Chapter 3
Numerical Methods

Part 3

3.4 Differential Algebraic Systems

3.5 Integration of Differential Equations

Outline

- 3.4 Differential Algebraic Systems
  - 3.4.1 Constrained Dynamics
  - 3.4.2 First and Second Order Constrained Kinematic Systems
  - 3.4.3 Lagrangian Dynamics
  - 3.4.4 Constraints
  - Summary
- 3.5 Integration of Differential Equations

Outline

- 3.4 Differential Algebraic Systems
  - 3.4.1 Constrained Dynamics
  - 3.4.2 First and Second Order Constrained Kinematic Systems
  - 3.4.3 Lagrangian Dynamics
  - 3.4.4 Constraints
  - Summary
- 3.5 Integration of Differential Equations

3.4.1.1. Augmented Systems

- Consider a differential equation with n states subject to m constraints:

$$\dot{\underline{x}} = f(\underline{x}, \underline{u})$$
$$\underline{c}(\underline{x}) = \underline{0}$$

- Linear equations could be substituted into the DE. Nonlinear is the case that matters to us.
- What does it mean? Both equations cannot be correct …
  - It means the DE applies in the subspace of $\mathcal{R}^n$ that satisfies the constraints.
  - The subspace is known as the constraint manifold.

Carnegie Mellon THE ROBOTICS INSTITUTE
4

3.4.1.3 Sequential Approach

- First Approach: Integrate the unconstrained DE one time step.

$$\underline{x}_{k+1} = \underline{x}_k + \Delta\underline{x}_k = \underline{x}_k + f(\underline{x}, \underline{u})\Delta t$$

- Use result as initial conditions for a rootfinding problem that enforces constraints $\underline{c}(\underline{x}) = \underline{0}$
- Should work but …
  - What if rootfinding step reverses the DE step?
  - Did it move by Δt or $< \Delta t$ or $> \Delta t$?
  - The two equations can disagree with each other. They need to be decoupled.
    - Idea: Make the DE step satisfy the constraints to first order.

3.4.1.4 Projection Approach

- Second Approach: Remove the component of the state derivative out of the constraint tangent plane.
  - Equivalently, project it into the tangent plane.
- Write step in terms of feasible and infeasible component:

$$\Delta \underline{x} = \Delta \underline{x}_{\perp} + \Delta \underline{x}_{||}$$

- Remove the component out of the tangent plane:

$$\boxed{\Delta \underline{x}_{\perp} = \underline{c}_{\underline{x}}^{T}[\underline{c}_{\underline{x}}\underline{c}_{\underline{x}}^{T}]^{-1}\underline{c}_{\underline{x}}\Delta \underline{x}}$$

- The matrix $P_C(M) = M[M^T M]^{-1} M^T$ performs a projection on the column space of M.
  - Here we project onto colspace of $\underline{c}_{\underline{x}}^{T}$ - which is the rowspace of $\underline{c}_{\underline{x}}$ (the Constraint Jacobian).
  - So this is the component that violates the constraints to first order.

Outline

- 3.4 Differential Algebraic Systems
  - 3.4.1 Constrained Dynamics
  - 3.4.2 First and Second Order Constrained Kinematic Systems
  - 3.4.3 Lagrangian Dynamics
  - 3.4.4 Constraints
  - Summary
- 3.5 Integration of Differential Equations

Carnegie Mellon
THE ROBOTICS INSTITUTE

3.4.2.1 Augmented First Order Systems

- Third approach: Remove the infeasible component right in the differential equation.
- For a feasible perturbation: $\underline{c}_{\underline{x}} \Delta \underline{x} = \underline{0}$
- The infeasible part is some unknown combination of the constraint gradients. Let it be of the form: $\Delta \underline{x}_{\perp} = \underline{c}_{\underline{x}}^T \underline{\lambda} \Delta t$
- Remove the infeasible component with:

$$\Delta \underline{x}_{\parallel} = f(\underline{x}, \underline{u}) \Delta t - \Delta \underline{x}_{\perp} = f(\underline{x}, \underline{u}) \Delta t - \underline{c}_{\underline{x}}^T \underline{\lambda} \Delta t$$

- Divide both equations by Δt and pass to the limit:

$$\dot{\underline{x}} + \underline{c}_{\underline{x}}^T \underline{\lambda} = f(\underline{x}, \underline{u})$$
$$\underline{c}_{\underline{x}} \dot{\underline{x}} = \underline{0}$$

3.4.2.2 Solving The Eqns of Motion

- This can be written in matrix form:

$$\dot{\underline{x}} + \underline{c}_{\underline{x}}^T \underline{\lambda} = \underline{f}(\underline{x}, \underline{u})$$
$$\underline{c}_{\underline{x}} \dot{\underline{x}} = \underline{0}$$

$$\boxed{\begin{bmatrix} I & \underline{c}_{\underline{x}}^T \\ \underline{c}_{\underline{x}} & 0 \end{bmatrix} \begin{bmatrix} \dot{\underline{x}} \\ \underline{\lambda} \end{bmatrix} = \begin{bmatrix} \underline{f}(\underline{x}, \underline{u}) \\ \underline{0} \end{bmatrix}}$$

We will see this Again in Lagrangian Dynamics

- To solve, multiply 1st by $\underline{c}_{\underline{x}}$:

$$\underline{c}_{\underline{x}} \dot{\underline{x}} + \underline{c}_{\underline{x}} \underline{c}_{\underline{x}}^T \underline{\lambda} = \underline{c}_{\underline{x}} \underline{f}(\underline{x}, \underline{u})$$

- By 2nd equation $\underline{c}_{\underline{x}} \dot{\underline{x}} = 0$ so solve for λ:

$$\underline{\lambda} = (\underline{c}_{\underline{x}} \underline{c}_{\underline{x}}^T)^{-1} \underline{c}_{\underline{x}} \underline{f}(\underline{x}, \underline{u})$$

Right Pseudo Inverse

3.4.2.2 Solving The Eqns of Motion

- Substitute for λ in first equation:

$$\boxed{\dot{\underline{x}} = [I - \underline{c}_{\underline{x}}^T(\underline{c}_{\underline{x}}\underline{c}_{\underline{x}}^T)^{-1}\underline{c}_{\underline{x}}]f(\underline{x}, \underline{u})}$$

- The matrix:

$$P_N(\underline{c}_{\underline{x}}^T) = I - P_C(\underline{c}_{\underline{x}}^T) = I - \underline{c}_{\underline{x}}^T(\underline{c}_{\underline{x}}\underline{c}_{\underline{x}}^T)^{-1}\underline{c}_{\underline{x}}$$

- Projects the state derivative directly into the nullspace of the constraints – i.e. directly into the tangent plane.
  - By simply removing the component normal to the tangent plane (i.e. a weighted sum of the gradients).

