Simulation of a simple crane lifting a load while rotating

April 9, 2023

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 860124.

Contents

1 Description of the simple crane model

The example simulates the motion of a simple crane. The configuration of the system is shown in Figure 1. The parameters of the system are given in Table 1. As sketched in Figure 1, the system subjected to two imposed motions: rotation of the crane around the vertical Z axis, or *slewing motion*, and rotation of the drive reel to lift the payload, or *winding motion*. The velocity profiles of the winding motion and slewing motion start at zero and smoothly reach a constant value following the laws:

$$V = \begin{cases} \frac{1}{2}V_c(1-\cos(\frac{\pi t}{t_c})) & t < t_c \\ V_c & t \geq t_c \end{cases} \tag{1}$$

$$\Omega = \begin{cases} \frac{1}{2}\Omega_c(1-\cos(\frac{\pi t}{t_c})) & t < t_c \\ \Omega_c & t \geq t_c \end{cases} \tag{2}$$

where the assumed constant velocity values are $V_c = 1\,\mathrm{m/s}$, and $\Omega_c = 0.3\,\mathrm{rad/s}$, and the acceleration period is $t_c = 0.2\,s$ for both motions.

Table 1: Reeving system parameters

Parameter	Value	Units	Description
d	3	m	Distance between drive reel and deviation pulley
l_0	8	m	Undeformed length of the vertical rope span at $t = 0$
b	0.3	m	Half side of cubic payload
R	0.1	m	Radius of the drive reel and deviation pulley
ρA	0.6205	$\mathrm{kg\,m^{-1}}$	Wire ropes linear density
EA	16.5	MN	Axial stiffness
EI	30.9	$\mathrm{N\,m^2}$	Bending stiffness
M	1000	kg	Mass of the payload

The simulation procedure is the Arbitrary Lagrangian-Eulerian Modal approach described in [1,2]. All details about the formulation can be found in the reference papers. The model includes 1 rigid body which is the payload, and 2 wire-rope elements. The total set of coordinates is given by:

$$\mathbf{p} = \begin{bmatrix} \mathbf{q}^{2T} & \mathbf{q}^{aT} & \mathbf{q}^{bT} \end{bmatrix}^T \tag{3}$$

where $\mathbf{q}^2$ is the set of coordinates of rigid body 2. These are 3 transnational coordinates and 4 Euler parameters:

$$\mathbf{q}^2 = \begin{bmatrix} \mathbf{r}^{2T} & \boldsymbol{\theta}^{2T} \end{bmatrix}^T \tag{4}$$

$$\mathbf{r}^2 = \begin{bmatrix} r_x^{2T} & r_y^{2T} & r_z^{2T} \end{bmatrix}^T, \boldsymbol{\theta}^2 = \begin{bmatrix} \theta_0^2 & \theta_1^2 & \theta_2^2 & \theta_3^2 \end{bmatrix}^T \tag{5}$$

Each ALEM wire-rope element a and b is kinematically described with a set of 6 absolute nodal coordinates $\mathbf{q}_a$, 2 arc-length material coordinates $\mathbf{q}_s$, and a maximum of 3 modal coordinates in each of the 3 local directions included in $\mathbf{q}_m$. For example, for the wire-rope a, these coordinates are given by:

$$\mathbf{q}_a^a = \begin{bmatrix} \mathbf{q}_a^{aT} & \mathbf{q}_s^{aT} & \mathbf{q}_m^{a\,T} \end{bmatrix}^T \tag{6}$$

$$\mathbf{q}_a^a = \begin{bmatrix} \mathbf{r}_1^{aT} & \mathbf{r}_2^{aT} \end{bmatrix}^T, \mathbf{q}_s^a = \begin{bmatrix} s_1^a & s_2^a \end{bmatrix}^T \tag{7}$$

$$\mathbf{q}_m^a = \begin{bmatrix} q_{x,1}^a & q_{x,2}^a & q_{x,3}^a & q_{y,1}^a & q_{y,2}^a & q_{y,3}^a & q_{z,1}^a & q_{z,2}^a & q_{z,3}^a \end{bmatrix}^T \tag{8}$$

