Theoretical Mechanics, Lab 12: DYN NEWTON EULER

Approaches to dynamic modeling

Model-oriented design

Newton-Euler method

Three global approaches to dynamic modeling

Modeling frameworks

Ex, Coding (1): Pinocchio, RBDL

Ex, Model-oriented design (2): MATLAB Simulink, SimInTech, Open Modelica

+ You understand your system fully and can control any part of it (1,2)
+ Easy to apply the solution to real system (1, 2)
+ Often open source frameworks (1)
+ For part of the tasks, no coding (2)
+ Fast solution for big systems (1)
- Complicated to read and debug (1) (2 for big systems)
- Big time consumptions (1)

Good for control applications (1,2)

Simulators

Ex: Gazebo Sim, Mujoco, CoppeliaSim

+ No need to code dynamics, only description for a system
+ Made for simulating the system as whole (with sensors and so on)
- Need special software (most — open source)
- Not so easy to transfer the solution to a real system
- Physics models are rather simple.
- Often can't work with closed-loop systems

Well suit for simulate the whole system

Computer Aided Engineering (CAE)

Ex, CAE: Siemens NX, ANSYS, Solid Works

+ The most intuitive systems (mostly interface based manipulation)
+ For most of the tasks, no coding
+ Physics models are quite complicated
- Have to make a CAD model, even if the system is simple.
- Need special software (most — proprietary)
- Very slow computations

Best for analyzing part of system

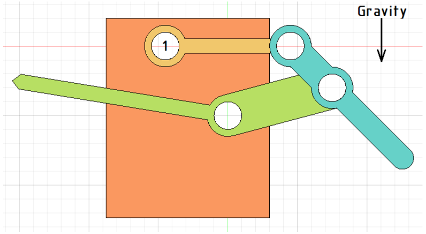
MaM HW CAE DYN1

Zip archive, which contains all needed data:

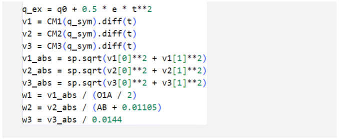
HWs/HW_CAE_DYN1/task_data

1st joint is controllable, others — not.

1. **Find angle limits** (where the mechanism stuck) for controllable joint: By code (solving kinematics problem for each angle)
   - Using NX (either Modeling, or Animation Designer);
2. Compare results, present them as a pie chart in report.
3. **Make the scene in Motion Analysis.** All links are made from «Bronze». You need to add joints, contacts, direct earth gravity correctly.
4. Choose the biggest angle gap between joint limits and put your link in the beginning of it.
5. Apply constant angular acceleration for 1st joint — 0.2 rad/s^2
6. **Find a torque for 1st joint** for such angle gap:
   - By code (solving Inverse dynamics problem)
   - Using NX (any solver);
7. Compare results

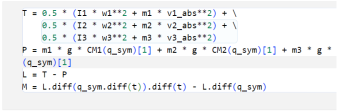


Solution (1)



```
q_ex = q0 + 0.5 * e * t**2
v1 = CM1(q_sym).diff(t)
v2 = CM2(q_sym).diff(t)
v3 = CM3(q_sym).diff(t)
v1_abs = sp.sqrt(v1[0]**2 + v1[1]**2)
v2_abs = sp.sqrt(v2[0]**2 + v2[1]**2)
v3_abs = sp.sqrt(v3[0]**2 + v3[1]**2)
w1 = v1_abs / (O1A / 2)
w2 = v2_abs / (AB + 0.01105)
w3 = v3_abs / 0.0144
```

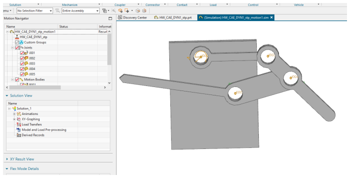
Energy of the system



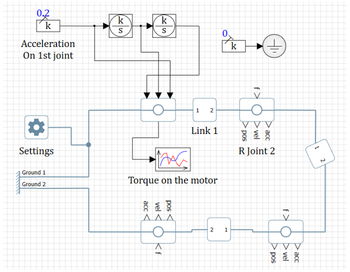
```
T = 0.5 * (I1 * w1**2 + m1 * v1_abs**2) + \
    0.5 * (I2 * w2**2 + m2 * v2_abs**2) + \
    0.5 * (I3 * w3**2 + m3 * v3_abs**2)
P = m1 * g * CM1(q_sym)[1] + m2 * g * CM2(q_sym)[1] + m3 * g *
(q_sym)[1]
L = T - P
M = L.diff(q_sym.diff(t)).diff(t) - L.diff(q_sym)
```

˅ Simulation (takes up to 1-2 minutes)

Coding

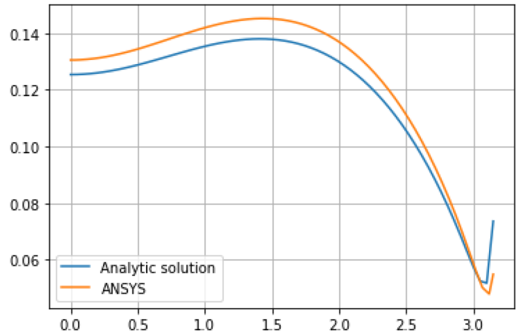


Siemens NX (CAE)

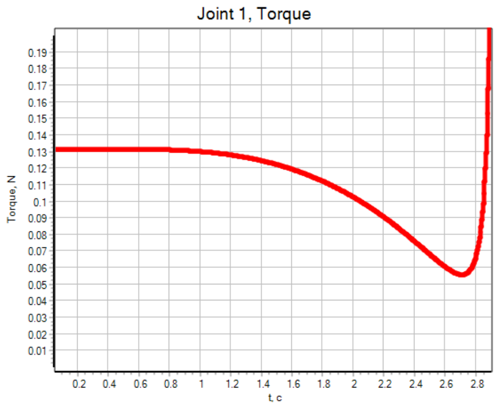


SimInTech (Model-oriented design)