3.4.2.3 Holonomic Constraints

- These are of the form:

$$\underline{c}(\underline{x}) = \underline{0}$$

- It is useful to differentiate constraints sometimes.
- Differentiating wrt time gives our standard form for a velocity constraint:

$$\dot{\underline{c}}(\underline{x}) = \underline{c}_{\underline{x}}\dot{\underline{x}} = \underline{0}$$

- If the DE has a holonomic constraint on $\underline{x}$, this implies that the derivative ($\dot{\underline{x}}$,) must be constrained too:
  - In fact it must be orthogonal to the constraint gradient.

3.4.2.4 Nonholonomic Constraints

- Consider a form that depends on the state derivative:

$$\underline{c}(\underline{x}, \dot{\underline{x}}) = \underline{0}$$

- Differentiate to get:

$$\dot{\underline{c}}(\underline{x}, \dot{\underline{x}}) = \underline{c}_{\underline{x}} \dot{\underline{x}} + \underline{c}_{\dot{\underline{x}}} \ddot{\underline{x}} = \underline{0}$$

- In general, all higher state derivatives are constrained too.
- A special form that is relevant to us is:

$$\underline{c}(\underline{x}, \dot{\underline{x}}) = \underline{w}(\underline{x})\dot{\underline{x}} = \underline{0}$$

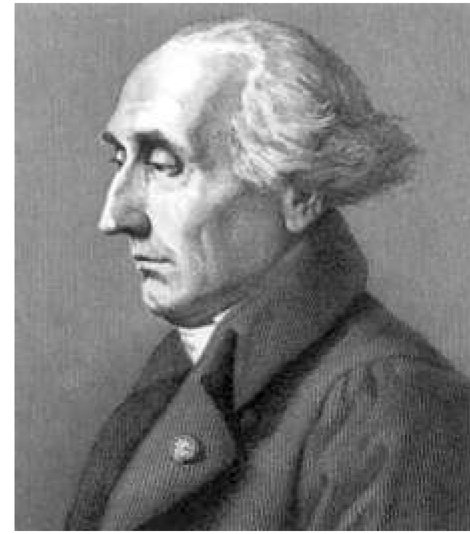
- This does not need to be differentiated to use.

Outline

- 3.4 Differential Algebraic Systems
  - 3.4.1 Constrained Dynamics
  - 3.4.2 First and Second Order Constrained Kinematic Systems
  - 3.4.3 Lagrangian Dynamics
  - 3.4.4 Constraints
  - Summary
- 3.5 Integration of Differential Equations

Compte Joseph-Louis Lagrange

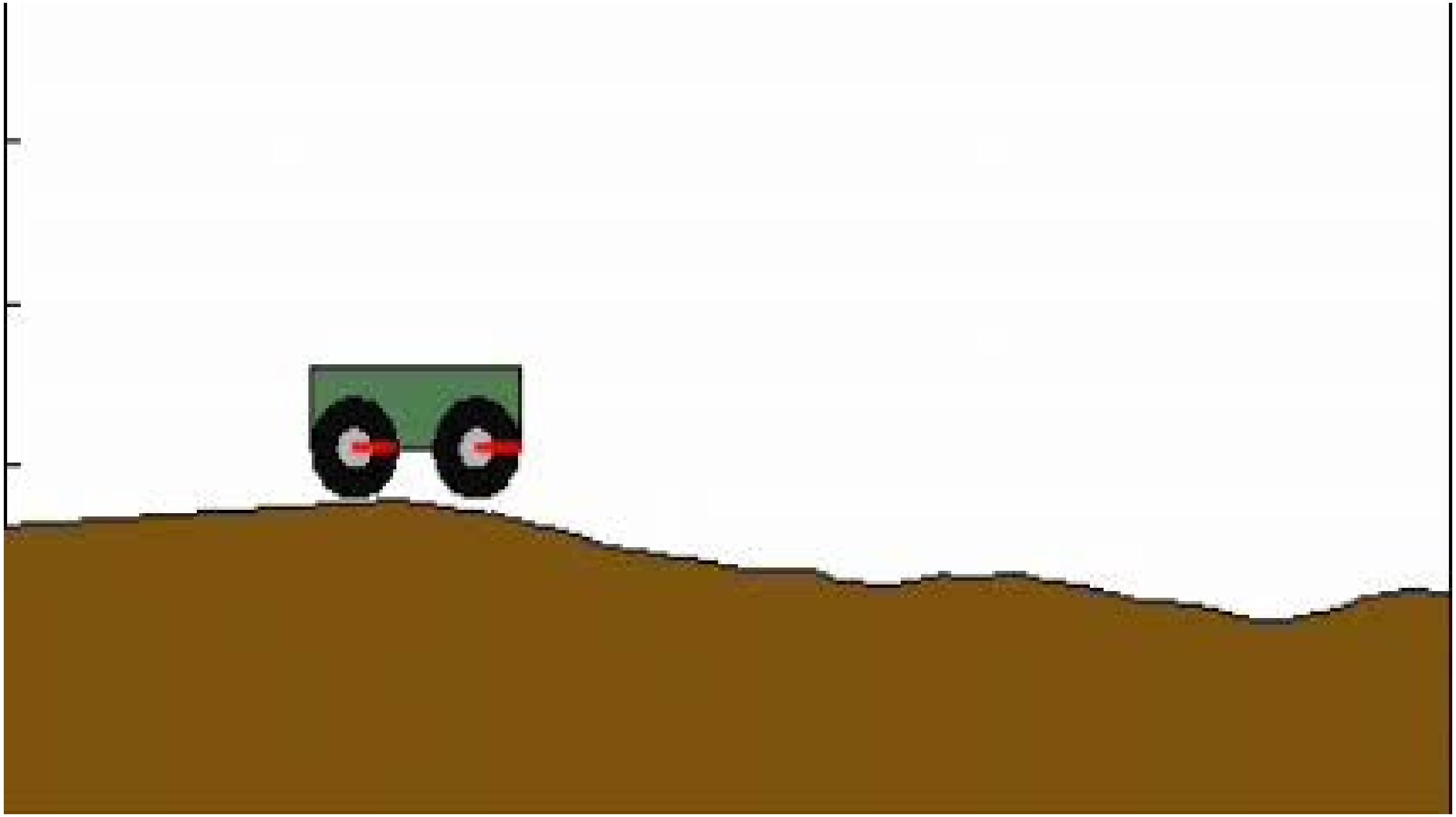
- Greatest mathematician of 18 century?
- Advised by Leonhard Euler (who was advised by Bernoulli)
- Notable students
  - Joseph Fourier
  - Simeon Poisson
- Reformulated Newtonian Mechanics Mécanique Analytique (Analytical Mechanics) (1788).
- Invented:
  - Theory of Differential Equations
  - Calculus of variations



