chapter 6)

- FPW has a slightly different notation:

$$\dot{\mathbf{x}} = \mathbf{F}\mathbf{x} + \mathbf{G}u,$$
$$y = \mathbf{H}\mathbf{x}.$$

$$\mathbf{x}(k+1) = \mathbf{\Phi}\mathbf{x}(k) + \mathbf{\Gamma}u(k),$$
$$y(k) = \mathbf{H}\mathbf{x}(k),$$

$$\mathbf{\Phi} = e^{\mathbf{F}T},$$
$$\mathbf{\Gamma} = \int_0^T e^{\mathbf{F}\eta} d\eta \mathbf{G},$$

Pole Placement

- Start with a simple feedback control law ("controller")

$$u = -\mathbf{K}\mathbf{x} = -[K_1 K_2 \dots] \begin{bmatrix} x_1 \\ x_2 \\ \vdots \end{bmatrix}$$

- It's actually a regulator

  ∵ it does not allow for a reference input to the system.
  (there is no "reference" **r** (r = 0))

- Substitute in the difference equation

$$x(k+1) = \Phi x(k) - \Gamma K x(k)$$

- $\mathcal{Z}$ Transform:

$$(zI - \Phi + \Gamma K)X(z) = 0$$

➔Characteristic Eqn:

$$\det|zI - \Phi + \Gamma K| = 0$$

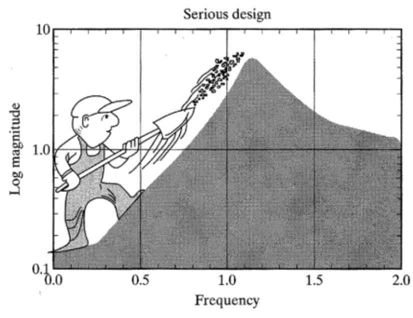
Pole Placement

Pole placement: Big idea:

- Arbitrarily select the desired root locations of the closed-loop system and see if the approach will work.
- AKA: full state feedback
  - ∵ enough parameters to influence all the closed-loop poles
- Finding the elements of K so that the roots are in the desired locations. Unlike classical design, where we iterated on parameters in the compensator (hoping) to find acceptable root locations, the full state feedback, pole-placement approach guarantees success and allows us to arbitrarily pick any root locations, providing that n roots are specified for an n^{th}-order system.

Meaning – No Free Lunch

- The energy (and sensitivity) moves around (in this case in "frequency")



- Sensitivity reduction at low frequency unavoidably leads to sensitivity increase at higher frequencies.

Source: Gunter Stein's interpretation of the water bed effect – G. Stein, *IEEE Control Systems Magazine*, 2003.

Back to Pole Placement

- Given:

$$z_i = \beta_1, \beta_2, \beta_3, \ldots$$

- This gives the desired control-characteristic equation as:

$$a_c(z) = (z - \beta_1)(z - \beta_2)(z - \beta_3) \ldots =$$

- Now we "just solve" for **K** and "bingo"

Pole Placement Example (FPW p. 241)

Example 6.1: Suppose we want to design a control law for the satellite attitude-control system described by (2.45) with $\boldsymbol{x} = [\boldsymbol{x}_1 \;\; \boldsymbol{x}_2]$. Example 2.13 showed that the discrete model for this system is

$$\boldsymbol{\Phi} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \quad \text{and} \quad \boldsymbol{\Gamma} = \begin{bmatrix} T^2/2 \\ T \end{bmatrix}.$$

We want to pick z-plane roots of the closed-loop characteristic equation so that the equivalent s-plane roots have a damping ratio of $\zeta = 0.5$ and real part of $s = -1.8$ rad/sec (i.e., $s = -1.8 \pm j3.12$ rad/sec). Using $z = e^{sT}$ with a sample period of $T = 0.1$ sec, we find that $z = 0.8 \pm j0.25$, as shown in Fig. 6.1. The desired characteristic equation is then

$$z^2 - 1.6z + 0.70 = 0, \tag{6.9}$$

and the evaluation of (6.7) for any control law $\mathbf{K}$ leads to

$$\det \left| z \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} T^2/2 \\ T \end{bmatrix} [K_1 \quad K_2] \right| = 0$$

or

$$z^2 + (TK_2 + (T^2/2)K_1 - 2)z + (T^2/2)K_1 - TK_2 + 1 = 0. \tag{6.10}$$

Pole Placement Example (FPW p. 241)

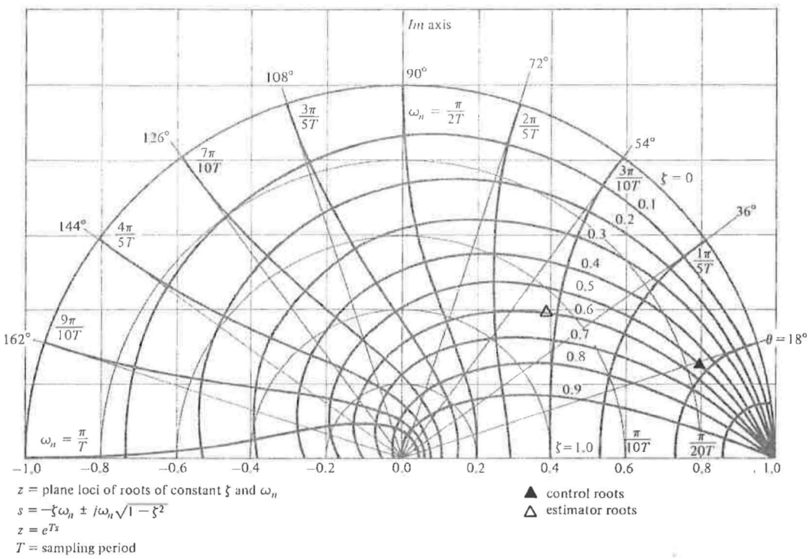
Equating coefficients in (6.9) and (6.10) with like powers of z, we obtain two simultaneous equations in the two unknown elements of $\mathbf{K}$:

$$TK_2 + (T^2/2)K_1 - 2 = -1.6,$$
$$(T^2/2)K_1 - TK_2 + 1 = 0.70,$$

which are easily solved for the coefficients and evaluated for $T = 0.1$ sec:

$$K_1 = \frac{0.10}{T^2} = 10, \qquad K_2 = \frac{0.35}{T} = 3.5.$$

Pole Placement Example (FPW p. 241)