In this example, no modal coordinates are considered for the rope element a. For the rope element b, 3 axial-modal coordinates in the longitudinal direction X_b, q_{mXi}^b, $i = 1, 2, 3$, 1 transverse-modal coordinate in the local direction Y_b, q_{mY1}^b, and 1 transverse-modal coordinate in the local direction Z_b, q_{mZ1}^b, are used in the model. Therefore, the set of total coordinates $\mathbf{p}$ is subjected to linear and nonlinear constraints. The linear constraints are expressed as:

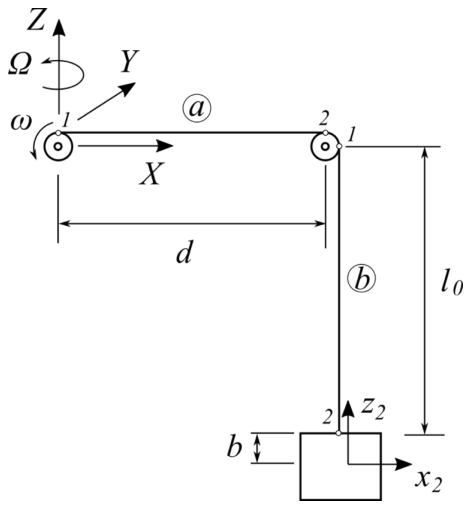

Figure 1: ALEM model of the reeving system of a simple crane

$$\mathbf{C}_{lin}\left(\mathbf{p},t\right)=\begin{bmatrix} \mathbf{r}_1^a-\left[\begin{array}{ccc} 0 & 0 & R \end{array}\right]^T \\ \mathbf{r}_2^a-\left[\begin{array}{ccc} d\cos\alpha(t) & d\sin\alpha(t) & R \end{array}\right]^T \\ \mathbf{r}_1^b-\left[\begin{array}{ccc} (d+R)\cos\alpha(t) & (d+R)\sin\alpha(t) & 0 \end{array}\right]^T \\ \mathbf{r}_2^b-\left(\mathbf{r}^2+\mathbf{A}^2\left[\begin{array}{ccc} 0 & 0 & b \end{array}\right]^T\right)^T \\ s_1^a-\int V dt \\ s_2^a-s_1^b \\ s_2^b-(d+l_0) \\ \mathbf{q}_m^a \\ q_{mY2}^b \\ q_{mY3}^b \\ q_{mZ2}^b \\ q_{mZ3}^b \end{bmatrix}=\mathbf{0} \tag{9}$$

This is a set of 28 equations, $m_l = 28$.

Nonlinear constraints are due to the force balance of the wire rope elements at the deviation sheave and to the unity of the quaternions used to describe the rotation of the payload, as follows:

$$\mathbf{C}_{nonlin}\left(\mathbf{p}\right)=\left[\begin{array}{c} F_{ax2}^a\left(\mathbf{q}^a\right)-F_{ax1}^b\left(\mathbf{q}^b\right) \\ \left(\theta_0^2\right)^2+\left(\theta_1^2\right)^2+\left(\theta_2^2\right)^2+\left(\theta_3^2\right)^2-1 \end{array}\right]=\mathbf{0} \tag{10}$$

This is a set of 2 equations, $m_{nl} = 2$.

