Robotics HW #5 Aly Khater

1) RP arm

$$^{C_1}I_1 = 0; \quad ^{C_2}I_2 = 0 \qquad ^1P_{C_1} = \begin{pmatrix} 0 \\ L_1 \\ 0 \end{pmatrix} \qquad ^2P_{C_2} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

$$^3f_3 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}; \quad ^3n_3 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \qquad ^0v_0 = {}^0w_0 = {}^0\dot{w}_0 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \qquad ^0\dot{v}_0 = \begin{pmatrix} 0 \\ g \\ 0 \end{pmatrix}$$

$$^0_1R = \begin{pmatrix} c\theta & -s\theta & 0 \\ s\theta & c\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}; \quad ^1_0R = \begin{pmatrix} c_1 & s_1 & 0 \\ -s_1 & c_1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad ^0P_1 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

$$^1_2R = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix}; \quad ^2_1R = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix}; \quad ^1P_2 = \begin{pmatrix} 0 \\ d_2 \\ 0 \end{pmatrix}$$

Outward Link 1 Revolute

$$^1w_1 = {}^1_0R\,\cancel{^0w_0}^{\,0} + \dot{\theta}_1\hat{z}_1 \implies {}^1w_1 = \begin{pmatrix} 0 \\ 0 \\ \dot{\theta}_1 \end{pmatrix}$$

$$^1\dot{w}_1 = {}^1_0R\,\cancel{^0\dot{w}_0}^{\,0} + {}^1_0R\,\cancel{^0w_0}^{\,0} \times \begin{pmatrix} 0 \\ 0 \\ \dot{\theta}_1 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \ddot{\theta}_1 \end{pmatrix} \implies {}^1\dot{w}_1 = \begin{pmatrix} 0 \\ 0 \\ \ddot{\theta}_1 \end{pmatrix}$$

$$^1\dot{v}_1 = {}^1_0R\left(\cancel{^0\dot{w}_0}^{\,0} \times {}^0P_1 + \cancel{^0w_0}^{\,0} \times ({}^0w_0 \times {}^0P_1) + {}^0\dot{v}_0\right) = {}^1_0R\begin{pmatrix} 0 \\ g \\ 0 \end{pmatrix} = \begin{pmatrix} gs_1 \\ gc_1 \\ 0 \end{pmatrix}$$

$$^1\dot{v}_{C_1} = \underbrace{^1\dot{w}_1 \times {}^1P_{C_1}} + {}^1w_1 \times ({}^1w_1 \times {}^1P_{C_1}) + {}^1\dot{v}_1$$

$$\begin{pmatrix} 0 \\ 0 \\ \ddot{\theta}_1 \end{pmatrix} \times \begin{pmatrix} 0 \\ 0 \\ L_1 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \dot{\theta}_1 \end{pmatrix} \times \begin{pmatrix} 0 \\ 0 \\ \dot{\theta}_1 \end{pmatrix} \times \begin{pmatrix} 0 \\ L_1 \\ 0 \end{pmatrix} + \begin{pmatrix} gs_1 \\ gc_1 \\ 0 \end{pmatrix} = \begin{pmatrix} -L_1\ddot{\theta}_1 + gs_1 \\ -L_1\dot{\theta}_1^2 + gc_1 \\ 0 \end{pmatrix}$$

$$^1F_1 = m_1\,{}^1\dot{v}_{C_1} = \begin{pmatrix} -m_1L_1\ddot{\theta}_1 + m_1gs_1 \\ -m_1L_1\dot{\theta}_1^2 + m_1gc_1 \\ 0 \end{pmatrix}$$

$$^1N_1 = {}^{C_1}I_1\,{}^1\dot{w}_1 + {}^1w_1 \times {}^{C_1}I_1\,{}^1w_1 = 0 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

Inward Link 2

$$^2f_2 = {}^2_3R\, \cancel{^3f_3}^{0} + {}^2F_2 = {}^2F_2$$

$$^2n_2 = \cancel{^2N_2}^{0} + {}^2_3R\, \cancel{^3n_3}^{0} + \cancel{^2P_{c_2}}^{0} \times {}^2F_2 + {}^2P_3 \times {}^2_3R\, \cancel{^3f_3}^{0} = 0$$