Ackermann's Formula (FPW p. 245)

- Gains maybe approximated with:

$$\boxed{\mathbf{K} = [0 \ldots 0 \quad 1][\mathbf{\Gamma} \quad \mathbf{\Phi\Gamma} \quad \mathbf{\Phi}^2\mathbf{\Gamma} \ldots \mathbf{\Phi}^{n-1}\mathbf{\Gamma}]^{-1}\alpha_c(\mathbf{\Phi}),}$$

- Where: C = controllability matrix, *n* is the order of the system (or number of state elements) and α_c:

$$\mathcal{C} = [\mathbf{\Gamma} \quad \mathbf{\Phi\Gamma} \ldots]$$

$$\alpha_c(\mathbf{\Phi}) = \mathbf{\Phi}^n + \alpha_1\mathbf{\Phi}^{n-1} + \alpha_2\mathbf{\Phi}^{n-2} + \cdots + \alpha_n\mathbf{I},$$

$$\alpha_c(z) = |z\mathbf{I} - \mathbf{\Phi} + \mathbf{\Gamma K}| = z^n + \alpha_1 z^{n-1} + \cdots + \alpha_n.$$

  - α_i: coefficients of the desired characteristic equation

Ackermann's Formula Example (FPW p.246)

Example 6.2: Applying Ackermann's formula to the satellite attitude-control system of Example 6.1, we find from (6.9) that

$$\alpha_1 = -1.6, \qquad \alpha_2 = +0.70,$$

and therefore

$$\alpha_c(\mathbf{\Phi}) = \begin{bmatrix} 1 & 2T \\ 0 & 1 \end{bmatrix} - 1.6\begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} + 0.70\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.1 & 0.4T \\ 0 & 0.1 \end{bmatrix}.$$

Furthermore, we find that

$$[\mathbf{\Gamma} \quad \mathbf{\Phi\Gamma}] = \begin{bmatrix} T^2/2 & 3T^2/2 \\ T & T \end{bmatrix}$$

and

$$[\mathbf{\Gamma} \quad \mathbf{\Phi\Gamma}]^{-1} = 1/T^2\begin{bmatrix} -1 & +3T/2 \\ 1 & -T/2 \end{bmatrix},$$

and finally

$$\mathbf{K} = [K_1 \quad K_2] = (1/T^2)[0 \quad 1]\begin{bmatrix} -1 & 3T/2 \\ 1 & -T/2 \end{bmatrix}\begin{bmatrix} 0.1 & 0.4T \\ 0 & 0.1 \end{bmatrix};$$

therefore

$$[K_1 \quad K_2] = \frac{1}{T^2}[0.1 \quad 0.35T]$$
$$= [10 \quad 3.5],$$

which is the same result as that obtained earlier.

Viewing State-Space as a Tool for Solving ODEs Simultaneously

State Space as an ODE

- The basic mathematical model for an LTI system consists of the state differential equation

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \qquad \mathbf{x}(t_0) = \mathbf{x}_0$$
$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t)$$

- The solution is can be expressed as a sum of terms owing to the initial state and to the input respectively:

$$x(t) = \underbrace{e^{at}x_0}_{\text{zero-input response}} + \underbrace{\int_0^t e^{a(t-\tau)}bu(\tau)d\tau}_{\text{zero-state response}} \qquad y(t) = ce^{at}x_0 + \int_0^t ce^{a(t-\tau)}bu(\tau)d\tau + du(t)$$

- This is a first-order solution similar to what we expect

State Equation Solution: Matrix Exponential

$$x(t) = \boxed{e^{at}x_0} + \int_0^t e^{a(t-\tau)} bu(\tau)d\tau \qquad y(t) = ce^{at}x_0 + \int_0^t ce^{a(t-\tau)}bu(\tau)d\tau + du(t)$$

- The first term can be handled via a Taylor Series

$$e^{\mathbf{A}(t-t_0)} = \sum_{k=0}^{\infty} \frac{1}{k!}\mathbf{A}^k(t-t_0)^k = \mathbf{I} + \mathbf{A}(t-t_0) + \frac{1}{2}\mathbf{A}^2(t-t_0)^2 + \frac{1}{6}\mathbf{A}^3(t-t_0)^3 + \ldots$$

→ This case is known as the matrix exponential function

→ Also referred to as the state-transition matrix, denoted by Φ (t, t_0):

$$\mathbf{x}(t) = \mathbf{\Phi}(t)\mathbf{x}_0 + \int_{t_0}^t \mathbf{\Phi}(t-\tau)\mathbf{Bu}(\tau)d\tau$$

- The state-transition matrix satisfies the homogeneous state equation, thus, it represents the free response of the system. That is, it governs the response that is excited by the initial conditions only

Output Equation Solution

- Having the solution for the complete state response, a solution for the complete output equation can be obtained as:

$$\mathbf{y}(t) = \underbrace{\mathbf{C}e^{\mathbf{A}t}\mathbf{x}_0}_{\text{zero-input response: } y_{zi}(t)} + \underbrace{\int_{t_0}^t \mathbf{C}e^{\mathbf{A}(t-\tau)}\mathbf{Bu}(\tau)d\tau + \mathbf{Du}(t)}_{y_{zs}(t)\text{: zero-state response}}$$

State Equation Solution

- Thus, the solution to the unforced system (u=0):

$$\begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_n(t) \end{bmatrix} = \begin{bmatrix} \phi_{11}(t) & \cdots & \phi_{1}(t) \\ \phi_{21}(t) & \cdots & \phi_{n2}(t) \\ \vdots & & \vdots \\ \phi_{n1}(t) & \cdots & \phi_{nn}(t) \end{bmatrix} \begin{bmatrix} x_1(0) \\ x_2(0) \\ \vdots \\ x_n(0) \end{bmatrix}$$

Note: the term $\phi_{ij}(t)$ can be interpreted as the response of the i^{th} state variable due to an initial condition on the j^{th} state variable when there are zero initial conditions on all other states.