1736-1813
Italian-French

Carnegie Mellon THE ROBOTICS INSTITUTE

Vehicle on Terrain Video



Carnegie Mellon THE ROBOTICS INSTITUTE

3.4.3.1 Equations of Motion

- Embedded Form:
  - Eliminates constraint forces
  - Fewest dof, fewest equations
  - Nonlinear, complex equations
  - Popular for manipulators
- Augmented Form:
  - Redundant coordinates
  - Explicit constraint forces
  - Simpler equations
  - Suitable for automation

Real wheels do slip. Hence nonholonomic “constraints” are not really “no slip” constraints, they are “slip like this” constraints.

Carnegie Mellon
THE ROBOTICS INSTITUTE

3.4.3.1 Equations of Motion – One Body

- Equations of motion are simple when the generalized coordinates (q) are absolute (inertial).
- Coordinates for one body:

$$\underline{x}_i = \begin{bmatrix} x_i & y_i & \theta_i \end{bmatrix}^T$$

- Equation of motion:

Unknowns

Eqn A

$$M_i \ddot{\underline{x}}_i = \underline{F}_i^{ext} + \underline{F}_i^{con}$$

mass

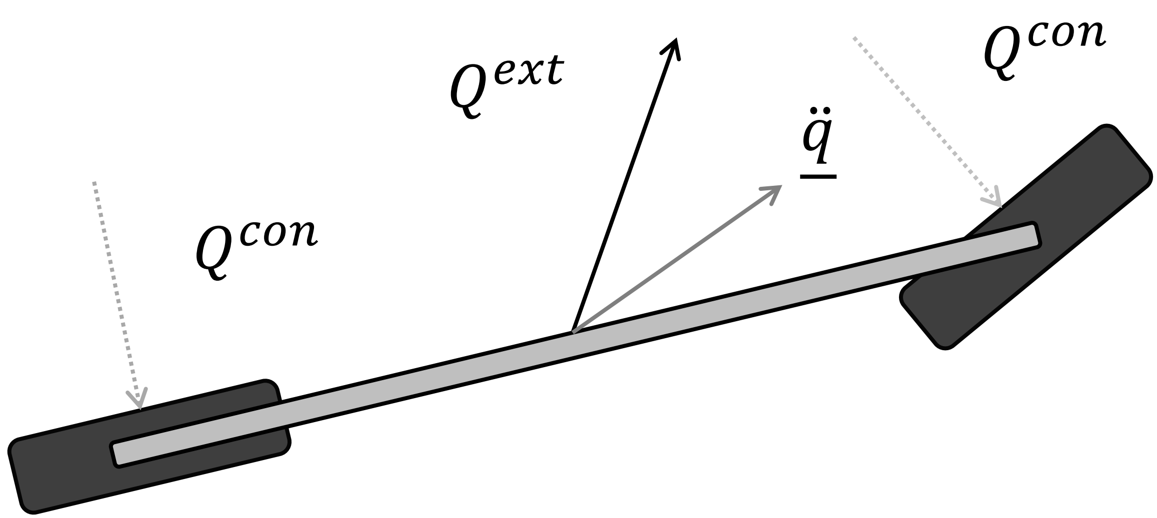
external force

constraint force

Segment forces into those known and those unknown

3.4.3.1 Applied and Constraint Forces

- Constraint forces are generated to oppose motion in the disallowed directions.
- The net force (parallel to acceleration) is therefore not in the direction of the applied force.



Carnegie Mellon THE ROBOTICS INSTITUTE

Center of Mass Reference

- Choose Center of mass as the body reference point. Then:

2x2 identity

$$M_i = \begin{bmatrix} m_i I & 0 \\ 0 & J_i \end{bmatrix}$$

Polar moment of inertia

Carnegie Mellon
THE ROBOTICS INSTITUTE

Underdetermined System – n bodies

- # of equations:
  - 3 n: one for each element of $\underline{\ddot{x}}$
- # of unknowns:
  - 3 n generalized accelerations $\underline{\ddot{x}}$
  - 3 c constraint forces
  - 3 n generalized velocities $\underline{\dot{x}}$
  - 3 n generalized coordinates $\underline{x}$
- Where do the other 3c "constraints" come from?
  - The constraints ☺
- Where do the velocities and positions come from?
  - Integration

3.4.3.2 Differentiated Constraints - Holonomic

- The 2nd derivative of a holonomic constraint is:

$$\ddot{\underline{c}}(\underline{x}) = \underline{c}_{\underline{x}t}\,\dot{\underline{x}} + \underline{c}_{\underline{x}}\,\ddot{\underline{x}} = \underline{0}$$

- Define:

$$\boxed{\underline{F}_d = -\,\underline{c}_{\underline{x}t}\,\dot{\underline{x}}}$$

- Then we have:

$$\underline{c}_{\underline{x}}\,\ddot{\underline{x}} = \underline{F}_d$$

- This makes the differentiated constraint look like Newton's 2nd law.

