Solve DAE for cart-pole system

Based on this example from documentation:

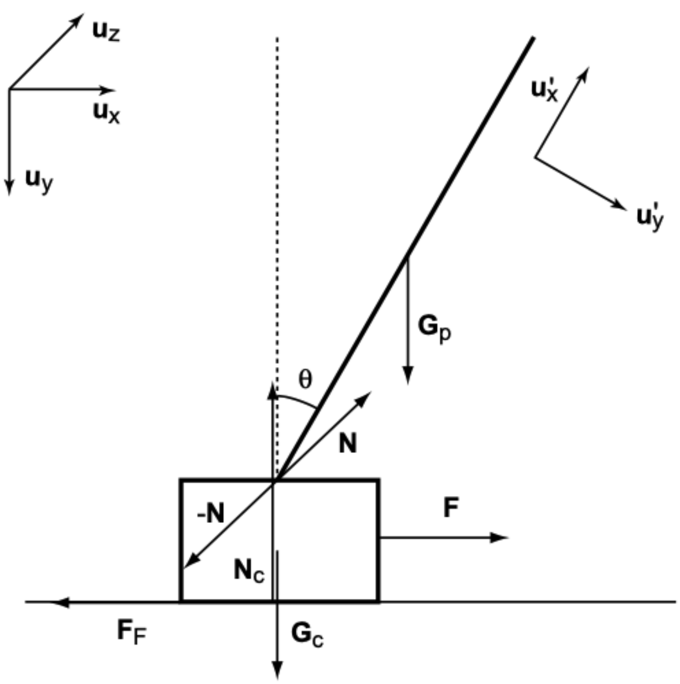
- https://www.mathworks.com/help/symbolic/solve-differential-algebraic-equations.html

```
clear variables

results_dir = 'results';
if ~isfolder(results_dir)
    mkdir(results_dir)
end
```

Equations and diagram below from:

- R. V. Florian, 2007, Correct equations for the dynamics of the cart-pole system.



State variables

- Horizontal position of cart (from left) $x(t)$
- Horizontal velocity of cart $\dot{x}(t)$
- Angle of pole (clockwise from vertical up position) $\theta(t)$
- Angular velocity of pole $\dot{\theta}(t)$

Other variables

- Downwards force on track from cart $N_c(t)$

- Frictional force exerted on cart by track $F_F(t)$
- Force exerted on cart by pole (and vice versa) $N(t)$

Potential input variables

- Horizontal diriving force on cart $F(t)$

Parameters

- Cart mass m_c
- Pole mass m_p
- Pendulum length l
- Acceleration due to gravity g
- Coefficient of friction for cart and track μ_c
- Coefficient of friction for pole and cart joint μ_p

System of DAEs

$$N_c = (m_c + m_p)g - m_p l\left(\ddot{\theta}\sin\theta + \dot{\theta}^2\cos\theta\right)$$

$$\ddot{\theta} = \frac{g\sin\theta + \cos\theta\left\{\dfrac{-F - m_p l\dot{\theta}^2\left(\sin\theta + \mu_c\,\mathrm{sgn}\left(N_c\dot{x}\right)\cos\theta\right)}{m_c + m_p} + \mu_c g\,\mathrm{sgn}\left(N_c\dot{x}\right)\right\} - \dfrac{\mu_p\dot{\theta}}{m_p l}}{l\left\{\dfrac{4}{3} - \dfrac{m_p\cos\theta}{m_c + m_p}\left(\cos\theta - \mu_c\,\mathrm{sgn}\left(N_c\dot{x}\right)\right)\right\}}$$

$$\ddot{x} = \frac{F + m_p l\left(\dot{\theta}^2\sin\theta - \ddot{\theta}\cos\theta\right) - \mu_c N_c\mathrm{sgn}\left(N_c\dot{x}\right)}{m_c + m_p}$$

First, define an autonomous system

i.e. `F(t) = 0`, `L(t) = l` (constant)

```
syms x(t) theta(t) Nc(t) F mc mp l g muc mup

eqn1 = 1000 * Nc == ...
    (mc + mp) * g - mp * l * (diff(theta(t), 2) * sin(theta(t)) + diff(theta(t))^2 * co
```

eqn1(t) =

$$1000\,\mathrm{Nc}(t) = g\,\left(\mathrm{mc} + \mathrm{mp}\right) - l\,\mathrm{mp}\,\left(\sin(\theta(t))\,\frac{\partial^2}{\partial t^2}\,\theta(t) + \cos(\theta(t))\,\left(\frac{\partial}{\partial t}\,\theta(t)\right)^2\right)$$

```
eqn2 = diff(theta(t), 2) == ...
    (g * sin(theta(t)) + cos(theta(t)) * ( ...
        (-F - mp * l * diff(theta(t))^2 * (sin(theta(t)) + muc * sign(Nc) * sign(diff(
     ) / (mc + mp) + muc * g * sign(Nc) * sign(diff(x(t)))) - mup * diff(theta(t)) / (
    / (l * (4/3 - mp * cos(theta(t)) / (mc + mp) * (cos(theta(t)) - muc * sign(Nc) * s
```

