METR4202: Robotics & Automation
Lecture Compendium

METR 4202: **Robotics** & Automation

Dr Surya Singh

July 27-October 30, 2016

metr4202@itee.uq.edu.au
http://robotics.itee.uq.edu.au/~metr4202/

[http://**metr4202.com**]

PART II:
(Stochastic) **Modelling, Planning & Control of Robotic/Autonomous Systems**

DRAFT

Lectures 8-13

Probabilistic Robotics: Localization

METR 4202: **Robotics** & Automation

Dr Surya Singh -- Lecture # 8

September 14, 2016

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

Localization: SFM → SLAM

Perfect World: Deterministic

- Exact pose from motion model
- Global localisation by triangulation
  - Even if range-only or bearing-only sensors, can localise given enough measurements
  - Solve simultaneous equations: N equations for N unknowns

Real World: Uncertain

- All measurements have errors
- In SLAM, measurement errors induce dependencies in the landmark and vehicle pose estimates
  - Everything is correlated

How to quantify uncertainty ?
Probability to the rescue…

Bertsekas & Tsitsiklis, Introduction to Probability.

- FATHER(F): Nurse, what is the probability that the drug will work?
- NURSE (N): I hope it works, we'll know tomorrow.
- F: Yes, but what is the probability that it will?
- N: Each case is different, we have to wait.
- F: But let's see, out of a hundred patients that are treated under similar conditions, how many times would you expect it to work?
- N (somewhat annoyed): I told you, every person is different, for some it works, for some it doesn't.
- F (insisting): Then tell me, if you had to bet whether it will work or not, which side of the bet would you take?
- N (cheering up for a moment): I'd bet it will work.
- F (somewhat relieved): OK, now, would you be willing to lose two dollars if it doesn't work, and gain one dollar if it does?
- N (exasperated): What a sick thought! You are wasting my time!

Probability review 1/4: Probabilistic Modeling

- View:
  - Experiments with random outcome.
  - Quantifiable properties of the outcome.
- Three components:
  - Sample space: Set of all possible outcomes.
  - Events: Subsets of sample space.
  - Probability: Quantify how likely an event occurs.

Probability review 2/4: Probability

- Probability: A function that maps events to real numbers satisfying these axioms:
  1. Non-negativity: $P(E) \geq 0$, where E is an event.
  2. Normalization: $P(S) = 1$, where S is the sample space.
  3. Additivity of finite / countably infinite events.

$$P\left(\bigcup_{i=1}^{\infty/n} E_i\right) = \sum_{i=1}^{\infty/n} P(E_i),$$

where E_i are disjoint / mutually exclusive, i : natural number.

Probability review 3/4: Random Variables

- Interest is on numerical values associated w. samples, e.g.:
  - Sample 50 students enrolled in METR4202, what's the major of most of the students.
  - Roll a fair dice, get \$5 if the outcome is even, & loose \$5 if the outcome is odd.
- Random variable X is a function $X : S \rightarrow Num$.
  - Num: countable set (e.g., integer) → discrete random variable.
  - Num: uncountable set (e.g., real) → continuous random variable.

Probability review 4/4: Characterizing Random Variables

- Cumulative distribution function (cdf)

$$F_X(x) = P(X \le x) = P\left(\{s | X(s) \le x, s \in S\}\right)$$

- Discrete: Probability mass function (pmf)

$$f_X[x] = P(X = x)$$

- Continuous: Probability density function/probability distribution function (pdf)

$$f_X(x) = \frac{dF_X(x)}{dx} \quad ; \quad P(a \le X \le b) = \int_a^b f_X(x)dx$$

Brief Overview of Probability Theory

- Probability density function (PDF) over N-D state space $\mathbf{x} \in \mathcal{X}$ is denoted $p(\mathbf{x})$

- Properties of a PDF

$$\mathbb{R}^N \mapsto \mathbb{R}$$

$$p(\mathbf{x}) \ge 0, \quad \forall\ \mathbf{x} \in \mathcal{X}$$

$$\int_{\mathcal{X}} p(\mathbf{x})\, d\mathbf{x} = 1$$

Brief Overview of Probability Theory

- State vector $\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \end{bmatrix}$
- Joint PDF is $p(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$
- Conditional PDF of x_1 *given* x_2 and x_3

  $p(\mathbf{x}_1 | \mathbf{x}_2, \mathbf{x}_3)$
- *Conditional independence*: if x_1 is independent of x_2 given x_3 then

  $$p(\mathbf{x}_1 | \mathbf{x}_2, \mathbf{x}_3) \overset{\text{indep}}{=} p(\mathbf{x}_1 | \mathbf{x}_3)$$

Two Essential Rules for Manipulating Probabilities

- Sum rule $p(\mathbf{x}_1 | \mathcal{H}) \triangleq \int p(\mathbf{x}_1, \mathbf{x}_2 | \mathcal{H})\, d\mathbf{x}_2$
- Product rule

$$\begin{aligned} p(\mathbf{x}_1, \mathbf{x}_2 | \mathcal{H}) &\triangleq p(\mathbf{x}_1 | \mathbf{x}_2, \mathcal{H})\, p(\mathbf{x}_2 | \mathcal{H}) \\ &\triangleq p(\mathbf{x}_2 | \mathbf{x}_1, \mathcal{H})\, p(\mathbf{x}_1 | \mathcal{H}) \end{aligned}$$

Implications of the Product Rule

- Conditionals $p(\mathbf{x}_1|\mathbf{x}_2,\mathcal{H}) = \frac{p(\mathbf{x}_1,\mathbf{x}_2|\mathcal{H})}{p(\mathbf{x}_2|\mathcal{H})}$
- Independence $p(\mathbf{x}_1,\mathbf{x}_2|\mathcal{H}) \overset{\text{indep}}{=} p(\mathbf{x}_1|\mathcal{H})\, p(\mathbf{x}_2|\mathcal{H})$
- Markov Models $p(\mathbf{x}_1|\mathcal{H}) = \int p(\mathbf{x}_1|\mathbf{x}_2,\mathcal{H})\, p(\mathbf{x}_2|\mathcal{H})\, d\mathbf{x}_2$
- Bayes theorem $p(\mathbf{x}_1|\mathbf{x}_2,\mathcal{H}) = \frac{p(\mathbf{x}_2|\mathbf{x}_1,\mathcal{H})\, p(\mathbf{x}_1|\mathcal{H})}{p(\mathbf{x}_2|\mathcal{H})}$

Marginalisation: Remove old states

- As per the sum rule

$$\begin{aligned} p(\mathbf{x}_1) &= \int p(\mathbf{x}_1,\mathbf{x}_2)\, d\mathbf{x}_2 \\ &= \int p(\mathbf{x}_1|\mathbf{x}_2)\, p(\mathbf{x}_2)\, d\mathbf{x}_2 \end{aligned}$$

- Marginal says: what is PDF of x_1 when we don't care what value x_2 takes; ie, $p(x_1)$ regardless of x_2
- Important distinction: x_1 is still dependent on x_2, but $p(x_1)$ is not a *function* of x_2

Bayesian Update: Inverse probability

- Bayes theorem $p\left(\mathbf{x}|\mathbf{z}\right) = \frac{p\left(\mathbf{z}|\mathbf{x}\right)p\left(\mathbf{x}\right)}{p\left(\mathbf{z}\right)}$
- Observation model $\mathbf{z} = \mathbf{h}\left(\mathbf{x}, \mathbf{r}\right)$
- Conditional probability

$$\begin{aligned} p\left(\mathbf{z}|\mathbf{x}\right) &= \int p\left(\mathbf{z}|\mathbf{x}, \mathbf{r}\right) p\left(\mathbf{r}\right) d\mathbf{r} \\ &= \int \delta\left(\mathbf{z} - \mathbf{h}\left(\mathbf{x}, \mathbf{r}\right)\right) p\left(\mathbf{r}\right) d\mathbf{r} \end{aligned}$$

- Likelihood function $\Lambda(\mathbf{x}) = p\left(\mathbf{z} = \mathbf{z}_0|\mathbf{x}\right)$

Bayes Update

- Update $p\left(\mathbf{x}|\mathbf{z} = \mathbf{z}_0\right) = \frac{\Lambda(\mathbf{x})p\left(\mathbf{x}\right)}{\int \Lambda(\mathbf{x})p\left(\mathbf{x}\right)d\mathbf{x}}$
- Denominator term often seen as just a normalising constant, but is important for saying how likely a model or hypothesis is
  - Used in FastSLAM for determining particle weights
  - Used in multi-hypothesis data association

Bayesian Estimation

- Standard theory for dealing with uncertain information in a consistent manner

More Cool Robotics Share!

Probabilistic Robotics: SLAM

METR 4202: **Robotics** & Automation

Dr Surya Singh -- Lecture # 9

September 21, 2016

metr4202@itee.uq.edu.au
http://robotics.itee.uq.edu.au/~metr4202/

[http://**metr4202.com**]

SLAM!
(Better than SMAL!)

What is SLAM?

- SLAM asks the following question:

  Is it possible for an autonomous vehicle to start at an unknown location in an unknown environment and then to incrementally build a map of this environment while simultaneously using this map to compute vehicle location?

- SLAM has many indoor, outdoor, in-air and underwater applications for both manned and autonomous vehicles.

- Examples
  - Explore and return to starting point (Newman)
  - Learn trained paths to different goal locations
  - Traverse a region with complete coverage (eg, mine fields, lawns, reef monitoring)
  - …

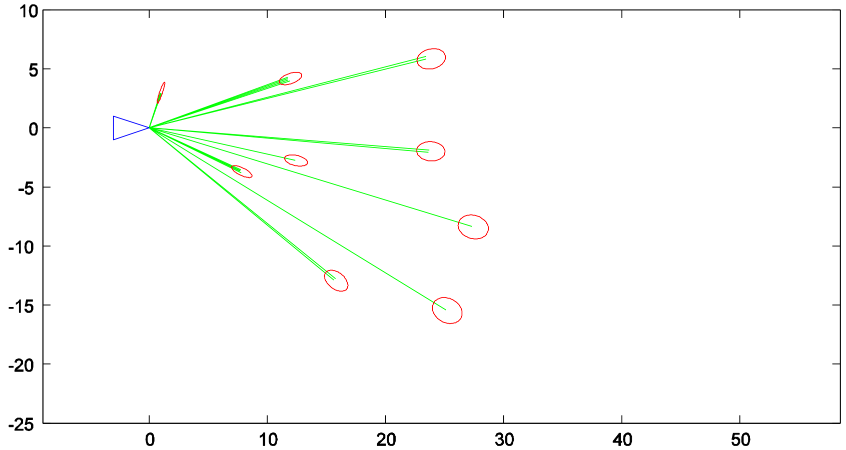
Components of SLAM

- Localisation
  - Determine pose given a priori map
- Mapping
  - Generate map when pose is accurately known from auxiliary source.
- SLAM
  - Define some arbitrary coordinate origin
  - Generate a map from on-board sensors
  - Compute pose from this map
  - Errors in map and in pose estimate are dependent.

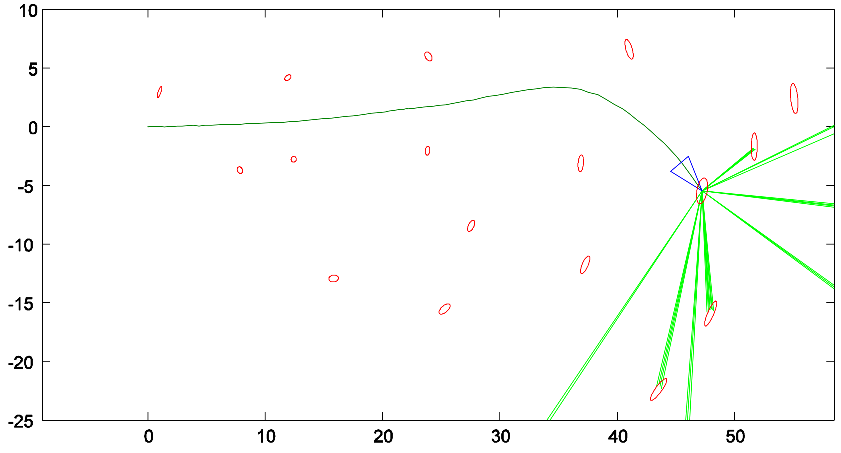
Basic SLAM Operation

Example: SLAM in Victoria Park

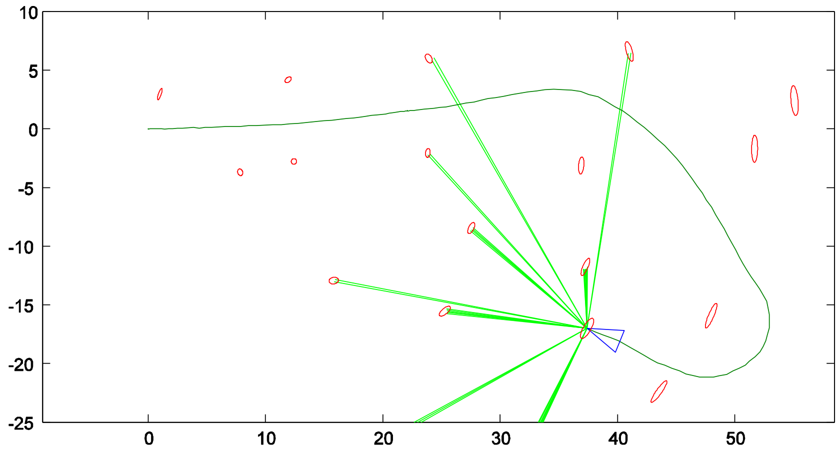
Basic SLAM Operation



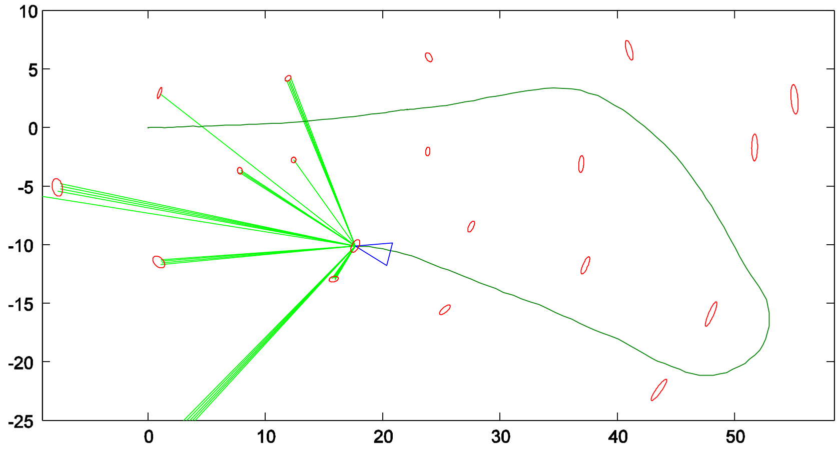
Basic SLAM Operation



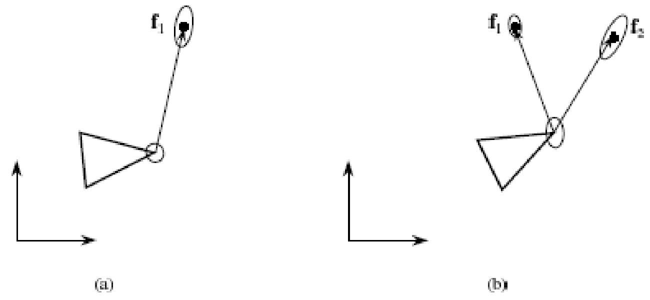
Basic SLAM Operation



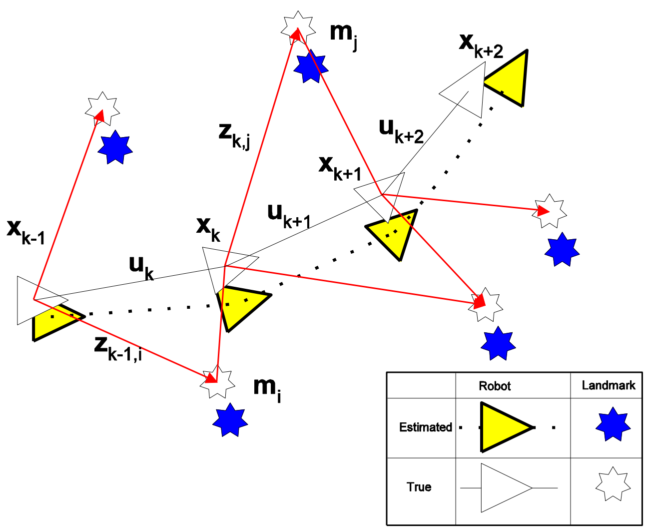
Basic SLAM Operation



Dependent Errors



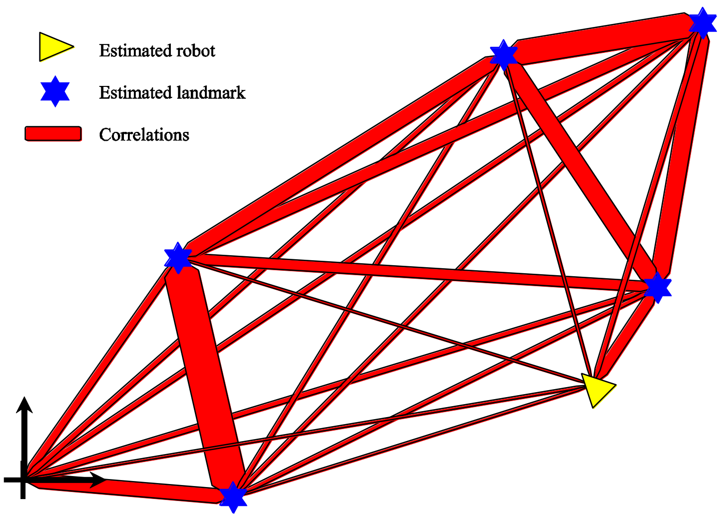
Correlated Estimates



SLAM Convergence

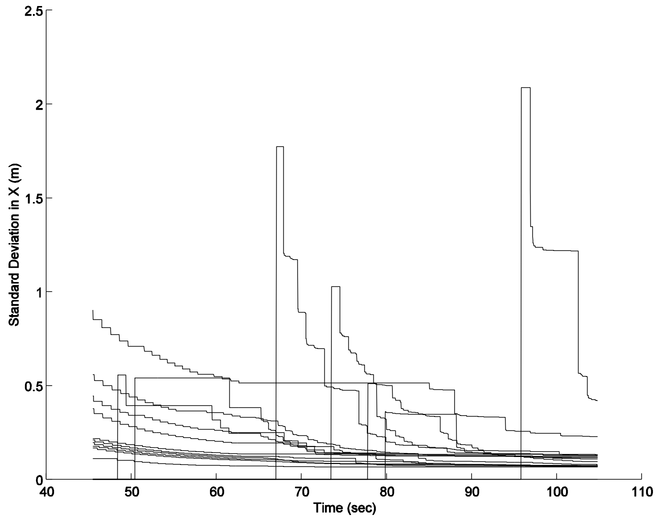
- An observation acts like a displacement to a spring system
  - Effect is greatest in a close neighbourhood
  - Effect on other landmarks diminishes with distance
  - Propagation depends on local stiffness (correlation) properties
- With each new observation the springs become increasingly (and monotonically) stiffer.
- In the limit, a rigid map of landmarks is obtained.
  - A perfect *relative* map of the environment
- The location accuracy of the robot is bounded by
  - The current quality of the map
  - The relative sensor measurement

Spring Analogy



Monotonic Convergence

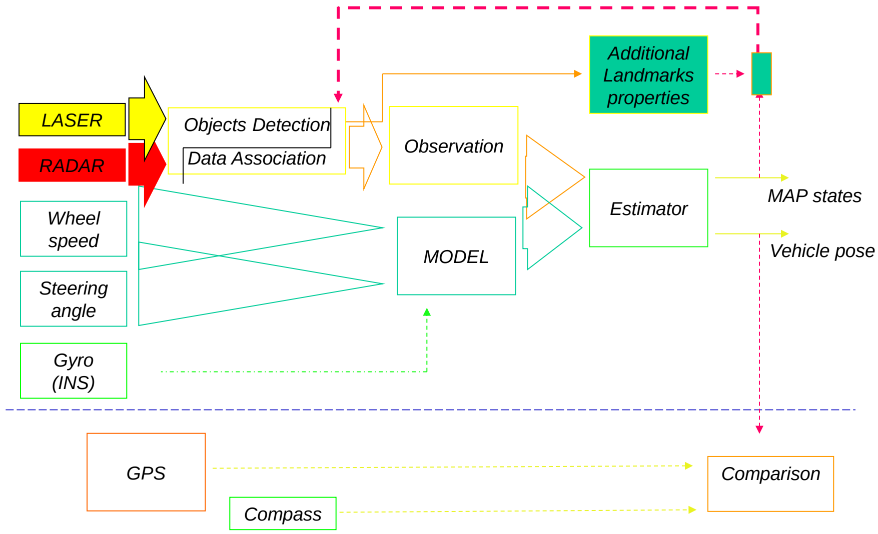
- With each new observation, the determinant decreases over the map and for any submatrix in the map.



