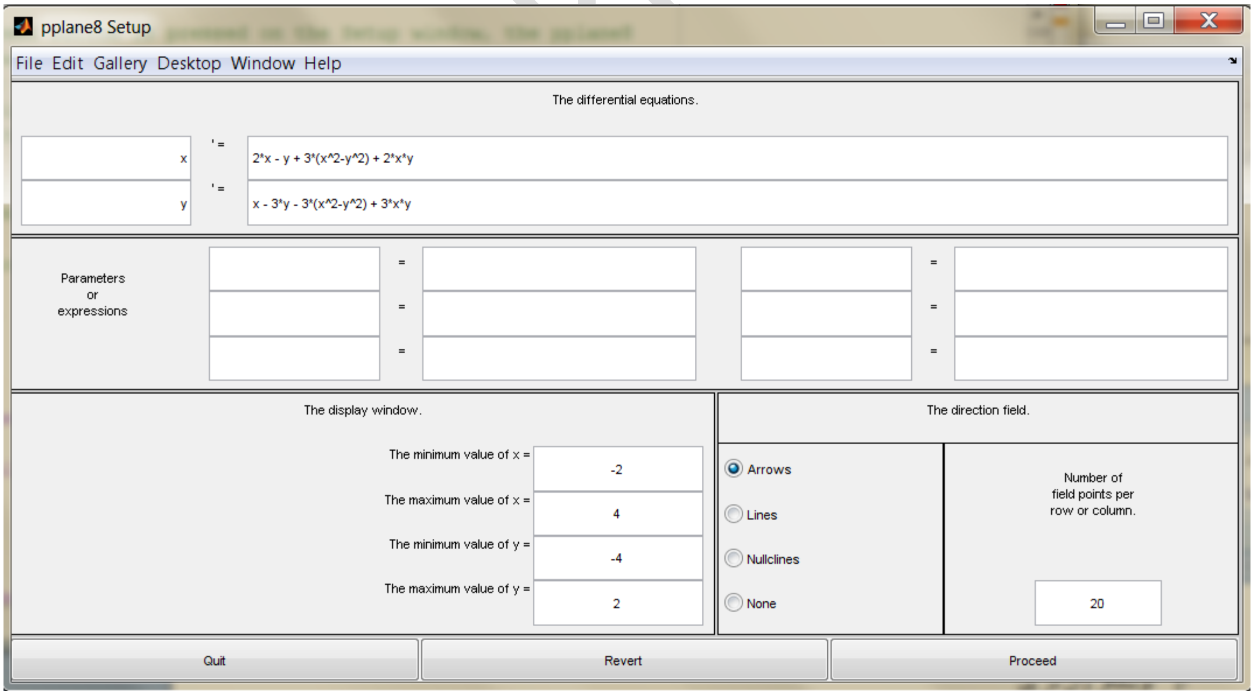
استفاده از متلب برای رسم منحنی‌های مسیر

یک برنامه متلب به نام pplane برای انجام رسم منحنی‌های مسیر یک سیستم خود مختار دارای دو معادله دیفرانسیل به پیوست ارائه می‌گردد. (معادلات به شکل $x' = \mathrm{F}(\mathrm{x},\mathrm{y}), y' = \mathrm{G}(\mathrm{x},\mathrm{y})$ هستند به گونه‌ای که متغیر مستقل t به طور صریح در معادلات ظاهر نمی‌گردد.)

پس از اجرای برنامه در متلب، پنجره ای به شکل زیر باز می گردد که با ورود معادلات دیفرانسیل و تنظیمات دلخواه برای نمایش مثل محدوده محورها، تعداد نقاط میدان در هر سطر یا ستون و ... می‌توان منحنی‌های مسیر را به راحتی رسم نمود . هنگامی که کلید انجام (proceed) در پنجره فشرده می شود پنجره نمایش نمودار فاز باز شده و میدان مورد نظر برای سیستم نمایان می شود. هنگامی که با ماوس روی قسمتی از پنجره نمودار فاز فشار داده شود، حل سیستم با آن شرایط اولیه محاسبه و رسم می گردد. این نمودارها به نمودارهای فاز نیز معروف هستند.



The differential equations.

x ' = 2*x - y + 3*(x^2-y^2) + 2*x*y

y ' = x - 3*y - 3*(x^2-y^2) + 3*x*y

Parameters or expressions

The display window.

The minimum value of x = -2

The maximum value of x = 4

The minimum value of y = -4

The maximum value of y = 2

The direction field.

