ME/CS 132:
Advanced Robotics:
Navigation and Vision

Lecture #2: Motion Simulation

Yoshiaki Kuwata

3/31/2011

Lecture Overview

- More vehicle dynamics models

- Why motion simulation?
- Analytical integration
  - LTI system
- Numerical integration schemes
  - Euler integration
  - Runge-Kutta integration

Point-mass Spacecraft Model

- Vehicle is treated as a point

Notation in LaValle's book:
(q: configuration)

$$\boldsymbol{F} = m\boldsymbol{a} = m\ddot{\boldsymbol{q}}$$

- Assume 6 thrusters attached on 6 planes can apply force in each of ±X, ±Y, ±Z.
  - Note $f_{x+}, f_{x-}, f_{y+}, f_{y-}, f_{z+}, f_{z-} \geq 0$

$$\boldsymbol{F} = \begin{bmatrix} f_{x+} - f_{x-} \\ f_{y+} - f_{y-} \\ f_{z+} - f_{z-} - mg \end{bmatrix}$$

- State space model: $\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t))$
  - State
  - Control
  - Dynamics

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{q} \\ \dot{\boldsymbol{q}} \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} \qquad \boldsymbol{u} = \begin{bmatrix} f_{x+} \\ f_{x-} \\ f_{y+} \\ f_{y-} \\ f_{z+} \\ f_{z-} \end{bmatrix} \qquad \dot{\boldsymbol{x}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ 0 \\ 0 \\ 0 \end{bmatrix} + \frac{1}{m} \begin{bmatrix} 0 \\ 0 \\ 0 \\ f_{x+} - f_{x-} \\ f_{y+} - f_{y-} \\ f_{z+} - f_{z-} - mg \end{bmatrix}$$

Side note: Linear Time-Invariant system

- Sometimes LTI system is "defined" as

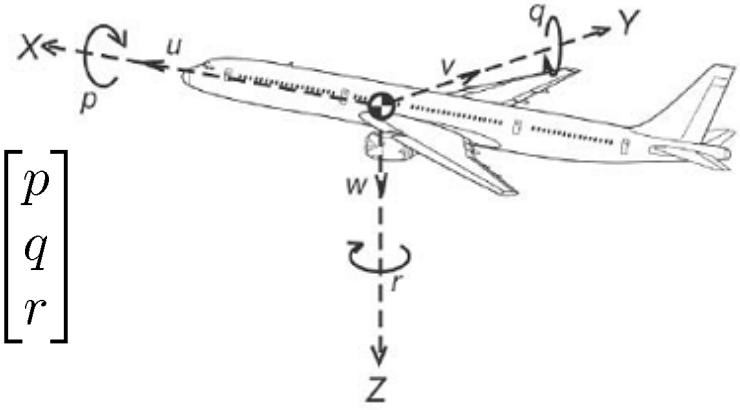
$$\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t)) = A\boldsymbol{x}(t) + B\boldsymbol{u}(t)$$

- However, it is derived from the superposition principle
  - If applying $\boldsymbol{u}_1(t), \quad t \in [0, t_f]$ from state $\boldsymbol{x}(0) = \boldsymbol{x}_0$ gives a trajectory $\boldsymbol{x}_1(t), \quad t \in [0, t_f]$, and
  - if applying $\boldsymbol{u}_2(t), \quad t \in [0, t_f]$ gives a trajectory $\boldsymbol{x}_2(t), \quad t \in [0, t_f]$, then
  - applying $\alpha\boldsymbol{u}_1(t) + \beta\boldsymbol{u}_2(t), \quad t \in [0, t_f]$ gives a trajectory $\alpha\boldsymbol{x}_1(t) + \beta\boldsymbol{x}_2(t), \quad t \in [0, t_f]$

  - If the dimension of the state is finite, then it can be shown that the LTI system can be expressed as $\dot{\boldsymbol{x}}(t) = A\boldsymbol{x}(t) + B\boldsymbol{u}(t)$ (proof: not trivial)

Dynamics of Quadrotor: Definitions

- State
  - Position (x,y,z) in the inertial frame & velocity in the inertial frame
  - Euler angles (ϕ, θ, ψ) and their time derivatives in the body frame
- Aircraft body frame convention
  - $+x_B$ pointing forward
  - $+y_B$ pointing right
  - $+z_B$ pointing down
  - roll ϕ: rotation around $+x_B$ axis
  - pitch θ: $+y_B$ axis
  - yaw ψ: $+z_B$ axis



$$\boldsymbol{\omega}_B = \frac{d}{dt}\begin{bmatrix}\phi\\ \theta\\ \psi\end{bmatrix} = \begin{bmatrix}p\\ q\\ r\end{bmatrix}$$

Transport Theorem

- Derivative of a vector in the coordinate frame that is rotating
  - body frame B

$$\frac{d\boldsymbol{x}}{dt} = \frac{\delta \boldsymbol{x}}{\delta t} + \boldsymbol{\omega}_B \times \boldsymbol{x}$$

Rate of change observed in the inertial frame

$$= \left(\frac{d\boldsymbol{x}}{dt}\right)_r + \boldsymbol{\omega}_B \times \boldsymbol{x}$$

$$= \dot{\boldsymbol{x}}_B + \boldsymbol{\omega}_B \times \boldsymbol{x}$$

Rate of change observed in the rotating frame

Change of the frame itself

Transport Theorem: proof

- Express the vector in the body frame: $\boldsymbol{x} = x_i \boldsymbol{i} + x_j \boldsymbol{j} + x_k \boldsymbol{k}$
- Apply chain rule:

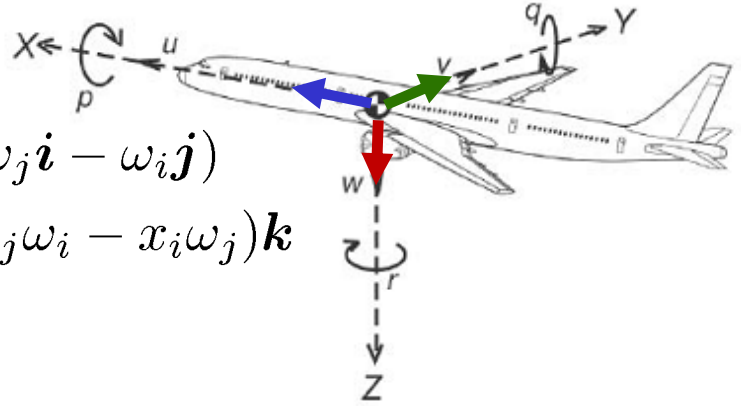
$$\frac{d\boldsymbol{x}}{dt} = \frac{dx_i}{dt}\boldsymbol{i} + \frac{dx_j}{dt}\boldsymbol{j} + \frac{dx_k}{dt}\boldsymbol{k} + x_i\frac{d\boldsymbol{i}}{dt} + x_j\frac{d\boldsymbol{j}}{dt} + x_k\frac{d\boldsymbol{k}}{dt}$$