Carnegie Mellon THE ROBOTICS INSTITUTE

3.4.3.2 Differentiated Constraints-Nonholonomic

- Also, for a nonholonomic constraint:

$$\dot{\underline{c}}(\underline{x}, \dot{\underline{x}}) = \underline{c}_{\underline{x}}\, \dot{\underline{x}} + \underline{c}_{\dot{\underline{x}}}\, \ddot{\underline{x}} = \underline{0}$$

- Define:

$$\boxed{\underline{F}_d = -\underline{c}_{\underline{x}}\, \dot{\underline{x}}}$$

- Then the constraint becomes:

$$\underline{c}_{\dot{\underline{x}}}\, \ddot{\underline{x}} = \underline{F}_d$$

- Once again, this makes the differentiated constraint look like Newton's 2nd law.

3.4.3.2 Differentiated Constraints-General

- Both earlier forms are of the form:

Eqn B

$$\boxed{C\,\ddot{\underline{x}} = \underline{F}_d}$$

$$\underline{c}_{\underline{x}}\,\ddot{\underline{x}} = \underline{F}_d$$

holonomic

$$\underline{c}_{\dot{\underline{x}}}\,\ddot{\underline{x}} = \underline{F}_d$$

nonholonomic

3.4.3.3 Principle of Virtual Work

- Credited to Aristotle(!) and/or Bernoulli.
- Work: The product of a force and a displacement in the direction of the force.
- Virtual Work: As above but either force or displacement is not real.

3.4.3.3 Lagrange Multipliers

- We will require the virtual work performed by constraint forces to vanish.
- This is accomplished by writing:

Constraint Jacobian

$$\underline{F}_i^{con} = -C^T \underline{\lambda}$$

Eqn C

- Why?
  - Columns of C^T (rows of C) are prohibited directions.
  - Constraint forces are confined above to those prohibited directions.
  - Dot product of constraint forces with any feasible displacement will be zero. Displacements parallel to the the rows of J are infeasible.

Carnegie Mellon
THE ROBOTICS INSTITUTE

3.4.3.4 Augmented System

- Recall the original equations of motion:

$$M_i \underline{\ddot{x}}_i = \underline{F}_i^{ext} + \underline{F}_i^{con}$$ Eqn A

- Substitute from Eqn C:

$$M\underline{\ddot{x}} + C^T\underline{\lambda} = \underline{F}^{ext}$$ Eqn A1

Forces in Infeasible Directions

- Combine this with Eqn B:

$$C\,\underline{\ddot{x}} = \underline{F}_d$$ Eqn B

Feasible Accelerations

Augmented System

- We now have c extra equations and have replaced the constraint forces with the Lagrange Multipliers as unknowns.

$$\begin{bmatrix} M & C^T \\ C & 0 \end{bmatrix} \begin{bmatrix} \ddot{\underline{x}} \\ \underline{\lambda} \end{bmatrix} = \begin{bmatrix} \underline{F}^{ext} \\ \underline{F}_d \end{bmatrix}$$

Augmented Mass Matrix

Augmented Force Vector

unknowns

- Solve and then integrate acceleration twice.

Outline

- 3.4 Differential Algebraic Systems
  - 3.4.1 Constrained Dynamics
  - 3.4.2 First and Second Order Constrained Kinematic Systems
  - 3.4.3 Lagrangian Dynamics
  - <u>3.4.4 Constraints</u>
  - Summary
- 3.5 Integration of Differential Equations

Carnegie Mellon THE ROBOTICS INSTITUTE

3.4.4.1 Constraint Trim

- Due to inevitable numerical error, enforcing differentiated constraints does not enforce the original constraints.
- Lt f(t) be the state derivative computed by integrating the second order system:

$$\underline{f}(t) = \underline{f}(0) + \int_0^t \ddot{\underline{x}}(t)dt$$

- The first order system is subject to the original constraints:

$$\dot{\underline{x}} = \underline{f}(t)$$
$$C\dot{\underline{x}} = \underline{0}$$

- $\dot{\underline{x}}$ generated by integration will likely not satisfy these constraints, so fix it with the following before integration:

$$\dot{\underline{x}} = [I - C^T(CC^T)^{-1}C]\underline{f}(t)$$

Carnegie Mellon
THE ROBOTICS INSTITUTE

3.4.4.2 Drift Control

- Constraints will drift over time since only derivatives are enforced.
- Elegant solution is to add compensation pseudoforces in PID loops…

$$\underline{F}_d \leftarrow \underline{F}_d - k_p\ \underline{c}(\underline{x})$$

$$\underline{F}_d \leftarrow \underline{F}_d - k_p\ \underline{c}(\underline{x}, \dot{\underline{x}})$$

- Gains relate to time constants:

$$k_p = \cancel{\frac{\tau}{\Delta t}} \quad \frac{\Delta t}{\tau}$$

3.4.4.4 Initial Conditions

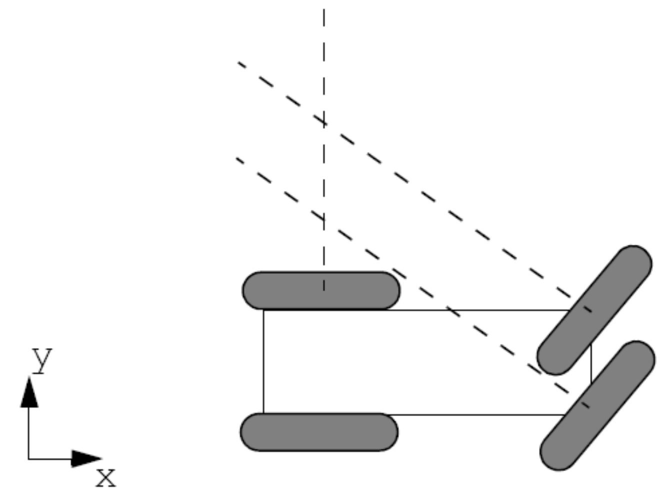
- Differentiated constraints will hold the constraints constant rather than at zero.
  - So they must start at zero to stay at zero.
- Two approaches.
  - 1) Start from zero velocity state which automatically satisfies constraints. Then activate system with forces.
  - 2) Start from moving state but guarantee constraints are satisfied by solving the rootfinding problem.

$$\underline{c}(\underline{q}) = 0$$

Carnegie Mellon
THE ROBOTICS INSTITUTE

Overconstraint

- Typical of wheeled vehicles.
- Leads to collapse of nullspace.
  - No motion possible.
  - … or constraints do work.
- Few approaches:
  - 1) Use any two independent constraints.
  - 2) Compute and equivalent bicycle model of constraints for each cycle.
  - 3) Use an equation solver that tolerates the situation
- It may or may not be appropriate to let the overconstrained system slow down.