● Arrows

○ Lines

○ Nullclines

○ None

Number of field points per row or column. 20

نمونه ستاپ نمودار فاز

مثال: مطلوبست رسم نمودار فاز برای دو سیستم معروف زیر به کمک کد متلب

أ)   نوسانگر وان در پل

ب) هیولای دو چشم

a) Van der Pol oscillator

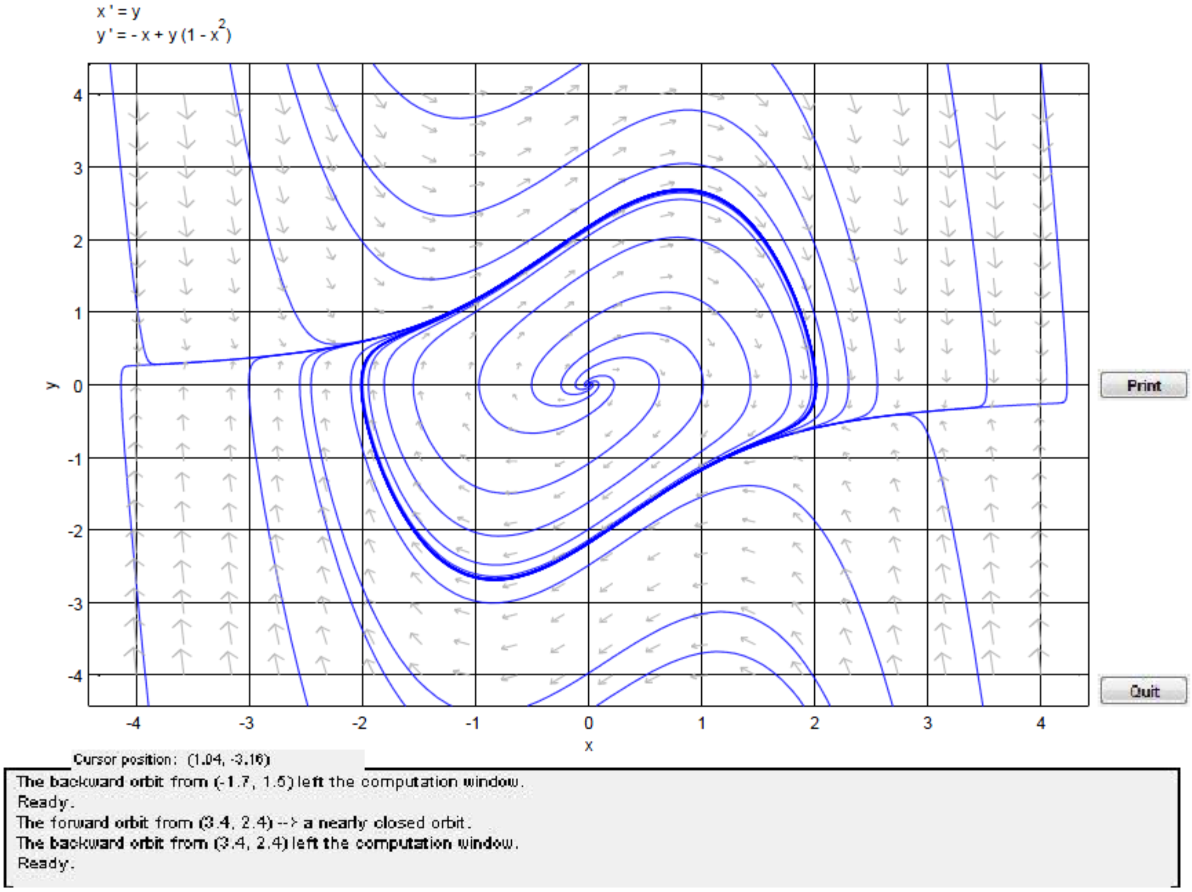
$$\dot{x} = y,\ \ \dot{y} = -x + y(1 - x^2)$$

b) Two-eyed monster

$$\dot{x} = y + y^2,\ \ \dot{y} = -\tfrac{1}{2}x + \tfrac{1}{5}y - xy + \tfrac{6}{5}y^2$$

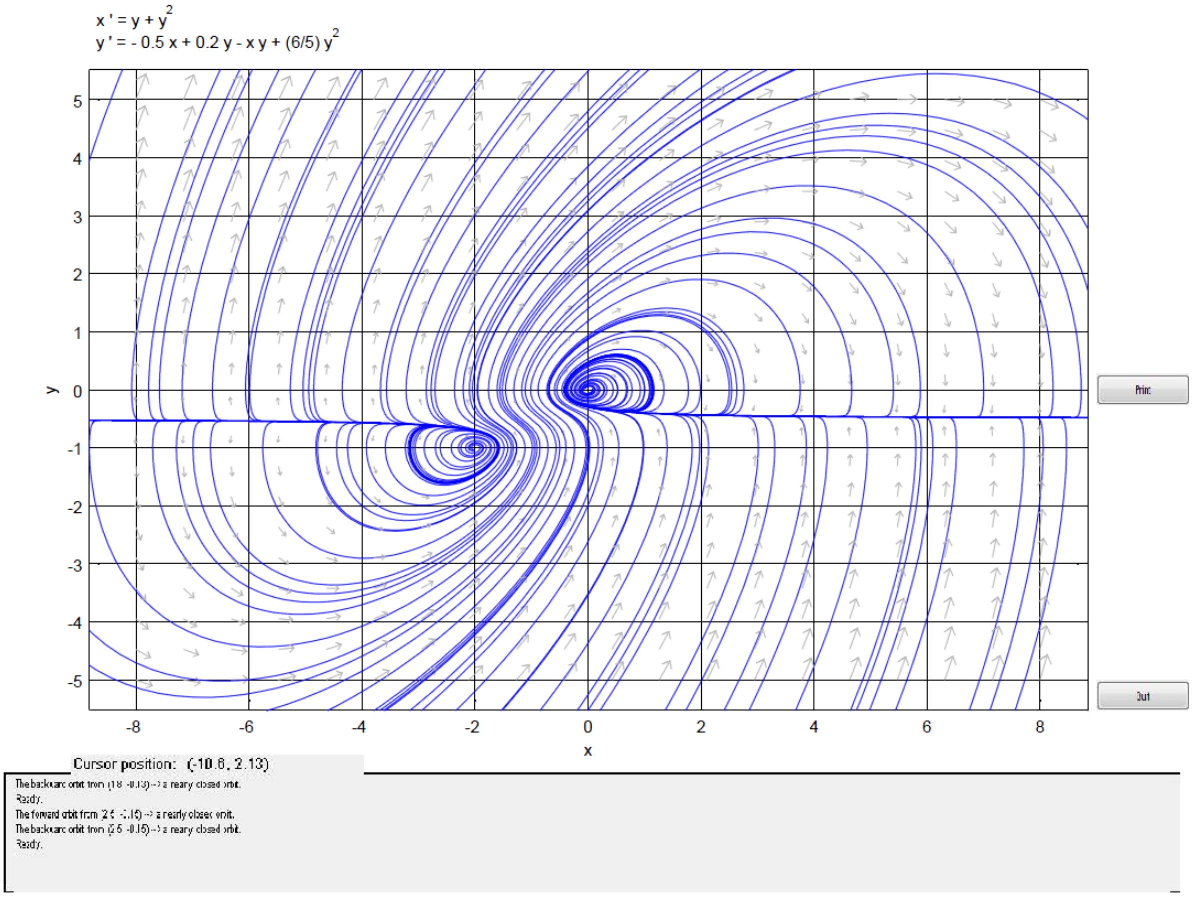
نتایج حاصل از این برنامه برای دو مثال به صورت

آ)



```
The backward orbit from (-1.7, 1.5) left the computation window.
Ready.
The forward orbit from (3.4, 2.4) --> a nearly closed orbit.
The backward orbit from (3.4, 2.4) left the computation window.
Ready.
```

ب)


x ' = y + y^2
y ' = - 0.5 x + 0.2 y - x y + (6/5) y^2
5
4
3
2
1
y 0
-1
-2
-3
-4
-5
-8
-6
-4
-2
0
2
4
6
8
x
Print
Quit
Cursor position: (-10.8 , 2.13)

```
The backward orbit from (1.8 -0.13) --> a nearly closed orbit.
Ready.
The forward orbit from (2.5 -0.15) --> a nearly closed orbit.
The backward orbit from (2.5 -0.15) --> a nearly closed orbit.
Ready.
```

