HW 8, matlab solution

Nathan Barton

October 27, 1999

1 Problem 4

The `matlab` files for the solution are given below. First is the output from the driver file, followed by the driver file itself and then the `m`-files used in solving the problem. The `m`-file `linkage_ag.m` does most of the work that is new in this part of the problem. The `m`-files are also available for download from the web page. The plot is included at the end.

All angles are measured counterclockwise from the horizontal axis, and the `matlab` functions used in the solution take advantage of the systematic way in which the data structure for the problem has been created. Much of the code in this solution was given in the solution to the previous part of the fan problem.

Note that in the solution `m`-file `linkage_ag.m`, the acceleration of the center of mass of the fan (point E) is found by marching along the links starting at point A, which is fixed in space. Many of you recognized that it was easier to recognize that point D is fixed and on the same rigid body as point E. Doing it that way is just fine, but you might want to look at how it is done in the solution, as that method is more generally useful when you have links which you are not assuming to be massless.

Also note that I have plotted the angular acceleration of the third link (connecting points C, D, and E). This was not required, but might be helpful for those of you trying to find where you might have gone wrong – if you aren't getting this angular acceleration right, you aren't going to get the linear acceleration of the point E correct either.

1.1 output from driver file

```
>> fan_driver

th_init =

    0.6435
    0.6435
   -1.5708

expect magnitudes to occur at same time, because
fan is pinned to ground at D and the linear acceleration
of the center of mass is a multiple of the angular
acceleration.



approximate max magnitudes of linear and angular
  accelerations are, respectively:
  3.81 m/s^2
 21.96 rad/s^2

both maxima occur roughly   0.96 seconds into the cycle
```

1.2 Driver file

fan_driver.m

```
% some of the values that are set in this file are not used until later
% parts of the problem

global r r2 rg lthg;
global iwk iwu1 iwu2 ium;
global w_data;
global gc g_by_gc;
global m ai;

% figure counter
ifig = 0;

% some of geometry of linkage
%
% lengths of bars in linkage (meters)
r = 0.01*[10.0 40.0 30.0 40.0]';
r2 = r*r';
% angle (relative to local coordinates on link) to the center of mass;
% see m-file linkage_ag.m
lthg = [0.0 0.0 pi/6]';
% distance to centers of mass from starting point of link
rg = [r(1)*0.5 r(2)*0.5 r(3)/cos(lthg(3))]';

% masses, moments of inertia
m = [0.0 0.0 10.0]'; % in kg
% moment of inertia for uniform distribution of mass over links
ai = zeros(3,1);
ai(1) = m(1)*r2(1,1)/12.0;
ai(2) = m(2)*r2(2,2)/12.0;
diam_fan = 1.0;
ai(3) = m(3)*diam_fan*diam_fan/16.0;

% gravitation
g       = [0.0 9.8]'; % direction 2 is down, in m/s^2
gc      = 1.0; % metric, trust mass to be in kg
g_by_gc = g/gc;

th1_init = atan2(3,4);
th_init = fan_angles(r,r2,th1_init)

% data for initial angular velocities
```

```
w_data = [2.0*pi 0.0 0.0]';
% indices for known (iwk) and unknown (iwu*) angular velocities
% and index for unknown moment (ium)
iwk = 1; iwu1 = 2; iwu2 = 3; ium = 1;

tstart = 0.0; tstop = 2.0*pi/abs(w_data(iwk)); nt = 101;
tspan = linspace(tstart, tstop, nt);
dtime = (tstop - tstart)/(nt-1);


options = odeset('AbsTol',1e-10);
[t, th_num] = ode45('fan_solve',tspan,th_init,options);



% obtain accelerations over a full cycle

ag_fan = zeros(nt,2);   mag_ag_fan = zeros(nt,1);
thddot_fan = zeros(nt); mag_thddot_fan = zeros(nt,1);
for it = 1:nt;
   [thdot,thddot,ag] = fan_solve(t(it), th_num(it,:));
   % store results at time it
   ag_fan(it,:) = ag(:,3)';
   mag_ag_fan(it) = norm(ag(:,3));
   thddot_fan(it) = thddot(3);
   mag_thddot_fan(it) = abs(thddot(3));
end;

% plot angular acceleration of link 3

ifig = ifig+1; figure(ifig);
plot(t,thddot_fan);
xlabel('time (s)'); ylabel('angular acceleration (rad/s^2)');
title('angular acceleration of fan for a full cycle');

% plot componenets of linear acceleration of fan's center of mass over cycle

ifig = ifig+1; figure(ifig);
plot(t,ag_fan(:,1),t,ag_fan(:,2));
xlabel('time (s)'); ylabel('linear acceleration (m/s^2)');
title('components of acceleration of fan for a full cycle');

disp(sprintf('expect magnitudes to occur at same time, because'));
disp(sprintf('fan is pinned to ground at D and the linear acceleration'));
disp(sprintf('of the center of mass is a multiple of the angular'));
disp(sprintf('acceleration. \n\n'));
```

```
% find maximum magnitudes of angular and linear acceleration of fan
[max_ag_fan,i_max_ag_fan] = max(mag_ag_fan);
[max_thddot_fan,i_max_thddot_fan] = max(mag_thddot_fan);

disp(sprintf('approximate max magnitudes of linear and angular'));
disp(sprintf('  accelerations are, respectively:\n%6.2f m/s^2 \n%6.2f rad/s^2\n', ...
               max_ag_fan, max_thddot_fan));

if (i_max_ag_fan == i_max_thddot_fan);
 disp(sprintf('both maxima occur roughly %6.2f seconds into the cycle\n', ...
         dtime*(i_max_ag_fan-1)));
else;
 disp(sprintf('something is wrong, should occur at same time\n'));
end;
```