Solution (2)



Plots CAE and Code



Plot Model-oriented design

Two basic concepts of solving dynamics by coding (1)

energy-based approach (Euler-Lagrange) $\rightleftharpoons$ **Newton-Euler method (balance of forces/torques)**

energy-based approach (Euler-Lagrange)

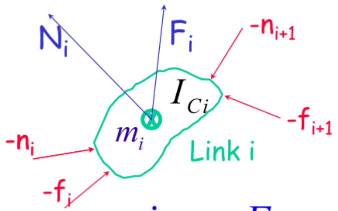
- multi-body robot seen as a whole
- constraint (internal) reaction forces between the links are automatically eliminated: in fact, they do not perform work
- closed-form (symbolic) equations are directly obtained
- best suited for study of dynamic properties and **analysis** of control schemes

Newton-Euler method (balance of forces/torques)

- dynamic equations written separately for each link/body
- inverse dynamics in real time
  - equations are evaluated in a numeric and recursive way
  - best for **synthesis** (=implementation) of model-based control schemes
- by elimination of reaction forces and back-substitution of expressions, we still get closed-form dynamic equations (identical to those of Euler-Lagrange!)

Two basic concepts of solving dynamics by coding (2)

Newton-Euler



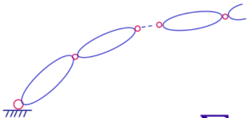
Newton: $m\dot{\mathbf{v}}_C = F$

Euler: $N_i = I_{C_i}\dot{\omega}_i + \omega_i \times I_{C_i}\omega_i$

Eliminate Internal Forces

$$\tau_i = \begin{cases} n_i^T . Z_i & \text{revolute} \\ f_i^T . Z_i & \text{prismatic} \end{cases}$$

Lagrange



Kinetic Energy: $\sum_i K_i$

Potential Energy

Generalized Coordinates

$$K = \frac{1}{2}\dot{q}^T M \dot{q}$$

$$M\ddot{q} + V + G = \tau$$

State of Art tools for coding

Classical methods are rarely used in practice anymore, although new life has been breathed into them by the ability to use GPUs for computation.

Inverse dynamics — recursive Newton-Euler algorithm (RNEA).
Forward dynamics — articulated body algorithm (ABA), Kane's method, Composite Rigid Body Algorithm (for finding the M matrix) + RNEA.
For adding links — Featherstone algorithm or Lagrange multipliers.

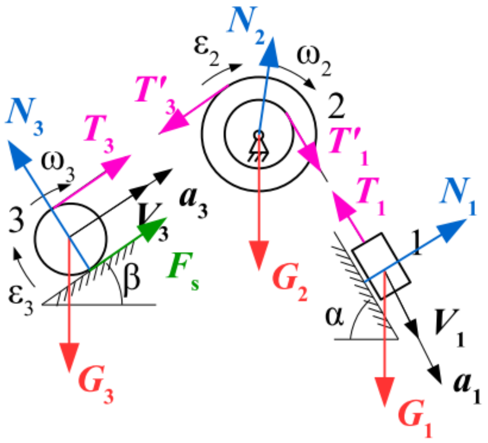
Libraries for automatic differentiation: Jax, CasADi, PyTorch.
Libraries for automated dynamics writing: Pinocchio, Rigid Body Dynamics Library (RBDL).

Newton-Euler method

R. O.	Eqn #	Equations	Applications	Extra Info
System	$6k + \sum_0^k (6 - m_i)$, k – amount of bodies, m_i – particular joint d.o.f	$\begin{cases} f_i = F_i + f_{i+1} \\ m_i = M_i + m_{i+1} + \vec{p}_{c_i} \times F_i + \vec{p}_{i+1} \times f_{i+1} \\ \tau_i = \begin{cases} m_i \cdot Z_i, \text{ if revolute} \\ f_i \cdot Z_i, \text{ if prismatic} \end{cases} \end{cases}$ Where $F_i = m\dot{\vec{v}}_{C_i}$ $M_i = I_{C_i}\dot{\omega}_i + \omega_i \times I_{C_i}\omega_i$	To find accelerations, forces, motion equations.	We divide a system into separate bodies, write equations for rotations and translations, plus constraints. At the end we have a dozens of equations, which should be solved numerically.

Task 1 (mine)

Determine the motion of the system. All needed parameters are given.

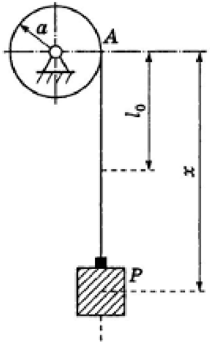


Task 2 (yours): M (rus) 48.5 (answer only)

Determine the motion of a mass m hanging on a homogeneous cable of mass m_1 and length l. The cable is wound on a drum of radius a and mass m_2.

The axis of rotation is horizontal; friction is neglected, the mass of the drum is considered to be uniformly distributed along its rim.
At the initial moment $t = 0$ the system was at rest, the length of the dangling part of the cable l_0.

Answer: $(m + m_1 + m_2)\ddot{x} = \dfrac{m_1}{l} gx + mg$



Reference material

- Robotics 2 course, Sapienza Di Roma, Alessandro De Luca
- Introduction to Robotics, Stanford, Oussama Khatib
- SimInTech
- SimInTech examples of «Mechanics» and «Mechanics 3D» libraries
- MATLAB Simulink
- Siemens NX

Deserve “A” grade!

– Oleg Bulichev

o.bulichev@innopolis.ru

@Lupasic

Room 105 (Underground robotics lab)