Inward Link 1

$$^1f_1 = {}^1_2R\, {}^2f_2 + {}^1F_1 = \begin{pmatrix} -m_2\ddot{\theta}_1 d_2 + m_2 g s_1 - 2m_2\dot{\theta}_1\dot{d}_2 - m_1 L_1\ddot{\theta}_1 + m_1 g s_1 \\ -m_2\dot{\theta}_1^2 d_2 + m_2 g c_1 + m_2\ddot{d}_2 - m_1 L_1\dot{\theta}_1^2 + m_1 g c_1 \\ 0 \end{pmatrix}$$

$$^1n_1 = \cancel{^1N_1}^{0} + {}^1_2R\, \cancel{^2n_2}^{0} + {}^1P_{c_1} \times {}^1F_1 + {}^1P_2 \times {}^1_2R\, {}^2f_2$$

$$= \begin{pmatrix} 0 \\ L_1 \\ 0 \end{pmatrix} \times \begin{pmatrix} -m_1L_1\ddot{\theta}_1 + m_1 g s_1 \\ -m_1L_1\dot{\theta}_1^2 + m_1 g c_1 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ d_2 \\ 0 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} -m_2\ddot{\theta}_1 d_2 + m_2 g s_1 - 2m_2\dot{\theta}_1\dot{d}_2 \\ 0 \\ -m_2\dot{\theta}_1^2 d_2 + m_2 g c_1 + m_2\ddot{d}_2 \end{pmatrix}$$

$$= \begin{pmatrix} 0 \\ 0 \\ m_1L_1^2\ddot{\theta}_1 - m_1 L g s_1 \end{pmatrix} + \begin{pmatrix} 0 \\ d_2 \\ 0 \end{pmatrix} \times \begin{pmatrix} -m_2\ddot{\theta}_1 d_2 + m_2 g s_1 - 2m_2\dot{\theta}_1\dot{d}_2 \\ -m_2\dot{\theta}_1^2 d_2 + m_2 g c_1 + m_2\ddot{d}_2 \\ 0 \end{pmatrix}$$

$$= \begin{pmatrix} 0 \\ 0 \\ m_1L_1^2\ddot{\theta}_1 - m_1 L g s_1 + m_2 d_2^2\ddot{\theta}_1 - m_2 g d_2 s_1 + 2m_2 d_2\dot{\theta}_1\dot{d}_2 \end{pmatrix}$$

$$\tau_2 = (f_2)_2 = -m_2 d_2 \dot{\theta}_1^2 + m_2 g c_1 + m_2 \ddot{d}_2$$

$$\tau_1 = (n_1)_z = m_1 L_1^2 \ddot{\theta}_1 + m_1 L g s_1 + m_2 d_2^2 \ddot{\theta}_1 + m_2 g d_2 s_1 + 2 m_2 d_2 \dot{\theta}_1 \dot{d}_2$$

$$\tau = M(\theta)\ddot{\theta} + V(\theta,\dot{\theta}) + G(\theta)$$

$$\tau = \begin{pmatrix} \tau_1 \\ \tau_2 \end{pmatrix} = \underbrace{\begin{bmatrix} m_1 L_1^2 + m_2 d_2^2 & 0 \\ 0 & m_2 \end{bmatrix}}_{M(\theta)} \begin{pmatrix} \ddot{\theta}_1 \\ \ddot{d}_2 \end{pmatrix}$$

$$+ \underbrace{\begin{bmatrix} 2 m_2 d_2 \dot{\theta}_1 \dot{d}_2 \\ -m_2 d_2 \dot{\theta}_1^2 \end{bmatrix}}_{V(\theta,\dot{\theta})} + \underbrace{\begin{pmatrix} -m_1 L g s_1 - m_2 g d_2 s_1 \\ m_2 g c_1 \end{pmatrix}}_{G(\theta)}$$

(2) $^2v_2 = {}^2_1R({}^1v_1 + {}^1\omega_1 \times {}^1P_2) + \dot{d}_2 \hat{z}_2$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix}\begin{pmatrix} -d_2\dot{\theta}_1 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \dot{d}_2 \end{pmatrix}$$

$M_x(\theta) = J^{-T}(\theta)M(\theta)J^{-1}(\theta)$

$V_x(\theta,\dot{\theta}) = J^{-T}(\theta)(V(\theta,\dot{\theta}) - M(\theta)J^{-1}(\theta)\dot{J}(\theta)\dot{\theta})$

