Lecture Summary

Innhold

15.01.2016: Model Types 2
21.01.2016: Energy, Passivity, Storage 3
22.01.2016: Passivity, Positive Real, Storage 5
28.01.2016: Modeling, Simulation, Euler's Method 8
29.01.2016: Explicit Runge-Kutta 12
04.02.2016: Electromechanical, Network Modeling, Passivity 20
05.02.2016: Explicit/Implicit Runge-Kutta, Stiff Systems 23
11.02.2016: Stability of Runge-Kutta, Pade Approximations 33
12.02.2016: Stability, Error Control, Interpolation 44
18.02.2016: Hydralic Motor, Sylinder, Valve, Four-Way Valve, Wave, Impedance 59
19.02.2016: Friksjon 66
03.03.2016: Rigid Body Kinematics, Kinetics, Dynamics 71
04.03.2016: Rigid Body Kinematics, Rotation Matrix, Euler Angles, Angle-Axis 74
31.03.2016: Rigid Body Kinematics, Rotation Matrices, Euler Angles, Angular Velocity 80
01.04.2016: Euler Parameters, Rotations, Angular Velocity, Differentiation of Vectors 84
07.04.2016: Newton-Euler EoM, Angular Momentum, Inertia Dyadic 91
08.04.2016: Newton-Euler EoM, Inertia Matrix 96
14.04.2016: Lagrang vs Newton-Euler 100
15.04.2016: Lagrang EoM 102
21.04.2016: Mass Balance, Lumped vs Distributed 106
28.04.2016: Momentum Balance, Energy Balance, Differential Balance 115

Lecture 2:

- About modeling and simulation (F1)
- Why&what do control engineers need to model?
  - Examples (mostly examples I have worked with)
- Model types (E1.1-1.3,E2.1-2.2)
  - State space models, transfer functions
  - Linear models, nonlinear models

Kinds of mathematical models

- Static vs dynamic
  - Does the model involve time?
- Continuous vs discrete

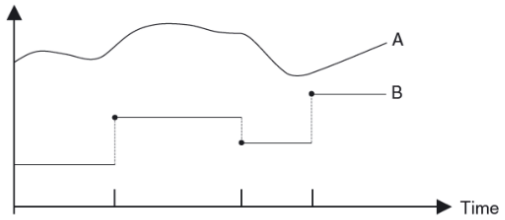


Figure 1.4 Discrete-time system B changes values only at certain points in time, whereas continuous-time systems like A evolve values continuously.

- Lumped vs distributed

$$\frac{\mathrm{d}}{\mathrm{d}t}T(t) = -aT(t)$$

$$\frac{\partial}{\partial t}T(t,x) = -\frac{\partial^2}{\partial x^2}T(t,x)$$

- Stochastic vs deterministic

- Empirical vs 'first principles'
  - Build models from measurement data, or derive from laws of physics?
  - Or both: "Grey box"

Lecture 3: Energy functions and passivity

Using ”energy” as a concept for characterizing system behavior

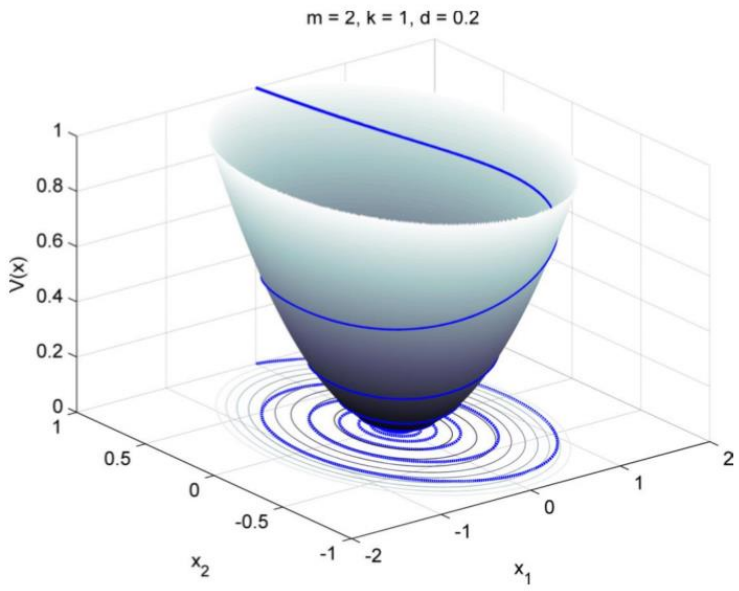
- Energy functions (aka Lyapunov functions)
  - If the ”internal energy” of a system decreases, the system is stable
  - ”Introvert” (not concerned with surroundings)
- Passivity
  - Does a system produce ”energy” to its surroundings?
  - ”Extrovert” (mainly conserned with surroundings, via inputs and outputs)
- The above concepts are connected via storage functions (next time)

Book: E2.3, E2.4

Mass-spring-damper

$$m\ddot{x} + d\dot{x} + kx = 0$$

$$V(x) = \frac{m}{2}\dot{x}^2 + \frac{k}{2}x^2$$



Why learn about passivity? Preview...

- Say you have several systems (or models), and you want to interconnect them
  - For instance, a process and a controller, or a motor and a load, or two buffer tanks in series, ...
  - Will the interconnection be stable?

- Bad news: The interconnection of stable systems is not necessarily stable

- **Good news: The interconnection of passive systems is passive (and therefore stable)!**

Lecture 4: Passivity

Passivity (E2.4)

- Brief recap of passivity
- Positive Real (PR) transfer functions
- Passivity and storage functions

Recap: Energy functions and passivity

Using ”energy” as a concept for characterizing system behavior

- Energy functions (aka Lyapunov functions)
  - If the ”internal energy” of a system decreases, the system is stable
  - ”Introvert” (not concerned with surroundings)
- Passivity
  - Does a system produce ”energy” to its surroundings?
  - ”Extrovert” (mainly conserned with surroundings, via inputs and outputs)
- The above concepts are connected via storage functions

Passivity

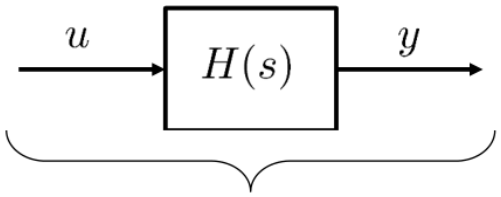
- A system with input *u* and output *y* is passive if

$$\int_0^t y(\tau)u(\tau)d\tau \geq -E_0$$

for all $t \geq 0$, for all input trajectories.

- If the product yu has power as unit, then if (for all *u*)
  - $\int_0^t y(\tau)u(\tau)d\tau \geq 0$: Energy is absorbed within the system, nothing delivered to the outside
  - $\int_0^t y(\tau)u(\tau)d\tau \geq -E_0$: Some energy can be delivered to the outside, limited (typically) by the initial energy in the system.
  - $\int_0^t y(\tau)u(\tau)d\tau \to -\infty$: There is an inexhaustible energy source in the system. Not passive!

Positive real transfer functions



Passive

$\Leftrightarrow$ $H(s)$ is positive real

Definition: The transfer function $H(s)$ (rational or irrational) is positive real if

1. $H(s)$ analytic in Re[s]>0.
2. $H(s)$ is real for all positive and real s.
3. Re[$H(s)$]$\geq$ 0 for all Re[s]>0.

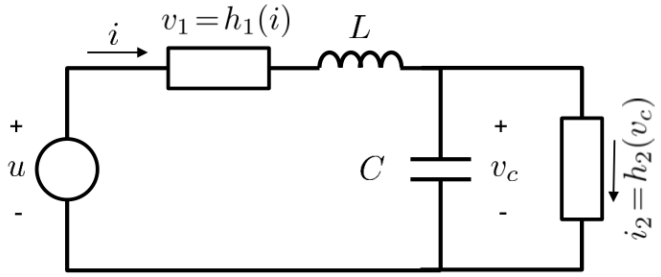
Check rational TFs for PRness

Theorem: A rational, proper transfer function $H(s)$ is positive real (and hence passive) if and only if

1. $H(s)$ has no poles in $\mathrm{Re}[s] > 0$.
2. $\mathrm{Re}[\mathrm{H}(j\omega)] \geq 0$ for all $\omega \in [-\infty, \infty]$ such that $j\omega$ is not a pole of $H(s)$.
3. If $j\omega_0$ is a pole of $H(s)$, then it is a simple pole, and the residual in $s = j\omega_0$ is real and greater than zero, that is,

$$\operatorname*{Res}_{s=j\omega_0} H(s) = \lim_{s \to j\omega_0} (s - j\omega_0) H(j\omega) > 0.$$

Example storage functions



- States: $x_1 = i, \quad x_2 = v_c$
- Model (Kirchoff's laws):

$$L\dot{x}_1 = u - h_1(x_1) - x_2$$
$$C\dot{x}_2 = x_1 - h_2(x_2)$$

- Output&input: $y = i, \quad u = u$
- Nonlinear resistors fulfilling

$$x_i h_i(x_i) \geq 0$$

- Storage (energy) function:

$$V(\mathbf{x}) = \frac{1}{2}Lx_1^2 + \frac{1}{2}Cx_2^2$$

- Differentiate:

$$\begin{aligned}\dot{V} &= Lx_1\dot{x}_1 + Cx_2\dot{x}_2 \\ &= x_1(u - h_1(x_1) - x_2) + x_2(x_1 - h_2(x_2)) \\ &= yu - x_1h_1(x_1) - x_2h(x_2)\end{aligned}$$

- Passive!

Lecture 5:

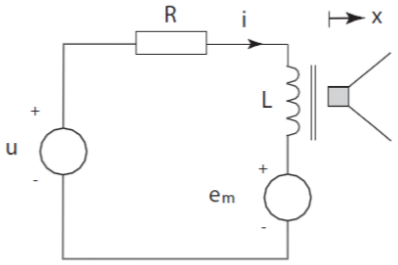
Two different classes of modeling tools

Modeling of complex systems – loudspeaker example (F4)

Introduction to simulation of continuous-time models (E14.2, E14.3.1)

- Notation, computation errors
- Test system and stability function
- Euler's method

Modeling of subsystems



Electrical subsystem:

$$u - Ri - L\frac{\mathrm{d}i}{\mathrm{d}t} - e_m = 0$$

Electromagnetic subsystem:

$$e_m = Blv$$
$$F_m = Bil$$

Mechanical subsystem:

$$m\frac{\mathrm{d}^2x}{\mathrm{d}t^2} = F_m - dv - kx$$

Linearization (14.2.4)

- System $\dot{y} = f(y,t), \quad y = (y_1, \ldots, y_d)^{\mathsf{T}}$
- Linearize around operating point y^*: $\Delta\dot{y} = J\Delta y, \quad J = \left.\frac{\partial f}{\partial y}\right|_{y=y^*}$
- Diagonalize: $Jm_i = \lambda_i m_i, \quad$ where $\begin{cases} m_i : \text{eigenvectors of } J \\ \lambda_i : \text{eigenvalues of } J \end{cases}$
- Define $q = M^{-1}\Delta y$:

$$\dot{q} = M^{-1}J\Delta y = M^{-1}JMq = \Lambda q, \qquad \Lambda = \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_d \end{pmatrix}$$

- That is, $\dot{q}_i = \lambda_i q_i$ from which we can find $\Delta y(t) = Mq = \sum_{i=1}^{d} q_i(t) m_i$

We can study properties of a method used to simulate the system $\Delta\dot{y} = J\Delta y$, by study properties of the method for the systems $\dot{q}_i = \lambda_i q_i, \quad i = 1, \ldots, d$.

Example linearization

- System:

$$\begin{aligned} \dot{y}_1 &= y_2 \\ \dot{y}_2 &= -y_1^3 - cy_2 \end{aligned}$$

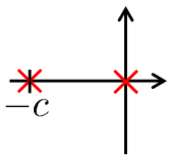
Linearization about $(y_1^*, y_2^*)^{\mathsf{T}}$:

$$\begin{pmatrix} \Delta\dot{y}_1 \\ \Delta\dot{y}_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -3\left(y_1^*\right)^2 & -c \end{pmatrix} \begin{pmatrix} \Delta y_1 \\ \Delta y_2 \end{pmatrix}$$

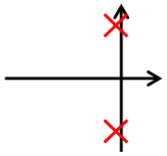
- Eigenvalues:

$$\lambda^2 + c\lambda + 3\left(y_1^*\right) = 0 \qquad \lambda_{1,2} = -\frac{c}{2} \pm \sqrt{\left(\frac{c}{2}\right)^2 - 3\left(y_1^*\right)^2}$$

$y_1^* = 0: \quad \lambda_1 = 0, \ \lambda_2 = -c$



$y_1^* = \text{large}: \quad \lambda_{1,2} \to \pm j\omega_0$



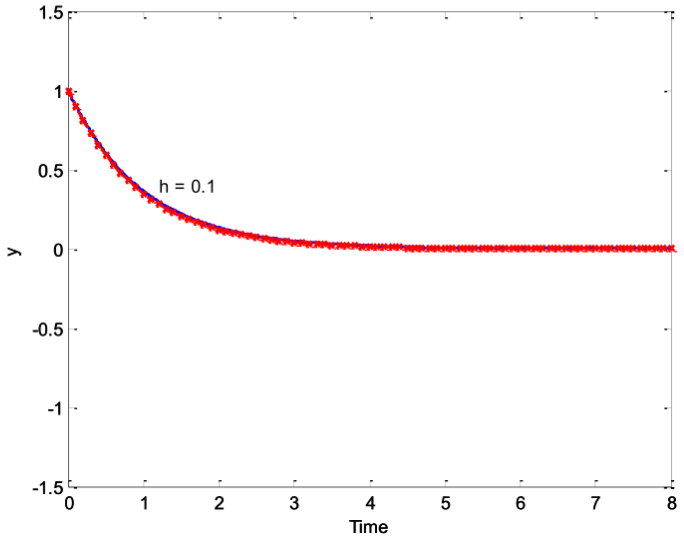
Example Euler's method

ODE: $\dot{y} = -y, \quad y(0) = 1$

Euler simulation: $y_{n+1} = y_n + h\,(-y_n), \quad y_0 = 1$

Example, h = 0.1:

n	t_n	y_n
0	0	1
1	0.1	
2	0.2	
3	0.3	
4	0.4	
…	…	…



Order (accuracy)

- Given IVP:

$$\dot{y} = f(y,t), \quad y(0) = y_0$$

- One-step method:

$$y_{n+1} = y_n + h\phi(y_n, t_n), \quad h = t_{n+1} - t_n$$

- If we can show that

$$y_{n+1} = y_n + hf(y_n,t) + \frac{h^2}{2}\frac{\mathrm{d}f(y_n,t)}{\mathrm{d}t} + \ldots + \frac{h^p}{p!}\frac{\mathrm{d}^{p-1}f(y_n,t)}{\mathrm{d}t^{p-1}} + O(h^{p+1})$$

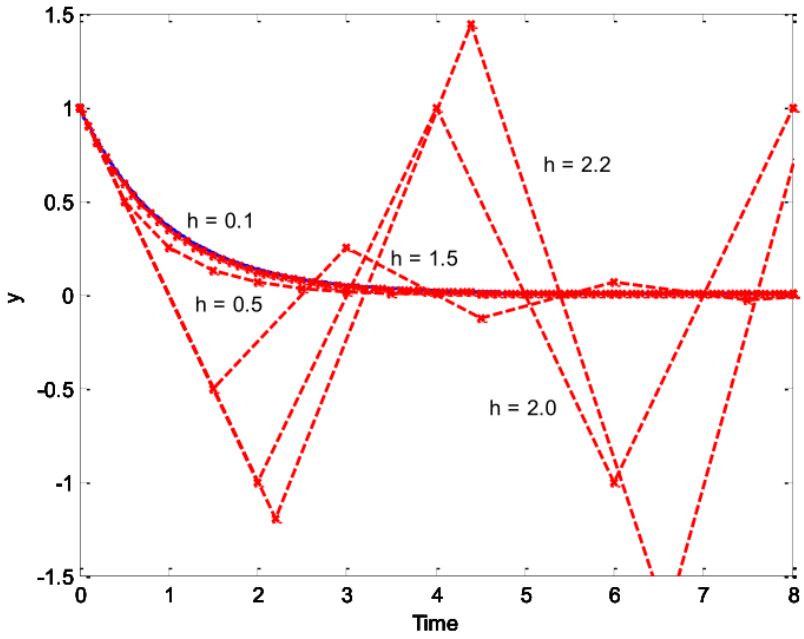
- Then:
  - Local error is $O(h^{p+1})$
  - Method is order p

Example Euler's method

ODE: $\dot{y} = -y, \quad y(0) = 1$

Euler simulation: $y_{n+1} = y_n + h\,(-y_n), \quad y_0 = 1$

Stability: $|R(h\lambda)| = |1 - h| \leq 1 \Rightarrow 0 \leq h \leq 2$



Lecture 6: Explicit Runge-Kutta Methods

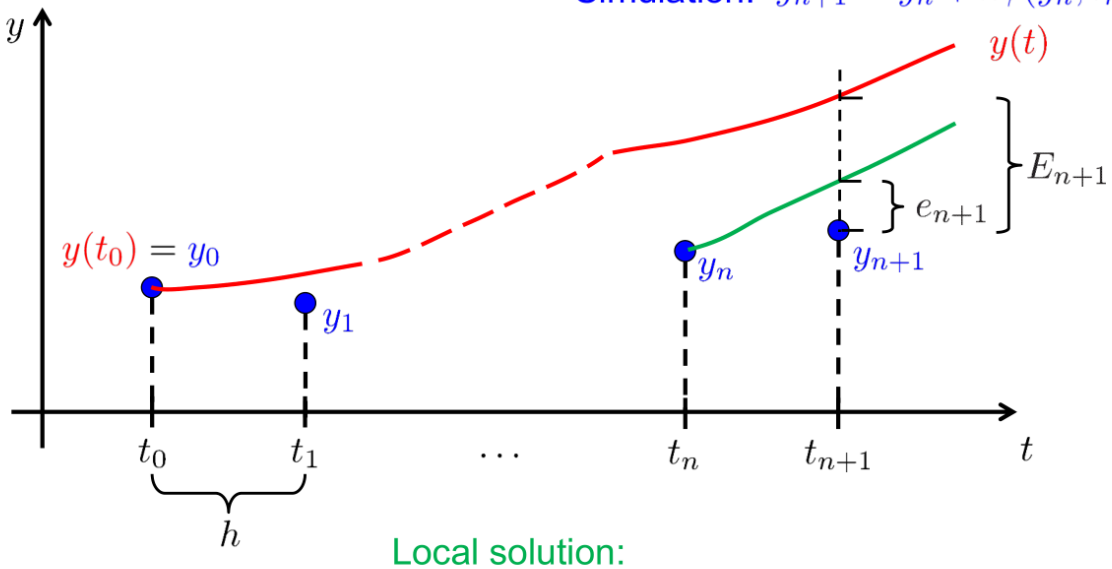
Explicit Runge-Kutta (ERK) methods, and their order and stability

Book: 14.3, 14.4

Recap: Notation

IVP: $\dot{y} = f(y,t), \quad y(t_0) = y_0$

Simulation: $y_{n+1} = y_n + h\phi(y_n, t_n)$

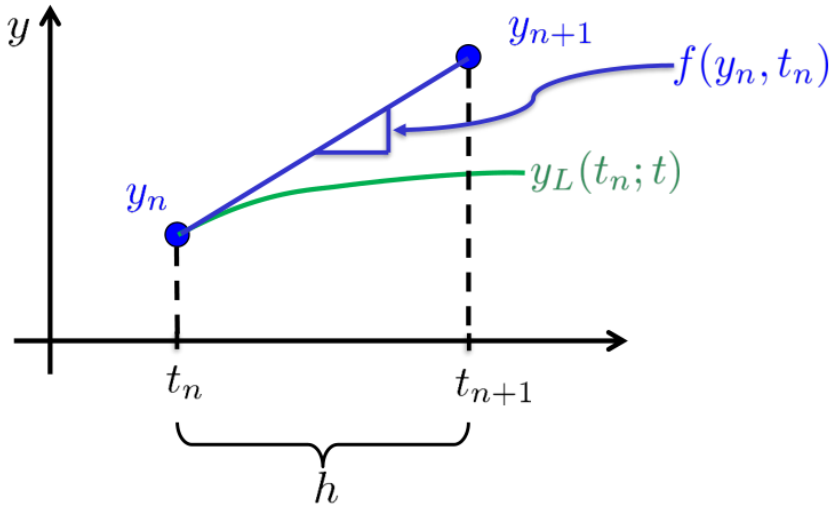


Local solution:

$$\dot{y}_L(t_n;t) = f(y_L(t_n;t),t), \quad y_L(t_n;t_n) = y_n$$

- Local error: $e_{n+1} = y_{n+1} - y_L(t_n;t_{n+1})$
- Global error: $E_{n+1} = y_{n+1} - y(t_{n+1})$
- If local error is $O(h^{p+1})$ then we say method is of order p

Simplest method: Euler



- Slope:

$$\frac{y_{n+1} - y_n}{h} = f(y_n, t_n)$$

- Euler's method:

$$y_{n+1} = y_n + hf(y_n, t_n)$$

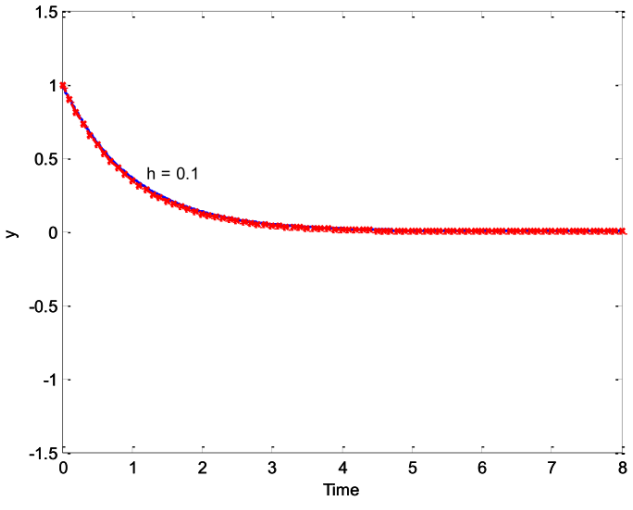
Example Euler's method

ODE: $\dot{y} = -y, \quad y(0) = 1$

Euler simulation: $y_{n+1} = y_n + h\,(-y_n)\,, \quad y_0 = 1$

Example, h = 0.1:

n	t_n	y_n
0	0	1
1	0.1	
2	0.2	
3	0.3	
4	0.4	
…	…	…



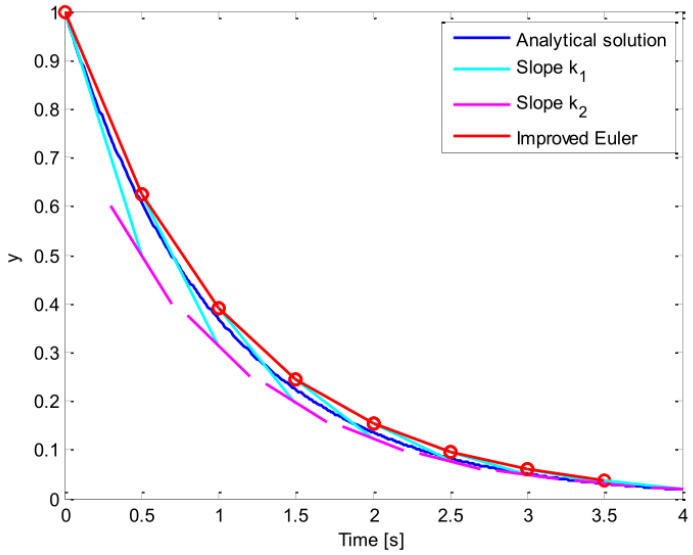
Improved Euler illustration

$$\dot{y} = -y, \quad y(0) = 1$$

Improved Euler:

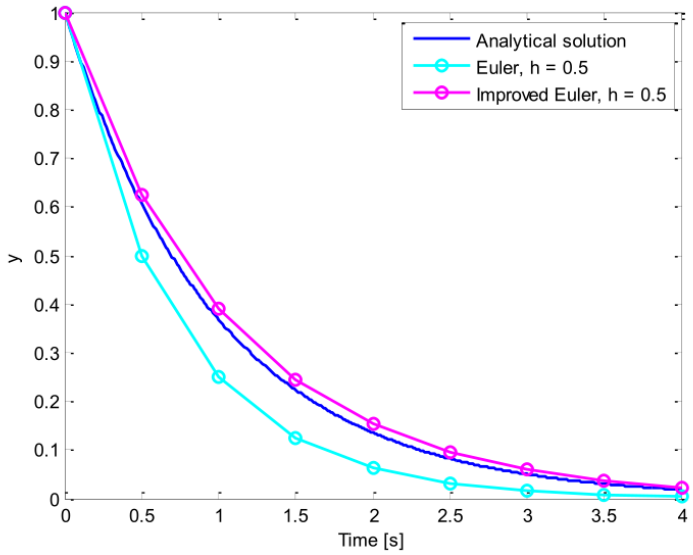
$$k_1 = f(y_n), \quad k_2 = f(y_n + hk_1)$$

$$y_{n+1} = y_n + \frac{h}{2}(k_1 + k_2)$$



Improved Euler vs Euler

$$\dot{y} = -y, \quad y(0) = 1$$



Accuracy and stability

- Lots of different methods, with different complexity. How to quantify their behaviour?

- Two aspects are important: accuracy and numerical stability.
  - Accuracy: How does the local error vary with step-size?
  - Numerical stability: How to avoid that the simulation diverges?

- What decides accuracy and numerical stability?
  - Accuracy: Method and choice of step-size
  - Stability: Method, system eigenvalues, and choice of step-size

- Why are we interested in both accuracy and numerical stability?
  - We always need stability, but stability not enough: Many stable methods are not very accurate!

Recap: Order (accuracy)

- Given IVP:

$$\dot{y} = f(y,t), \quad y(0) = y_0$$

- One-step method:

$$y_{n+1} = y_n + h\phi(y_n, t_n), \quad h = t_{n+1} - t_n$$

- If we can show that

$$y_{n+1} = y_n + hf(y_n,t) + \frac{h^2}{2}\frac{\mathrm{d}f(y_n,t)}{\mathrm{d}t} + \ldots + \frac{h^p}{p!}\frac{\mathrm{d}^{p-1}f(y_n,t)}{\mathrm{d}t^{p-1}} + O(h^{p+1})$$

- Then:
  - Local error is $O(h^{p+1})$
  - Method is order p

Recap: Test system, stability function

- One step method:

$$y_{n+1} = y_n + h\phi(y_n, t_n)$$

- Apply it to scalar test system:

$$\dot{y} = \lambda y$$

- We get:

$$y_{n+1} = R(h\lambda)y_n$$

  where $R(h\lambda)$ is stability function

- The method is stable (for test system!) if

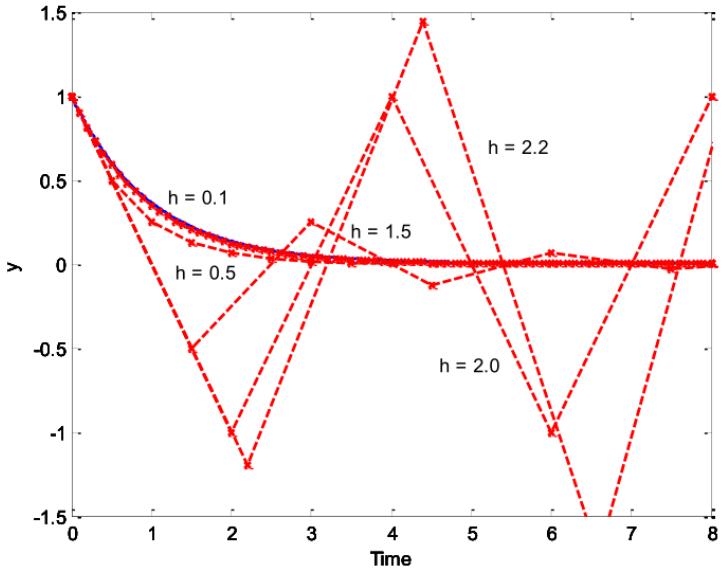
$$|R(h\lambda)| \leq 1$$

Example Euler's method

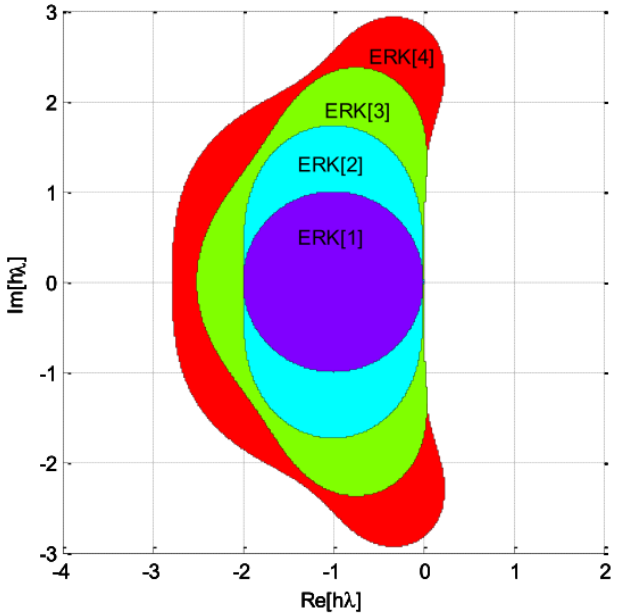
ODE: $\dot{y} = -y, \quad y(0) = 1$

Euler simulation: $y_{n+1} = y_n + h\,(-y_n)\,, \quad y_0 = 1$

Stability: $|R(h\lambda)| = |1 - h| \leq 1 \Rightarrow 0 \leq h \leq 2$



Stability regions for ERK methods



Fact: For $1 \leq \sigma \leq 4$, one can devise ERK methods with order $p = \sigma$

- For these methods, per definition

$$y_{n+1} = y_n + hf(y_n, t_n) + \ldots + \frac{h^p}{p!}\frac{\mathrm{d}^{p-1}}{\mathrm{d}t^{p-1}} f(y_n, t_n) + O(h^{p+1})$$

- For test system $\dot{y} = \lambda y$,

$$\begin{aligned} y_{n+1} &= y_n + h\lambda y_n + \ldots + \frac{h^p \lambda^p}{p!} y_n + O(h^{p+1}) \\ &= \left(1 + h\lambda + \ldots + \frac{h^p \lambda^p}{p!}\right) y_n + O(h^{p+1}) \end{aligned}$$

- From before, we know that $y_{n+1} = R_E(h\lambda) y_n$, where $R_E(h\lambda)$ is polynomial of degree less than or equal to $\sigma = p$

That is: For ERK methods with order $p = \sigma$, for $\sigma \leq 4$:

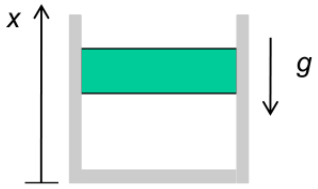
$$R_E(h\lambda) = 1 + h\lambda + \ldots + \frac{h^p \lambda^p}{p!}$$

Case: Pneumatic spring

- Model from Newton's 2nd law:

$$\ddot{x} + g(1 - x^{-\kappa}) = 0$$

"mass-spring-damper with nonlinear spring and no damping"



- On states-space form $\dot{y} = f(y,t)$

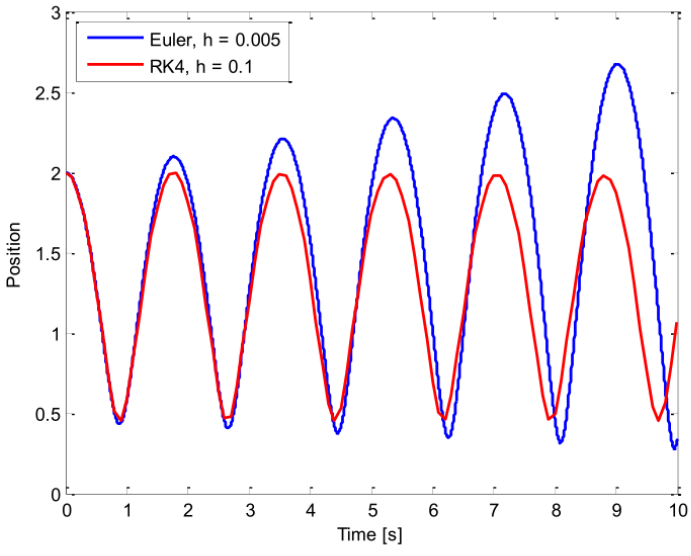
$$\begin{pmatrix} \dot{y}_1 \\ \dot{y}_2 \end{pmatrix} = \begin{pmatrix} y_2 \\ -g(1 - y_1^{-\kappa}) \end{pmatrix}$$

- Linearization about equilibrium:

$$\frac{\partial f}{\partial y} = \begin{pmatrix} 0 & 1 \\ -g\kappa & 0 \end{pmatrix}, \qquad \lambda_{1,2} = \pm j\omega_0, \quad \omega_0 = \sqrt{g\kappa} \approx 3.7$$

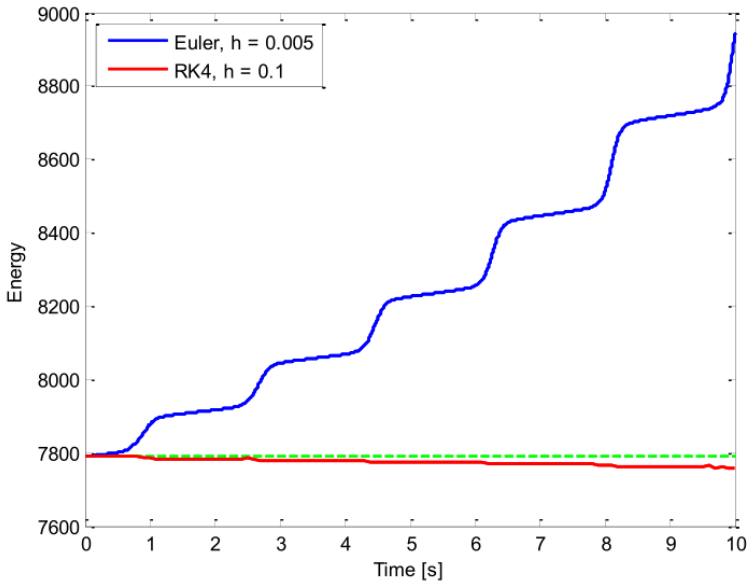
Simulation

Euler: 2000 function evaluations
RK4: 400 function evaluations



- Stability, RK4
  - Theoretical: $\omega_0 h \approx 2.83 \quad \Rightarrow \quad h \approx 0.76$
  - Practically: $h \approx 0.52$

Accuracy: Energy should be constant



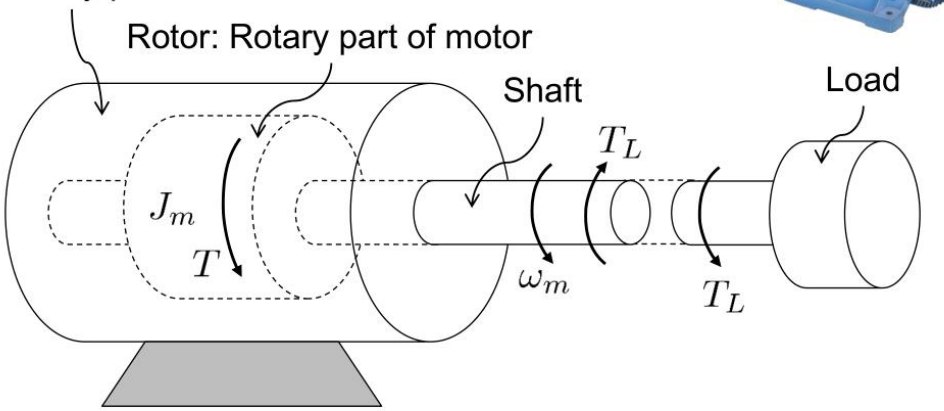
Lecture 7: Electromechanical systems

- Electrical motors
- DC motor with constant field
- Some network modeling, passivity, ...