1.3 m files

fan_solve.m

```
function [thdot,thddot,ag] = fan_solve(t, th)
%
% given angles of components, find:
% thdot  -- angular velocities
% thddot -- angular accelerations
% ag     -- linear acceleration of mass centers
%
global r r2 rg lthg;
global iwk iwu1 iwu2 ium;
global w_data;
global gc g_by_gc;
global m ai;
%
nlink = 3; % number of moving links
if (length(th) ~= nlink);
      disp('error, th of wrong length');
      return;
end;
%
% get unit vectors:
% vectors nvec are along the links and vectors tvec are perpendicular to
% the links and in the direction of motion of the end of the link when
% the angular velocity is positive
%
[nvec, tvec] = linkage_vec(th);
%
% solve for angular velocitiess
A = [r(iwu1)*tvec(:,iwu1) r(iwu2)*tvec(:,iwu2)];
```

```
y = [w_data(iwk)*r(iwk)*tvec(:,iwk)];
x = -A\y;
%
% set values in thdot
thdot = zeros(3,1);
thdot(iwk)  = w_data(iwk);
thdot(iwu1) = x(1);
thdot(iwu2) = x(2);
%
%
if (nargout > 1);
% find angular accelerations
%
% get squares of angular velocities
thdot2 = thdot.*thdot;
%
A = [r(iwu1)*tvec(:,iwu1) r(iwu2)*tvec(:,iwu2)];
y = [-thdot2(iwk)* r(iwk)* nvec(:,iwk)  + ...
     -thdot2(iwu1)*r(iwu1)*nvec(:,iwu1) + ...
     -thdot2(iwu2)*r(iwu2)*nvec(:,iwu2)];
x = -A\y;
%
% set values in thddot
thddot = zeros(3,1);
thddot(iwk)  = 0.0;
thddot(iwu1) = x(1);
thddot(iwu2) = x(2);
%
end; % end of calculation of angular accelerations
%
%
if (nargout > 2);
% find linear acceleration of mass centers
%
[ag,ng] = linkage_ag(thdot2,thddot,r,rg,lthg,nvec,tvec);
%
end; % end calculation of linear acceleration of mass centers
%
%
```

---

linkage_ag.m

```
function [ag,ng,tg] = linkage_ag(thdot2,thddot,r,rg,lthg,nvec,tvec)
%
% find linear acceleration of centers of mass
% thdot2 -- squares of angular velocity of links
% thddot -- angular acceleration of links
```

```
% r      -- length of links
% rg     -- length from start of link to center of mass
% lthg   -- angle in local coordinates to center of mass from link start point
% nvec, tvec -- vectors as computed by linkage_vec
%
% all input variables should have entries for each of the links in the
% order in which the links are connected
%
% ag -- linear acceleration of center of mass
% ng -- unit vector pointing from start of link to center of mass
% tg -- unit vector perpendicular to ng such that ng^tg = k
%
% need to find ng and tg because center of mass of link is not
% necessarily along the line between the ends of the link and ng and tg
% are used to locate the center of mass;
%
% more specifically, the variable lthg gives the angle between the line of
% the link and the line from the starting point of the link to the center
% of mass; ng is a unit vector pointing from the start of the link to the
% center of mass and tg is contructed such that ng ^ tg = k
%
nlink = length(thddot);
%
ag = zeros(2,nlink);
ng = zeros(2,nlink);
tg = zeros(2,nlink);
%
% move along links, adding up the acceleration as we go;
% a_base is the linear accleration of the beginning of the current link
% and is set to zero at the start because the first link is pinned to
% "ground"; for each link find the acceleration of the center of
% gravity and also update a_base to be the acceleration of the end of the
% current link (and thus of the beginning of the next link)
%
a_base = [0.0 0.0]';
for i_link = 1:nlink;
   ng(:,i_link) = ...
      nvec(:,i_link)*cos(lthg(i_link)) + tvec(:,i_link)*sin(lthg(i_link));
   tg(:,i_link) = ...
     -nvec(:,i_link)*sin(lthg(i_link)) + tvec(:,i_link)*cos(lthg(i_link));
   ag(:,i_link) = a_base + ...
      -thdot2(i_link)*rg(i_link)*ng(:,i_link) + ...
      +thddot(i_link)*rg(i_link)*tg(:,i_link);
   a_base = a_base + ...
      -thdot2(i_link)*r(i_link)*nvec(:,i_link) + ...
      +thddot(i_link)*r(i_link)*tvec(:,i_link);
end;
```