Models

- Models are central to creating a representation of the world.
- Must have a mapping between sensed data (eg, laser, cameras, odometry) and the states of interest (eg, vehicle pose, stationary landmarks)
- Two essential model types:
  - Vehicle motion
  - Sensing of external objects

An Example System



States, Controls, Observations

Joint state with momentary pose

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_{v_k} \\ \mathbf{m}_1 \\ \vdots \\ \mathbf{m}_N \end{bmatrix}$$

Joint state with pose history

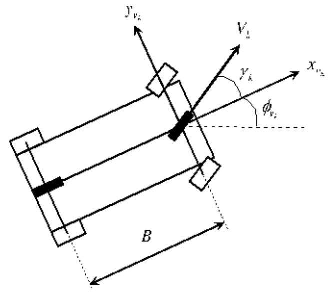
$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_{v_k} \\ \mathbf{x}_{v_{k-1}} \\ \vdots \\ \mathbf{x}_{v_0} \\ \mathbf{m}_1 \\ \vdots \\ \mathbf{m}_N \end{bmatrix}$$

Control inputs: $\mathbf{U}_{0:k} = \{\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_k\} = \{\mathbf{U}_{0:k-1}, \mathbf{u}_k\}$

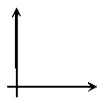
Observations: $\mathbf{Z}_{0:k} = \{\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_k\} = \{\mathbf{Z}_{0:k-1}, \mathbf{z}_k\}$

Vehicle Motion Model

- Ackerman steered vehicles: Bicycle model



- Discrete time model:



$$\mathbf{x}_{v_k} = \mathbf{f}_v\left(\mathbf{x}_{v_{k-1}}, \mathbf{u}_k\right) = \begin{bmatrix} x_{v_{k-1}} + V_k \Delta T \cos(\phi_{v_{k-1}} + \gamma_k) \\ y_{v_{k-1}} + V_k \Delta T \sin(\phi_{v_{k-1}} + \gamma_k) \\ \phi_{v_{k-1}} + \frac{V_k \Delta T}{B} \sin(\gamma_k) \end{bmatrix}$$

SLAM Motion Model

$$\mathbf{x}_{v_k} = \mathbf{f}_v\left(\mathbf{x}_{v_{k-1}}, \mathbf{u}_k\right) = \begin{bmatrix} x_{v_{k-1}} + V_k \Delta T \cos(\phi_{v_{k-1}} + \gamma_k) \\ y_{v_{k-1}} + V_k \Delta T \sin(\phi_{v_{k-1}} + \gamma_k) \\ \phi_{v_{k-1}} + \frac{V_k \Delta T}{B} \sin(\gamma_k) \end{bmatrix}$$

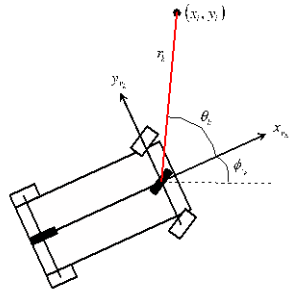
- Joint state: Landmarks are assumed stationary

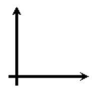
$$\mathbf{x}_k = \begin{bmatrix} \mathbf{f}_v(\mathbf{x}_{v_{k-1}}, \mathbf{u}_k) \\ \mathbf{m}_1 \\ \vdots \\ \mathbf{m}_N \end{bmatrix} \qquad \mathbf{x}_k = \begin{bmatrix} \mathbf{f}_v(\mathbf{x}_{v_{k-1}}, \mathbf{u}_k) \\ \mathbf{x}_{v_{k-1}} \\ \vdots \\ \mathbf{x}_{v_0} \\ \mathbf{m}_1 \\ \vdots \\ \mathbf{m}_N \end{bmatrix}$$

Observation Model

- Range-bearing measurement

$$\mathbf{z}_{i_k} = \mathbf{h}_i(\mathbf{x}_k) = \begin{bmatrix} \sqrt{(x_i - x_{v_k})^2 + (y_i - y_{v_k})^2} \\ \arctan \frac{y_i - y_{v_k}}{x_i - x_{v_k}} - \phi_{v_k} \end{bmatrix}$$





Applying Bayes to SLAM: Available Information

- States $\mathbf{x}_k$ (Hidden or inferred values)
  - Vehicle poses
  - Map; typically composed of discrete parts called landmarks or features
- Controls $\mathbf{U}_{0:k}$
  - Velocity
  - Steering angle
- Observations $\mathbf{Z}_{0:k}$
  - Range-bearing measurements

Augmentation: Adding new poses and landmarks

- Add new pose

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{f}_v(\mathbf{x}_{v_{k-1}}, \mathbf{u}_k) \\ \mathbf{x}_{v_{k-1}} \\ \vdots \\ \mathbf{x}_{v0} \\ \mathbf{m}_1 \\ \vdots \\ \mathbf{m}_N \end{bmatrix}$$

- Conditional probability is a Markov Model

$$\begin{aligned} p\left(\mathbf{x}_{v_k}|\mathbf{x}_{k-1}\right) &= \int p\left(\mathbf{x}_{v_k}|\mathbf{x}_{k-1}, \mathbf{u}_k\right) p\left(\mathbf{u}_k\right) d\mathbf{u}_k \\ &= \int \delta\left(\mathbf{x}_{v_k} - \mathbf{f}_v\left(\mathbf{x}_{v_{k-1}}, \mathbf{u}_k\right)\right) p\left(\mathbf{u}_k\right) d\mathbf{u}_k \\ &= p\left(\mathbf{x}_{v_k}|\mathbf{x}_{v_{k-1}}\right) \end{aligned}$$

Augmentation

$$\begin{aligned} p\left(\mathbf{x}_{v_k}|\mathbf{x}_{k-1}\right) &= \int p\left(\mathbf{x}_{v_k}|\mathbf{x}_{k-1}, \mathbf{u}_k\right) p\left(\mathbf{u}_k\right) d\mathbf{u}_k \\ &= \int \delta\left(\mathbf{x}_{v_k} - \mathbf{f}_v\left(\mathbf{x}_{v_{k-1}}, \mathbf{u}_k\right)\right) p\left(\mathbf{u}_k\right) d\mathbf{u}_k \\ &= p\left(\mathbf{x}_{v_k}|\mathbf{x}_{v_{k-1}}\right) \end{aligned}$$

- Product rule to create joint PDF $p(x_k)$

$$p\left(\mathbf{x}_{v_k}, \mathbf{x}_{k-1}\right) = p\left(\mathbf{x}_{v_k}|\mathbf{x}_{v_{k-1}}\right) p\left(\mathbf{x}_{v_{k-1}}, \ldots, \mathbf{x}_{v0}, \mathbf{m}_1, \ldots, \mathbf{m}_N\right)$$

- Same method applies to adding new landmark states

Marginalisation: Removing past poses and obsolete landmarks

- Augmenting with new pose and marginalising the old pose gives the classical SLAM prediction step

$$p\left(\mathbf{x}_{v_k}, \mathbf{m}_1, \ldots, \mathbf{m}_N\right) = \int p\left(\mathbf{x}_{v_k}, \mathbf{x}_{v_{k-1}}, \mathbf{m}_1, \ldots, \mathbf{m}_N\right) d\mathbf{x}_{v_{k-1}}$$

Fusion: Incorporating observation information

- Conditional PDF according to observation model

$$\begin{aligned} p\left(\mathbf{z}_{i_k} | \mathbf{x}_k\right) &= \int p\left(\mathbf{z}_{i_k} | \mathbf{x}_{v_k}, \mathbf{m}_i, \mathbf{r}_k\right) p\left(\mathbf{r}_k\right) d\mathbf{r}_k \\ &= \int \delta\left(\mathbf{z}_{i_k} - \mathbf{h}\left(\mathbf{x}_{v_k}, \mathbf{m}_i, \mathbf{r}_k\right)\right) p\left(\mathbf{r}_k\right) d\mathbf{r}_k \end{aligned}$$

- Bayes update:
  proportional to product of likelihood and prior

$$p\left(\mathbf{x}_k | \mathbf{Z}_{0:k}\right) = \frac{p\left(\mathbf{z}_{i_k} = \mathbf{z}_0 | \mathbf{x}_k\right) p\left(\mathbf{x}_k | \mathbf{Z}_{0:k-1}\right)}{p\left(\mathbf{z}_{i_k} = \mathbf{z}_0\right)}$$

Implementing Probabilistic SLAM

- The problem is that Bayesian operations are intractable in general.
  - General equations are good for analytical derivations, not good for implementation
- We need approximations
  - Linearised Gaussian systems (EKF, UKF, EIF, SAM)
  - Monte Carlo sampling methods (Rao-Blackwellised particle filters)

EKF SLAM

- The complicated Bayesian equations for augmentation, marginalisation, and fusion have simple and efficient closed form solutions for linear Gaussian systems
- For non-linear systems, just linearise
  - EKF, EIF: Jacobians
  - UKF: use deterministic samples

Kalman Implementation

- So can we just plug the process and observation models into the standard EKF equations and turn the crank?

- Several additional issues:
  - Structure of the SLAM problem permits more efficient implementation than naïve EKF.
  - Data association.
  - Feature initialisation.

Structure of SLAM

- Key property of stochastic SLAM
  - Largely a *parameter* estimation problem

- Since the map is stationary
  - No process model, no process noise

- For Gaussian SLAM
  - Uncertainty in each landmark reduces monotonically after landmark initialisation
  - Map converges

- Examine computational consequences of this structure in next session.

Data Association

- Before the Update Stage we need to determine if the feature we are observing is:
  - An old feature
  - A new feature

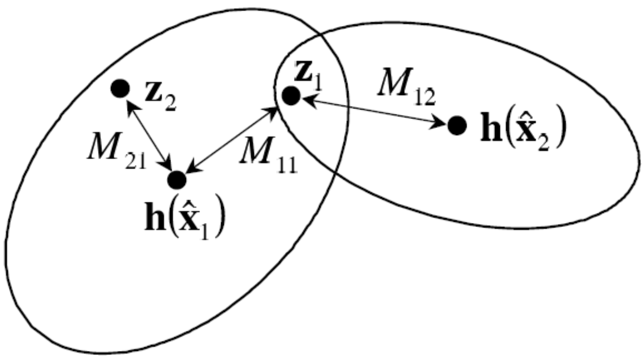
- If there is a match with only one known feature, the Update stage is run with this feature information.

$$\mu(k) = z(k) - h(\hat{x}(k/k-1))$$

$$S(k) = \nabla h_x(k) P(k/k-1) \nabla h_x^T(k) + R$$

$$\alpha = \mu^T(k) S^{-1}(k)\ \mu(k) < \chi^2_{0.95}$$

Validation Gating



New Features

- If there is no match then a potential new feature has been detected

- We do not want to incorporate a spurious observation as a new feature
  - It will not be observed again and will consume computational time and memory
  - It will add clutter, increasing risk of future mis-associations
  - The features are assumed to be static. We don't not want to accept dynamic objects as features: cars, people etc.

Acceptance of New Features

- **APPROACH 1**

  - Get the feature in a list of potential features
  - Incorporate the feature once it has been observed for a number of times

  - Advantages:
    - Simple to implement
    - Appropriate for High Frequency external sensor

  - Disadvantages:
    - Loss of information
    - Potentially a problem with sensor with small field of view: a feature may only be seen very few times

Acceptance of New Features

- **APPROACH 2**
  - The state vector is extended with past vehicle positions and the estimation of the cross-correlation between current and previous vehicle states is maintained. With this approach improved data association is possible by combining data form various points
    - J. J. Leonard and R. J. Rikoski. Incorporation of delayed decision making into stochastic mapping
    - Stephan Williams, PhD Thesis, 2001, University of Sydney
  - Advantages:
    - No Loss of Information
    - Well suited to low frequency external sensors (ratio between vehicle velocity and feature rate information)
    - Absolutely necessary for some sensor modalities (eg, range-only, bearing-only)
  - Disadvantages:
    - Cost of augmenting state with past poses
    - The implementation is more complicated

Incorporation of New Features

- **We have the vehicle states and previous map**

$$P_0 = \begin{bmatrix} P^0_{v,v} & P^0_{v,m} \\ P^0_{m,v} & P^0_{m,m} \end{bmatrix}$$

We observed a new feature and the covariance and cross-covariance terms need to be evaluated

$$P_1 = \begin{bmatrix} P^0_{v,v} & P^0_{v,m} & ? \\ P^0_{m,v} & P^0_{m,m} & ? \\ ? & ? & ? \end{bmatrix}$$

Incorporation of New Features

- Approach 1

$$P_0 = \begin{bmatrix} P_{vv}^0 & P_{vm}^0 & 0 \\ P_{mv}^0 & P_{mm}^0 & 0 \\ 0 & 0 & A \end{bmatrix}$$

With A very large

$$W(k) = P(k/k-1)\nabla h_x^T(k) S^{-1}(k)$$
$$S(k) = \nabla h_x(k) P(k/k-1)\nabla h_x^T(k) + R$$
$$P(k/k) = P(k/k-1) - W(k)S(k)W^T(k)$$

$$P_1 = \begin{bmatrix} P_{vv}^1 & P_{vm}^1 & P_{vn}^1 \\ P_{mv}^1 & P_{mm}^1 & P_{mn}^1 \\ P_{nv}^1 & P_{nm}^1 & P_{nn}^1 \end{bmatrix}$$

- **Easy to understand and implement**
- **Very large values of A may introduce numerical problems**

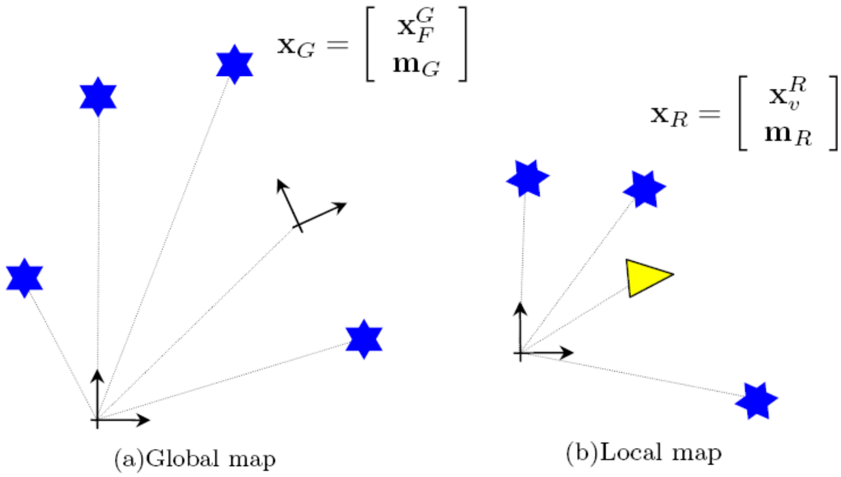
Analytical Approach

$$P_0 = \begin{bmatrix} P_{v,v}^0 & P_{v,m}^0 \\ P_{m,v}^0 & P_{m,m}^0 \end{bmatrix}$$

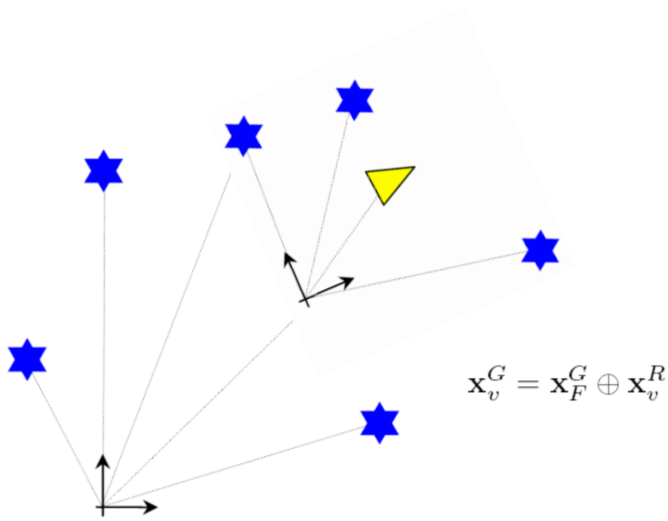
- **We can also evaluate the analytical expressions of the new terms**

$$P_1 = \begin{bmatrix} P_{v,v}^0 & P_{v,m}^0 & ? \\ P_{m,v}^0 & P_{m,m}^0 & ? \\ ? & ? & ? \end{bmatrix}$$

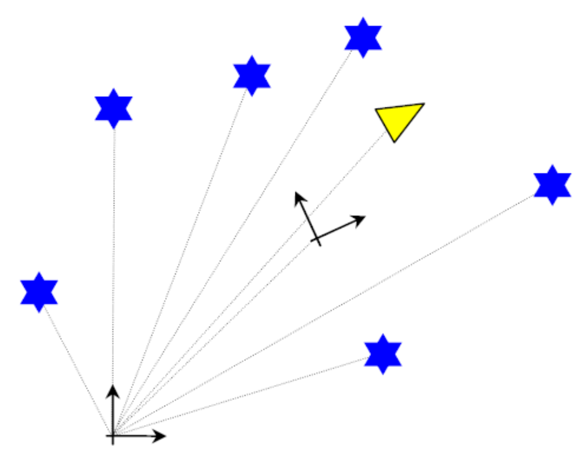
Constrained Local Submap Filter



CLSF Registration



CLSF Global Estimate



Motion Planning

METR 4202: **Robotics** & Automation

Dr Surya Singh -- Lecture # 10

October 5, 2016

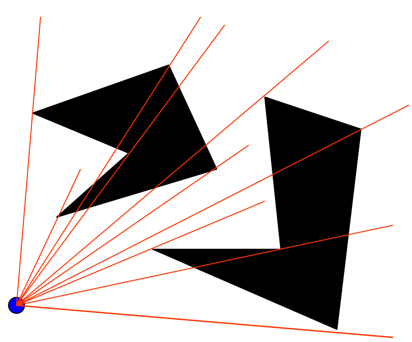
metr4202@itee.uq.edu.au
http://robotics.itee.uq.edu.au/~metr4202/ [http://**metr4202.com**]

Path-Planning Approaches

- Roadmap
  Represent the connectivity of the free space by a network of 1-D curves
- Cell decomposition
  Decompose the free space into simple cells and represent the connectivity of the free space by the adjacency graph of these cells
- Potential field
  Define a function over the free space that has a global minimum at the goal configuration and follow its steepest descent

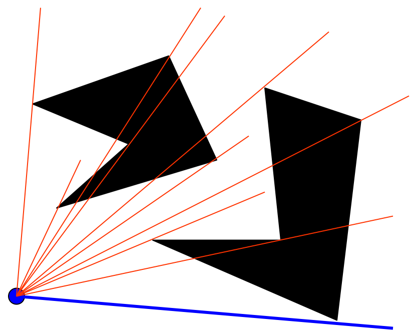
Slide from Latombe, CS326A

I. Rotational Sweep



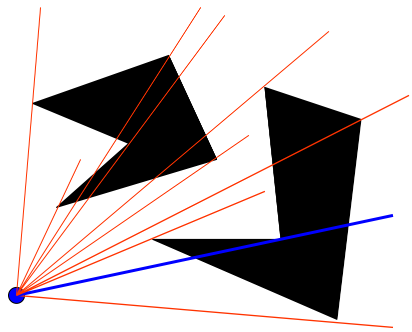
Slide from Latombe, CS326A

Rotational Sweep



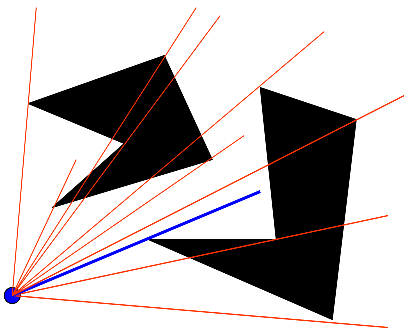
Slide from Latombe, CS326A

Rotational Sweep



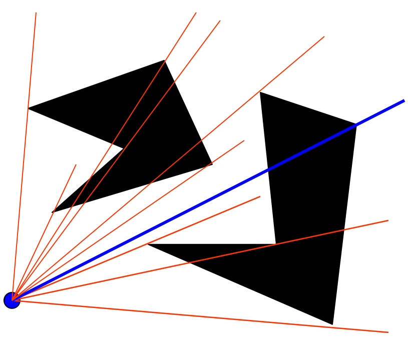
Slide from Latombe, CS326A

Rotational Sweep



Slide from Latombe, CS326A

Rotational Sweep



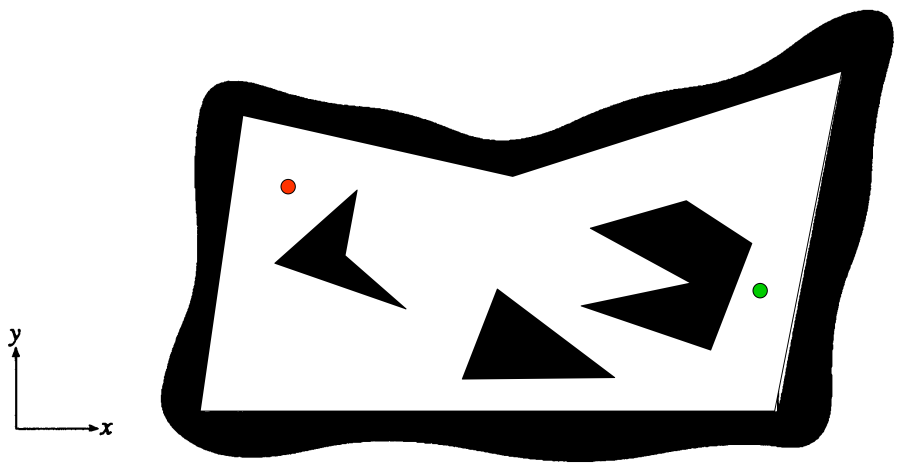
Slide from Latombe, CS326A

II. Cell-Decomposition Methods

Two classes of methods:

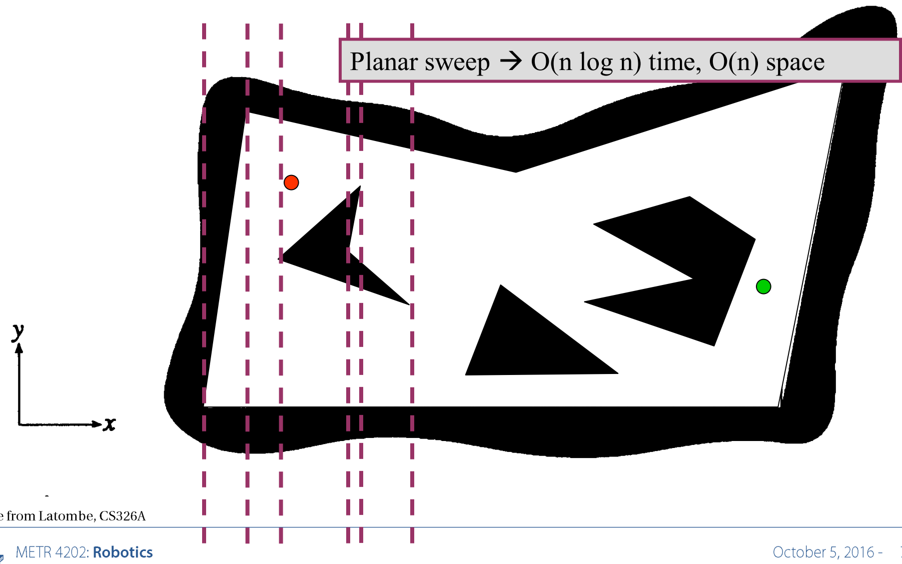
- Exact cell decomposition
  - The free space **F** is represented by a collection of non-overlapping cells whose union is exactly **F**
  - Example: trapezoidal decomposition
- Approximate cell decomposition
  - **F** is represented by a collection of non-overlapping cells whose union is contained in **F**
    Examples: quadtree, octree, 2n-tree