Book: 3.2, 3.3

Motors

Stator: Stationary part of motor



- Equation of motion for motor shaft:

$$J_m\dot{\omega}_m = T - T_L$$

where

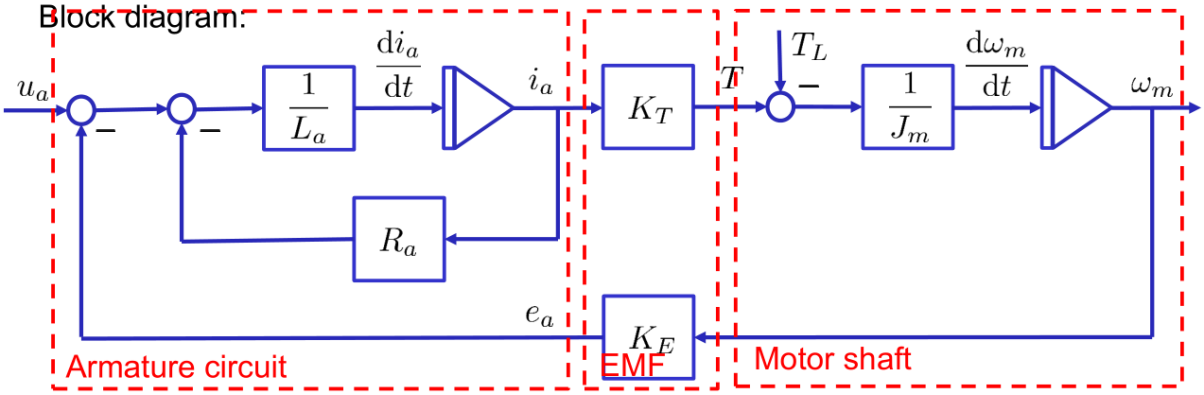
- T : Motor torque (set up by some device, e.g. DC motor)
- T_L : Load torque
- J_m : Moment of inertia for rotor and shaft
- ω_m: Angular velocity/motor speed [rad/s, or rev./min]

Network modeling of DC-motor:

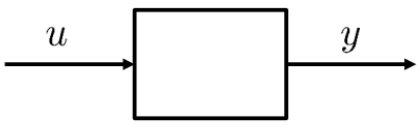
$$L_a \frac{\mathrm{d}i_a}{\mathrm{d}t} = -R_a i_a - e_a + u_a \qquad T = K_T i_a \qquad e_a = K_E \omega_m \qquad J_m \frac{\mathrm{d}\omega_m}{\mathrm{d}t} = T - T_L$$

Signal flow modeling of DC-motor:

Block diagram:



Passivity



- A system with input u and output y is passive if

$$\int_0^t y(\tau)u(\tau)d\tau \geq -E_0$$

for all $t \geq 0$, for all input trajectories.

- If the product yu has power as unit, then if
  - $\int_0^t y(\tau)u(\tau)d\tau \geq 0$: Energy is absorbed within the system, nothing delivered to the outside
  - $\int_0^t y(\tau)u(\tau)d\tau \geq -E_0$: Some energy can be delivered to the outside, limited (typically) by the initial energy in the system.
  - $\int_0^t y(\tau)u(\tau)d\tau \to -\infty$: There is an inexhaustible energy source in the system. Not passive!

Recap: Explicit Runge-Kutta (ERK) methods

- IVP: $\dot{y} = f(y,t), \quad y(0) = y_0$
- One-step methods: $y_{n+1} = y_n + h\phi(y_n, t_n), \quad h = t_{n+1} - t_n$
- ERK:

$$
\begin{aligned}
k_1 &= f(y_n, t_n) \\
k_2 &= f(y_n + ha_{21}k_1, t_n + c_2 h) \\
k_3 &= f(y_n + h(a_{31}k_1 + a_{32}k_2), t_n + c_3 h) \\
&\vdots \\
k_\sigma &= f(y_n + h(a_{\sigma,1}k_1 + a_{\sigma,2}k_2 + \ldots + a_{\sigma,\sigma-1}k_{\sigma-1}), t_n + c_\sigma h) \\
y_{n+1} &= y_n + h(b_1 k_1 + b_2 k_2 + \ldots + b_\sigma k_\sigma)
\end{aligned}
$$

- Butcher array:

$\mathbf{c}$	$\mathbf{A}$
	$\mathbf{b}^\mathsf{T}$

0					
c_2	a_{21}				
c_3	a_{31}	a_{32}			
$\vdots$	$\vdots$	$\vdots$	$\ddots$		
c_σ	$a_{\sigma,1}$	$a_{\sigma,2}$	$\cdots$	$a_{\sigma,\sigma-1}$	
	b_1	b_2	$\ldots$	$b_{\sigma-1}$	b_σ

Lecture 8: Implicit Runge-Kutta Methods

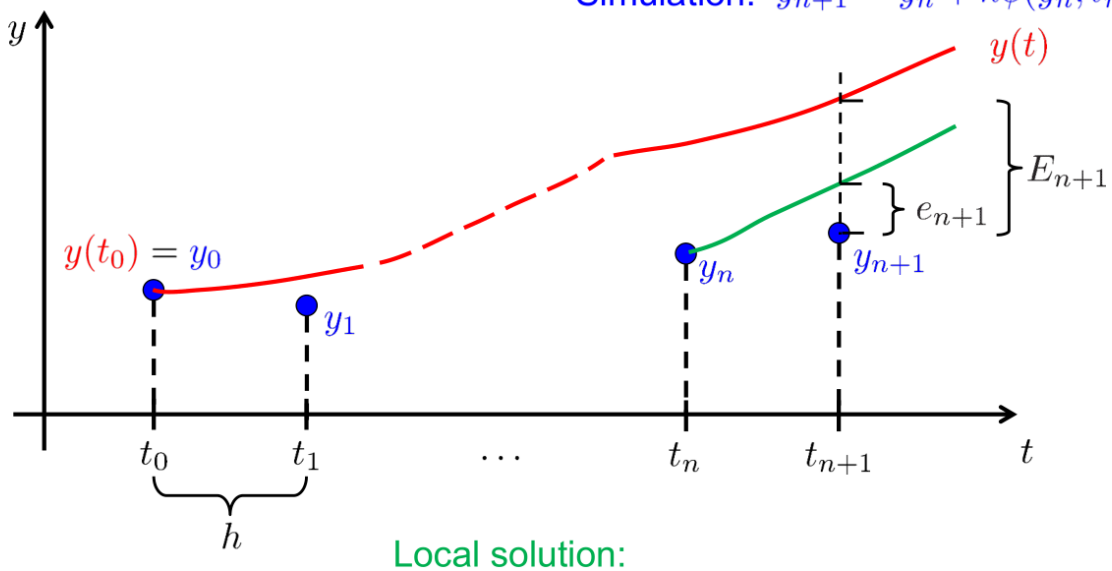
- Recap Explicit Runge-Kutta (ERK) methods
- Stiff systems
- Implicit Runge-Kutta (IRK) ODE solvers

Book: 14.5 (+ 14.8.1)

Notation

IVP: $\dot{y} = f(y,t), \quad y(t_0) = y_0$

Simulation: $y_{n+1} = y_n + h\phi(y_n, t_n)$



Local solution:

$$\dot{y}_L(t_n;t) = f(y_L(t_n;t),t), \quad y_L(t_n;t_n) = y_n$$

- Local error: $e_{n+1} = y_{n+1} - y_L(t_n;t_{n+1})$
- Global error: $E_{n+1} = y_{n+1} - y(t_{n+1})$
- If local error is $O(h^{p+1})$ then we say method is of order p

Recap: Explicit Runge-Kutta (ERK) methods

- IVP: $\dot{y} = f(y,t), \quad y(0) = y_0$
- One-step methods: $y_{n+1} = y_n + h\phi(y_n, t_n), \quad h = t_{n+1} - t_n$
- ERK:

$$\begin{aligned} k_1 &= f(y_n, t_n) \\ k_2 &= f(y_n + ha_{21}k_1, t_n + c_2 h) \\ k_3 &= f(y_n + h(a_{31}k_1 + a_{32}k_2), t_n + c_3 h) \\ &\vdots \\ k_\sigma &= f(y_n + h(a_{\sigma,1}k_1 + a_{\sigma,2}k_2 + \ldots + a_{\sigma,\sigma-1}k_{\sigma-1}), t_n + c_\sigma h) \\ y_{n+1} &= y_n + h(b_1 k_1 + b_2 k_2 + \ldots + b_\sigma k_\sigma) \end{aligned}$$

- Butcher array:

$\mathbf{c}$	$\mathbf{A}$
	$\mathbf{b}^\mathsf{T}$

0					
c_2	a_{21}				
c_3	a_{31}	a_{32}			
$\vdots$	$\vdots$	$\vdots$	$\ddots$		
c_σ	$a_{\sigma,1}$	$a_{\sigma,2}$	$\cdots$	$a_{\sigma,\sigma-1}$	
	b_1	b_2	$\ldots$	$b_{\sigma-1}$	b_σ

Recap: Test system, stability function

- One step method:

$$y_{n+1} = y_n + h\phi(y_n, t_n)$$

- Apply it to scalar test system:

$$\dot{y} = \lambda y$$

- We get:

$$y_{n+1} = R(h\lambda)y_n$$

where $R(h\lambda)$ is stability function

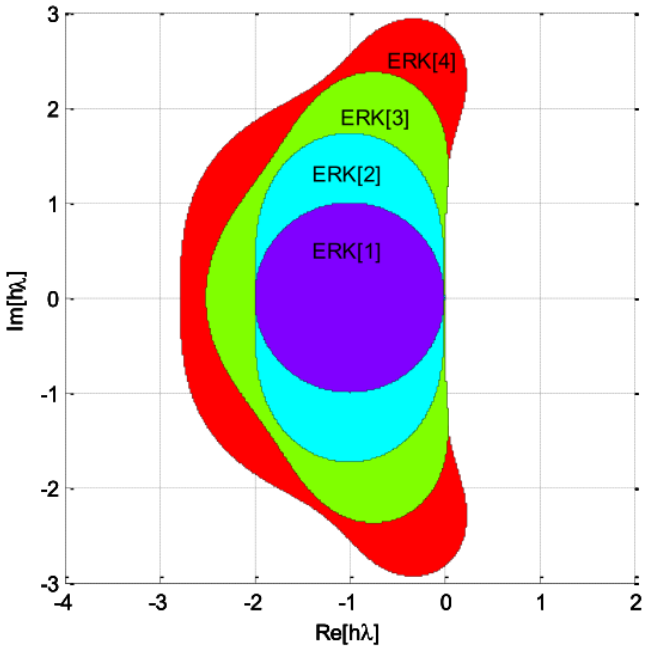
- The method is stable (for test system!) if

$$|R(h\lambda)| \leq 1$$

Stability function for RK-methods

- Two alternative, equivalent expressions can be derived:
  - Either
$$R(h\lambda) = 1 + h\lambda \mathbf{b}^{\mathsf{T}} \left(\mathbf{I} - h\lambda \mathbf{A}\right)^{-1} \mathbf{1}$$
  - or
$$R(h\lambda) = \frac{\det\left[\mathbf{I} - h\lambda\left(\mathbf{A} - \mathbf{1}\mathbf{b}^{\mathsf{T}}\right)\right]}{\det\left[\mathbf{I} - h\lambda\mathbf{A}\right]}$$
- The latter can be simplified for ERK (since A is lower triangular):
$$R_E(h\lambda) = \det\left[\mathbf{I} - h\lambda\left(\mathbf{A} - \mathbf{1}\mathbf{b}^{\mathsf{T}}\right)\right]$$
- Two observations can be made
  1. $|R_E(h\lambda)|$ will tend to infinity when λ goes to infinity.
  2. $R_E(h\lambda)$ is a polynomial in $h\lambda$ of order less than or equal to σ.

Stability regions for ERK methods



Order and stages

- For number of stages less than or equal to 4 ($\sigma \leq 4$) it is possible to develop ERK methods (find combinations of a_{ij}, c_i, b_i) with order equal to number of stages ($p = \sigma$). These are the ones that are used.

- These methods have stability function of the type

$$R_E(h\lambda) = 1 + h\lambda + \frac{(h\lambda)^2}{2} + ... + \frac{(h\lambda)^p}{p!}$$

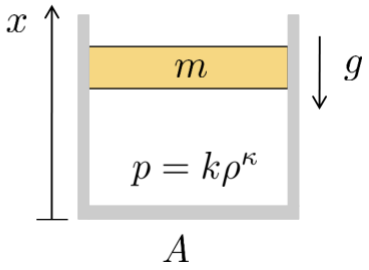
- To obtain higher order, requires more stages:
  - Order 5 requires 6 stages
  - Order 6 requires 7 stages
  - Order 7 requires 9 stages
  - Order 8 requires 11 stages
  - …

ERK example: Pneumatic spring

- Model from Newton's 2nd law:

$$\ddot{x} + g(1 - x^{-\kappa}) = 0$$

"mass-spring-damper with nonlinear spring"



- On state-space form $\dot{y} = f(y,t)$

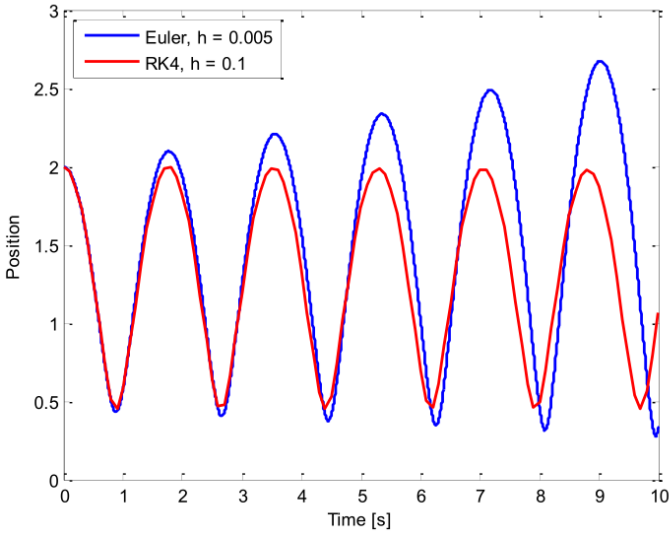
$$\begin{pmatrix} \dot{y}_1 \\ \dot{y}_2 \end{pmatrix} = \begin{pmatrix} y_2 \\ -g(1 - y_1^{-\kappa}) \end{pmatrix}$$

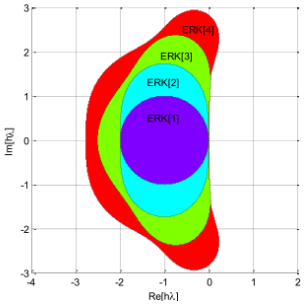
- Linearization about equilibrium:

$$\frac{\partial f}{\partial y} = \begin{pmatrix} 0 & 1 \\ -g\kappa & 0 \end{pmatrix}, \qquad \lambda_{1,2} = \pm j\omega_0, \quad \omega_0 = \sqrt{g\kappa} \approx 3.7$$

Simulation

Euler: 2000 function evaluations
RK4: 400 function evaluations

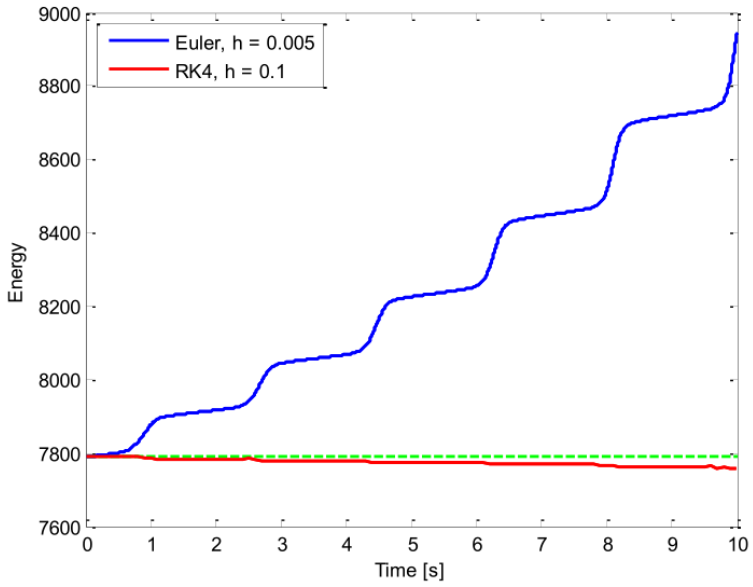




- Stability, RK4
  - Theoretical: $\omega_0 h \approx 2.83 \quad \Rightarrow \quad h \approx 0.76$
  - Practically: $h \approx 0.52$

Pneumatic spring: Accuracy

- Energy should be constant



Recap: Explicit Runge-Kutta (ERK) methods

- IVP: $\dot{y} = f(y,t), \quad y(0) = y_0$
- One-step methods: $y_{n+1} = y_n + h\phi(y_n, t_n), \quad h = t_{n+1} - t_n$
- ERK:

$$
\begin{aligned}
k_1 &= f(y_n, t_n) \\
k_2 &= f(y_n + ha_{21}k_1, t_n + c_2 h) \\
k_3 &= f(y_n + h(a_{31}k_1 + a_{32}k_2), t_n + c_3 h) \\
&\vdots \\
k_\sigma &= f(y_n + h(a_{\sigma,1}k_1 + a_{\sigma,2}k_2 + \ldots + a_{\sigma,\sigma-1}k_{\sigma-1}), t_n + c_\sigma h) \\
y_{n+1} &= y_n + h(b_1 k_1 + b_2 k_2 + \ldots + b_\sigma k_\sigma)
\end{aligned}
$$

- Butcher array:

$\mathbf{c}$	$\mathbf{A}$
	$\mathbf{b}^\mathsf{T}$

0					
c_2	a_{21}				
c_3	a_{31}	a_{32}			
$\vdots$	$\vdots$	$\vdots$	$\ddots$		
c_σ	$a_{\sigma,1}$	$a_{\sigma,2}$	$\cdots$	$a_{\sigma,\sigma-1}$	
	b_1	b_2	$\ldots$	$b_{\sigma-1}$	b_σ

Implicit Runge-Kutta (IRK) methods

- IVP: $\dot{y} = f(y,t), \quad y(0) = y_0$
- IRK:

$$
\begin{aligned}
k_1 &= f(y_n + h(a_{1,1}k_1 + a_{1,2}k_2 + \ldots + a_{1,\sigma}k_\sigma), t_n + c_1 h) \\
k_2 &= f(y_n + h(a_{2,1}k_1 + a_{2,2}k_2 + \ldots + a_{2,\sigma}k_\sigma), t_n + c_2 h) \\
k_3 &= f(y_n + h(a_{3,1}k_1 + a_{3,2}k_2 + \ldots + a_{3,\sigma}k_\sigma), t_n + c_3 h) \\
&\vdots \\
k_\sigma &= f(y_n + h(a_{\sigma,1}k_1 + a_{\sigma,2}k_2 + \ldots + a_{\sigma,\sigma}k_\sigma), t_n + c_\sigma h) \\
y_{n+1} &= y_n + h(b_1 k_1 + b_2 k_2 + \ldots + b_\sigma k_\sigma)
\end{aligned}
$$

- Butcher array:

$\mathbf{c}$	$\mathbf{A}$
	$\mathbf{b}^\mathsf{T}$

c_1	a_{11}	a_{12}	$\cdots$	$a_{1,\sigma-1}$	$a_{1,\sigma}$
c_2	a_{21}	a_{22}	$\cdots$	$a_{2,\sigma-1}$	$a_{2,\sigma}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
c_σ	$a_{\sigma,1}$	$a_{\sigma,2}$	$\cdots$	$a_{\sigma,\sigma-1}$	$a_{\sigma,\sigma}$
	b_1	b_2	$\ldots$	$b_{\sigma-1}$	b_σ

Recap: Order (accuracy)

- Given IVP:

$$\dot{y} = f(y,t), \quad y(0) = y_0$$

- One-step method:

$$y_{n+1} = y_n + h\phi(y_n, t_n), \quad h = t_{n+1} - t_n$$

- If you can show that

$$y_{n+1} = y_n + hf(y_n,t) + \frac{h^2}{2}\frac{\mathrm{d}f(y_n,t)}{\mathrm{d}t} + \ldots + \frac{h^p}{p!}\frac{\mathrm{d}^{p-1}f(y_n,t)}{\mathrm{d}t^{p-1}} + O(h^{p+1})$$

- Then:
  - Local error is $O(h^{p+1})$
  - Method is order p

Recap: Test system, stability function

- One step method:

$$y_{n+1} = y_n + h\phi(y_n, t_n)$$

- Apply it to scalar test system:

$$\dot{y} = \lambda y$$

- We get:

$$y_{n+1} = R(h\lambda)y_n$$

  where $R(h\lambda)$ is stability function

- The method is stable (for test system!) if

$$|R(h\lambda)| \leq 1$$

Example: “Lambert’s problem”

- IVP:

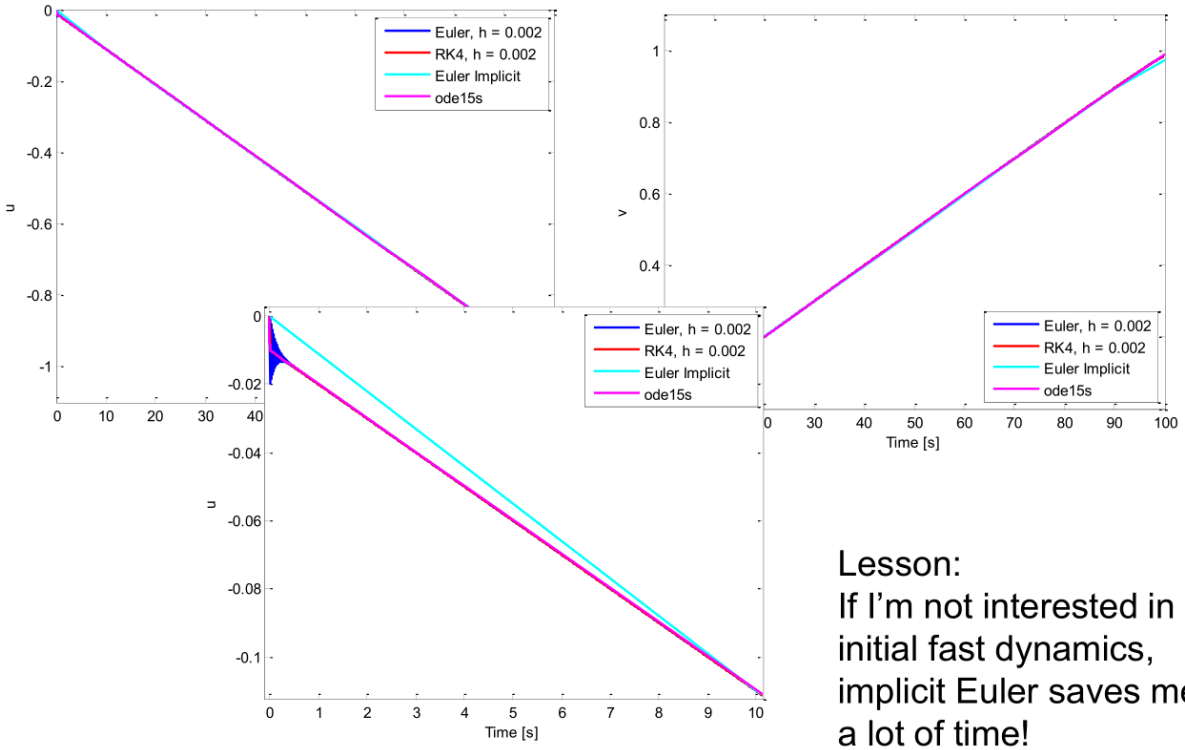
$$\dot{u} = \frac{1}{100} - (\frac{1}{100} + u + v)(1 + (u + 1000)(u + 1)), \quad u(0) = 0$$
$$\dot{v} = \frac{1}{100} - (\frac{1}{100} + u + v)(1 + v^2), \quad v(0) = 0$$

- Task: Simulate from t = 0 s til t = 100 s

- Eigenvalues:

$$(u, v) = (0, 0) \Rightarrow \lambda_1 \approx -1000, \lambda_2 \approx -0.01$$
$$(u, v) = (-.5, .5) \Rightarrow \lambda_1 \approx -500, \lambda_2 \approx -0.03$$
$$(u, v) = (-1, 1) \Rightarrow \lambda_1 \approx -11, \lambda_2 \approx -1$$

Comparisons



Lesson:
If I’m not interested in initial fast dynamics, implicit Euler saves me a lot of time!

Some implicit Runge-Kutta methods

- Implicit Euler:

$$\begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array}$$

Implicit midpoint rule

Trapezoidal rule

- Gauss (or Gauss-Legendre) methods:

Order 2	Order 4	Order 6
$\begin{array}{c\|c} 1/2 & 1/2 \\ \hline & 1 \end{array}$	$\begin{array}{c\|cc} \frac{1}{2}-\frac{\sqrt{3}}{6} & \frac{1}{4} & \frac{1}{4}-\frac{\sqrt{3}}{6} \\ \frac{1}{2}+\frac{\sqrt{3}}{6} & \frac{1}{4}+\frac{\sqrt{3}}{6} & \frac{1}{4} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$	$\begin{array}{c\|ccc} \frac{1}{2}-\frac{\sqrt{15}}{10} & \frac{5}{36} & \frac{2}{9}-\frac{\sqrt{15}}{15} & \frac{5}{36}-\frac{\sqrt{15}}{30} \\ \frac{1}{2} & \frac{5}{36}+\frac{\sqrt{15}}{24} & \frac{2}{9} & \frac{5}{36}-\frac{\sqrt{15}}{24} \\ \frac{1}{2}+\frac{\sqrt{15}}{10} & \frac{5}{36}+\frac{\sqrt{15}}{30} & \frac{2}{9}+\frac{\sqrt{15}}{15} & \frac{5}{36} \\ \hline & \frac{5}{18} & \frac{4}{9} & \frac{5}{18} \end{array}$

- Lobatto methods:

	Order 2	Order 4
Lobatto IIIA	$\begin{array}{c\|cc} 0 & 0 & 0 \\ 1 & 1/2 & 1/2 \\ \hline & 1/2 & 1/2 \end{array}$	$\begin{array}{c\|cccc} 0 & 0 & 0 & 0 \\ 1/2 & 5/24 & 1/3 & -1/24 \\ 1 & 1/6 & 2/3 & 1/6 \\ \hline & 1/6 & 2/3 & 1/6 \end{array}$
Lobatto IIIB	$\begin{array}{c\|cc} 0 & 1/2 & 0 \\ 1 & 1/2 & 0 \\ \hline & 1/2 & 1/2 \end{array}$	$\begin{array}{c\|cccc} 0 & 1/6 & -1/6 & 0 \\ 1/2 & 1/6 & 1/3 & 0 \\ 1 & 1/6 & 5/6 & 0 \\ \hline & 1/6 & 2/3 & 1/6 \end{array}$
Lobatto IIIC	$\begin{array}{c\|cc} 0 & 1/2 & -1/2 \\ 1 & 1/2 & 1/2 \\ \hline & 1/2 & 1/2 \end{array}$	$\begin{array}{c\|cccc} 0 & 1/6 & -1/3 & 1/6 \\ 1/2 & 1/6 & 5/12 & -1/12 \\ 1 & 1/6 & 2/3 & 1/6 \\ \hline & 1/6 & 2/3 & 1/6 \end{array}$

Radau methods:

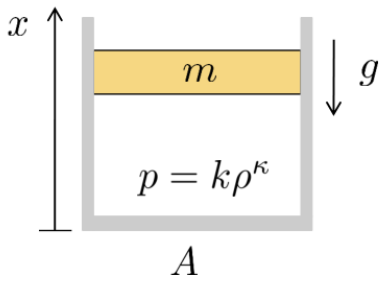
	Order 3	Order 5
Radau IA	$\begin{array}{c\|cc} 0 & 1/4 & -1/4 \\ 2/3 & 1/4 & 5/12 \\ \hline & 1/4 & 3/4 \end{array}$	$\begin{array}{c\|ccc} 0 & \frac{1}{9} & \frac{-1-\sqrt{6}}{18} & \frac{-1+\sqrt{6}}{18} \\ \frac{3}{5}-\frac{\sqrt{6}}{10} & \frac{1}{9} & \frac{11}{45}+\frac{7\sqrt{6}}{360} & \frac{11}{45}-\frac{43\sqrt{6}}{360} \\ \frac{3}{5}+\frac{\sqrt{6}}{10} & \frac{1}{9} & \frac{11}{45}+\frac{43\sqrt{6}}{360} & \frac{11}{45}-\frac{7\sqrt{6}}{360} \\ \hline & \frac{1}{9} & \frac{4}{9}+\frac{\sqrt{6}}{36} & \frac{4}{9}-\frac{\sqrt{6}}{36} \end{array}$
Radau IIA	$\begin{array}{c\|cc} 1/3 & 5/12 & -1/12 \\ 1 & 3/4 & 1/4 \\ \hline & 3/4 & 1/4 \end{array}$	$\begin{array}{c\|ccc} \frac{2}{5}-\frac{\sqrt{6}}{10} & \frac{11}{45}-\frac{7\sqrt{6}}{360} & \frac{37}{225}-\frac{169\sqrt{6}}{1800} & -\frac{2}{225}+\frac{\sqrt{6}}{75} \\ \frac{2}{5}+\frac{\sqrt{6}}{10} & \frac{37}{225}+\frac{169\sqrt{6}}{1800} & \frac{11}{45}+\frac{7\sqrt{6}}{360} & -\frac{2}{225}-\frac{\sqrt{6}}{75} \\ 1 & \frac{4}{9}-\frac{\sqrt{6}}{36} & \frac{4}{9}+\frac{\sqrt{6}}{36} & \frac{1}{9} \\ \hline & \frac{4}{9}-\frac{\sqrt{6}}{36} & \frac{4}{9}+\frac{\sqrt{6}}{36} & \frac{1}{9} \end{array}$

Pneumatic spring example, again (preview)

- Model from Newton's 2nd law:

$$\ddot{x} + g(1 - x^{-\kappa}) = 0$$

"mass-spring-damper with nonlinear spring"



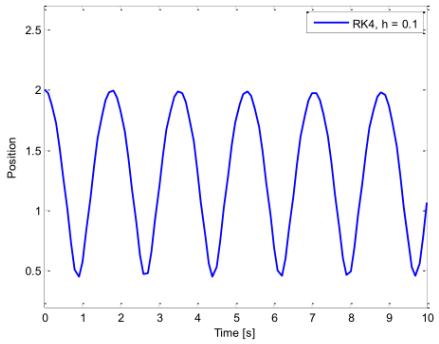
- On state-space form $\dot{y} = f(y,t)$

$$\begin{pmatrix} \dot{y}_1 \\ \dot{y}_2 \end{pmatrix} = \begin{pmatrix} y_2 \\ -g(1 - y_1^{-\kappa}) \end{pmatrix}$$

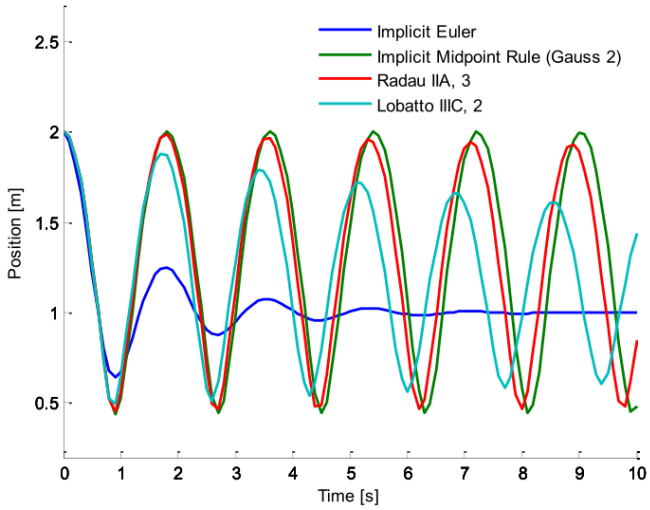
- Linearization about equilibrium:

$$\frac{\partial f}{\partial y} = \begin{pmatrix} 0 & 1 \\ -g\kappa & 0 \end{pmatrix}, \qquad \lambda_{1,2} = \pm j\omega_0, \quad \omega_0 = \sqrt{g\kappa} \approx 3.7$$

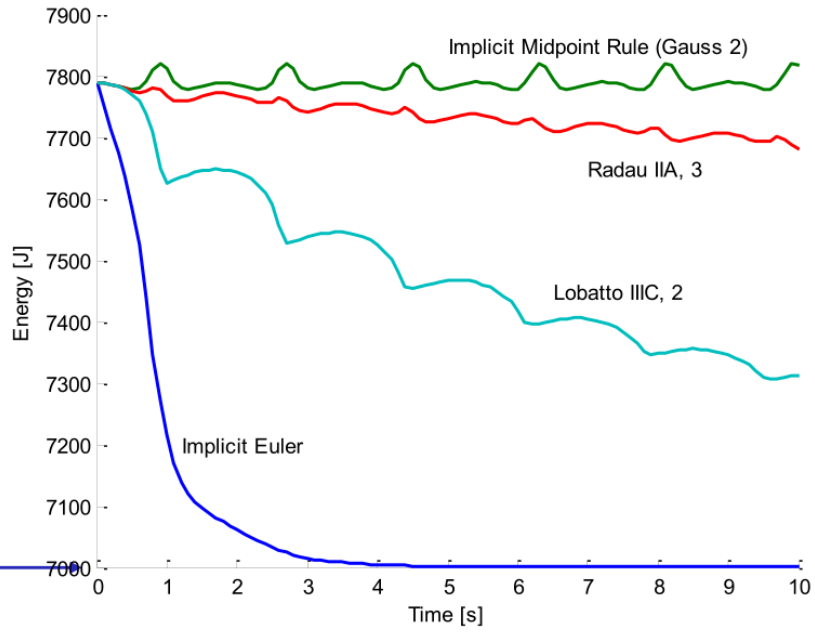
Simulation



h = 0.5 (stability limit for RK4)



Energy



Equilibrium energy

Lecture 9: Stability, Padé approximations

- Stability of Runga-Kutta methods
  - Aliasing
  - **A-stability**
  - **L-stability**
  - Padé approximations
  - (Nonlinear stability: AN-, B- and algebraic stability)

Book: 14.6

Explicit Runge-Kutta (ERK) methods

- IVP: $\dot{y} = f(y,t), \quad y(0) = y_0$
- ERK:

$$
\begin{aligned}
k_1 &= f(y_n, t_n) \\
k_2 &= f(y_n + h a_{21} k_1, t_n + c_2 h) \\
k_3 &= f(y_n + h(a_{31} k_1 + a_{32} k_2), t_n + c_3 h) \\
&\vdots \\
k_\sigma &= f(y_n + h(a_{\sigma,1} k_1 + a_{\sigma,2} k_2 + \ldots + a_{\sigma,\sigma-1} k_{\sigma-1}), t_n + c_\sigma h) \\
y_{n+1} &= y_n + h(b_1 k_1 + b_2 k_2 + \ldots + b_\sigma k_\sigma)
\end{aligned}
$$

- Butcher array:

$\mathbf{c}$	$\mathbf{A}$
	$\mathbf{b}^\mathsf{T}$

$$
\begin{array}{c|ccccc}
0 & & & & & \\
c_2 & a_{21} & & & & \\
c_3 & a_{31} & a_{32} & & & \\
\vdots & \vdots & \vdots & \ddots & & \\
c_\sigma & a_{\sigma,1} & a_{\sigma,2} & \cdots & a_{\sigma,\sigma-1} & \\
\hline
 & b_1 & b_2 & \ldots & b_{\sigma-1} & b_\sigma
\end{array}
$$

Recap: Test system, stability function

- One step method (typically: Runge-Kutta):

$$y_{n+1} = y_n + h\phi(y_n, t_n)$$

- Apply it to scalar test system:

$$\dot{y} = \lambda y$$

- We get:

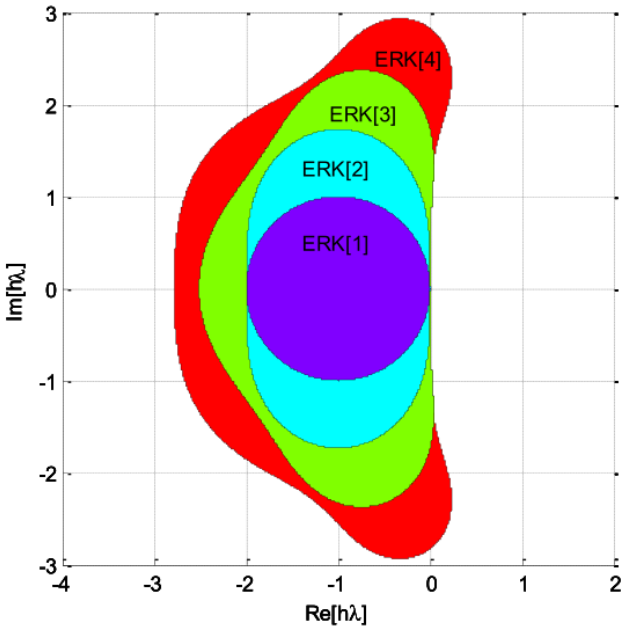
$$y_{n+1} = R(h\lambda)y_n$$

  where $R(h\lambda)$ is stability function

- The method is stable (for test system!) if

$$|R(h\lambda)| \leq 1$$

Stability regions for ERK methods



Implicit Runge-Kutta (IRK) methods

- IVP: $\dot{y} = f(y,t), \quad y(0) = y_0$
- IRK:

$$
\begin{aligned}
k_1 &= f(y_n + h(a_{1,1}k_1 + a_{1,2}k_2 + \ldots + a_{1,\sigma}k_\sigma), t_n + c_1 h) \\
k_2 &= f(y_n + h(a_{2,1}k_1 + a_{2,2}k_2 + \ldots + a_{2,\sigma}k_\sigma), t_n + c_2 h) \\
k_3 &= f(y_n + h(a_{3,1}k_1 + a_{3,2}k_2 + \ldots + a_{3,\sigma}k_\sigma), t_n + c_3 h) \\
&\vdots \\
k_\sigma &= f(y_n + h(a_{\sigma,1}k_1 + a_{\sigma,2}k_2 + \ldots + a_{\sigma,\sigma}k_\sigma), t_n + c_\sigma h) \\
y_{n+1} &= y_n + h(b_1 k_1 + b_2 k_2 + \ldots + b_\sigma k_\sigma)
\end{aligned}
$$

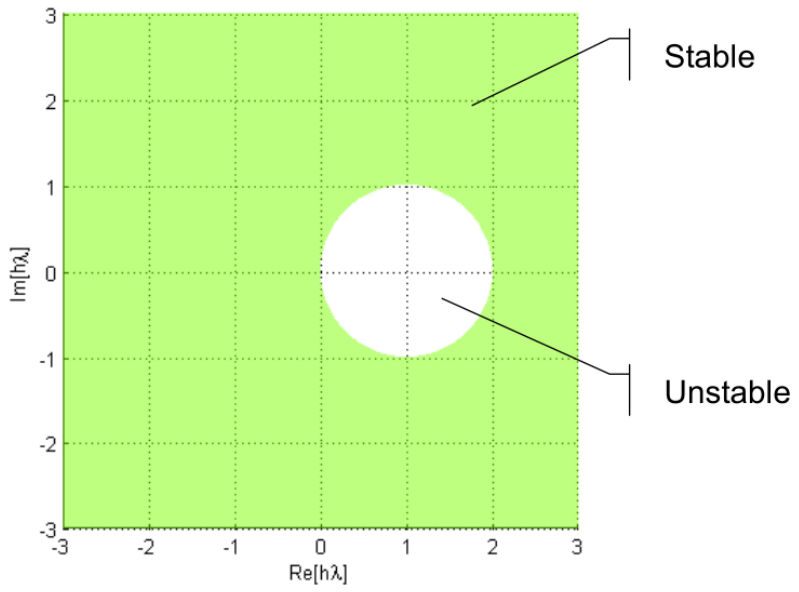
- Butcher array:

$\mathbf{c}$	$\mathbf{A}$
	$\mathbf{b}^\mathsf{T}$

c_1	a_{11}	a_{12}	$\cdots$	$a_{1,\sigma-1}$	$a_{1,\sigma}$
c_2	a_{21}	a_{22}	$\cdots$	$a_{2,\sigma-1}$	$a_{2,\sigma}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
c_σ	$a_{\sigma,1}$	$a_{\sigma,2}$	$\cdots$	$a_{\sigma,\sigma-1}$	$a_{\sigma,\sigma}$
	b_1	b_2	$\ldots$	$b_{\sigma-1}$	b_σ

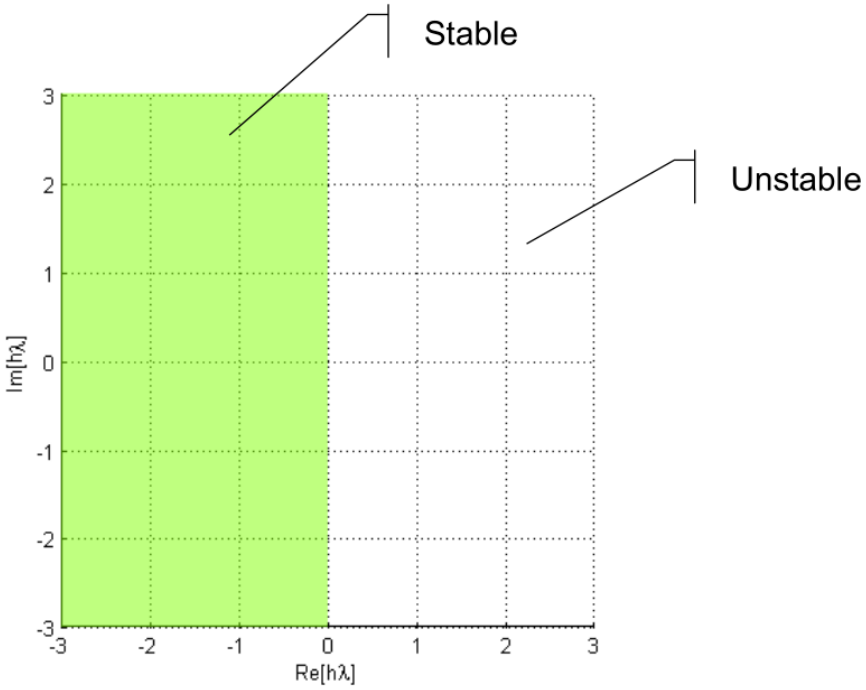
Stability regions for implicit methods

- Implicit Euler



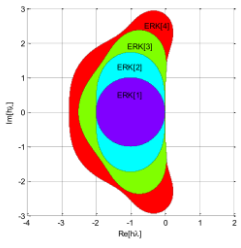
Stability regions for implicit methods

- Trapezoidal rule and implicit midpoint rule:

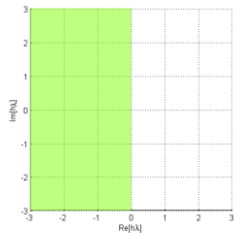


Why use IRK methods?

- IRK methods are much more complex (since we have to solve a set of nonlinear equations for each step) than ERK methods, so why use them?
  - Accuracy? Stability?



- Not because of accuracy
  - Even if an IRK method may have higher accuracy for a given number of stages, it is easy and cheap to achieve same accuracy for IRK by increasing the number of stages



- It's because of the much larger stability region!

- When is this important?
  - **Stiff systems!**

Example: ”Lambert’s problem”

- IVP:

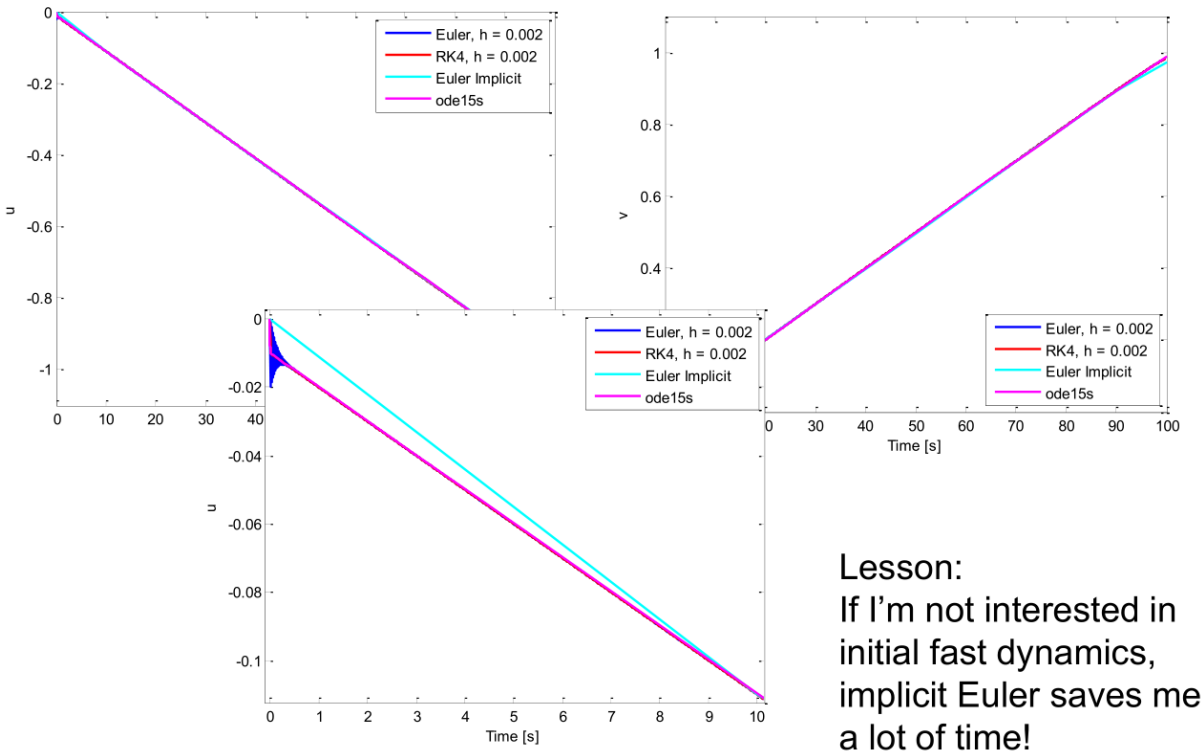
$$\dot{u} = \frac{1}{100} - (\frac{1}{100} + u + v)(1 + (u + 1000)(u + 1)), \quad u(0) = 0$$
$$\dot{v} = \frac{1}{100} - (\frac{1}{100} + u + v)(1 + v^2), \quad v(0) = 0$$

- Task: Simulate from t = 0 s til t = 100 s

- Eigenvalues:

$$(u, v) = (0, 0) \Rightarrow \lambda_1 \approx -1000, \lambda_2 \approx -0.01$$
$$(u, v) = (-.5, .5) \Rightarrow \lambda_1 \approx -500, \lambda_2 \approx -0.03$$
$$(u, v) = (-1, 1) \Rightarrow \lambda_1 \approx -11, \lambda_2 \approx -1$$

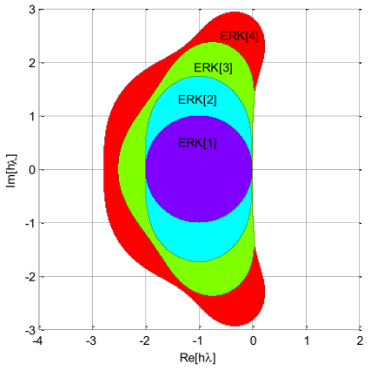
Comparisons



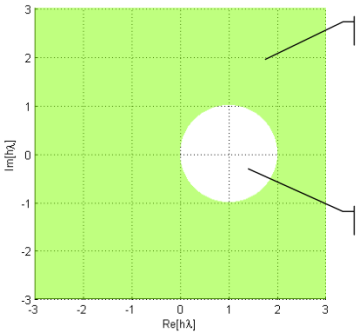
Lesson:
If I’m not interested in initial fast dynamics, implicit Euler saves me a lot of time!

ERK vs IRK stability regions

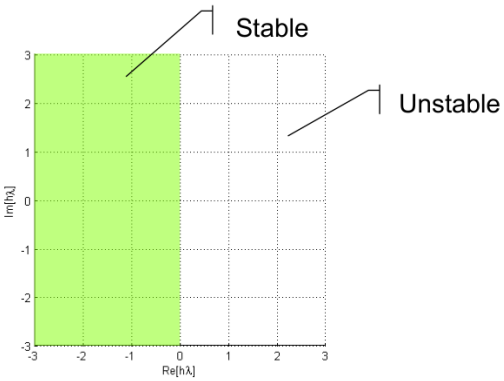
ERK-methods



Implicit Euler



Trapezoidal/
Implicit Midpoint



Padé approximations to e^s

m \ k	0	1	2	3
0	$\frac{1}{1}$	$\frac{1+s}{1}$	$\frac{1+s+\frac{1}{2}s^2}{1}$	$\frac{1+s+\frac{1}{2}s^2+\frac{1}{6}s^3}{1}$
1	$\frac{1}{1-s}$	$\frac{1+\frac{1}{2}s}{1-\frac{1}{2}s}$	$\frac{1+\frac{2}{3}s+\frac{1}{6}s^2}{1-\frac{1}{3}s}$	$\frac{1+\frac{3}{4}s+\frac{1}{4}s^2+\frac{1}{24}s^3}{1-\frac{1}{4}s}$
2	$\frac{1}{1-s+\frac{1}{2}s^2}$	$\frac{1+\frac{1}{3}s}{1-\frac{2}{3}s+\frac{1}{6}s^2}$	$\frac{1+\frac{1}{2}s+\frac{1}{12}s^2}{1-\frac{1}{2}s+\frac{1}{12}s^2}$	$\frac{1+\frac{3}{5}s+\frac{3}{20}s^2+\frac{1}{60}s^3}{1-\frac{2}{5}s+\frac{1}{20}s^2}$
3	$\frac{1}{1-s+\frac{1}{2}s^2-\frac{1}{6}s^3}$	$\frac{1+\frac{1}{4}s}{1-\frac{3}{4}s+\frac{1}{4}s^2-\frac{1}{24}s^3}$	$\frac{1+\frac{2}{5}s+\frac{1}{20}s^2}{1-\frac{3}{5}s+\frac{3}{20}s^2-\frac{1}{60}s^3}$	$\frac{1+\frac{1}{2}s+\frac{1}{10}s^2+\frac{1}{120}s^3}{1-\frac{1}{2}s+\frac{1}{10}s^2-\frac{1}{120}s^3}$

L-stable L-stable A-stable

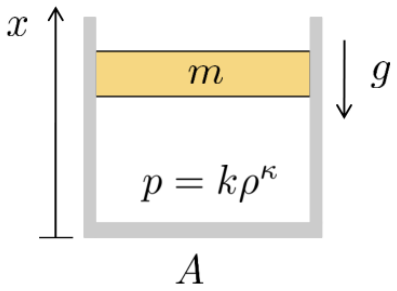
- $m = 0$: Explicit Runge-Kutta methods with $p = \sigma$
- $m = k$: Gauss, Lobatto IIIA/IIIB (incl. implicit mid-point, trapezoidal)
- $m = k+1$: Radau-methods (incl. implicit Euler)
- $m = k+2$: Lobatto IIIC

Pneumatic spring example, again

- Model from Newton's 2nd law:

$$\ddot{x} + g(1 - x^{-\kappa}) = 0$$

"mass-spring-damper with nonlinear spring"



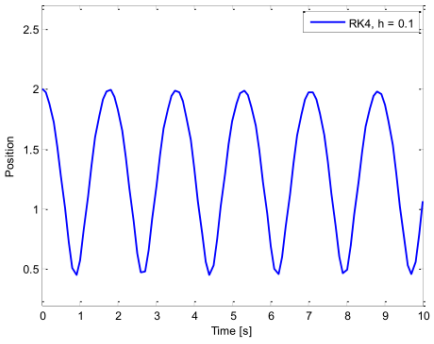
- On state-space form $\dot{y} = f(y, t)$

$$\begin{pmatrix} \dot{y}_1 \\ \dot{y}_2 \end{pmatrix} = \begin{pmatrix} y_2 \\ -g(1 - y_1^{-\kappa}) \end{pmatrix}$$

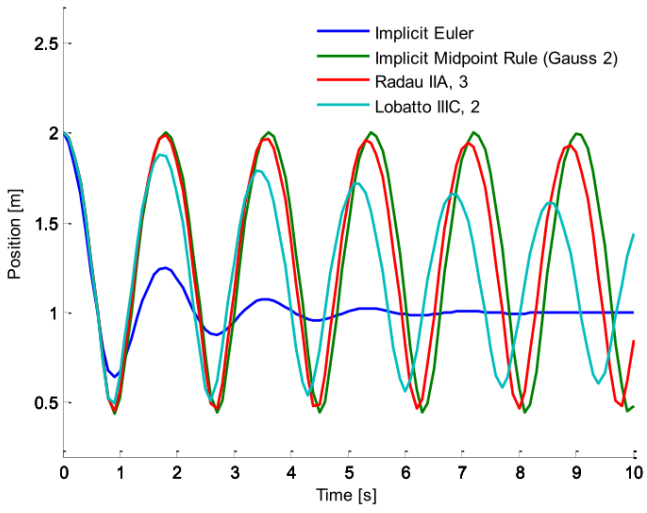
- Linearization about equilibrium:

$$\frac{\partial f}{\partial y} = \begin{pmatrix} 0 & 1 \\ -g\kappa & 0 \end{pmatrix}, \qquad \lambda_{1,2} = \pm j\omega_0, \quad \omega_0 = \sqrt{g\kappa} \approx 3.7$$

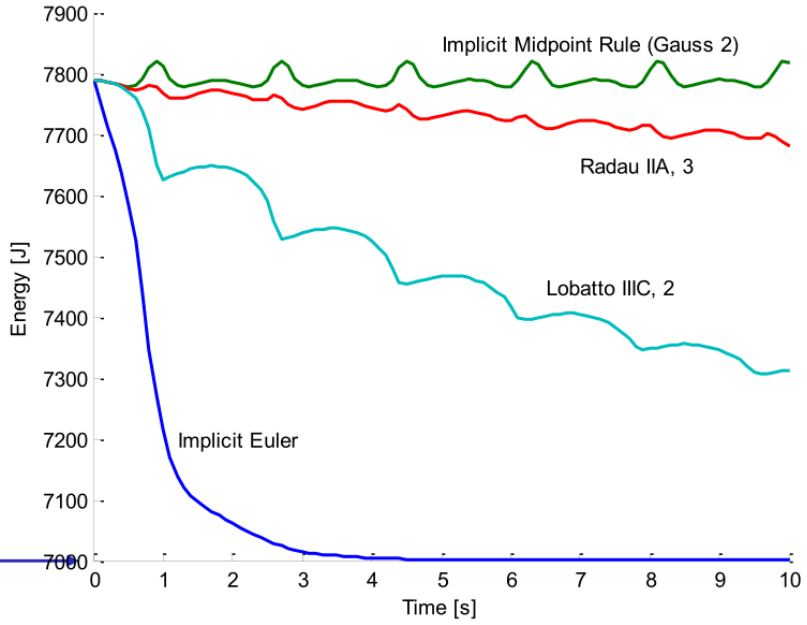
Simulation



h = 0.5 (stability limit for RK4)



Energy



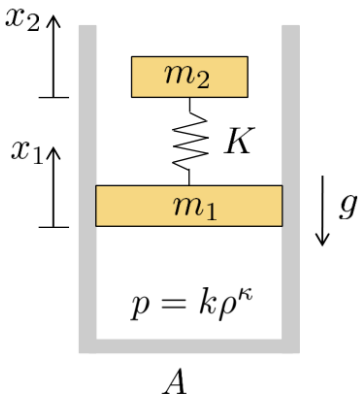
Equilibrium energy

Pneumatic spring with resonant load

- Equations of motion (Newton's law):

$$\ddot{x}_1 + g\left(1 - \frac{m_1+m_2}{m_1}x_1^{-\kappa}\right) + \frac{\omega_2^2}{2}(x_1 - x_2) = 0$$

$$\ddot{x}_2 + g + \frac{\omega_2^2}{2}(x_2 - x_1) = 0$$



- Linearization around equilibrium:

$$J = \begin{pmatrix} 0 & 1 & 0 & 0 \\ -g\frac{m_1+m_2}{m_1}\kappa(x_1^*)^{-(\kappa-1)} - \frac{\omega_2^2}{2} & 0 & \frac{\omega_2^2}{2} & 0 \\ 0 & 0 & 0 & 1 \\ \frac{\omega_2^2}{2} & 0 & -\frac{\omega_2^2}{2} & 0 \end{pmatrix}$$

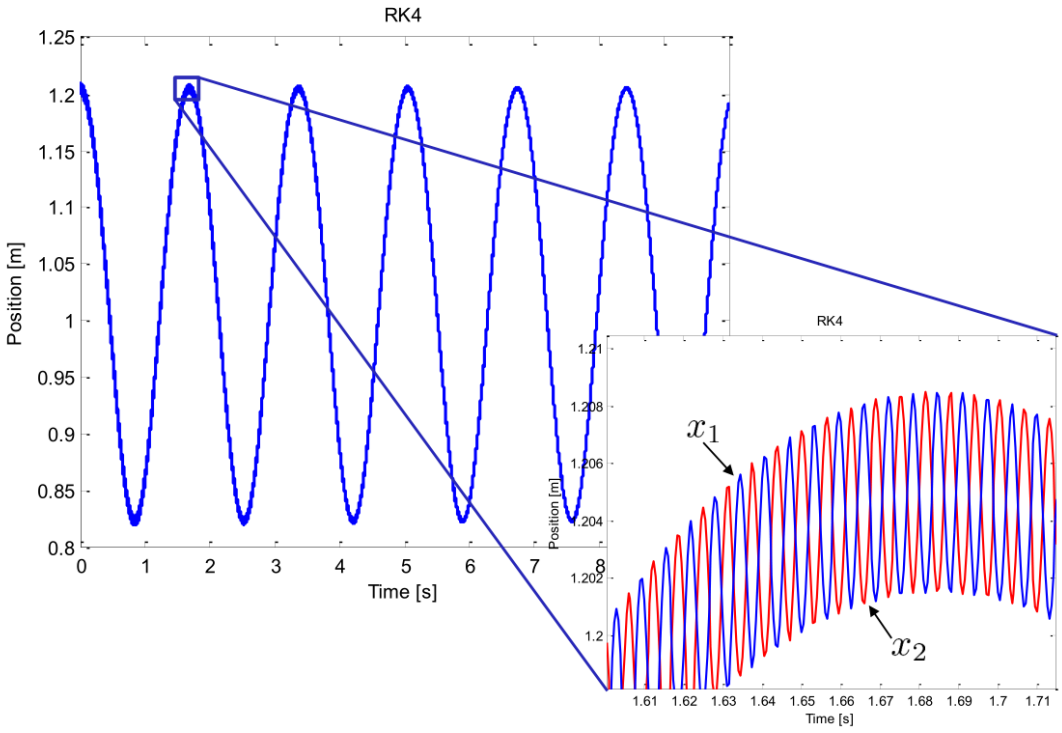
- Eigenvalues:

$$\lambda_{1,2} = \pm j\omega_1, \quad \omega_1 = 3.7 \text{ rad/s}$$

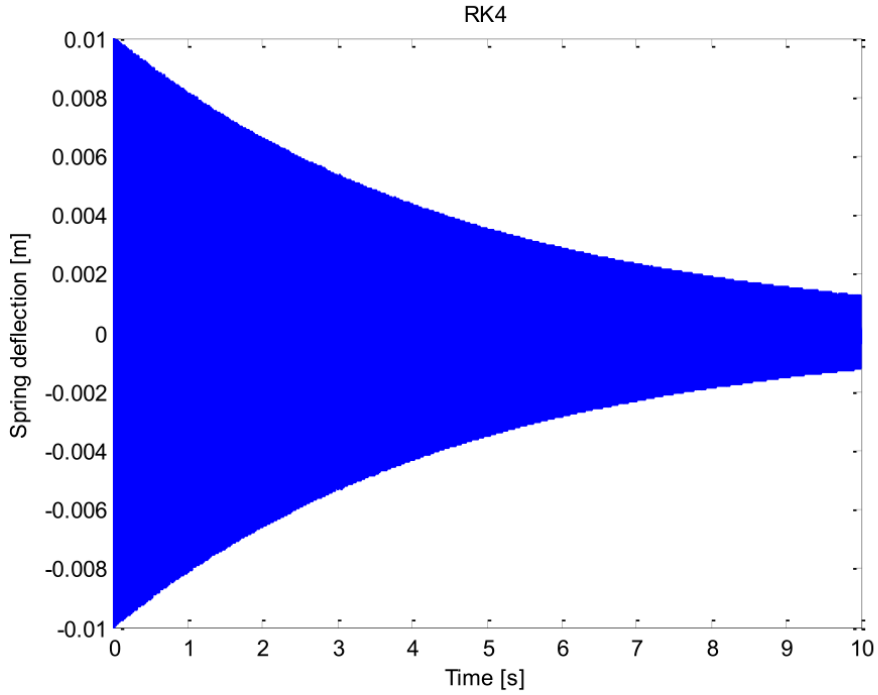
$$\lambda_{3,4} = \pm j\omega_2, \quad \omega_2 = 1000 \text{ rad/s}$$

Position of the two masses

RK4 with time step h = 0.0005

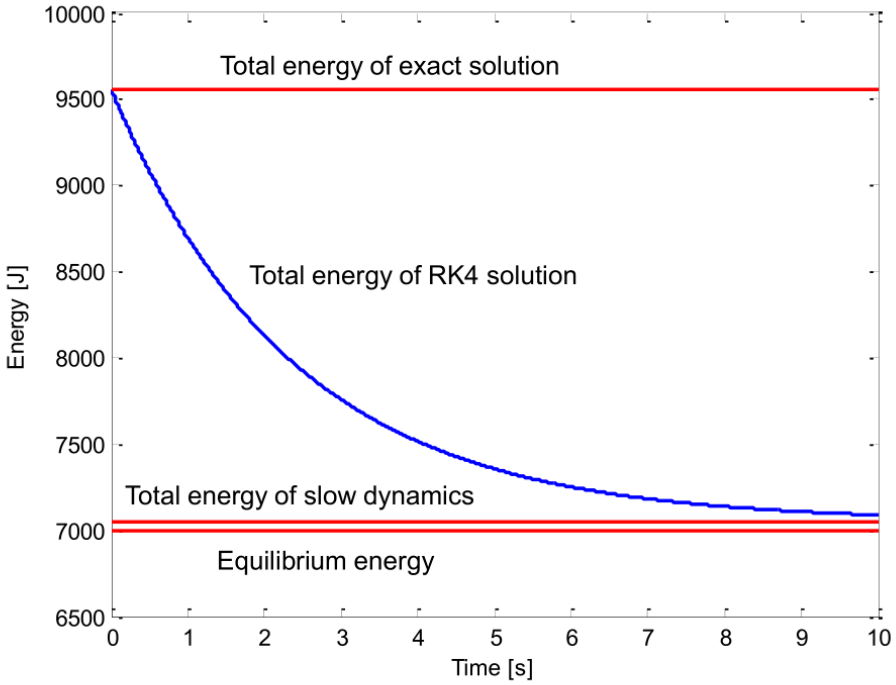


Spring deflection, RK4 with h = 0.0005



- Oscillation is lightly damped by integration method

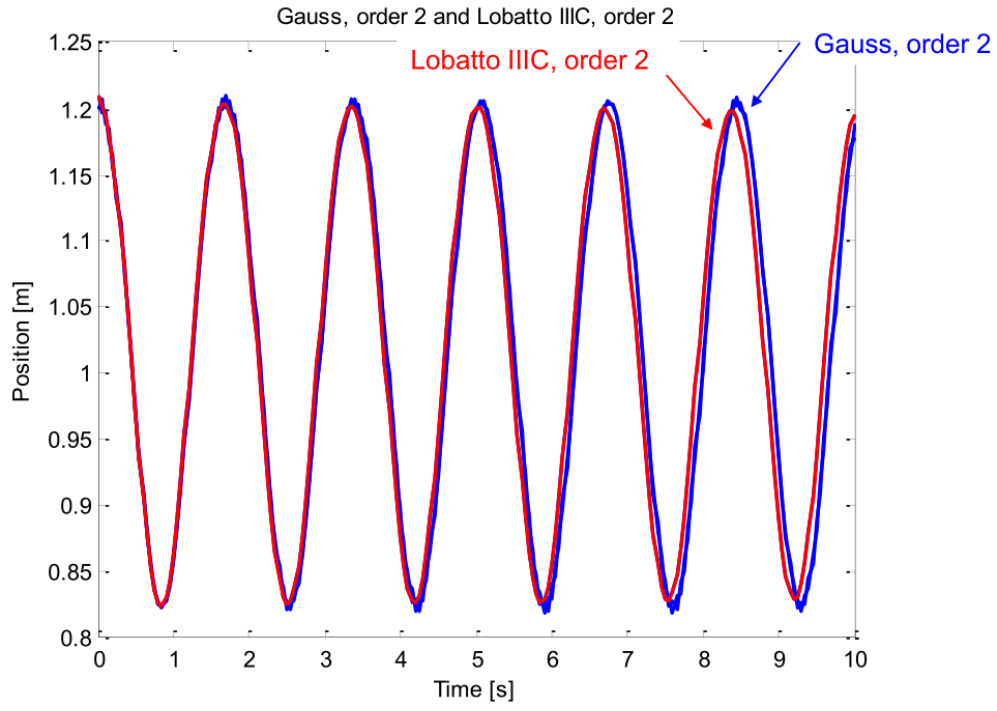
Energy of RK4 solution, h = 0.0005



- Energy related to fast dynamics slowly damped out

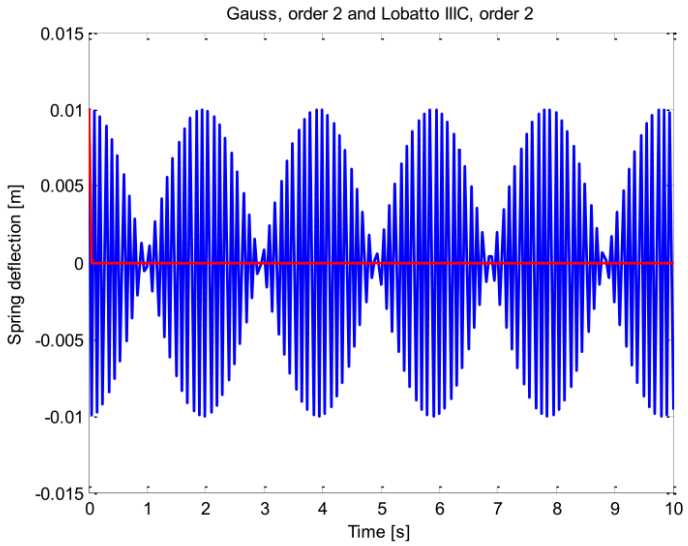
Position of the two masses

Gauss, order 2 and Lobatto IIIC, order 2, h = 0.05



Spring deflection

Gauss, order 2 and Lobatto IIIC, order 2, h = 0.05



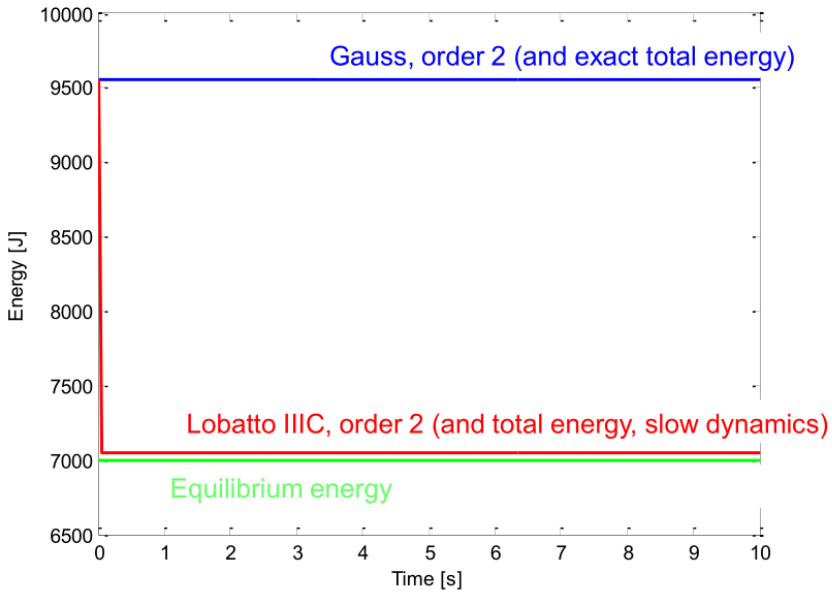
Gauss, order 2

Lobatto IIIC, order 2

- Gauss method gives no damping, but shifts fast dynamics and energy to frequencies below Nyquist frequency, $\omega_N = \frac{\pi}{h} = \frac{\pi}{0.05} = 62.8$
- Lobatto IIIC dampens out fast dynamics in one step

Total energies

Gauss, order 2 and Lobatto IIIC, order 2, h = 0.05



- Gauss does not dampen energies at all (same as exact total energy)
- Lobatto IIIC dampens out energy associated with fast dynamics in very few steps, to the energy of slow dynamics

Lecture 10: Stability and frequency properties (mostly recap), error control and software

- Stability (mostly recap), frequency properties (14.6)
- Error control: Automatic adjustment of step-size (14.7)
  - Solver vs Integrator
- Interpolation and events
- Briefly:
  - Multistep methods
    - ”Tend to be more efficient than single-step methods for systems with smooth solutions and high accuracy requirements”
    - Often used in advanced modeling software (e.g. Dymola)
  - Differential-algebraic systems
  - Software

Implicit Runge-Kutta (IRK) methods

- IVP: $\dot{y} = f(y,t), \quad y(0) = y_0$
- IRK:

$$
\begin{aligned}
k_1 &= f(y_n + h(a_{1,1}k_1 + a_{1,2}k_2 + \ldots + a_{1,\sigma}k_\sigma), t_n + c_1 h) \\
k_2 &= f(y_n + h(a_{2,1}k_1 + a_{2,2}k_2 + \ldots + a_{2,\sigma}k_\sigma), t_n + c_2 h) \\
k_3 &= f(y_n + h(a_{3,1}k_1 + a_{3,2}k_2 + \ldots + a_{3,\sigma}k_\sigma), t_n + c_3 h) \\
&\vdots \\
k_\sigma &= f(y_n + h(a_{\sigma,1}k_1 + a_{\sigma,2}k_2 + \ldots + a_{\sigma,\sigma}k_\sigma), t_n + c_\sigma h) \\
y_{n+1} &= y_n + h(b_1 k_1 + b_2 k_2 + \ldots + b_\sigma k_\sigma)
\end{aligned}
$$

- Butcher array:

$\mathbf{c}$	$\mathbf{A}$
	$\mathbf{b}^\mathsf{T}$

$$
\begin{array}{c|ccccc}
c_1 & a_{11} & a_{12} & \cdots & & a_{1,\sigma} \\
c_2 & a_{21} & a_{22} & \cdots & & a_{2,\sigma} \\
\vdots & \vdots & \vdots & \ddots & & \\
c_\sigma & a_{\sigma,1} & a_{\sigma,2} & \cdots & a_{\sigma,\sigma-1} & a_{\sigma,\sigma} \\
\hline
 & b_1 & b_2 & \ldots & b_{\sigma-1} & b_\sigma
\end{array}
$$

Recap: Test system, stability function

- One step method (typically: Runge-Kutta):

$$y_{n+1} = y_n + h\phi(y_n, t_n)$$

- Apply it to scalar test system:

$$\dot{y} = \lambda y$$

- We get:

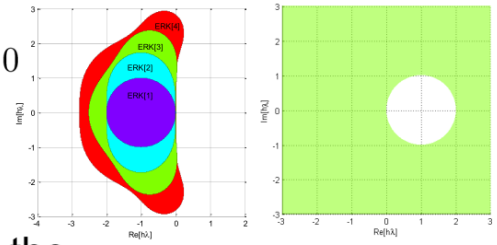
$$y_{n+1} = R(h\lambda)y_n$$

  where $R(h\lambda)$ is stability function

- The method is stable (for test system!) if

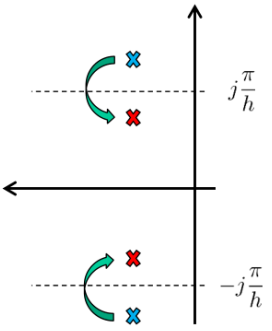
$$|R(h\lambda)| \leq 1$$

Linear stability: A- and L-stability



- A-stability: $|R(h\lambda)| \leq 1$ for all $\mathrm{Re}\,\lambda \leq 0$
  - Relevant (mostly) for **stiff** systems
  - No explicit methods are A-stable
  - Many implicit methods are A-stable
- Aliasing: Frequencies larger than the «Nyquist frequency» *σ*/*h* are mapped to within the Nyquist frequency
- A method is L-stable if it is A-stable, and

$$|R(hj\omega)| \to 0 \text{ when } \omega \to \infty$$



  - Relevant for (stiff) systems with **oscillatory modes**
  - Dampens out fast frequencies
  - We often want L-stability in implicit methods, but not always:
    - We typically want to suppress dynamics that are faster than step length («stiff decay»)
    - However, we may want to not dampen oscillatory modes
    - We may want to **not** dissipate energy (numerically) in simulations