- Rotation of the unit vector

$$\frac{d\boldsymbol{i}}{dt} = (\omega_k \boldsymbol{j} - \omega_j \boldsymbol{k}), \quad \frac{d\boldsymbol{j}}{dt} = (\omega_i \boldsymbol{k} - \omega_k \boldsymbol{i}), \quad \frac{d\boldsymbol{k}}{dt} = (\omega_j \boldsymbol{i} - \omega_i \boldsymbol{j})$$

- Then **this** becomes

$$\begin{aligned} & x_i(\omega_k \boldsymbol{j} - \omega_j \boldsymbol{k}) + x_j(\omega_i \boldsymbol{k} - \omega_k \boldsymbol{i}) + x_k(\omega_j \boldsymbol{i} - \omega_i \boldsymbol{j}) \\ &= (x_k\omega_j - x_j\omega_k)\boldsymbol{i} + (x_i\omega_k - x_k\omega_i)\boldsymbol{j} + (x_j\omega_i - x_i\omega_j)\boldsymbol{k} \\ &= \boldsymbol{\omega} \times \boldsymbol{x} \end{aligned}$$



- Finally

$$\frac{d\boldsymbol{x}}{dt} = \dot{\boldsymbol{x}}_B + \boldsymbol{\omega} \times \boldsymbol{x}$$

Dynamics of Quadrotor: EOM

- Translational

Inertial frame

$$\boldsymbol{F} = \frac{d(m\boldsymbol{v})}{dt} = m \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix}$$

- Rotational

Body frame

Transport Theorem

$$\boldsymbol{N} = \frac{d(I\boldsymbol{\omega}_B)}{dt} = I\dot{\boldsymbol{\omega}}_B + \boldsymbol{\omega}_B \times I\boldsymbol{\omega}_B$$

$$= \begin{bmatrix} I_x\ddot{\phi} \\ I_y\ddot{\theta} \\ I_z\ddot{\psi} \end{bmatrix} + \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \times \begin{bmatrix} I_x\dot{\phi} \\ I_y\dot{\theta} \\ I_z\dot{\psi} \end{bmatrix}$$

$$= \begin{bmatrix} I_x\ddot{\phi} \\ I_y\ddot{\theta} \\ I_z\ddot{\psi} \end{bmatrix} + \begin{bmatrix} \dot{\theta}\dot{\psi}(I_z - I_y) \\ \dot{\phi}\dot{\psi}(I_x - I_z) \\ \dot{\phi}\dot{\theta}(I_y - I_x) \end{bmatrix}$$

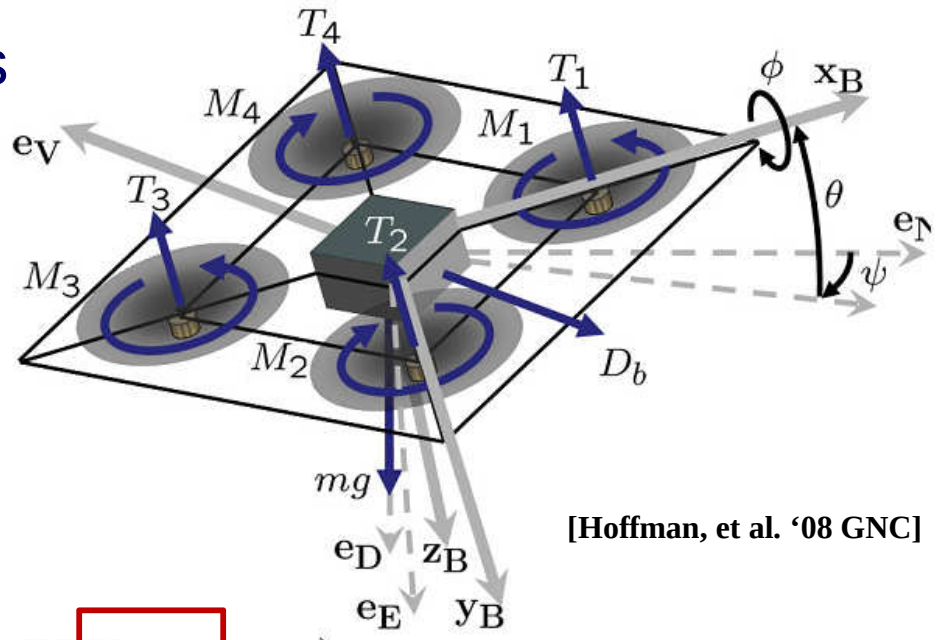
$$\omega_B = \frac{d}{dt}\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

$$I = \begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix}$$

- What are F and N?

Dynamics of Quadrotor: Forces

- Four rotors as control inputs
  - Thrust: T_i (i=1, 2, 3, 4)
  - Moment: N_i
    (N_i is a function of T_i)
  - Arm length (from C.G.): l_i
  - Rotor plane frame: R_i
- Then,



[Hoffman, et al. '08 GNC]

$$\mathbf{F} = \underbrace{-D_b \mathbf{e_v}}_{\text{Drag}} + \underbrace{mg\mathbf{e_D}}_{\text{Gravity}} + \sum_{i=1}^{4} (-T_i R_{R_i,I} \mathbf{z}_{R_i})$$

$$\boldsymbol{N} = \sum_{i=1}^{4} (\boldsymbol{N}_i + \boldsymbol{l}_i \times (-T_i R_{R_i,B} \boldsymbol{z}_{R_i}))$$

Unit vector of i^{th} rotor's plane

Rotation matrix from the frame of i^{th} rotor to inertial coordinates

Rotation matrix from the frame of i^{th} rotor to body coordinates

Dynamics of Quadrotor

- The rotation matrix from inertial frame to body frame in 3D

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = R_1(\phi)R_2(\theta)R_3(\psi) \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

