MS_Project-1 ODE Simulation

Given is an ODE simulator written in Matlab command language for numerical integration of ODE systems. The simulator is preconfigured as "Simulation demo", see attachment.

(a) Implement the following ODE system model in the simulator:

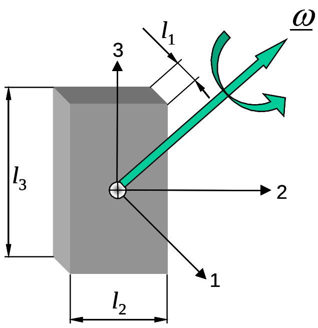
$$\boxed{\dot{x} = \lambda \cdot x} \tag{1}$$

Test the simulator now with the new system model and the built in ODE-solver (Runge-Kutta-3, RK3) and compare with the exact solution for different integration step size (simulation model verification).

(b) Implement as alternative numerical integration algorithm the EULER-algorithm.
Test the simulator with system model (1) and compare with the exact solution for different integration step size (simulation model verification).
Compare the numerical stability regions of the RK3 and EULER algorithms.

(c) Implement as alternative numerical integration algorithm the TRAPEZOID integration algorithm.
Test the simulator with system model (1) and compare with the exact solution for different integration step size (simulation model verification).
Compare the numerical stability regions of the RK3, EULER and TRAPEZOID algorithms.

Simulation demo: Free rotation of a rigid body



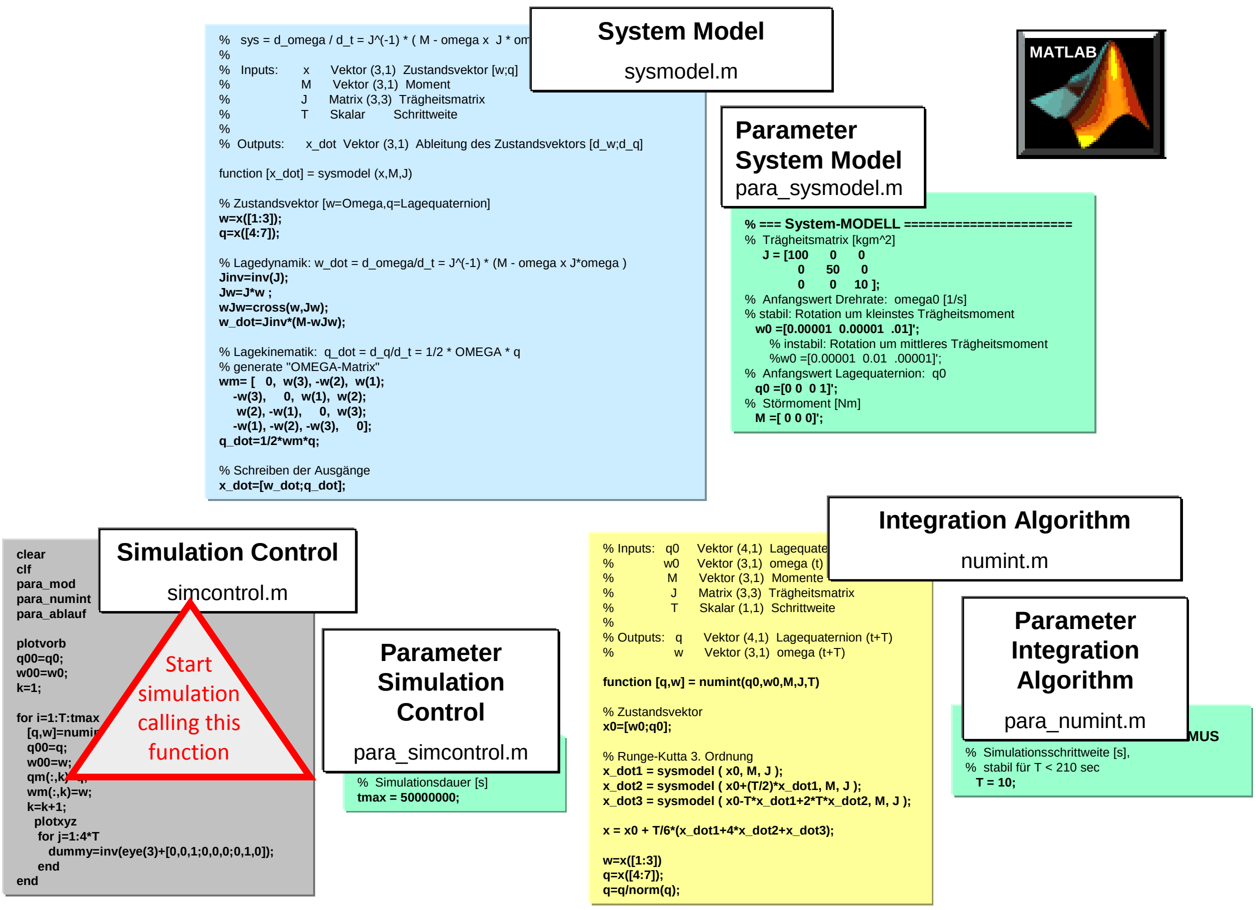
$$I_1 = \frac{m}{12}\left(l_2^2 + l_3^2\right)$$
$$I_1 > I_2 > I_3$$