Padé approximations

- The local solution to test system:

$$y_L(t_n; t_{n+1}) = e^{h\lambda} y_n$$

- Stability function:

$$y_{n+1} = R(h\lambda) y_n$$

- A method is «good» if

$$R(s) \approx e^s$$

- Explicit Runge-Kutta methods with $\sigma = p \leq 4$: Taylor expansion!

$$R(s) = 1 + s + \ldots + \frac{s^p}{p!}$$

- Implicit Runge-Kutta methods:

$$R(s) = \frac{1 + \beta_1 s + \ldots + \beta_k s^k}{1 + \gamma_1 s + \ldots + \gamma_m s^m}$$

- Best approximation (for given *k* and *m*): Padé-approximation

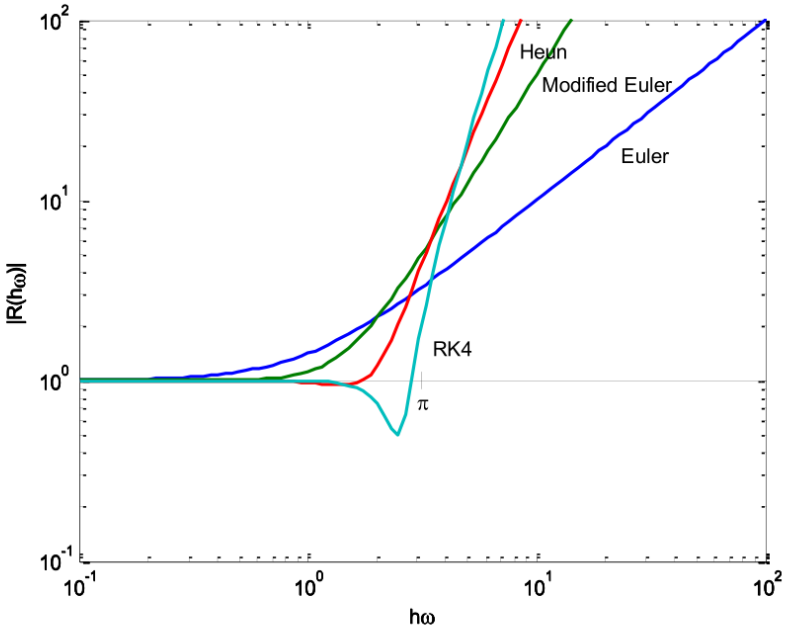
Padé approximations to e^s

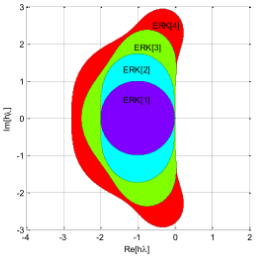
m \ k	0	1	2	3
0	$\frac{1}{1}$	$\frac{1+s}{1}$	$\frac{1+s+\frac{1}{2}s^2}{1}$	$\frac{1+s+\frac{1}{2}s^2+\frac{1}{6}s^3}{1}$
1	$\frac{1}{1-s}$	$\frac{1+\frac{1}{2}s}{1-\frac{1}{2}s}$	$\frac{1+\frac{2}{3}s+\frac{1}{6}s^2}{1-\frac{1}{3}s}$	$\frac{1+\frac{3}{4}s+\frac{1}{4}s^2+\frac{1}{24}s^3}{1-\frac{1}{4}s}$
2	$\frac{1}{1-s+\frac{1}{2}s^2}$	$\frac{1+\frac{1}{3}s}{1-\frac{2}{3}s+\frac{1}{6}s^2}$	$\frac{1+\frac{1}{2}s+\frac{1}{12}s^2}{1-\frac{1}{2}s+\frac{1}{12}s^2}$	$\frac{1+\frac{3}{5}s+\frac{3}{20}s^2+\frac{1}{60}s^3}{1-\frac{2}{5}s+\frac{1}{20}s^2}$
3	$\frac{1}{1-s+\frac{1}{2}s^2-\frac{1}{6}s^3}$	$\frac{1+\frac{1}{4}s}{1-\frac{3}{4}s+\frac{1}{4}s^2-\frac{1}{24}s^3}$	$\frac{1+\frac{2}{5}s+\frac{1}{20}s^2}{1-\frac{3}{5}s+\frac{3}{20}s^2-\frac{1}{60}s^3}$	$\frac{1+\frac{1}{2}s+\frac{1}{10}s^2+\frac{1}{120}s^3}{1-\frac{1}{2}s+\frac{1}{10}s^2-\frac{1}{120}s^3}$

L-stable L-stable A-stable

- *m* = *0*: Explicit Runge-Kutta methods with $p = \sigma$
- *m* = *k*: Gauss, Lobatto IIIA/IIIB (incl. implicit mid-point, trapezoidal)
- *m* = *k*+1: Radau-methods (incl. implicit Euler)
- *m* = *k*+2: Lobatto IIIC

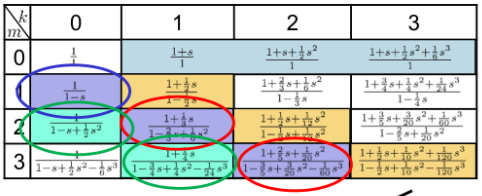
Frequency properties, explicit methods

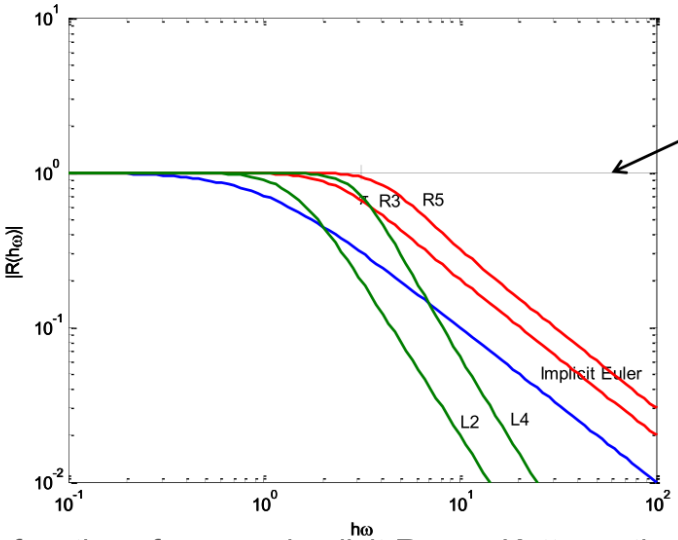




- Stability functions for explicit Runge-Kutta methods plotted as a function of $s = jh\omega$
- «Nyquist frequency» $h\omega = \pi$ indicated

Frequency properties, implicit methods



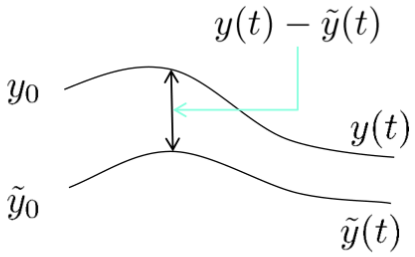


Gauss methods has $|R(jh\omega)| = 1$, i.e. no damping

- Stability functions for some implicit Runge-Kutta methods plotted as a function of $s = jh\omega$
- «Nyquist frequency» $h\omega = \pi$ indicated
- Dampens out high frequencies: «Rolloff» of -1 for Radau methods vs -2 for Lobatto IIIC methods

Nonlinear stability

- AN-stability
  - Stability for time-varying linear system
  - Implies A-stability



- B-stability:
  - Given "contracting" system $\dot{y} = f(y,t)$

    $$\|y(t) - \tilde{y}(t)\| \to 0 \text{ exponentially}$$
  - A Runge-Kutta method is B-stable if the solutions fulfill

    $$\|y_{n+1} - \tilde{y}_{n+1}\| \leq \|y_n - \tilde{y}_n\|$$

    for all contracting systems
  - B-stability implies AN-stability
  - "Difficult" to check (in general), but…

Nonlinear stability, cont'd

$$\begin{array}{c|c} \mathbf{c} & \mathbf{A} \\ \hline & \mathbf{b}^{\mathsf{T}} \end{array}$$

Algebraic stability:

- An (implicit) Runge-Kutta method is *algebraically stable* if
  - $b_i \geq 0, \quad i = 1, \ldots, \sigma$
  - $\mathbf{M} = \mathrm{diag}(\mathbf{b})\mathbf{A} + \mathbf{A}^{\mathsf{T}}\mathrm{diag}(\mathbf{b}) - \mathbf{b}\mathbf{b}^{\mathsf{T}} \geq 0$ (positive semidefinite)
- Easy to check

- The nonlinear stability concepts implies A-stability
  - Algebraic stability implies B-stability
  - B-stability implies AN-stability
  - AN-stability implies A-stability

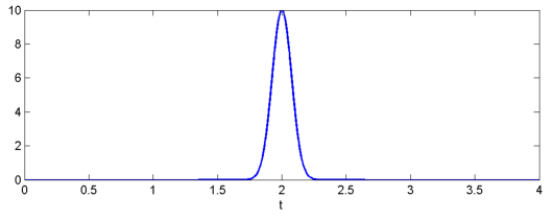
- Interesting fact:
  - Trapezoid and Implicit Midpoint have same stability function (same linear stability), but only Implicit Midpoint is algebraically stable (and B-stable)

Automatic adjustment of steplength

- We have seen that accuracy depends on step length *h*
  - (e.g. A-stable methods: Always stable, but not necessarily accurate)
- How to choose step lengths?
  - Systems that are (close to) linear with eigenvalues in limited range:
    - "Easy" to choose *h* to have stability & desired accuracy everywhere
  - Systems that are (linear or nonlinear) and stiff or highly time-dependent:
    - How to choose *h*?
    - *h* too large: inaccurate (or even unstable) in some periods/regions
    - *h* too small: inefficient in some periods/regions
- Would it not be nice if we could specify what accuracy we want, and let the ODE solver choose appropriate step-lengths?

Example

- We want to simulate

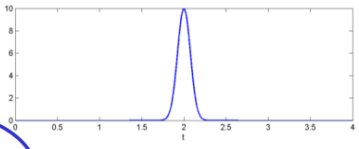


$$\dot{y} = -0.6y + 10e^{-\frac{(t-2)^2}{2\cdot 0.075^2}}, \quad y(0) = 0.5$$

- Matlab, using ode23

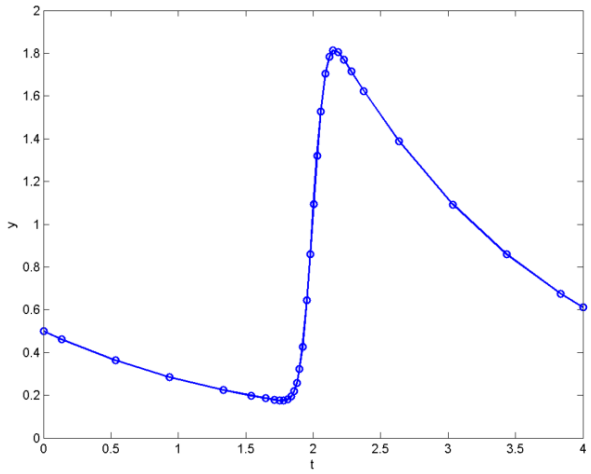
```
% f(t,y)
f = @(t,y) ( -0.6*y + 10*exp(-(t-2).^2/(2*(0.075^2))) );

% Set desired accuracy
options = odeset('RelTol',10^-3);

% Simulate
[t,y] = ode23(f, [0 4], 0.5, options);

% Plot solution
plot(t,y,'-o');
```

Example, cont'd

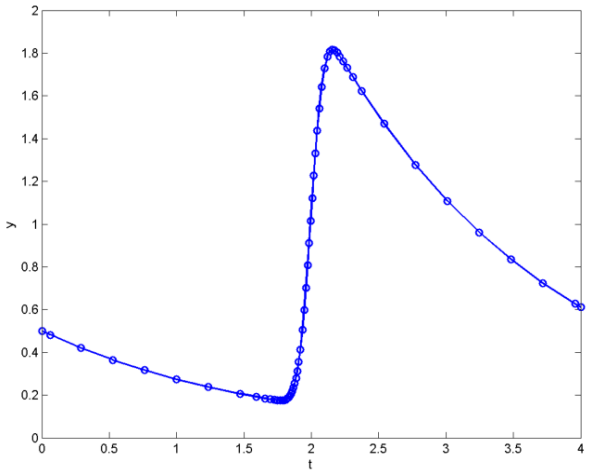


$$\dot{y} = -0.6y + 10e^{-\frac{(t-2)^2}{2\cdot 0.075^2}}, \quad y(0) = 0.5$$

RelTol = 10^{-3}:

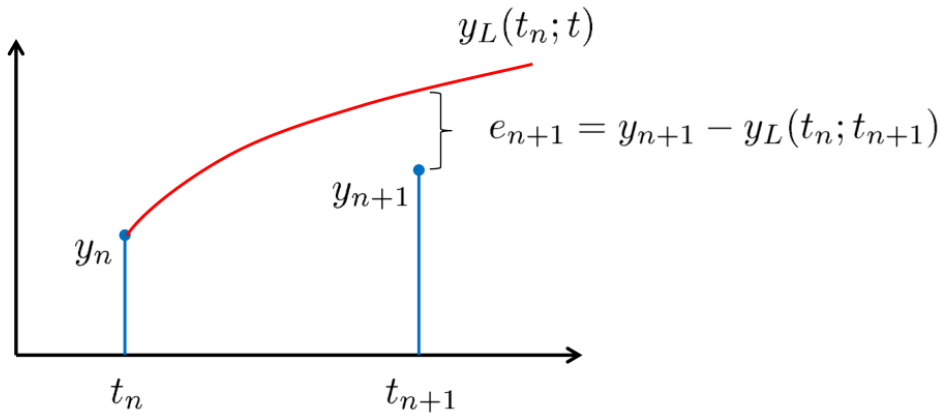


RelTol = 10^{-4}:



Estimation of local error

- Recall: Local error is error from y_n to y_{n+1}



Estimation of local error

- Starting at y_n,

Calculate y_{n+1} using (E)RK

$$\begin{array}{c|c} \mathbf{c} & \mathbf{A} \\ \hline & \mathbf{b}^{\mathsf{T}} \end{array}$$

with order *p*
local error $e_{n+1} = O(h^{p+1})$

Calculate $\hat{y}_{n+1}$ using (E)RK

$$\begin{array}{c|c} \hat{\mathbf{c}} & \hat{\mathbf{A}} \\ \hline & \hat{\mathbf{b}}^{\mathsf{T}} \end{array}$$

with order *p+1*
local error $\hat{e}_{n+1} = O(h^{p+2})$

- Local solution (per def.):

$$y_L(t_n; t_{n+1}) = y_{n+1} + e_{n+1}$$

$$y_L(t_n; t_{n+1}) = \hat{y}_{n+1} + \hat{e}_{n+1}$$

- Combine:

$$\hat{y}_{n+1} - y_{n+1} = e_{n+1} - \hat{e}_{n+1} \approx e_{n+1}$$

- Gives estimate of local error:

$$e_{n+1} \approx \hat{y}_{n+1} - y_{n+1}$$

RK4(5) Runge-Kutta-Fehlberg (1969)

- σ = 6, p = 4, $\hat{p}$ = 5

0						
1/4	1/4					
3/8	3/32	9/32				
12/13	1932/2197	−7200/2197	7296/2197			
1	439/216	−8	3680/513	−845/4104		
1/2	-8/27	2	−3544/2565	1859/4104	−11/40	
	25/216	0	1408/2565	2197/4104	−1/5	0
	16/135	0	6656/12825	28561/56430	−9/50	2/55

- Issue: Why use y_{n+1} (p = 4) when we have calculated more accurate $\hat{y}_{n+1}$ ($\hat{p}$ = 5)?
  - Use $\hat{y}_{n+1}$ instead: “local extrapolation”
  - Some numerical issues/optimizations concerning accuracy comes into play

Methods using local extrapolation

- Dormand-Prince 5(4) – DP5(4) (1980)

0							
1/5	1/5						
3/10	3/40	9/40					
4/5	44/45	−56/15	32/9				
8/9	19372/6561	−25360/2187	64448/6561	−212/729			
1	9017/3168	−355/33	46732/5247	49/176	−5103/18656		
1	35/384	0	500/1113	125/192	−2187/6784	11/84	
	5179/57600	0	7571/16695	393/640	−92097/339200	187/2100	1/40
	35/384	0	500/1113	125/192	−2187/6784	11/84	0

  - Implemented as ode45 in Matlab (and GNU Octave)
    - Freeware Fortran code Dopri5

- Bogacki-Shampine 2(3) – BS2(3) (1989)
  - ode23 in Matlab
  - Faster than ode45 if low accuracy demands

0				
1/2	1/2			
3/4	0	3/4		
1	2/9	1/3	4/9	
	2/9	1/3	4/9	0
	7/24	1/4	1/3	1/8

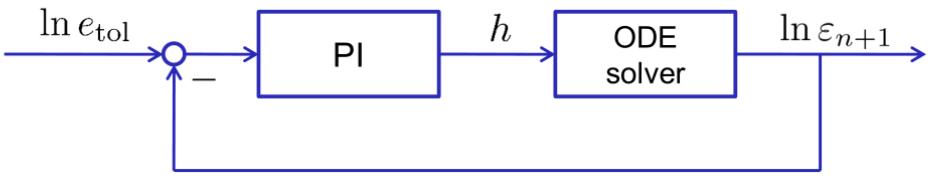
Use local error estimate to adjust step-size

- Local error estimate $e_{n+1} = (e_{1,n+1}, e_{2,n+1}, \ldots, e_{d,n+1})^\top$
- Measure of error: $\varepsilon_{n+1} = \|e_{n+1}\|_p$ (for instance $p = \infty$)

- Want error to be smaller than given tolerance e_{tol}:
  - Have: $\varepsilon_{n+1} \approx Ch^{p+1}$
  - Want: $\varepsilon_{n+1} \approx e_{\text{tol}} \approx Ch_{\text{new}}^{p+1}$

$$\frac{e_{\text{tol}}}{\varepsilon_{n+1}} \approx \left(\frac{h_{\text{new}}}{h}\right)^{p+1}$$

- Achieve this by choosing $h_{\text{new}} = h\left(\frac{e_{\text{tol}}}{\varepsilon_{n+1}}\right)^{\frac{1}{p+1}}$
  - In practice, update somewhat smoother



How to choose e_{tol} in practice?

- Say you want to simulate a model of a chemical reaction, using SI units, and have:

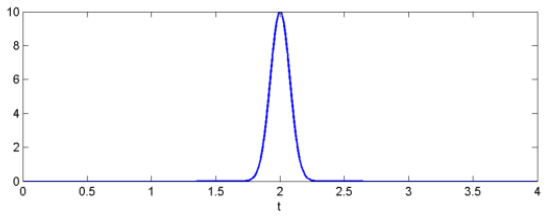
State	Nominal values	Tolerances
Pressure	10^5 Pa	10 Pa
Concentration	0.01	10^{-6}

- To give the solver a single tolerance value, you have to scale your model!
  - (often a good idea also for other reasons)
  - In practice, a perfectly scaled model is difficult to achieve
- Alternatively, use solvers that implement relative tolerance (possibly in addition to absolute tolerance)
  - Matlab: $e_{\mathrm{tol},i} = \max\{r|y_i|, a_i\}$
  - CVode: $e_{\mathrm{tol},i} = r|y_i| + a_i$

r : RelTol (scalar) [10^{-3}]
a_i : AbsTol (vector) [10^{-6}]

Example

- We want to simulate

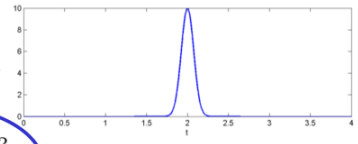


$$\dot{y} = -0.6y + 10e^{-\frac{(t-2)^2}{2\cdot 0.075^2}}, \quad y(0) = 0.5$$

- Matlab, using ode23

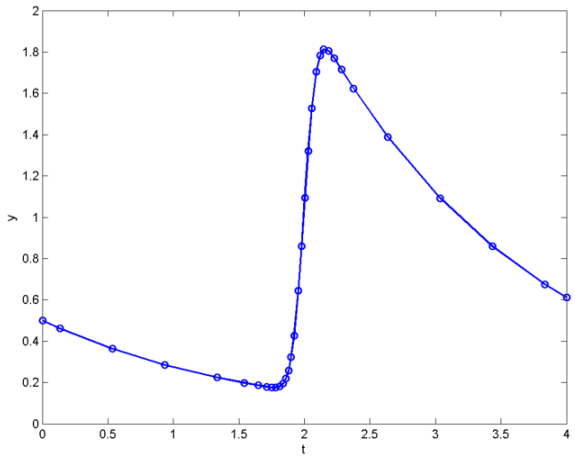
```
% y' = f(t,y)
f = @(t,y) ( -0.6*y + 10*exp(-(t-2).^2/(2*(0.075^2))) );

% Set desired accuracy
options = odeset('RelTol',10^-3);

% Simulate
[t,y] = ode23(f, [0 4], 0.5, options);

% Plot solution
plot(t,y,'-o');
```

Example, cont'd

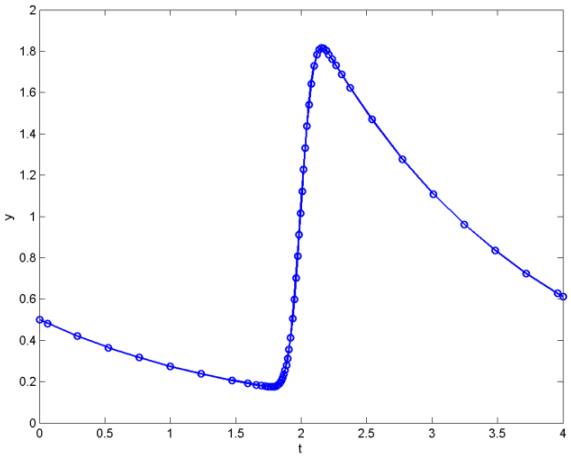


$$\dot{y} = -0.6y + 10e^{-\frac{(t-2)^2}{2 \cdot 0.075^2}}, \quad y(0) = 0.5$$

RelTol = 10^{-3}:



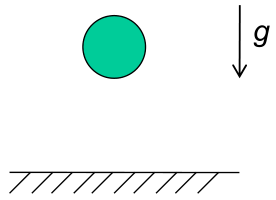
RelTol = 10^{-4}:



Event example: Bouncing ball

- Newton's law:

$$m\ddot{x} = -mg$$



- State-space:

$$\dot{x} = v$$
$$\dot{v} = -g$$

- Implementation of derivative

```
function dy = f_bb(t,y)
dy = zeros(2,1);     % column vector
g = 9.81;            % gravity
dy(1) = y(2);        % derivative of height
dy(2) = -g;          % derivative of velocity
```

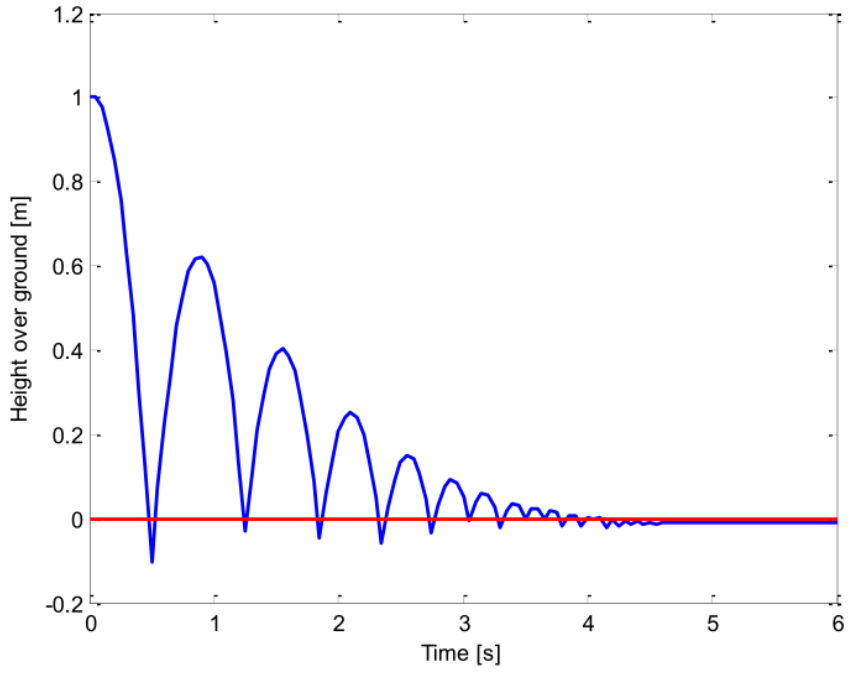
- What if we hit the ground?

```
if (y(1) <=0),       % Check if we hit the ground
   y(2) = - 0.8*y(2);      % 80% elastic
end
```

Bouncing ball: Euler implementation

```
t_0 = 0; t_end = 6; h = 0.05; timespan = t_0:h:t_end;
y = zeros(2,length(timespan)+1); % Allocate space

x_0 = 1;          % Initial height above ground [m]
v_0 = 0;          % Initial velocity [m/s]
y(:,1) = [x_0; v_0];

for i = 1:length(timespan),
    t = timespan(i);
    y(:,i+1) = y(:,i) + h*f_bb(t,y(:,i)); % Euler integration

    if (y(1,i+1) <=0),              % Check if we hit the ground
        y(2,i+1) = - 0.8*y(2,i);    % 80% elastic
    end
end
```

Bouncing ball solved using Euler



ODE-solver with event detection

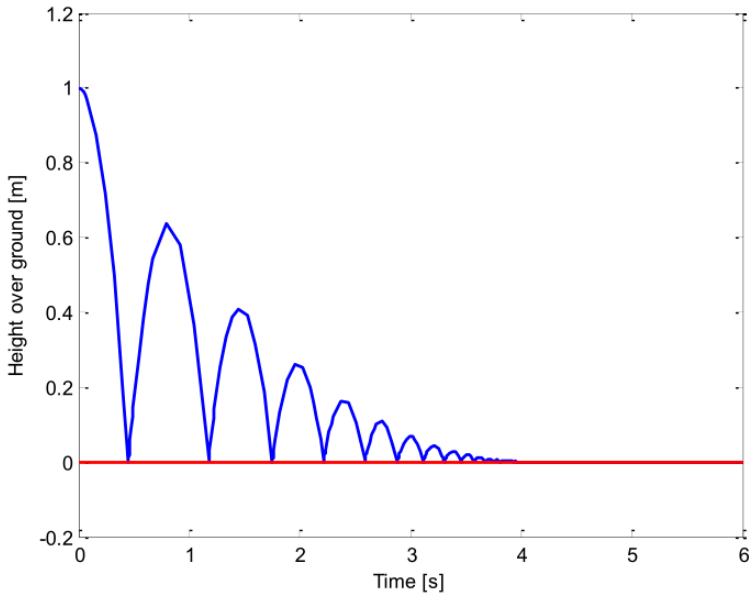
- Event-function

```
function [value,isterminal,direction] = g_bb(t,y)
value = y(1);        % Value of event-function
isterminal = 1;      % Should we stop at event? (1/0)
direction = -1;      % Event at negative to positive (1),
                     % positive to negative (-1), or both (0)
```

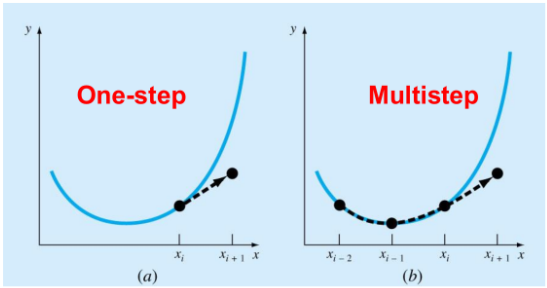
- Simulate using ode45 with event detection:

```
options = odeset('events','g_bb');          % Set event-function
t_curr = t_0; x_curr = x_0; v_curr = v_0;
while (t_curr < t_end),
    [ttemp,ytemp] = ode45(@f_bb,[t_curr t_end],[x_curr; v_curr],options);
    t_curr = ttemp(end);
    if t_curr<t_end, % Not simulated to end yet, therefore it is an event
        x_curr = 0;
        v_curr = -.8*ytemp(end,2); % 80% elastic
    end
    t = [t;ttemp]; y = [y;ytemp]; % Add to solution
end
```

Bouncing ball solved using event-function



Multi-step methods



- One-step

$$y_{n+1} = y_n + h\phi(y_n, t_n)$$

- Multi-step

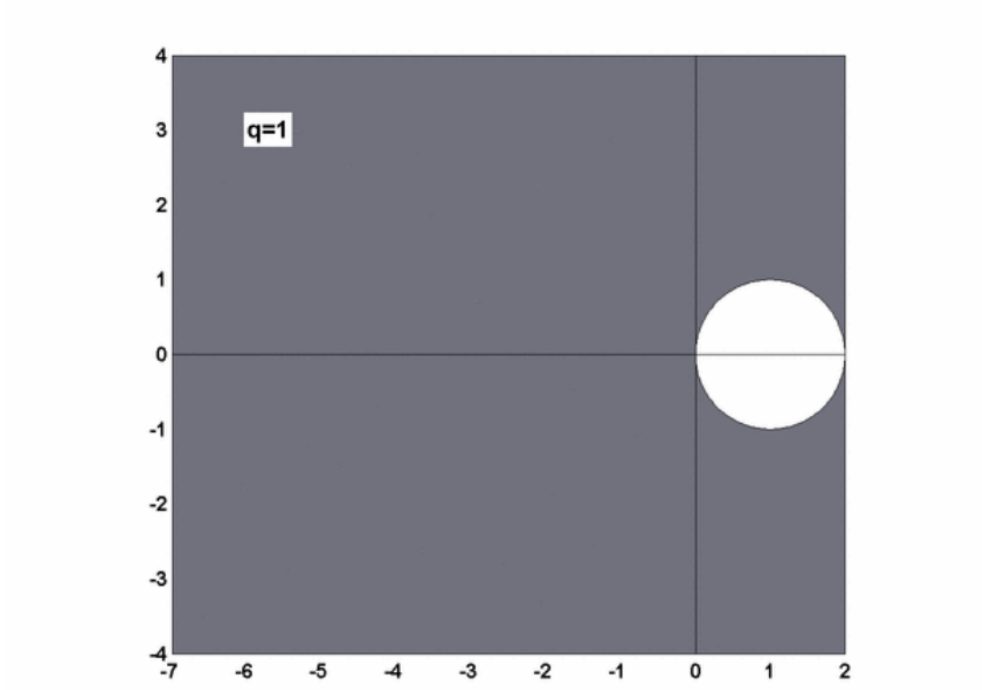
$$y_{n+1} = \alpha_1 y_n + \alpha_2 y_{n-1} + \ldots + h\left(\beta_0 f(y_{n+1}, t_{n+1}) + \beta_1 f(y_n, t_n) + \beta_2 f(y_{n-1}, t_{n-1}) + \ldots\right)$$

- Derived by fitting polynomials to previous steps

- Multi-step methods:
  - Adams-Bashforth (Explicit – $\beta_0 = 0$)
  - Adams-Moulton (Implicit)
    - PECE (Adams-Bashforth-Moulton)
  - Backward Differentiation Formula (BDF) (Implicit)
    - Numerical Differential Formula (NDF)

- Same stability concepts as for one-step RK methods apply

Stability region for Adams-Moulton



Differential-Algebraic Equations (DAE)

- ODE: $\dot{y} = f(y), \quad y(0) = y_0$
- DAE: $\dot{y} = f(y, z), \quad y(0) = y_0$
  $0 = g(y, z)$
- Advanced simulation tools (like those based on Modelica) in general generate DAEs

- If $\frac{\partial g(y,z)}{\partial z}$ is invertible [DAE is index 1] then
  $$0 = g(y, z) \text{ can be solved to } z = z(y)$$
  either symbolically (by hand/computer) or numerically

- Then the DAE can be written as ODE:
  $$\dot{y} = f(y, z(y)) = \tilde{f}(y)$$
  and ODE solvers (ERK, IRK, BDF, ...) can be used

Differential-Algebraic equations (DAE), II

- Some higher-order index systems ($\frac{\partial g(y,z)}{\partial z}$ not invertible) can reduced to index 1 [index reduction] by using certain tricks
  - And thereby be transformed to ODE system

- Not always possible *nor desirable* to solve DAE as ODE
  - Some numerical solvers can solve (low index) DAE problems directly (especially *implicit* solvers, who must solve nonlinear equations anyway)
  - See book for examples (IRK: 14.12.1, BDF: 14.12.2)

- DAE Software
  - Matlab: ode15s, ode23t
  - DASSL/DASPK
  - Sundials IDA
  - And others...

Lecture 11: Hydraulic motors, transmission lines

- Hydraulic motors
- Hydraulic transmission lines
- (Electrical transmission lines)

Book: 4.1-4.6, (1.6)

Hydraulic sylinder

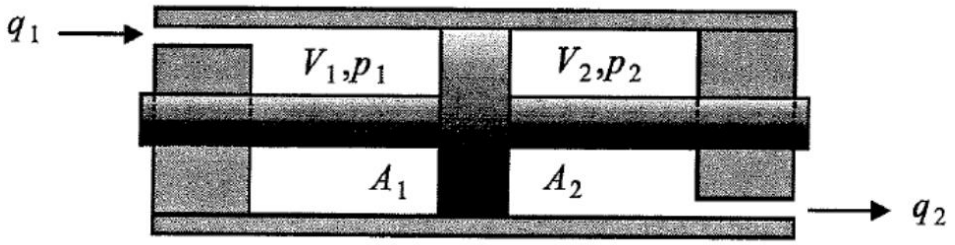


Figure 4.9: Symmetric hydraulic cylinder

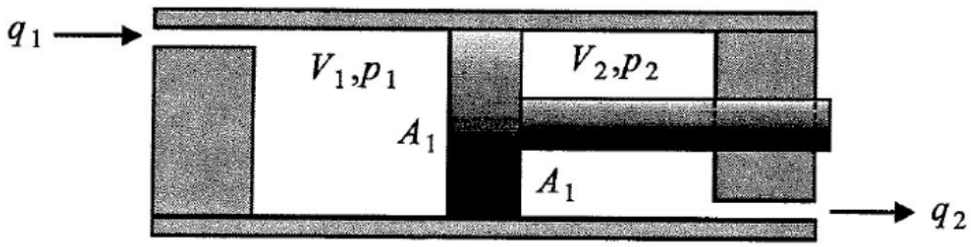


Figure 4.10: Single-rod hydraulic piston

Rotational hydraulic motor

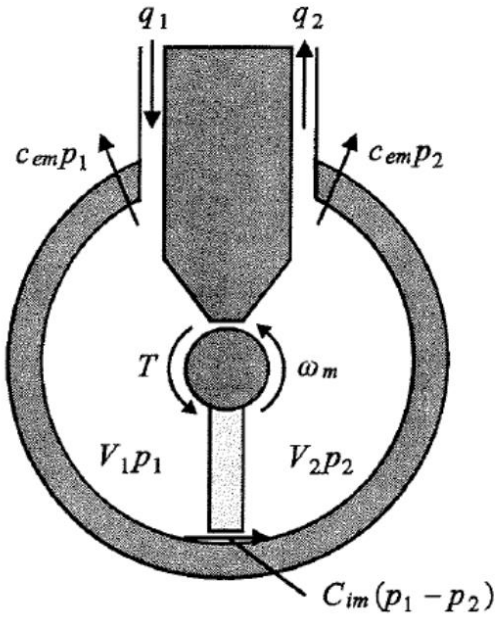
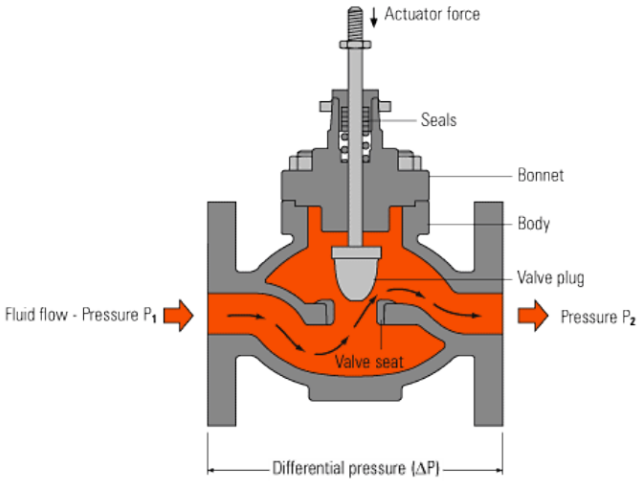
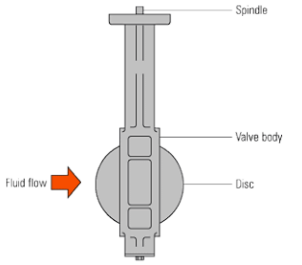


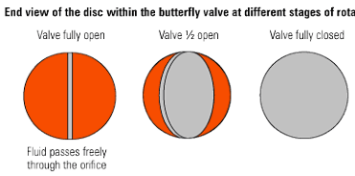
Figure 4.7: Rotational hydraulic motor of the single vane type with limited travel.