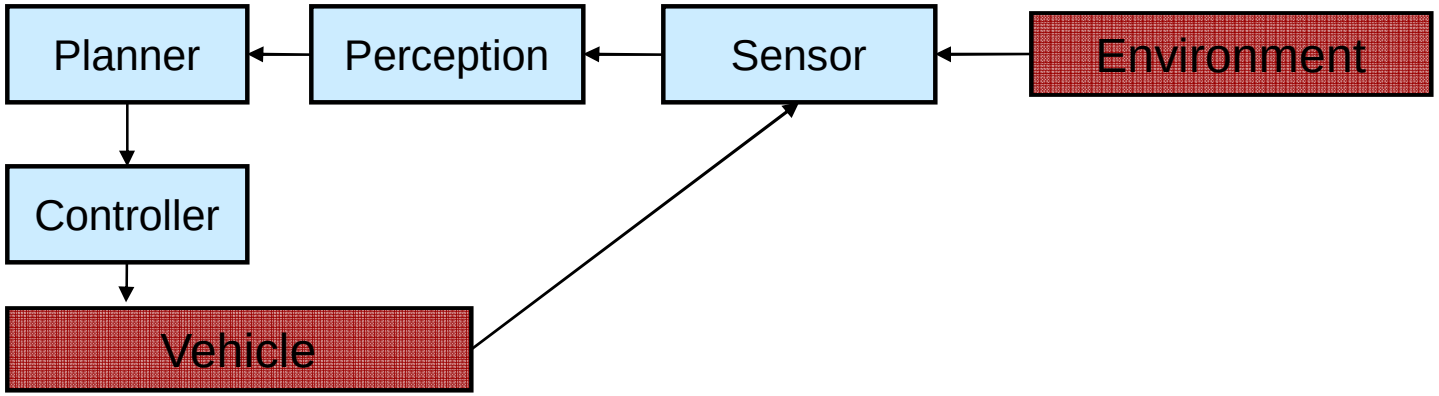
$$= \begin{bmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \\ -\cos\phi\sin\psi + \sin\phi\sin\theta\cos\psi & \cos\phi\cos\psi + \sin\phi\sin\theta\sin\psi & \sin\phi\cos\theta \\ \sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi & -\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi & \cos\phi\cos\theta \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

- Rotor planes are tilted from the body frame
  → more rotation matrices
- So it's clearly very non-linear

Motion Simulation

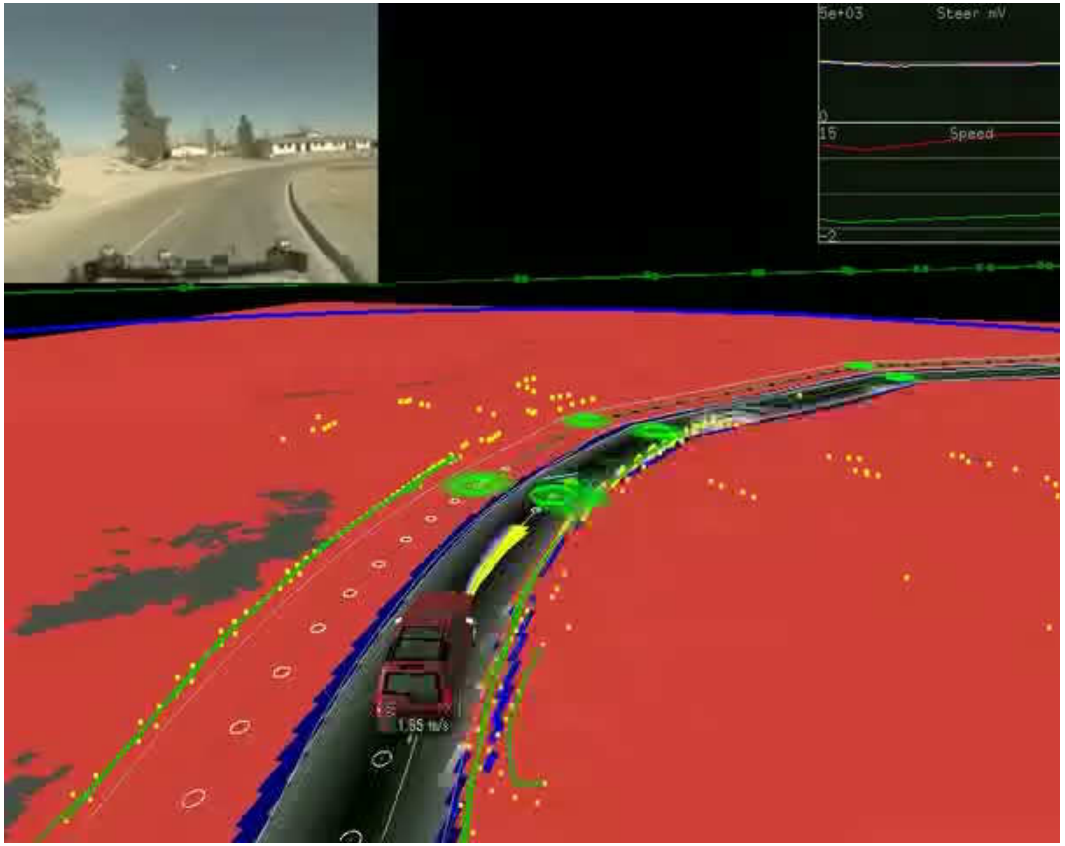
Why Motion Simulation?

- Testing/Debugging
  - Verify the algorithm & the software implementation
    ➔ before testing on the real hardware
  - Can do many things that are difficult on hardware
    - Can pause/restart the motion
    - Can replay the exact same scenario
    - Can run numerous test instances
    - Can be much cheaper



Why Motion Simulation?

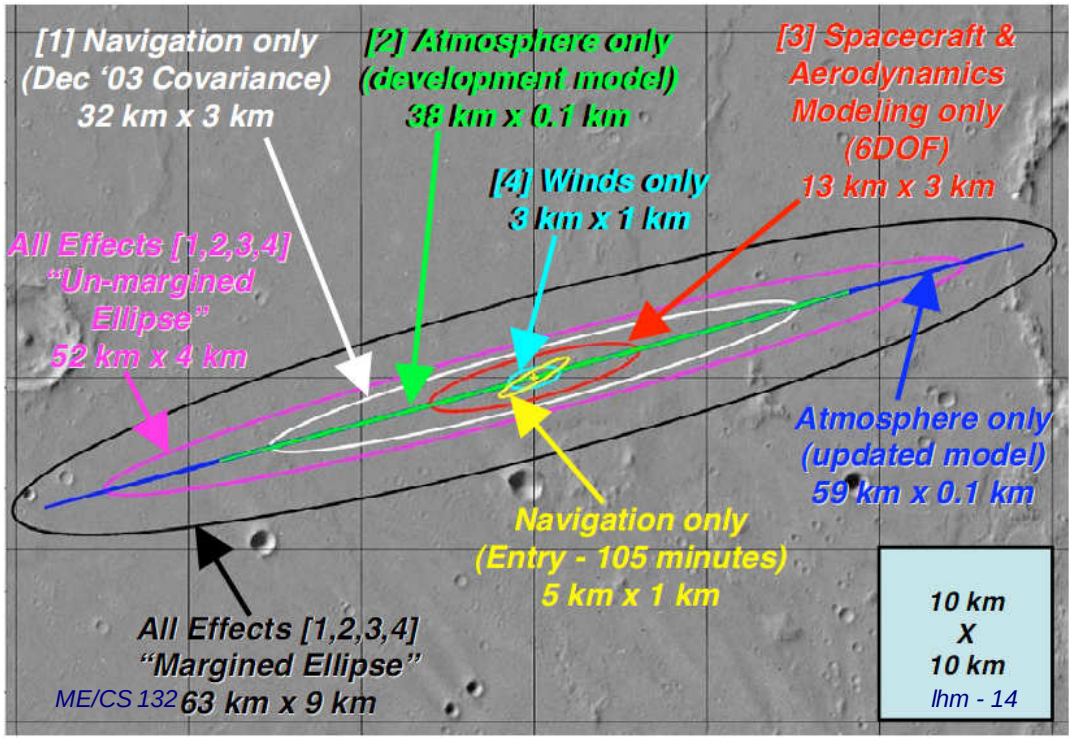
- Predictive planning
  - Motion planner can use a simulator to generate trajectories
    ➔ More deliberative planner than reactive
  - Typically better performance than simply reacting



Why Motion Simulation?