Trapezoidal decomposition



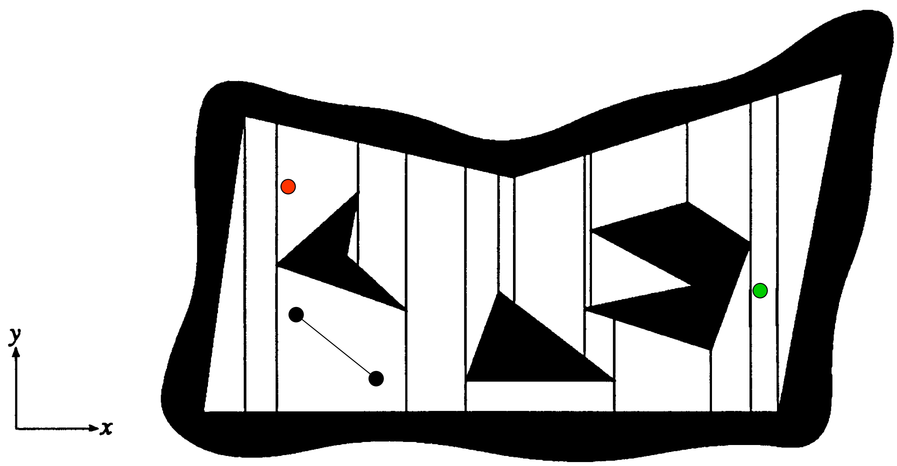
Slide from Latombe, CS326A

Trapezoidal decomposition



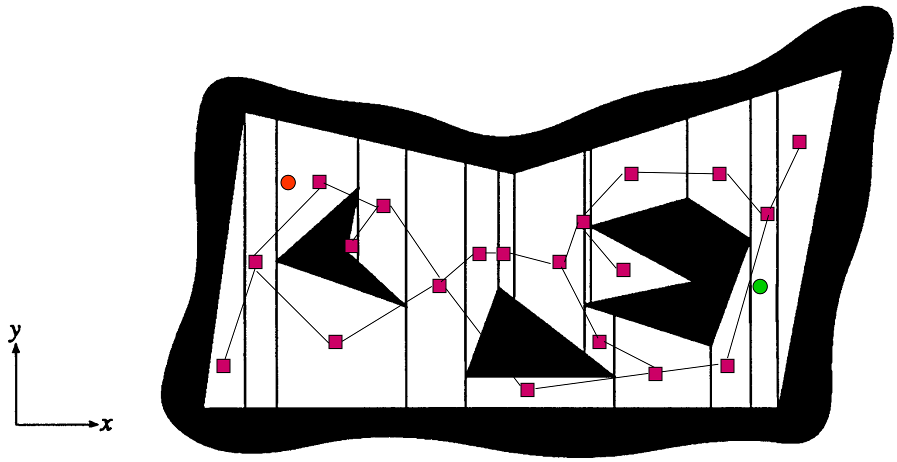
Slide from Latombe, CS326A

Trapezoidal decomposition



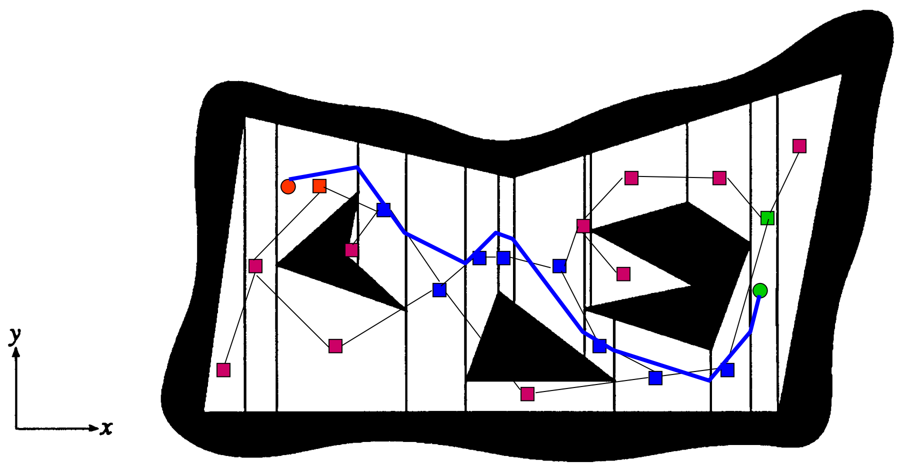
Slide from Latombe, CS326A

Trapezoidal decomposition



Slide from Latombe, CS326A

Trapezoidal decomposition



Slide from Latombe, CS326A

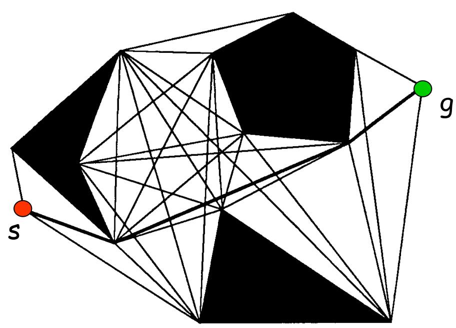
III. Roadmap Methods

- **Visibility graph**
- **Voronoi diagram**
- Silhouette
  First complete general method that applies to spaces of any dimension and is singly exponential in # of dimensions [Canny, 87]
- **Probabilistic roadmaps (PRMS)
  and Rapidly-exploring Randomized Trees (RRTs)**

Slide from Latombe, CS326A

Roadmap Methods

- **Visibility graph**
  Introduced in the Shakey project at SRI in the late 60s.
  Can produce shortest paths in 2-D configuration spaces

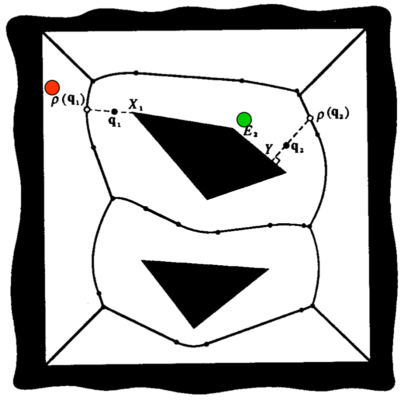


Slide from Latombe, CS326A

Roadmap Methods

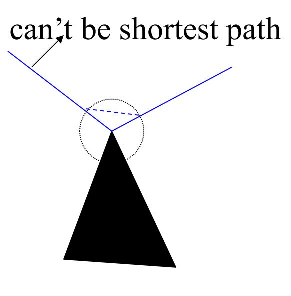
- Voronoi diagram Introduced by Computational Geometry researchers. Generate paths that maximizes clearance.

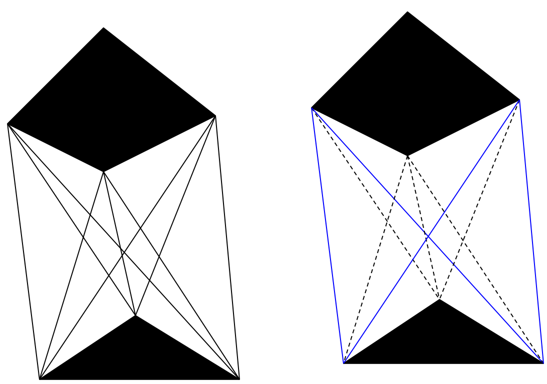
  O(n log n) time
  O(n) space



Slide from Latombe, CS326A

II.Visibility Graph



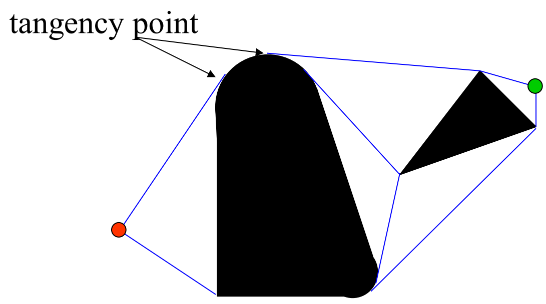


tangent segments

→ Eliminate concave obstacle vertices

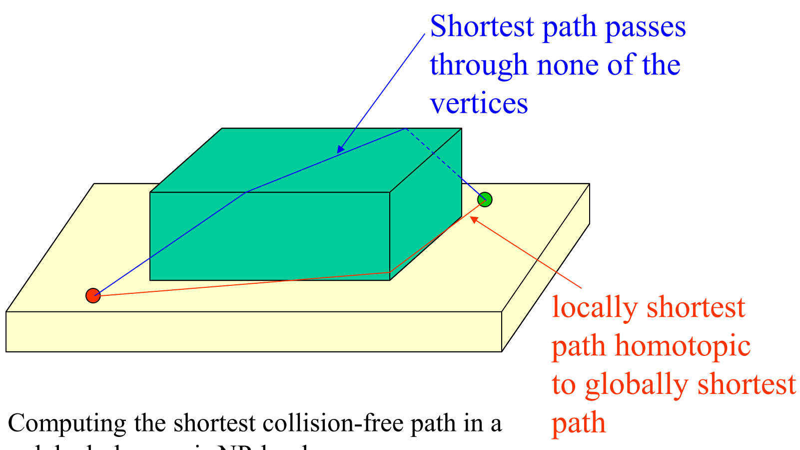
Slide from Latombe, CS326A

Generalized (Reduced) -- Visibility Graph



Slide from Latombe, CS326A

Three-Dimensional Space

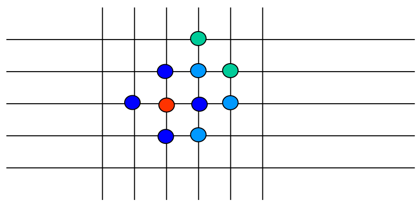


Computing the shortest collision-free path in a polyhedral space is NP-hard

Slide from Latombe, CS326A

Sketch of Grid Algorithm (with best-first search)

- Place regular grid G over space
- Search G using best-first search algorithm with potential as heuristic function



Slide from Latombe, CS326A

Simple Algorithm (for Visibility Graphs)

- Install all obstacles vertices in VG, plus the start and goal positions
- For every pair of nodes u, v in VG

  If segment(u,v) is an obstacle edge then

  insert (u,v) into VG

  else

  for every obstacle edge e

  if segment(u,v) intersects e

  then go up to segment

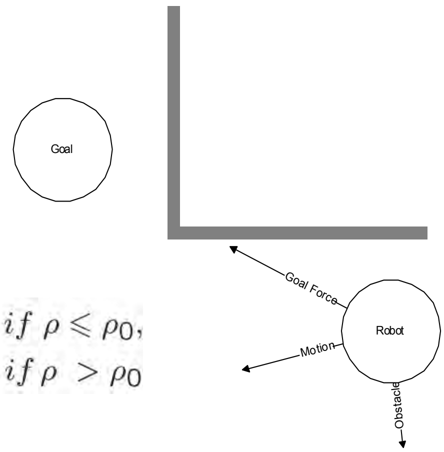
  insert (u,v) into VG
- Search VG using A*

Slide based on Latombe, CS326A

IV. Potential Field Methods

- Approach initially proposed for real-time collision avoidance [Khatib, 86]

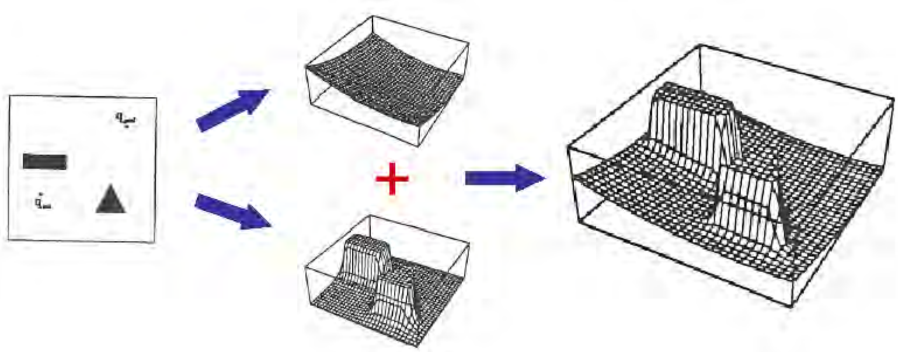
$$F_{Goal} = -k_p(x - x_{Goal})$$



$$F_{Obstacle} = \begin{cases} \eta \left(\frac{1}{\rho} - \frac{1}{\rho_0}\right) \frac{1}{\rho^2}\frac{\partial \rho}{\partial x} & if\ \rho \leqslant \rho_0, \\ 0 & if\ \rho > \rho_0 \end{cases}$$

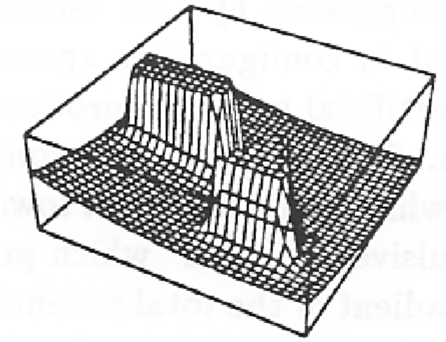
Slide based on Latombe, CS326A

Attractive and Repulsive fields



Slide from Latombe, CS326A

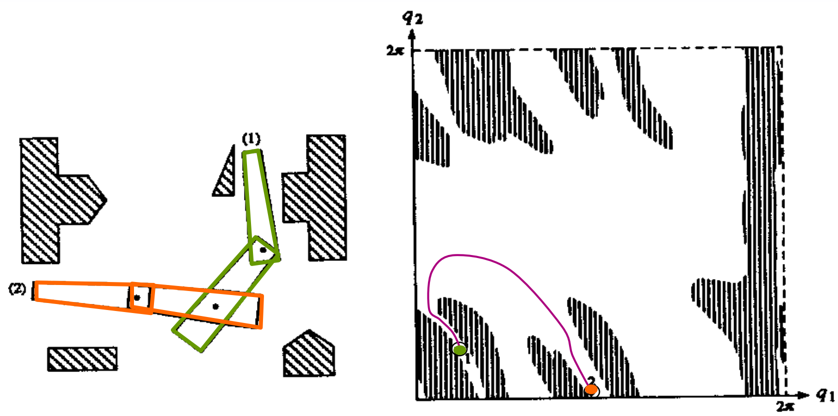
Local-Minimum Issue



- Perform best-first search (possibility of combining with approximate cell decomposition)
- Alternate descents and random walks
- Use local-minimum-free potential (navigation function)

Slide from Latombe, CS326A

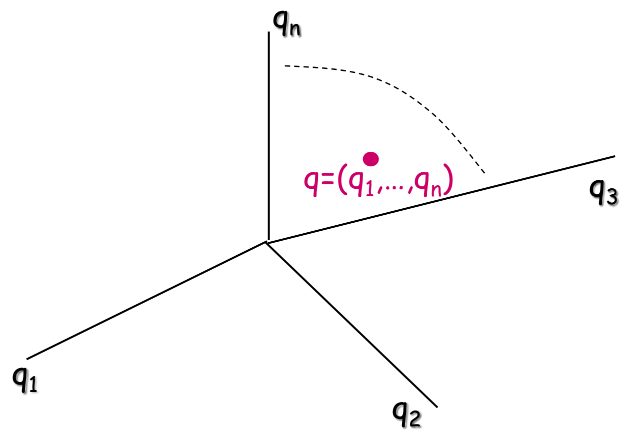
Configuration Space



- A robot configuration is a specification of the positions of all robot points relative to a fixed coordinate system
- Usually a configuration is expressed as a "vector" of position/orientation parameters

Slide from Latombe, CS326A

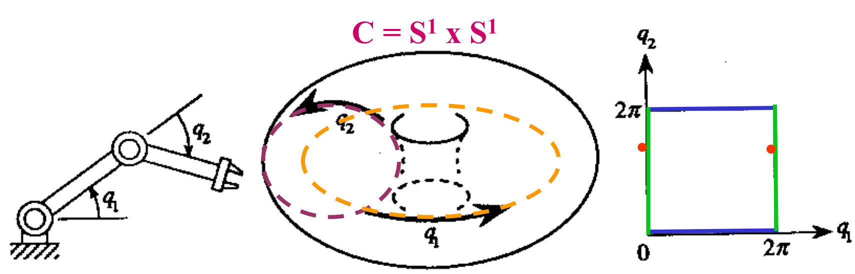
Motion Planning in C-Space



Slide from Latombe, CS326A

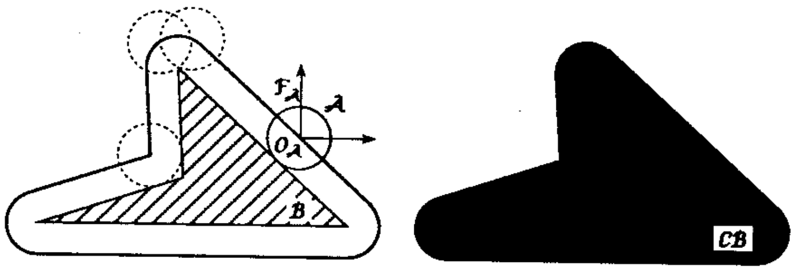
Configuration Space of a Robot

- Space of all its possible configurations
- But the topology of this space is usually not that of a Cartesian space



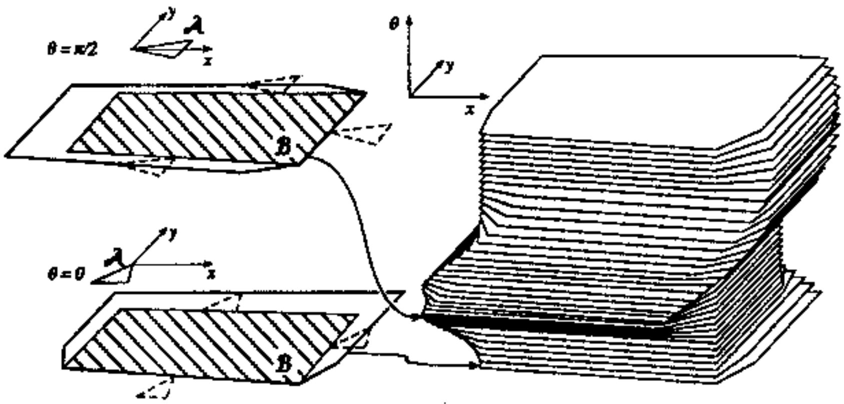
Slide from Latombe, CS326A

Disc Robot in 2-D Workspace



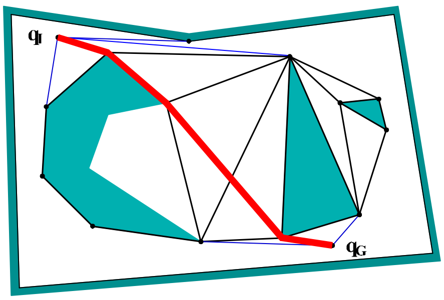
Slide from Latombe, CS326A

Rigid Robot Translating and Rotating in 2-D



Slide from Latombe, CS326A

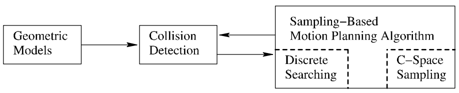
Geometric Planning Methods



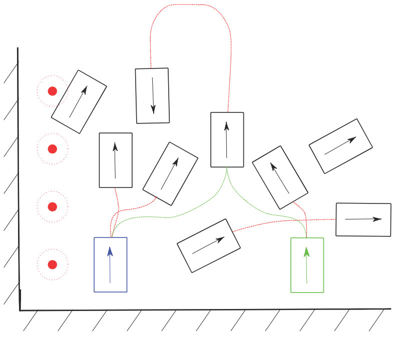
- Several Geometric Methods:
  - Vertical (Trapezoidal) Cell Decomposition
  - **Roadmap Methods**
    - **Cell (Triangular) Decomposition**
    - Visibility Graphs
    - Veroni Graphs

Artwork from LaValle, Ch. 6

Sample-Based Motion Planning

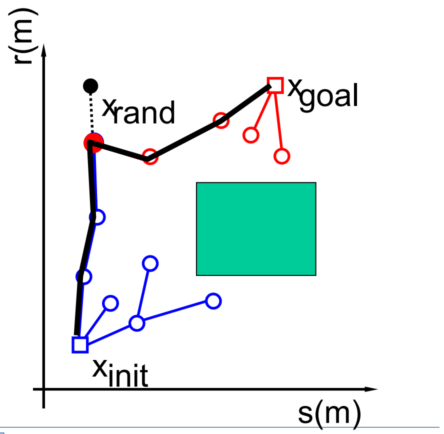


- PRMs
- RRTs

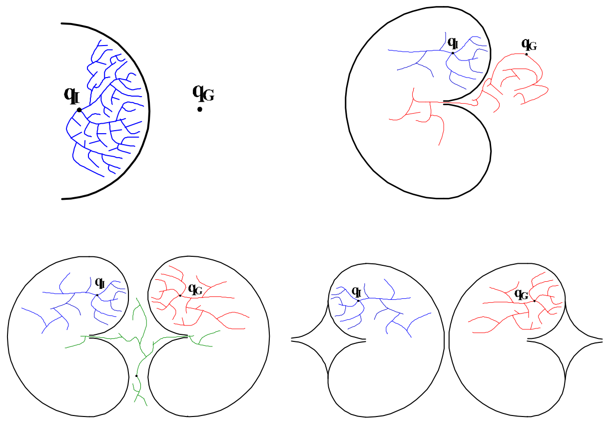


Artwork based on LaValle, Ch. 5

Rapidly Exploring Random Trees (RRT)

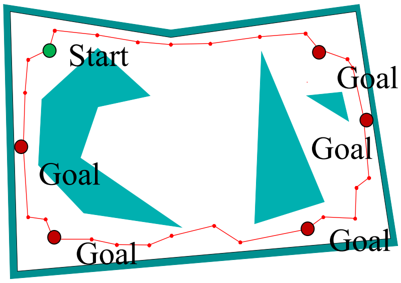


Sampling and the "Bug Trap" Problem



Artwork based on LaValle, Ch. 5

Multiple Points & Sequencing



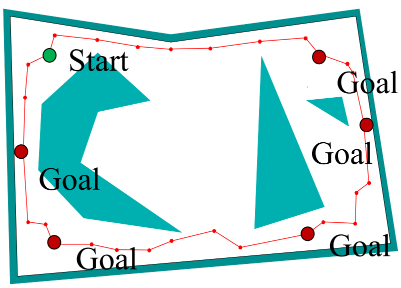
- Sequencing
  - Determining the “best” order to go in

➔ Travelling Salesman Problem

A salesman has to visit each city on a given list exactly once. In doing this, he **starts** from his home city and in the **end he has to return to his home** city. It is plausible for him to select the order in which he visits the cities so that the **total of the distances travelled** in his tour is as small as possible.

Artwork based on LaValle, Ch. 6

Travelling Salesman Problem

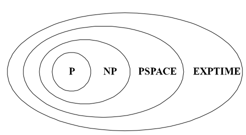


- Given a $n \times n$ distance matrix **C**=(c_{ij})

- Minimize:

$$c(\pi) = \sum_{i=1}^{n} c_{i\pi(i)}$$

- Note that this problem is NP-Hard

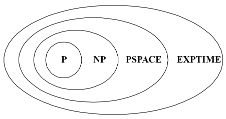


➔ BUT, Special Cases are Well-Solvable!

Artwork based on LaValle, Ch. 6

Travelling Salesman Problem [2]

- This problem is NP-Hard



➔ BUT,
Special Cases are Well-Solvable!

For the Euclidean case
(where the points are on the 2D Euclidean plane) :

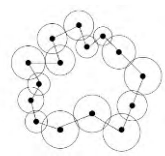
- The shortest TSP tour does not intersect itself, and thus geometry makes the problem somewhat easier.
- If all cities lie on the boundary of a convex polygon, the optimal tour is a cyclic walk along the boundary of the polygon (in clockwise or counterclockwise direction).

The k-line TSP

- The a special case where the cities lie on k parallel (or almost parallel) lines in the Euclidean plane.
- EG: Fabrication of printed circuit boards
- Solvable in $\mathbf{O}(n^3)$ time by Dynamic Programming (Rote's algorithm)

The necklace TSP

- The special Euclidean TSP case where there exist n circles around the n cities such that every cycle intersects exactly two adjacent circles



State-Space Modelling

METR 4202: **Robotics** & Automation

Dr Surya Singh -- Lecture # 11 **October 12, 2016**

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

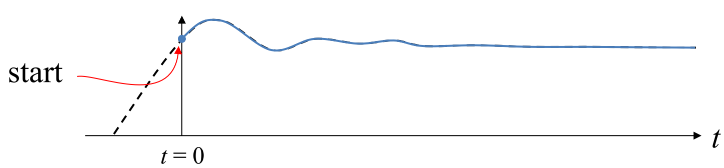
State-Space Modelling

("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?