eqn2(t) =

$$\frac{\partial^2}{\partial t^2}\ \theta(t) = \frac{\cos(\theta(t)) \left(\frac{F + l\,\mathrm{mp}\ (\sin(\theta(t)) + \mathrm{muc}\ \mathrm{sign}(\mathrm{Nc}(t))\,\sigma_1 \cos(\theta(t)))\ \left(\frac{\partial}{\partial t}\ \theta(t)\right)^2}{\mathrm{mc} + \mathrm{mp}} - g\ \mathrm{muc}\ \mathrm{sign}(\mathrm{Nc}(\right.}{l\ \left(\frac{\mathrm{mp}\cos(\theta(t))\ (\cos(\theta(t)) - \mathrm{muc}\ \mathrm{sign}(\mathrm{Nc}(t))\,\sigma_1)}{\mathrm{mc} + \mathrm{mp}} - 1.3333\right.}$$

where

$$\sigma_1 = \mathrm{sign}\left(\frac{\partial}{\partial t}\ x(t)\right)$$

```
eqn3 = diff(x(t), 2) == ...
    (F + mp * l * (diff(theta(t))^2 * sin(theta(t)) - diff(theta(t), 2) * cos(theta(t)
     - muc * 1000 * Nc * sign(Nc) * sign(diff(x(t)))) ...
    / (mc + mp)
```

eqn3(t) =

$$\frac{\partial^2}{\partial t^2}\ x(t) = \frac{F + l\,\mathrm{mp}\ \left(\sin(\theta(t))\ \left(\frac{\partial}{\partial t}\ \theta(t)\right)^2 - \cos(\theta(t))\,\frac{\partial^2}{\partial t^2}\ \theta(t)\right) - 1000\ \mathrm{muc}\ \mathrm{sign}(\mathrm{Nc}(t))\ \mathrm{sign}\left(\frac{\partial}{\partial t}\ x(t)\right)\ \mathrm{N}}{\mathrm{mc} + \mathrm{mp}}$$

```
eqns = [eqn1 eqn2 eqn3];
vars = [x(t); theta(t); Nc(t)];
origVars = length(vars)
```

origVars = 3

Check Incidence of Variables

```
M = incidenceMatrix(eqns, vars)
```

M = 3×3

0	1	1
1	1	1
1	1	1

Reduce Differential Order

```
[eqns, vars] = reduceDifferentialOrder(eqns, vars)
```

eqns =

$$\left(\begin{array}{c} 1000\,\mathrm{Nc}(t) - g\,(\mathrm{mc}+\mathrm{mp}) + l\,\mathrm{mp}\,\left(\sin(\theta(t))\,\frac{\partial}{\partial t}\,\mathrm{Dthetat}(t) + \right. \\ \frac{\partial}{\partial t}\,\mathrm{Dthetat}(t) - \dfrac{\cos(\theta(t))\,\left(\dfrac{l\,\mathrm{mp}\,(\sin(\theta(t)) + \mathrm{muc}\,\mathrm{sign}(\mathrm{Dxt}(t))\,\mathrm{sign}(\mathrm{Nc}(t))\,\cos(\theta(t)))\,\mathrm{Dthetat}(t)^2 + F}{\mathrm{mc}+\mathrm{mp}}\right.}{l\,\left(\dfrac{\mathrm{mp}\cos(\theta(t))\,(\cos(\theta(t)) - \mathrm{muc}\,\mathrm{sign}(\mathrm{Dxt}(t)}{\mathrm{mc}+\mathrm{mp}}\right.} \\ \frac{\partial}{\partial t}\,\mathrm{Dxt}(t) + \dfrac{l\,\mathrm{mp}\,\left(\cos(\theta(t))\,\frac{\partial}{\partial t}\,\mathrm{Dthetat}(t) - \sin(\theta(t))\,\mathrm{Dthetat}(t)^2\right) - F + 10}{\mathrm{mc}+\mathrm{mp}} \\ \mathrm{Dxt}(t) - \frac{\partial}{\partial t}\,x(t) \\ \mathrm{Dthetat}(t) - \frac{\partial}{\partial t}\,\theta(t) \end{array}\right.$$

vars =

$$\begin{pmatrix} x(t) \\ \theta(t) \\ \mathrm{Nc}(t) \\ \mathrm{Dxt}(t) \\ \mathrm{Dthetat}(t) \end{pmatrix}$$