کد برنامه به صورت زیر است:

```
function output = pplane8(action,input1,input2,input3)


startstr = 'pplane8';
if nargin < 1
  action ='initialize';
end



if get(0,'screendepth') == 1
   style = 'bw';
end

if strcmp(action,'initialize')
```

```
   % First we make sure that there is no other copy of pplane8
   % running, since this causes problems.
   pph = findobj('tag','pplane8');
   if ~isempty(pph);
      qstring = {'There are some pplane8 figures open although they may be
invisible.  ';...
            'What do you want to do?'};
      tstring = 'Only one copy of pplane8 can be open at one time.';
      b1str = 'Restart pplane8.';
      b2str = 'Just close those figures.';
      b3str = 'Do nothing.';
      answer = questdlg(qstring,tstring,b1str,b2str,b3str,b1str);
      if strcmp(answer,b1str)
         delete(pph);
         %pplane8;return
         eval(startstr);return
      elseif strcmp(answer,b2str)
         delete(pph);return
      else
         return
      end
   end  % if ~isempty(pph);

   % Make sure tempdir is on the MATLABPATH.  We want to be sure that we
   % return the path to its current position when we exit.

   p = path;
   tmpdir = tempdir;
   ll = length(tmpdir);
   tmpdir = tmpdir(1:ll-1);
   ud.remtd = 0;
   if isempty(findstr(tmpdir,p))
     ud.remtd = 1;
     addpath(tempdir)
   end

   % Next we look for old files created by pplane8.
   % First in the current directory.

   oldfiles = dir('pptp*.m');
   ofiles = cell(0,1);
   kk = zeros(0,1);
   for k = 1:length(oldfiles)
      fn = oldfiles(k).name;
      fid = fopen(fn,'r');
      ll = fgetl(fid);
      ll = fgetl(fid);
      ll = fgetl(fid);
      fclose(fid);
      if strcmp(ll,'%% Created by pplane8')
         delete(fn)
      else
    l= length(ofiles);
    ofiles{l+1} = fn;
      end
```

```
end

%Then in the temp directory.

oldfiles = dir([tempdir,'pptp*.m']);
for k = 1:length(oldfiles)
   fn = [tempdir,oldfiles(k).name];
   fid = fopen(fn,'r');
   ll = fgetl(fid);
   ll = fgetl(fid);
   ll = fgetl(fid);
   fclose(fid);
   if strcmp(ll,'%% Created by pplane8')
      delete(fn)
   else
 l= length(ofiles);
 ofiles{l+1} = fn;
   end
end
lll = length(ofiles);
if lll >0
  if lll == 1
    astr = 'The file';
    bstr = 'If so it can be safely deleted.';
  else
    astr = 'The files';
    bstr = 'If so they can be safely deleted.';
  end
  fprintf([astr,'\n']);
  for j = 1:lll
    fn = ofiles{j};
    disp(['     ',fn]);
  end
  fprintf('may have been created by an older version of PPLANE.\n');
  fprintf(bstr,'\n\n');
end

style = 'white';
ppdir = pwd;
ssize = get(0,'defaultaxesfontsize');
npts = 20;
solver = 'Dormand Prince';
tol = 1e-4;
stepsize = 0.1;
if exist('ppstart','file')
  H = ppstart;
  if ~isempty(H)
    if isfield(H,'style')
  style = H.style;
    end
    if isfield(H,'size')
  ssize = H.size;
    end
    if isfield(H,'npts')
  npts = H.npts;
    end
    if isfield(H,'solver')
  solver = H.solver;
    end
    if isfield(H,'ppdir')
  ppdir = H.ppdir;
    end
    if isfield(H,'stepsize')
  stepsize = H.stepsize;
    end
    if isfield(H,'tolerance')
  tol = H.tolerance;
    end
  end
end
if get(0,'screendepth') == 1
  style = 'bw';
end

ud.ssize = ssize;
ud.ppdir = ppdir;
comp = computer;
if strcmp(comp,'PCWIN')
  ud.fontsize = 0.8*ud.ssize;
else
  ud.fontsize = ud.ssize;
end

% Set up for the menu of systems.

system.name = 'default system';
system.xvar = 'x';
system.yvar = 'y';
system.xder = ' 2*x - y + 3*(x^2-y^2) + 2*x*y';
system.yder = ' x - 3*y - 3*(x^2-y^2) + 3*x*y';
system.pname = {};
system.pval = {};
system.fieldtype = 'arrows';
system.npts = npts;
system.wind = [-2 4 -4 2];

system(2).name = 'linear system';
system(2).xvar = 'x';
system(2).yvar = 'y';
system(2).xder = ' A*x + B*y';
system(2).yder = ' C*x + D*y';
system(2).pname = {'A', 'B', 'C', 'D', '', ''};
system(2).pval = {'2', '2', '-2',  '-3'};
system(2).fieldtype = 'arrows';
system(2).npts = npts;
system(2).wind = [-5 5 -5 5];

system(3).name = 'vibrating spring';
system(3).xvar = 'x';
system(3).yvar = 'v';
system(3).xder = ' v';
system(3).yder = ' -(k*x + d*v)/m';
system(3).pname = {'k', 'm', 'd', '', '', ''};
system(3).pval = {'3', '1', '0'};
system(3).fieldtype = 'arrows';
system(3).npts = npts;
system(3).wind = [-5 5 -5 5];

system(4).name = 'pendulum';
system(4).xvar = '\theta';
system(4).yvar = '\omega';
system(4).xder = ' \omega';
system(4).yder = ' -sin(\theta) - D*\omega';
system(4).pname = {'D', '', '', '', '', ''};
system(4).pval = {'0'};
system(4).fieldtype = 'arrows';
system(4).npts = npts;
system(4).wind = [-10 10 -4 4];

system(5).name = 'predator prey';
system(5).xvar = 'prey';
system(5).yvar = 'predator';
system(5).xder = ' (A - B*predator)*prey';
system(5).yder = ' (D*prey - C)*predator';
system(5).pname = {'A', 'B', 'C', 'D', '', ''};
system(5).pval = {'0.4', '0.01', '0.3', '0.005'};
system(5).fieldtype = 'arrows';
system(5).npts = npts;
system(5).wind = [0 120 0 80];

system(6).name = 'competing species';
system(6).xvar = 'x';
system(6).yvar = 'y';
system(6).xder = ' r*(1 - x - A*y)*x';
system(6).yder = ' s*(1 - y - B*x)*y';
system(6).pname = {'r', 's', 'A', 'B', '', ''};
system(6).pval = {'0.4', '0.6', '5', '4'};
system(6).fieldtype = 'nullclines';