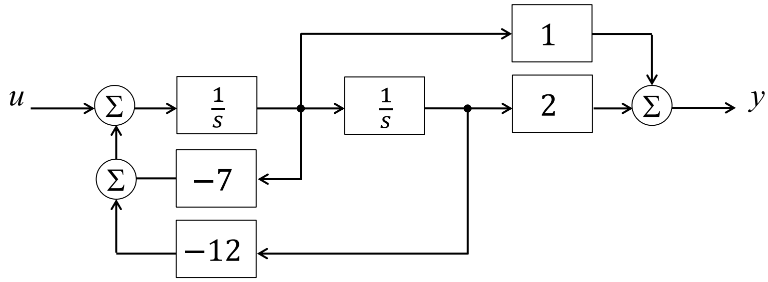
Eg. how would you setup a simulation of a step response, mid-step?



Introduction to state-space

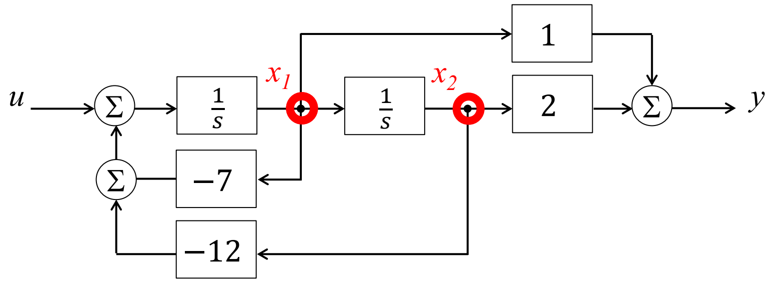
- Linear systems can be written as networks of simple dynamic elements:

$$H = \frac{s+2}{s^2+7s+12} = \frac{2}{s+4} + \frac{-1}{s+3}$$



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-space representation

- We can write linear systems in matrix form:

$$\begin{aligned} \dot{\boldsymbol{x}} &= \begin{bmatrix} -7 & 12 \\ 1 & 0 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u \\ \boldsymbol{y} &= \begin{bmatrix} 1 & 2 \end{bmatrix} \boldsymbol{x} + 0u \end{aligned}$$

Or, more generally:

$$\left.\begin{aligned} \dot{\boldsymbol{x}} &= \mathbf{A}\boldsymbol{x} + \mathbf{B}u \\ y &= \mathbf{C}\boldsymbol{x} + Du \end{aligned}\right\} \text{“State-space equations”}$$

State-space representation

- State-space matrices are not necessarily a unique representation of a system
  - There are two common forms
- Control canonical form
  - Each node – each entry in $\boldsymbol{x}$ – represents a state of the system (each order of s maps to a state)
- Modal form
  - Diagonals of the state matrix **A** are the poles (“modes”) of the transfer function

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:

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

We can write:

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

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

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

Controllability matrix

- To convert an arbitrary state representation in **F**, **G**, **H** and *J* to control canonical form **A**, **B**, **C** and *D*, the "controllability matrix"

$$\boldsymbol{\mathcal{C}} = [\mathbf{G} \quad \mathbf{FG} \quad \mathbf{F}^2\mathbf{G} \quad \cdots \quad \mathbf{F}^{n-1}\mathbf{G}]$$

must be nonsingular.

Why is it called the "controllability" matrix?

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:

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

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)

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

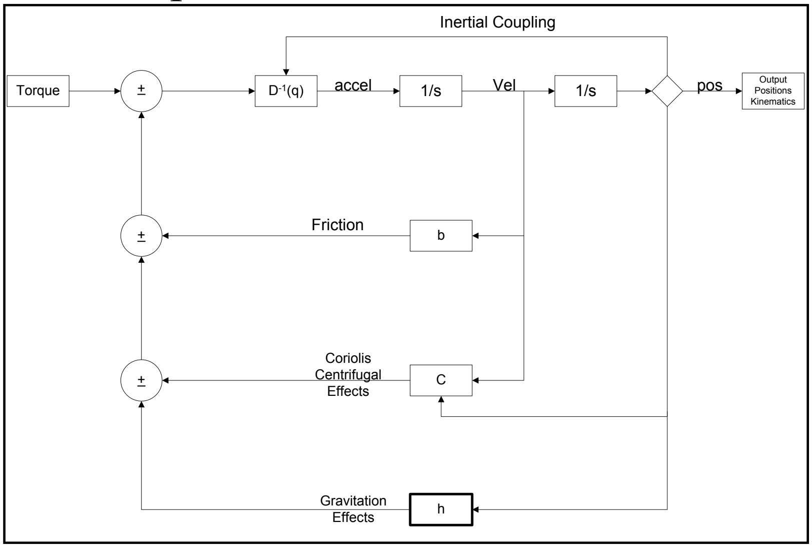
Considering Control:

- Each Link's torque is influenced by each other links motion
  - We say that the links are highly coupled

- Solution then suggests that control should come from a simultaneous solution of these torques

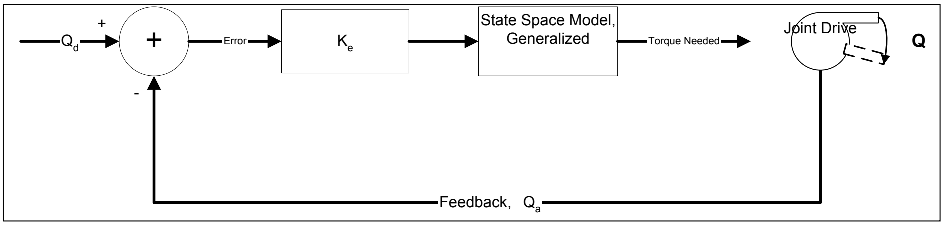
- We will model the solution as a "State Space" design and try to balance the torque-in with *positional control*-out – the most common way it is done!
  - But we could also use 'force control' to solve the control problem!

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:

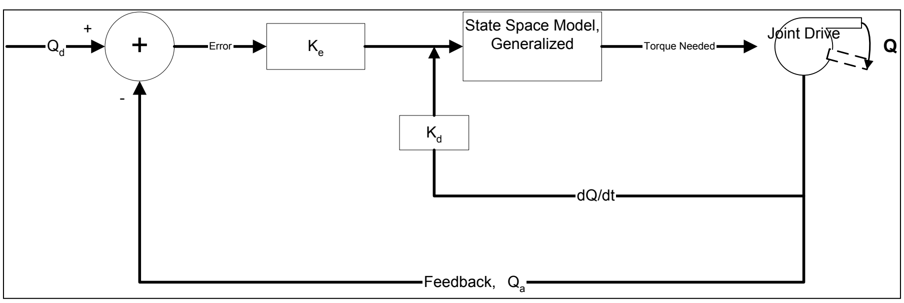
$$-k_{pe} * \theta_a = J\ddot{\theta} + F\dot{\theta}$$

the solution of which is:

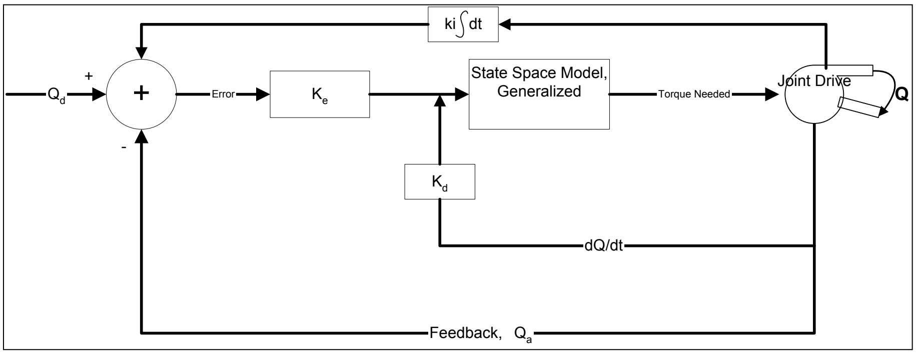
θ_a is a function of the servo feedback as a function of time!

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

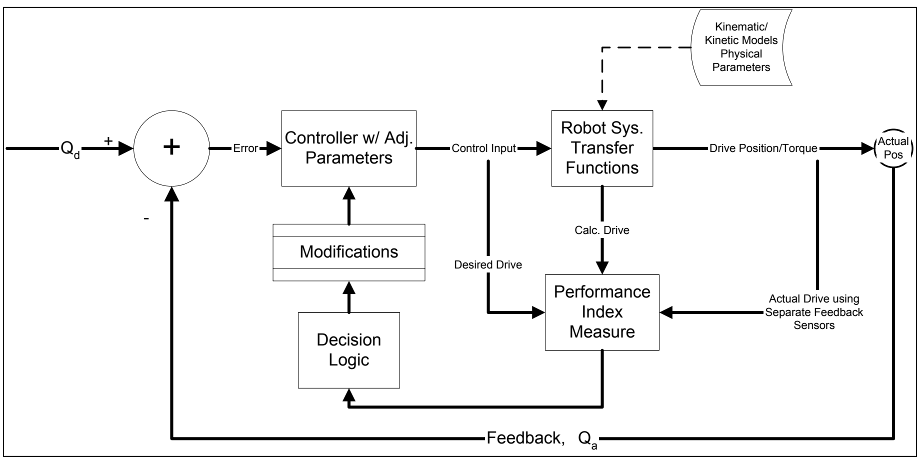
State Space Model of PD:



PID State Space Model:



State Model of Adjustable Controller



Controllability

Controllability matrix

- If you can write it in CCF, then the system equations must be linearly independent.
- Transformation by any nonsingular matrix preserves the controllability of the system.
- Thus, a nonsingular controllability matrix means $\boldsymbol{x}$ can be driven to any value.

State evolution

- Consider the system matrix relation:

$$\dot{\boldsymbol{x}} = \mathbf{F}\boldsymbol{x} + \mathbf{G}u$$
$$y = \mathbf{H}\boldsymbol{x} + Ju$$

The time solution of this system is:

$$\boldsymbol{x}(t) = e^{\mathbf{F}(t-t_0)}\, \boldsymbol{x}(t_0) + \int_{t_0}^{t} = e^{\mathbf{F}(t-\tau)}\, \mathbf{G}u(\tau)d\tau$$

If you didn't know, the matrix exponential is:

$$e^{\mathbf{K}t} = \mathbf{I} + \mathbf{K}t + \frac{1}{2!}\mathbf{K}^2t^2 + \frac{1}{3!}\mathbf{K}^3t^3 + \cdots$$

Stability

- We can solve for the natural response to initial conditions $\boldsymbol{x}_0$:

$$\boldsymbol{x}(t) = e^{p_i t}\boldsymbol{x}_0$$
$$\therefore \dot{\boldsymbol{x}}(t) = p_i e^{p_i t}\boldsymbol{x}_0 = \mathbf{F}e^{p_i t}\boldsymbol{x}_0$$

Clearly, a system will be stable provided
eig($\mathbf{F}$) < 0

Characteristic polynomial

- From this, we can see $\mathbf{F}\boldsymbol{x}_0 = p_i \boldsymbol{x}_0$

  or, $(p_i \mathrm{I} - \mathbf{F})\boldsymbol{x}_0 = 0$

  which is true only when $\det(p_i \mathrm{I} - \mathbf{F})\boldsymbol{x}_0 = 0$

  Aka. the characteristic equation!

- We can reconstruct the CP in s by writing:

$$\det(s\mathrm{I} - \mathbf{F})\boldsymbol{x}_0 = 0$$

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.

Just scratching the surface

- There is a lot of stuff to state-space control

- One lecture (or even two) can't possibly cover it all in depth

Go play with Matlab and check it out!

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

$\boldsymbol{\mathcal{C}} = [\boldsymbol{\Gamma} \quad \boldsymbol{\Phi}\boldsymbol{\Gamma} \quad \boldsymbol{\Phi}^2\boldsymbol{\Gamma} \quad \cdots \quad \boldsymbol{\Phi}^{n-1}\boldsymbol{\Gamma}]$ to be full-rank.

Solving State Space (optional notes) …

Time-invariant dynamics The simplest form of the general differential equation of the form (3.1) is the "homogeneous," i.e., unforced equation

$$\dot{x} = Ax \tag{3.2}$$

where A is a constant k by k matrix. The solution to (3.2) can be expressed as

$$x(t) = e^{At}c \tag{3.3}$$

where e^{At} is the matrix exponential function

$$e^{At} = I + At + A^2\frac{t^2}{2} + A^3\frac{t^3}{3!} + \cdots \tag{3.4}$$

and c is a suitably chosen constant vector. To verify (3.3) calculate the derivative of $x(t)$

$$\frac{dx(t)}{dt} = \frac{d}{dt}(e^{At})c \tag{3.5}$$

and, from the defining series (3.4),

$$\frac{d}{dt}(e^{At}) = A + A^2t + A^3\frac{t^2}{2!} + \cdots = A\left(I + At + A^2\frac{t^2}{2!} + \cdots\right) = A\,e^{At}$$

Thus (3.5) becomes

$$\frac{dx(t)}{dt} = Ae^{At}c = Ax(t)$$

Solving State Space (optional notes)

which was to be shown. To evaluate the constant c suppose that at some time τ the state $x(\tau)$ is given. Then, from (3.3),

$$x(\tau) = e^{A\tau}c \tag{3.6}$$

Multiplying both sides of (3.6) by the inverse of $e^{A\tau}$ we find that

$$c = (e^{A\tau})^{-1}x(\tau)$$

Thus the general solution to (3.2) for the state $x(t)$ at time t, given the state $x(\tau)$ at time τ, is

$$x(t) = e^{At}(e^{A\tau})^{-1}x(\tau) \tag{3.7}$$

The following property of the matrix exponential can readily be established by a variety of methods—the easiest perhaps being the use of the series definition (3.4)—

$$e^{A(t_1+t_2)} = e^{At_1}e^{At_2} \tag{3.8}$$

for any t_1 and t_2. From this property it follows that

$$(e^{A\tau})^{-1} = e^{-A\tau} \tag{3.9}$$

and hence that (3.7) can be written

$$x(t) = e^{A(t-\tau)}x(\tau) \tag{3.10}$$

Solving State Space (optional notes)

The matrix $e^{A(t-\tau)}$ is a special form of the *state-transition matrix* to be discussed subsequently.

We now turn to the problem of finding a "particular" solution to the nonhomogeneous, or "forced," differential equation (3.1) with A and B being constant matrices. Using the "method of the variation of the constant,"[1] we seek a solution to (3.1) of the form

$$x(t) = e^{At}c(t) \tag{3.11}$$

where $c(t)$ is a function of time to be determined. Take the time derivative of $x(t)$ given by (3.11) and substitute it into (3.1) to obtain:

$$Ae^{At}c(t) + e^{At}\dot{c}(t) = Ae^{At}c(t) + Bu(t)$$

or, upon cancelling the terms $A\,e^{At}c(t)$ and premultiplying the remainder by e^{-At},

$$\dot{c}(t) = e^{-At}Bu(t) \tag{3.12}$$

Thus the desired function $c(t)$ can be obtained by simple integration (the mathematician would say "by a quadrature")

$$c(t) = \int_T^t e^{-A\lambda}Bu(\lambda)\,d\lambda$$

The lower limit T on this integral cannot as yet be specified, because we will need to put the particular solution together with the solution to the

Solving State Space (optional notes)

homogeneous equation to obtain the complete (general) solution. For the present, let T be undefined. Then the particular solution, by (3.11), is

$$x(t) = e^{At}\int_T^t e^{-A\lambda}Bu(\lambda)\,d\lambda = \int_T^t e^{A(t-\lambda)}Bu(\lambda)\,d\lambda \tag{3.13}$$

In obtaining the second integral in (3.13), the exponential e^{At}, which does not depend on the variable of integration λ, was moved under the integral, and property (3.8) was invoked to write $e^{At}e^{-A\lambda} = e^{A(t-\lambda)}$.

The complete solution to (3.1) is obtained by adding the "complementary solution" (3.10) to the particular solution (3.13). The result is

$$x(t) = e^{A(t-\tau)}x(\tau) + \int_T^t e^{A(t-\lambda)}Bu(\lambda)\,d\lambda \tag{3.14}$$

We can now determine the proper value for lower limit T on the integral. At $t = \tau$ (3.14) becomes

$$x(\tau) = x(\tau) + \int_T^\tau e^{A(\tau-\lambda)}Bu(\lambda)\,d\lambda \tag{3.15}$$

Thus, the integral in (3.15) must be zero for any $u(t)$, and this is possible only if $T = \tau$. Thus, finally we have the complete solution to (3.1) when A and B are constant matrices

$$x(t) = e^{A(t-\tau)}x(\tau) + \int_\tau^t e^{A(t-\lambda)}Bu(\lambda)\,d\lambda \tag{3.16}$$

Solving State Space (optional notes)

This important relation will be used many times in the remainder of the book. It is worthwhile dwelling upon it. We note, first of all, that the solution is the sum of two terms: the first is due to the "initial" state $x(\tau)$ and the second—the integral—is due to the input $u(\tau)$ in the time interval $\tau \leqq \lambda \leqq t$ between the "initial" time τ and the "present" time t. The terms initial and present are enclosed in quotes to denote the fact that these are simply convenient definitions. There is no requirement that $t \geqq \tau$. The relationship is perfectly valid even when $t \leqq \tau$.

Another fact worth noting is that the integral term, due to the input, is a "convolution integral": the contribution to the state $x(t)$ due to the input u is the convolution of u with $e^{At}B$. Thus the function $e^{At}B$ has the role of the impulse response[1] of the system whose output is $x(t)$ and whose input is $u(t)$.

If the output y of the system is not the state x itself but is defined by the observation equation

$$y = Cx$$

then this output is expressed by

$$y(t) = Ce^{A(t-\tau)}x(t) + \int_{\tau}^{t} Ce^{A(t-\lambda)}Bu(\lambda)\,d\lambda \tag{3.17}$$

Solving State Space (optional notes)

and the impulse response of the system with y regarded as the output is $Ce^{A(t-\lambda)}B$.

The development leading to (3.16) and (3.17) did not really require that B and C be constant matrices. By retracing the steps in the development it is readily seen that when B and C are time-varying, (3.16) and (3.17) generalize to

$$x(t) = e^{A(t-\tau)}x(\tau) + \int_{\tau}^{t} e^{A(t-\lambda)}B(\lambda)u(\lambda)\,d\lambda \tag{3.18}$$

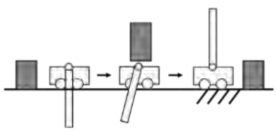
and

$$y(t) = C(t)\,e^{A(t-\tau)}x(\tau) + \int_{\tau}^{t} C(t)\,e^{A(t-\lambda)}B(\lambda)u(\lambda)\,d\lambda \tag{3.19}$$

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:

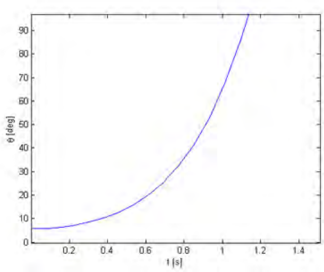
$$L = \frac{1}{2}(M+m)\dot{x}^2 - m\ell\dot{x}\dot{\theta}\cos\theta + \frac{1}{2}m\ell^2\dot{\theta}^2 - mg\ell\cos\theta$$

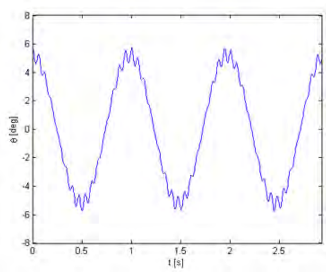
and the equations of motion are:

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

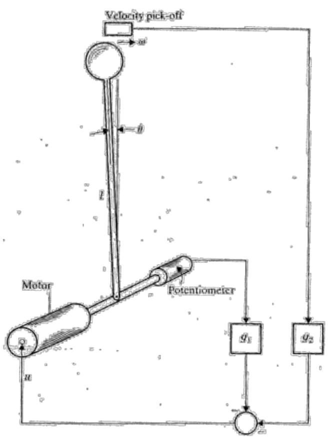
substituting L in these equations and simplifying leads to the equations that describe the motion o

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





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:

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

Stability

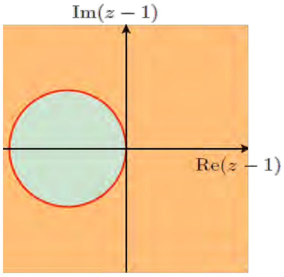
Fast sampling revisited

- For small T:

$$z = e^{sT} = 1 + sT + \frac{(sT)^2}{2} + \ldots \approx 1 + sT$$

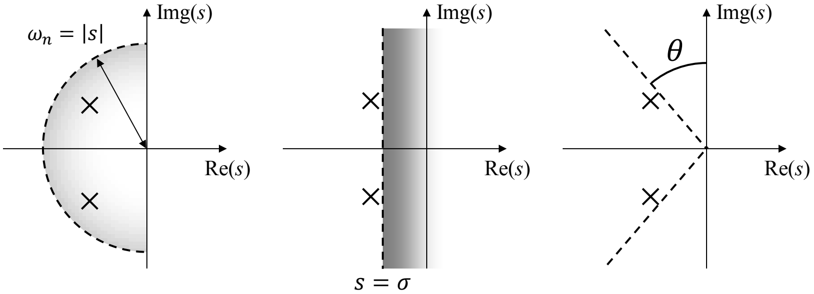
$$\rightarrow z \approx 1 + sT \rightarrow s = \frac{z-1}{T}$$

- Hence, the unit circle under the map from z to s-plane becomes:



Specification bounds

- Recall in the continuous domain, response performance metrics map to the s-plane:

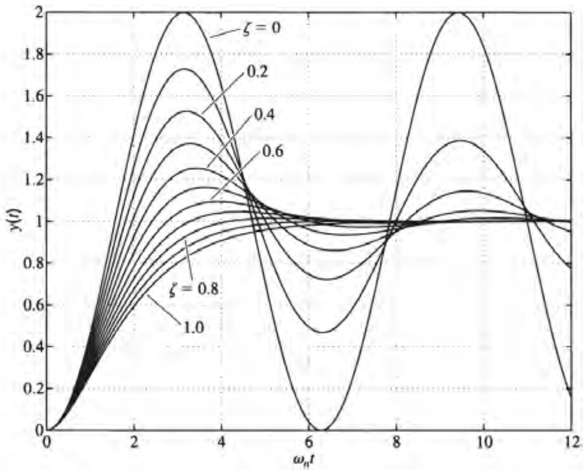


$|s| = \frac{1.8}{t_r}$ $s = \frac{4.6}{t_s}$ $\theta = \sin^{-1}\zeta$

2nd Order System Response

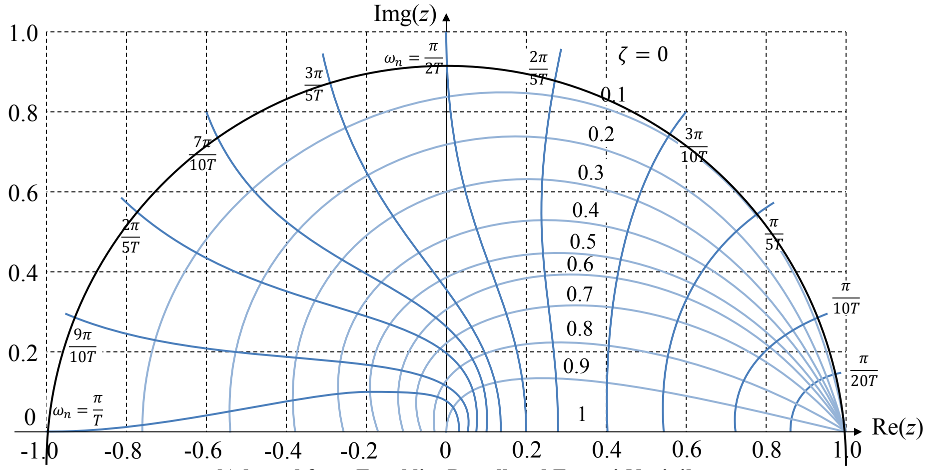
2nd Order System Response

- Response of a 2nd order system to increasing levels of damping:



Damping and natural frequency

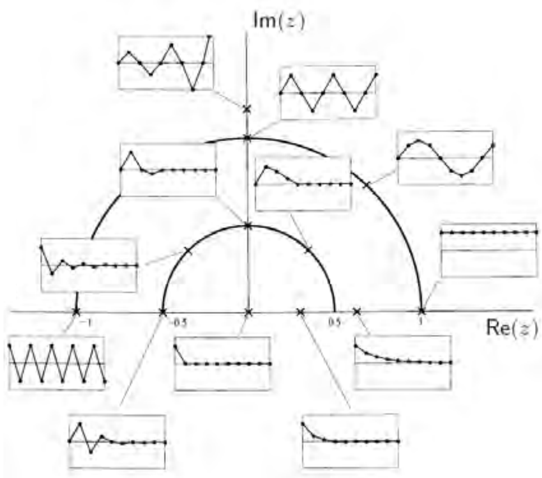
$$z = e^{sT} \text{ where } s = -\zeta\omega_n \pm j\omega_n\sqrt{1-\zeta^2}$$



[Adapted from Franklin, Powell and Emami-Naeini]

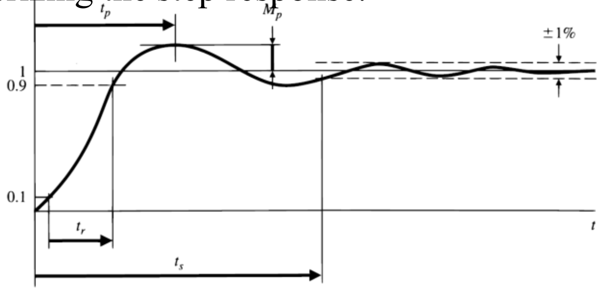
Pole positions in the z-plane

- Poles inside the unit circle are **stable**
- Poles outside the unit circle **unstable**
- Poles on the unit circle are oscillatory
- Real poles at 0 < z < 1 give exponential response
- Higher frequency of oscillation for larger
- Lower apparent damping for larer and r



2^{nd} Order System Specifications

Characterizing the step response:

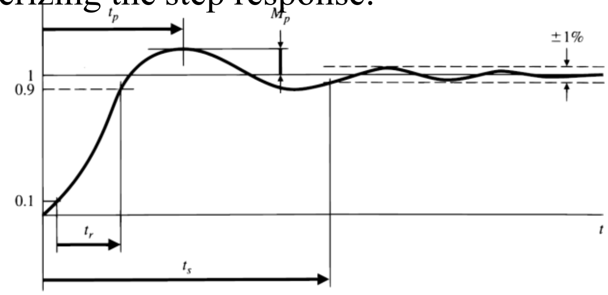


- Rise time (10% → 90%): $t_r \approx \frac{1.8}{\omega_0}$
- Overshoot: $M_p \approx \frac{e^{-\pi\zeta}}{\sqrt{1-\zeta^2}}$
- Settling time (to 1%): $t_s = \frac{4.6}{\zeta\omega_0}$
- Steady state error to unit step: e_{ss}
- Phase margin:

$$\phi_{PM} \approx 100\zeta$$

2^{nd} Order System Specifications

Characterizing the step response:



- Rise time (10% → 90%) & Overshoot:
  t_r, M_p → ζ, ω_0 : Locations of dominant poles
- Settling time (to 1%):
  t_s → radius of poles: $|z| < 0.01^{\frac{T}{t_s}}$
- Steady state error to unit step:
  e_{ss} → final value theorem $e_{ss} = \lim_{z\to 1} \{(z-1)F(z)\}$

Ex: System Specifications → Control Design [1/4]

Design a controller for a system with:

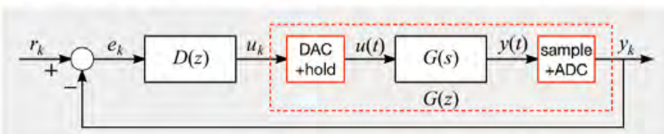
- A continuous transfer function: $G(s) = \dfrac{0.1}{s(s+0.1)}$
- A discrete ZOH sampler
- Sampling time (T_s): T_s= 1s
- C $u_k = -0.5u_{k-1} + 13(e_k - 0.88e_{k-1})$

The closed loop system is required to have:

- $M_p < 16\%$
- $t_s < 10$ s
- $e_{ss} < 1$

Ex: System Specifications → Control Design [2/4]

1. (a) Find the pulse transfer function of $G(s)$ plus the ZOH



$$G(z) = (1 - z^{-1})\mathcal{Z}\left\{\frac{G(s)}{s}\right\} = \frac{(z-1)}{z}\mathcal{Z}\left\{\frac{0.1}{s^2(s+0.1)}\right\}$$

e.g. look up $\mathcal{Z}\{a/s^2(s+a)\}$ in tables:

$$G(z) = \frac{(z-1)}{z}\frac{z\left((0.1 - 1 + e^{-0.1})z + (1 - e^{-0.1} - 0.1e^{-0.1})\right)}{0.1(z-1)^2(z - e^{-0.1})}$$

$$= \frac{0.0484(z + 0.9672)}{(z-1)(z-0.9048)}$$

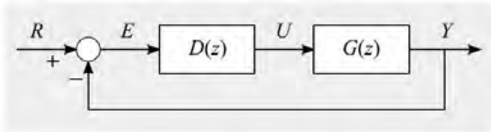
(b) Find the controller transfer function (using z = shift operator):

$$\frac{U(z)}{E(z)} = D(z) = 13\frac{(1 - 0.88z^{-1})}{(1 + 0.5z^{-1})} = 13\frac{(z - 0.88)}{(z + 0.5)}$$

Ex: System Specifications → Control Design [3/4]

2. Check the steady state error e_{ss} when r_k = unit ramp

$$e_{ss} = \lim_{k\to\infty} e_k = \lim_{z\to 1}(z-1)E(z)$$



$$\frac{E(z)}{R(z)} = \frac{1}{1+D(z)G(z)}$$

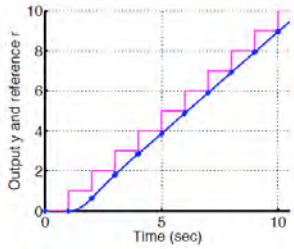
$$R(z) = \frac{Tz}{(z-1)^2}$$

so $$e_{ss} = \lim_{z\to 1}\left\{(z-1)\frac{Tz}{(z-1)^2}\frac{1}{1+D(z)G(z)}\right\} = \lim_{z\to 1}\frac{T}{(z-1)D(z)G(z)}$$

$$= \lim_{z\to 1}\frac{T}{(z-1)\dfrac{0.0484(z+0.9672)}{(z-1)(z-0.9048)}D(1)}$$

$$= \frac{1-0.9048}{0.0484(1+0.9672)D(1)} = 0.96$$

$\Longrightarrow$ $e_{ss} < 1$ (as required)



Ex: System Specifications → Control Design [4/4]

3. Step response: overshoot $M_p < 16\%$ $\Longrightarrow$ $\zeta > 0.5$

settling time $t_s < 10$ $\Longrightarrow$ $|z| < 0.01^{1/10} = 0.63$

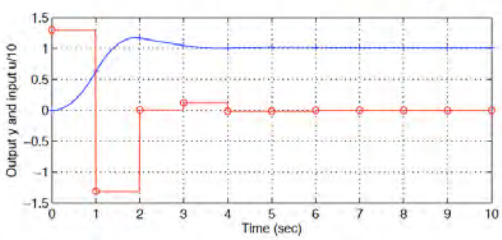
The closed loop poles are the roots of $1 + D(z)G(z) = 0$, i.e.

$$1 + 13\frac{(z-0.88)}{(z+0.5)}\frac{0.0484(z+0.9672)}{(z-1)(z-0.9048)} = 0$$

$$\Longrightarrow z = 0.88,\ -0.050 \pm j0.304$$

But the pole at $z = 0.88$ is cancelled by controller zero at $z = 0.88$, and

$$z = -0.050 \pm j0.304 = re^{\pm j\theta} \quad \Longrightarrow \begin{cases} r = 0.31,\ \theta = 1.73 \\ \zeta = 0.56 \end{cases}$$



all specs satisfied!

Discretization
→ Digital State Space

Digital State Space:

- Difference equations in state-space form:

$$x[n+1] = Ax[n] + Bu[n]$$
$$y[n] = Cx[n] + Du[n]$$

- Where:
  - u[n], y[n]: input & output (scalars)
  - x[n]: state vector

Discretisation FTW!

- We can use the time-domain representation to produce difference equations!

$$\boldsymbol{x}(kT+T) = e^{\mathbf{F}T}\ \boldsymbol{x}(kT) + \int_{kT}^{kT+T} e^{\mathbf{F}(kT+T-\tau)}\mathbf{G}u(\tau)d\tau$$

Notice $\boldsymbol{u}(\tau)$ is not based on a discrete ZOH input, but rather an integrated time-series.

We can structure this by using the form:

$$u(\tau) = u(kT), \qquad kT \le \tau \le kT+T$$

Discretisation FTW!

- Put this in the form of a new variable:

$$\eta = kT + T - \tau$$

Then:

$$\boldsymbol{x}(kT+T) = e^{\boldsymbol{F}T}\boldsymbol{x}(kT) + \left(\int_{kT}^{kT+T} e^{\boldsymbol{F}\eta}d\eta\right)\boldsymbol{G}u(kT)$$

Let's rename $\boldsymbol{\Phi} = e^{\boldsymbol{F}T}$ and $\boldsymbol{\Gamma} = \left(\int_{kT}^{kT+T} e^{\boldsymbol{F}\eta}d\eta\right)\boldsymbol{G}$

Discrete state matrices

So,

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

Again, $\boldsymbol{x}(k+1)$ is shorthand for $\boldsymbol{x}(kT+T)$

Note that we can also write $\mathbf{\Phi}$ as:

$$\mathbf{\Phi} = \mathbf{I} + \mathbf{F}T\mathbf{\Psi}$$

where

$$\mathbf{\Psi} = \mathbf{I} + \frac{\mathbf{F}T}{2!} + \frac{\mathbf{F}^2T^2}{3!} + \cdots$$

Simplifying calculation

- We can also use $\mathbf{\Psi}$ to calculate $\mathbf{\Gamma}$
  - Note that:

$$\Gamma = \sum_{k=0}^{\infty} \frac{\mathbf{F}^k T^k}{(k+1)!} T\mathbf{G}$$
$$= \mathbf{\Psi}T\mathbf{G}$$

$\mathbf{\Psi}$ itself can be evaluated with the series:

$$\mathbf{\Psi} \cong \mathbf{I} + \frac{\mathbf{F}T}{2}\left\{\mathbf{I} + \frac{\mathbf{F}T}{3}\left[\mathbf{I} + \cdots \frac{\mathbf{F}T}{n-1}\left(\mathbf{I} + \frac{\mathbf{F}T}{n}\right)\right]\right\}$$

State-space z-transform

We can apply the z-transform to our system:

$$(z\mathbf{I} - \mathbf{\Phi})\boldsymbol{X}(z) = \mathbf{\Gamma} U(k)$$
$$Y(z) = \mathbf{H}\boldsymbol{X}(z)$$

which yields the transfer function:

$$\frac{Y(z)}{\boldsymbol{X}(z)} = G(z) = \mathbf{H}(z\mathbf{I} - \mathbf{\Phi})^{-1}\mathbf{\Gamma}$$

Digital Control Law Design

In Chapter 2, we saw that the state-space description of a continuous system is given by (2.43),

$$\dot{\mathbf{x}} = \mathbf{F}\mathbf{x} + \mathbf{G}u, \tag{6.1}$$

and (2.44),

$$y = \mathbf{H}\mathbf{x}. \tag{6.2}$$

We assume the control is applied from the computer by a ZOH as shown in Fig. 1.1. Therefore, (6.1) and (6.2) have an exact discrete representation as given by (2.57),

$$\mathbf{x}(k+1) = \mathbf{\Phi}\mathbf{x}(k) + \mathbf{\Gamma}u(k),$$
$$y(k) = \mathbf{H}\mathbf{x}(k), \tag{6.3}$$

where

$$\boxed{\mathbf{\Phi} = e^{\mathbf{F}T},} \tag{6.4a}$$
$$\mathbf{\Gamma} = \int_0^T e^{\mathbf{F}\eta} d\eta \mathbf{G}, \tag{6.4b}$$

Discretisation FTW!

- We can use the time-domain representation to produce difference equations!

$$\boldsymbol{x}(kT+T) = e^{\mathbf{F}T}\,\boldsymbol{x}(kT) + \int_{kT}^{kT+T} e^{\mathbf{F}(kT+T-\tau)}\mathbf{G}u(\tau)d\tau$$

Notice $\boldsymbol{u}(\tau)$ is not based on a discrete ZOH input, but rather an integrated time-series.

We can structure this by using the form:

$$u(\tau) = u(kT), \qquad kT \le \tau \le kT+T$$

State-space z-transform

We can apply the z-transform to our system:

$$(z\mathbf{I} - \boldsymbol{\Phi})\boldsymbol{X}(z) = \boldsymbol{\Gamma}U(k)$$
$$Y(z) = \mathbf{H}\boldsymbol{X}(z)$$

which yields the transfer function:

$$\frac{Y(z)}{\boldsymbol{X}(z)} = G(z) = \mathbf{H}(z\mathbf{I} - \boldsymbol{\Phi})^{-1}\boldsymbol{\Gamma}$$

State-space control design -- Controllability

- 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 $\boldsymbol{\mathcal{C}} = [\boldsymbol{\Gamma} \quad \boldsymbol{\Phi\Gamma} \quad \boldsymbol{\Phi}^2\boldsymbol{\Gamma} \quad \cdots \quad \boldsymbol{\Phi}^{n-1}\boldsymbol{\Gamma}]$ to be full-rank.

Φ: Solving State Space

- In the conventional, frequency-domain approach the differential equations are converted to transfer functions as soon as possible
  - The dynamics of a system comprising several subsystems is obtained by combining the transfer functions!
- With the state-space methods, on the other hand, the description of the system dynamics in the form of differential equations is retained throughout the analysis and design.

State-transition matrix Φ(t)

- Describes how the state *x(t)* of the system at some time *t* evolves into (or from) the state x(τ) at some other time *T*.

$$x(t) = \Phi(t, \tau)\, x(\tau)$$

- $\mathbf{\Phi}(\mathrm{s}) = [\mathrm{s}\mathbf{I} - \mathbf{A}]^{-1}$ ➔ $\Phi(t) = e^{\boldsymbol{A}t}$

- Matrix Exponential:

$$e^{At} = \exp(At) = I + At + \frac{A^2t^2}{2!} + \cdots + \frac{A^k t^k}{k!} + \cdots$$

- Similar idea, but different result, for the control **u** ➔ Γ

Γ: Gamma: Comes from Integrating $\dot{\boldsymbol{x}}$

- $\Gamma = \left(\sum_{k=0}^{\infty} \frac{A^k T^{k+1}}{(k+1)!}\right) TB \approx \left(IT + A\frac{T^2}{2}\right) B$

Why?

- $x(t) = e^{A(t-t_0)}x(t_0) + \int_0^t e^{A(t-\tau)} Bu(\tau)\, d\tau$
- $x(kT+T) = e^{AT}x(kT) + \int_{kT}^{kT+T} e^{A(kt+t-\tau)} Bu(\tau) d\tau$
- u(t) is specified in terms of a continuous time history, though we often assume u(t) is a ZOH:
- $u(\tau) = u(kT) \Rightarrow$ Introduce $\eta = kT + T - \tau$
- $x(kT+T) = e^{AT}x(kT) + \int_{kT}^{kT+T} e^{F\eta} d\eta Bu(kT)$

➔$\Phi = e^{AT}$, $\Gamma = \int_0^T e^{A\eta} d\eta B$

Shaping the Dynamic Response

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**]

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:

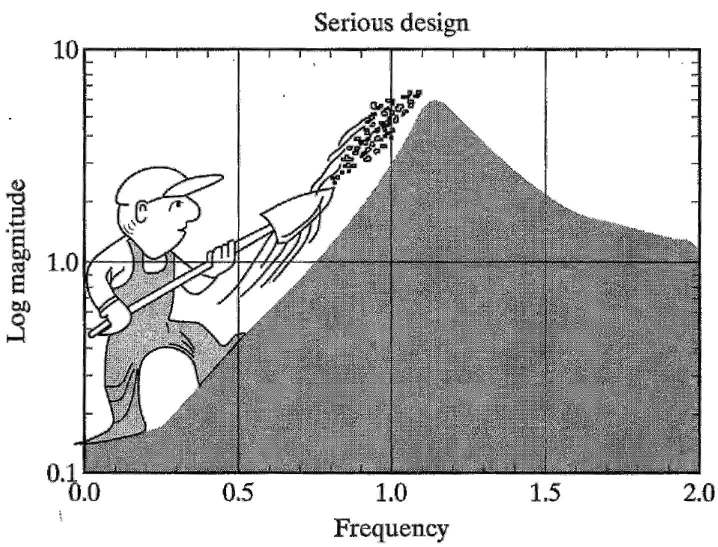
$$\det|zI - \Phi + \Gamma K| = 0$$

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…



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

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

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)

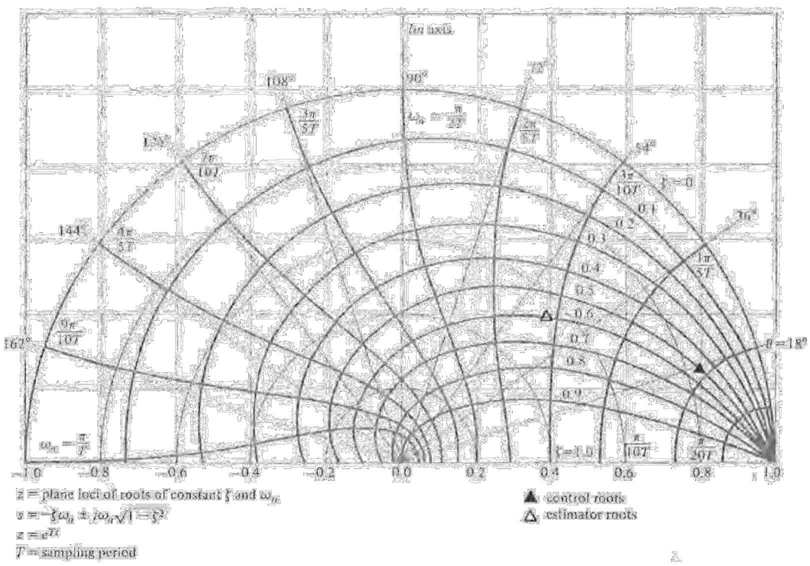
Equating coefficients in (6.9) and (6.10) with like powers of z, we obtain two simultaneous equations in the two unknown elements of $\mathbf{K}$:

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

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:

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

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

  - 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.

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”

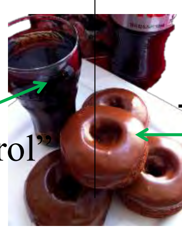
- Integral

  – Eliminates offsets (makes regulation ☺)

  – Leads to Oscillatory behaviour

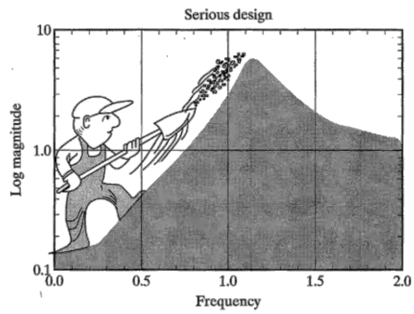
  – Adds an “order” but instability (Makes a 2nd order system 3rd order)

→ “**I**nteresting cake of control”



Seeing PID – 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.

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

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

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

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

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

Regulator Design

- Here the problem is to determine the gain matrix G in a linear feedback law $u = -Gx - G_0 x_0$
  - Where: x_0 is the vector of exogenous variables. The reason it is necessary to separate the exogenous variables from the process state x, rather than deal directly with the metastate $\mathbf{x} = \begin{bmatrix} x \\ x_0 \end{bmatrix}$ is that we must assume that the underlying process is controllable.
    - Since the exogenous variables are not true state variables, but additional inputs that cannot be affected by the control action, they cannot be included in the state vector when using a design method that requires controllability.
    - HOWEVER, they can be used in a process for Observability!
      ∴ when we are doing this as part of the sensing/mapping process!!

Regulator Design

- The assumption that all the state variables are accessible to measurement in the regulator means that the gain matrix G in is permitted to be any function of the state **x** that the design method requires

$$y = Cx$$
$$u = -G_y y$$
$$u = -G\hat{x}$$

  - Where: $\hat{x}$ is the state of an appropriate dynamic system known as an "observer."

SISO Regulator Design

- Design of a gain matrix

$$G = g' = [g_1, g_2, \ldots, g_k]$$

for the single-input, single-output system

$$\dot{x} = Ax + Bu$$

where

$$B = b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_k \end{bmatrix}$$

With the control law $u = -Gx = -g'x$ (6.7) becomes

$$\dot{x} = (A - bg')x$$

- Our objective is to find the matrix G = g' which places the poles of the closed-loop dynamics matrix $A_c = A - bg'$ at the locations desired.

SISO Regulator Design [2]

- One way of determining the gains would be to set up the characteristic polynomial for *Ac:*

$$|sI - A_c| = |sI - A + bg'| = s^k + \bar{a}_1 s^{k-1} + \cdots + \bar{a}_k$$

- The coefficients $a_1, a_2, \ldots, a_k$ of the powers of s in the characteristic polynomial will be functions of the k unknown gains. Equating these functions to the numerical values desired for $a_1, a_2, \ldots, a_k$ will result in k simultaneous equations the solution of which will yield the desired gains $g_1, \ldots, g_k$.