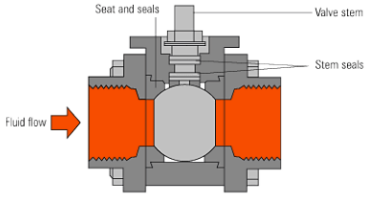
Valves

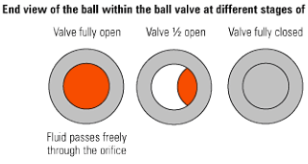
- Device that regulates flow
- Many different types of valves exist
  - Globe valve, ball valve, butterfly valve, ...









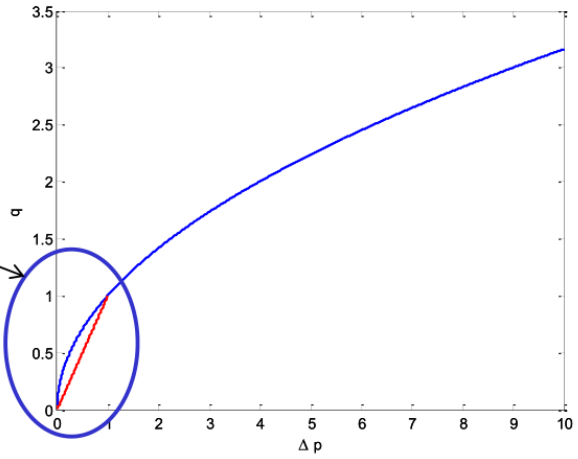


Valve models (book 4.2)

- Flow through a restriction is generally turbulent

$$q = C_d A \sqrt{\frac{2}{\rho} \Delta p}$$

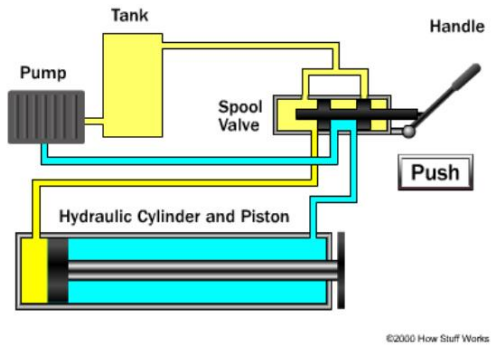
Problematic!



- Solution: Regularize by assuming laminar flow for small Δp

$$q = C_l \Delta p$$

- Book: Make transition smooth



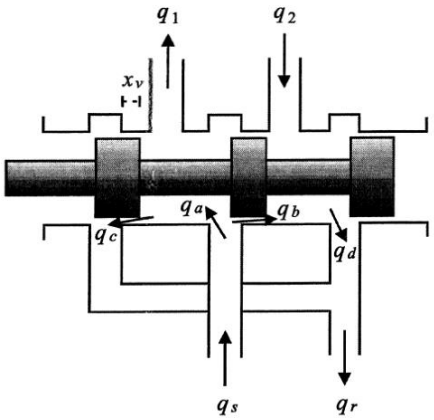
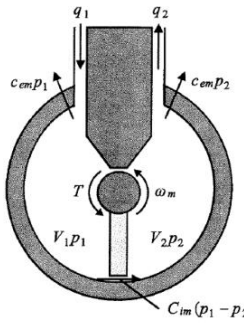


Figure 4.2: A matched and symmetric four-way valve.

Four-way valve



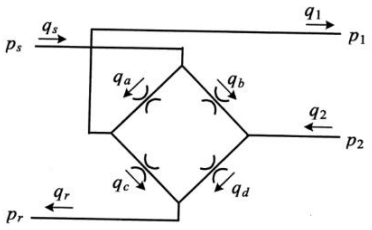


Figure 4.1: Four-way valve

Modeling of four-way valve

- Define load pressure

$$p_L = p_1 - p_2$$

- Define load flow

$$q_L = \frac{q_1 + q_2}{2}$$

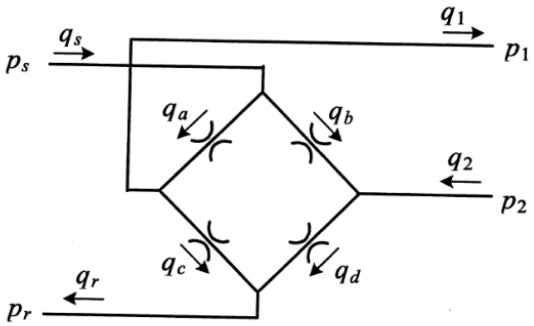


Figure 4.1: Four-way valve

- Symmetric load assumption (motor)

$$q_1 = q_2$$

- Symmetric valve and symmetric load

$$q_L = C_d b x_v \sqrt{\frac{1}{\rho}(p_s - \text{sign}(x_v)p_L)}$$

Characteristic of four-way valve

$$q_L = C_d b x_v \sqrt{\frac{1}{\rho}(p_s - \text{sign}(x_v)p_L)}$$

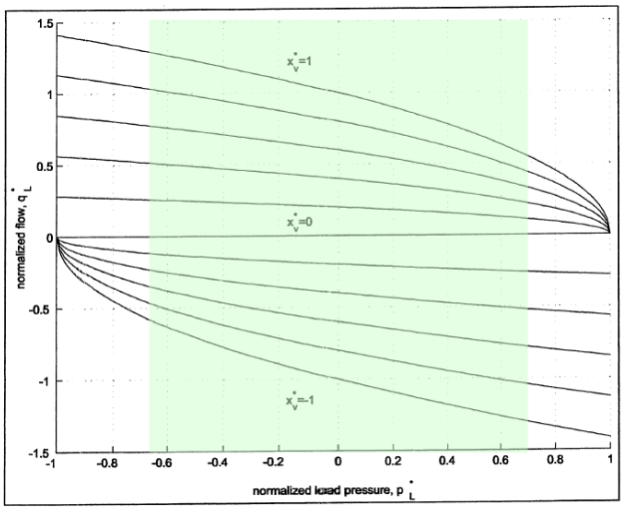


Figure 4.3: Valve characteristic

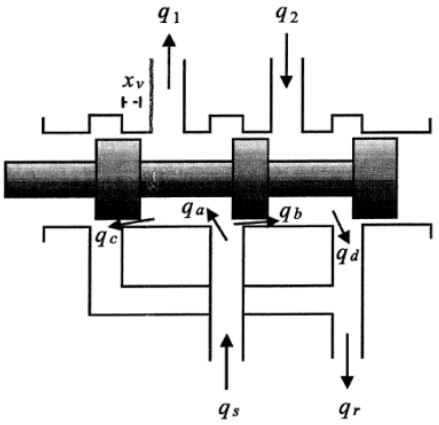
Linearized model:

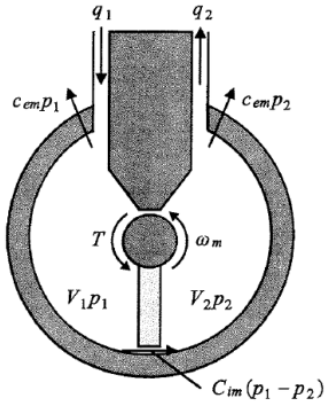
$$|p_L| \leq \frac{2}{3}p_s : \quad q_L = K_q x_v - K_c p_L$$

Gain uncertainty:

$$0.58K_{q0} \leq K_q \leq 1.29K_{q0}$$

Transfer function valve+motor

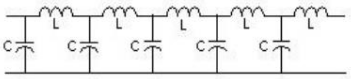




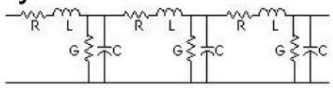
$$\theta_m(s) = \frac{\frac{K_q}{D_m}x_v(s) - \frac{K_{ce}}{D_m^2}\left(1+\frac{s}{\omega_t}\right)T_L(s)}{s\left(1+2\zeta_h\frac{s}{\omega_h}+\frac{s^2}{\omega_h^2}\right)}$$

Telegrapher's equation (Wave equation)

- Lossless:



Lossy:



- Model (Ch. 1.6):

$$\frac{\partial u(x,t)}{\partial x} = -Ri(x,t) - L\frac{\partial i(x,t)}{\partial t}$$
$$\frac{\partial i(x,t)}{\partial x} = -Gu(x,t) - C\frac{\partial u(x,t)}{\partial t}$$

- Laplace:

$$\frac{\partial u(x,s)}{\partial x} = -X(s)i(x,s)$$
$$\frac{\partial i(x,s)}{\partial x} = -Y(s)u(x,s)$$

Series impedance:
$X(s) = R + Ls$
Parallel admittance:
$Y(s) = G + Cs$

Characteristic impedance:
$$Z_c(s) = \sqrt{\frac{X(s)}{Y(s)}}$$

Same equations for electrical and fluid/hydraulical transmission lines

Electrical transmission lines:

$$\frac{\partial u(x,t)}{\partial x} = -Ri(x,t) - L\frac{\partial i(x,t)}{\partial t}$$
$$\frac{\partial i(x,t)}{\partial x} = -Gu(x,t) - C\frac{\partial u(x,t)}{\partial t}$$

Fluid transmission lines:

$$\frac{\partial p(x,t)}{\partial t} = -\frac{\beta}{A}\frac{\partial q(x,t)}{\partial x}$$
$$\frac{\partial q(x,t)}{\partial t} = -\frac{A}{\rho}\frac{\partial p(x,t)}{\partial x} - \frac{F\left[q(x,t)\right]}{\rho}$$

- Current and flow "same" variables, as is voltage and pressure
- In both cases, we can define line impedance, characteristic impedance, propagation operator, etc.
- Solution to equations have same structure/form: waves propagating back and forth

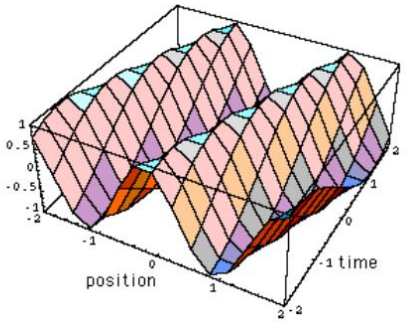
Solution: Waves

- Solution:

$$u_{out}(s) = e^{-\Gamma(s)}u_{in}(s)$$

- Propagation operator $\Gamma(s) = L\sqrt{X(s)Y(s)}$
  - Attenuation factor α = Re[$\Gamma(j\omega)$]: How much is wave reduced
  - Phase factor: β = Im[$\Gamma(j\omega)$]: How long does it take

- Lossless (R = G = 0): $\Gamma(s) = Ts$
  - Attenuation factor: 0
  - Phase factor: Pure time-delay



When should we care?

- Solution lossless case: Time delay

$$e^{-Ts}$$

- Rule-of-thumb from control theory: We can ignore time-delay for frequencies much less than 1/*T*

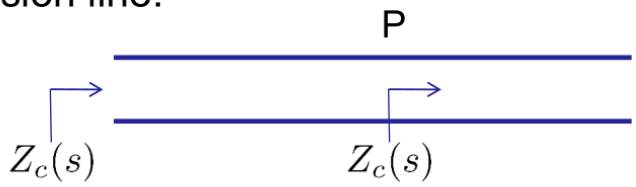
$$\omega \leq \frac{1}{T} \Rightarrow 2\pi\frac{c}{\lambda} \leq \frac{c}{L} \Rightarrow L \leq \frac{\lambda}{2\pi}$$

- Rule-of-thumb for transmission lines: When L is larger than one tenth of wavelength, treat as transmission line

- Power lines, *f* = 50Hz: λ = 6000km
- Personal computers, *f* = 10GHz: λ = 1.5cm

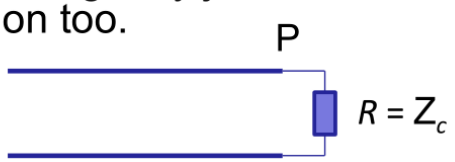
Impedance matching

- Suppose we have an imaginary joint at P in a very long transmission line.



The wave goes through the joint without reflection because there is actually no joint (just imagined).

- Now, let us terminate a resistance of value Z_c *at the same position of this* imaginary joint. The wave will go through without reflection too.



This is called a **matched load.**

Lecture 12: Friction

- What is friction?
- Why is it important to know about friction for a control engineer?
- Static friction models
- Dynamic friction models

Book: Ch. 5

Static friction models

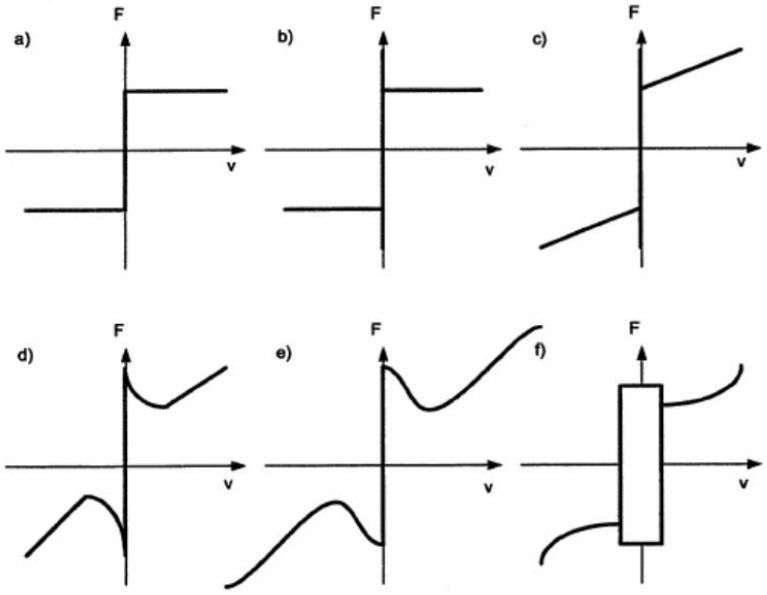


Figure 5.3: Static friction models: a) Colomb friction b) Coulomb+stiction c) Coulomb+stiction+viscous d) Stribeck effect e) Hess and Soom; Armstrong f) Karnopp model

Dynamic friction models

The Dahl model

$$\frac{\mathrm{d}F}{\mathrm{d}t} = \sigma\left(v - |v|\frac{F}{F_c}\right)$$

The LuGre model

$$F = \sigma_0 z + \sigma_1 \frac{\mathrm{d}z}{\mathrm{d}t} + \sigma_2 v$$

$$\frac{\mathrm{d}z}{\mathrm{d}t} = v - \sigma_0 \frac{|v|}{g(v)} z$$

$$g(v) = F_c + (F_s - F_c)e^{-\left(\frac{v}{v_s}\right)^2}$$

Why dynamic friction models?

- Easier to simulate
- Easier to analyze
- They reproduce (to some extent) dynamic friction phenomena
  - Presliding displacement
    - friction force act as a spring in sticking region
  - Frictional lag
    - Dynamic friction force depends on direction of velocity
  - Varying break-away force
    - Break-away force depends on rate-of-change of applied force

Generalized Stribeck curve

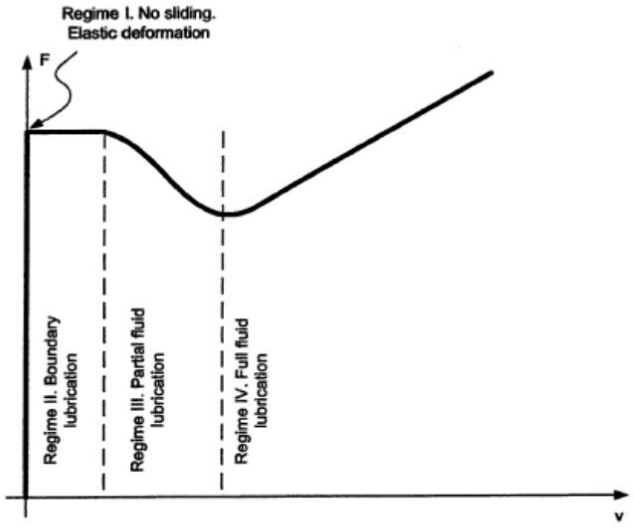
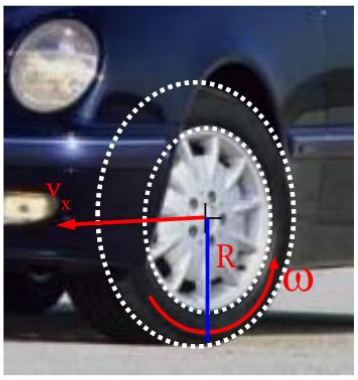


Figure 5.2: The generalized Stribeck curve, showing friction as a function of velocity for low velocities, (Armstrong-Hélouvry et al. 1994).

Slipp – relativ hastighetsforskjell

- I langsretning:

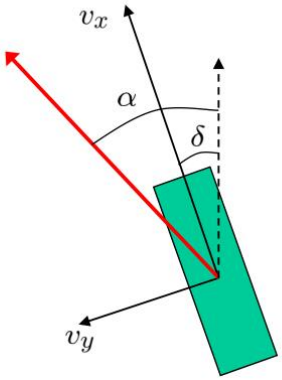
$$\lambda_x := \frac{v_x - R\omega}{v_x}$$



- I sideretning:

$$\lambda_y := \sin\alpha$$

$$\alpha := \delta + \arctan\frac{v_y}{v_x}$$



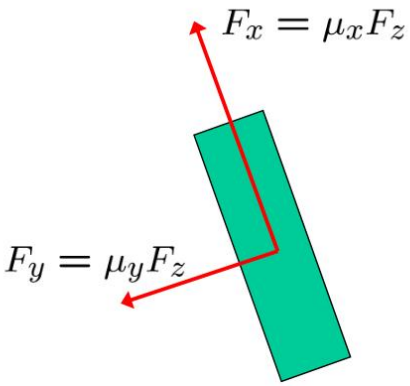
Friksjonskrefter

Coloumbs lov:

- Friksjonskrefter gitt av vertikale krefter og friksjonskoeffisient

- Friksjonskoeffisient gitt av slipp og underlag

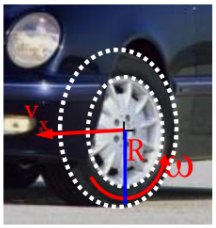
$$\mu_x \approx \mu_x(\lambda_x, \lambda_y, \mu_H)$$
$$\mu_y \approx \mu_y(\lambda_y, \lambda_x, \mu_H)$$

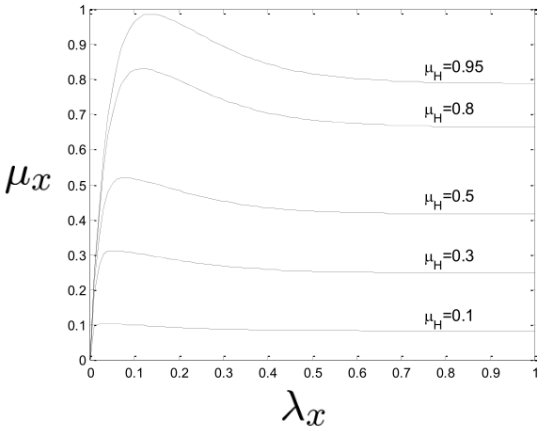


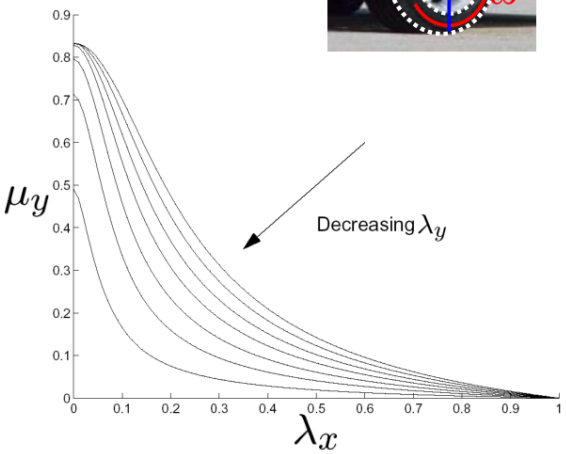
Friksjonskoeffisienter under bremsing

- Bremsing reduserer hjulhastighet i forhold til bilhastighet

$$\lambda_x := \frac{v_x - R\omega}{v_x}$$

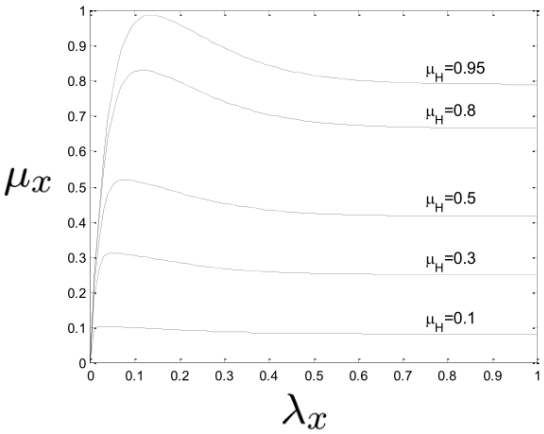


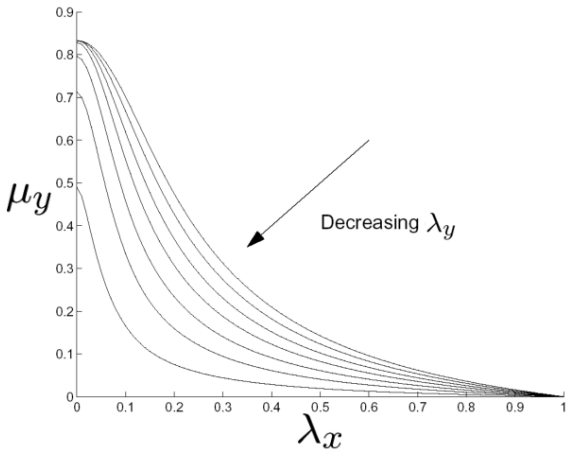




Blokkeringsfrie bremser – ABS

- Ønsker konstant lav slipp under bremsing fordi
  - Det gjør bremsing mest effektivt
  - Kan styre bilen under bremsing

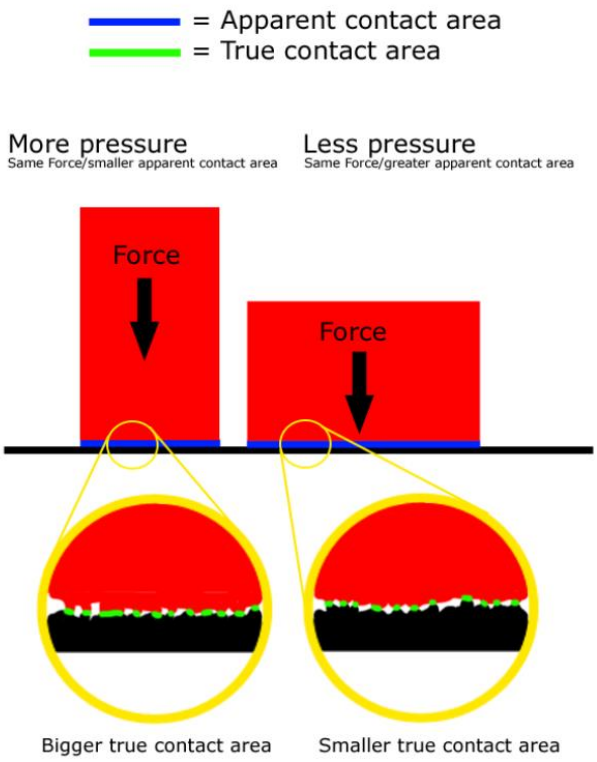




Dry friction is

- Independent of area
  - Da Vinci
- Proportional to normal force
  - Amonton, Euler
- Independent of velocity
  - Coloumb

$$F = \mu F_N$$



Lecture 13: Rigid body kinematics – vectors, dyadics, rotation matrices

- What is rigid body kinematics?
- Vectors and dyadics
- Rotations

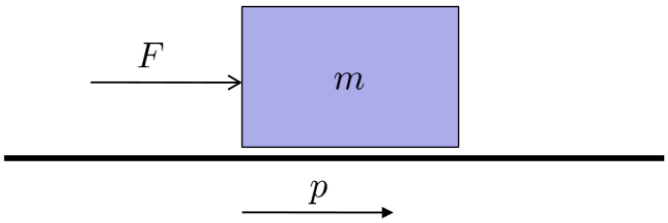
Book: Ch. 6.2, 6.3, 6.4

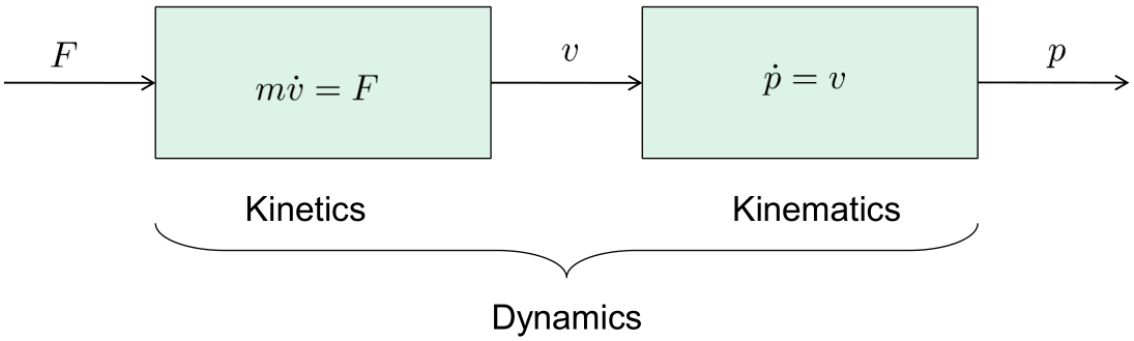
What is rigid body dynamics?

- Rigid body:
  - Wikipedia: "…a rigid body is an idealization of a solid body of finite size in which deformation is neglected."
- Dynamics = Kinematics + Kinetics
- Kinematics
  - eb.com: "…branch of physics (…) concerned with the geometrically possible motion of a body or system of bodies without consideration of the forces involved (i.e., causes and effects of the motions)."
  - Book: Ch. 6
- Kinetics
  - eb.com: "…the effect of forces and torques on the motion of bodies having mass."
  - Book: Ch. 7, 8.

Remark: Sometimes "dynamics" is used for "kinetics" only

Simplest scalar case





	Kinematics	Kinetics
	Derivatives of position and orientation as function of velocity and angular velocity	Derivatives of velocity and angular velocity as function of applied forces and torques
Translation	1D: $\dot{r} = v$ 3D: $\dot{\mathbf{r}}_c^i = \mathbf{v}_c^i$	1D: $m\dot{v} = F$ 3D: $m\dot{\mathbf{v}}_c^i = \mathbf{F}_{bc}^i$
Note! By definition $\vec{v}_c := \frac{^i\mathrm{d}}{\mathrm{d}t}\vec{r}_c$	$\dot{\mathbf{r}}_c^i = \mathbf{v}_c^i = \mathbf{R}_b^i \mathbf{v}_c^b$	Usually convenient to have forces and velocities in body system: $m\left(\dot{\mathbf{v}}_c^b + \left(\boldsymbol{\omega}_{ib}^b\right)^\times \mathbf{v}_c^b\right) = \mathbf{F}_{bc}^b$
Rotation/ orientation	1D: $\dot{\theta} = \omega$ 3D: Depends on parameterization Rotation matrix: $\dot{\mathbf{R}}_b^i = \mathbf{R}_b^i \left(\boldsymbol{\omega}_{ib}^b\right)^\times$ Euler angles: $\dot{\boldsymbol{\phi}} = \mathbf{E}_d^{-1}(\boldsymbol{\phi})\boldsymbol{\omega}_{ib}^b$ Euler parameters: $\dot{\eta} = -\frac{1}{2}\boldsymbol{\epsilon}^\top \boldsymbol{\omega}_{ib}^b$ $\dot{\boldsymbol{\epsilon}} = \frac{1}{2}\left(\eta\mathbf{I} + \boldsymbol{\epsilon}^\times\right)\boldsymbol{\omega}_{ib}^b$	1D: $J\dot{\omega} = T$ 3D: $\mathbf{M}_{b/c}^b \dot{\boldsymbol{\omega}}_{ib}^b + \left(\boldsymbol{\omega}_{ib}^b\right)^\times \mathbf{M}_{b/c}^b \boldsymbol{\omega}_{ib}^b = \mathbf{T}_{bc}^b$

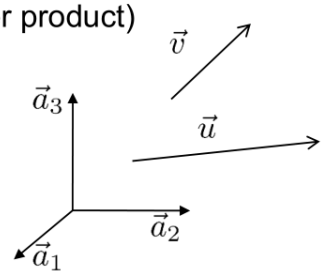
The scalar product (dot product, inner product)

Vectors:

$$\vec{u} = u_1\vec{a}_1 + u_2\vec{a}_2 + u_3\vec{a}_3$$
$$\vec{v} = v_1\vec{a}_1 + v_2\vec{a}_2 + v_3\vec{a}_3$$

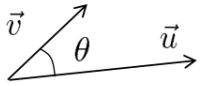
Coordinate vectors:

$$\mathbf{u} = \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} \quad \mathbf{v} = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix}$$



- Definition of scalar product:

$$\vec{u} \cdot \vec{v} = |\vec{u}||\vec{v}| \cos\theta$$

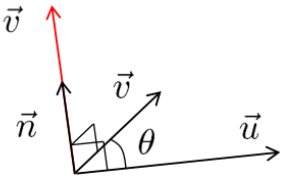


- Can also be calculated from coordinate-vectors:

$$\begin{aligned} \vec{u} \cdot \vec{v} &= (u_1\vec{a}_1 + u_2\vec{a}_2 + u_3\vec{a}_3) \cdot (v_1\vec{a}_1 + v_2\vec{a}_2 + v_3\vec{a}_3) \\ &= u_1v_1 + u_2v_2 + u_3v_3 = \mathbf{u}^{\mathsf{T}}\mathbf{v} \end{aligned}$$

The cross product

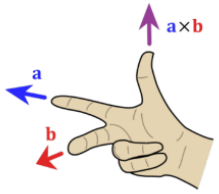
$$\vec{w} = \vec{u} \times \vec{v}$$



- Definition:

$$\vec{w} = \vec{u} \times \vec{v} = \vec{n}|\vec{u}||\vec{v}| \sin\theta$$

- Calculation:



$$\vec{w} = \vec{u} \times \vec{v} = \begin{vmatrix} \vec{a}_1 & \vec{a}_2 & \vec{a}_3 \\ u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \end{vmatrix}$$
$$= (u_2v_3 - u_3v_2)\vec{a}_1 - (u_3v_1 - u_1v_3)\vec{a}_2 + (u_1v_2 - u_2v_1)\vec{a}_3$$

- Introduce *the skew-symmetric form* of vector ***u***

$$\mathbf{u}^{\times} = \begin{pmatrix} 0 & -u_3 & u_2 \\ u_3 & 0 & -u_1 \\ -u_2 & u_1 & 0 \end{pmatrix}$$

- Easy to check that

$$\mathbf{w} = \mathbf{u}^{\times}\mathbf{v} \quad \Leftrightarrow \quad \vec{w} = \vec{u} \times \vec{v}$$

Lecture 14: Rigid body kinematics – the rotation matrix

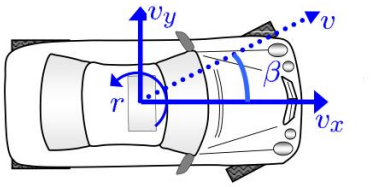
- What are rotation matrices used for?

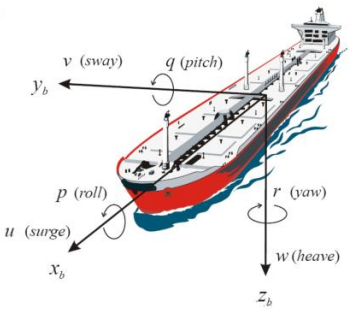
- Rotation matrices
  - Composite rotations, simple rotations
  - Homogenous transformation matrices
- Euler angles
  - 3-parameter specification of rotations
  - Roll-pitch-yaw
- Angle-axis
  - 4-parameter specification of rotations

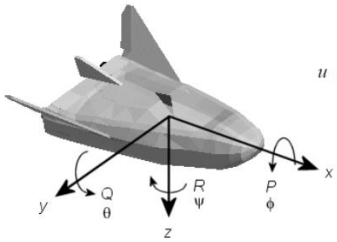
Book: Ch. 6.4, 6.5, 6.6

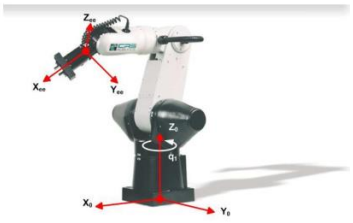
Why rotation matrices?