system(6).npts = npts;
system(6).wind = [0 1 0 1];

system(7).name = 'cooperative species';
system(7).xvar = 'x';
system(7).yvar = 'y';
system(7).xder = ' r*(1 - x + A*y)*x';
system(7).yder = ' s*(1 - y + B*x)*y';
system(7).pname = {'r', 's', 'A', 'B', '', ''};
system(7).pval = {'0.4', '0.6', '2', '0.3'};
system(7).fieldtype = 'nullclines';
system(7).npts = npts;
system(7).wind = [0 9 0 5];

system(8).name = 'van der Pol''s equation';
system(8).xvar = 'x';
system(8).yvar = 'y';
system(8).xder = ' M*x - y - x^3';
system(8).yder = ' x';
system(8).pname = {'M', '', '', '', '', ''};
system(8).pval = {'2'};
system(8).fieldtype = 'arrows';
system(8).npts = npts;
system(8).wind = [-5 5 -5 5];

system(9).name = 'Duffing''s equation';
system(9).xvar = 'x';
system(9).yvar = 'y';
system(9).xder = ' y';
system(9).yder = ' -(k*x + c*y + l*x^3)/m';
system(9).pname = {'k', 'c', 'm', 'l', '', ''};
system(9).pval = {'-1', '0', '1', '1'};
system(9).fieldtype = 'arrows';
system(9).npts = npts;
system(9).wind = [-3 3 -3 3];

system(10).name = 'square limit set';
system(10).xvar = 'x';
system(10).yvar = 'y';
system(10).xder = ' (y + x/5)*(1-x^2)';
system(10).yder = ' -x*(1-y^2)';
system(10).pname = {'', '', '', '', '', ''};
system(10).pval = {'', '', '', ''};
system(10).fieldtype = 'arrows';
system(10).npts = npts;
system(10).wind = [-1.5 1.5 -1 1];


ud.c = system(1); % Changed values.
pname = ud.c.pname;
for kk = length(pname)+1:6
  pname{kk} = '';
end
pval = ud.c.pval;
for kk = length(pval)+1:6
  pval{kk} = '';
end
ud.c.pname = pname;
ud.c.pval = pval;
ud.o = ud.c;  % Original values.
% ud.h = system(1);   % This will be the handles in the
          % setup window.

ud.style = style;
ud.npts = npts;
ud.settings.magn = 1.25;
ud.settings.refine = 8;
ud.settings.tol = tol;
ud.settings.solver = solver;
ud.settings.stepsize = stepsize;
ud.settings.speed = 100;
ud.system = system;
ud.solvers = {'ppdp45';
      'rk4';
      'ode45';
      'ode23';
      'ode113';
      'ode15s';
      'ode23s';
      'ode23t';
      'ode23tb'};

switch style
 case 'black'
  color.temp = [1 0 0]; % red for temporary orbits
  color.orb = [1 1 0];  % yellow for orbits
  color.eqpt = [1 0 0];  % red for eq. pts.
             %    color.arrows = [0 1 1]; % cyan for arrows
             % color.arrows = [.5 .5 .9];  % purple for arrows
  color.arrows = .5*[1 1 1];  % gray for arrows
  color.narrows = .7*[1 1 1];  % gray for nullcline arrows
  color.tx = [1 1 0]; % yellow for xt plots & 3D plots
  color.ty = [1 0 0]; % red for yt plots
  color.ta = [1 0 0]; % red for axis plots
  color.sep = [.2,1,0]; % green for separatrices
  color.xcline = [1 1 1]; % white for xclines
  color.ycline = [1 0 1]; % magenta for yclines
  color.level = [1,.5,.5];

 case 'white'
  color.temp = [1 0 0]; % red for temporary orbits
  color.orb = [0 0 1];  % blue for orbits
  color.eqpt = [1 0 0];  % red for eq. pts.
  color.arrows = 0.7*[1 1 1]; % gray for arrows
  color.narrows = .4*[1 1 1];  % gray for nullcline arrows
  color.tx = [0 0 1]; % blue for xt plots & 3D plots
  color.ty = [1 0 0]; % red for yt plots
  color.ta = [1 0 0]; % red for axis plots
  color.sep = [0,1,0];% green for separatrices
  color.xcline = [1 0 .75]; % magenta for xclines
  color.ycline = [1 .5 0]; %  orange for yclines
  color.level = 0.8*[.9,.5,.8];

 case 'test'
  color.temp = [1 0 0]; % red for temporary orbits
  color.orb = [0 0 1];  % blue for orbits
  color.arrows = .7*[1 1 1]; % gray for arrows
  color.eqpt = [1 0 0];  % red for eq. pts.
  color.narrows = .4*[1 1 1];  % gray for nullcline arrows
  color.tx = [0 0 1]; % blue for xt plots & 3D plots
  color.ty = [1 0 0]; % red for yt plots
  color.ta = [1 0 0]; % red for axis plots
  color.sep = [0,1,0];% green for separatrices
  color.xcline = [1 0 .75]; % magenta for xclines
  color.ycline = [1 .5 0]; %  orange for yclines
  color.level = [1,.5,.5];

 case 'display'
  color.temp = [1 0 0]; % red for temporary orbits
  color.orb = [0 0 1];  % blue for orbits
  color.arrows = .4*[1 1 1]; % gray for arrows
  color.eqpt = [1 0 0];  % red for eq. pts.
  color.narrows = .4*[1 1 1];  % gray for nullcline arrows
  color.tx = [0 0 1]; % blue for xt plots & 3D plots
  color.ty = [1 0 0]; % red for yt plots
  color.ta = [1 0 0]; % red for axis plots
  color.sep = 2*[.5 0 .5];% [0,1,0];% green for separatrices
  color.xcline = [1 0 .75]; % magenta for xclines
  color.ycline = [1 .5 0]; %  orange for yclines
  color.level = 0.8*[.9,.5,.8];

 case 'bw'
  color.temp = [1 1 1]; % white for everything
  color.orb = [1 1 1];
  color.eqpt = [1 1 1];
  color.arrows = [1 1 1];
  color.narrows = [1 1 1];
  color.tx = [1 1 1];
  color.ty = [1 1 1];
  color.ta = [1 1 1];
  color.sep = [1 1 1];
  color.xcline = [1 1 1];
  color.ycline = [1 1 1];
  color.level = [1,1,1];
end
ud.color = color;
ud.level = ' ';
ppset = figure('name','pplane8 Setup','numb','off',...
       'tag','pplane8','visible','off',...
       'user',ud);

pplane8('figdefault',ppset);
frame(1) = uicontrol('style','frame','visible','off');
eq(1)=uicontrol('style','text',...
        'horizon','center',...
        'string','The differential equations.','visible','off');
ext = get(eq(1),'extent');