- The solution of the state differential equation can also be obtained using the Laplace transform:

$$L[\dot{\mathbf{x}}(t) = \mathbf{Ax}(t) + \mathbf{Bu}(t)]$$

$$L[\dot{\mathbf{x}}(t)] = L[\mathbf{Ax}(t)] + L[\mathbf{Bu}(t)]$$

$$s\mathbf{X}(s) - \mathbf{x}_0 = \mathbf{AX}(s) + \mathbf{BU}(s)$$

$$\mathbf{X}(s) = (s\mathbf{I} - \mathbf{A})^{-1}\mathbf{x}_0 + (s\mathbf{I} - \mathbf{A})^{-1}\mathbf{BU}(s)$$

$$\mathbf{X}(s) = \mathbf{\Phi}(s)\mathbf{x}_0 + \mathbf{\Phi}(s)\mathbf{BU}(s)$$

$$\Rightarrow \quad L[\mathbf{\Phi}(t)] = \mathbf{\Phi}(s) = [s\mathbf{I} - \mathbf{A}]^{-1} \longrightarrow \mathbf{\Phi}(t) = L^{-1}[s\mathbf{I} - \mathbf{A}]^{-1}$$

Properties of the Matrix Exponential

- Note that $e^{\mathbf{A}t}$ is just a notation used to represent a power series.

$$e^{\mathbf{A}t} \neq [e^{a_{ij}t}]$$

- Example 1: Consider the following 4x4 matrix

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Let's obtain the first terms of the power series:

$$A^2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad A^3 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{bmatrix} \quad A^4 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad A^k = 0 \quad \forall k \geq 4$$

The power series contains only a finite number of nonzero terms:

$$e^{\mathbf{A}t} = \mathbf{I} + \mathbf{A}t + \frac{1}{2}\mathbf{A}^2t^2 + \frac{1}{6}\mathbf{A}^3t^3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -t & 1 & 0 & 0 \\ \frac{1}{2}t^2 & -t & 1 & 0 \\ -\frac{1}{6}t^3 & \frac{1}{2}t^2 & -t & 1 \end{bmatrix} \neq [e^{a_{ij}t}] = \begin{bmatrix} 0 & 0 & 0 & 0 \\ e^{-t} & 0 & 0 & 0 \\ 0 & e^{-t} & 0 & 0 \\ 0 & 0 & e^{-t} & 0 \end{bmatrix}$$

Properties of the matrix exponential

- For any real nxn matrix **A**, the matrix exponential $e^{\mathbf{A}t}$ satisfies:
  1. $e^{\mathbf{A}t}$ is the unique matrix for which: $\frac{d}{dt}e^{\mathbf{A}t} = \mathbf{A}e^{\mathbf{A}t} \quad e^{\mathbf{A}t}\big|_{t=0} = \mathbf{I}(nxn)$
  2. For any t_1 and t_2: $e^{\mathbf{A}(t_1+t_2)} = e^{\mathbf{A}t_1}e^{\mathbf{A}t_2}$

     As a consequence: $e^{\mathbf{A}(0)} = e^{\mathbf{A}(t-t)} = e^{\mathbf{A}t}e^{-\mathbf{A}t} = \mathbf{I}$

     Thus, $e^{\mathbf{A}t}$ is invertible for all t, being the inverse: $\left[e^{\mathbf{A}t}\right]^{-1} = e^{-\mathbf{A}t}$
  3. For all t, A and $e^{\mathbf{A}t}$ commute with respect to matrix product: $\mathbf{A}e^{\mathbf{A}t} = e^{\mathbf{A}t}\mathbf{A}$
  4. For all t: $\left[e^{\mathbf{A}t}\right]^T = e^{\mathbf{A}^T t}$
  5. For any real nxn matrix B, $e^{(\mathbf{A}+\mathbf{B})t} = e^{\mathbf{A}t}e^{\mathbf{B}t}$ for all t if and only if **AB=BA**
  6. Finally, a useful property of the matrix exponential is that it can be reduced to a finite power series involving n scalar analytic functions $\alpha_j(t)$

$$e^{\mathbf{A}t} = \sum_{k=0}^{n-1} \alpha_k(t)\mathbf{A}^k$$

Using this to Solve State Space Problems

- Example:
  - Solve the following linear second-order ordinary differential

    $\ddot{y}(t) + 7\dot{y}(t) + 12y(t) = u(t)$
  - Consider the input u(t) is a step of magnitude 3 and the initial conditions $\dot{y}(0) = 0.05 \quad y(0) = 0.10$

State-Space Exercise

- Solve the following linear second-order ordinary differential eq:
  a. Using standard solution techniques
  b. Using S-S solution techniques

$$\ddot{y}(t)+7\dot{y}(t)+12y(t)=u(t)$$

Consider the input u(t) is a step of magnitude 3
and the initial conditions: $\dot{y}(0)=0.05 \quad y(0)=0.10$

The first question can be solved by the students in order to review the techniques exposed in previous courses.