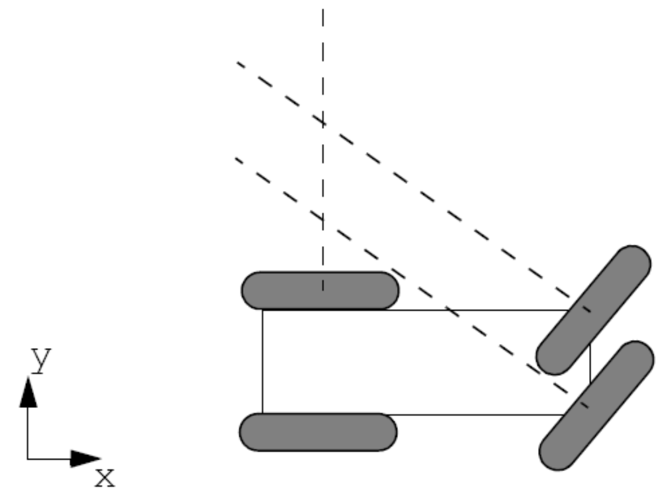


Redundant Constraints

- Example: two rear wheels of car generate the same constraint equation.
- Leads to singularity of the system.
- Good approach is avoid inversion. Compute least residual norm:

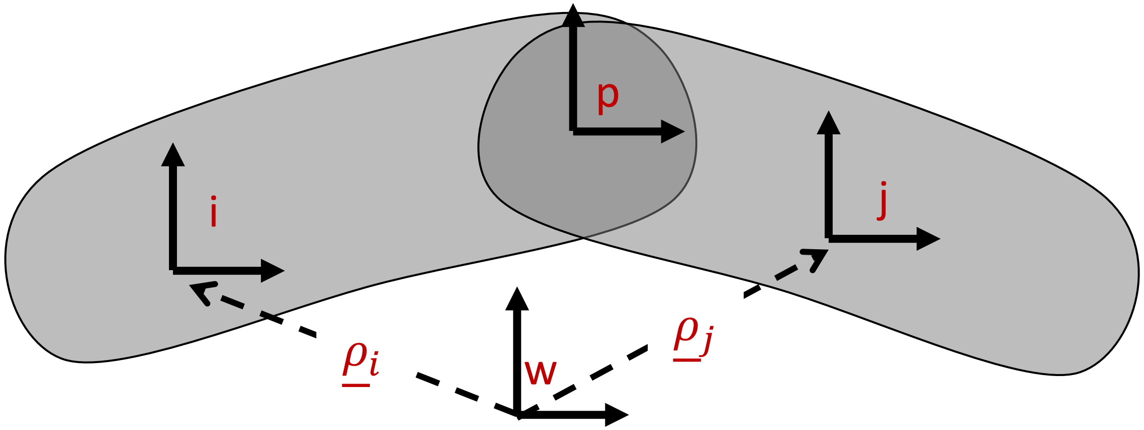
$$\underline{\lambda}^{*} = argmin \left\| C M^{-1} C^{T} \underline{\lambda} - F \right\|_{2}$$

- Conjugate gradient algorithm…



3.4.4.5 Basic Rigid Body Constraint

- Two bodies have poses known with respect to the world frame.
- If there is a rotary constraint, frame p is at the pivot point.



Rigidity Constraint

- Express this in terms of pose composition as follows:

$$\underline{\rho}_i^j = \underline{\rho}_w^j * \underline{\rho}_i^w = (\underline{\rho}_j^w)^{-1} * \underline{\rho}_i^w = const$$

- Holonomic of the form:

$$\underline{g}(\underline{x}) = const$$

- Equivalent to:

$$\underline{c}(\underline{x}) = \underline{g}(\underline{x}) - const = \underline{0}$$

Rigidity Constraint

- Gradient contains two elements:

$$C_{\underline{\rho}_i} = \frac{\partial \underline{\rho}_i^j}{\partial \underline{\rho}_i^w} \qquad C_{\underline{\rho}_j} = \frac{\partial \underline{\rho}_i^j}{\partial \underline{\rho}_j^w}$$

- The first is a right pose Jacobian:

$$C_{\underline{\rho}_i} = \frac{\partial \underline{\rho}_i^j}{\partial \underline{\rho}_i^w} = \begin{bmatrix} c\theta_w^j & -s\theta_w^j & 0 \\ s\theta_w^j & c\theta_w^j & 0 \\ 0 & 0 & 1 \end{bmatrix} = \boxed{\begin{bmatrix} c\theta_j^w & s\theta_j^w & 0 \\ -s\theta_j^w & c\theta_j^w & 0 \\ 0 & 0 & 1 \end{bmatrix}}$$

Carnegie Mellon
THE ROBOTICS INSTITUTE

Rigidity Constraint

- The second is more complicated. By the chain rule:

$$C_{\underline{\rho}_j} = \frac{\partial \underline{\rho}_i^j}{\partial \underline{\rho}_j^w} = \left(\frac{\partial \underline{\rho}_i^j}{\partial \underline{\rho}_w^j}\right)\left(\frac{\partial \underline{\rho}_w^j}{\partial \underline{\rho}_j^w}\right)$$

Left Pose Jacobian

Inverse Pose Jacobian

- This is:

$$C_{\underline{\rho}_j} = -\begin{bmatrix} 1 & 0 & -(y_i^j - y_w^j) \\ 0 & 1 & (x_i^j - x_w^j) \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\theta_j^w & s\theta_j^w & -y_w^j \\ -s\theta_j^w & c\theta_j^w & x_w^j \\ 0 & 0 & 1 \end{bmatrix}$$

$$C_{\underline{\rho}_j} = \boxed{-\begin{bmatrix} c\theta_j^w & s\theta_j^w & -y_i^j \\ -s\theta_j^w & c\theta_j^w & x_i^j \\ 0 & 0 & 1 \end{bmatrix}}$$