Experiment: free rotation, external torques $M_1 = M_2 = M_3 = 0$
rigid body
rotation about principal axis

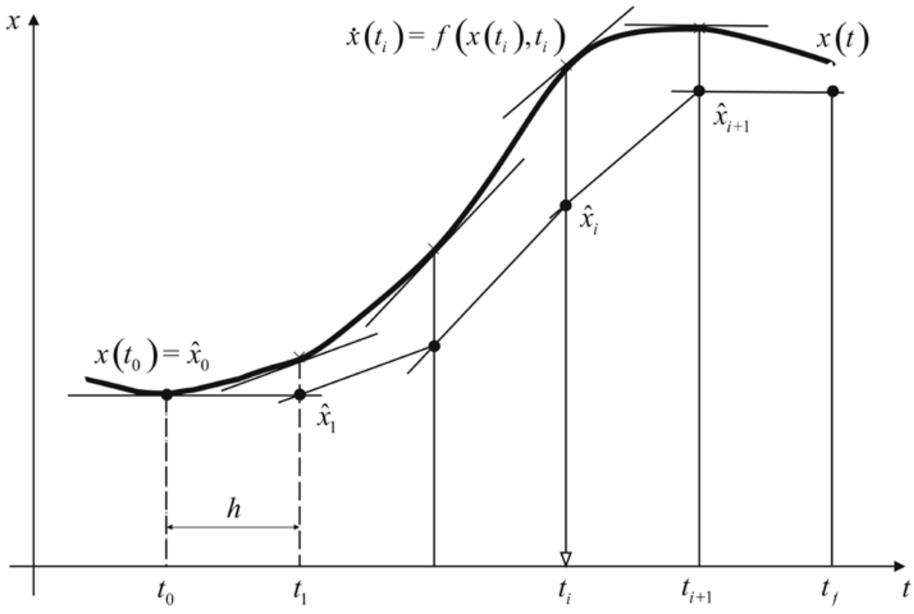
Abstract model: rigid body, Euler gyroscopic equations (nonlinear)

$$I_1\dot{\omega}_1 + \omega_2\omega_3\left(I_3 - I_2\right) = M_1$$
$$I_2\dot{\omega}_2 + \omega_1\omega_3\left(I_1 - I_3\right) = M_2$$
$$I_3\dot{\omega}_3 + \omega_1\omega_2\left(I_2 - I_1\right) = M_3$$

Architecture of the simulation program



EULER algorithm (EUL)