To solve the second question, we first choose state variables using phase-variable choice.

$$x_1 = y$$
$$x_2 = \dot{y} = \dot{x}_1$$
$$\dot{x}_2 = \ddot{y} = u_s - 12x_1 - 7x_2$$

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix}=\begin{bmatrix}0 & 1\\ -12 & -7\end{bmatrix}\cdot\begin{bmatrix}x_1\\ x_2\end{bmatrix}+\underbrace{\begin{bmatrix}0\\ 1\end{bmatrix}}_{B}u(t) \qquad \mathbf{A}=\begin{bmatrix}0 & 1\\ -12 & -7\end{bmatrix}$$

$$\begin{bmatrix}x_1(0)\\ x_2(0)\end{bmatrix}=\begin{bmatrix}0.10\\ 0.05\end{bmatrix}$$

$$y(t)=\underbrace{\begin{bmatrix}1 & 0\end{bmatrix}}_{C}\cdot\begin{bmatrix}x_1\\ x_2\end{bmatrix}+\underbrace{[0]}_{D}\cdot u(t)$$

Powers of A are not nulls, thus, obtaining the state transition matrix as a power series is not practical

G.Oliver, UIB

State-Space Exercise

- The expression $\mathbf{\Phi}(t)=L^{-1}[s\mathbf{I}-\mathbf{A}]^{-1}$ is recommended:

$$s\mathbf{I}-\mathbf{A}=\begin{bmatrix}s & -1\\ 12 & s+7\end{bmatrix}$$

$$\det(s\mathbf{I}-\mathbf{A})=|s\mathbf{I}-\mathbf{A}|=s^2+7s+12$$

$$\mathbf{\Phi}(s)=(s\mathbf{I}-\mathbf{A})^{-1}=\frac{1}{s^2+7s+12}\begin{bmatrix}s+7 & 1\\ -12 & s\end{bmatrix}$$

- Thus, from $\mathbf{X}(s)=(s\mathbf{I}-\mathbf{A})^{-1}\mathbf{x}_0+(s\mathbf{I}-\mathbf{A})^{-1}\mathbf{B}U(s)$

$$\mathbf{X}(s)=\frac{1}{s^2+7s+12}\begin{bmatrix}s+7 & 1\\ -12 & s\end{bmatrix}\begin{bmatrix}0.10\\ 0.05\end{bmatrix}+\frac{1}{s^2+7s+12}\begin{bmatrix}s+7 & 1\\ -12 & s\end{bmatrix}\begin{bmatrix}0\\ 1\end{bmatrix}\frac{3}{s}=$$

$$=\frac{1}{s^2+7s+12}\begin{bmatrix}0.1s+0.75+\frac{3}{s}\\ 0.05s+1.8\end{bmatrix}=\begin{bmatrix}\dfrac{0.1s^2+0.75s+3}{s(s+3)(s+4)}\\ \dfrac{0.05s+1.8}{(s+3)(s+4)}\end{bmatrix}=\begin{bmatrix}X_1(s)\\ X_2(s)\end{bmatrix}$$

$$X_1(s)=\frac{0.1s^2+0.75s+3}{s(s+3)(s+4)}=\frac{0.25}{s}-\frac{0.55}{s+3}+\frac{0.4}{s+4} \qquad X_2(s)=\frac{0.05s+1.8}{(s+3)(s+4)}=\frac{1.65}{s+3}-\frac{1.60}{s+4}$$

LQR

Linear Quadratic Regulator

- $x' = Ax + Bu$
- Objective: minimize quadratic cost

$$\int \underbrace{\boxed{x^T Q\, x}}_{\text{Error term}} + \underbrace{\boxed{u^T R\, u\, dt}}_{\text{"Effort" penalization}}$$

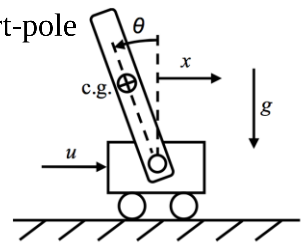
- Over an infinite horizon

Closed form LQR solution

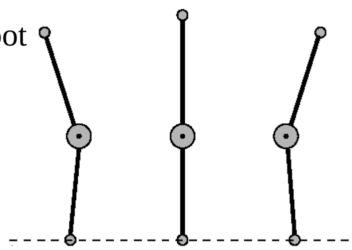
- Closed form solution
  $u = -K x$, with $K = R^{-1}BP$
- Where P is a symmetric matrix that solves the *Riccati equation*
  - $A^TP + PA - PBR^{-1}B^TP + Q = 0$
  - Derivation: calculus of variations
- Packages available for finding solution

Toy Nonlinear Systems

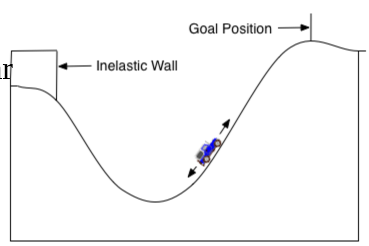
Cart-pole



Acrobot



Mountain car



From Linear to Nonlinear

- We know how to solve (assuming g_t, U_t, X_t convex):

$$\begin{aligned} \min_{u,x} \quad & \sum_{t=0}^{H} g_t(x_t, u_t) \\ \text{subject to} \quad & x_{t+1} = A_t x_t + B_t u_t + c_t \quad \forall t \\ & u_t \in \mathcal{U}_t, x_t \in \mathcal{X}_t \quad \forall t \end{aligned} \qquad (1)$$

- How about nonlinear dynamics:

$$x_{t+1} = f(x_t, u_t) \quad \forall t$$

Shooting Methods (feasible)	**Collocation Methods (infeasible)**
Iterate for i=1, 2, 3, …	Iterate for i=1, 2, 3, …
Execute $u_0^{(i)}, u_1^{(i)}, \ldots, u_T^{(i)}$ (from solving (1))	--- (no execution)---
Linearize around resulting trajectory	Linearize around current solution of (1)
Solve (1) for current linearization	Solve (1) for current linearization