- No analytical solution available
- Mars EDL (Entry, Descent, and Landing)
  - Want to predict where the spacecraft will land
  - Several sources of uncertainties
    - Uncertainties in the entry states (position, velocity, attitude)
    - Vehicle aerodynamics
    - Navigation error build-up from inertial sensor noise
    - Variability in atmospheric density and winds
  - Non-linear

[Knocke, et al. '04 ASC]



Motion Simulation

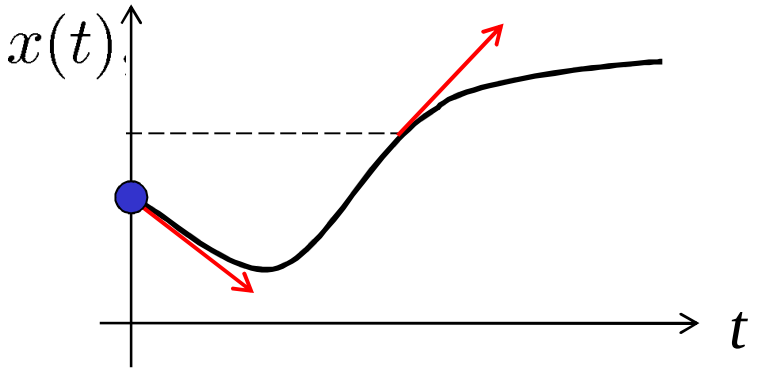
- System dynamics written as an ODE (ordinary differential equation)

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

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

LaValle's book (q: configuration)

- "Compute the state trajectory $\boldsymbol{x}(t), \quad t \in [0, t_f]$ given the initial state $\boldsymbol{x}(0) = \boldsymbol{x}_0$ and a control input profile $\boldsymbol{u}(t), \quad t \in [0, t_f]$, by integrating the system dynamics $\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t))$"



"Initial value problem"

Analytical Integration

Analytical Integration

- Certain class of ODEs have analytical solutions
  - Separable

$$\dot{x}(t) = f(x,t) = \frac{g(t)}{h(x)}$$

$$h(x)\,dx = g(t)\,dt$$

$$\int h(x)\,dx = \int g(t)\,dt + C$$

- Example: linear autonomous system

$$\dot{\boldsymbol{x}}(t) = A\boldsymbol{x}(t), \quad \boldsymbol{x}(0) = \boldsymbol{x}_0$$

➔ $$\boldsymbol{x}(t) = \exp(At)\boldsymbol{x}_0$$

where $$\exp(At) = I + At + \frac{(At)^2}{2!} + \frac{(At)^3}{3!} + \cdots$$

  - Note that MATLAB “exp” function is element-wise
  - Use “expm” to compute the matrix exponential

Example: MATLAB

```
>> syms a b c d; exp([a, b; c d])

ans =

[ exp(a), exp(b)]
[ exp(c), exp(d)]
```

```
>> exp([0, 1; 0, 0])

ans =

    1.0000    2.7183
    1.0000    1.0000
```

```
>> expm([0, 1; 0, 0])

ans =

     1     1
     0     1
```

Example: LTI systems

- LTI system

$$\dot{\boldsymbol{x}}(t) = A\boldsymbol{x}(t) + B\boldsymbol{u}(t), \quad \boldsymbol{x}(0) = \boldsymbol{x}_0$$

- If A and B are scalar, and *u(t)* is given, then the solution is

$$x(t) = e^{at}x(0) + \int_0^t e^{a(t-\tau)}bu(\tau)d\tau$$

$$\frac{d}{dt}\left[e^{at}x(0)\right] = ae^{at}x(0)$$

$$= e^{at}\int_0^t e^{-a\tau}bu(\tau)d\tau$$

$$\frac{d}{dt}\left[e^{at}\int_0^t e^{-a\tau}bu(\tau)d\tau\right] = ae^{at}\int_0^t e^{-a\tau}bu(\tau)d\tau + e^{at}e^{-at}bu(t)$$

$$\frac{d}{dt}\left[e^{at}\int_0^t e^{-a\tau}bu(\tau)d\tau\right] = ax(t) + bu(t)$$

Holds true with matrices A, B

Example: Bicycle model

- Bicycle model
  - Input: steering angle

$$\begin{cases} \dot{x} = v\cos\theta \\ \dot{y} = v\sin\theta \\ \dot{\theta} = \dfrac{v}{L}\tan\delta \end{cases} \Rightarrow \begin{cases} \dot{x} = v \\ \dot{y} = v\theta \\ \dot{\theta} = \dfrac{v}{L}\delta \end{cases}$$

- Linearized model
  - Assume constant speed
  - Small angle approximation

$$\theta, \delta \ll 1: \quad \cos\theta \simeq 1, \quad \sin\theta \simeq \theta, \quad \tan\delta \simeq \delta$$

$$\begin{cases} x(t) = vt \\ \dfrac{d}{dt}\begin{bmatrix} y \\ \theta \end{bmatrix} = \begin{bmatrix} 0 & v \\ 0 & 0 \end{bmatrix}\begin{bmatrix} y \\ \theta \end{bmatrix} + \begin{bmatrix} 0 \\ v/L \end{bmatrix}\delta \end{cases}$$

  - LTI system with matrices

$$A = \begin{bmatrix} 0 & v \\ 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ v/L \end{bmatrix}$$

Note: $A^2 = O$

Example: Bicycle model

- Can write down the matrix exponential of A by hand

$$\exp(At) = I + At + \frac{(At)^2}{2!} \cdots = I + At = \begin{bmatrix} 1 & vt \\ 0 & 1 \end{bmatrix}$$