$G_x = J^{-T}(\theta)G(\theta)$

$$^2v_2 = \begin{pmatrix} -d_2\dot{\theta}_1 \\ 0 \\ \dot{d}_2 \end{pmatrix} = \begin{pmatrix} -d_2 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} \dot{\theta}_1 \\ \dot{d}_2 \end{pmatrix}$$

$$^2J(\theta) = \begin{pmatrix} -d_2 & 0 \\ 0 & 1 \end{pmatrix}; \quad {}^2J^{-1}(\theta) = \begin{pmatrix} \frac{1}{-d_2} & 0 \\ 0 & 1 \end{pmatrix} \quad {}^2\dot{J}(\theta) = \begin{pmatrix} -\dot{d}_2 & 0 \\ 0 & 0 \end{pmatrix}$$

$$M_x(\theta) = \begin{pmatrix} -\frac{1}{d_2} & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} m_1L_1^2 + m_2d_2^2 & 0 \\ 0 & m_2 \end{pmatrix}\begin{pmatrix} -1/d_2 & 0 \\ 0 & 1 \end{pmatrix}$$

$$M_x(\theta) = \begin{pmatrix} M_2 + \frac{m_1L_1^2}{d_2^2} & 0 \\ 0 & M_2 \end{pmatrix}$$

$$V_x(\theta,\dot{\theta}) = J^{-T}V(\theta,\dot{\theta}) - M_x(\theta)\dot{J}(\dot{\theta})$$

$$= \begin{pmatrix} -1/d_2 & 0 \\ 0 & 1 \end{pmatrix}\begin{bmatrix} 2M_2d_2\dot{d}_2\dot{\theta}_1 \\ -M_2d_2\dot{\theta}_1^2 \end{bmatrix} - M_x(\theta)\begin{pmatrix} -\dot{\theta}_1\dot{d}_2 \\ 0 \end{pmatrix}$$

$$= \begin{bmatrix} -2M_2\dot{d}_2\dot{\theta}_1 \\ -M_2d_2\dot{\theta}_1^2 \end{bmatrix} + \begin{bmatrix} \dot{\theta}_1\dot{d}_2\left(M_2 + \frac{m_1L_1^2}{d_2^2}\right) \\ 0 \end{bmatrix}$$

$$V_x(\theta,\dot{\theta}) = \begin{bmatrix} \left(\frac{m_1L_1^2}{d_2^2}\right)\dot{\theta}_1\dot{d}_2 - M_2\dot{\theta}_1\dot{d}_2 \\ -M_2\dot{\theta}_1^2 d_2 \end{bmatrix}$$

$$G_x(\theta) = J^{-T}G(\theta) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

wrong

3) $\tau_1 = m_1(\cancel{d_1} + d_2)\ddot{\theta}_1 + m_2 d_2^2 \ddot{\theta}_1 + 2 m_2 d_2 \dot{d}_2 \dot{\theta}_1$
$+ g\cos(\theta_1)[m_1(\cancel{d_1} + \cancel{d_2}\dot{\theta}_1) + m_2(d_2 + \cancel{d_2})]$

$\tau_2 = m_1 \cancel{\ddot{d}_2}\dot{\theta}_1 + m_2 \ddot{d}_2 - m_1 \cancel{d_1} \dot{d}_2 - m_2 d_2 \dot{\theta}^2 + m_2(d_2 + 1) g \sin\theta_1$

~~4) a) Input - position, velocity, acceleration, external force, gravity~~

~~Output - joint torque required to achieve inputted position, velocity, and acceleration.~~

~~SEE MATLAB~~

4) SEE MATLAB

(6) a) 3-DOF x, y, z

b) 5-DOF x, y, z, roll, pitch

c) 3-DOF

(7) a) i) Articulated

ii) SCARA

b) T T F T

c) Grubler's formula is for non-SCM robots (planar robots)

d) No added friction or flexibility
too low torque and too high speed, Mechanical Advantage

e) reduces speed and amplifies torque

f) Allows remote installation of actuators.

h) Joint compliance and material properties

i) Differentiation of position data, Allows flexibility in design, but can be inaccurate