rr=ext(4)/10;

texth =ext(4)+4;    % 19;         % Height of text boxes.
varw = 40*rr;     % Length of variable boxes.
equalw =13*rr;        % Length of equals.(30)
eqlength = 230*rr;        % Length of right hand sides of equations.
winstrlen = 120*rr;   % Length of string boxes in display frame.
left = 0;     % Left margin of the frames.
frsep = 1;    %3;     % Separation between frames.
separation = texth;   % Separation between boxes.

dfigwidth =2*left + varw+equalw+eqlength+10;  % Width of the figure.
dfigurebot = 30;  % Bottom of the figure.
buttw = dfigwidth/3;
qwind = [0,frsep,buttw,separation];   % Quit button
rwind = [buttw,frsep,buttw,separation];   % Revert "
pwind = [2*buttw,frsep,buttw,separation]; % Proceed "

disfrbot = 2*frsep + separation;  % Display frame.
disfrw = winstrlen + varw +10;
disfrht = 5*separation + 10;
```

```
disfrwind = [left, disfrbot, disfrw, disfrht];

pfrbot = disfrbot + disfrht +frsep;   % Parameter frame.
pfrw = dfigwidth -2*left;
pfrht = 3*separation + 10;
pfrwind = [left, pfrbot, pfrw, pfrht];

defrbot = pfrbot + pfrht + frsep; % Equation frame.
defrw = pfrw;
defrht = 3*separation + 10;
defrwind = [left, defrbot, defrw, defrht];

ffrbot = disfrbot;             % Field frame.
ffrleft = left + disfrw + frsep;
ffrw = dfigwidth -left - ffrleft;
ffrht = disfrht;
ffrwind = [ffrleft, ffrbot, ffrw, ffrht];

dfigureheight = defrbot + defrht +frsep;  % Height of the figure.

set(ppset,'pos',[30 dfigurebot dfigwidth dfigureheight]);

set(frame(1),'pos',defrwind);


xname=[
    'ud = get(gcf,''user'');'...
    'Xname=get(ud.h.xvar,''string'');'...
    'minxstr = [''The minimum value of '',Xname,'' = ''];',...
    'set(ud.h.twind(1),''string'',minxstr);'...
    'maxxstr = [''The maximum value of '',Xname,'' = ''];',...
    'set(ud.h.twind(2),''string'',maxxstr);'...
    'ud.c.xvar = Xname;'...
    'ud.flag = 0;'...
    'ud.c.name = '''';',...
    'set(gcf,''user'',ud);'];

yname=[
    'ud = get(gcf,''user'');'...
    'Yname=get(ud.h.yvar,''string'');'...
    'minystr = [''The minimum value of '',Yname,'' = ''];',...
    'set(ud.h.twind(3),''string'',minystr);'...
    'maxystr = [''The maximum value of '',Yname,'' = ''];',...
    'set(ud.h.twind(4),''string'',maxystr);'...
    'ud.c.yvar = Yname;'...
    'ud.flag = 0;'...
    'ud.c.name = '''';',...
    'set(gcf,''user'',ud);'];

xder =[
    'ud = get(gcf,''user'');'...
    'ud.c.xder = get(ud.h.xder,''string'');'...
    'ud.flag = 0;'...
    'ud.c.name = '''';',...
    'set(gcf,''user'',ud);'];
```

```
yder =[
    'ud = get(gcf,''user'');'...
    'ud.c.yder = get(ud.h.yder,''string'');'...
    'ud.flag = 0;'...
    'ud.c.name = '''';',...
    'set(gcf,''user'',ud);'];

equationbot = defrbot + 5;
eqlabelbot = equationbot + 2*separation;
xbot = equationbot + separation;      % Bottom of x equation.
ybot = equationbot;                   % Bottom of y equation.

lablen =200*rr;

eqlableft = (dfigwidth-lablen)/2;
eqleft = left + 5;
fudge = 0.15*separation;

set(eq(1),'pos',[eqlableft eqlabelbot lablen texth]);
tcolor = get(gcf,'defaultuicontrolbackgroundcolor');
ecolor = 'w';

ud.h.xvar=uicontrol('pos',[eqleft, xbot, varw, texth],...
            'style','edit',...
            'horizon','right',...
            'string',ud.o.xvar,...
            'call',xname,...
            'backgroundcolor',ecolor,...
            'visible','off');

eq(2) = uicontrol('style','text',...
          'pos',[eqleft+varw xbot equalw texth],...
          'horizon','center',...
          'string',''' = ',...
          'backgroundcolor',tcolor,...
          'visible','off');

ud.h.xder=uicontrol('pos',[eqleft+varw+equalw xbot eqlength texth],...
            'string',ud.o.xder,...
            'horizon','left','style','edit',...
            'backgroundcolor',ecolor,...
            'call',xder,'visible','off');

ud.h.yvar=uicontrol('pos',[eqleft ybot varw texth],...
            'style','edit',...
            'horizon','right',...
            'string',ud.o.yvar,...
            'backgroundcolor',ecolor,...
            'call',yname,'visible','off');

eq(3) = uicontrol('style','text',...
          'pos',[eqleft+varw ybot equalw texth],...
          'horizon','center','string',''' = ',...
          'backgroundcolor',tcolor',...
          'visible','off');

ud.h.yder=uicontrol('pos',[eqleft+varw + equalw ybot eqlength texth],...
            'string',ud.o.yder,...
            'horizon','left','style','edit',...
            'backgroundcolor',ecolor,...
            'call',yder,'visible','off');


frame(2) = uicontrol('style','frame','pos',disfrwind,'visible','off');

w1 = [
    'ud = get(gcf,''user'');'...
    'nnn =    str2num(get(ud.h.wind(1),''string''));'...
    'if isempty(nnn),',...
    '  set(ud.h.wind(1),''string'',''?'');',...
    '  nnn = NaN;',...
    'end,',...
    'ud.c.wind(1) = nnn;',...
    'ud.c.name = '''';',...
    'set(gcf,''user'',ud);'];

w2 = [
    'ud = get(gcf,''user'');'...
    'nnn =    str2num(get(ud.h.wind(2),''string''));'...
    'if isempty(nnn),',...
    '  set(ud.h.wind(2),''string'',''?'');',...
    '  nnn = NaN;',...
    'end,',...
    'ud.c.wind(2) = nnn;',...
    'ud.c.name = '''';',...
    'set(gcf,''user'',ud);'];

w3 = [
    'ud = get(gcf,''user'');'...