Check Differential Index of System

```
if ~isLowIndexDAE(eqns, vars)
    disp("Reducing Differential Index...")
    % Reduce Differential Index with reduceDAEIndex
    [DAEs, DAEvars] = reduceDAEIndex(eqns, vars)
    % Eliminate redundant equations and variables
    [DAEs, DAEvars] = reduceRedundancies(DAEs, DAEvars)
    % Check the differential index of the new system
    assert(isLowIndexDAE(DAEs, DAEvars))
else
    DAEs = eqns;
    DAEvars = vars;
end
```

Convert DAE system to MATLAB function

```
pDAEs = symvar(DAEs);
pDAEvars = symvar(DAEvars);
extraParams = setdiff(pDAEs, pDAEvars)
```

extraParams = $\begin{pmatrix} F & g & l & \mathrm{mc} & \mathrm{mp} & \mathrm{muc} & \mathrm{mup} \end{pmatrix}$

Create the function handle.

```
%f = daeFunction(DAEs, DAEvars, F, g, l, mc, mp, muc, mup);
```

To save the DAE equations as a function script use this option

```
filename = 'cartpoleDAEFunction.m';
```

```
f = daeFunction(DAEs, DAEvars, F, g, l, mc, mp, muc, mup, 'File', filename);
```

Set parameter values

```
F = 10;
g = 10;
l = 2;
mc = 5;
mp = 1;
muc = 0.2;
mup = 0.2;

% Parameter values as struct
params = struct();
params.F = F;
params.g = g;
params.l = l;
params.mc = mc;
params.mp = mp;
params.muc = muc;
params.mup = mup;
```

Create function for ode15i

```
F_DAE = @(t, Y, YP) f(t, Y, YP, F, g, l, mc, mp, muc, mup);
```

Find initial condition

```
DAEvars
```

DAEvars =

$$\begin{pmatrix} x(t) \\ \theta(t) \\ \mathrm{Nc}(t) \\ \mathrm{Dxt}(t) \\ \mathrm{Dthetat}(t) \end{pmatrix}$$

Note that `Dxt(t)`, `Dthetat(t)`, ... etc. are the first derivatives of `x(t)` ... etc.

Provide an estimate of the initial condition

```
% Variables
% 1 degrees = 0.0172
y0est = [0; pi/12; 0.001*9.81*5; 0; 0];
% Their derivatives
yp0est = [0; 0; 0; 0; 0];
```

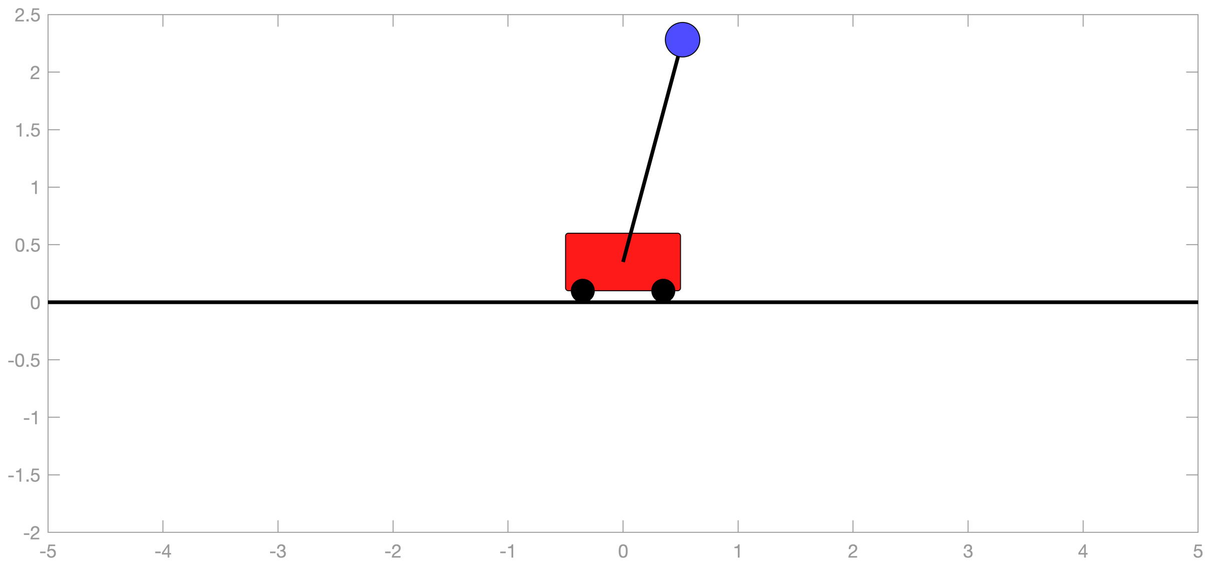
Set tolerances and do numerical search.

```
opt = odeset('RelTol', 10.0^(-7), 'AbsTol', 10.0^(-7));
FIXED_Y0 = [1 1 0 1 1]';
FIXED_YP0 = [0 0 0 0 0]';
```

```
[y0, yp0] = decic(F_DAE, 0, y0est, FIXED_Y0, yp0est, FIXED_YP0, opt)
```

```
y0 = 5×1
         0
    0.2618
    0.0598
         0
         0
yp0 = 5×1
         0
         0
         0
    1.5329
    0.4153
```

```
figure(1); clf
draw_cartpole(y0([1 3 2 4]),params)
```