- Then, the solution is

$$\boldsymbol{x}(t) = e^{At}\boldsymbol{x}(0) + \int_0^t e^{A(t-\tau)} B\boldsymbol{u}(\tau) d\tau$$

$$\begin{aligned} \begin{bmatrix} y(t) \\ \theta(t) \end{bmatrix} &= \begin{bmatrix} 1 & vt \\ 0 & 1 \end{bmatrix} \begin{bmatrix} y_0 \\ \theta_0 \end{bmatrix} + \begin{bmatrix} 1 & vt \\ 0 & 1 \end{bmatrix} \int_0^t \begin{bmatrix} 1 & -v\tau \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ v/L \end{bmatrix} \delta(\tau) d\tau \\ &= \begin{bmatrix} 1 & vt \\ 0 & 1 \end{bmatrix} \begin{bmatrix} y_0 \\ \theta_0 \end{bmatrix} + \begin{bmatrix} 1 & vt \\ 0 & 1 \end{bmatrix} \int_0^t \begin{bmatrix} -v^2\tau/L \\ v/L \end{bmatrix} \delta(\tau) d\tau \end{aligned}$$

- If the steering input is sinusoidal $\delta(t) = \delta_{\max} \sin \omega t$

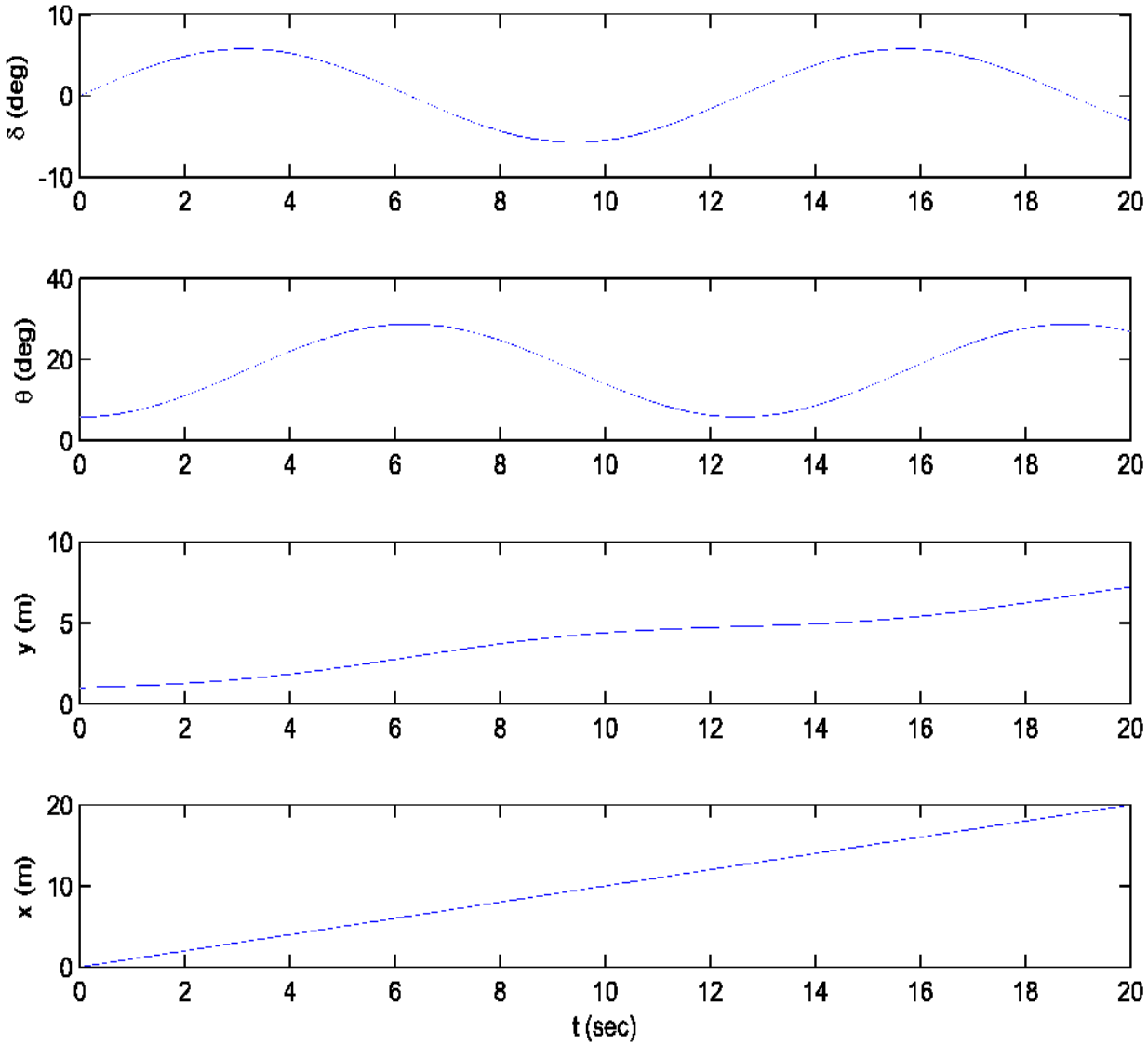
$$\int_0^t \delta(\tau)\, d\tau = \frac{\delta_{\max}}{\omega}(1 - \cos \omega t)$$

$$\int_0^t \tau\delta(\tau)\, d\tau = \frac{\delta_{\max}}{\omega^2}(\sin \omega t - \omega t \cos \omega t)$$

Example: Bicycle model

- MATLAB
  - Initial condition
    - x = 0
    - y = 1
    - θ = 0.1
  - Parameters
    - ω = 0.5
    - δ_{max}= 0.1
  - Valid when

$$\theta, \delta \ll 1: \quad \cos\theta \simeq 1$$
$$\sin\theta \simeq \theta$$
$$\tan\delta \simeq \delta$$



Side note: Continuous → Discrete

- It is sometime convenient to use a discrete form when implementing on digital computer
  - Kalman filter's dynamics equation

$$\boldsymbol{x}_{k+1} = A_d \boldsymbol{x}_k + B_d \boldsymbol{u}_k$$

  - How do the matrices A & B relate to those of the continuous form?

$$\dot{\boldsymbol{x}}(t) = A\boldsymbol{x}(t) + B\boldsymbol{u}(t), \quad \boldsymbol{x}(0) = \boldsymbol{x}_0$$

  - Use the result from the previous slide

$$\boldsymbol{x}(\Delta t) = e^{A\Delta t}\boldsymbol{x}(0) + \int_0^{\Delta t} e^{A(\Delta t - \tau)} B\boldsymbol{u}(\tau) d\tau$$

  - If the control input u(t) is constant over $0 \le t \le \Delta t$ ("zero-order hold")

$$\boldsymbol{x}(\Delta t) = e^{A\Delta t}\boldsymbol{x}(0) + \left(e^{A\Delta t} \int_0^{\Delta t} e^{-A\tau} d\tau B \right) \boldsymbol{u}(0)$$