    'nnn =    str2num(get(ud.h.wind(3),''string''));'...
    'if isempty(nnn),',...
    '  set(ud.h.wind(3),''string'',''?'');',...
    '  nnn = NaN;',...
    'end,',...
    'ud.c.wind(3) = nnn;',...
    'ud.c.name = '''';',...
    'set(gcf,''user'',ud);'];

w4 = [
    'ud = get(gcf,''user'');'...
    'nnn =    str2num(get(ud.h.wind(4),''string''));'...
    'if isempty(nnn),',...
    '  set(ud.h.wind(4),''string'',''?'');',...
    '  nnn = NaN;',...
    'end,',...
    'ud.c.wind(4) = nnn;',...
    'ud.c.name = '''';',...
    'set(gcf,''user'',ud);'];
```

```
winbot1 = disfrbot + disfrht - 5 - separation;
winbot2 = winbot1 - separation;
winbot3 = winbot2 - separation;
winbot4 = winbot3 - separation;
winbot5 = winbot4 - separation;

dwindow = uicontrol('style','text',...
          'pos',[eqleft winbot1 disfrw-10 texth],...
          'horizon','center',...
          'string','The display window.','visible','off');

% ud.h.twind contains the handles to the text windows, and ud.h.wind
% contains the handles to the edit windows.

twstr1 = ['The minimum value of ',ud.o.xvar,' = '];
ud.h.twind(1) = uicontrol('style','text',...
              'pos',[eqleft winbot2 winstrlen texth],...
              'horizon','right',...
              'string',twstr1,...
              'backgroundcolor',tcolor',...
              'visible','off');

ud.h.wind(1) = uicontrol('style','edit',...
             'pos',[eqleft+winstrlen winbot2 40*rr texth],...
             'string',num2str(ud.o.wind(1)),...
             'backgroundcolor',ecolor,...
             'call',w1,'visible','off');

twstr2 = ['The maximum value of ',ud.o.xvar,' = '];
ud.h.twind(2) = uicontrol('style','text',...
              'pos',[eqleft winbot3 winstrlen texth],...
              'horizon','right',...
              'string',twstr2,...
              'backgroundcolor',tcolor',...
              'visible','off');

ud.h.wind(2) = uicontrol('style','edit',...
             'pos',[eqleft+winstrlen winbot3 40*rr texth],...
             'string',num2str(ud.o.wind(2)),...
             'backgroundcolor',ecolor,...
             'call',w2,'visible','off');

twstr3 = ['The minimum value of ',ud.o.yvar,' = '];
ud.h.twind(3)= uicontrol('style','text',...
             'pos',[eqleft winbot4 winstrlen texth],...
             'horizon','right',...
             'string',twstr3,...
              'backgroundcolor',tcolor',...
             'visible','off');

ud.h.wind(3) = uicontrol('style','edit',...
             'pos',[eqleft+winstrlen winbot4 40*rr texth],...
             'string',num2str(ud.o.wind(3)),...
             'backgroundcolor',ecolor,...
             'call',w3,'visible','off');

twstr4 = ['The maximum value of ',ud.o.yvar,' = '];
ud.h.twind(4) = uicontrol('style','text',...
              'pos',[eqleft winbot5 winstrlen texth],...
              'horizon','right',...
              'string',twstr4,...
              'backgroundcolor',tcolor',...
              'visible','off');

ud.h.wind(4) = uicontrol('style','edit',...
             'pos',[eqleft+winstrlen winbot5 40*rr texth],...
             'string',num2str(ud.o.wind(4)),...
             'backgroundcolor',ecolor,...
             'call',w4,'visible','off');


frame(3)=uicontrol('style','frame','pos',pfrwind,'visible','off');

pncall = [
    '[h,fig] = gcbo;'...
    'ud =  get(fig,''user'');'...
    'num = get(h,''user'');'...
    'ud.c.pname{num} = get(ud.h.pname(num),''string'');'...
    'ud.flag = 0;'...
    'set(gcf,''user'',ud);'];

pvcall = [
    '[h,fig] = gcbo;'...
    'ud =  get(fig,''user'');'...
    'num = get(h,''user'');'...
    'ud.c.pval{num} = get(ud.h.pval(num),''string'');'...
    'ud.flag = 0;'...
    'set(gcf,''user'',ud);'];


pnamew = 40*rr;
pvalw = 50*rr;
peqw = 10*rr;
pbot(3) = pfrbot + 5;
pbot(2) = pbot(3) + separation;
pbot(1) = pbot(2) + separation;

pleft1 = eqleft + 50*rr + 5;
peqleft1 = pleft1 + pnamew;
pvleft1 = peqleft1 + peqw;
pleft2 = dfigwidth - 10 - pnamew - pvalw - peqw;
peqleft2 = pleft2 + pnamew;
pvleft2 = peqleft2 + peqw;

paratit=uicontrol('style','text',...
          'horizon','center',...
          'string',{'Parameters';'or';'expressions'},...
          'backgroundcolor',tcolor',...
          'visible','off');
```

```
ext = get(paratit,'extent');
paratitw = ext(3);
pos = [eqleft pfrbot+2+texth/2 paratitw 2.1*texth];
set(paratit,'pos',pos);
psep = 20;
pvalw = (dfigwidth - 2*eqleft - paratitw)/2 - psep - pnamew - peqw;
pval = ud.c.pval;
pname = ud.c.pname;
for jj = 1:3
  for kk = 1:2
    pleft = eqleft + paratitw +psep +(kk-1)*(pnamew+peqw+pvalw+ psep);
    peqleft = pleft + pnamew;
    pvleft = peqleft + peqw;
    K = kk +2*(jj-1);
    name = pname{K};
    value = pval{K};
    ud.h.pname(K) = uicontrol('style','edit',...
              'pos',[pleft pbot(jj) pnamew texth],...
              'horizon','right','string',name,...
              'user',K,...
              'call',pncall,...
              'visible','off',...
              'backgroundcolor','w');
    equal(K) = uicontrol('style','text',...
             'pos',[peqleft pbot(jj)-fudge peqw texth],...
             'horizon','center',...
             'string','=',...
             'visible','off');

    ud.h.pval(K) = uicontrol('style','edit',...