8) $g_1 = 100, \quad g_2 = 5$ RP

a) $Q = [\pi/2, 0.1]$

$\alpha_1 = g_1 Q_1 = 100(\pi/2)$

$\alpha_1 = 50\pi$

$\alpha_2 = g_2 Q_2 = 5(0.1)$

$\alpha_2 = .05$

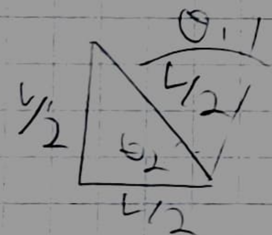
$$A = \begin{pmatrix} \theta_1 \\ d_2 \end{pmatrix} = \begin{pmatrix} 50\pi \\ 0.5 \end{pmatrix}$$

b) $X = \begin{pmatrix} 0 \\ 2L \end{pmatrix} \Rightarrow Q = \begin{pmatrix} \pi/2 \\ 2L \end{pmatrix}$

$\alpha_1 = 50\pi$

$\alpha_2 = g_2 Q_2 = 5(2L) = 10L$

$$A = \begin{pmatrix} 50\pi \\ 10L \end{pmatrix}$$

c) 


equilateral triangle

$\theta_1 = 30°$

$\theta_2 = 60°$

```
%Problem 4
clc;
close all;
```

```
%Problem 4a
help rne.m
```

```
 rne    Compute inverse dynamics via recursive Newton-Euler formulation

     TAU = rne(ROBOT, Q, QD, QDD)
     TAU = rne(ROBOT, [Q QD QDD])

     Returns the joint torque required to achieve the specified joint position,
     velocity and acceleration state.

     Gravity vector is an attribute of the robot object but this may be
     overriden by providing a gravity acceleration    vector [gx gy gz].

     TAU = rne(ROBOT, Q, QD, QDD, GRAV)
     TAU = rne(ROBOT, [Q QD QDD], GRAV)

     An external force/moment acting on the end of the manipulator may also be
     specified by a 6-element vector [Fx Fy Fz Mx My Mz].

     TAU = rne(ROBOT, Q, QD, QDD, GRAV, FEXT)
     TAU = rne(ROBOT, [Q QD QDD], GRAV, FEXT)

     where    Q, QD and QDD are row vectors of the manipulator state; pos, vel, and accel.

     The torque computed also contains a contribution due to armature
     inertia.

     See also ROBOT, FROBOT, accel, gravload, inertia.

     Should be a MEX file.

    Other uses of rne
```

```
%input = position, velocity, acceleration
%        gravity, external force
%output = joint torque
```

```
%Q = [0 0 0]
TAU1 = rne(SCURonelink,0,0,0)
```

```
TAU1 = 0
```

```
%Q = [0 0 1]
TAU2 = rne(SCURonelink,0,0,1)
```

```
TAU2 = 1
```

```
%Makes sense as only acceleration contributes
%to the torque
```

```
%Q = [pi/2 0 1]
TAU3 = rne(SCURonelink,pi/2,0,1)
```

```
TAU3 = 1
```

```
%Makes sense as acceleration is still only
%contributing to the torque
```

```
%Q = [0 1 1]
TAU4 = rne(SCURonelink,0,1,1)
```

```
TAU4 = 2
```

```
%Makes sense since both acceleration and velocity
%affect the torque
```

```
%Q = [0 0 1] link length = 2
TAU5 = rne(SCURonelink,0,0,1)
```

```
TAU5 = 4
```

```
%Makes sense since there is a squared relationship
%between length and inertia.
```

```
%Problem 4c
%1) Since Q is a row vector with degree, with 50
%steps, q,qd,qdd would be 3x50 matrices.
%Q entries would ultimately be nxm; where
%n is number of dof and m is number of time steps
```

```
%2)
Q = [[0 0],[0 0],[0 1]];
TAUc2 = rne(SCURRtwolink,Q)
```

```
TAUc2 = 1×2
     2     1
```

```
%We have unit length, friction, and mass.
%By craigs equations:
%T1 = 1+1+0-0-0+0+0 = 2
%T2 = 0+0+0+1 = 1
%Values are the same
```

```
%3)
Q = [[0 pi/2],[0 0],[0 1]];
TAUc3 = rne(SCURRtwolink,Q)
```

```
TAUc3 = 1×2
     1     1
```

```
%No, it is at at a zero pose.
```

```
%4)
Q = [[0 0], [0 0], [0 0]];
TAUc4 = rne(SCURRtwolink,[0 0],[0 0],[1 1],0)
```

```
TAUc4 = 1×2
     3     1
```

```
%Only the Mass matrix is left.
```

```
%Problem 5
clc;
close all;
```

```
help accel.m
```

```
 accel Compute manipulator forward dynamics
 
     QDD = accel(ROBOT, Q, QD, TORQUE)
     QDD = accel(ROBOT, [Q QD TORQUE])
 
  Returns a vector of joint accelerations that result from applying the
  actuator TORQUE to the manipulator ROBOT in state Q and QD.
 
  Uses the method 1 of Walker and Orin to compute the forward dynamics.
  This form is useful for simulation of manipulator dynamics, in
  conjunction with a numerical integration function.
 
  See also: rne, ROBOT, ode45.
```

```
%a
%i)
%Input - Takes position and velocity.
%Output - Outputs joint acceleration.

%ii)
%It's so you can get your mass, gravity and coriolis torque
```

```
help fdyn.m
```

```
 fdyn Integrate forward dynamics
 
     [T Q QD] = fdyn(ROBOT, T0, T1)
     [T Q QD] = fdyn(ROBOT, T0, T1, TORQFUN)
     [T Q QD] = fdyn(ROBOT, T0, T1, TORQFUN, Q0, QD0)
     [T Q QD] = fdyn(ROBOT, T0, T1, TORQFUN, Q0, QD0, ARG1, ARG2, ...)
 
  Integrates the dynamics of manipulator ROBOT dynamics over the time
  interval T0 to T1 and returns vectors of joint position and velocity.
  ROBOT is a robot object and describes the manipulator dynamics and
  kinematics, and Q is an n element vector of joint state.
 
  A control torque may be specified by a user specified function
 
      TAU = TORQFUN(T, Q, QD, ARG1, ARG2, ...)
 
  where Q and QD are the manipulator joint coordinate and velocity state
  respectively], and T is the current time. Optional arguments passed to fdyn
  will be passed through to the user function.
 
  If TORQFUN is not specified, or is given as 0,  then zero torque is
  applied to the manipulator joints.
 
  See also: accel, nofriction, rne, ROBOT, ode45.
```

```
help ode45.m
```