Example: Continuous vs Discrete

- 1-D double integrator with force as the control input

$$m\ddot{x}(t) = u(t)$$

- Continuous form

$$\boldsymbol{x}(t) = \begin{bmatrix} r(t) \\ v(t) \end{bmatrix} \qquad \frac{d}{dt}\begin{bmatrix} r \\ v \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} r \\ v \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} u(t)$$

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

- Discrete form (assuming $u(t)$ is constant from t_k to t_{k+1})

$$\boldsymbol{x}_k = \begin{bmatrix} r_k \\ v_k \end{bmatrix} \qquad \begin{bmatrix} r_{k+1} \\ v_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}\begin{bmatrix} r_k \\ v_k \end{bmatrix} + \begin{bmatrix} \frac{\Delta t^2}{2m} \\ \frac{\Delta t}{m} \end{bmatrix} u_k$$

$$x_{k+1} = f_d(x_k, u_k)$$

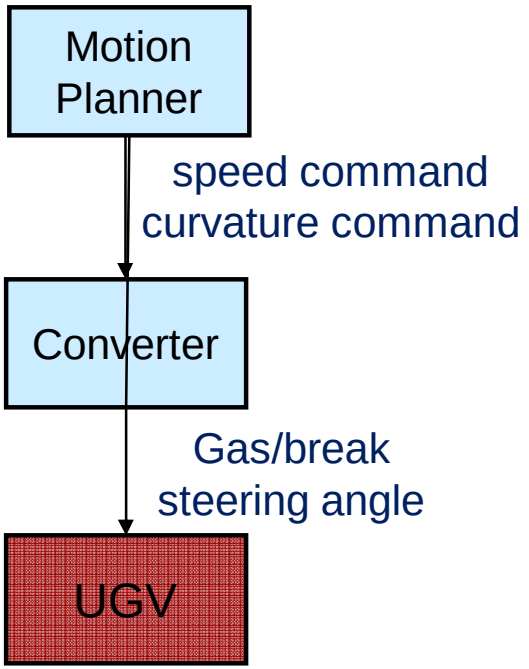
- There is a nice formula to convert one from the other for LTI systems, but not addressed in this class

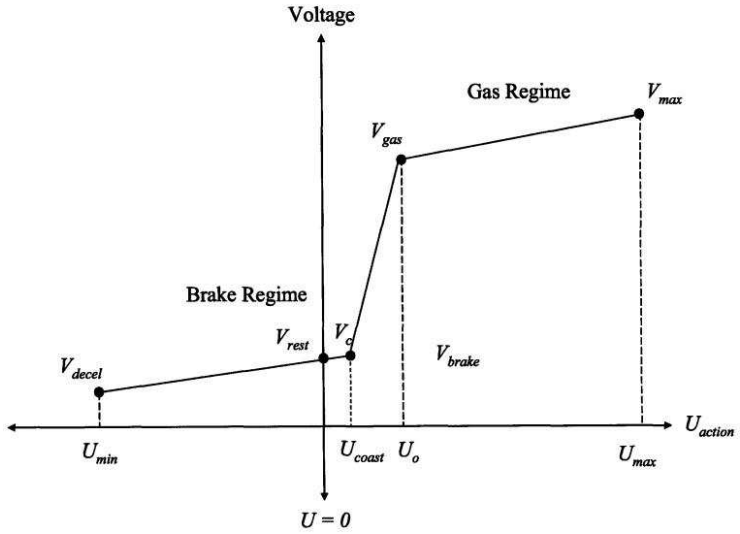
Different Sets of Control Inputs

- Actual interface to the vehicle might be
  - Steering voltage
  - Gas & break voltage
- Users might want to command with
  - steering angle
  - turn rate
  - curvature
- Conversion could be nonlinear

$$\dot{\theta} = \frac{v}{L}\tan\delta$$

$$\kappa = \frac{1}{r} = \frac{\dot{\theta}}{v} = \frac{\tan\delta}{L}$$





Closed-loop Simulation

- In the previous slides, the **control input** sequence was defined as a function of time $\boldsymbol{u}(t), \quad t \in [0, t_f]$
  - Controller blindly applies the pre-determined input no matter what → "Open-loop simulation"
- The control input could be defined as a function of states $\boldsymbol{u}(t) = \pi(\boldsymbol{x}(t))$
  - Controller might adjust the input depending on the states → can reject disturbances
  - $\pi(\cdot)$ is called state feedback "**Control law**"
  - If the control law is linear, $\boldsymbol{u}(t) = K\boldsymbol{x}(t)$. K is called "feedback gain"
- "Compute the state trajectory $\boldsymbol{x}(t), \quad t \in [0, t_f]$, given the initial state $\boldsymbol{x}(0) = \boldsymbol{x}_0$ and a control law $\boldsymbol{u}(t) = \pi(\boldsymbol{x}(t), t)$."
- Analytical integration more difficult with nonlinear control laws

Numerical Integration

Numerical Integration

- Not needed if the system dynamics are simple and have a analytical/closed-form solution
  - Might not apply to real systems due to nonlinear dynamics, uncertain terms, complex control law, saturation, etc.
- Start from some initial condition

$$x(0) = x_0$$

- Break t into small intervals of size Δt, and iterate

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

$$\Longleftrightarrow x(t_{k+1}) = x(t_k) + \int_{t_k}^{t_{k+1}} f(x(t), u(t))dt$$