                 'pos',[pvleft pbot(jj) pvalw texth],...
                 'string',value,...
                 'call',pvcall,...
                 'visible','off',...
                 'user',K,...
                 'backgroundcolor','w');
  end
end

ud.c.pname = pname;
ud.c.pval = pval;

butt(1) = uicontrol('style','push',...
            'pos',qwind,...
            'string','Quit','call',...
            'pplane8(''quit'')',...
            'visible','off');

butt(2) = uicontrol('style','push',...
            'pos',rwind,...
            'string','Revert',...
            'call','pplane8(''revert'')',...
            'visible','off');

butt(3) = uicontrol('style','push',...
            'pos',pwind,...
            'string','Proceed',...
            'call','pplane8(''proceed'')',...
            'visible','off');

fframe = uicontrol('style','frame','pos',ffrwind,'visible','off');



ffrtitle = uicontrol('style','text',...
             'pos',[ffrleft+5,winbot1,ffrw-10,texth],...
             'string','The direction field.',...
             'horizon','center','visible','off');

radleft = ffrleft + 3;
radw = 50*rr;

typewindw = radw +6;
typewind = [ffrleft, ffrbot, typewindw, ffrht-separation-3];
textwindl = ffrleft+typewindw;
textleft = textwindl + 3;
textw = ffrw - typewindw;
textwind = [textwindl, ffrbot,textw, ffrht-separation-3];
typeframe = uicontrol('style','frame','pos',typewind,'visible','off');

textframe = uicontrol('style','frame','pos',textwind,'visible','off');

switch ud.o.fieldtype
 case 'nullclines'
  rval1 = 1;rval2 = 0;rval3 = 0;rval4 = 0;
 case 'lines'
  rval1 = 0;rval2 = 2;rval3 = 0;rval4 = 0;
 case 'arrows'
  rval1 = 0;rval2 = 0;rval3 = 3;rval4 = 0;
 case 'none'
  rval1 = 0;rval2 = 0;rval3 = 0;rval4 = 4;
 otherwise
  error(['Unknown fieldtype ',ud.o.fieldtype,'.'])
end

ud.h.rad(1) = uicontrol('style','radio',...
            'pos',[radleft winbot4 radw texth],...
            'string','Nullclines',...
            'value',rval1,...
            'visible','off');

ud.h.rad(2) = uicontrol('style','radio',...
            'pos',[radleft winbot3 radw texth],...
            'string','Lines',...
            'value',rval2,...
            'max',2,...
            'visible','off');

ud.h.rad(3) = uicontrol('style','radio',...
            'pos',[radleft winbot2 radw texth],...
            'string','Arrows',...
            'value',rval3,...
            'max',3,...
            'visible','off');

ud.h.rad(4) = uicontrol('style','radio',...
            'pos',[radleft winbot5 radw texth],...
            'string','None',...
            'value',rval4,...
            'max',4,...
            'visible','off');

for i=1:4
  set(ud.h.rad(i),'user',ud.h.rad(:,[1:(i-1),(i+1):4]));
end

callrad = [
    'me = get(gcf,''currentobject'');',...
    'kk = get(me,''max'');',...
    'set(get(me,''user''),''value'',0),',...
    'set(me,''value'',kk);',...
    'ud = get(gcf,''user'');',...
    'switch kk,',...
    '       case 1, ud.c.fieldtype = ''nullclines'';',...
    '       case 2, ud.c.fieldtype = ''lines'';',...
    '       case 3, ud.c.fieldtype = ''arrows'';',...
    '       case 4, ud.c.fieldtype = ''none'';',...
    'end,',...
    'set(gcf,''user'',ud);'];

set(ud.h.rad,'call',callrad);

nfptsstr = {'Number of'; 'field points per'; 'row or column.'};
nfptstext = uicontrol('style','text',...
          'pos',[textleft winbot4 textw-5 2.5*texth],...
          'string',nfptsstr,...
          'horizon','center',...
          'visible','off');

callnfpts = [
    'ppset = findobj(''name'',''pplane8 Setup'');',...
    'ud = get(ppset,''user'');'...
    'me = ud.h.npts;',...
    'kk = str2num(get(me,''string''));',...
    'if isempty(kk),',...
    '  set(me,''string'',''?'');',...
    '  kk = NaN;',...
    'else,',...
    '  kk = floor(kk);',...
    '  [m,N] = computer;'...
    '  if (N <= 8192),',...
    ' N = 32;',...
    '  else,',...
    ' N = 50;',...
    '  end,'...
    '  kk = min([N,max([5,kk])]);'...
    '  set(me,''string'',num2str(kk));'...
    'end,'...
    'ud.c.npts = kk;',...
    'set(ppset,''user'',ud);'];

npos = [textleft+(textw -30*rr)/2,winbot5 30*rr,texth];
ud.h.npts = uicontrol('style','edit',...
          'pos',npos,...
          'string',ud.o.npts,...
          'call',callnfpts,...
          'backgroundcolor','w',...
          'visible','off');

delgall = ['sud = get(gcf,''user'');',...
       'mh = get(sud.h.gallery,''children'');',...
       'add = findobj(sud.h.gallery,''tag'',''add system'');',...
       'mh(find(mh == add)) = [];',...
       'delete(mh);',...
       'set(sud.h.gallery,''user'',[]);',...
       'set(findobj(''tag'',''load default''),''enable'',''on'')'];


% Menus

hhsetup = get(0,'showhiddenhandles');
set(0,'showhiddenhandles','on');
mefile = findobj(ppset,'label','&File');
meedit = findobj(ppset,'label','&Edit');
delete(findobj(ppset,'label','&Tools'));
delete(findobj(ppset,'label','&View'));
delete(findobj(ppset,'label','&Insert'));

% File menu

meexp = findobj(mefile,'label','&Export...');
meprev = findobj(mefile,'label','Print Pre&view...');
mepset = findobj(mefile,'label','Pa&ge Setup...');
set(get(mefile,'child'),'vis','off');
meload = uimenu(mefile,'label','Load a system ...',...