SISO Regulator Design [3]

If the original system is in the companion form given in (3.90), the task is particularly easy, because

$$A = \begin{bmatrix} -a_1 & -a_2 & \cdots & -a_{k-1} & -a_k \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \tag{6.11}$$

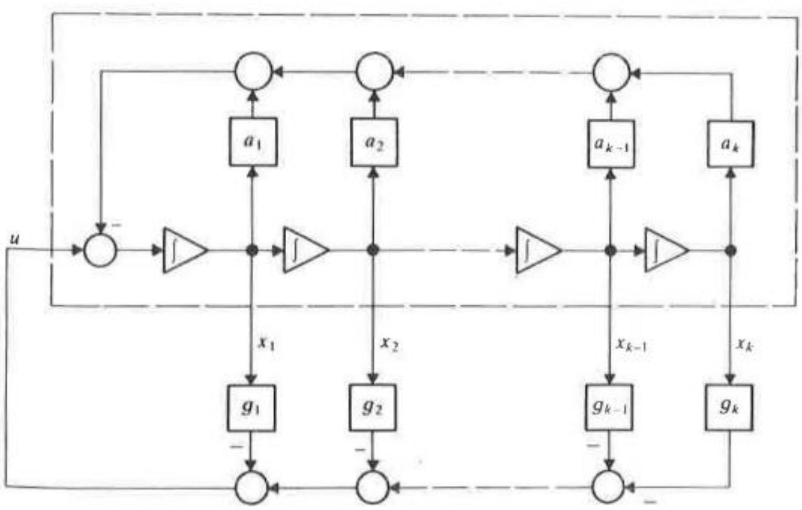
$$bg' = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} [g_1, g_2, \ldots, g_k] = \begin{bmatrix} g_1 & g_2 & \cdots & g_k \\ 0 & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

Hence

$$A_c = A - bg' = \begin{bmatrix} -a_1 - g_1 & -a_2 - g_2 & \cdots & -a_k - g_k \\ 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

The gains $g_1, \ldots, g_k$ are simply added to the coefficients of the open-loop A matrix to give the closed-loop matrix A_c. This is also evident from the block-diagram representation of the closed-loop system as shown in Fig. 6.1.

SISO Regulator Design [4]



SISO Regulator Design [4]

- But how to get this in companion form?

$$\bar{x} = Tx \tag{6.14}$$

Then, as shown in Chap. 3,

$$\dot{\bar{x}} = \bar{A}\bar{x} + \bar{b}u \tag{6.15}$$

where

$$\bar{A} = TAT^{-1} \qquad \text{and} \qquad \bar{b} = Tb$$

For the transformed system the gain matrix is

$$\bar{g} = \hat{a} - \bar{a} = \hat{a} - a \tag{6.16}$$

since $\bar{a} = a$ (the characteristic equation being invariant under a change of state variables). The desired control law in the original system is

$$u = -g'x = -g'T^{-1}\bar{x} = -\bar{g}'\bar{x} \tag{6.17}$$

From (6.17) we see that

$$\bar{g}' = g'T^{-1}$$

Thus the gain in the original system is

$$g = T'\bar{g} = T'(\hat{a} - a) \tag{6.18}$$

SISO Regulator Design [5]

In words, the desired gain matrix for a general system is the difference between the coefficient vectors of the desired and actual characteristic equation, premultiplied by the inverse of the transpose of the matrix T that transforms the general system into the companion form of (3.90), the A matrix of which has the form (6.11).

The desired matrix T is obtained as the product of two matrices U and V:

$$T = VU \tag{6.19}$$

The first of these matrices transforms the original system into an intermediate system

$$\dot{\tilde{x}} = \tilde{A}\tilde{x} \tag{6.20}$$

in the second companion form (3.107) and the second transformation U transforms the intermediate system into the first companion form.

Consider the intermediate system

$$\dot{\tilde{x}} = \tilde{A}\tilde{x} + \tilde{b}u \tag{6.21}$$

with $\tilde{A}$ and $\tilde{b}$ in the form of (3.107). Then we must have

$$\tilde{A} = UAU^{-1} \qquad \text{and} \qquad \tilde{b} = Ub \tag{6.22}$$

SISO Regulator Design [6]

The desired matrix U is precisely the inverse of the controllability test matrix Q of Sec. 5.4. To prove this fact, we must show that

$$U^{-1}\tilde{A} = AU^{-1} \tag{6.23}$$

or

$$Q\tilde{A} = AQ \tag{6.24}$$

Now, for a single-input system

$$Q = [b, Ab, \ldots, A^{k-1}b]$$

Thus, with $\tilde{A}$ given by (3.107), the left-hand side of (6.23) is

$$Q\tilde{A} = [b, Ab, \ldots, A^{k-1}b] \begin{bmatrix} 0 & 0 & \cdots & -a_k \\ 1 & 0 & \cdots & -a_{k-1} \\ 0 & 1 & \cdots & -a_{k-2} \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & -a_1 \end{bmatrix}$$

$$= [Ab, A^2b, \ldots, A^{k-1}b, -a_k b - a_{k-1}Ab - \cdots - a_k A^{k-1}b] \tag{6.25}$$

The last term in (6.25) is

$$(-a_k I - a_{k-1}A - \cdots - a_k A^{k-1})b \tag{6.26}$$

SISO Regulator Design [7]

Now, by the Cayley-Hamilton theorem, (see Appendix):

$$A^k = -a_1 A^{k-1} - a_2 A^{k-2} - \cdots - a_k I$$

so (6.26) is $A^k b$. Thus the left-hand side of (6.24) as given by (6.25) is

$$Q\tilde{A} = [Ab, A^2b, \ldots, A^k b] = A[b, Ab, \ldots, A^{k-1}b] = AQ$$

which is the desired result.

If the system is not controllable, then Q^{-1} does not exist and there is no general method of transforming the original system into the intermediate system (6.21); in fact it is not possible to place the closed-loop poles anywhere one desires. Thus, controllability is an essential requirement of system design by pole placement. If the system is *stabilizable* (i.e., the uncontrollable part is asymptotically stable, as discussed in Chap. 5) a stable closed-loop system can be achieved by placing the poles of the controllable subsystem where one wishes and accepting the pole locations of the uncontrollable subsystem. In order to apply the formula of this section, it is necessary to first separate the uncontrollable subsystem from the controllable subsystem.

The control matrix $\bar{b}$ of the intermediate system is given by

$$\bar{b} = Ub \tag{6.27}$$

We now show that

$$\bar{b} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{6.28}$$

SISO Regulator Design [8]

Multiply (6.28) by Q to obtain

$$Q\bar{b} = [b, Ab, \ldots, A^{k-1}b]\begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = b$$

which is the same as (6.27), since $Q^{-1} = U$.

The final step is to find the matrix V that transforms the intermediate system (6.21) into the final system (6.15). We must have

$$\bar{x} = V\bar{\bar{x}} \tag{6.29}$$

For the transformation (6.29) to hold, we must have

$$\bar{\bar{A}} = V\bar{A}V^{-1}$$

or

$$V^{-1}\bar{A} = \bar{\bar{A}}V^{-1} \tag{6.30}$$

SISO Regulator Design [9]

The matrix V^{-1} that satisfies (6.30) is the transpose of the upper left-hand k-by-k submatrix of the (triangular Toeplitz) matrix appearing in (3.103)

$$V^{-1} = \begin{bmatrix} 1 & a_1 & \cdots & a_{k-1} \\ 0 & 1 & \cdots & a_{k-2} \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} = W \tag{6.31}$$

To prove this, we note that the left-hand side of (6.30) is

$$V^{-1}\bar{A} = \begin{bmatrix} 1 & a_1 & \cdots & a_{k-1} \\ 0 & 1 & \cdots & a_{k-2} \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}\begin{bmatrix} -a_1 & -a_2 & \cdots & -a_k \\ 1 & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 & \cdots & 0 & -a_k \\ 1 & a_1 & \cdots & a_{k-2} & 0 \\ 0 & 1 & \cdots & a_{k-3} & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \tag{6.32}$$

(Note that the zeros in the first row of $V^{-1}\bar{A}$ are the result of the difference of

SISO Regulator Design [10]

two terms $a_1 = a_{11}$, $a_2 = a_{21}$, etc.) and the right-hand side of (6.30) is

$$\bar{A}V^{-1} = \begin{bmatrix} 0 & 0 & \cdots & -a_k \\ 1 & 0 & \cdots & -a_{k-1} \\ 0 & 1 & \cdots & -a_{k-2} \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & -a_1 \end{bmatrix} \begin{bmatrix} 1 & a_1 & \cdots & a_{k-1} \\ 0 & 1 & \cdots & a_{k-2} \\ 0 & 0 & \cdots & a_{k-3} \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 & \cdots & 0 & -a_k \\ 1 & a_1 & \cdots & a_{k-2} & 0 \\ 0 & 1 & \cdots & a_{k-3} & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix}$$

which is the same as (6.32). Thus (6.30) is proved.

We also need

$$\bar{b} = V\tilde{b}$$

We will show that

$$\tilde{b} = \bar{b}$$

Consider

$$\tilde{b} = V^{-1}\bar{b}$$

with

$$\tilde{b} = V^{-1}\bar{b} = \begin{bmatrix} 1 & a_1 & \cdots & a_{k-1} \\ 0 & 1 & \cdots & a_{k-2} \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

SISO Regulator Design [11]

Thus $\tilde{b}$ and $\bar{b}$ are the same.

The result of this calculation is that the transformation matrix T whose transpose is needed in (6.18) is the inverse of the product of the controllability test matrix and the triangular matrix (6.31).

The above results may be summarized as follows. The desired gain matrix g, by (6.18) and (6.19), is given by

$$g = (VU)'(\hat{a} - a) \tag{6.33}$$

where

$$V = W^{-1} \quad \text{and} \quad U = Q^{-1}$$

Thus

$$VU = W^{-1}Q^{-1} = (QW)^{-1}$$

How to Get the Gains?

Ackermann's Formula (FPW p. 245) [**ELEC3004**]

- Gains maybe approximated with:

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

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

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

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

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

  - α_i: coefficients of the desired characteristic equation

Ackermann's Formula [2] (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

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

and therefore

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

Furthermore, we find that

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

and

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

and finally

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

therefore

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

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

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

→ This case is known as the matrix exponential function

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

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

- 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:

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

State Equation Solution

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

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

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:

$$L[\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t)]$$
$$L[\dot{\mathbf{x}}(t)] = L[\mathbf{A}\mathbf{x}(t)] + L[\mathbf{B}\mathbf{u}(t)]$$
$$s\mathbf{X}(s) - \mathbf{x}_0 = \mathbf{A}\mathbf{X}(s) + \mathbf{B}\mathbf{U}(s)$$
$$\mathbf{X}(s) = (s\mathbf{I}-\mathbf{A})^{-1}\mathbf{x}_0 + (s\mathbf{I}-\mathbf{A})^{-1}\mathbf{B}\mathbf{U}(s)$$
$$\mathbf{X}(s) = \mathbf{\Phi}(s)\mathbf{x}_0 + \mathbf{\Phi}(s)\mathbf{B}\mathbf{U}(s)$$

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

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.

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

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

$$\mathbf{A}=\begin{bmatrix}0 & 1\\ -12 & -7\end{bmatrix}$$

B

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

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

C D

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

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

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

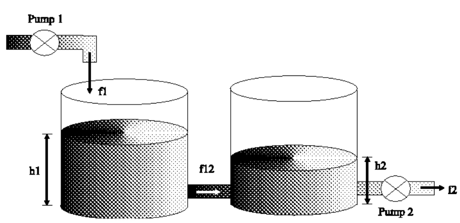
$$X_1(s)=\frac{0.1s^2+0.75s+3}{s(s+3)(s+4)}=\frac{0.25}{s}-\frac{0.55}{s+3}+\frac{0.4}{s+4}$$

$$X_2(s)=\frac{0.05s+1.8}{(s+3)(s+4)}=\frac{1.65}{s+3}-\frac{1.60}{s+4}$$

Water Tank Example

Example

- Figure 18.1: Schematic diagram of two coupled tanks



- Water flows into the first tank through pump 1 a rate fi(t) that obviously affects the head of water in tank 1 (denoted by h1(t)). Water flows out of tank 1 into tank 2 at a rate f12(t), affecting both h1(t) and h2(t). Water than flows out of tank 2 at a rate fe controlled by pump 2.

- Given this information, the challenge is to build a virtual sensor (or observer) to estimate the height of liquid in tank 1 from measurements of the height of liquid in tank 2 and the flows f1(t) and f2(t).

- Before we continue with the observer design, we first make a model of the system. The height of liquid in tank 1 can be described by the equation

$$\frac{dh_1(t)}{dt} = \frac{1}{A}(f_i(t) - f_{12}(t))$$

- Similarly, h2(t) is described by

$$\frac{dh_2(t)}{dt} = \frac{1}{A}(f_{12}(t) - f_e)$$

- The flow between the two tanks can be approximated by the free-fall velocity for the difference in height between the two tanks:

$$f_{12}(t) = \sqrt{2g(h_1(t) - h_2(t))}$$

- We can linearize this model for a nominal steady-state height difference (or operating point). Let

- This yields the following linear model:

$$h_1(t) - h_2(t) = \Delta h(t) = H + h_d(t)$$

- where

$$\frac{d}{dt}\begin{bmatrix} h_1(t) \\ h_2(t) \end{bmatrix} = \begin{bmatrix} -k & k \\ k & -k \end{bmatrix}\begin{bmatrix} h_1(t) \\ h_2(t) \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}\begin{bmatrix} f_1(t) - \frac{K\sqrt{H}}{2} \\ f_2(t) + \frac{K\sqrt{H}}{2} \end{bmatrix}$$

$$k = \frac{K}{2\sqrt{H}}$$

- We are assuming that h2(t) can be measured and h1(t) cannot, so we set C = [0 1] and D = [0 0]. The resulting system is both controllable and observable (as you can easily verify). Now we wish to design an observer

$$J = \begin{bmatrix} J_1 \\ J_2 \end{bmatrix}$$

- to estimate the value of h2(t). The characteristic polynomial of the observer is readily seen to be

$$s^2 + (2k + J_1)s + J_2 k + J_1 k$$

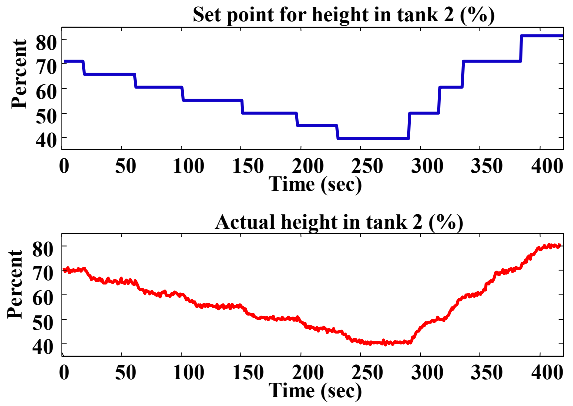
- so we can choose the observer poles; that choice gives us values for J1 and J2.

- If we assume that the operating point is H = 10%, then k = 0.0411. If we wanted poles at s = -0.9291 and s = -0.0531, then we would calculate that J1 = 0.3 and J2 = 0.9. If we wanted two poles at s = -2, then J2 = 3.9178 and J1 = 93.41.

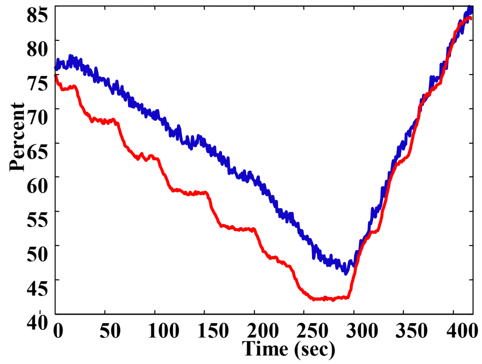
- The equation for the final observer is then

$$\frac{d}{dt}\begin{bmatrix}\hat{h}_1(t)\\ \hat{h}_2(t)\end{bmatrix} = \begin{bmatrix}-k & k\\ k & -k\end{bmatrix}\begin{bmatrix}\hat{h}_1(t)\\ \hat{h}_2(t)\end{bmatrix} + \begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}\begin{bmatrix}f_1(t) - \frac{K\sqrt{H}}{2}\\ f_2(t) + \frac{K\sqrt{H}}{2}\end{bmatrix} + J(h_2(t) - \hat{h}_2(t))$$

• The data below has been collected from the real system shown earlier



• The performance of the observer for tank height is compared below with the true tank height which is actually measured on this system.



Actual height in tank 1 (*blue*), Observed height in tank 1 (*red*)

Revisiting Pole Placement

Pole Assignment by State Feedback

- We begin by examining the problem of closed-loop pole assignment. For the moment, we make a simplifying assumption that all of the system states are measured. We will remove this assumption later. We will also assume that the system is completely controllable. The following result then shows that the closed-loop poles of the system can be arbitrarily assigned by feeding back the state through a suitably chosen constant-gain vector.

State-Feedback Control Objectives

- Regulation: Force state x to equilibrium state (usually 0) with a desirable dynamic response.

- Tracking: Force the output of the system y to tracks a given desired output y_d with a desirable dynamic response.

Pole Placement Problem as an Eignenvalue Problem

Choose the state feedback gain to place the poles of the closed-loop system, i.e.,

$$\textbf{Eigenvalues of } \overline{G} := G - HK$$

At specified locations $\lambda_1^{des}, \ldots, \lambda_n^{des}$

State Feedback Control of a System in CCF

Consider a SISO system in CCF: $\hat{\mathbf{x}}(k+1) = \mathbf{G}_c \hat{\mathbf{x}}(k) + \mathbf{H}_c \mathbf{u}$

$$\mathbf{G}_c = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 1 \\ -a_n & -a_{n-1} & \cdots & -a_2 & -a_1 \end{bmatrix}, \mathbf{H}_c = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$$

$$\Phi(s) = |zI - \mathbf{G}| = z^n + a_1 z^{n-1} + \cdots + a_{n-1} z + a_n$$

State Feedback Control

$$\mathbf{u} = -\mathbf{K}\mathbf{x} + \mathbf{r}, \quad \mathbf{K} = \begin{bmatrix} k_1 & \cdots & k_{n1} \end{bmatrix}$$

Closed-Loop CCF System

Closed loop A matrix:

$$\overline{\mathbf{G}} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 1 \\ -a_n & -a_{n-1} & \cdots & -a_2 & -a_1 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} k_1 & \cdots & k_n \end{bmatrix}$$

$$\overline{\mathbf{G}} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 1 \\ -a_n + k_1 & -a_{n-1} + k_2 & \cdots & -a_2 + k_{n-1} & -a_1 + k_n \end{bmatrix}$$

Choosing the Gain-CCF

Closed-loop Characteristic Equation

$$\Phi(z) = z^n + (a_1 + k_n)z^{n-1} + \cdots + (a_{n-1} + k_2)z + (a_n + k_1)$$

Desired Characteristic Equation:

$$\Phi^{des}(z) = \prod_{i=1}^{n}\left(z - \lambda_i^{des}\right) = z^n + a_1^{des} z^{n-1} + \cdots + a_{n-1}^{des} z + a_n^{des}$$

Control Gains:

$$\boxed{\mathbf{K_i = a_{n-i+1}^{des} - a_{n-i+1}, \quad i = 1,2,\ldots,n}}$$

Transformation to CCF

Transform system $\dot{\mathbf{x}} = \mathbf{Gx} + \mathbf{Hu}$ To CCF

$$\hat{\mathbf{x}}^+ = \mathbf{G}_c\hat{\mathbf{x}} + \mathbf{H}_c\mathbf{u} \Rightarrow \begin{cases} \hat{x}_1^{\ +} = \hat{x}_2 \\ \hat{x}_2^{\ +} = \hat{x}_3 \\ \vdots \\ \hat{x}_n^{\ +} = -a_n\hat{x}_1 - a_{n-1}\hat{x}_2 - \cdots - a_1\hat{x}_n + u \end{cases}$$

Where $x^+(k)=x(k+1)$ (for simplicity)

First, find how new state z_1 is related to x:

$$\hat{\mathbf{x}}_1 = \mathbf{px}, \quad \mathbf{p} = [p_1 \quad \cdots \quad p_n] \quad \text{(row vector)}$$

Transformed State Equations

Necessary Conditions for p:

$$
\begin{cases}
\hat{x}_1^{\,+} = \mathbf{p}\mathbf{x}^{+} = \mathbf{p}\mathbf{G}\mathbf{x} + \cancelto{0}{\mathbf{p}\mathbf{H}}u = \hat{x}_2 \\
\hat{x}_2^{\,+} = \mathbf{p}\mathbf{G}\mathbf{x}^{+} = \mathbf{p}\mathbf{G}^2\mathbf{x} + \cancelto{0}{\mathbf{p}\mathbf{G}\mathbf{H}}\,u = \hat{x}_3 \\
\quad\vdots \\
\hat{x}_{n-1}^{\,+} = \mathbf{p}\mathbf{G}^{n-2}\mathbf{x}^{+} = \mathbf{p}\mathbf{G}^{n-1}\mathbf{x} + \cancelto{0}{\mathbf{p}\mathbf{G}^{n-2}\mathbf{H}}u = \hat{x}_n \\
\hat{x}_n^{\,+} = \mathbf{p}\mathbf{G}^{n-1}\mathbf{x}^{+} = \mathbf{p}\mathbf{G}^{n}\mathbf{x} + \cancelto{1}{\mathbf{p}\mathbf{G}^{n-1}\mathbf{H}}u
\end{cases}
$$

$$\mathbf{p}\begin{bmatrix}\mathbf{H} & \mathbf{G}\mathbf{H} & \cdots & \mathbf{G}^{n-1}\mathbf{H}\end{bmatrix} = \begin{bmatrix}\mathbf{0} & \cdots & \mathbf{0} & \mathbf{1}\end{bmatrix}$$

Vector p can be found if the system is controllable: $\boxed{\mathbf{p} = \mathbf{e}_n{}^{T}\mathbf{M}^{-1}}$

State Transformation Invertibility

State transformation:

$$\begin{bmatrix}\hat{x}_1 \\ \hat{x}_2 \\ \vdots \\ \hat{x}_n\end{bmatrix} = \mathbf{T}\mathbf{x} = \begin{bmatrix}\mathbf{p} \\ \mathbf{p}\mathbf{G} \\ \vdots \\ \mathbf{p}\mathbf{G}^{n-1}\end{bmatrix}\mathbf{x}$$

Matrix T is invertible since

$$\begin{bmatrix}\mathbf{p} \\ \mathbf{p}\mathbf{G} \\ \vdots \\ \mathbf{p}\mathbf{G}^{n-1}\end{bmatrix}\begin{bmatrix}\mathbf{H} & \mathbf{G}\mathbf{H} & \cdots & \mathbf{G}^{n-1}\mathbf{H}\end{bmatrix} = \begin{bmatrix}\mathbf{0} & \cdots & \mathbf{0} & \mathbf{1} \\ \mathbf{0} & \cdots & \mathbf{1} & \mathbf{p}\mathbf{G}^{n}\mathbf{H} \\ \vdots & \iddots & \iddots & \vdots \\ \mathbf{1} & \mathbf{p}\mathbf{G}^{n}\mathbf{H} & \cdots & \mathbf{p}\mathbf{G}^{2n-2}\mathbf{H}\end{bmatrix}$$

By the Cayley-Hamilton theorem.