$$\approx f(x(t_k), u(t_k))(t_{k+1} - t_k)$$

With a small Δt

Slope

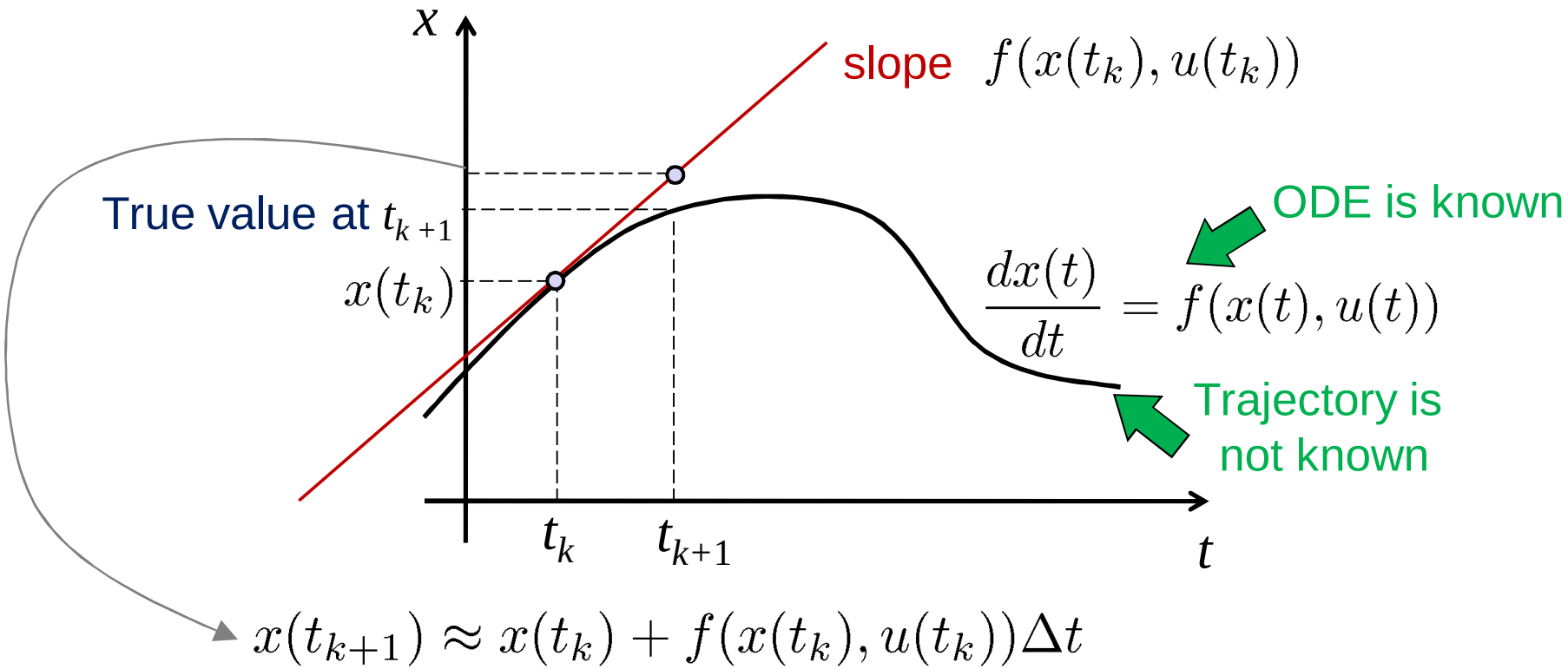
Step size Δt

Numerical Integration of ODEs

- Basic algorithm flow
  1. Set the initial condition $x(t_k) = x(0) = x_0$
  2. **Numerically approximate the slope** in $t \in [t_k, t_{k+1}]$ using the system equation $\dot{x}(t) = f(x(t), u(t))$ → call it $\bar{f}$
  3. Add $x(t_{k+1}) = x(t_k) + \bar{f}\Delta t$
  4. Increment *k* and go to step 2
- Two very popular integration scheme
  - Euler integration
    - First-order → Error term is O(dt^2)
    - Enough for most simple applications
  - Runge-Kutta integration
    - Second-order → Error term is O(dt^3)
    - Fourth-order → Error term is O(dt^5)
  - **The difference is only in the way $\bar{f}$ is approximated**

(Forward) Euler Integration

- Use the slope at t_k as the slope between t_k and t_{k+1}

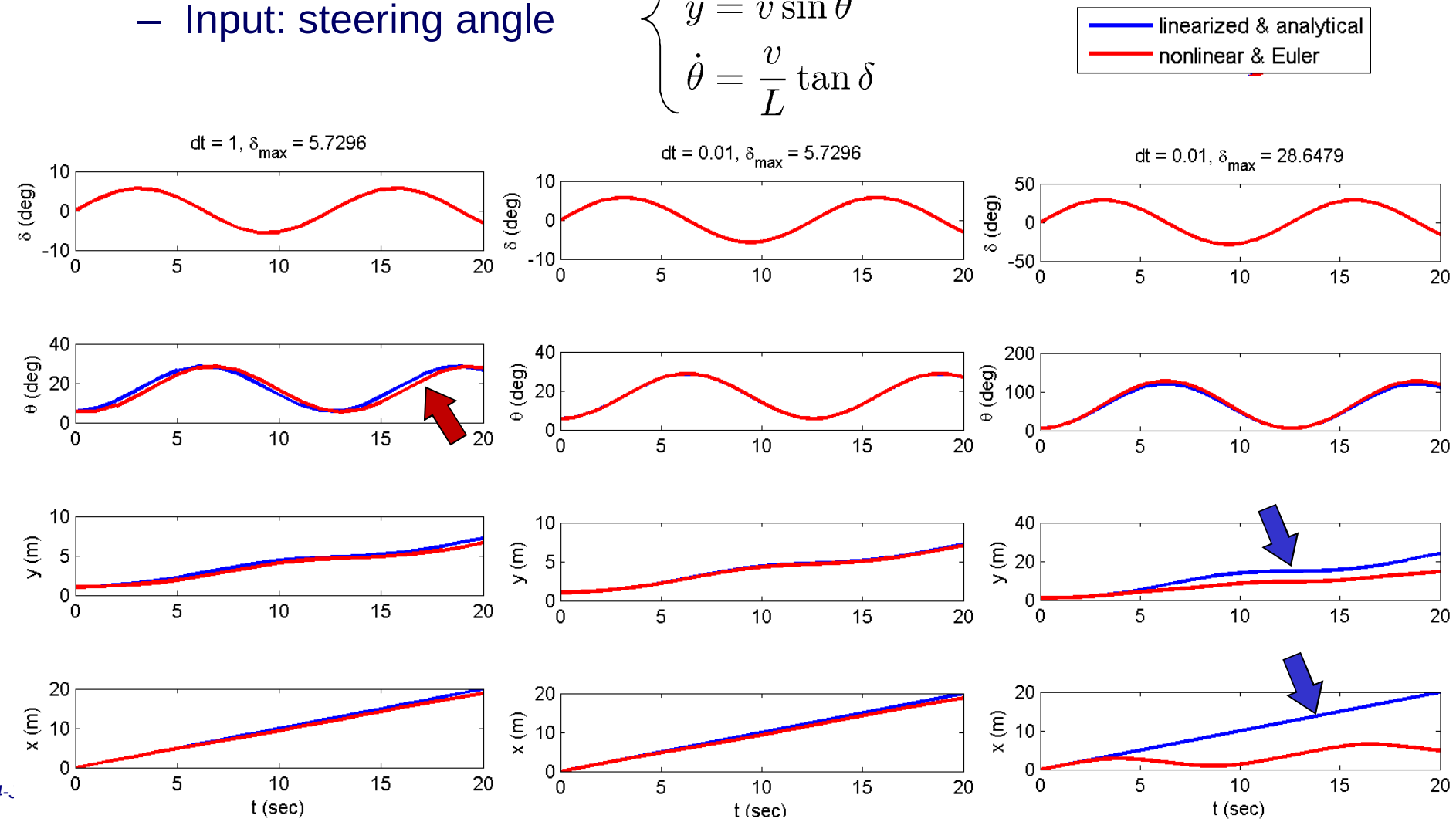


- Backward Euler:
  Use the slope at t_{k+1} as the slope between t_k and t_{k+1}