Sequential Quadratic Programming (SQP) = either of the above methods, but instead of using linearization, linearize equality constraints, convex-quadratic approximate objective function

Model Predictive Control

- Given: $\bar{x}_0$
- For k=0, 1, …, T
  - Solve

$$\begin{aligned} \min_{x,u} \quad & \sum_{t=k}^{T} g_t(x_t, u_t) \\ \text{s.t.} \quad & x_{t+1} = f_t(x_t, u_t) \qquad \forall t \in \{k, k+1, \ldots, T-1\} \\ & x_k = \bar{x}_k \end{aligned}$$

  - Execute u_k
  - Observe resulting state, $\bar{x}_{k+1}$

Iterative LQR versus Sequential Convex Programming

- Both can solve

$$\begin{aligned} \min_{u,x} \quad & \sum_{t=0}^{H} g_t(x_t, u_t) \\ \text{subject to} \quad & x_{t+1} = f_t(x_t, u_t) \quad \forall t \\ & u_t \in \mathcal{U}_t, x_t \in \mathcal{X}_t \quad \forall t \end{aligned}$$

- Can run iterative LQR both as a shooting method or as a collocation method, it's just a different way of executing "Solve (1) for current linearization." In case of shooting, the sequence of linear feedback controllers found can be used for (closed-loop) execution.

- Iterative LQR might need some outer iterations, adjusting "t" of the log barrier

Shooting Methods	**Collocation Methods**
Iterate for i=1, 2, 3, …	Iterate for i=1, 2, 3, …
Execute feedback controller (from solving (1))	--- (no execution)---
Linearize around resulting **trajectory**	Linearize around current **solution** of (1)
Solve (1) for current linearization	Solve (1) for current linearization

Sequential Quadratic Programming (SQP) = either of the above methods, but instead of using linearization, linearize equality constraints, convex-quadratic approximate objective function

Example Shooting

```
%% a nonlinear control problem: cartpole
clear; clc; close all;

% shooting:
T = 100;
u = randn(1,T)*0.1;
max_iters = 10;
x_init = [-10; 0; 0; 0];
nX = 4; nU = 1;
x_eps = 0.1;
u_eps = 0.1;
dt = 0.1;
Q = eye(nX); R = eye(nU); Q_final = 100*eye(nX);
clear A B c
for iter = 1:max_iters
    % simulate and linearize
    x(:,1) = x_init;
    for t=1:T-1
        x(:,t+1) = sim_cartpole(x(:,t), u(:,t), dt);
        [A{t} B{t} c{t}] = compute_jacobian(@sim_cartpole, x(:,t), u(:,t), dt);
        %cartpole_draw(t*dt, x(:,t));
    end
    figure(1); subplot(3,1,1); hold on; plot(u); ylabel('u');
    subplot(3,1,2); hold on; plot(x(1,:)); ylabel('x'); subplot(3,1,3); hold on; plot(x(2,:));ylabel('\theta');
    cost(iter) = 0;
    for t=1:T-1
        cost(iter) = cost(iter) + x(:,t)'*Q*x(:,t) + u(:,t)'*R*u(:,t) + norm(u(:,t+1)-u(:,t),2);
    end
    cost(iter) = cost(iter) + x(:,T)'*Q_final*x(:,T);
    cost
    % solve convex problem
    cvx_begin
    variables x_cvx(nX,T)  u_cvx(nU,T) s_cvx(1,T);
    minimize( sum(s_cvx(1:T)) )
    subject to
    for t=1:T-1
        x_cvx(:,t+1) == A{t}*(x_cvx(:,t)-x(:,t)) + B{t}*(u_cvx(:,t)-u(:,t)) + c{t};
    end
    for t=1:T-1
        s_cvx(1,t) >= x_cvx(:,t)'*Q*x_cvx(:,t) + u_cvx(:,t)'*R*u_cvx(:,t) + norm(u_cvx(:,t+1)-u_cvx(:,t),2);
    end
    s_cvx(1,T) >= x_cvx(:,T)'*Q_final*x_cvx(:,T);
    for t=1:T
        norm(x_cvx(:,t) - x(:,t),2) <= x_eps;
        norm(u_cvx(:,t) - u(:,t),2) <= u_eps;
    end
    x_cvx(:,1) == x_init;
    cvx_end
    u = u_cvx;
end
```