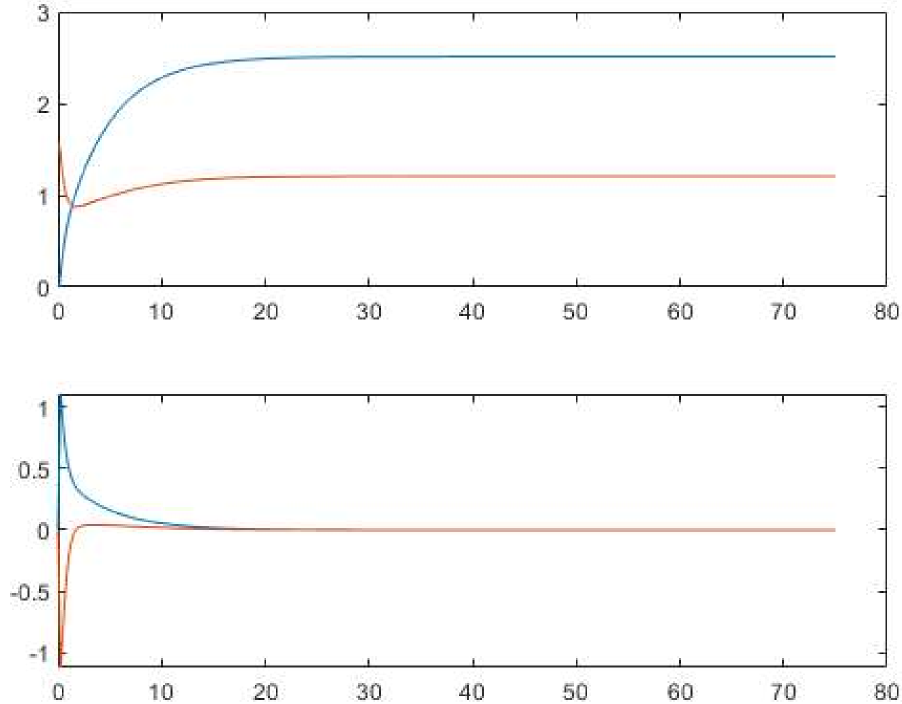
```
ode45  Solve non-stiff differential equations, medium order method.
   [TOUT,YOUT] = ode45(ODEFUN,TSPAN,Y0) integrates the system of
   differential equations y' = f(t,y) from time TSPAN(1) to TSPAN(end)
   with initial conditions Y0. Each row in the solution array YOUT
   corresponds to a time in the column vector TOUT.
     * ODEFUN is a function handle. For a scalar T and a vector Y,
       ODEFUN(T,Y) must return a column vector corresponding to f(t,y).
     * TSPAN is a two-element vector [T0 TFINAL] or a vector with
       several time points [T0 T1 ... TFINAL]. If you specify more than
       two time points, ode45 returns interpolated solutions at the
       requested times.
     * YO is a column vector of initial conditions, one for each equation.

   [TOUT,YOUT] = ode45(ODEFUN,TSPAN,Y0,OPTIONS) specifies integration
   option values in the fields of a structure, OPTIONS. Create the
   options structure with odeset.

   [TOUT,YOUT,TE,YE,IE] = ode45(ODEFUN,TSPAN,Y0,OPTIONS) produces
   additional outputs for events. An event occurs when a specified function
   of T and Y is equal to zero. See ODE Event Location for details.

   SOL = ode45(...) returns a solution structure instead of numeric
   vectors. Use SOL as an input to DEVAL to evaluate the solution at
   specific points. Use it as an input to ODEXTEND to extend the
   integration interval.

   ode45 can solve problems M(t,y)*y' = f(t,y) with mass matrix M that is
   nonsingular. Use ODESET to set the 'Mass' property to a function handle
   or the value of the mass matrix. ODE15S and ODE23T can solve problems
   with singular mass matrices.

   ODE23, ode45, ODE78, and ODE89 are all single-step solvers that use
   explicit Runge-Kutta formulas of different orders to estimate the error
   in each step.
     * ode45 is for general use.
     * ODE23 is useful for moderately stiff problems.
     * ODE78 and ODE89 may be more efficient than ode45 on non-stiff problems
       that are smooth except possibly for a few isolated discontinuities.
     * ODE89 may be more efficient than ODE78 on very smooth problems, when
       integrating over long time intervals, or when tolerances are tight.

   Example
         [t,y]=ode45(@vdp1,[0 20],[2 0]);
         plot(t,y(:,1));
     solves the system y' = vdp1(t,y), using the default relative error
     tolerance 1e-3 and the default absolute tolerance of 1e-6 for each
     component, and plots the first component of the solution.

   Class support for inputs TSPAN, Y0, and the result of ODEFUN(T,Y):
     float: double, single

   See also ode23, ode78, ode89, ode113, ode15s, ode23s, ode23t, ode23tb,
            ode15i, odeset, odeplot, odephas2, odephas3, odeprint, deval,
            odeexamples, function_handle.

   Documentation for ode45
```

```
%b
%Takes the time interval from fdyn and the robot parameters,
%solves for differential equations, to output an interpolation
```