For `fan_angles.m`, `linkage_vec.m`, `law_cos_len.m`, and `law_cos_ang.m`, see previous solutions.

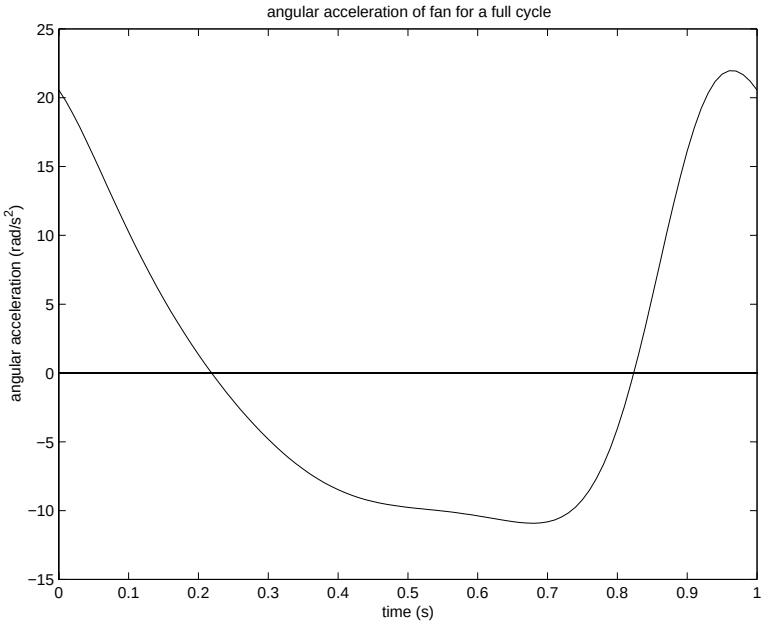

Figure 1: angular acceleration

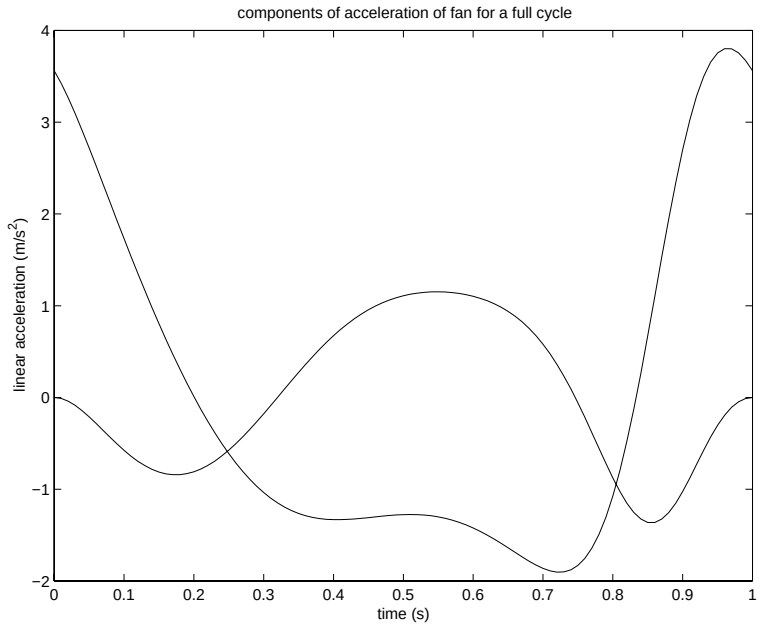

Figure 2: linear acceleration