Rigidity Constraint

- Total Constraint Jacobian:

$$\underline{c}_{\underline{x}} = \begin{array}{c} \qquad\quad \text{Posn i} \qquad\qquad\qquad\qquad \text{Posn j} \\ \begin{bmatrix} \dots & c\theta & s\theta & 0 & \dots & -c\theta & -s\theta & y_i^j & \dots \\ \dots & -s\theta & c\theta & 0 & \dots & s\theta & -c\theta & -x_i^j & \dots \\ \dots & 0 & 0 & 1 & \dots & 0 & 0 & -1 & \dots \end{bmatrix} \end{array}$$

Where

$$\theta = \theta_j^w$$

- Time Derivative:

$$\underline{c}_{\underline{x}t} = \begin{bmatrix} \dots & -\omega s\theta & \omega c\theta & 0 & \dots & \omega s\theta & -\omega c\theta & 0 & \dots \\ \dots & -\omega c\theta & -\omega s\theta & 0 & \dots & \omega c\theta & \omega s\theta & 0 & \dots \\ \dots & 0 & 0 & 0 & \dots & 0 & 0 & 0 & \dots \end{bmatrix}$$

Where

$$\omega = \dot{\theta}$$

Rotary Joint Constraint

- Let p denote a reference frame attached to the point of rotation.
- The constraints for the rotary joint at the front wheel can be expressed as the first two elements of the equation:

$$\underline{\rho}_p^w - \underline{\rho}_p^w = 0$$

"p" means axis of rotation

$$\underline{\rho}_i^w * \underline{\rho}_p^i - \underline{\rho}_j^w * \underline{\rho}_p^j = 0$$

Rewritten in terms of generalized coordinates.

Unknowns

Knowns

Rotary Joint Constraint

- This gives:

$$J_{\underline{\rho}_i} = \frac{\partial \underline{\rho}_p^w}{\partial \rho_i^w} = \begin{bmatrix} 1 & 0 & -(y_p^w - y_i^w) \\ 0 & 1 & (x_p^w - x_i^w) \\ 0 & 0 & 1 \end{bmatrix} \quad J_{\underline{\rho}_j} = -\frac{\partial \underline{\rho}_p^w}{\partial \rho_j^w} = -\begin{bmatrix} 1 & 0 & -(y_p^w - y_j^w) \\ 0 & 1 & (x_p^w - x_j^w) \\ 0 & 0 & 1 \end{bmatrix}$$

Minus sign

- Hence, total constraint Jacobian is:

$$\begin{array}{cc} & \begin{array}{cc} \text{Posn i} & \text{Posn j} \end{array} \\ \underline{c}_{\underline{x}} = & \begin{bmatrix} \ldots & 1 & 0 & -\Delta y_i & \ldots & -1 & 0 & \Delta y_j & \ldots \\ \ldots & 0 & 1 & \Delta x_i & \ldots & 0 & -1 & -\Delta x_j & \ldots \end{bmatrix} \end{array}$$

$$\Delta x_i = (x_p^w - x_i^w)$$

$$\Delta y_i = (y_p^w - y_i^w)$$

$$\Delta x_j = (x_p^w - x_j^w)$$

$$\Delta y_j = (y_p^w - y_j^w)$$

Carnegie Mellon THE ROBOTICS INSTITUTE

Rotary Joint Constraint

- Time Derivative:

$$\begin{array}{cc} & \begin{array}{cc}\text{Posn i} & \text{Posn j}\end{array} \\ \underline{c}_{xt} = & \begin{bmatrix} \dots & 0 & 0 & -\Delta x_i \omega_i & \dots & 0 & 0 & \Delta x_j \omega_j & \dots \\ \dots & 0 & 0 & -\Delta y_i \omega_i & \dots & 0 & 0 & \Delta y_j \omega_j & \dots \end{bmatrix} \end{array}$$

$$\omega_i = \omega_i^w = \dot{\theta}_i^w$$

$$\omega_j = \omega_j^w = \dot{\theta}_j^w$$

- Fd vector is:

$$\underline{F}_d = -\underline{c}_{xt}\,\dot{\underline{x}}$$

$$\underline{F}_d = \begin{bmatrix} 0 & 0 & \Delta x_i \omega_i & 0 & 0 & -\Delta x_j \omega_j \\ 0 & 0 & \Delta y_i \omega_i & 0 & 0 & -\Delta y_j \omega_j \end{bmatrix} \begin{bmatrix} \dot{x}_i \\ \dot{y}_i \\ \dot{\theta}_i \\ \dot{x}_j \\ \dot{y}_j \\ \dot{\theta}_j \end{bmatrix}$$

Outline

- 3.4 Differential Algebraic Systems
  - 3.4.1 Constrained Dynamics
  - 3.4.2 First and Second Order Constrained Kinematic Systems
  - 3.4.3 Lagrangian Dynamics
  - 3.4.4 Constraints
  - <u>Summary</u>
- 3.5 Integration of Differential Equations

Summary - DAEs

- Simplest formulation for some problems.
- Only practical formulation for some problems.
- Can be really fast for mobile robots.
- Can be written in completely general way to simulate anything.

Outline

- 3.4 Differential Algebraic Systems
- 3.5 Integration of Differential Equations
  – <u>3.5.1 Dynamic Models in State Space</u>
  – 3.5.2 Integration of State Space Models

Carnegie Mellon
THE ROBOTICS INSTITUTE

3.5.1 State Space

- State space = a minimal set of variables which can be used to predict future state given inputs:
  - Number of initial conditions in a differential equation.

Carnegie Mellon
THE ROBOTICS INSTITUTE

3.5.1 Dynamic Models in State Space

- Predicting the future involves predicting trajectories caused by motion commands.
- General case:

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

Known as the State Space representation of the system

- The "inputs" u are a new addition.
- Known as a "forced" system although the inputs need not be forces.

Carnegie Mellon THE ROBOTICS INSTITUTE

Constraints

- The dynamics of wheeled mobile robots are constrained dynamics in 3D of systems of rigid bodies.
- Often must consider:
  - Actuator kinematics
  - Lateral and longitudinal wheel slip (or nonslip).
  - Terrain following.