- Rotation matrices are used to describe rotations and orientations of rigid bodies
  - Road vehicles
  - Marine vessels
  - Airplanes, satellites
  - Robotics

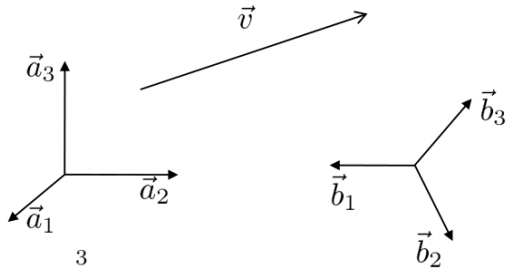








Rotation matrices



- The vector $\vec{v}$ can be written as

$$\vec{v} = \sum_{j=1}^{3} v_j^a \vec{a}_j \qquad \text{or} \qquad \vec{v} = \sum_{j=1}^{3} v_j^b \vec{b}_j$$

- These must be the same:

$$\sum_{j=1}^{3} v_j^a \vec{a}_j = \sum_{j=1}^{3} v_j^b \vec{b}_j$$

- Scalar product with $\vec{a}_i$ on both sides:

$$\sum_{j=1}^{3} v_j^a \vec{a}_j \cdot \vec{a}_i = \sum_{j=1}^{3} v_j^b \vec{b}_j \cdot \vec{a}_i \quad \Rightarrow \quad v_i^a = \sum_{j=1}^{3} v_j^b \vec{a}_i \cdot \vec{b}_j$$

- Gives

$$\mathbf{v}^a = \begin{pmatrix} v_1^a \\ v_2^a \\ v_3^a \end{pmatrix} = \begin{pmatrix} \vec{a}_1 \cdot \vec{b}_1 & \vec{a}_1 \cdot \vec{b}_2 & \vec{a}_1 \cdot \vec{b}_3 \\ \vec{a}_2 \cdot \vec{b}_1 & \vec{a}_2 \cdot \vec{b}_2 & \vec{a}_2 \cdot \vec{b}_3 \\ \vec{a}_3 \cdot \vec{b}_1 & \vec{a}_3 \cdot \vec{b}_2 & \vec{a}_3 \cdot \vec{b}_3 \end{pmatrix} \begin{pmatrix} v_1^b \\ v_2^b \\ v_3^b \end{pmatrix} = \mathbf{R}_b^a \mathbf{v}^b$$

Rotation matrices, properties

- We have shown

$$\mathbf{v}^a = \begin{pmatrix} v_1^a \\ v_2^a \\ v_3^a \end{pmatrix} = \begin{pmatrix} \vec{a}_1 \cdot \vec{b}_1 & \vec{a}_1 \cdot \vec{b}_2 & \vec{a}_1 \cdot \vec{b}_3 \\ \vec{a}_2 \cdot \vec{b}_1 & \vec{a}_2 \cdot \vec{b}_2 & \vec{a}_2 \cdot \vec{b}_3 \\ \vec{a}_3 \cdot \vec{b}_1 & \vec{a}_3 \cdot \vec{b}_2 & \vec{a}_3 \cdot \vec{b}_3 \end{pmatrix} \begin{pmatrix} v_1^b \\ v_2^b \\ v_3^b \end{pmatrix} = \mathbf{R}_b^a \mathbf{v}^b$$

- Switching *a* and *b*, we obtain

$$\mathbf{v}^b = \begin{pmatrix} v_1^b \\ v_2^b \\ v_3^b \end{pmatrix} = \begin{pmatrix} \vec{b}_1 \cdot \vec{a}_1 & \vec{b}_1 \cdot \vec{a}_2 & \vec{b}_1 \cdot \vec{a}_3 \\ \vec{b}_2 \cdot \vec{a}_1 & \vec{b}_2 \cdot \vec{a}_2 & \vec{b}_2 \cdot \vec{a}_3 \\ \vec{b}_3 \cdot \vec{a}_1 & \vec{b}_3 \cdot \vec{a}_2 & \vec{b}_3 \cdot \vec{a}_3 \end{pmatrix} \begin{pmatrix} v_1^a \\ v_2^a \\ v_3^a \end{pmatrix} = \mathbf{R}_a^b \mathbf{v}^a$$

- We see that $\mathbf{R}_a^b = (\mathbf{R}_b^a)^{\mathsf{T}}$

- From $\mathbf{v}^a = \mathbf{R}_b^a \mathbf{v}^b = \mathbf{R}_b^a \mathbf{R}_a^b \mathbf{v}^a$, we see that $\mathbf{R}_b^a \mathbf{R}_a^b = \mathbf{I}$

$$\mathbf{R}_a^b = (\mathbf{R}_b^a)^{\mathsf{T}} = (\mathbf{R}_b^a)^{-1}$$

The set of rotation matrices

For a matrix $\mathbf{R}$ to be a rotation matrix:

- The matrix must be orthogonal:

$$\mathbf{R}\mathbf{R}^{\mathsf{T}} = \mathbf{I}$$

- The determinant must be one

$$\det \mathbf{R} = 1$$

- The set of these matrices has a name: SO(3), or Special Orthogonal group of order 3:

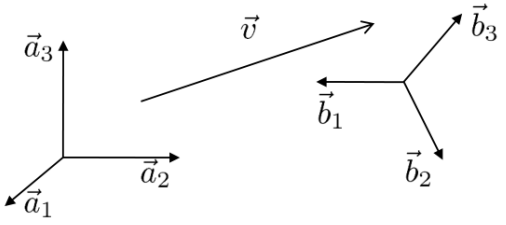
$$\mathrm{SO}(3) = \{\mathbf{R} \in \mathbb{R}^{3\times 3} \mid \mathbf{R}^{\mathsf{T}}\mathbf{R} = \mathbf{I},\ \det \mathbf{R} = 1\}$$

Rotation matrices

The rotation matrix from *a* to *b* $\mathbf{R}_b^a$ is used to

- **Transform** a coordinate vector from *b* to *a*

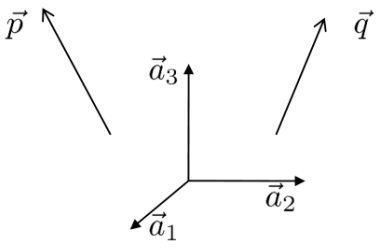
$$\mathbf{v}^a = \mathbf{R}_b^a \mathbf{v}^b$$



- **Rotate** a vector $\vec{p}$ to vector $\vec{q}$. If decomposed in *a*,

$$\mathbf{q}^a = \mathbf{R}_b^a \mathbf{p}^a$$

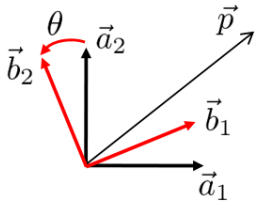
such that $\mathbf{q}^b = \mathbf{p}^a$.



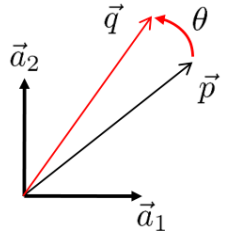
Rotation vs transformation (same, again)

- A coordinate vector may change either as a result of a rotation of a coordinate system (a **coordinate transformation**) or a rotation of the vector itself (a **rotation**).

- That is, a rotation from *a* to *b* can be interpreted in two ways:



$\mathbf{p}^b = \mathbf{R}_a^b \mathbf{p}^a$ (or $\mathbf{p}^a = \mathbf{R}_b^a \mathbf{p}^b$)

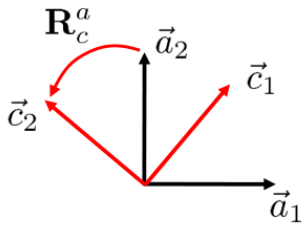


$\mathbf{q}^a = \mathbf{R}_b^a \mathbf{p}^a$ such that $\mathbf{q}^b = \mathbf{p}^a$

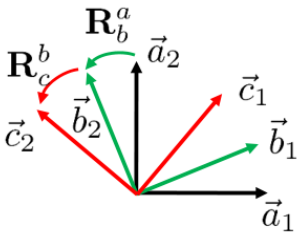
- That is, the matrix $\mathbf{R}_b^a$ rotates from *a* to *b*, but transforms from *b* to *a*!

- (Sometimes these two interpretations of the rotations originating from a rotation matrix are called passive vs active transformations, or alias vs alibi transformations)

Composite rotations



$\mathbf{v}^a = \mathbf{R}_c^a \mathbf{v}^c$



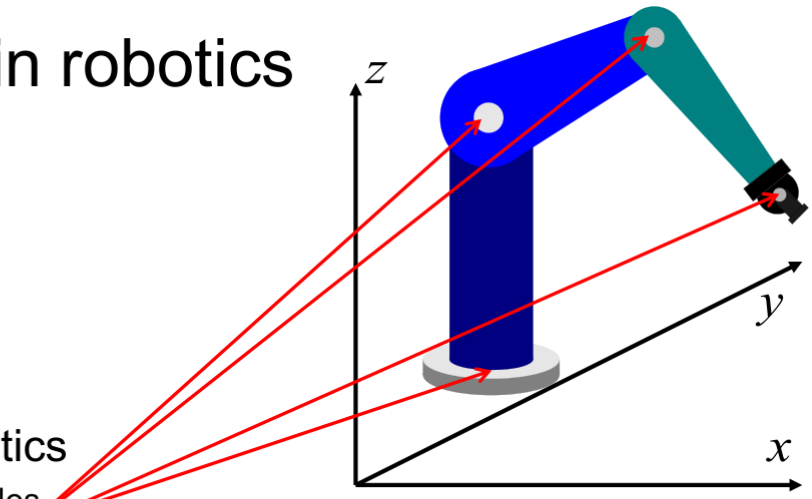
$\mathbf{v}^b = \mathbf{R}_c^b \mathbf{v}^c$

$\mathbf{v}^a = \mathbf{R}_b^a \mathbf{v}^b = \mathbf{R}_b^a \mathbf{R}_c^b \mathbf{v}^c$

$$\mathbf{R}_c^a = \mathbf{R}_b^a \mathbf{R}_c^b$$

(and $\mathbf{R}_d^a = \mathbf{R}_b^a \mathbf{R}_c^b \mathbf{R}_d^c$, etc.)

Kinematics in robotics

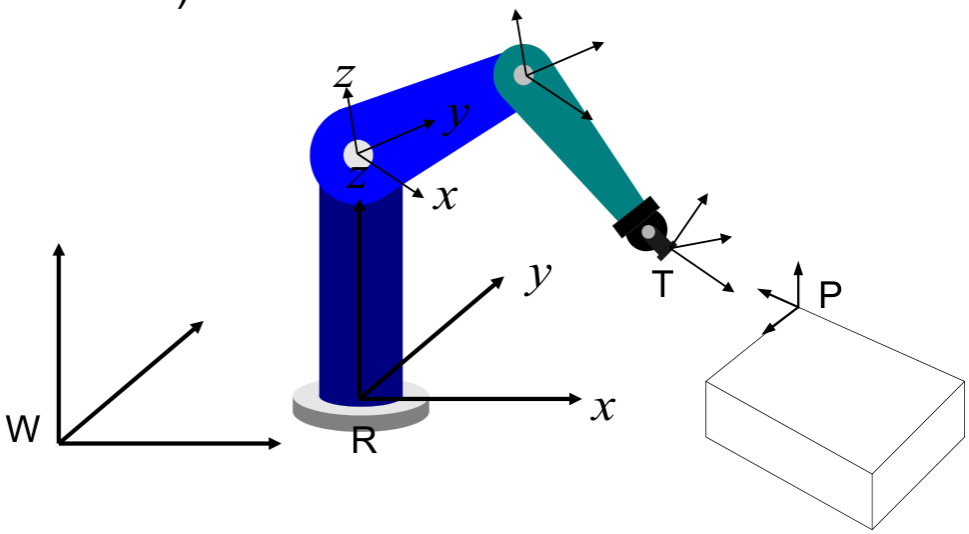


- Forward kinematics
  - Given joint variables
    $$q = (q_1, q_2, q_3, \ldots, q_n)$$
  - What are end-effector position and orientation?

- Inverse kinematics
  - Given (desired) end-effector position and orientation.
  - What are the corresponding joint variables?

Coordinate systems in robotics

- World frame
- Joint frame
- Tool (end-effector) frame



Representations of rotations

- Rotation matrix
  - Simple, but over-parameterized (9 parameters)

> Euler's Theorem:
> "Any two independent orthonormal coordinate frames can be related by a sequence of rotations (not more than three) about coordinate axes, where no two successive rotations may be about the same axis."

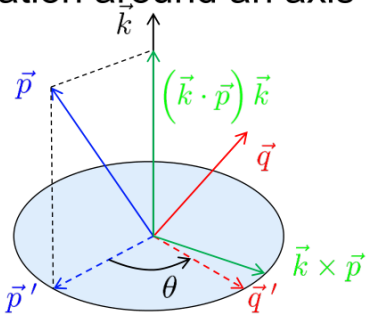
- Three rotations about axes are enough to specify any rotation
  - These representations are called Euler angles
    - 12 different combinations possible
    - Most common: Roll-pitch-yaw
  - Natural and (in many cases) simple to use, very much used
  - Problem: Singularity (more on this later)

- Angle-axis, Euler-parameters
  - 4-parameters are used
  - No singularity problems

Rotation of vectors based on angle-axis representation

- Angle-axis: All rotations can be represented as a simple rotation around an axis

Somewhat different derivation of the rotation dyadic. Compare p. 228 in book.



$$\vec{p}\,' = \vec{p} - (\vec{k}\cdot\vec{p})\,\vec{k}$$

$$\vec{q}\,' = \vec{q} - (\vec{k}\cdot\vec{q})\,\vec{k} = \vec{q} - (\vec{k}\cdot\vec{p})\,\vec{k}$$

$$\vec{q}\,' = \cos\theta\,\vec{p}\,' + \sin\theta\,\vec{k}\times\vec{p}$$

$$\vec{q} - (\vec{k}\cdot\vec{p})\,\vec{k} = \cos\theta\,\left(\vec{p} - (\vec{k}\cdot\vec{p})\,\vec{k}\right) + \sin\theta\,\vec{k}\times\vec{p}$$

$$\vec{q} = \cos\theta\,\vec{p} + \sin\theta\,\vec{k}\times\vec{p} + (1-\cos\theta)\,(\vec{k}\cdot\vec{p})\,\vec{k}$$

Lecture 15: Rigid body kinematics – Rotations, angular velocity

Representations of rotation

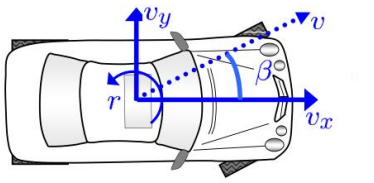
- Rotation matrices (before Easter)
- Euler angles (before Easter)
  - 3-parameter specification of rotations
  - Roll-pitch-yaw
- Angle-axis, Euler-parameters
  - 4-parameter specification of rotations

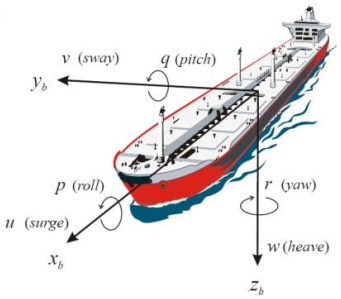
- Angular velocity

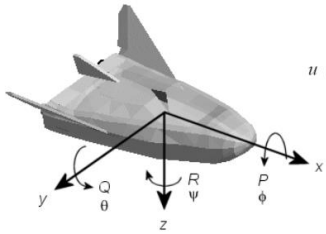
Book: Ch. 6.6, 6.7, 6.8

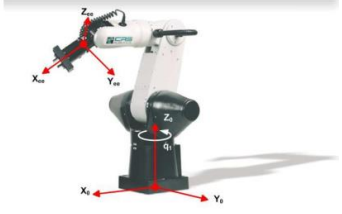
Why rotation matrices?

- Rotation matrices are used to describe rotations and orientations of rigid bodies
  - Road vehicles
  - Marine vessels
  - Airplanes, satellites
  - Robotics







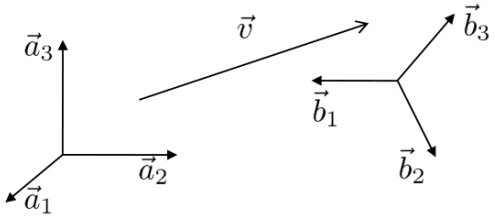


Rotation matrices

The rotation matrix from *a* to *b* $\mathbf{R}_b^a$ is used to

- Transform a coordinate vector from *b* to *a*

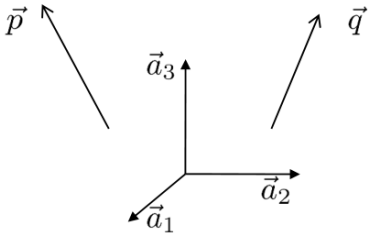
$$\mathbf{v}^a = \mathbf{R}_b^a \mathbf{v}^b$$



- Rotate a vector $\vec{p}$ to vector $\vec{q}$. If decomposed in *a*,

$$\mathbf{q}^a = \mathbf{R}_b^a \mathbf{p}^a$$

such that $\mathbf{q}^b = \mathbf{p}^a$.



Representations of rotations

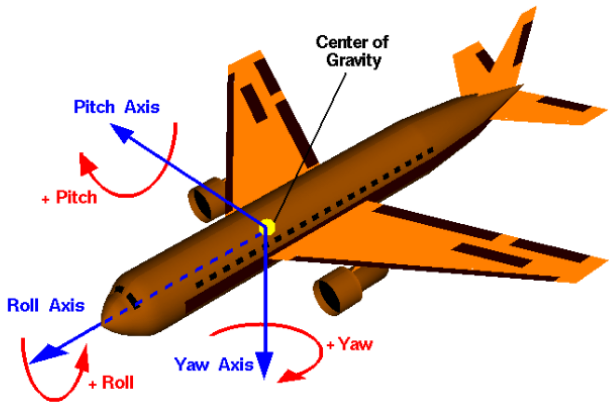
- Rotation matrix
  - Simple, but over-parameterized (9 parameters)

> Euler's Theorem:
> "Any two independent orthonormal coordinate frames can be related by a sequence of rotations (not more than three) about coordinate axes, where no two successive rotations may be about the same axis."

- Three rotations about axes are enough to specify any rotation
  - These representations are called Euler angles
    - 12 different combinations possible
    - Most common: Roll-pitch-yaw
  - Natural and (in many cases) simple to use, very much used
  - Problem: Singularity (more on this later)

- Angle-axis, Euler-parameters
  - 4-parameters are used
  - No singularity problems

Euler-angles: Roll-pitch-yaw



- Rotation ψ about *z*-axis, θ about (rotated) *y*-axis, ϕ about (rotated) *x*-axis

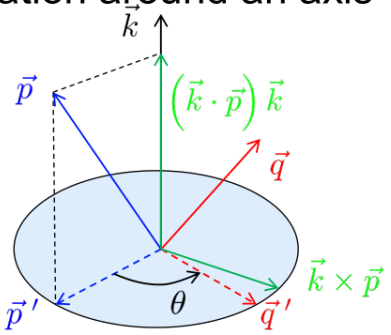
$$\mathbf{R}_b^a = \mathbf{R}_{z,\psi}\mathbf{R}_{y,\theta}\mathbf{R}_{x,\phi}$$

$$\mathbf{R}_b^a = \begin{pmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{pmatrix}$$

Rotation of vectors based on angle-axis representation

- Angle-axis: All rotations can be represented as a simple rotation around an axis

Somewhat different derivation of the rotation dyadic. Compare p. 228 in book.



$$\vec{p}\,' = \vec{p} - \left(\vec{k}\cdot\vec{p}\right)\vec{k} \qquad \vec{q}\,' = \vec{q} - \left(\vec{k}\cdot\vec{q}\right)\vec{k} = \vec{q} - \left(\vec{k}\cdot\vec{p}\right)\vec{k}$$

$$\vec{q}\,' = \cos\theta\,\vec{p}\,' + \sin\theta\,\vec{k}\times\vec{p}$$

$$\vec{q} - \left(\vec{k}\cdot\vec{p}\right)\vec{k} = \cos\theta\,\left(\vec{p} - \left(\vec{k}\cdot\vec{p}\right)\vec{k}\right) + \sin\theta\,\vec{k}\times\vec{p}$$

$$\vec{q} = \cos\theta\,\vec{p} + \sin\theta\,\vec{k}\times\vec{p} + (1-\cos\theta)\left(\vec{k}\cdot\vec{p}\right)\vec{k}$$

Angle-axis rotation dyadic, rotation matrix

- Rotation θ about an axis $\vec{k}$

$$\vec{q} = \cos\theta\ \vec{p} + \sin\theta\ \vec{k} \times \vec{p} + (1 - \cos\theta)\ \vec{k}\left(\vec{k} \cdot \vec{p}\right)$$

- Angle-axis rotation by a dyadic

$$\vec{q} = \Big(\underbrace{\cos\theta\ \vec{I} + \sin\theta\ \vec{k}^{\times} + (1 - \cos\theta)\ \vec{k}\vec{k}}_{\vec{R}_{\vec{k},\theta}}\Big) \cdot \vec{p}$$

$$\vec{q} = \vec{R}_{\vec{k},\theta} \cdot \vec{p}$$

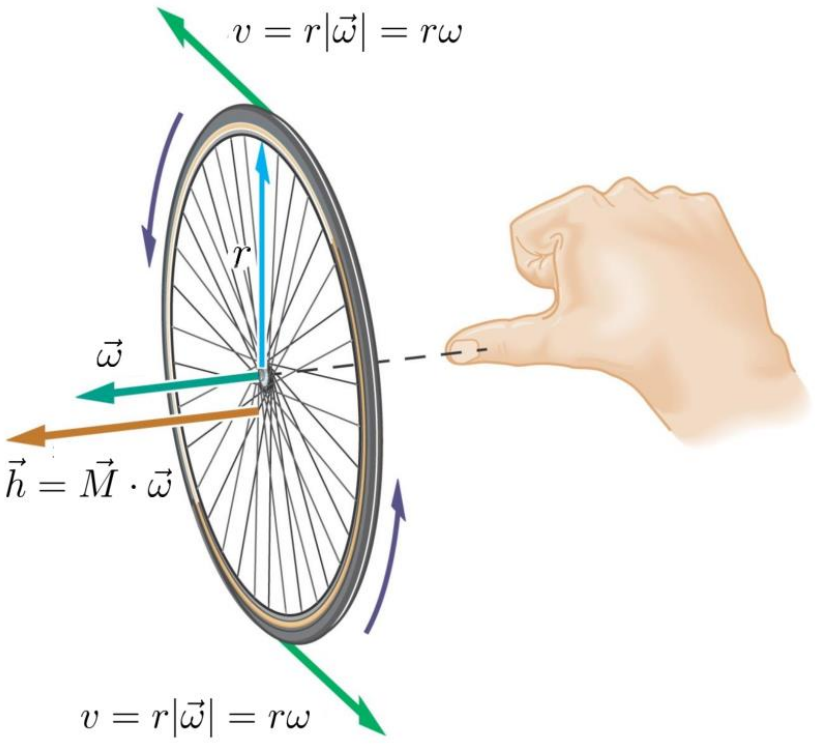
- Angle-axis rotation matrix

$$\mathbf{R}_b^a = \mathbf{R}_{\mathbf{k},\theta} = \cos\theta\ \mathbf{I} + \sin\theta\ (\mathbf{k}^a)^{\times} + (1 - \cos\theta)\ \mathbf{k}^a(\mathbf{k}^a)^{\mathsf{T}}$$

- Alternative expression (using $\mathbf{k}^a = \mathbf{k}$ and $\mathbf{k}^{\times}\mathbf{k}^{\times} = \mathbf{k}(\mathbf{k})^{\mathsf{T}} - \mathbf{I}$):

$$\mathbf{R}_b^a = \mathbf{R}_{\mathbf{k},\theta} = \mathbf{I} + \sin\theta\ \mathbf{k}^{\times} + (1 - \cos\theta)\ \mathbf{k}^{\times}\mathbf{k}^{\times}$$

Angular velocity



Lecture 16: Rigid body kinematics – Kinematic differential equations

- Brief recap of representations of rotation
  - Rotation matrices (6.4)
  - Euler angles (6.5)
    - 3-parameter representation of rotations
    - Roll-pitch-yaw
  - Angle-axis, Euler-parameters (6.6, 6.7)
    - 4-parameter representation of rotations
  - Angular velocity (6.8)

- Today:
  - Kinematic differential equations
  - Rigid body kinematics: Configuration

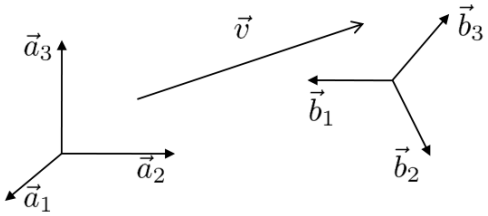
Book: Ch. 6.9, 6.12, 6.13

Rotation matrices

The rotation matrix from *a* to *b* $\mathbf{R}_b^a$ is used to

- **Transform** a coordinate vector from *b* to *a*

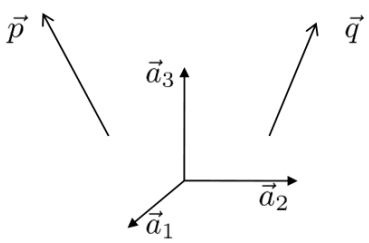
$$\mathbf{v}^a = \mathbf{R}_b^a \mathbf{v}^b$$



- **Rotate** a vector $\vec{p}$ to vector $\vec{q}$. If decomposed in *a*,

$$\mathbf{q}^a = \mathbf{R}_b^a \mathbf{p}^a$$

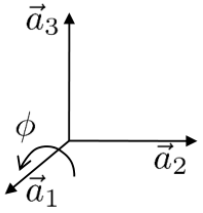
such that $\mathbf{q}^b = \mathbf{p}^a$.



Simple rotations

- Simple rotation = rotation about an axis
- Example: Rotation matrix for rotation about *x*-axis:

$$\mathbf{R}_{x,\phi} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{pmatrix}$$

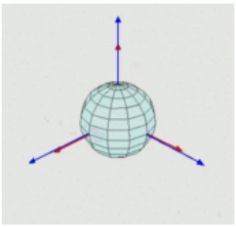


Representations of rotations

- Rotation matrix
  - Easy to use, but not to visualize (also over-parameterized, 9 parameters)

> Euler's Theorem:
> "Any two independent orthonormal coordinate frames can be related by a sequence of rotations (not more than three) about coordinate axes, where no two successive rotations may be about the same axis."

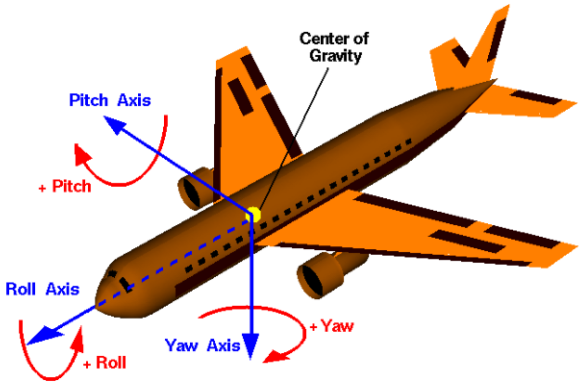
- Three rotations about axes are enough to specify any rotation
  - These representations are called Euler angles
    - 12 different combinations possible
    - Most common(?): Roll-pitch-yaw
  - Natural and (in many cases) simple to use, very much used
  - Problem: Singularity (more on this today)



Source: Wikipedia

- Angle-axis, Euler-parameters
  - 4-parameter representations of rotations
  - No singularity problems

Euler-angles: Roll-pitch-yaw



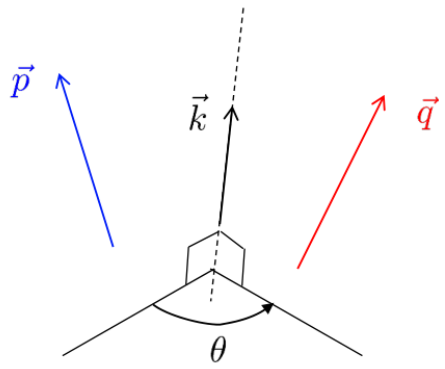
- Rotation ψ about *z*-axis, θ about (rotated) *y*-axis, ϕ about (rotated) *x*-axis

$$\mathbf{R}_b^a = \mathbf{R}_{z,\psi}\mathbf{R}_{y,\theta}\mathbf{R}_{x,\phi}$$

$$\mathbf{R}_b^a = \begin{pmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{pmatrix}$$

Angle-axis representation of rotations

All rotations can be represented as a simple rotation around an axis



- Angle-axis parameters:
  - Coordinate free: $\vec{k}$, θ

$$\vec{q} = \underbrace{\left(\cos\theta\ \vec{I} + \sin\theta\ \vec{k}^{\times} + (1-\cos\theta)\,\vec{k}\vec{k}\right)}_{\vec{R}_{\vec{k},\theta}} \cdot \vec{p}$$

  - With coordinates: $\mathbf{k}^a$, θ

$$\mathbf{R}_b^a = \mathbf{R}_{\mathbf{k},\theta} = \cos\theta\ \mathbf{I} + \sin\theta\ (\mathbf{k}^a)^{\times} + (1-\cos\theta)\,\mathbf{k}^a(\mathbf{k}^a)^{\mathsf{T}}$$

Euler parameters

- Euler parameters are closely related to angle-axis:
  - Coordinate-free:

$$\eta = \cos\frac{\theta}{2}$$
$$\vec{\epsilon} = \vec{k}\sin\frac{\theta}{2}$$

  With coordinates:

$$\eta = \cos\frac{\theta}{2}$$
$$\boldsymbol{\epsilon} = \mathbf{k}\sin\frac{\theta}{2}$$

- Rotation matrix (on coordinate form):

$$\mathbf{R}(\eta, \boldsymbol{\epsilon}) = \mathbf{I} + 2\eta\boldsymbol{\epsilon}^{\times} + 2\boldsymbol{\epsilon}^{\times}\boldsymbol{\epsilon}^{\times}$$

- Much used, since:
  - Compact, singularity-free representation of orientation
  - No trigonometric terms in expression for rotation matrix
  - $\eta^2 + \vec{\epsilon}\cdot\vec{\epsilon} = 1$: Easy to normalize (avoid roundoff errors)
    - Rotation matrices may tend to become non-orthogonal when simulated
  - Euler parameters are (*unit) quaternions*:
    - Quaternions are generalized complex numbers
    - Can use algebra of quaternions for calculations and analysis

Derivatives of rotations

- Derivative of position $\mathbf{r}$ is velocity, $\dot{\mathbf{r}} = \mathbf{v}$.
- Derivative of rotation matrix $\mathbf{R}_b^a$ is $\dot{\mathbf{R}}_b^a$. What is this?
- Seems natural that a concept of angular velocity should be involved, but how?
  - (Yesterday, repeated next slide)
- What are derivatives of representations of rotations?
  - Derivatives of Euler angles? Euler parameters?
  - These are the kinematic differential equations (today's main topic)

Angular velocity

- The rotation matrix is orthogonal:

$$\mathbf{R}_b^a \left(\mathbf{R}_b^a\right)^{\mathsf{T}} = \mathbf{I}$$

- Differentiate:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left[\mathbf{R}_b^a \left(\mathbf{R}_b^a\right)^{\mathsf{T}}\right] = \dot{\mathbf{R}}_b^a \left(\mathbf{R}_b^a\right)^{\mathsf{T}} + \mathbf{R}_b^a \left(\dot{\mathbf{R}}_b^a\right)^{\mathsf{T}} = \mathbf{0}$$

- If we define $\mathbf{S} = \dot{\mathbf{R}}_b^a \left(\mathbf{R}_b^a\right)^{\mathsf{T}}$, this says that $\mathbf{S} + \mathbf{S}^{\mathsf{T}} = \mathbf{0}$ which means that $\mathbf{S}$ is skew symmetric.

$$\mathbf{S} = \begin{pmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{pmatrix} = \left(\boldsymbol{\omega}_{ab}^a\right)^{\times}$$

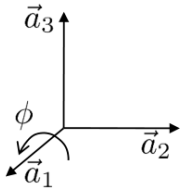
- The vector $\boldsymbol{\omega}_{ab}^a$ defined by $\left(\boldsymbol{\omega}_{ab}^a\right)^{\times} = \dot{\mathbf{R}}_b^a \left(\mathbf{R}_b^a\right)^{\mathsf{T}}$ is the angular velocity of frame *b* relative to frame *a* (decomposed in *a*)

- The equation $\dot{\mathbf{R}}_b^a = \left(\boldsymbol{\omega}_{ab}^a\right)^{\times} \mathbf{R}_b^a$ is the kinematic differential equation for rotation matrices

Angular velocity of simple rotations

- Rotation about *x*-axis:

$$\mathbf{R}_{x,\phi} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{pmatrix}$$



- We calculate $\left(\boldsymbol{\omega}_{ab}^a\right)^{\times} = \dot{\mathbf{R}}_b^a \left(\mathbf{R}_b^a\right)^{\mathsf{T}}$:

$$\dot{\mathbf{R}}_{x,\phi}\left(\mathbf{R}_{x,\phi}\right)^{\mathsf{T}} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & -\sin\phi & -\cos\phi \\ 0 & \cos\phi & -\sin\phi \end{pmatrix} \dot{\phi} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -\dot{\phi} \\ 0 & \dot{\phi} & 0 \end{pmatrix}$$

- That is:

$$\boldsymbol{\omega}_x = \begin{pmatrix} \dot{\phi} \\ 0 \\ 0 \end{pmatrix}$$

- Similar for rotations around *y*- and *z*-axis: $\boldsymbol{\omega}_y = \begin{pmatrix} 0 \\ \dot{\theta} \\ 0 \end{pmatrix}$, $\boldsymbol{\omega}_z = \begin{pmatrix} 0 \\ 0 \\ \dot{\psi} \end{pmatrix}$

- Angle-axis representations (constant axis):

$$\boldsymbol{\omega}_{ab}^a = \dot{\theta}\mathbf{k}^a$$

Composite rotations

- Given
  - composite rotation $\mathbf{R}_d^a = \mathbf{R}_b^a \mathbf{R}_c^b \mathbf{R}_d^c$, and
  - individual angular velocities $\boldsymbol{\omega}_{ab}^a$, $\boldsymbol{\omega}_{bc}^b$, and $\boldsymbol{\omega}_{cd}^c$

  How to calculate the composite angular velocity $\boldsymbol{\omega}_{ad}^a$?

- It can be shown (easy, see book p. 241) that

$$\vec{\omega}_{ad} = \vec{\omega}_{ab} + \vec{\omega}_{bc} + \vec{\omega}_{cd}$$

- On coordinate form:

$$\boldsymbol{\omega}_{ad}^a = \boldsymbol{\omega}_{ab}^a + \boldsymbol{\omega}_{bc}^a + \boldsymbol{\omega}_{cd}^a$$

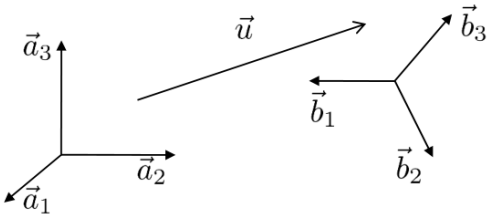
- So:

$$\boldsymbol{\omega}_{ad}^a = \boldsymbol{\omega}_{ab}^a + \mathbf{R}_b^a \boldsymbol{\omega}_{bc}^b + \mathbf{R}_b^a \mathbf{R}_c^b \boldsymbol{\omega}_{cd}^c$$

Differentiation of vectors (6.8.5, 6.8.6)

- Coordinate representation:

$$\mathbf{u}^a = \mathbf{R}_b^a \mathbf{u}^b$$



- Differentiation:

$$\dot{\mathbf{u}}^a = \mathbf{R}_b^a \dot{\mathbf{u}}^b + \dot{\mathbf{R}}_b^a \mathbf{u}^b$$

$$\dot{\mathbf{R}}_b^a = \mathbf{R}_b^a \left(\boldsymbol{\omega}_{ab}^b\right)^\times$$

$$\dot{\mathbf{u}}^a = \mathbf{R}_b^a \left[\dot{\mathbf{u}}^b + \left(\boldsymbol{\omega}_{ab}^b\right)^\times \mathbf{u}^b\right]$$

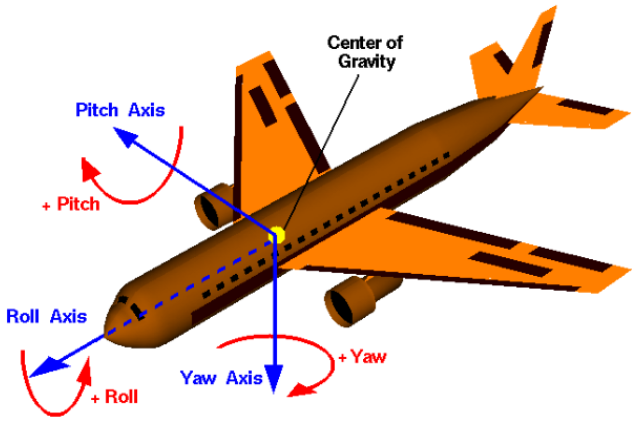
Note! Generally,

$$\dot{\mathbf{u}}^a \neq \mathbf{R}_b^a \dot{\mathbf{u}}^b$$

- On vector form:

$$\frac{{}^a\mathrm{d}}{\mathrm{d}t}\vec{u} = \frac{{}^b\mathrm{d}}{\mathrm{d}t}\vec{u} + \vec{\omega}_{ab} \times \vec{u}$$

Euler-angles: Roll-pitch-yaw

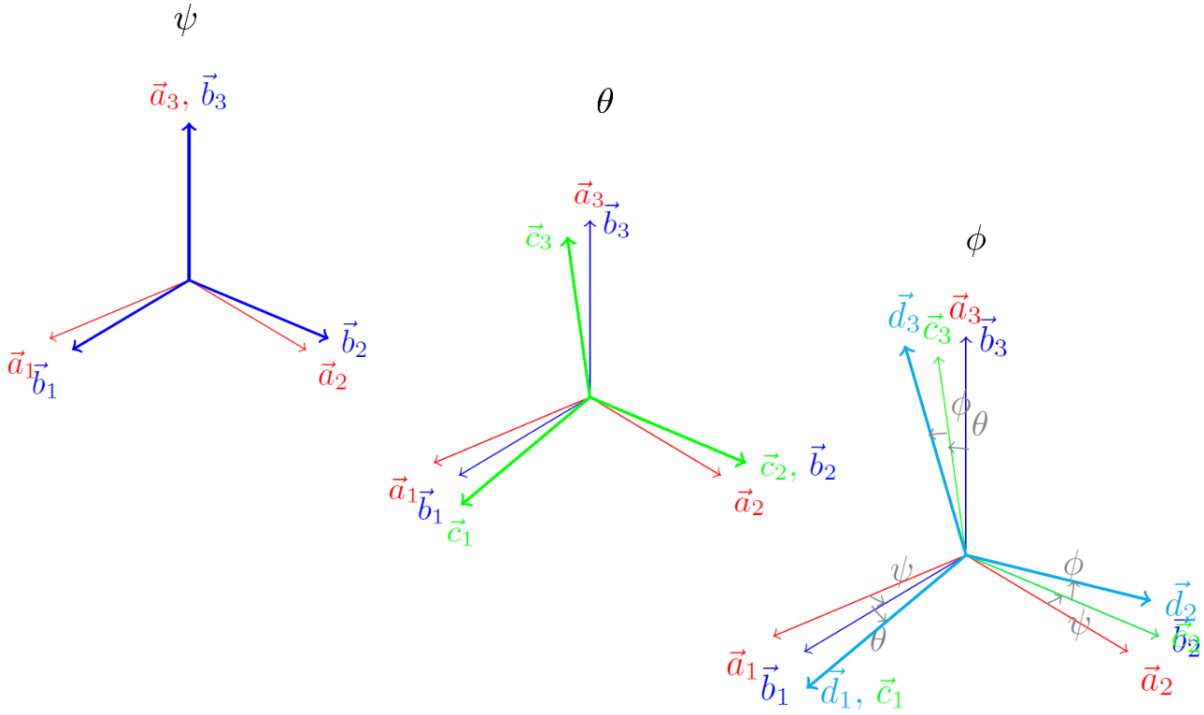


- Rotation ψ about *z*-axis, θ about (rotated) *y*-axis, ϕ about (rotated) *x*-axis

$$\mathbf{R}_b^a = \mathbf{R}_{z,\psi}\mathbf{R}_{y,\theta}\mathbf{R}_{x,\phi}$$

$$\mathbf{R}_b^a = \begin{pmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{pmatrix}$$

Euler angles



Lecture 17: Newton-Euler equations of motion

- Rigid body kinetics (Newton-Euler equations of motion)
  - Newton's law
  - Angular momentum
  - Inertia dyadic

Book: Ch. 7.3

What is rigid body dynamics?