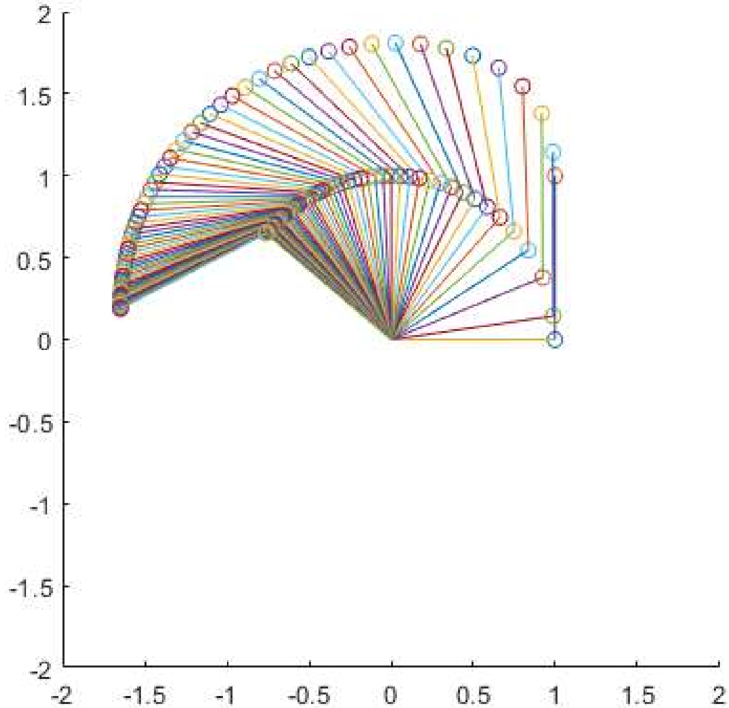
```
%of the solutions at requested times.
```

```
```

```
%Forward Dynamics Example

clear all
duration=75;                           %set duration of simulation

%Declare manipulator to be simulated by un-commenting the appropriate line
%RONELINK;                             %SCURonelink.m: Example 1-link R manipulator
SCURRTWOLINK                           %SCURRtwolink.m: Example 2-link RR manipulator

%Compute Forward Dynamics for the selected manipulator using designated torque profile
[T Q QD] = fdyn(SCURRtwolink, 0, duration, 'SCURRtwolinkTAU', [0;pi/2],[0;0]);

%Plotting Options
%   flag = 0:   plot joint position and velocity
%   flag = 1:   plot overhead view (interactive or batch - comment pause statement)

flag=0    ;

if flag==0                           %plot joint position and velocity
    subplot(2,1,1), plot(T, Q)
    subplot(2,1,2), plot(T, QD)

else if flag == 1                    %plot overhead view
    t=0:.05:duration;
    q=interp1( T, Q, t');
    %L1=1; X1=[L1*cos(q) L1*sin(q)]; %define link endpoints for onelink
    L1=1;L2=1;
    X1=[L1*cos(q(:,1)) L1*sin(q(:,1))];
    X2=[L1*cos(q(:,1))+L2*cos(q(:,1)+q(:,2)) L1*sin(q(:,1))+L2*sin(q(:,1)+q(:,2))];

    axis('square'); axis([-2 2 -2 2]); axis manual; hold on;
    for z=1:5:length(T)              %%%%% CHANGE THE STEP VALUE FOR MORE/LESS GRAPHICAL INTERPOLATION IN THE PLOT
        plot(X1(z,1),X1(z,2), 'o')  %plot endpoint and link line
        plot(X2(z,1),X2(z,2), 'o')
        plot([0;X1(z,1)],[0;X1(z,2)])
        plot([X1(z,1),X2(z,1)],[X1(z,2),X2(z,2)])
        pause                        %comment this line to force a non-interactive plot
    end

else
    plot(T, Q)


end
end
```