Toeplitz Matrix

The Cayley-Hamilton theorem can further be used to show that

$$\mathbf{TM}\begin{bmatrix} a_{n-1} & \cdots & a_1 & 1 \\ a_{n-2} & \cdots & 1 & 0 \\ \vdots & \iddots & \iddots & \vdots \\ 1 & 0 & \cdots & 0 \end{bmatrix} = \mathbf{I}$$

Matrix on the right is called Toeplitz matrix

State Transformation Formulas

<u>Formula 1</u>:

$$\mathbf{T} = \begin{bmatrix} \mathbf{p} \\ \mathbf{pG} \\ \vdots \\ \mathbf{pG}^{n-1} \end{bmatrix}, \quad \mathbf{p} = \mathbf{e}_n^{\ T}\mathbf{M}^{-1}$$

<u>Formula 2</u>:

$$\mathbf{T} = \left(\mathbf{M}\begin{bmatrix} a_{n-1} & \cdots & a_1 & 1 \\ a_{n-2} & \cdots & 1 & 0 \\ \vdots & \iddots & \iddots & \vdots \\ 1 & 0 & \cdots & 0 \end{bmatrix} \right)^{-1}$$

State Feedback Control Gain Selection

$$\mathbf{u} = -\hat{\mathbf{K}}\hat{\mathbf{x}} + \mathbf{r}, \quad \hat{\mathbf{K}} = \begin{bmatrix} a_n^{des} - a_n - & \cdots & a_1^{des} - a_1 \end{bmatrix}$$

$$\mathbf{u} = -\underbrace{\hat{\mathbf{K}}\mathbf{T}}_{\mathbf{K}} x + \mathbf{r} \Rightarrow K = \begin{bmatrix} a_n^{des} - a_n - & \cdots & a_1^{des} - a_1 \end{bmatrix} \begin{bmatrix} \mathbf{p} \\ \mathbf{pG} \\ \vdots \\ \mathbf{pG}^{n-1} \end{bmatrix}$$

By Cayley Hamilton: $\mathbf{a}_n\mathbf{I} + \mathbf{a}_{n-1}\mathbf{G} + \cdots + \mathbf{a}_1\mathbf{G}^{n-1} = -\mathbf{G}^n$

$$\mathbf{K} = \mathbf{p}\left(\mathbf{G}^n + \mathbf{a}_1^{des}\mathbf{G}^{n-1} + \cdots + \mathbf{a}_{n-1}^{des}\mathbf{G} + \mathbf{a}_n^{des}\mathbf{I}\right) \quad \text{or}$$

$$\boxed{\mathbf{K} = \mathbf{e}_n{}^T\mathbf{M}^{-1}\mathbf{\Phi}^{des}(\mathbf{G})}$$

Bass-Gura Formula

$$\mathbf{u} = -\hat{\mathbf{K}}\hat{\mathbf{x}} + \mathbf{r}, \quad \hat{\mathbf{K}} = \begin{bmatrix} a_n^{des} - a_n - & \cdots & a_1^{des} - a_1 \end{bmatrix}$$

$$\mathbf{K} = \begin{bmatrix} a_n - a_n^{des} \\ a_{n-1} - a_{n-1}^{des} \\ \vdots \\ a_1 - a_1^{des} \end{bmatrix}^T \left(\mathbf{M} \begin{bmatrix} a_{n-1} & \cdots & a_1 & 1 \\ a_{n-2} & \cdots & 1 & 0 \\ \vdots & ⋰ & ⋰ & \vdots \\ 1 & 0 & \cdots & 0 \end{bmatrix} \right)^{-1}$$

$$\boxed{\mathbf{K} = \begin{bmatrix} a_1^{des} - a_1 \\ \vdots \\ a_{n-1}^{des} - a_{n-1} \\ a_n^{des} - a_n \end{bmatrix}^T \left(\mathbf{M} \begin{bmatrix} 1 & a_1 & \cdots & a_{n-1} \\ 0 & 1 & \cdots & a_{n-2} \\ \vdots & ⋱ & ⋱ & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} \right)^{-1}}$$

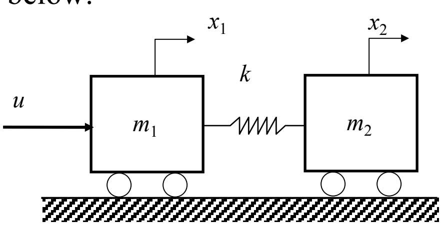
Double Integrator-Matlab Solution

```
T=0.5;
lam=[0;0];
G=[1 T;0 1];
H=[T^2/2;T];
C=[1 0];

K=acker(G,H,lam);
Gcl=G-H*K;
clsys=ss(Gcl,H,C,0,T);
step(clsys);
```


Flexible System Example

Consider the linear mass-spring system shown below:



Parameters:

$m_1=m_2=1$Kg.
K=50 N/m

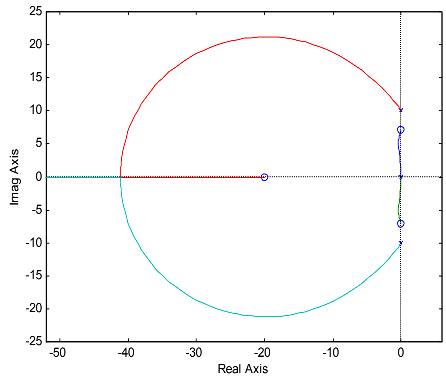
- Analyze PD controller based on a)x_1, b)x_2
- Design state feedback controller, place poles at $-20,-20,5\sqrt{2}(-1\pm j)$

Collocated Control

Transfer Function: $G_p = \dfrac{X_1(s)}{U(s)} = \dfrac{s^2+50}{s^2\left(s^2+100\right)}$

PD Control: $G_c = K\left(s+a\right), \quad a = 20$

Root-Locus



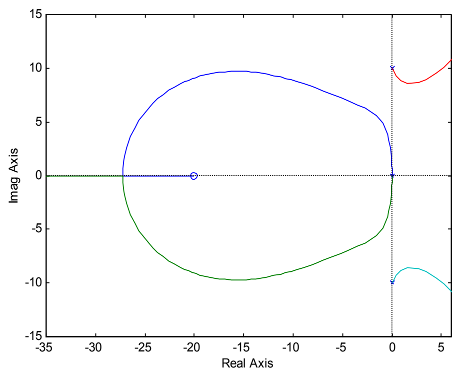
Non-Collocated Control

Transfer Function: $G_p = \dfrac{X_2(s)}{U(s)} = \dfrac{50}{s^2\left(s^2+100\right)}$

PD Control: $G_c = K\left(s+a\right), \quad a = 20$

Root-Locu

Unstable



Discrete Time State Model

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -50 & 50 & 0 & 0 \\ 50 & -50 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} u$$

<u>Discretized Model</u>: x(k+1)=Gx(k)+Hu(k)

$$G = \begin{bmatrix} 0.9975 & 0.0025 & 0.01 & 0 \\ 0.0025 & 0.9975 & 0 & 0.01 \\ -0.4992 & 0.4992 & 0.9975 & 0.0025 \\ 0.4992 & 0.4992 & 0.0025 & 0.9975 \end{bmatrix}, \quad H = \begin{bmatrix} 0 \\ 0 \\ 0.01 \\ 0 \end{bmatrix}$$

Open-Loop System Information

Controllability matrix:

$$\mathbf{M} = [\mathbf{H} \quad \mathbf{GH} \quad \mathbf{G(GH)} \quad \mathbf{G(G^2H)}]$$

$$\mathbf{M} = \begin{bmatrix} 0 & 0.0001 & 0.0002 & 0.0003 \\ 0 & 0 & 0 & 0 \\ 0.01 & 0.0099 & 0.0098 & 0.0097 \\ 0 & 0.001 & 0.0002 & 0.003 \end{bmatrix}$$

Characteristic equation:

$|zI-G|=(z-1)^2(z^2-1.99z+1)=z^4-3.99z^3+5.98z^2-3.99z+6$

State Feedback Controller

Characteristic Equations:

$|zI-G|=(z-1)^2(z^2-1.99z+1)=z^4-3.99z^3+5.98z^2-3.99z+6$

$$\Phi^{des}(s)=(z-0.8187)^2\left((z-0.9294)^2+0.0658^2\right)$$
$$\Phi^{des}(s)=z^4-3.4963z^3+4.5822z^2-2.6675z+0.5819$$

$$\mathbf{K}=\begin{bmatrix}-3.4963+3.99\\4.5822-5.98\\-2.6675+3.99\\0.5819-6\end{bmatrix}^T\left(\mathbf{M}\begin{bmatrix}1&-3.99&5.98&-3.99\\0&1&-3.99&5.98\\0&0&1&-3.99\\0&0&0&-3.99\end{bmatrix}\right)^{-1}$$

$$\boxed{\begin{array}{l}\mathbf{K}=[757.00\quad -144.17\quad 45.54\quad 105.75]\\ u=-757x_1+144.17x_2-45.54x_3-105.75x_4+r\end{array}}$$

Matlab Solution

```
%System Matrices
m1=1; m2=1; k=50; T=0.01;
syst=ss(A,B,C,D);
A=[0 0 1 0;0 0 0 1;-50 50 0 0;50 -50 0 0];
B=[0; 0; 1; 0];
C=[1 0 0 0;0 1 0 0]; D=zeros(2,1);
cplant=ss(A,B,C,D);

%Discrete-Time Plant
plant=c2d(cplant,T);
[G,H,C,D]=ssdata(plant);
```

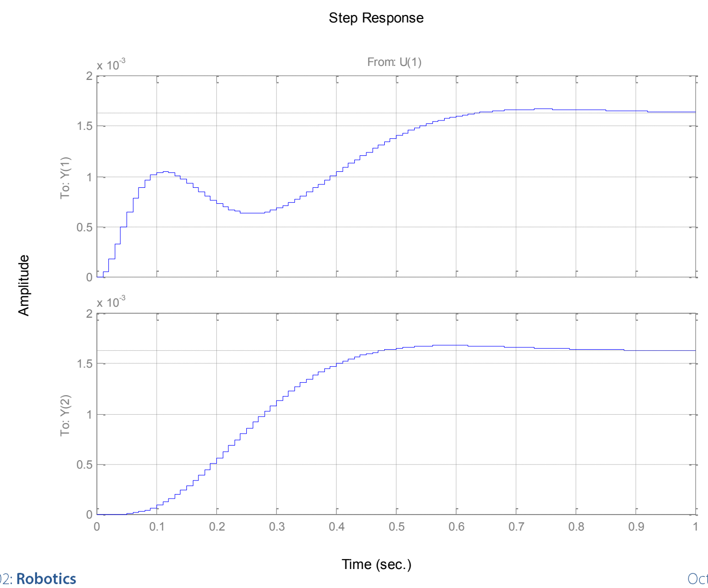
Matlab Solution

```
%Desired Close-Loop Poles
pc=[-20;-20;
    -5*sqrt(2)*(1+j); 5*sqrt(2)*(1-j)];
pd=exp(T*pc);

% State Feedback Controller
K=acker(G,H,pd);

%Closed-Loop System
clsys=ss(G-H*K,H,C,0,T);
grid
step(clsys,1)
```


Time Respone



Steady-State Gain

Closed-loop system: $x(k+1)=G_{cl}x(k)+Hr(k)$, $Y=Cx(k)$

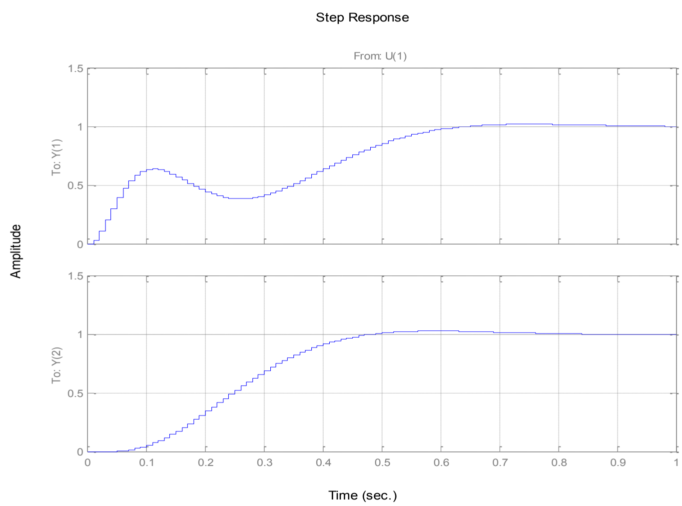
$Y(z)=C(zI-G_{cl})^{-1}H\ R(z)$

If $r(k)=r.1(k)$ then $y_{ss}=C(I-G_{cl})^{-1}H$

Thus if the desired output is constant

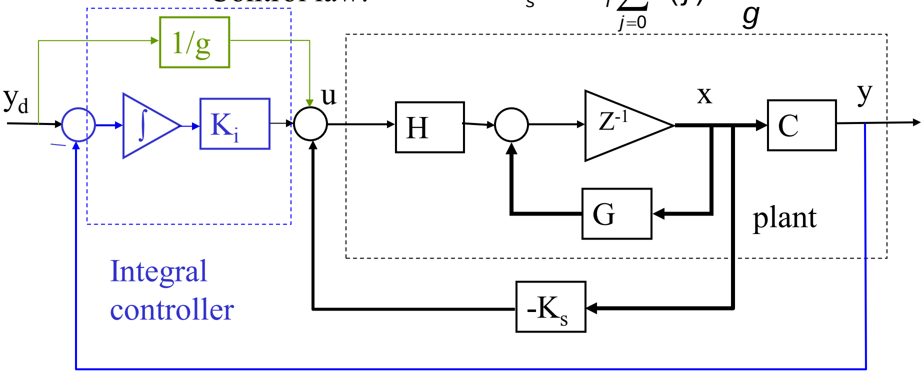
$r=y_d/\text{gain}$, $\text{gain}= C(I-G_{cl})^{-1}H$

Time Response



Integral Control

Control law: $\mathbf{u} = -\mathbf{K}_s\mathbf{x} - \mathbf{K}_I \sum_{j=0}^{k-1} e(j) + \frac{y_d}{g}$



Automatically generates reference input r!

Closed-Loop Integral Control System

Plant: $\mathbf{x}(k+1) = \mathbf{G}\mathbf{x}(k) + \mathbf{H}\mathbf{u}(k)$

$\mathbf{y}(k) = \mathbf{C}\mathbf{x}(k)$

Control: $\mathbf{u} = \mathbf{r} - \mathbf{K}_s\mathbf{x} - \mathbf{K}_I v(k),\ \mathbf{e} = \mathbf{y}_d - \mathbf{y}$

Integral state: $\mathbf{v}(k+1) = \mathbf{v}(k) - \mathbf{e}(k)$

Closed-loop system

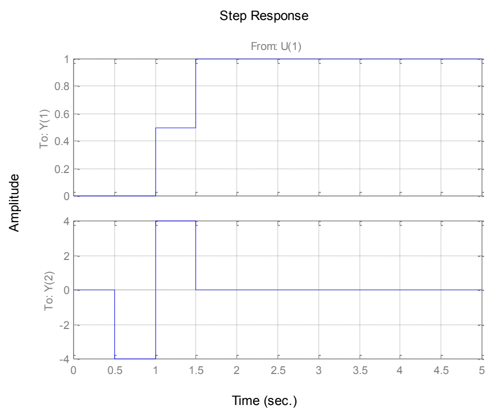
$$\begin{bmatrix} \mathbf{x}(k+1) \\ \mathbf{v}(k+1) \end{bmatrix} = \begin{bmatrix} \mathbf{G} & \mathbf{0} \\ \mathbf{C} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{x}(k) \\ \mathbf{v}(k) \end{bmatrix} - \begin{bmatrix} \mathbf{H} \\ \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{K}_s & \mathbf{K}_I \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{v} \end{bmatrix} + \begin{bmatrix} \mathbf{H}r \\ -y_d \end{bmatrix}$$

Double Integrator-Matlab Solution

```
T=0.5;
lam=[0;0;0];
G=[1 T;0 1]; H=[T^2/2;T]; C=[1 0];

Gbar=[G zeros(2,1);C 1];
Hbar=[H;0];
K=acker(Gbar,Hbar,lam);
Gcl=Gbar-Hbar*K;
yd=1; r=0; %unknown gain
clsys=ss(Gcl,[H*r;-yd],[C 0;K],0,T);
step(clsys);
```


Closed-Loop Step Response



- Lemma 18.1: Consider the state space nominal model
- Let $\bar{r}(t)$ denote an external signal.

$$\dot{x}(t) = \mathbf{A_o}x(t) + \mathbf{B_o}u(t)$$
$$y(t) = \mathbf{C_o}x(t)$$

- Then, provided that the pair (A0, B0) is completely controllable, there exists

$$u(t) = \bar{r} - \mathbf{K}x(t)$$
$$\mathbf{K} \triangleq [k_0, k_1, \ldots, k_{n-1}]$$

- such that the closed-loop characteristic polynomial is $A_{cl}(s)$, where $A_{cl}(s)$ is an arbitrary polynomial of degree n.

- Note that state feedback does not introduce additional dynamics in the loop, because the scheme is based only on proportional feedback of certain system variables. We can easily determine the overall transfer function from $\bar{r}(t)$ to y(t). It is given by

$$\frac{Y(s)}{\overline{R}(s)} = \mathbf{C_o}(s\mathbf{I} - \mathbf{A_o} + \mathbf{B_o}\mathbf{K})^{-1}\mathbf{B_o} = \frac{\mathbf{C_o}Adj\{s\mathbf{I} - \mathbf{A_o} + \mathbf{B_o}\mathbf{K}\}\mathbf{B_o}}{F(s)}$$

- where

$$F(s) \stackrel{\triangle}{=} \det\{s\mathbf{I} - \mathbf{A_o} + \mathbf{B_o}\mathbf{K}\}$$

- and Adj stands for adjoint matrices.

[Matrix inversion lemma]

- We can further simplify the expression given above. To do this, we will need to use the following results from Linear Algebra.
- (Matrix inversion lemma).
  Consider three matrices A,B,C
  Then, if A + BC is nonsingular, we have that

$$(\mathbf{A} + \mathbf{BC})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}\left(\mathbf{I} + \mathbf{C}\mathbf{A}^{-1}\mathbf{B}\right)^{-1}\mathbf{C}\mathbf{A}^{-1}$$

- In the case for which B = g ∈ ▯n and CT = h ∈ ▯n, the above result becomes

$$\left(\mathbf{A} + gh^T\right)^{-1} = \left(\mathbf{I} - \mathbf{A}^{-1}\frac{gh^T}{1 + h^T\mathbf{A}^{-1}g}\right)\mathbf{A}^{-1}$$

- Lemma 18.3: Given a matrix $W \in \mathbb{R}^{n \times n}$ and a pair of arbitrary vectors $\phi_1 \in \mathbb{R}^n$ and $\phi_2 \in \mathbb{R}^n$, then provided that W and $\qquad$ are nonsingular,

$$W + \phi_1 \phi_2^T \; ,$$

- Proof: See the book.

$$Adj(W + \phi_1 \phi_2^T)\phi_1 = Adj(W)\phi_1$$
$$\phi_2^T Adj(W + \phi_1 \phi_2^T) = \phi_2^T Adj(W)$$

LQR + Course Review

METR 4202: **Robotics** & Automation

Dr Surya Singh -- Lecture # 13

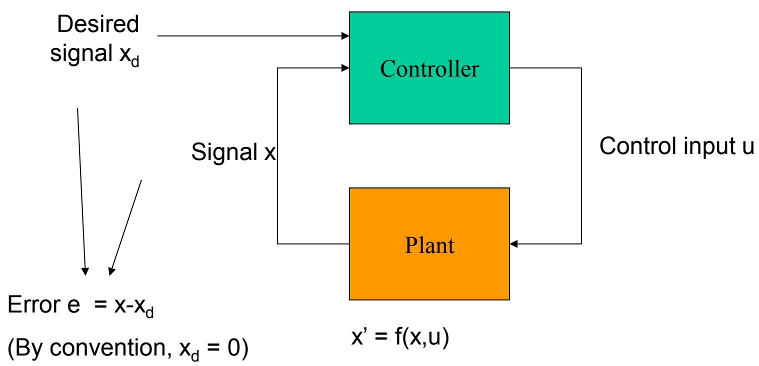
October 26, 2016

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

LQR

Control Theory

- The use of feedback to regulate a signal

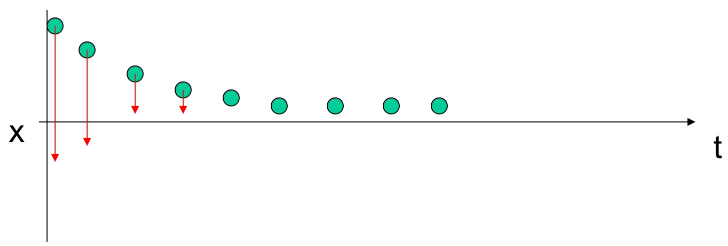


Model-free vs model-based

- 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

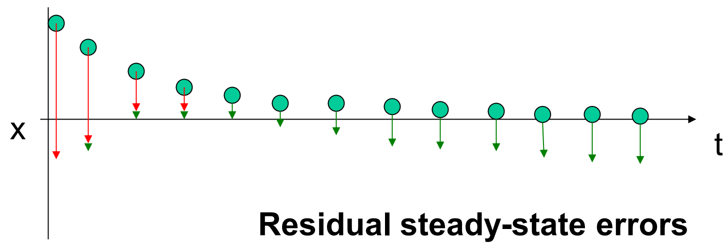
PID control

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



PID control: Integral term

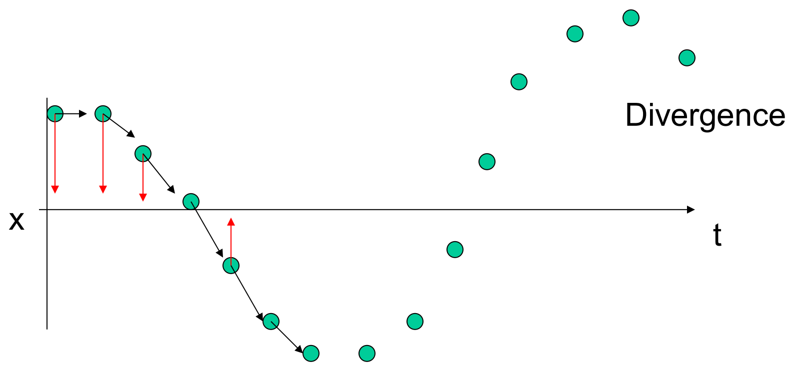
- $u(t) = -K_p\, x(t) - \underbrace{K_i}_{\text{Integral gain}} I(t)$
- $I(t) = \int_0^t x(t)dt$ (accumulation of errors)