The number of generalized coordinates $\mathbf{q}$ of the system , n, equals the number of coordinates in $\mathbf{p}$, n_p, minus the number of linear constraints: $n = n_p - m_l = 41 - 28 = 13$. The array $\mathbf{q}$ is given by:

$$\mathbf{q}=\left[\begin{array}{ccccccccccccc} r_x^2 & r_y^2 & r_z^2 & \theta_0^2 & \theta_1^2 & \theta_2^2 & \theta_3^2 & s_2^a & q_{mX1}^b & q_{mX2}^b & q_{mX3}^a & q_{mY1}^b & q_{mZ1}^b \end{array}\right]^T \tag{11}$$

The generalized coordinates of the system $\mathbf{q}$ is partitioned into independent coordinates, $\mathbf{q}_{ind}$, and dependent coordinates, $\mathbf{q}_{dep}$, as follows:

$$\mathbf{q}^{ind}=\left[\begin{array}{ccccccccccc} r_x^2 & r_y^2 & r_z^2 & \theta_0^2 & \theta_1^2 & \theta_2^2 & q_{mX1}^b & q_{mX2}^b & q_{mX3}^a & q_{mY1}^b & q_{mZ1}^b \end{array}\right]^T \tag{12}$$

$$\mathbf{q}^{dep}=\left[\begin{array}{cc} \theta_0^3 & s_2^a \end{array}\right]^T \tag{13}$$

where the number of independent coordinates n_{dep} equals the number of nonlinear constraints, m_{nl}. That way, the set $\mathbf{q}_{dep}$ can be obtained solving the nonlinear constraints $\mathbf{C}_{nonlin}$ once the value of set $\mathbf{q}_{ind}$ is known. Once the set $\mathbf{q}$ is known, the linear constraints $\mathbf{C}_{lin}$ is used to find the value of the total set of coordinates $\mathbf{p}$.

2 Description of code

The code implements a systematic formulation of the dynamics of wire-rope reeving systems modeled as flexible multibody systems using the arbitrary Lagrangian-Eulerian modal approach (ALEM) described in [1]. This method can efficiently be used for the simulation of many industrial applications such as cranes, printers, elevators, and deployable structures for common operating conditions.

2.1 Organization of the code

The program is a set of Matlab files that has been grouped into 5 main sub-folders as:

- **GeneralReevingSystems**: This folder contains the general m-files to simulate the dynamics of any reeving system.
- **WireRopeForces**: This folder contains m-files that compute the forces and mass matrix of an ALEM wire-rope element. These files have been generated using symbolic calculation in Matlab.
- **Solver**: This folder contains a few numerical methods for equations solving that can be used as an alternative to the Matlab built-in functions.
- **mexInterface**: This folder contains MEX files replace some m-files to decrease the computational time needed to solve the systems dynamics.
- **simpleCrane**: This folder contains the m-files related to the presented example. These m-files describe the model parameters, coordinates, and the prescribed motion of the model.

2.2 Phases of simulation

The phases of the simulation are described following the main file **simulateSimpleCrane.m** that can be found in the folder *src/simpleCrane/*. The inputs and outputs of the used functions are described next.

PHASE 1. Adding folders to Matlab path

In the lines 1 - 5, the folders containing the m-files used in the simulation are added to the Matlab path, as follows:

```
restoredefaultpath;
addpath("../../GeneralReevingSystems");
addpath("../../WireRopeForces");
addpath("../../mexInterface");
addpath("RB_kinematics");
```

PHASE 2. Getting system parameters

In line 8, the system parameters are stored in a data structure called `reevSys` that is the output of the function **CraneParams.m**. The different coordinate sets, the geometric, inertial and viscoelastic parameters of the system, the prescribed motion and the simulation parameters are defined and included as fields of the data structure `reevSys`.

```
% Getting system parameters %
reevSys = CraneParams;
```