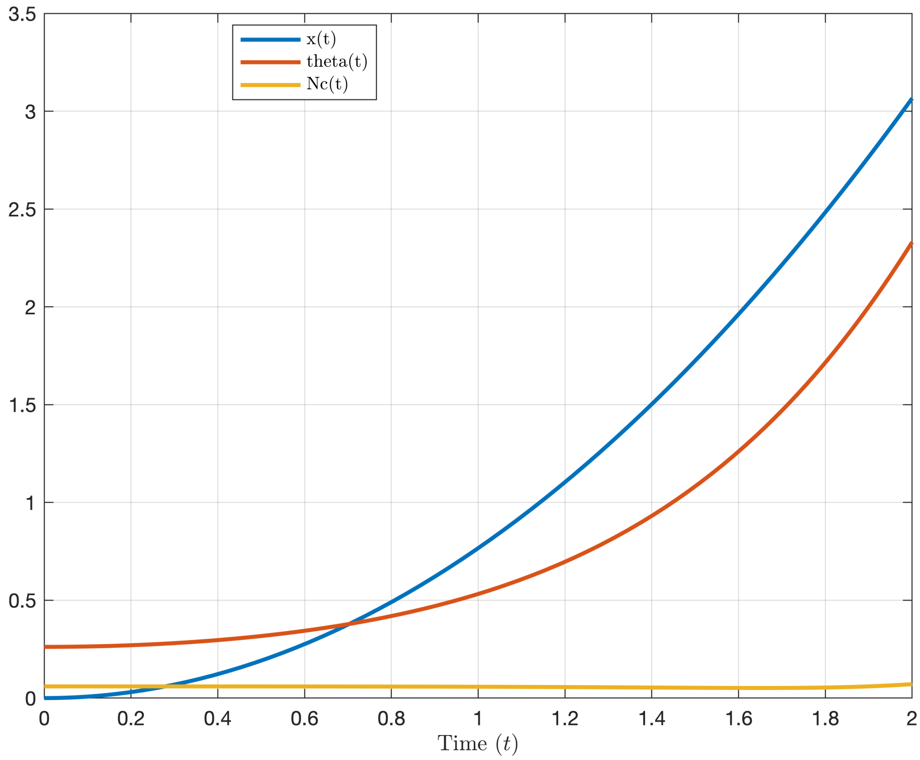


Solve DAEs Using ode15i

```
[tSol, ySol] = ode15i(F_DAE, [0 2], y0, yp0, opt);
```

Plot solution

```
figure(2); clf
plot(tSol, ySol(:, 1:origVars), 'LineWidth', 2)
xlabel("Time ($t$)", 'Interpreter', 'latex')
labels = arrayfun(@(i) char(DAEvars(i)), 1:origVars, 'UniformOutput', false);
legend(labels, 'Location', 'Best', 'Interpreter', 'latex')
grid on
```



Save results to file

```
labels = arrayfun(@(i) char(DAEvars(i)), 1:numel(DAEvars), 'UniformOutput', false);
sim_results = array2table([tSol ySol], 'VariableNames', [{'t'} labels]);
filename = "cartpole_benchmark_sim.csv";
writetable(sim_results, fullfile(results_dir, filename))
```

Solve using function script file

Solve the equations using the function script created by `daeFunction` above and check results are identical.

```
% Solve DAEs
F_DAE2 = @(t, Y, YP) cartpoleDAEFunction(t, Y, YP, F, g, l, mc, mp, muc, mup);
[tSol2, ySol2] = ode15i(F_DAE2, [0 2], y0, yp0, opt);
assert(isequal(tSol, tSol2))
assert(max(ySol - ySol2, [], [1 2]) < 1e-14)
```

Solve using the function in cartpole_DAEs.m

```
% Solve DAEs
F_DAE2 = @(t, Y, YP) cartpole_DAEs(t, Y, YP, params);
[tSol2, ySol2] = ode15i(F_DAE2, [0 2], y0, yp0, opt);
assert(isequal(tSol, tSol2))
assert(max(ySol - ySol2, [], [1 2]) < 1e-14)
```