Example Collocation

```
clear; clc; close all;

T = 100;
u = randn(1,T)*0.1;
max_iters = 10;
x_init = [-10; pi/10; -0.1; 0.1];
nX = 4; nU = 1;
x_eps = 100;
u_eps = 1;
dt = 0.1;
Q = eye(nX); R = eye(nU); Q_final = 100*eye(nX);
clear A B c
x_target = [0;0;0;0];
for i=1:nX
    x_iter1(i,:) = x_init(i):(x_target(i)-x_init(i))/(T-1):x_target(i);
end
u_iter1=zeros(nU,T);

for iter = 1:max_iters
    % linearize (but no simulation)
    if(iter==1)
        x = x_iter1; u = u_iter1;
    else
        x = x_cvx; u = u_cvx;
    end
    for t=1:T-1
        %   x(:,t+1) = sim_cartpole(x(:,t), u(:,t), dt);
        [A(t) B(t) c(t)] = compute_jacobian(@sim_cartpole, x(:,t), u(:,t), dt);
        %cartpole_draw(t*dt, x(:,t));
    end
    figure(2); subplot(3,1,1); hold on; plot(u); ylabel('u');
    subplot(3,1,2); hold on; plot(x(1,:)); ylabel('x'); subplot(3,1,3); hold on; plot(x(2,:));ylabel('\theta');
    cost(iter) = 0;
    for t=1:T-1
        cost(iter) = cost(iter) + x(:,t)'*Q*x(:,t) + u(:,t)'*R*u(:,t) + norm(u(:,t+1)-u(:,t),2);
    end
    cost(iter) = cost(iter) + x(:,T)'*Q_final*x(:,T);
    cost
    % solve convex problem
    cvx_begin
    variables x_cvx(nX,T)  u_cvx(nU,T) s_cvx(1,T);
    minimize( sum(s_cvx(1:T)) )
    subject to
    for t=1:T-1
        x_cvx(:,t+1) == A(t)*(x_cvx(:,t)-x(:,t)) + B(t)*(u_cvx(:,t)-u(:,t)) + c(t);
    end
    for t=1:T-1
        s_cvx(1,t) >= x_cvx(:,t)'*Q*x_cvx(:,t) + u_cvx(:,t)'*R*u_cvx(:,t) + norm(u_cvx(:,t+1)-u_cvx(:,t),2);
    end
    s_cvx(1,T) >= x_cvx(:,T)'*Q_final*x_cvx(:,T);
    for t=1:T
        norm(x_cvx(:,t) - x(:,t),2) <= x_eps;
        norm(u_cvx(:,t) - u(:,t),2) <= u_eps;
    end
    x_cvx(:,1) == x_init;
    cvx_end
    u = u_cvx;
end

% let's evaluate the resulting open-loop sequence:
x(:,1) = x_init;
for t=1:T-1
    x(:,t+1) = sim_cartpole(x(:,t), u(:,t), dt);
end
figure(3); subplot(3,1,1); hold on; plot(u); ylabel('u');
subplot(3,1,2); hold on; plot(x(1,:)); ylabel('x'); subplot(3,1,3); hold on; plot(x(2,:));ylabel('\theta');
final_cost = 0;
for t=1:T-1
  final_cost = final_cost + x(:,t)'*Q*x(:,t) + u(:,t)'*R*u(:,t) + norm(u(:,t+1)-u(:,t),2);
end
final_cost = final_cost + x(:,T)'*Q_final*x(:,T);
final_cost
```


Practical Benefits and Issues with Shooting

+ :

At all times the sequence of controls is meaningful, and the objective function optimized directly corresponds to the current control sequence

-- :

For unstable systems, need to run feedback controller during forward simulation

- Why? Open loop sequence of control inputs computed for the linearized system will not be perfect for the nonlinear system. If the nonlinear system is unstable, open loop execution would give poor performance.
- Fixes:
  - Run Model Predictive Control for forward simulation
  - Compute a linear feedback controller from the 2[nd] order Taylor expansion at the optimum

Practical Benefits and Issues with Collocation

+ :

Can initialize with infeasible trajectory. Hence if you have a rough idea of a sequence of states that would form a reasonable solution, you can initialize with this sequence of states without needing to know a control sequence that would lead through them, and without needing to make them consistent with the dynamics

-- :

Sequence of control inputs and states might never converge onto a feasible sequence

Direct policy synthesis: Optimal control

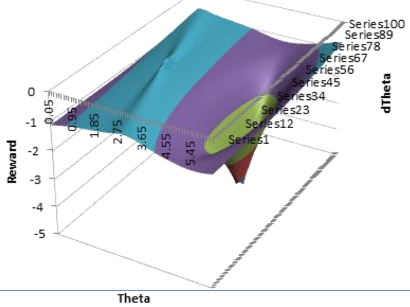
- *Input*: cost function J(x), estimated dynamics f(x,u), finite state/control spaces X, U

- Two basic classes:
  - **Trajectory optimization**: Hypothesize control sequence u(t), simulate to get x(t), perform optimization to improve u(t), repeat.
  - *Output*: optimal trajectory u(t) (in practice, only a locally optimal solution is found)
  - **Dynamic programming:** Discretize state and control spaces, form a discrete search problem, and solve it.
  - *Output*: Optimal policy u(x) across all of X

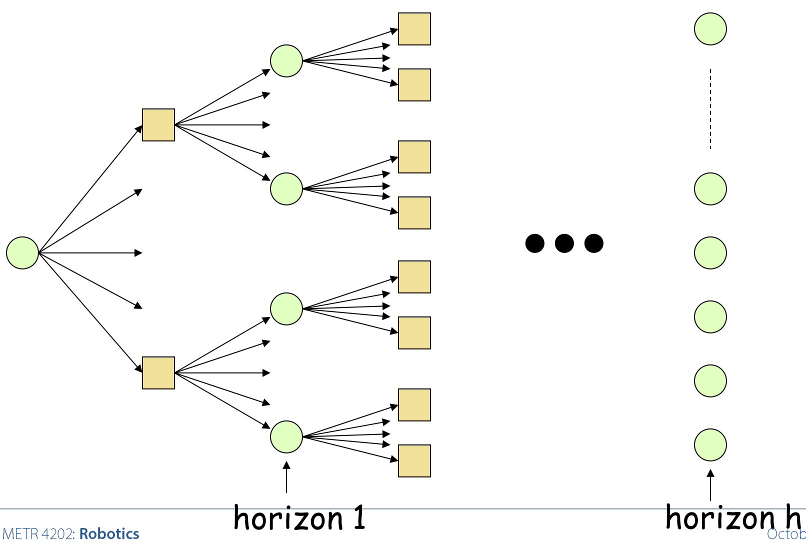
Discrete Search example

- Split X, U into cells $x_1,\ldots,x_n$, $u_1,\ldots,u_m$
- Build transition function $x_j = f(x_i,u_k)dt$ for all i,k
- State machine with costs dt $J(x_i)$ for staying in state I
- Find $u(x_i)$ that minimizes sum of total costs.
- **Value iteration**: repeated dynamic programming over $V(x_i)$ = sum of total future costs

Value function for 1-joint acrobot



Receding Horizon Control (aka model predictive control)

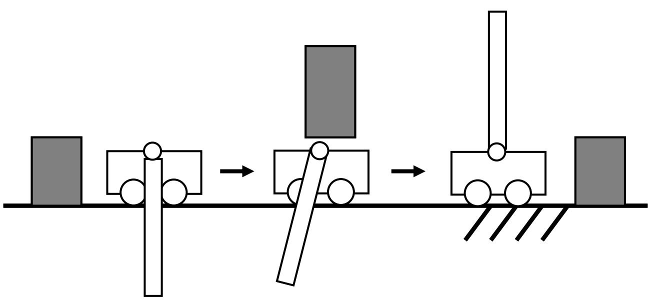


Integrated Planning & Control

(Hooray!!!)