Function: `CraneParams.m`
Input: No input
Output: Data structure `reevSys`

PHASE 3. Calculation of static equilibrium position

The calculation of the static equilibrium position requires the solution of the non-linear algebraic equations in which the equations of motion turn when the generalized velocities and accelerations are set to zero. The unknowns of these equations are the value of the generalized coordinates and the Lagrange multipliers in static equilibrium. Lines 14 - 24 sets the initial guess of these variables. For the generalized coordinates, the initial guess is the reference value of the coordinates at the initial instant `qref`, corrected with an estimation of the rope deformation. For the Lagrange multipliers, a zero initial guess is used.

```
% Reference value of coordinates (undeformed configuration) %
qref = [reevSys.d+reevSys.R 0.0 -(reevSys.l0 + reevSys.b) 1.0
   0.0 0.0 0.0 reevSys.d 0.0 0.0 0.0 0.0 0.0]';
n = reevSys.n;

% Estimation of static displacement %
delta = reevSys.m2*9.81/(reevSys.EA/(reevSys.l0+reevSys.d));
   % Elongation of both ropes
qref(3) = qref(3) - delta;

% Initial guess of Lagrange multipliers associated with axial
   load
% constraint and Euler parameters %
lam = [0 0]';
```

In lines 26 - 31, the static equilibrium position is calculated using the function `StaticEqReevingSystem.m`. Depending on the value of the parameter `reevSys.NLeq-method` different solvers can be used.

```
% Calculation of static equilibrium position %
if strcmp(reevSys.NLeq_method,'NewtonRaphson')
    q0lam = NewtonRaphson(@StaticEqReevingSystem,[qref' lam
       ']',reevSys);
elseif strcmp(reevSys.NLeq_method,'fsolve')
    q0lam = fsolve(@(qlam) StaticEqReevingSystem(qlam,reevSys)
       ,[qref' lam']',optimset('TolFun',1e-3));
end
```

Function: `StaticEqReevingSystem.m`
Inputs: Array with generalized coordinates and Lagrange multipliers `qlam`. Data structure `reevSys.qdep0`
Output: Non-linear equilibrium equations

In lines 33 -41, the initial value of the independent generalized coordinates `qind0` and the initial guess of the dependent generalized coordinates `reevSys.qdep0` are extracted from the calculated static equilibrim position. Besides, the value of all coordinates at the static equilibrium position is obtained with function `CalculateAllCoordinates.m`.

```
% Generalized coordinates in static equilibrium %
qst = q0lam(1:n,1);
% Initial value of independent coordinates %
qind0 = q0lam(reevSys.Ind);
% Initial guess of dependent coordinates %
reevSys.qdep0 = q0lam(reevSys.Dep);

% Calculation of all coordinates in static equilibrium %
pst = CalculateAllCoordinates(0.0,qst,reevSys);
```

Function: `CalculateAllCoordinates.m`
Inputs: Time instant `t`. Generalized coordinates `q`. Data structure `reevSys.qdep0`
Output: Total set of coordinates `p`

Line 43 - 53 is a post-process of the static equilibrium position. The axial force field along the two rope elements is computed and plotted.

```
% Calculation of axial force field along rope %
sa = 0:0.01:qst(8);
for i = 1:length(sa)
    Pa(i) = AxialLoad(pst(7+1:7+17),sa(i),reevSys.EA_wr(1),
       reevSys);
end
sb = qst(8):0.01:(reevSys.d + reevSys.l0);
for i = 1:length(sb)
    Pb(i) = AxialLoad(pst(7+17+1:7+17+17),sb(i),reevSys.EA_wr
       (2),reevSys);
end

% Plots axial force field %
figure(10);
plot(sa,0.001*Pa,'b',sb,0.001*Pb,'g');
title('Axial load on wire ropes. Initial static position')
xlabel('Arc-length along rope (m)');
ylabel('Axial load (KN)');
legend('Horizontal element a','Vertical element b');
```

Function: `AxialLoad.m`
Inputs: Nodal coordinates of the element `qi`. Arc-length coordinate `s`. Axial stiffness of the element `EA`. Data structure `reevSys.qdep0`
Output: Axial force `P`