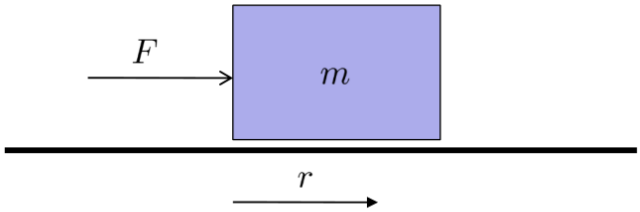
- Rigid body:
  - Wikipedia: "…a rigid body is an idealization of a solid body of finite size in which deformation is neglected."

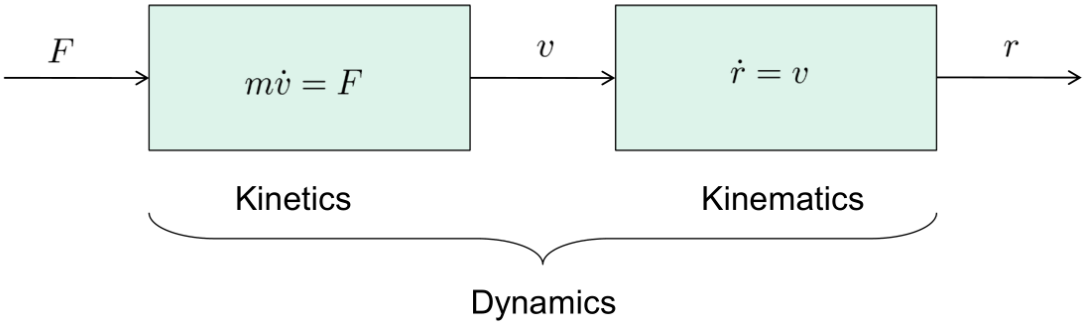
- Dynamics = Kinematics + Kinetics

- Kinematics
  - eb.com: "…branch of physics (…) concerned with the geometrically possible motion of a body or system of bodies without consideration of the forces involved (i.e., causes and effects of the motions)."
  - Book: Ch. 6
- Kinetics
  - eb.com: "…the effect of forces and torques on the motion of bodies having mass."
  - Book: Ch. 7, 8.

Remark: Sometimes "dynamics" is used for "kinetics" only

Simplest scalar case

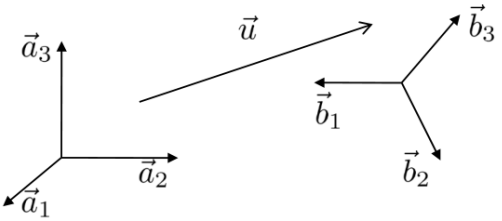




Differentiations of vectors (6.8.5, 6.8.6)

- Coordinate representation:

$$\mathbf{u}^a = \mathbf{R}_b^a \mathbf{u}^b$$



- Differentiation:

$$\dot{\mathbf{u}}^a = \mathbf{R}_b^a \dot{\mathbf{u}}^b + \dot{\mathbf{R}}_b^a \mathbf{u}^b$$

$$\dot{\mathbf{R}}_b^a = \mathbf{R}_b^a \left(\boldsymbol{\omega}_{ab}^b\right)^\times$$

$$\dot{\mathbf{u}}^a = \mathbf{R}_b^a \left[\dot{\mathbf{u}}^b + \left(\boldsymbol{\omega}_{ab}^b\right)^\times \mathbf{u}^b\right]$$

Note! Generally,

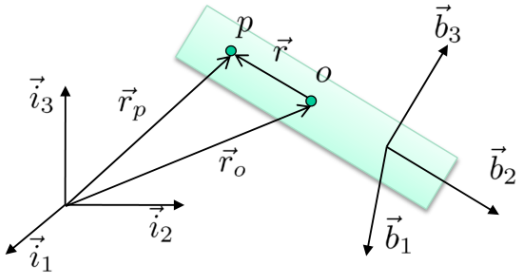
$$\dot{\mathbf{u}}^a \neq \mathbf{R}_b^a \dot{\mathbf{u}}^b$$

- On vector form:

$$\frac{{}^a\mathrm{d}}{\mathrm{d}t}\vec{u} = \frac{{}^b\mathrm{d}}{\mathrm{d}t}\vec{u} + \vec{\omega}_{ab} \times \vec{u}$$

Rigid body kinematics

- Velocities and accelerations (Ch. 6.12)



$$\vec{v}_o := \frac{^i\mathrm{d}}{\mathrm{d}t}\vec{r}_o, \quad \vec{v}_p := \frac{^i\mathrm{d}}{\mathrm{d}t}\vec{r}_p$$

$$\vec{a}_o := \frac{^i\mathrm{d}^2}{\mathrm{d}t^2}\vec{r}_o, \quad \vec{a}_p := \frac{^i\mathrm{d}^2}{\mathrm{d}t^2}\vec{r}_p$$

$$\vec{\alpha}_{ib} := \frac{^i\mathrm{d}}{\mathrm{d}t}\vec{\omega}_{ib} = \frac{^b\mathrm{d}}{\mathrm{d}t}\vec{\omega}_{ib}$$

$$\begin{aligned}\vec{v}_p &= \vec{v}_o + \frac{^i\mathrm{d}}{\mathrm{d}t}\vec{r} \\ &= \vec{v}_o + \frac{^b\mathrm{d}}{\mathrm{d}t}\vec{r} + \vec{\omega}_{ib} \times \vec{r} \\ &= \vec{v}_o + \vec{\omega}_{ib} \times \vec{r}, \quad \vec{r} \text{ fixed.}\end{aligned}$$

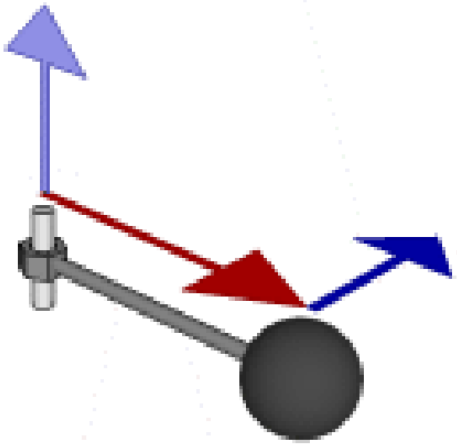
$$\vec{a}_p = \vec{a}_o + \frac{^b\mathrm{d}^2}{\mathrm{d}t^2}\vec{r} + 2\vec{\omega}_{ib} \times \frac{^b\mathrm{d}}{\mathrm{d}t}\vec{r} + \vec{\alpha}_{ib} \times \vec{r} + \vec{\omega}_{ib} \times (\vec{\omega}_{ib} \times \vec{r})$$

$$\vec{a}_p = \vec{a}_o + \vec{\alpha}_{ib} \times \vec{r} + \vec{\omega}_{ib} \times (\vec{\omega}_{ib} \times \vec{r}), \quad \vec{r} \text{ fixed.}$$

Torque, and linear/angular momentum

$\boldsymbol{\tau} = \mathbf{r} \times \mathbf{F}$

$\mathbf{L} = \mathbf{r} \times \mathbf{p}$



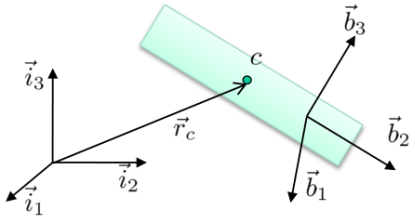
Source: Wikipedia

- Book:
  - Torque: $\vec{N}, \vec{T}$
  - Angular momentum: $\vec{h}$

Newton-Euler EoM

- Referenced to center of mass (CoM):

$$\vec{F}_{bc} = m\vec{a}_c$$
$$\vec{T}_{bc} = \vec{M}_{b/c} \cdot \vec{\alpha}_{ib} + \vec{\omega}_{ib} \times \left(\vec{M}_{b/c} \cdot \vec{\omega}_{ib}\right)$$



- Sometimes convenient to have them referenced to other point *o*:
  - Forces and moments in *o*:

$$\vec{F}_{bo} = \vec{F}_{bc}$$
$$\vec{T}_{bo} = \vec{T}_{bc} + \vec{r}_g \times \vec{F}_{bc}$$

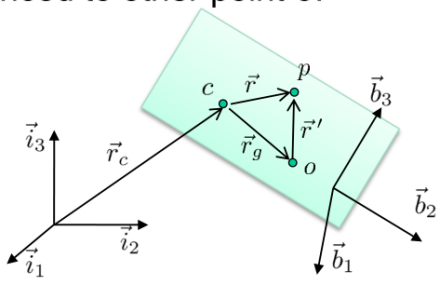
  - Use

$$\vec{a}_c = \vec{a}_o + \vec{\alpha}_{ib} \times \vec{r}_g + \vec{\omega}_{ib} \times (\vec{\omega}_{ib} \times \vec{r}_g)$$

  - Define

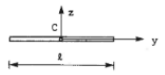
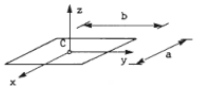
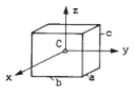
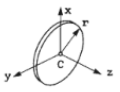
$$\vec{M}_{b/o} := -\int_b (\vec{r}\,')^{\times}(\vec{r}\,')^{\times} \mathrm{d}m$$

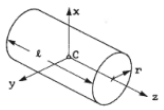
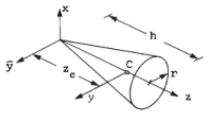
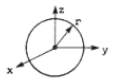
$$\vec{F}_{bo} = m\left(\vec{a}_o + \vec{\alpha}_{ib} \times \vec{r}_g + \vec{\omega}_{ib} \times (\vec{\omega}_{ib} \times \vec{r}_g)\right)$$
$$\vec{T}_{bo} = \vec{r}_g \times \vec{a}_o + \vec{M}_{b/o} \cdot \vec{\alpha}_{ib} + \vec{\omega}_{ib} \times \left(\vec{M}_{b/o} \cdot \vec{\omega}_{ib}\right)$$



  - Useful when CoM changes – no need to recalculate inertia matrix – still need to know CoM

Finding moments of inertia

Homogen slank stav 	$I_z = \frac{1}{12}ml^2$ $I_f = \frac{1}{3}ml^2$
Tynn rektangulær plate 	$I_z = \frac{1}{12}m\,(a^2+b^2)$ $I_x = \frac{1}{12}mb^2$ $I_y = \frac{1}{12}ma^2$
Rektangulært prisme 	$I_z = \frac{1}{12}m\,(a^2+b^2)$
Tynn sirkulær skive 	$I_z = \frac{1}{2}mr^2$ $I_x = I_y = \frac{1}{4}mr^2$

Sirkulær sylinder 	$I_z = \frac{1}{2}mr^2$ $I_x = I_y = \frac{1}{12}m(3r^2+l^2)$
Tynt sylinderskall	$I_z = mr^2$ $I_x = I_y = \frac{1}{2}mr^2 + \frac{1}{12}ml^2$
Rett sirkulær kjegle 	$I_z = \frac{1}{10}mr^2$ $I_y = \frac{3}{20}mr^2 + \frac{3}{80}mh^2$ $I_{\bar{y}} = \frac{3}{20}mr^2 + \frac{3}{5}mh^2$ $z_c = 3h/4$
Kule 	$I_C = \frac{2}{5}mr^2$
Kuleskall	$I_C = \frac{2}{3}mr^2$

From F. Irgens, Dynamikk

- http://en.wikipedia.org/wiki/List_of_moment_of_inertia_tensors
- For other/general rigid bodies (vessels/planes/etc.), computer programs can find moments of inertia

	Kinematics	Kinetics
	Derivatives of position and orientation as function of velocity and angular velocity	Derivatives of velocity and angular velocity as function of applied forces and torques
Translation	1D: $\dot{r} = v$ 3D: $\dot{\mathbf{r}}_c^i = \mathbf{v}_c^i$	1D: $m\dot{v} = F$ 3D: $m\dot{\mathbf{v}}_c^i = \mathbf{F}_{bc}^i$
Note! By definition $\vec{v}_c := \frac{^i\mathrm{d}}{\mathrm{d}t}\vec{r}_c$	$\dot{\mathbf{r}}_c^i = \mathbf{v}_c^i = \mathbf{R}_b^i \mathbf{v}_c^b$	Usually convenient to have forces and velocities in body system: $m\left(\dot{\mathbf{v}}_c^b + \left(\boldsymbol{\omega}_{ib}^b\right)^\times \mathbf{v}_c^b\right) = \mathbf{F}_{bc}^b$
Rotation/ orientation	1D: $\dot{\theta} = \omega$ 3D: Depends on parameterization Rotation matrix: $\dot{\mathbf{R}}_b^i = \mathbf{R}_b^i \left(\boldsymbol{\omega}_{ib}^b\right)^\times$ Euler angles: $\dot{\boldsymbol{\phi}} = \mathbf{E}_d^{-1}(\boldsymbol{\phi})\boldsymbol{\omega}_{ib}^b$ Euler parameters: $\dot{\eta} = -\frac{1}{2}\boldsymbol{\epsilon}^\top \boldsymbol{\omega}_{ib}^b$ $\dot{\boldsymbol{\epsilon}} = \frac{1}{2}\left(\eta\mathbf{I} + \boldsymbol{\epsilon}^\times\right)\boldsymbol{\omega}_{ib}^b$	1D: $J\dot{\omega} = T$ 3D: $\mathbf{M}_{b/c}^b \dot{\boldsymbol{\omega}}_{ib}^b + \left(\boldsymbol{\omega}_{ib}^b\right)^\times \mathbf{M}_{b/c}^b \boldsymbol{\omega}_{ib}^b = \mathbf{T}_{bc}^b$

Lecture 18: Newton-Euler equations of motion, Modelica/Dymola: The Multibody library

- Newton-Euler equations of motion
  - Recap
  - Kinetic energy
  - Example

- Software
  - Dymola and the Modelica.Mechanics.Multibody library

Book: 7.3

Newton-Euler equations of motion

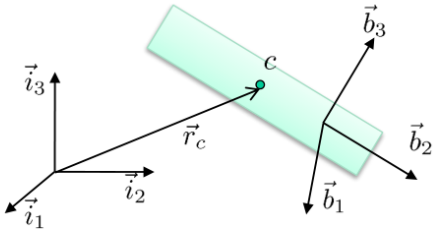
- Newton's law (for particle *k*)

$$m_k \vec{a}_k = \vec{F}^{(r)}$$

- Newton-Euler EoM for rigid bodies:
  - Integrate Newton's law over body, define center of mass
  - Define torque/moment and angular momentum to handle forces that give rotation about center of mass
  - Define inertia dyadic/matrix

$$\vec{F}_{bc} = m\vec{a}_c$$

$$\vec{T}_{bc} = \vec{M}_{b/c} \cdot \vec{\alpha}_{ib} + \vec{\omega}_{ib} \times \left(\vec{M}_{b/c} \cdot \vec{\omega}_{ib} \right)$$



(Here: Referenced to center of mass)

- Implemented in e.g. Dymola (Modelica.Multibody library)

Traits of Newton-Euler EoM

(and a preview: Lagrange EoM)

$$\vec{F}_{bc} = m\vec{a}_c$$
$$\vec{T}_{bc} = \vec{M}_{b/c} \cdot \vec{\alpha}_{ib} + \vec{\omega}_{ib} \times \left(\vec{M}_{b/c} \cdot \vec{\omega}_{ib}\right)$$

Newton-Euler EoM:

- Involves working with vectors
  - Lagrange: Algebraic manipulations
- Forces and moments are central
  - Lagrange: Energy and work are central
- All forces in the system must be considered
  - Lagrange: Forces of constraint are implicitly eliminated with the use of generalized coordinates (and generalized forces)
- Somewhat complicated to use by hand, but can be implemented in computer systems
  - Lagrange: Easier to do by hand, not suitable for complex systems

- d'Alembert's principle: Elimination of forces of constraint (Ch. 7.7)
  - Can simplify application of Newton-Euler EoM
    - Kane's EoM (Ch. 7.8, 7.9)
  - Starting point for Lagrange EoM (Ch. 8.2)

Inertia matrix

- Found for each rigid body by calculating

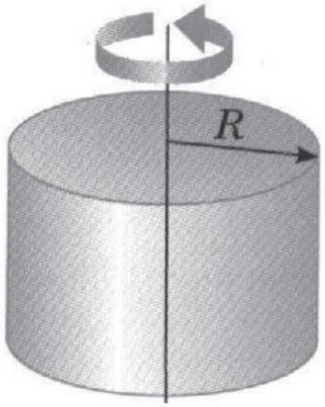
$$M^b_{b/c} = \int_b (\mathbf{r}^b)^\mathsf{T}\mathbf{r}^b I - \mathbf{r}^b(\mathbf{r}^b)^\mathsf{T} \mathrm{d}m = \int_b \begin{pmatrix} y^2+z^2 & -xy & -xz \\ -xy & x^2+z^2 & -yz \\ -xz & -yz & x^2+y^2 \end{pmatrix} \mathrm{d}m$$

- Constant in body-fixed coordinate system!
- Not constant in inertial coordinate system

$$M^i_{b/c} = R^i_b M^b_{b/c} (R^i_b)^\mathsf{T}$$

- Books and wikipedia have tables for common geometries, otherwise computer programs calculates, or can be calculated/identified based on experiments
- Typically, axis in body-system chosen as body symmetri axis, giving zeros in inertia matrix. If symmetric about all axis, the inertia matrix becomes diagonal.

Inertia matrix, examples



Homogeneous Disk

$$I_{disk} = \frac{1}{4}mr^2 \begin{bmatrix} 1+\frac{1}{3}\frac{h}{r^2} & 0 & 0 \\ 0 & 1+\frac{1}{3}\frac{h}{r^2} & 0 \\ 0 & 0 & \frac{1}{2} \end{bmatrix}$$

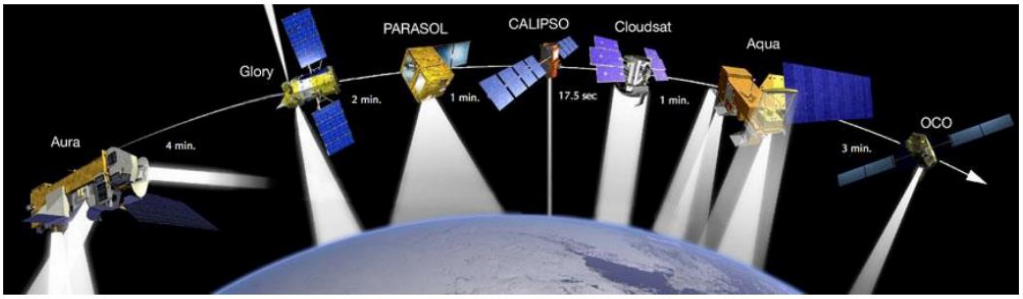
F/A-18

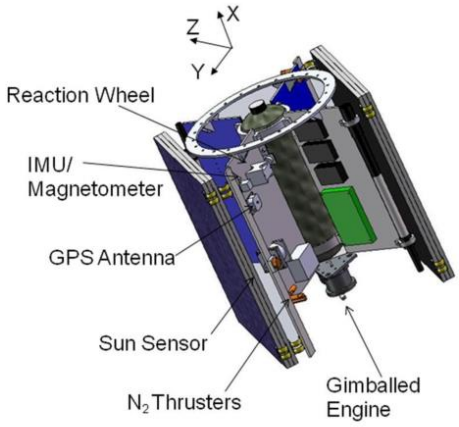
$$I = \begin{bmatrix} 23 & 0 & 2.97 \\ 0 & 15.13 & 0 \\ 2.97 & 0 & 16.99 \end{bmatrix} kslug - ft^2$$

1 slug = 14.6 kg
1 ft = 0.304 m

	Kinematics	Kinetics
	Derivatives of position and orientation as function of velocity and angular velocity	Derivatives of velocity and angular velocity as function of applied forces and torques
Translation	1D: $\dot{r} = v$ 3D: $\dot{\mathbf{r}}_c^i = \mathbf{v}_c^i$	1D: $m\dot{v} = F$ 3D: $m\dot{\mathbf{v}}_c^i = \mathbf{F}_{bc}^i$
Note! By definition $\vec{v}_c := \frac{^i\mathrm{d}}{\mathrm{d}t}\vec{r}_c$	$\dot{\mathbf{r}}_c^i = \mathbf{v}_c^i = \mathbf{R}_b^i \mathbf{v}_c^b$ ⟸	Usually convenient to have forces and velocities in body system: $m\left(\dot{\mathbf{v}}_c^b + \left(\boldsymbol{\omega}_{ib}^b\right)^\times \mathbf{v}_c^b\right) = \mathbf{F}_{bc}^b$
Rotation/ orientation	1D: $\dot{\theta} = \omega$ 3D: Depends on parameterization Rotation matrix: $\dot{\mathbf{R}}_b^i = \mathbf{R}_b^i \left(\boldsymbol{\omega}_{ib}^b\right)^\times$ Euler angles: $\dot{\boldsymbol{\phi}} = \mathbf{E}_d^{-1}(\boldsymbol{\phi})\boldsymbol{\omega}_{ib}^b$ Euler parameters: $\dot{\eta} = -\frac{1}{2}\boldsymbol{\epsilon}^\top \boldsymbol{\omega}_{ib}^b$ $\dot{\boldsymbol{\epsilon}} = \frac{1}{2}\left(\eta\mathbf{I} + \boldsymbol{\epsilon}^\times\right)\boldsymbol{\omega}_{ib}^b$	1D: $J\dot{\omega} = T$ 3D: $\mathbf{M}_{b/c}^b \dot{\boldsymbol{\omega}}_{ib}^b + \left(\boldsymbol{\omega}_{ib}^b\right)^\times \mathbf{M}_{b/c}^b \boldsymbol{\omega}_{ib}^b = \mathbf{T}_{bc}^b$

Satellite attitude dynamics





http://satellite.mit.edu/

$$\vec{F}_{bc} = m\vec{a}_c$$

$$\vec{T}_{bc} = \vec{M}_{b/c} \cdot \vec{\alpha}_{ib} + \vec{\omega}_{ib} \times \left(\vec{M}_{b/c} \cdot \vec{\omega}_{ib} \right)$$

Airplane EoM (from book about airplane dynamics)

$$\begin{aligned} X - mgS_\theta &= m(\dot{u} + qw - rv) \\ Y + mgC_\theta S_\Phi &= m(\dot{v} + ru - pw) \\ Z + mgC_\theta C_\Phi &= m(\dot{w} + pv - qu) \end{aligned}$$

Force equations

$$m\left(\dot{\mathbf{v}}_c^b + \left(\boldsymbol{\omega}_{ib}^b\right)^\times \mathbf{v}_c^b\right) = \mathbf{F}_{bc}^b$$

$$\begin{aligned} L &= I_x\dot{p} - I_{xz}\dot{r} + qr(I_z - I_y) - I_{xz}pq \\ M &= I_y\dot{q} + rp(I_x - I_z) + I_{xz}(p^2 - r^2) \\ N &= -I_{xz}\dot{p} + I_z\dot{r} + pq(I_y - I_x) + I_{xz}qr \end{aligned}$$

Moment equations

$$\mathbf{M}_{b/c}^b \dot{\boldsymbol{\omega}}_{ib}^b + \left(\boldsymbol{\omega}_{ib}^b\right)^\times \mathbf{M}_{b/c}^b \boldsymbol{\omega}_{ib}^b = \mathbf{T}_{bc}^b$$

$$\begin{aligned} p &= \dot{\Phi} - \dot{\psi}S_\theta \\ q &= \dot{\theta}C_\Phi + \dot{\psi}C_\theta S_\Phi \\ r &= \dot{\psi}C_\theta C_\Phi - \dot{\theta}S_\Phi \end{aligned}$$

Body angular velocities in terms of Euler angles and Euler rates

$$\begin{aligned} \dot{\theta} &= qC_\Phi - rS_\Phi \\ \dot{\Phi} &= p + qS_\Phi T_\theta + rC_\Phi T_\theta \\ \dot{\psi} &= (qS_\Phi + rC_\Phi)\sec\theta \end{aligned}$$

Euler rates in terms of Euler angles and body angular velocities

$$\dot{\boldsymbol{\phi}} = \mathbf{E}_d^{-1}(\boldsymbol{\phi})\boldsymbol{\omega}_{ib}^b$$

Velocity of aircraft in the fixed frame in terms of Euler angles and body velocity components

$$\begin{bmatrix} \frac{dx}{dt} \\ \frac{dy}{dt} \\ \frac{dz}{dt} \end{bmatrix} = \begin{bmatrix} C_\theta C_\psi & S_\Phi S_\theta C_\psi - C_\Phi S_\psi & C_\Phi S_\theta C_\psi + S_\Phi S_\psi \\ C_\theta S_\psi & S_\Phi S_\theta S_\psi + C_\Phi C_\psi & C_\Phi S_\theta S_\psi - S_\Phi C_\psi \\ -S_\theta & S_\Phi C_\theta & C_\Phi C_\theta \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix}$$

$$\dot{\mathbf{r}}_c^i = \mathbf{v}_c^i = \mathbf{R}_b^i \mathbf{v}_c^b$$

Lecture 19: Lagrangian mechanics (Lagrange's equation of motion)

Newton's law (for particles, or Newton-Euler EoM for rigid bodies) in combination with

- d'Alembert's principle
- Generalized coordinates

gives Lagrange's equations of motion

- Brief examples

Book: Ch. 7.7, 8.2

Newton-Euler equations of motion

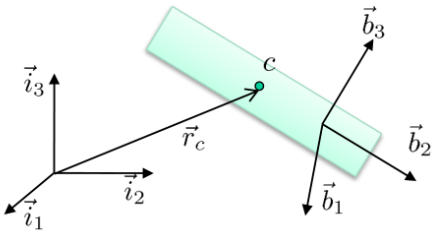
- Newton's law (for particle *k*)

$$m_k \vec{a}_k = \vec{F}^{(r)}$$

- Newton-Euler EoM for rigid bodies:
  - Integrate Newton's law over body, define center of mass
  - Define torque/moment and angular momentum to handle forces that give rotation about center of mass
  - Define inertia dyadic/matrix

$$\vec{F}_{bc} = m\vec{a}_c$$

$$\vec{T}_{bc} = \vec{M}_{b/c} \cdot \vec{\alpha}_{ib} + \vec{\omega}_{ib} \times \left(\vec{M}_{b/c} \cdot \vec{\omega}_{ib} \right)$$



(Here: Referenced to center of mass)

- Implemented in e.g. Dymola (Modelica.Multibody library)

Lagrange vs Newton-Euler

Newton-Euler

- Vectors
- Forces and moments
- Does not eliminate forces of constraints:
  - Obtains solutions for all forces and kinematic variables
  - ”Inefficient” (large DAE models)
- More general
  - Large systems can be handled, but for some configurations tricks are needed
  - Used in advanced modeling software

Lagrange

- Algebraic
- Energy
- Eliminates forces of constraints
  - Solutions only for generalized coordinates (and forces)
  - ”Efficient” (smaller ODE models)
- Less general
  - Need independent generalized coordinates
  - Difficult to automate for large/complex problems

Lecture 20: Rigid body dynamics, summing up

- Brief recap: Newton-Euler equations of motion
- Brief recap: Lagrange's equation of motion

- Pendulum example using both Newton-Euler and Lagrange

- Old exam(s) (using Lagrange)

Lagrange vs Newton-Euler

Newton-Euler

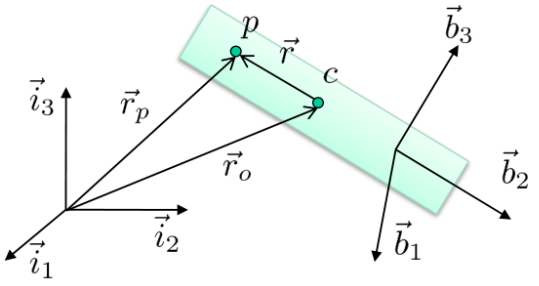
- Vectors
- Forces and moments
- Does not eliminate forces of constraints:
  - Obtains solutions for all forces and kinematic variables
  - ”Inefficient” (large DAE models)
- More general
  - Large systems can be handled (but for some configurations tricks are needed)
  - Used in advanced modeling software

Lagrange

- Algebraic
- Energy
- Eliminates forces of constraints
  - Solutions only for generalized coordinates (and forces)
  - ”Efficient” (smaller ODE models)
- Less general
  - Need independent generalized coordinates
  - Difficult to automate for large/complex problems

Newton-Euler EoM for rigid bodies

- Velocities and accelerations (Ch. 6.12)



$$\vec{v}_c := \frac{{}^i\mathrm{d}}{\mathrm{d}t}\vec{r}_c, \quad \vec{v}_p := \frac{{}^i\mathrm{d}}{\mathrm{d}t}\vec{r}_p$$

$$\vec{a}_c := \frac{{}^i\mathrm{d}^2}{\mathrm{d}t^2}\vec{r}_c, \quad \vec{a}_p := \frac{{}^i\mathrm{d}^2}{\mathrm{d}t^2}\vec{r}_p$$

$$\begin{aligned}\vec{v}_p &= \vec{v}_c + \frac{{}^i\mathrm{d}}{\mathrm{d}t}\vec{r} \\ &= \vec{v}_c + \frac{{}^b\mathrm{d}}{\mathrm{d}t}\vec{r} + \vec{\omega}_{ib} \times \vec{r} \\ &= \vec{v}_c + \vec{\omega}_{ib} \times \vec{r}, \quad \vec{r} \text{ fixed.}\end{aligned}$$

$$\frac{{}^i\mathrm{d}}{\mathrm{d}t}\vec{u} = \frac{{}^b\mathrm{d}}{\mathrm{d}t}\vec{u} + \vec{\omega}_{ib} \times \vec{u}$$

$$\vec{a}_p = \vec{a}_c + \vec{\alpha}_{ib} \times \vec{r} + \vec{\omega}_{ib} \times (\vec{\omega}_{ib} \times \vec{r}), \quad \vec{r} \text{ fixed.}$$

- Newton-Euler equations of motion (Ch. 7.3)

$$\begin{aligned}\vec{F}_{bc} &= m\vec{a}_c \\ \vec{T}_{bc} &= \vec{M}_{b/c} \cdot \vec{\alpha}_{ib} + \vec{\omega}_{ib} \times \left(\vec{M}_{b/c} \cdot \vec{\omega}_{ib}\right)\end{aligned}$$

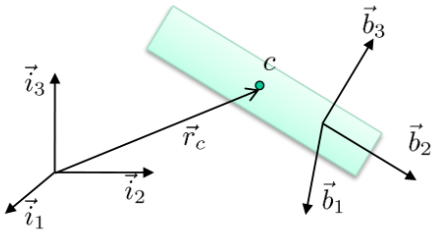
Newton-Euler equations of motion

- Newton's law (for particle *k*)

$$m_k \vec{a}_k = \vec{F}^{(r)}$$

- Newton-Euler EoM for rigid bodies:
  - Integrate Newton's law over body, define center of mass
  - Define torque/moment and angular momentum to handle forces that give rotation about center of mass
  - Define inertia dyadic/matrix

$$\begin{aligned}\vec{F}_{bc} &= m\vec{a}_c \\ \vec{T}_{bc} &= \vec{M}_{b/c} \cdot \vec{\alpha}_{ib} + \vec{\omega}_{ib} \times \left(\vec{M}_{b/c} \cdot \vec{\omega}_{ib}\right)\end{aligned}$$



(Here: Referenced to center of mass)

- Implemented in e.g. Dymola (Modelica.Multibody library)

Lagrange equations of motion

Generalized coordinates

- Find *n* generalized coordinates that parametrize ”degrees of freedom” (allowed motion).
  - That is, all positions are function of generalized coordinates

$$\vec{r}_k = \vec{r}_k(\mathbf{q}) \qquad \mathbf{q} = \begin{pmatrix} q_1 & q_2 & \dots & q_n \end{pmatrix}^{\mathsf{T}}$$

- Differentiate to find velocity

$$\vec{v}_k(\mathbf{q}, \dot{\mathbf{q}}) = \frac{\mathrm{d}}{\mathrm{d}t}\vec{r}_k(\mathbf{q}) = \sum_{i=1}^{N} \frac{\partial \vec{r}_k}{\partial q_i}\dot{q}_i$$

  - For rigid bodies: velocity of center(s) of mass, and also angular velocity $\vec{\omega}_{ib}(\mathbf{q}, \dot{\mathbf{q}})$

- Find the generalized (actuator) forces τ_i associated with q_i
  - If q_i angle, then τ_i torque
  - If q_i displacement, then τ_i force

$$\tau_i = \sum_{i=1}^{N} \frac{\partial \vec{r}_k}{\partial q_i} \cdot \vec{F}_k$$

- On coordinate form:

$$k = 1, \dots, N \text{ particles: } \mathbf{r}_k^i(\mathbf{q}), \quad \mathbf{v}_k^i(\mathbf{q}, \dot{\mathbf{q}})$$

$$k = 1, \dots, N \text{ rigid bodies: } \mathbf{r}_{ck}^i(\mathbf{q}), \quad \mathbf{v}_{ck}^b(\mathbf{q}, \dot{\mathbf{q}}), \quad \boldsymbol{\omega}_{ik}^b(\mathbf{q}, \dot{\mathbf{q}}), \quad \mathbf{M}_{k/c}^b$$

Lagrange equations of motion

Kinetic and potential energy

- Find kinetic energy:
  - N particles:

$$T = \textstyle\sum_{k=1}^{N} \frac{1}{2} m_k \vec{v}_k \cdot \vec{v}_k$$

  - Each rigid body (p. 273):

$$T = \int_b \tfrac{1}{2}\vec{v}_p \cdot \vec{v}_p \mathrm{d}m = \tfrac{1}{2} m \vec{v}_c \cdot \vec{v}_c + \tfrac{1}{2}\vec{\omega}_{ib} \cdot \vec{M}_{b/c} \cdot \vec{\omega}_{ib}$$

- On coordinate form:

$$N \text{ particles: } T = \textstyle\sum T_k, \quad T_k(\mathbf{q}, \dot{\mathbf{q}}, t) = \frac{1}{2} m_k (\mathbf{v}_k^i)^{\mathsf{T}} \mathbf{v}_k^i = \frac{1}{2} m_k (\mathbf{v}_k^b)^{\mathsf{T}} \mathbf{v}_k^b$$

$$N \text{ rigid bodies: } T = \textstyle\sum T_k, \quad T_k(\mathbf{q}, \dot{\mathbf{q}}, t) = \frac{1}{2} m_k (\mathbf{v}_{ck}^b)^{\mathsf{T}} \mathbf{v}_{ck}^b + \frac{1}{2} (\boldsymbol{\omega}_{ik}^b)^{\mathsf{T}} \mathbf{M}_{k/c}^b \boldsymbol{\omega}_{ik}^b$$

- Find (total) potential energy $U = U(\mathbf{q}) = \sum U_k(\mathbf{q})$
  - Gravity: $U_k(\mathbf{q}) = m_k g h(\mathbf{q})$
  - Spring: $U_k(\mathbf{q}) = \frac{1}{2} k x^2(\mathbf{q})$
  - ...