Integrated Planning and Control Methods…

- A motivating problem (for agility)
  - Cart and pole in a cluttered workspace …

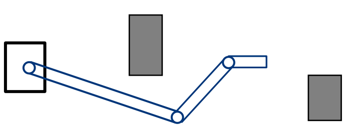


$$\dot{x} = F(x, u)$$

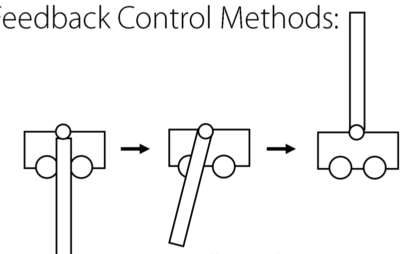
Trajectory Generation with Constraints: Solutions from the Robotics Domain

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

Motion Planning Methods:



Feedback Control Methods:

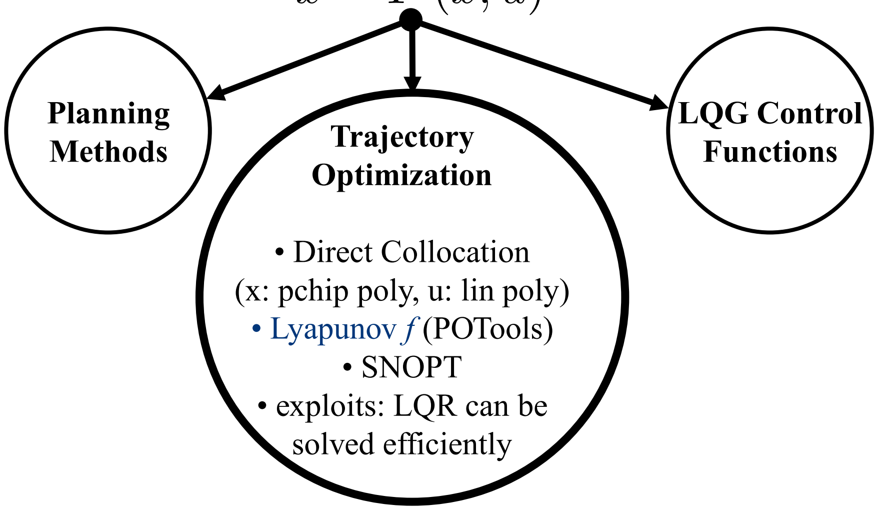


$$(M+m)\ddot{y} + ml\cos\theta\ddot{\theta} - ml\dot{\theta}^2\sin\theta = f$$

$$m\left(l\cos\theta\ddot{y} + l^2\ddot{\theta} - gl\sin\theta\right) = 0$$

Trajectory Optimization:
→ Integrated Planning & Feedback:

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



Few Questions Before Starting…

- *Can it possibly be this hard?*

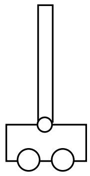
  Yes!
  (1) Dynamic systems are nonlinear $m\left(l^2\ddot{\theta} + gl\sin\theta\right) = 0$
  (2) Decision-theoretic planning problems are combinatorial

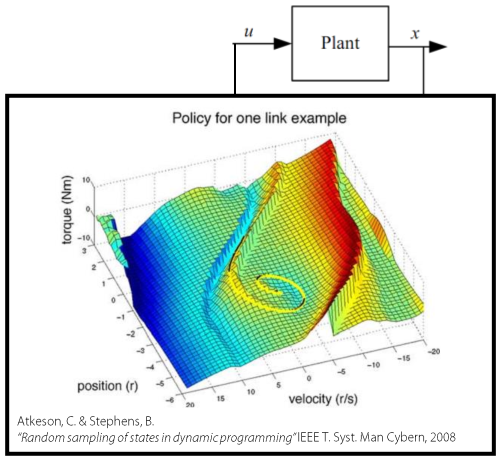
- ***Underactuated system***?

  [≙ control input cannot drive the state to any arbitrary direction]
  → DOF>actuators: car-like robot, airplane, cart and pole, etc
  ➔ Actuator saturation!

- *Why Now?*
  1. State-of-the-art (LQR-Trees 2009 RSS best paper, kNitro, SDPARA).
  2. Convex Optimization (c/o relaxation) is ~ “online-able”

Viewing This From a Controls & Policy Perspective





Core Idea:

Feedback motion planning (Assistance function) built from a prior model and updated online

Gain-Scheduled RRT

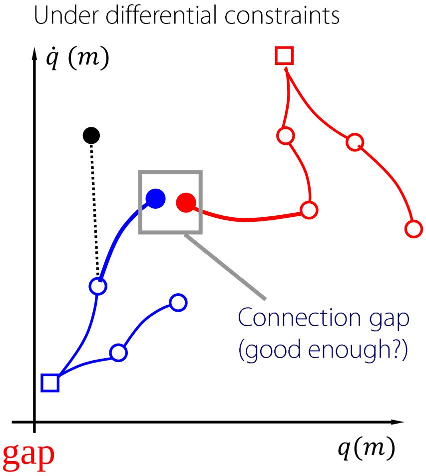
- Rapidly Exploring Random Trees (RRT) (Background):

Features (+):