Outline

- 3.4 Differential Algebraic Systems
- 3.5 Integration of Differential Equations
  - 3.5.1 Dynamic Models in State Space
  - 3.5.2 Integration of State Space Models
  - Summary

3.5.2.1 Euler's Method

- For the nonlinear differential equation:

$$\dot{\underline{x}}(t) = \underline{f}(\underline{x}, t)$$

- Seems reasonable to use the definition of integration:

$$\underline{x}(t + \Delta t) = \underline{x}(t) + \underline{f}(\underline{x}, t)\Delta t$$

- In discrete time:

$$\underline{x}_{k+1} = \underline{x}_k + \underline{f}(\underline{x}_k, t_k)\Delta t_k$$

- Works well if f() is nearly linear. Errors are 2nd order.

3.5.2.2 Midpoint Method

- Let's try for a 2nd order approximation. A 2nd order Taylor series is:

h= Δ t

$$\underline{x}(t+h) \approx \underline{x}(t) + \underline{f}(\underline{x}, t)h + \frac{d\underline{f}}{dt}(\underline{x}, t)\frac{h^2}{2}$$

- Which can be written as (factor out an h):

$$\underline{x}(t+h) \approx \underline{x}(t) + \left\{\underline{f}(\underline{x}, t) + \frac{d\underline{f}}{dt}(\underline{x}, t)\frac{h}{2}\right\}h \qquad \text{Eqn A}$$

- Now, the part in brackets is the first degree Taylor series for the first time derivative evaluated at the midpoint of the step because:

$$\underline{f}(\underline{x}(t+h/2), t+h/2) \approx \underline{f}(\underline{x}, t) + \frac{d\underline{f}}{dt}(\underline{x}, t)\frac{h}{2}$$

3.5.2.2 Midpoint Method

- The derivative $d\underline{f}(\underline{x},t)/dt$ is typically expensive computationally. Instead, invert the last formula to produce a finite difference approximation:

$$\frac{d\underline{f}(\underline{x},t)}{dt}\frac{h}{2} \approx \underline{f}(\underline{x}(t+h/2), t+h/2) - \underline{f}(\underline{x},t)$$

- Substituting into Eqn A produces:

$$\underline{x}(t+h) \approx \underline{x}(t) + h\underline{f}(\underline{x}(t+h/2), t+h/2)$$

- And, the value of x at the midpoint can be approximated:

$$\underline{x}(t+h/2) \approx \underline{x}(t) + \underline{f}(\underline{x},t)(h/2)$$

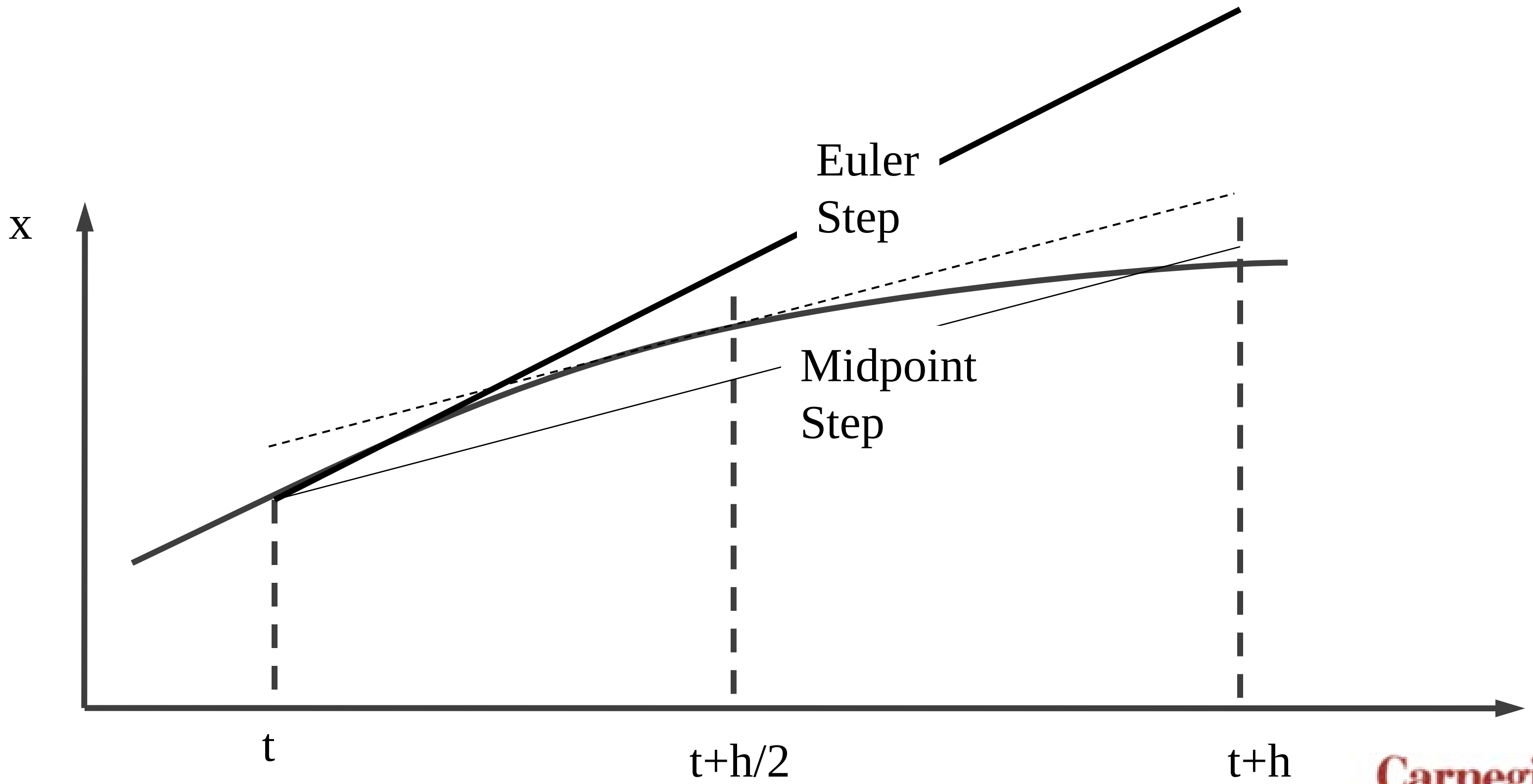
- This gives finally:

$$\underline{x}(t+h) \approx \underline{x}(t) + h\underline{f}[\underline{x}(t) + \underline{f}(\underline{x},t)(h/2), t+h/2]$$