PHASE 4. Simulation

Lines 65 - 78 perform the dynamic simulation. In line 66, the value of the independent velocities `vind` is set to zero. Depending on the value of the parameter `reevSys.ODE-method` different solvers for the equations of motion can be used.

```
% Initial value of independent velocities %
vind0 = zeros(size(reevSys.Ind));

if strcmp(reevSys.ODE_method,'ode15s')
    tspan = 0:0.001:5.0;
    options = odeset('MaxStep',0.001);
    [t y] = ode15s(@(t,y) EqMotReevingSystem(t,y,reevSys),
       tspan, [qind0' vind0']',options);
elseif strcmp(reevSys.ODE_method,'ode45')
     tspan = 0:0.001:5.0;
    options = odeset('MaxStep',0.001);
    [t y] = ode45(@(t,y) EqMotReevingSystem(t,y,reevSys),tspan
       , [qind0' vind0']',options);
elseif strcmp(reevSys.ODE_method,'RungeKutta4')
    [t y] = RungeKutta4(@EqMotReevingSystem,[0 5.0], [qind0'
       vind0']', 0.001, reevSys);
end
```

Function: `EqMotReevingSystem.m`
Inputs: Time instant `t`. State vector `y` including independent coordinates `qind` and velocities `vind`. Data structure `reevSys`
Output: Independent accelerations `qind`

PHASE 5. Post-process

In lines 84 - 105, the total set of coordinates `p` at all instants and the axial loads at end 1 of element a and at end 2 of element b are computed and plotted.

```
% Calculation of all coordinates all instants %
qdep = reevSys.qdep0;

p = zeros(reevSys.np,length(t));
% Axial load at end '1' of element 'a' %
P1 = zeros(length(t),1);
% Axial load at end '2' of element 'b' %
P2 = zeros(length(t),1);

for i = 1:length(t)
    [p(:,i), Cp, qdep] = Calculate_p_from_qind(t(i),y(i,1:
       length(reevSys.Ind))',qdep,reevSys);
    P1(i) = AxialLoad(p(7+1:7+17,i),p(7+16,i),reevSys.EA_wr(1)
       ,reevSys);
    P2(i) = AxialLoad(p(7+17+1:7+17+17,i),p(7+17+17,i),reevSys
       .EA_wr(2),reevSys);
end

% Plot time-history of axial loads %
figure(20);
plot(t,0.001*P1,'b',t,0.001*P2,'g')
title('Axial load on wire ropes')
xlabel('Time (s)');
ylabel('Axial load (KN)');
legend('End 1 of element a','End 2 of element b');
```

Function: `Calculate-p-from-qind.m`
Inputs: Time instant `t`. State vector `y` including independent coordinates `qind` and velocities `vind`. Initial guess of dependent coordinates `qdep0`. Data structure `reevSys`

Outputs: Total set of coordinates `p`. Jacobian of nonlinear contraints `Cp`. Dependent coordinates `qdep`

In lines 107 - 121, the total set of coordinates are re-sampled (to avoid too long animation) and the animation of the motion of the crane is performed.

```
% Animation of crane motion %
% Number of frames of the animation %
N = 100;

% Resamples the data %
t2 = t(1):(t(end)-t(1))/(N-1):t(end);
p2 = zeros(reevSys.np,N);
for i = 1:reevSys.np
    p2(i,:) = interp1(t,p(i,:),t2);
end

% Animates the motion of the system %
for i = 1:length(t2)
    Fot(i) = AnimateCrane(t2(i),p2(:,i),reevSys);
end
```

Function: `AnimateCrane.m`
Inputs: Time instant `t`. Total set of coordinates `p`. Data structure `reevSys`
Outputs: Video frame `Fot`

3 Installation and running

This code does not require any installation. To execute it in a Matlab environment, simply run **simulateSimpleCrane.m** in *src/simpleCrane/*. By running the code, the system solves the problem for the given parameters, plots simulation results and animates the motion of the crane.

References

[1] J. Escalona and N. Mohammadi, "Advances in the modeling and dynamic simulation of reeving systems using the arbitrary lagrangian–eulerian modal method.," *Nonlinear Dynamics*, vol. 108, no. 4, pp. 3985–4003, 2022.

[2] J. Escalona, "An arbitrary lagrangian–eulerian discretization method for modeling and simulation of reeving systems in multibody dynamics.," *Mechanism and Machine Theory*, vol. 112.