Residual steady-state errors driven asymptotically to 0

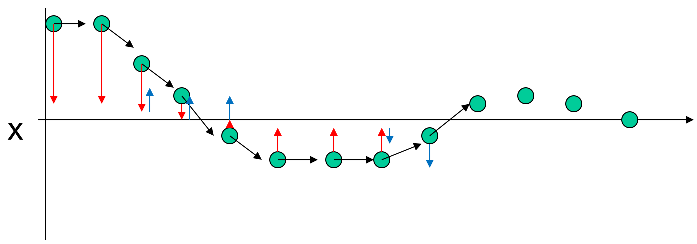
PID control: Integral term: Instability

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



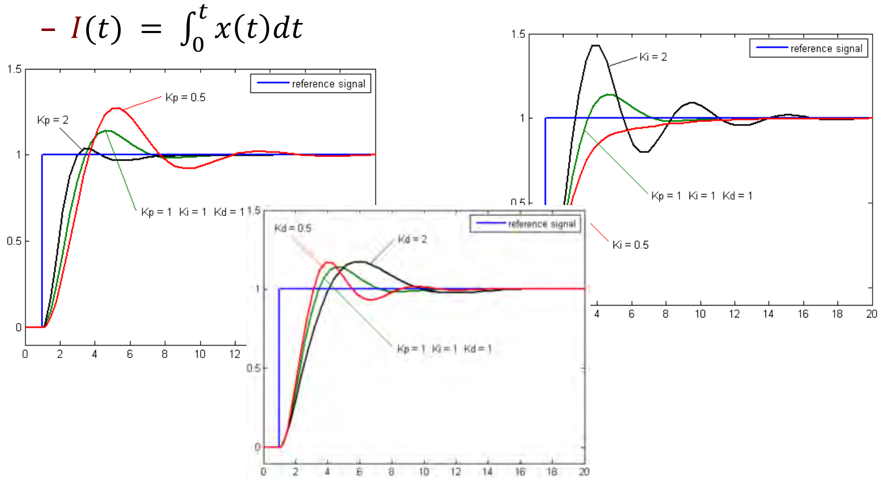
PID control: Derivative term

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



PID control: Together

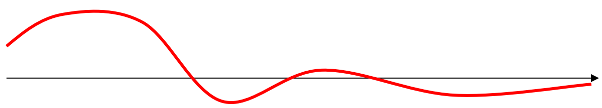
- P+I+D:
  - $u(t) = -K_p\, x(t) - K_i\, I(t) - K_d\, x'(t)$
  - $I(t) = \int_0^t x(t)dt$



Stability and Convergence

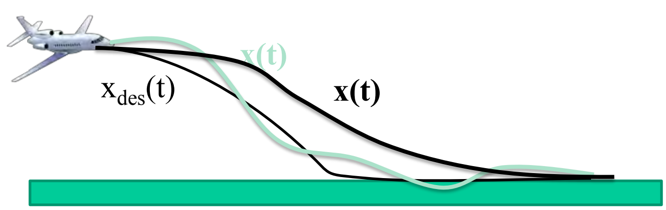
- System is *stable* if errors stay bounded

- 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

- $x' = f(x, u)$
- $x \quad X \quad Rn$
- $u \quad U \quad Rm$

- Because m ≠ n, and variables are coupled,
- This is not as easy as setting n PID controllers

Linear Quadratic Regulator

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

$$\int \boxed{x^T Q\, x} + \boxed{u^T R\, u\, dt}$$

Error term "Effort" penalization

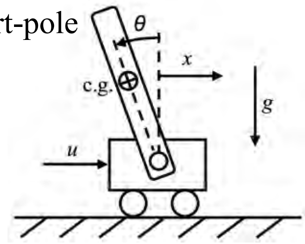
- Over an infinite horizon

Closed form LQR solution

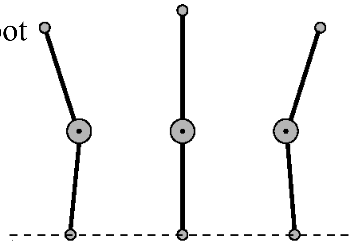
- Closed form solution
  $u = -K x$, with $K = R^{-1}BP$
- Where P is a symmetric matrix that solves the *Riccati equation*
  - $A^TP + PA - PBR^{-1}B^TP + Q = 0$
  - Derivation: calculus of variations
- Packages available for finding solution

Toy Nonlinear Systems

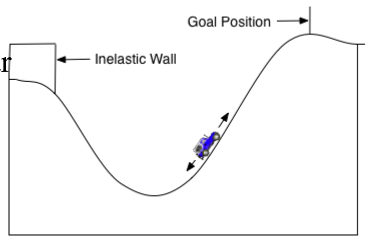
Cart-pole



Acrobot



Mountain car



Deterministic Linear Quadratic Regulation

Figure 20.1 shows the feedback configuration for the *linear quadratic regulation (LQR) problem*. The process is assumed to be a continuous-time LTI system of the form

$$\begin{aligned} \dot{x} &= Ax + Bu, & x &\in \mathbb{R}^n,\ u \in \mathbb{R}^k, \\ y &= Cx, & y &\in \mathbb{R}^m, \\ z &= Gx + Hu, & z &\in \mathbb{R}^\ell, \end{aligned}$$

and has two distinct outputs.

1. The *measured output* $y(t)$ corresponds to the signal(s) that can be measured

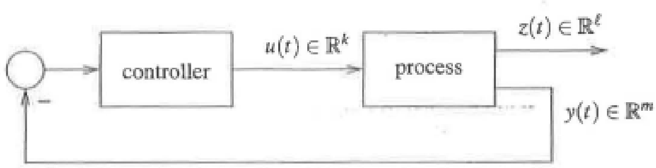


Figure 20.1. Linear quadratic regulation (LQR) feedback configuration

Deterministic Linear Quadratic Regulation

2. The *controlled output* $z(t)$ corresponds to the signal(s) that one would like to make as small as possible in the shortest possible time.

   Sometimes $z(t) = y(t)$, which means that our control objective is simply to make the measured output very small. At other times one may have

   $$z(t) = \begin{bmatrix} y(t) \\ \dot{y}(t) \end{bmatrix},$$

   which means that we want to make both the measured output $y(t)$ and its derivative $\dot{y}(t)$ very small. Many other options are possible.

Optimal Regulation

The LQR problem is defined as follows. Find the control input $u(t)$, $t \in [0, \infty)$ that makes the following criterion as small as possible:

$$J_{\text{LQR}} := \int_0^\infty \|z(t)\|^2 + \rho\,\|u(t)\|^2 dt, \qquad (20.1)$$

where ρ is a positive constant. The term

$$\int_0^\infty \|z(t)\|^2 dt$$

corresponds to the *energy of the controlled output*, and the term

$$\int_0^\infty \|u(t)\|^2 dt$$

corresponds to the *energy of the control signal*. In LQR one seeks a controller that minimizes both energies. However, decreasing the energy of the controlled output will require a large control signal, and a small control signal will lead to large controlled outputs. The role of the constant ρ is to establish a trade-off between these conflicting goals.

Optimal Regulation

1. When we chose ρ very large, the most effective way to decrease J_{LQR} is to employ a small control input, at the expense of a large controlled output.
2. When we chose ρ very small, the most effective way to decrease J_{LQR} is to obtain a very small controlled output, even if this is achieved at the expense of employing a large control input.

Often the optimal LQR problem is defined more generally and consists of finding the control input that minimizes

$$J_{\text{LQR}} := \int_0^\infty z(t)'\bar{Q}z(t) + \rho\, u(t)'\bar{R}u(t)\, dt, \tag{20.2}$$

Optimal Regulation

where $\bar{Q} \in \mathbb{R}^{\ell \times \ell}$ and $\bar{R} \in \mathbb{R}^{m \times m}$ are symmetric positive-definite matrices and ρ is a positive constant.

We shall consider the most general form for a quadratic criterion, which is

$$J_{\text{LQR}} := \int_0^\infty x(t)'Qx(t) + u(t)'Ru(t) + 2x(t)'Nu(t)dt. \tag{J-LQR}$$

Since $z = Gx + Hu$, the criterion in (20.1) is a special form of the criterion (J-LQR) with

$$Q = G'G, \qquad R = H'H + \rho I, \qquad N = G'H$$

and (20.2) is a special form of the criterion (J-LQR) with

$$Q = G'\bar{Q}G, \qquad R = H'\bar{Q}H + \rho\bar{R}, \qquad N = G'\bar{Q}H.$$

Optimal State Feedback

It turns out that the LQR criterion

$$J_{\text{LQR}} := \int_0^\infty x(t)'Qx(t) + u(t)'Ru(t) + 2x(t)'Nu(t)dt \qquad \text{(J-LQR)}$$

can be expressed as in (20.3) for an appropriate choice of feedback invariant. In fact, the feedback invariant in Proposition 20.1 will work, provided that we choose the matrix P appropriately. To check that this is so, we add and subtract this feedback invariant to the LQR criterion and conclude that

$$\begin{aligned}J_{\text{LQR}} &:= \int_0^\infty x'Qx + u'Ru + 2x'Nu\ dt\\ &= H\big(x(\cdot); u(\cdot)\big)\\ &\quad + \int_0^\infty x'Qx + u'Ru + 2x'Nu + (Ax+Bu)'Px + x'P(Ax+Bu)\ dt\\ &= H\big(x(\cdot); u(\cdot)\big) + \int_0^\infty x'(A'P + PA + Q)x + u'Ru + 2u'(B'P + N')x\ dt.\end{aligned}$$

Optimal State Feedback

By completing the square, we can group the quadratic term in u with the cross-term in u times x:

$$\begin{aligned}&(u' + x'K')R(u + Kx)\\ &\qquad = u'Ru + x'(PB + N)R^{-1}(B'P + N')x + 2u'(B'P + N')x,\end{aligned}$$

where

$$K := R^{-1}(B'P + N'),$$

from which we conclude that

$$\begin{aligned}J_{\text{LQR}} = H\big(x(\cdot); u(\cdot)\big) + \int_0^\infty & x'\big(A'P + PA + Q - (PB + N)R^{-1}(B'P + N')\big)x\\ &+ (u' + x'K')R(u + Kx)\ dt.\end{aligned}$$

Optimal State Feedback

If we are able to select the matrix P so that

$$A'P + PA + Q - (PB + N)R^{-1}(B'P + N') = 0, \qquad (20.5)$$

we obtain precisely an expression such as (20.3) with

$$\Lambda(x, u) := (u' + x'K')R(u + Kx),$$

which has a minimum equal to zero for

$$u = -Kx, \qquad K := R^{-1}(B'P + N'),$$

leading to the closed-loop system

$$\dot{x} = \big(A - BR^{-1}(B'P + N')\big)x.$$

The following has been proved.

Theorem 20.1. *Assume that there exists a symmetric solution* P *to the* algebraic Riccati equation (20.5) *for which* $A - BR^{-1}(B'P + N')$ *is a stability matrix. Then the feedback law*

$$u(t) := -Kx(t), \quad \forall t \geq 0, \qquad K := R^{-1}(B'P + N') \qquad (20.6)$$

minimizes the LQR criterion (J-LQR) *and leads to*

$$J_{\text{LQR}} := \int_0^\infty x'Qx + u'Ru + 2x'Nu\, dt = x'(0)Px(0). \qquad \square$$

LQR In MATLAB

MATLAB® Hint 42 (`lqr`). The command `[K,P,E]=lqr(A,B,Q,R,N)` solves the algebraic Riccati equation

$$\mathrm{A'P + PA + Q - (PB + N)}R^{-1}\mathrm{(B'P + N')} = 0$$

and computes the (negative feedback) optimal state feedback matrix gain

$$\mathrm{K = R^{-1}(B'P + N')}$$

that minimizes the LQR criteria

$$J := \int_0^\infty x'\mathrm{Q}x + u'\mathrm{R}u + 2x'\mathrm{N}u\, dt$$

for the continuous-time process

$$\dot{x} = \mathrm{A}x + \mathrm{B}u.$$

This command also returns the poles E of the closed-loop system

$$\dot{x} = (\mathrm{A - BK})x. \qquad \square$$

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

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

  - 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 2nd 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

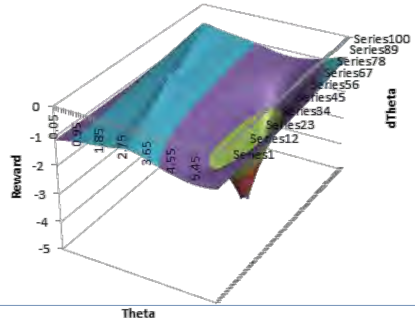
Direct policy synthesis: Optimal control

- *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

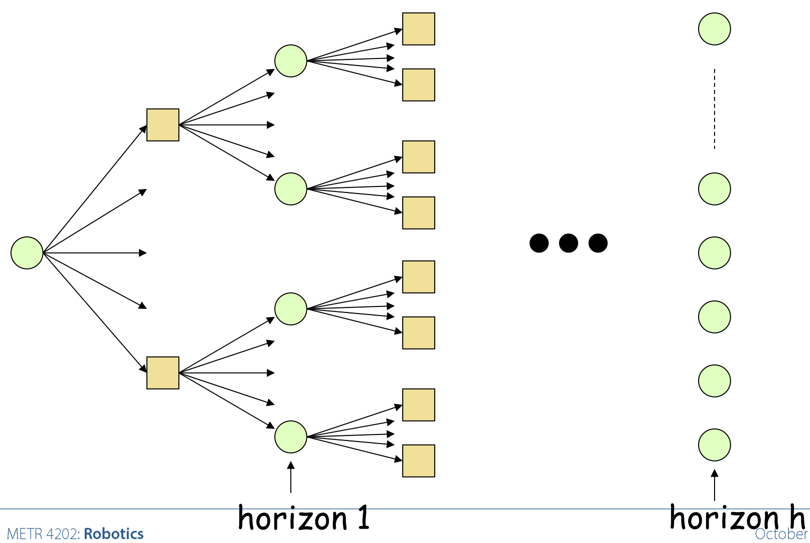
Discrete Search example

- 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)



Estimation

Along multiple dimensions

State Space

- We collect our set of uncertain variables into a vector …
  $\mathbf{x} = [x_1, x_2, \ldots, x_N]^T$
- The set of values that **x** might take on is termed the *state space*
- There is a *single* true value for **x**,
  but it is unknown

State Space Dynamics

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$$
$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u}$$

$$H(s) = C(sI - A)^{-1}B$$

Measured versus True

- Measurement errors are inevitable
- So, add Noise to State...
  - State Dynamics be
$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} + w$$
$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u} + v$$
- Can represent this as a "Normal" Distribution

$$\mathcal{N}(x;\mu,\sigma) = \frac{1}{\left(\sqrt{2\pi}\right)\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

Recovering The Truth

- Numerous methods
- Termed "Estimation" because we are trying to estimate the truth from the signal
- A strategy discovered by Gauss
- Least Squares in Matrix Representation

$$\begin{bmatrix} p_0 \\ p_1 \end{bmatrix} = \begin{bmatrix} n & \sum_1^n t_i \\ \sum_1^n t_i & \sum_1^n t_i^2 \end{bmatrix}^{-1} \begin{bmatrix} \sum_1^n z_i \\ \sum_1^n t_i z_i \end{bmatrix}$$

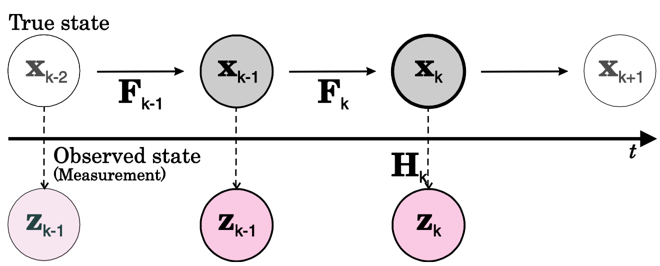
Recovering the Truth: Terminology

$$\dot{\mathbf{x}} = \mathbf{F}\mathbf{x} + \mathbf{G}\mathbf{u} + \mathbf{w}$$

$$\mathbf{z} = \mathbf{H}\mathbf{x} + \mathbf{v}$$

$\mathbf{x}$: the state vector
$\mathbf{x}_{A|B}$: the state of $\mathbf{x}$ at time A based on data taken up to time B
$\hat{\mathbf{x}}$: estimate of the true state vector
$\mathbf{F}$: system dynamics matrix in continuous time (equivalent to $\mathbf{A}$ in Eq. 1)
$\mathbf{G}$: system control matrix relating deterministic input, $\mathbf{u}$, to the state (equivalent to $\mathbf{B}$ in Eq. 1)
$\mathbf{H}$: measurement matrix in continuous time (equivalent to $\mathbf{C}$ in Eq. 2)
$\mathbf{F}_i$: system model in **discrete** time at $t = t_i$
$\mathbf{H}_i$: measurement model in **discrete** time at $t = t_i$
$\mathbf{P}_i$: estimate covariance in **discrete** time at $t = t_i$
$\mathbf{w}$: process uncertainty (noise) vector (of type $\mathcal{N}(0, s)$)
$\mathbf{Q}$: process noise matrix, $\mathbf{Q} = E\left[ww^{\mathsf{T}}\right]$
$\mathbf{Q}_i$: $\mathbf{Q}$ in discrete time at $t = t_i$
$\mathbf{v}$: measurement noise vectors (of type $\mathcal{N}(0, \sigma)$)
$\mathbf{R}_i$: the measurement variance matrix, $\mathbf{R} = E\left[vv^{\mathsf{T}}\right]$, in discrete time at $t = t_i$

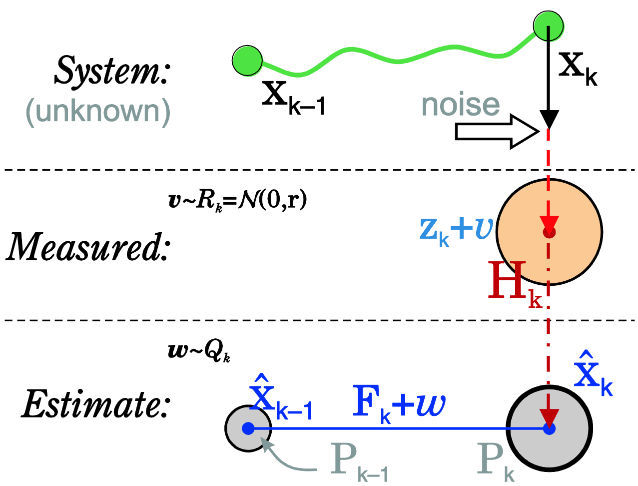
General Problem…



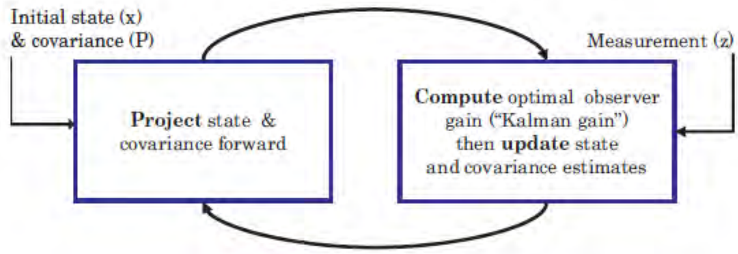
Duals and Dual Terminology

	Estimation		Control
Model:	$\dot{\mathbf{x}} = \mathbf{F}\mathbf{x}$ (discrete: $\mathbf{x} = \mathbf{F}_k\mathbf{x}$)	$\leftrightarrow$	$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x}$, $\mathbf{A} = \mathbf{F}^\top$
Regulates:	P (covariance)	$\leftrightarrow$	M (performance matrix)
Minimized function:	Q (or $GQG^\top$)	$\leftrightarrow$	V
Optimal Gain:	K	$\leftrightarrow$	G
Completeness law:	Observability	$\leftrightarrow$	Controllability

Estimation Process in Pictures



Kalman Filter Process



KF Process in Equations

Prediction: $\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}_{k-1}\hat{\mathbf{x}}_{k-1|k-1},$ (state prediction)

$\mathbf{P}_{k|k-1} = \mathbf{Q}_{k-1} + \mathbf{F}_{k-1}\mathbf{P}_{k-1|k-1}\mathbf{F}^{\mathsf{T}}{}_{k-1},$ (covariance prediction)

Kalman Gain: $\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{H}^{\mathsf{T}}[\mathbf{H}\mathbf{P}_{k|k-1}\mathbf{H}^{\mathsf{T}} + \mathbf{R}_k]^{-1},$

Update: $\mathbf{P}_{k|k} = [\mathbf{I} - \mathbf{K}_k\mathbf{H}]\mathbf{P}_{k|k-1},$ (covariance update)

$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_{k|k-1})$ (state update)

KF Considerations

$$\underbrace{\hat{\mathbf{x}}_{k|k-1}}_{n\times 1} = \underbrace{\mathbf{F}_{k-1}}_{n\times n}\hat{\mathbf{x}}_{k-1|k-1} + \underbrace{\mathbf{G}_{k-1}}_{n\times j}\underbrace{\mathbf{u}_{k-1}}_{j\times 1}$$

$$\underbrace{\mathbf{P}_{k|k-1}}_{n\times n} = \underbrace{\mathbf{Q}_{k-1}}_{n\times n} + \mathbf{F}_{k-1}\mathbf{P}_{k-1|k-1}\mathbf{F}^{\mathsf{T}}{}_{k-1}$$

$$\underbrace{\mathbf{K}_k}_{n\times m} = \mathbf{P}_{k|k-1}\underbrace{\mathbf{H}^{\mathsf{T}}}_{n\times m}\underbrace{\left[\mathbf{H}\mathbf{P}_{k|k-1}\mathbf{H}^{\mathsf{T}} + \mathbf{R}_k\right]^{-1}}_{m\times m}$$

$$\mathbf{P}_{k|k} = [\mathbf{I} - \mathbf{K}_k\mathbf{H}]\mathbf{P}_{k|k-1}$$

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k(\underbrace{\mathbf{z}_k}_{m\times 1} - \underbrace{\mathbf{H}}_{m\times n}\hat{\mathbf{x}}_{k|k-1} - \mathbf{H}\mathbf{G}_k\mathbf{u}_{k-1})$$

Ex: Kinematic KF: Tracking

- Consider a System with Constant Acceleration

$$\ddot{y} = -g$$
$$\dot{y} = gt + p_1$$
$$y = p_0 + p_1 t + \frac{gt^2}{2}$$

$$\begin{bmatrix} \dot{y} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} y \\ \dot{y} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & -1 \end{bmatrix}\begin{bmatrix} 0 \\ g \end{bmatrix}$$

$$\mathbf{F} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \mathbf{F_k} = \begin{bmatrix} 0 & t_s & \frac{t_s{}^2}{2} \\ 0 & 0 & t_s \\ 0 & 0 & 0 \end{bmatrix}$$

$$\hat{\mathbf{x}}_k = \mathbf{F}_{k-1}\hat{\mathbf{x}}_{k-1} + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}\mathbf{F}_{k-1}\hat{\mathbf{x}}_{k-1})$$

In Summary

- KF:
  - The true state (x) is separate from the measured (z)
  - Lets you **combine** prior controls knowledge with measurements to filter signals and find the truth
  - It **regulates** the covariance (P)
    - As P is the scatter between z and x
    - So, if P → 0, then $z \rightarrow x$ (measurements → truth)
- EKF:
  - Takes a Taylor series approximation to get a local "F" (and "G" and "H")