Lagrange equations of motion

- Construct Lagrangian

$$\mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}, t) = T(\mathbf{q}, \dot{\mathbf{q}}, t) - U(\mathbf{q})$$

- Find 2*n* partial derivatives (scalars)

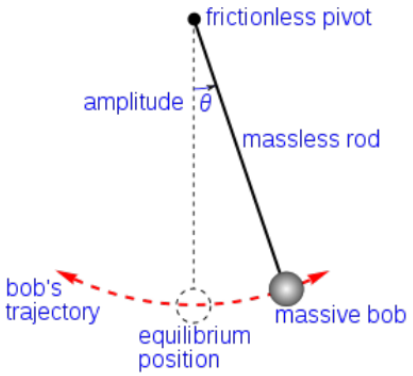
$$\frac{\partial \mathcal{L}}{\partial \dot{q}_i} \qquad \frac{\partial \mathcal{L}}{\partial q_i}$$

- Write up *n* equations of motion
  - That is, *n* 2nd order differential equations

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}_i}\right) - \frac{\partial \mathcal{L}}{\partial q_i} = \tau_i$$

Example: Pendulum

- Pendulum (bob) as particle:
  - Using Newton-Euler EoM, in inertial and body system
  - Using Lagrange EoM
- Pendulum as rigid body
  - Using Lagrange EoM



Lecture 21: Process modeling & balance laws

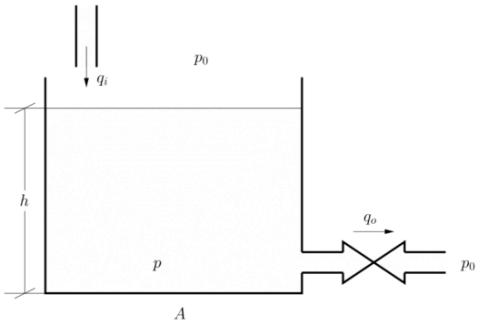
- Process modeling, structure and methodology
- Balance laws
  - Mass balances
  - Mass balances for multi-component systems

Book: 10.4, 11.1-11.4

Process modeling: Structure and methodology

- Goal of process modeling: Construct mathematical models of the process under study.
- These mathematical models consists of process variables (variables and parameters) and the equations that link these

Process



Process variables
(variables and parameters)

- Level
  - Variability: Variable
  - Symbol: *h*
  - Value: 1.1
  - Unit: m
  - ...
- Area
  - Variability: Parameter
  - Symbol: *A*
  - Value: 2.2
  - Unit: m^2
  - ...
- ...

Process equations

$$\frac{\mathrm{d}h}{\mathrm{d}t} = \frac{1}{A}\left(q_i - q_o\right)$$
$$q_o = C\sqrt{p - p_0}$$
$$p = p_0 + \rho g h$$

Number of equations must match number of (unknown) variables.

Process equations

- Balance laws
  - Mass
  - Momentum
  - Energy
  - ...
- Constitutive equations
  - For (generalized) flows
  - Thermodynamic equations of state (e.g. ideal gas law)
  - Phenomenological relationships (e.g. between friction force and flow in a pipe)
  - ...
- Constraints
  - Geometric relationships
  - Equilibrium conditions
  - ...

Also called «closure relations» as they «close» the balance laws (such that #equations = #variables)

Example Tank:

$$\frac{\mathrm{d}h}{\mathrm{d}t} = \frac{1}{A}\left(q_i - q_o\right) \qquad q_o = C\sqrt{p - p_0} \qquad p = p_0 + \rho g h \qquad V = Ah$$

Process variables

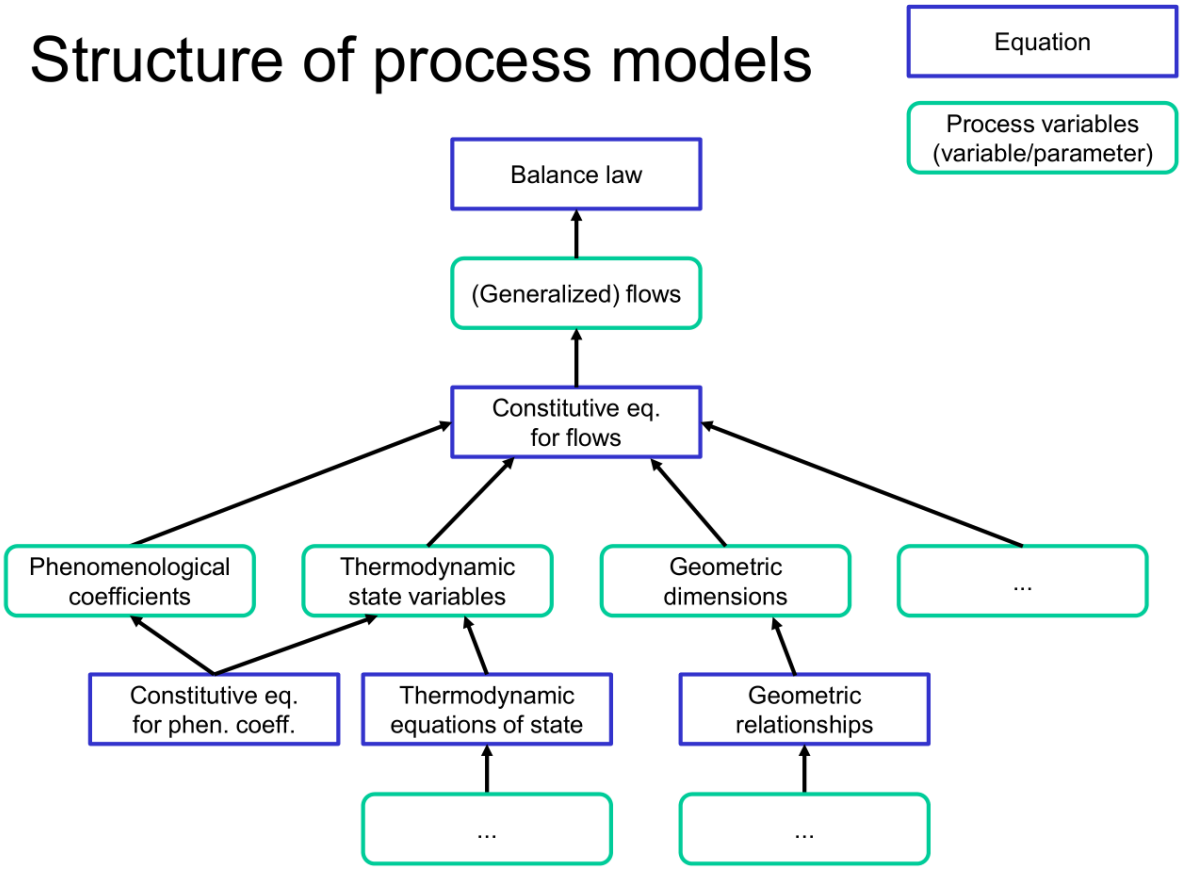
- Thermodynamic state variables
  - Mass, pressures, levels, ...
  - Velocities
  - Temperatures
  - ...
- (Generalized) flows
  - Transport (single phase)
  - Exchange (between phases)
  - Sources (reactions)
  - ...
- Phenomenological coefficients
  - Viscosity
  - Reaction rates
  - Valve constants
  - ...
- Geometric dimensions
  - Lengths, Areas, Volumes
  - ...

Example Tank:

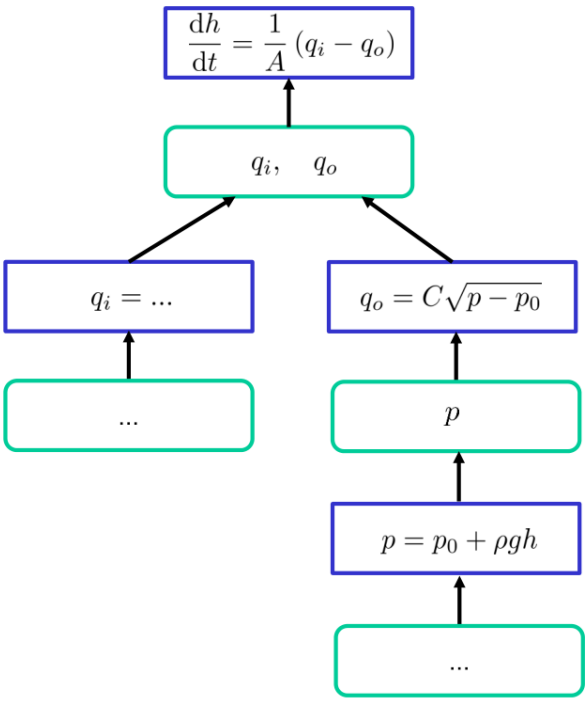
h, ρ, p, p_0 $\qquad$ q_i, q_o $\qquad$ C, g $\qquad$ A

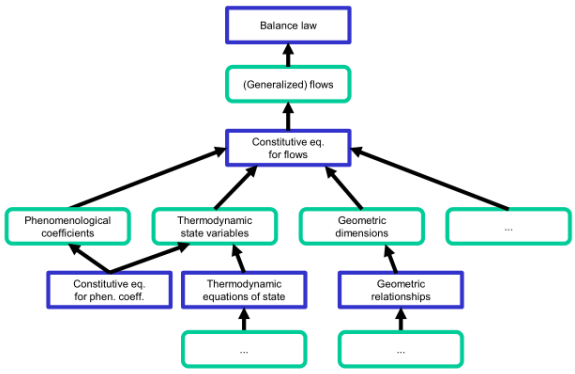
$$\frac{\mathrm{d}h}{\mathrm{d}t} = \frac{1}{A}\left(q_i - q_o\right)$$
$$q_o = C\sqrt{p - p_0}$$
$$p = p_0 + \rho g h$$

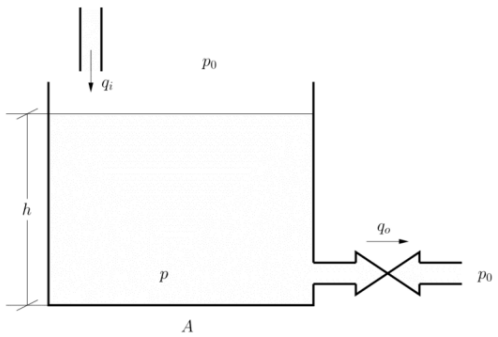
Structure of process models



Example: Tank







Physical balance principles are based on

Conservation laws

That a physical property is *conserved*, means that it will remain constant in a closed system

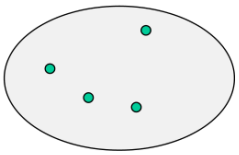
The following physical quantities are conserved:

- Mass
- Energy
- Momentum (norsk: impuls)
  - Linear and angular

No one has ever observed that conservation laws have been violated

- Conservation laws are exact laws

The basic physical principles



Consider a volume consisting of a fixed number of fluid particles, with total mass m, total momentum $\vec{p}$ and total energy E. From basic physics (conservation laws), we know the following principles hold:

- Conservation of mass (mass balance):

$$\frac{\mathrm{d}m}{\mathrm{d}t} = 0$$

- Newton's second law (momentum balance)

$$\frac{{}^{i}\mathrm{d}\vec{p}}{\mathrm{d}t} = \vec{F}$$

Also holds for angular momentum, $\vec{h} = \vec{r} \times \vec{p}$:

$$\frac{{}^{i}\mathrm{d}}{\mathrm{d}t}\vec{h} = \vec{r} \times \vec{F} = \vec{T}$$

- First law of thermodynamics (conservation of energy, energy balance):

$$\frac{\mathrm{d}E}{\mathrm{d}t} = \mathrm{d}Q - \mathrm{d}W$$

Rate of heat flowing into volume from surroundings

Rate at which work is done by the body at surroundings

State variables for process systems

- What variables are relevant as state variable(s) for balance laws based on conservation of mass (that is, mass balances)?
  - Mass
  - Density
  - Moles, and mole concentration
  - Derived quantities: Pressure, level, …
  - (number of particles, etc.)
- For energy balances:
  - Internal energy
  - Temperature
- For momentum balances
  - Linear or angular momentum
  - Velocities

Extensive and intensive properties

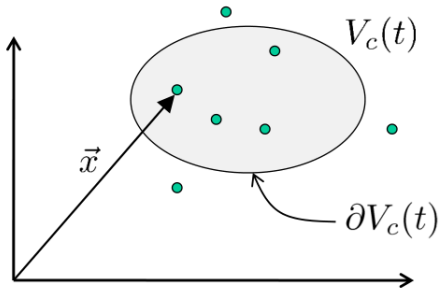
- We must choose properties (thermodynamic variables) to describe our process, and these are either *intensive* or *extensive*
- Intensive properties are scale invariant (does not change if we divide a volume in two), while extensive variables are proportional to amount of material
- In this course, we use mostly *mass-intensive* properties/variables (that is, we measure amount of material with mass):

Extensive properties	Symbol	Unit	Intensive properties	Symbol	Unit
mass	m	kg	1	-	-
volume	V	m^3	specific volume	v	m^3/kg
internal energy	U	J	specific internal energy	u	J/kg
enthalpy	H	J	specific enthalpy	h	J/kg
entropy	S	J/K	Specific entropy	s	J/K/kg

- What alternatives are there to mass-intensive?
- Is temperature an intensive or extensive property? Pressure? (yes and no…)

The concept of control volume

- We use a control volume for separating what we are interested in from the rest of the world (surroundings)
- Generally, material flow into (or out of) the control volume, across the surface



Extensive Property B of one particle

$$\mathrm{d}B = \rho(\mathbf{x})\beta(\mathbf{x},t)\mathrm{d}V$$

Summed over all particles in $V_c(t)$

$$B = \iiint_{V_c(t)} \rho(\mathbf{x})\beta(\mathbf{x},t)\mathrm{d}V$$

- We are interested in
  - knowing how the extensive property B varies inside the control volume
  - ***or*** (equivalently?) how the intensive property $\beta(\mathbf{x},t)$ varies inside the volume
- Control volumes can move or change shape, but we will assume they are fixed (more on this in fluid mechanics)

Lumped vs distributed modeling

- If we do *lumped modeling*, we assume that intensive properties are constant (or averaged) over the control volume

$$B = \iiint_{V_c} \rho\beta(\mathbf{x})\mathrm{d}V = \iiint_{V_c} \rho\bar{\beta}\mathrm{d}V = \bar{\beta}\iiint_{V_c} \rho\mathrm{d}V = m\bar{\beta}$$

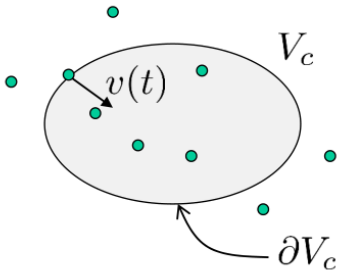
- The balance laws used for lumped modeling are the *integral* (or *macroscopic*) balance laws
  - Formulated for extensive (e.g. mass), or averaged intensive (e.g. average temperature), variables

- The alternative to lumped modeling is *distributed* modeling, where we are interested in how $\beta(\mathbf{x},t)$ varies as a function of position ***x***
- The balance laws for distributed modeling are the differential balance laws

(This course: Mainly lumped modeling and integral balance laws)

The balance laws

- Assume a fixed control volume (of arbitrary size and shape), where fluid flows across the control volume

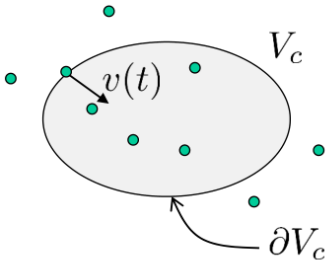


$$B = \iiint_{V_c} \rho\beta(\mathbf{x}, t)\mathrm{d}V$$

- The general integral (macroscopic) balance law is

$$\frac{\mathrm{d}}{\mathrm{d}t}B = \left\{ \begin{array}{c} \text{transfer of } B \text{ through} \\ \text{surface } \partial V_c \text{ by} \\ \text{fluid flow (convection)} \end{array} \right\} + \left\{ \begin{array}{c} \text{other effects that} \\ \text{transfer } B \text{ into } V_c \\ \text{(indep. of fluid flow)} \end{array} \right\}$$

The balance laws



- Mass balance (without reactions/phase transfer)

$$\frac{\mathrm{d}}{\mathrm{d}t}m = \left\{ \begin{array}{c} \text{transfer of mass into} \\ V_c \text{ by fluid flow} \\ \text{across surface } \partial V_c \end{array} \right\}$$

- Momentum (note: momentum is a vector)

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{p} = \left\{ \begin{array}{c} \text{transfer of momentum into} \\ V_c \text{ by fluid flow} \\ \text{across surface } \partial V_c \end{array} \right\} + \left\{ \begin{array}{c} \text{generation of momentum} \\ \text{in } V_c \text{ due to forces} \\ \text{acting on } V_c \end{array} \right\}$$

- Energy

$$\frac{\mathrm{d}}{\mathrm{d}t}E = \left\{ \begin{array}{c} \text{transfer of energy into} \\ V_c \text{ by fluid flow} \\ \text{across surface } \partial V_c \end{array} \right\} + \left\{ \begin{array}{c} \text{transfer of energy into} \\ V_c \text{ by heat transfer} \\ \text{and by work} \end{array} \right\}$$

Mathematical formulation of convection

$$\frac{\mathrm{d}}{\mathrm{d}t}B = \left\{ \begin{array}{c} \text{transfer of } B \text{ through} \\ \text{surface } \partial V_c \text{ by} \\ \text{fluid flow (convection)} \end{array} \right\} + \left\{ \begin{array}{c} \text{other effects that} \\ \text{transfer } B \text{ into } V_c \\ \text{(indep. of fluid flow)} \end{array} \right\}$$

$$-\iint_{\partial V_c} \rho\beta\vec{v}\cdot\vec{n}\mathrm{d}A$$

- (draw blackboard)

Mathematical formulation of mass balance

- For mass, the intensive variable is $\beta(\mathbf{x},t)=1$ $\quad B=\iiint_{V_c}\rho\beta\mathrm{d}V$

$$\frac{\mathrm{d}}{\mathrm{d}t}m = \left\{ \begin{array}{c} \text{transfer of mass into} \\ V_c \text{ by fluid flow} \\ \text{across surface } \partial V_c \end{array} \right\} \qquad -\iint_{\partial V_c}\rho\beta\vec{v}\cdot\vec{n}\mathrm{d}A$$

$$\frac{\mathrm{d}}{\mathrm{d}t}m = \frac{\mathrm{d}}{\mathrm{d}t}\iiint_{V_c}\rho\mathrm{d}V = -\iint_{\partial V_c}\rho\vec{v}\cdot\vec{n}\mathrm{d}A$$

  - (Example: Flow in pipe)
- Often, we have one (or more) «point inflows» $w_{\mathrm{in},i}$, and outflows $w_{\mathrm{out},i}$. Then mass balance can be formulated as

$$\frac{\mathrm{d}}{\mathrm{d}t}m = \sum_i w_{\mathrm{in},i} - \sum_i w_{\mathrm{out},i}$$

  - (Example: Tank)

Mass-type balance laws with generation

- Assume B is an extensive variable «equivalent to» mass
  - that is, mass of a component in a volume, or number of molecules of a component, number of particles, etc.
- These types of mass balance laws can have *internal generation*:

$$\frac{\mathrm{d}}{\mathrm{d}t}B = \sum_i W_{\mathrm{in},i} - \sum_i W_{\mathrm{out},i} + W_{\mathrm{generated}}$$

- More generally, if the local rate of generation of *B* is r_B:

$$\frac{\mathrm{d}}{\mathrm{d}t}B = \frac{\mathrm{d}}{\mathrm{d}t}\iiint_{V_c} \rho\beta \mathrm{d}V = -\iint_{\partial V_c} \rho\beta\vec{v}\cdot\vec{n}\mathrm{d}A + \iiint_{V_c} r_B \mathrm{d}V$$

- (Notes on multi-component systems)

Lecture 22: Balance equations – Momentum and energy balances

- Recap balance laws
- The momentum balance
- The energy balance
- Differential balance laws

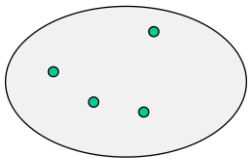
Book: Ch. 11.2, 11.4

- Remember: Survey on It's Learning
- Date for «spørretime» (Q&A session): June 2nd?

Process modeling and balance laws

- The balance laws are formulated for «conserved quantities»:
  - Mass (or other quantities that are «equivalent» to mass, such as moles, particles, etc.)
  - Momentum
  - Energy
- Process modeling is done by
  1. formulating the relevant balance laws, and
  2. finding the «closure relations» that is used to determine the flows in a balance law, as function of the state («inventory») of the balance law
- The state («inventory») of a balance law is what is used as a measure for the conserved quantity
  - Such as mass, moles, concentration, level, pressure, … for mass balance,
  - velocity or flows for momentum balance, and
  - temperature for energy balance

The basic physical principles



Consider a volume consisting of a fixed number of fluid particles, with total mass m, total momentum $\vec{p}$ and total energy E. From basic physics (conservation laws), we know the following principles hold:

- Conservation of mass (mass balance):

$$\frac{\mathrm{d}m}{\mathrm{d}t} = 0$$

- Newton's second law (momentum balance)

$$\frac{{}^{i}\mathrm{d}\vec{p}}{\mathrm{d}t} = \vec{F}$$

Also holds for angular momentum, $\vec{h} = \vec{r} \times \vec{p}$:

$$\frac{{}^{i}\mathrm{d}}{\mathrm{d}t}\vec{h} = \vec{r} \times \vec{F} = \vec{T}$$

- First law of thermodynamics (conservation of energy, energy balance):

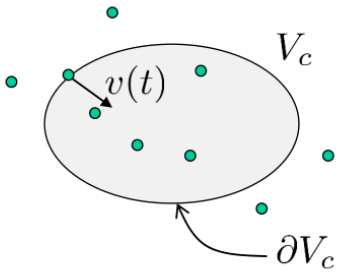
$$\frac{\mathrm{d}E}{\mathrm{d}t} = \dot{Q} - \dot{W}$$

Rate of heat flowing into volume from surroundings

Rate at which work is done by the body at surroundings

The balance laws

- Assume a fixed control volume (of arbitrary size and shape), where fluid flows across the control volume

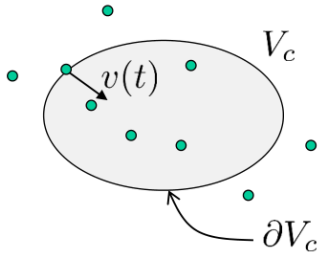


$$B = \iiint_{V_c} \rho\beta(\mathbf{x}, t)\mathrm{d}V$$

- The general integral (macroscopic) balance law for *B* is

$$\frac{\mathrm{d}}{\mathrm{d}t}B = \left\{ \begin{array}{c} \text{transfer of } B \text{ through} \\ \text{surface } \partial V_c \text{ by} \\ \text{fluid flow (convection)} \end{array} \right\} + \left\{ \begin{array}{c} \text{other effects that} \\ \text{transfer } B \text{ into } V_c \\ \text{(indep. of fluid flow)} \end{array} \right\}$$

The integral balance laws



- Mass balance (without reactions/phase transfer)

$$\frac{\mathrm{d}}{\mathrm{d}t}m = \left\{ \begin{array}{c} \text{transfer of mass into} \\ V_c \text{ by fluid flow} \\ \text{across surface } \partial V_c \end{array} \right\}$$

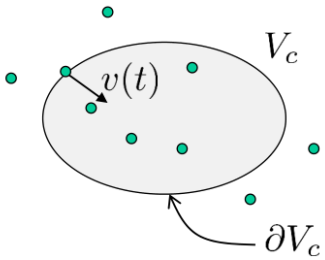
- Momentum (note: momentum is a vector)

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{p} = \left\{ \begin{array}{c} \text{transfer of momentum into} \\ V_c \text{ by fluid flow} \\ \text{across surface } \partial V_c \end{array} \right\} + \left\{ \begin{array}{c} \text{generation of momentum} \\ \text{in } V_c \text{ due to forces} \\ \text{acting on } V_c \end{array} \right\}$$

- Energy

$$\frac{\mathrm{d}}{\mathrm{d}t}E = \left\{ \begin{array}{c} \text{transfer of energy into} \\ V_c \text{ by fluid flow} \\ \text{across surface } \partial V_c \end{array} \right\} + \left\{ \begin{array}{c} \text{transfer of energy into} \\ V_c \text{ by heat transfer} \\ \text{and by work} \end{array} \right\}$$

The mass balance



- In words

$$\frac{\mathrm{d}}{\mathrm{d}t}m = \left\{ \begin{array}{c} \text{transfer of mass into} \\ V_c \text{ by fluid flow} \\ \text{across surface } \partial V_c \end{array} \right\}$$

- Mathematically

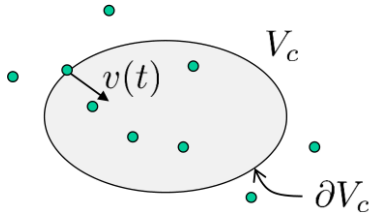
"Convection"

$$\frac{\mathrm{d}}{\mathrm{d}t}m = \frac{\mathrm{d}}{\mathrm{d}t}\iiint_{V_c} \rho \mathrm{d}V = -\iint_{\partial V_c} \rho \vec{v} \cdot \vec{n} \mathrm{d}A$$

- Often, we have one (or more) «point inflows» $w_{\mathrm{in},i}$, and outflows $w_{\mathrm{out},i}$. Then mass balance can be formulated as

$$\frac{\mathrm{d}}{\mathrm{d}t}m = \sum_i w_{\mathrm{in},i} - \sum_i w_{\mathrm{out},i}$$

The momentum balance



- In words

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{p} = \left\{ \begin{array}{c} \text{transfer of momentum into} \\ V_c \text{ by fluid flow} \\ \text{across surface } \partial V_c \end{array} \right\} + \left\{ \begin{array}{c} \text{generation of momentum} \\ \text{in } V_c \text{ due to forces} \\ \text{acting on } V_c \end{array} \right\}$$

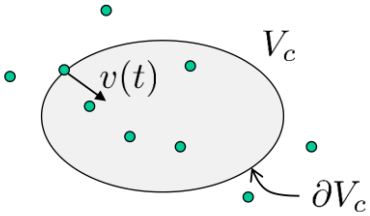
- Mathematically

$$\frac{{}^{i}\mathrm{d}}{\mathrm{d}t}\vec{p} = \frac{{}^{i}\mathrm{d}}{\mathrm{d}t}\iiint_{V_c} \rho\vec{v}\mathrm{d}V = -\iint_{\partial V_c} \rho\vec{v}\vec{v}\cdot\vec{n}\mathrm{d}A + \vec{F}^{(r)}$$

where $\vec{F}^{(r)}$ is resultant force on fluid in control volume

(often: gravity (hydrostatic) and/or friction (hydrodynamic))

The energy balance



- In words

$$\frac{\mathrm{d}}{\mathrm{d}t}E = \left\{ \begin{array}{c} \text{transfer of energy into} \\ V_c \text{ by fluid flow} \\ \text{across surface } \partial V_c \end{array} \right\} + \left\{ \begin{array}{c} \text{transfer of energy into} \\ V_c \text{ by heat transfer} \\ \text{and by work} \end{array} \right\}$$

- Mathematically

$$\frac{\mathrm{d}}{\mathrm{d}t}E = \frac{\mathrm{d}}{\mathrm{d}t}\iiint_{V_c} \rho e\mathrm{d}V = \underbrace{-\iint_{\partial V_c} \rho e\vec{v}\cdot\vec{n}\mathrm{d}A}_{\text{Energy flow by convection}} + \dot{Q} - \dot{W}$$

- What is the energy of a fluid?

Energy

$$\frac{\mathrm{d}}{\mathrm{d}t}E = \frac{\mathrm{d}}{\mathrm{d}t}\iiint_{V_c} \rho e \mathrm{d}V = -\iint_{\partial V_c} \rho e \vec{v}\cdot\vec{n}\mathrm{d}A + \dot{Q} - \dot{W}$$

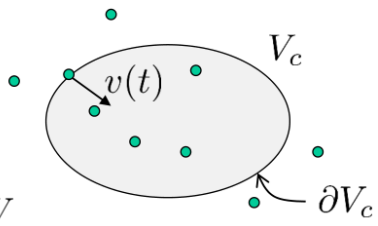
- The energy of a fluid of mass *m*, moving with a velocity *v* at a height *z* in a gravitational field:

$$E = \underbrace{U}_{\substack{\text{internal}\\ \text{energy}}} + \underbrace{\frac{1}{2}mv^2}_{\substack{\text{kinetic}\\ \text{energy}}} + \underbrace{mgz}_{\substack{\text{potential}\\ \text{energy}}}$$

- Specific energy:

$$e = u + \frac{1}{2}v^2 + gz$$

Heat and work flow



$$\frac{\mathrm{d}}{\mathrm{d}t}E = \frac{\mathrm{d}}{\mathrm{d}t}\iiint_{V_c} \rho e \mathrm{d}V = -\iint_{\partial V_c} \rho e \vec{v}\cdot\vec{n}\mathrm{d}A + \dot{Q} - \dot{W}$$

- Heat flow

$$\dot{Q} = \iint_{\partial V_c} \vec{j}_Q \cdot \vec{n}\mathrm{d}A$$

- Work flow

$$\dot{W} = \underbrace{\iint_{\partial V_c} p\vec{v}\cdot\vec{n}\mathrm{d}A}_{\text{flow work}} + \underbrace{\dot{W}_s}_{\text{shaft work}}$$

Enthalpy

- The energy balance can be written

$$\frac{\mathrm{d}}{\mathrm{d}t}\iiint_{V_c}\rho e \mathrm{d}V = -\iint_{\partial V_c}\rho\left(e+\frac{p}{\rho}\right)\vec{v}\cdot\vec{n}\mathrm{d}A - \dot{W}_s + \dot{Q}$$

  where the first term on the RHS is convection and flow work
- Define **enthalpy** as

$$h = u + \frac{p}{\rho}$$

- Then

$$\frac{\mathrm{d}}{\mathrm{d}t}\iiint_{V_c}\rho\left(u+\frac{1}{2}v^2+gz\right)\mathrm{d}V = -\iint_{\partial V_c}\rho\left(h+\frac{1}{2}v^2+gz\right)\vec{v}\cdot\vec{n}\mathrm{d}A - \dot{W}_s + \dot{Q}$$

Internal energy and enthalpy

- Specific heat capacities:

$$c_v := \left.\frac{\partial u}{\partial T}\right|_{\text{constant volume}} \qquad c_p := \left.\frac{\partial h}{\partial T}\right|_{\text{constant pressure}}$$

  (found in tables for different fluids, often assumed constant)
- If assumed constant, implies that energy and enthalpy is (linear) function of temperature only:

$$\frac{\mathrm{d}u}{\mathrm{d}t} = c_v\frac{\mathrm{d}T}{\mathrm{d}t} \qquad u(T_2)-u(T_1) = c_v\left(T_2-T_1\right)$$

$$\frac{\mathrm{d}h}{\mathrm{d}t} = c_p\frac{\mathrm{d}T}{\mathrm{d}t} \qquad h(T_2)-h(T_1) = c_p\left(T_2-T_1\right)$$

- For ideal gases:

$$c_v = c_p + R$$

- For incompressible fluids (often assumed for liquids):

$$c_v = c_p$$

Differential mass balance

- Recall the integral mass balance:

$$\underbrace{\frac{\mathrm{d}}{\mathrm{d}t}\iiint_{V_c}\rho \mathrm{d}V}_{\text{Mathematics (obvious?)}} = -\underbrace{\iint_{\partial V_c}\rho\vec{v}\cdot\vec{n}\mathrm{d}A}_{\text{Divergence theorem}}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\iiint_{V_c}\rho \mathrm{d}V = \iiint_{V_c}\frac{\partial\rho}{\partial t}\mathrm{d}V \qquad \iint_{\partial V_c}\rho\vec{v}\cdot\vec{n}\mathrm{d}A = \iiint_{V_c}\vec{\nabla}\cdot(\rho\vec{v})\mathrm{d}V$$

- That is:

$$\iiint_{V_c}\frac{\partial\rho}{\partial t} + \vec{\nabla}\cdot(\rho\vec{v})\mathrm{d}V = 0$$

- This must hold for arbitrary control volumes, which implies

$$\frac{\partial\rho}{\partial t} + \vec{\nabla}\cdot(\rho\vec{v}) = 0$$

Differential mass balance, also called *continuity equation* or *advection equation*

Alternative formulations

- The differential mass balance

$$\frac{\partial\rho}{\partial t} + \vec{\nabla}\cdot(\rho\vec{v}) = 0$$

- From definition of nabla operator, this is the same as

$$\frac{\partial\rho}{\partial t} + \sum_{i=1}^{3}\frac{\partial}{\partial x_i}(\rho v_i) = 0, \quad \mathbf{v} = (v_1, v_2, v_3)^{\mathsf{T}}$$

- If we introduce the *material derivative*,

$$\frac{\mathrm{D}\phi}{\mathrm{D}t} := \frac{\partial\phi}{\partial t} + \mathbf{v}^{\mathsf{T}}\nabla\phi = \frac{\partial\phi}{\partial t} + \sum_{i=1}^{3}\frac{\partial\phi}{\partial x_i}v_i$$

The material derivative is the derivative following a particle (as opposed to the derivative at a fixed point in space)

and use product rule, we can write

$$\frac{\mathrm{D}\rho}{\mathrm{D}t} + \nabla\rho\cdot\vec{v} = 0$$

Differential momentum and energy balances

- Differential momentum balance for inviscid fluid (*Euler's equation*)

$$\rho\frac{\mathrm{D}\vec{v}}{\mathrm{D}t} = -\vec{\nabla}p + \rho\vec{f}, \quad \text{where } \rho\vec{f} \text{ is the mass force (e.g. gravity)}$$

- For viscous (Newtonian) fluids, the differential momentum balance is the famous *Navier-Stokes* equation:

$$\rho\frac{\mathrm{D}\vec{v}}{\mathrm{D}t} = -\vec{\nabla}p + \mu\vec{\nabla}^2\vec{v} + \rho\vec{f}$$

- Differential energy balance (for example)

$$\rho\frac{\mathrm{D}}{\mathrm{D}t}\left(\frac{1}{2}\vec{v}^2 + u\right) = -\vec{\nabla}\cdot(p\vec{v}) - \vec{\nabla}\cdot\vec{j}_Q + \rho\vec{v}\cdot\vec{f}$$

Example of differential energy balances: The heat equation of a solid

- The energy balance:

$$\rho\frac{\mathrm{D}}{\mathrm{D}t}\left(\frac{1}{2}\vec{v}^2 + u\right) = -\vec{\nabla}\cdot(p\vec{v}) - \vec{\nabla}\cdot\vec{j}_Q + \rho\vec{v}\cdot\vec{f}$$

- Solid: Disregard kinetic and potential energy, no velocity:

$$\rho\frac{\partial u}{\partial t} = -\vec{\nabla}\cdot\vec{j}_Q$$

- We need a «closure relation». Here in the form of *Fourier's law*:

$$\vec{j}_Q = -\alpha\vec{\nabla}(\rho c_p T)$$

- Combined with

$$\frac{\partial u}{\partial t} = c_p\frac{\partial T}{\partial t}$$

we get

$$\frac{\partial T}{\partial t} - \alpha\vec{\nabla}\cdot\vec{\nabla}T = 0$$

In one dimension:

$$\frac{\partial T(x,t)}{\partial t} - \alpha\frac{\partial^2 T(x,t)}{\partial x^2} = 0$$