Example: Bicycle model

- Bicycle model
  - Input: steering angle

$$\begin{cases} \dot{x} = v\cos\theta \\ \dot{y} = v\sin\theta \\ \dot{\theta} = \dfrac{v}{L}\tan\delta \end{cases}$$



Runge-Kutta

- Developed by two mathematicians Runge and Kutta around 1900
- 4-th order Runge-Kutta is widely used

$$x(\Delta t) \approx x(0) + \frac{\Delta t}{6}(w_1 + 2w_2 + 2w_3 + w_4)$$

$$w_1 = f(x(0), u(0))$$
$$w_2 = f(x(0) + \tfrac{1}{2}\Delta t\ w_1,\ u(\tfrac{1}{2}\Delta t))$$
$$w_3 = f(x(0) + \tfrac{1}{2}\Delta t\ w_2,\ u(\tfrac{1}{2}\Delta t))$$
$$w_4 = f(x(0) + \Delta t\ w_3,\ u(\Delta t)).$$

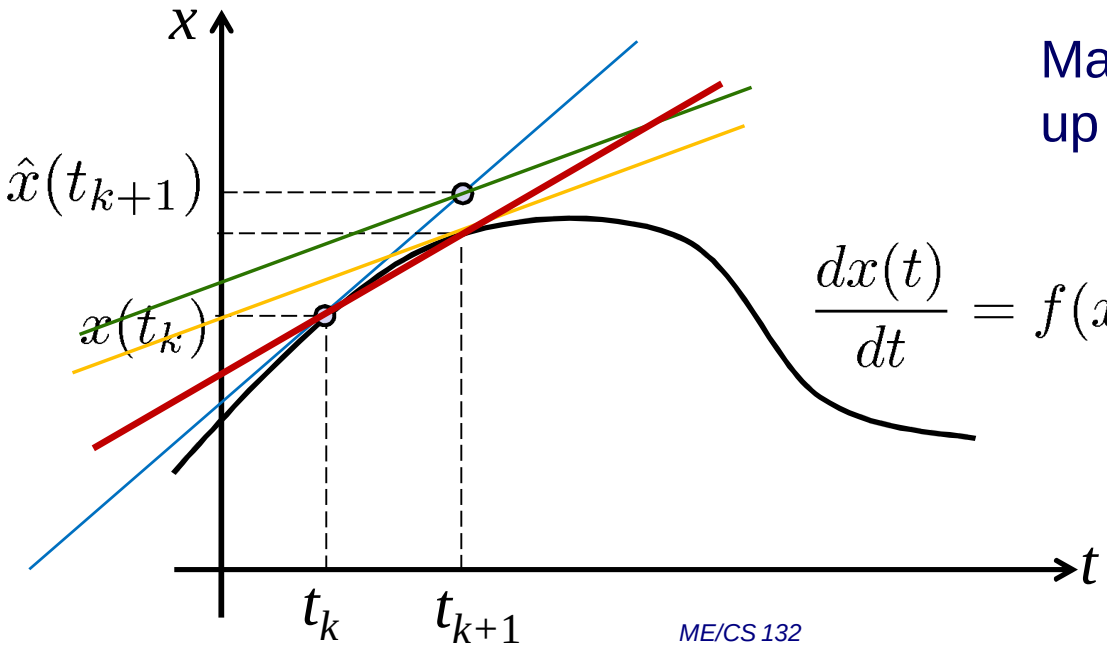
[LaValle's Book, Ch 14.3.2]

- What is it trying to do?
  - Try to approximate the slope by **sampling several points** and **averaging them** with proper weights

2nd order Runge-Kutta

- Refinement of the Euler method: use the slope at t_k and t_{k+1}

$$
\begin{cases}
w_1 = f(x(t_k), u(t_k)) & \leftarrow \textbf{Slope at } t_k \\
\hat{x}(t_{k+1}) = x(t_k) + w_1 \Delta t & \leftarrow \text{Estimate } x(t_{k+1}) \text{ using Euler method} \\
w_2 = f(\hat{x}(t_{k+1}), u(t_{k+1})) & \leftarrow \textbf{Estimated slope at } t_{k+1} \\
x(t_{k+1}) = x(t_k) + \dfrac{w_1 + w_2}{2} \Delta t & \leftarrow \textbf{Average of two slopes}
\end{cases}
$$



Matches the Taylor series up to two terms

$$\frac{dx(t)}{dt} = f(x(t), u(t))$$

4th order RK

- 2nd order RK:

$$x(t_{k+1}) = x(t_k) + \frac{\Delta t}{2}(w_1 + w_2)$$

$$w_1 = f(x(t_k), u(t_k)) \quad \leftarrow \text{Slope at } t_k$$

$$w_2 = f(x(t_k) + w_1 \Delta t, u(t_{k+1})) \quad \leftarrow \text{Slope at } t_{k+1} \text{ using } w_1 \text{ \& Euler}$$

- 4th order RK:

$$x(t_{k+1}) = x(t_k) + \frac{\Delta t}{6}(w_1 + 2w_2 + 2w_3 + w_4)$$

Larger weights on the slopes at mid points

$$w_1 = f(x(t_k), u(t_k)) \quad \leftarrow \text{Slope at } t_k$$

$$w_2 = f\left(x(t_k) + \frac{1}{2} w_1 \Delta t, u(t_k + \frac{1}{2}\Delta t)\right) \quad \leftarrow \text{Slope at } t_k + \tfrac{1}{2}\Delta t \text{ using } w_1 \text{ \& Euler}$$

$$w_3 = f\left(x(t_k) + \frac{1}{2} w_2 \Delta t, u(t_k + \frac{1}{2}\Delta t)\right) \quad \leftarrow \text{Slope at } t_k + \tfrac{1}{2}\Delta t \text{ using } w_2 \text{ \& Euler}$$

$$w_4 = f\left(x(t_k) + w_3 \Delta t, u(t_k + \Delta t)\right) \quad \leftarrow \text{Slope at } t_{k+1} \text{ using } w_3 \text{ \& Euler}$$

Current Research

- DARTS group at JPL **http://dartslab.jpl.nasa.gov/**
  - EDL simulation (Monte-Carlo simulation → risk analysis)
  - SOA (Spatial Operator Algebra): multi-body dynamics
- Planning using forward simulation
  - Readily incorporate nonlinear dynamics/constraints
  - Planning with SLAM (simulated observations)
- GPU
  - Can run hundreds of simulations in parallel
  - Crowd simulation
  - Millions of particles interacting with each other (e.g., sand, molecular dynamics)