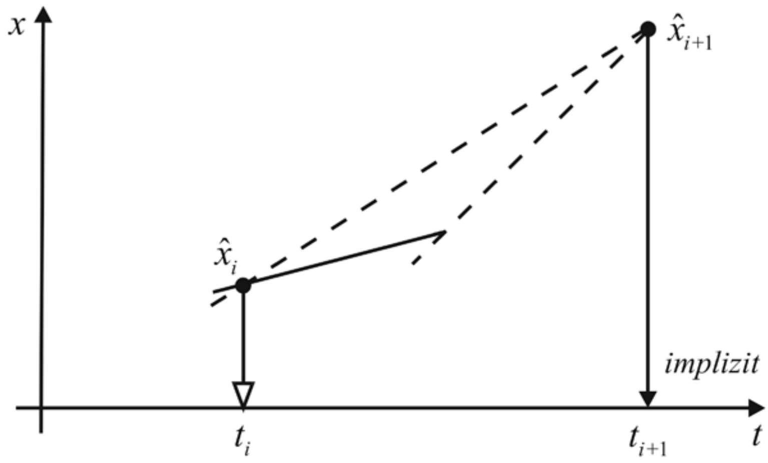


∇... evaluation of right side of ODE $\underline{f}\left(\underline{x}, t\right)$; $h = t_{i+1} - t_i$

Integration

$$\boxed{\begin{aligned} \underline{\hat{x}}_0 &= \underline{x}\left(t_0\right) \\ \underline{\hat{x}}_{i+1} &= \underline{\hat{x}}_i + h \cdot \underline{f}\left(\underline{\hat{x}}_i, t_i\right) \end{aligned}}$$

Trapezoid algorithm (TRA)



∇... evaluation of right side of ODE $\underline{f}\left(\underline{x},t\right)$; $h = t_{i+1} - t_i$

Integration

$$\boxed{\begin{aligned}&\underline{\hat{x}}_0 = \underline{x}\left(t_0\right)\\&\underline{\hat{x}}_{i+1} = \underline{\hat{x}}_i + \frac{h}{2}\left(\underline{f}\left(\underline{\hat{x}}_i, t_i\right) + \underline{f}\left(\underline{\hat{x}}_{i+1}, t_{i+1}\right)\right)\end{aligned}}$$

Implicit numerical integration of ODE systems – trapezoid algorithm

ODE system $\dot{\mathbf{x}} = \mathbf{f}\left(\mathbf{x}, t\right)$ $\quad \mathbf{x} \in \mathbb{R}^n$

General solution approach

TRAPEZOID ALGORITHM

$$\hat{\mathbf{x}}_{i+1} = \hat{\mathbf{x}}_i + \frac{h}{2}\left[\mathbf{f}\left(\hat{\mathbf{x}}_i, t_i\right) + \mathbf{f}\left(\hat{\mathbf{x}}_{i+1}, t_{i+1}\right)\right]$$

IMPLICIT NONLINEAR ALGEBRAIC EQUATION SOLVER

$$\mathbf{p} := \hat{\mathbf{x}}_{i+1}$$

$$\varphi\left(\mathbf{p}\right) = \mathbf{p} - \hat{\mathbf{x}}_i - \frac{h}{2}\left[\mathbf{f}\left(\hat{\mathbf{x}}_i, t_i\right) + \mathbf{f}\left(\mathbf{p}, t_{i+1}\right)\right]$$

➔ $\varphi\left(\mathbf{p}\right) = \mathbf{0}$

NEWTON-RAPHSON $\quad \mathbf{p}_{k+1} = \mathbf{p}_k - \mathbf{J}\left(\mathbf{p}\right)^{-1} \cdot \varphi\left(\mathbf{p}_k\right)$

Iterations over k, until $\left\|\mathbf{p}_{k+1} - \mathbf{p}_k\right\| \le \varepsilon$

JACOBIAN

$$\mathbf{J}\left(\mathbf{p}\right) = \frac{\partial \varphi}{\partial \mathbf{p}} = \begin{bmatrix} \frac{\partial \varphi_1}{\partial p_1} & \cdots & \frac{\partial \varphi_1}{\partial p_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial \varphi_n}{\partial p_1} & \cdots & \frac{\partial \varphi_n}{\partial p_n} \end{bmatrix}$$

➔ The JACOBIAN determines the solvability of the NEWTON-RAPHSON algorithm and such the solvability of the implicit numerical integration algorithm.