3
2
1
0
0 10 20 30 40 50 60 70 80
1
0.5
0
-0.5
-1
0 10 20 30 40 50 60 70 80

```matlab
%Forward Dynamics Example

clear all
duration=75;                         %set duration of simulation

%Declare manipulator to be simulated by un-commenting the appropriate line
%RONELINK;                           %SCURonelink.m: Example 1-link R manipulator
SCURRTWOLINK                         %SCURRtwolink.m: Example 2-link RR manipulator

%Compute Forward Dynamics for the selected manipulator using designated torque profile
[T Q QD] = fdyn(SCURRtwolink, 0, duration, 'SCURRtwolinkTAU', [0;pi/2],[0;0]);

%Plotting Options
%   flag = 0:   plot joint position and velocity
%   flag = 1:   plot overhead view (interactive or batch - comment pause statement)

flag=1    ;

if flag==0                          %plot joint position and velocity
    subplot(2,1,1), plot(T, Q)
    subplot(2,1,2), plot(T, QD)

else if flag == 1                   %plot overhead view
    t=0:.05:duration;
    q=interp1( T, Q, t');
    %L1=1; X1=[L1*cos(q) L1*sin(q)]; %define link endpoints for onelink
    L1=1;L2=1;
    X1=[L1*cos(q(:,1)) L1*sin(q(:,1))];
    X2=[L1*cos(q(:,1))+L2*cos(q(:,1)+q(:,2)) L1*sin(q(:,1))+L2*sin(q(:,1)+q(:,2))];

    axis('square'); axis([-2 2 -2 2]); axis manual; hold on;
    for z=1:5:length(T)             %%%%% CHANGE THE STEP VALUE FOR MORE/LESS GRAPHICAL INTERPOLATION IN THE PLOT
        plot(X1(z,1),X1(z,2), 'o')  %plot endpoint and link line
        plot(X2(z,1),X2(z,2), 'o')
        plot([0;X1(z,1)],[0;X1(z,2)])
        plot([X1(z,1),X2(z,1)],[X1(z,2),X2(z,2)])
        pause                       %comment this line to force a non-interactive plot
    end

else
    plot(T, Q)


end
end
```




*Published with MATLAB® R2023b*