1. Solve a control problem
2. Scalable
3. Constrained environments

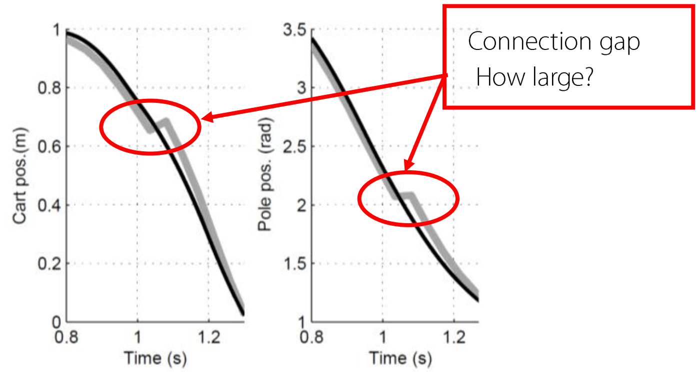
Concerns (--):

4. Works poorly under differential constraints
5. Hard to avoid the connection gap



Gain-Scheduled RRT: RRT Connection Gap

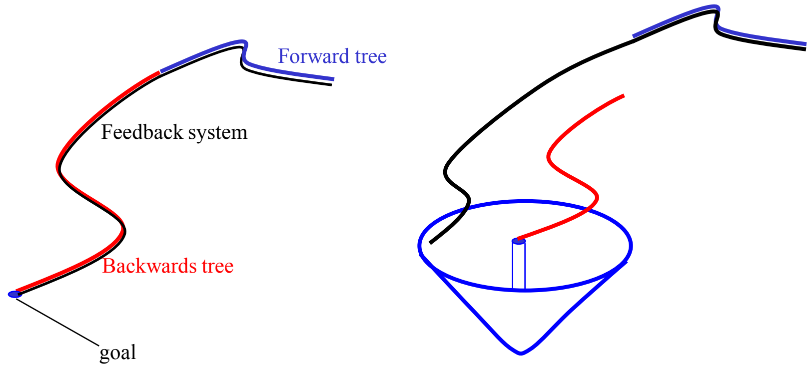
- A RRT solution rarely reaches the goal (or connect the two trees) with zero error



Gain-Scheduled RRT: Relaxing the Search

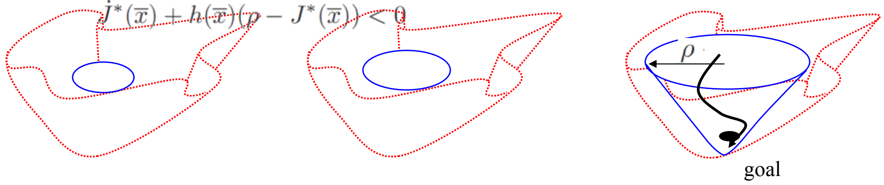
Single state search:
Extensive exploration

Region search
RRT connection is relaxed



Gain-Scheduled RRT: RoA & Verification

- Find a candidate
  In the LQR case: J: optimal cost-to-go S: Algebraic Ricatti Eq.
  V(x) $= J^*(x_{goal} - x) = (x_{goal} - x)^T S(x_{goal} - x)$
-
  Maximize candidate (ρ)
  $\dot{J}^*(\bar{x}) + h(\bar{x})(\rho - J^*(\bar{x})) < 0$



- Verify candidate

Sum of squares relaxation

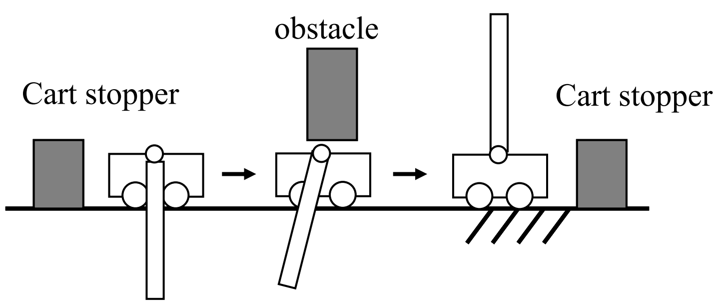
$V(x)$ locally $\boxed{\text{positive definite}}$ in B_ρ

$\dot{V}(x)$ locally $\boxed{\text{negative definite}}$ in B_ρ

**R. Tedrake, *"LQR-Trees: Feedback Motion Planning on Sparse Randomized Trees"*, RSS 2009

Gain-Scheduled RRT: Result

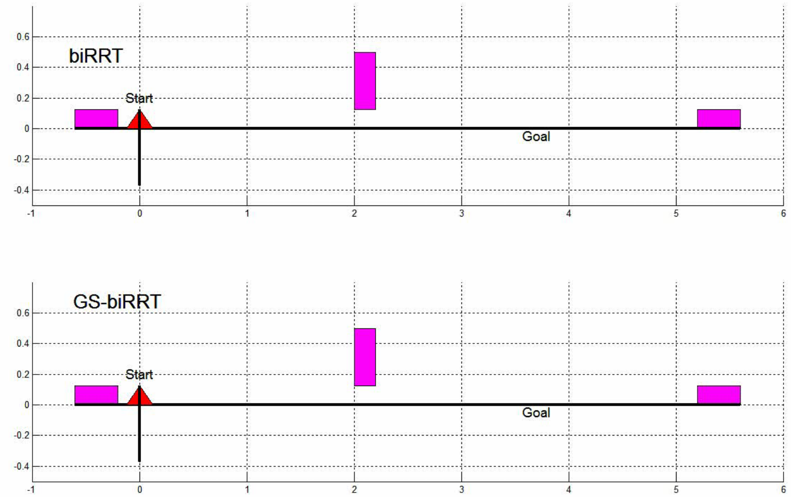
- Cart and pole in a cluttered workspace …



Same initial and final conditions.

Every solution is different due to the random sampling

Gain-Scheduled RRT: Result



Conclusion
No one answer…
Much left to do!

(it's not really magic ☺)