3.5.2.2 Midpoint Method

- For future reference, this is best written as:

$$\underline{k} = \underline{x}(t) + \underline{f}(\underline{x}, t)(h/2)$$
$$\underline{x}(t+h) \approx \underline{x}(t) + h\underline{f}(\underline{k}, t+h/2)$$



Example

- Example, integrate a general curve with respect to distance:

$$\begin{bmatrix} x(s) \\ y(s) \\ \theta(s) \end{bmatrix} = \begin{bmatrix} x(0) \\ y(0) \\ \theta(0) \end{bmatrix} + \int_0^s \begin{bmatrix} \cos\theta(s) \\ \sin\theta(s) \\ \kappa(s) \end{bmatrix} ds$$

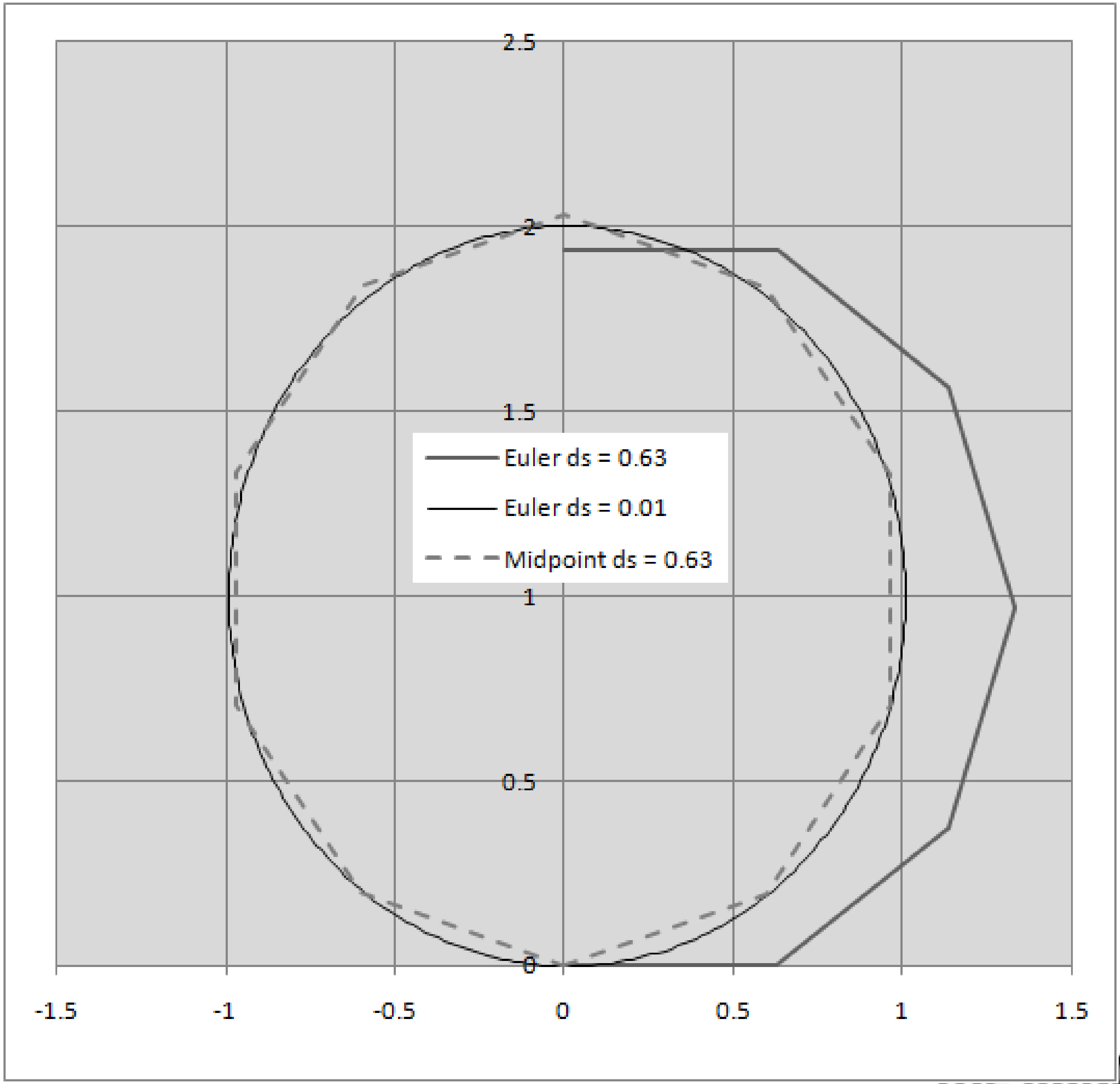
- In discrete time:

$$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}_{k+1} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}_k + \begin{bmatrix} \cos\theta \\ \sin\theta \\ \kappa \end{bmatrix}_k \Delta s$$

- Specific curvature profile is constant (arc):

$$\kappa(s) = \kappa_0 = 1$$

Example



Midpoint Algorithm Reduces Error By a Factor Of 20

Carnegie Mellon
THE ROBOTICS INSTITUTE

3.5.2.3 (4th Order) Runge Kutta

- Closest thing to a definitive algorithm for integration.

$$
\begin{aligned}
\underline{k}_1 &= h\underline{f}(\underline{x}, \underline{u}, t)\\
\underline{k}_2 &= h\underline{f}[\underline{x}(t) + \underline{k}_1/2, \underline{u}(t+h/2), t+h/2]\\
\underline{k}_3 &= h\underline{f}[\underline{x}(t) + \underline{k}_2/2, \underline{u}(t+h/2), t+h/2]\\
\underline{k}_4 &= h\underline{f}[\underline{x}(t) + \underline{k}_3, \underline{u}(t+h), t+h]\\
\underline{x}(t+h) &= \underline{x}(t) + \underline{k}_1/6 + \underline{k}_2/3 + \underline{k}_3/3 + \underline{k}_4/6
\end{aligned}
$$

- This can be 1000 times more accurate than Euler’s method.

Outline

- 3.4 Differential Algebraic Systems
- 3.5 Integration of Differential Equations
  - 3.5.1 Dynamic Models in State Space
  - 3.5.2 Integration of State Space Models
  - Summary

Summary

- This is worth knowing about.
- A few lines of code can be the difference between:
  - 100 mm of error after moving 10 meters
  - 0.1 mm of error after moving 10 meters