        'call','pplane8(''loadsyst'',''system'');',...
        'pos',1);
mesave = uimenu(mefile,'label','Save the current system ...',...
        'call','pplane8(''savesyst'',''system'');',...
        'pos',2);
meloadg = uimenu(mefile,'label','Load a gallery ...',...
         'call','pplane8(''loadsyst'',''gallery'');',...
         'separator','on','pos',3);
mesaveg = uimenu(mefile,'label','Save a gallery ...',...
         'call','pplane8(''savesyst'',''gallery'');',...
         'tag','savegal','pos',4);
medelg = uimenu(mefile,'label','Delete the current gallery',...
        'call',delgall,'pos',5);
melddg = uimenu(mefile,'label','Load the default gallery',...
        'call','pplane8(''loadsyst'',''default'');',...
        'enable','on',...
        'tag','load default','pos',6);
```

```
merevert = uimenu(mefile,'label','Revert','call',...
          'pplane8(''revert'')',...
          'separator','on','pos',7);
meproceed = uimenu(mefile,...
           'label','Proceed',...
           'call','pplane8(''proceed'')',...
           'separator','off',...
           'accelerator','G','pos',8);
set(mepset,'vis','on','pos',9);
set(meprev,'vis','on','pos',10);
set(meexp,'vis','on','pos',11,'separator','off');
merestart = uimenu(mefile,'label',...
           'Restart pplane8',...
           'call','pplane8(''restart'')',...
           'separator','on','pos',12);

mequit = uimenu(mefile,...
        'label','Quit pplane8',...
        'call','pplane8(''quit'')',...
        'separator','off','pos',13);

% Edit menu

set(get(meedit,'child'),'vis','off');
meclrf = uimenu(meedit,'label','Clear equations',...
        'call',['ud = get(gcf,''user'');h = ud.h;',...
          'set([h.xvar,h.xder,h.yvar,h.yder],''string'','''');'],...
        'accelerator','E');

pclear = [
    'ud = get(gcf,''user'');h = ud.h;',...
    'set([h.pname,h.pval],''string'','''');',...
    'ud.c.pname = {'''','''','''','''','''','''',''''};',...
    'ud.c.pval = {'''','''','''','''','''','''',''''};',...
    'set(gcf,''user'',ud);',...
     ];

meclrp = uimenu(meedit,'label','Clear parameters',...
    'call',pclear,...
    'accelerator','N');

meclrwind = uimenu(meedit,'label','Clear display window',...
    'call',['ud = get(gcf,''user'');',...
       'set(ud.h.wind,''string'','''');'],...
    'accelerator','D');

allclear = [
    'ud = get(gcf,''user'');h = ud.h;',...
    'set([h.xvar,h.xder,h.yvar,h.yder],''string'','''');',...
    'set([h.pname,h.pval,h.wind],''string'','''');',...
    'ud.c.pname = {'''','''','''','''','''','''',''''};',...
    'ud.c.pval = {'''','''','''','''','''','''',''''};',...
    'set(gcf,''user'',ud);',...
       ];
```

```
  meclrall = uimenu(meedit,'label','Clear all',...
      'call',allclear,...
      'accelerator','A',...
      'separator','on');

  % Gallery menu.

   sysmenu = uimenu('label','Gallery','visible','off','pos',3);

   meadd = uimenu(sysmenu,'label','Add current system to the gallery',...
      'call','pplane8(''addgall'');','tag','add system');
   sep = 'on';
   for kk = 1:length(system)
      kkk = num2str(kk);
      if kk == 2, sep = 'off';end
      sysmen(kk) = uimenu(sysmenu,'label',system(kk).name,...
         'call',['pplane8(''system'',',kkk,')'],...
         'separator',sep,'visible','off');
   end
   set(sysmenu,'user',system);
   ud.h.gallery = sysmenu;
   ud.flag = 0;
   ud.egg = (exist('EASTEREGG') ==2);

   % Record the handles in the User Data of the Set Up figure.

   set(ppset,'user',ud);
   hhhh = findobj(ppset,'type','uicontrol');
   set(hhhh,'units','normal')

   set(ppset,'visible','on','resize','on');
   set(get(ppset,'children'),'visible','on');
   set(get(sysmenu,'children'),'visible','on');
 %end
  set(0,'showhiddenhandles',hhsetup);

elseif strcmp(action,'savesyst')

   ppset = findobj('name','pplane8 Setup');
   type = input1;
   sud = get(ppset,'user');

   switch type
   case 'system'
      systems = get(sud.h.gallery,'user');

      newsyst = sud.c;
      fn = newsyst.name;
      if ~isempty(fn)
         fn(find(abs(fn)==32))='_';    % Replace spaces by underlines.
      end
      fn = [fn, '.pps'];
      comp = computer;
      switch  comp
      case 'PCWIN'
         filter = [sud.ppdir, '\',fn];
      case 'MAC2'
         filter = [sud.ppdir,':', fn];
      otherwise
         filter = [sud.ppdir,'/', fn];
      end
      [fname,pname] = uiputfile(filter,'Save the system as:');
      if fname == 0,return;end
      if ~strcmp(fname,fn)
         ll = length(fname);
         if (ll>4 & strcmp(fname(ll-3:ll),'.pps'))
            fn = fname;
         else
            fn = [fname, '.pps'];
         end
         newsyst.name = fn;
         sud.c.name = fn;
         set(ppset,'user',sud);
      end
      newsysts = newsyst;

   case 'gallery'
      systems = get(sud.h.gallery,'user');
      ll = length(systems);
      if ll == 0
    warndlg(['There are no systems to make up a gallery.'],'Warning');
    return
      end
      names = cell(ll,1);
      for j=1:ll
    names{j} = systems(j).name;
      end
      [sel,ok] = listdlg('PromptString','Select the systems',...
             'Name','Gallery selection',...
             'ListString',names);
      if isempty(sel)
    return
      else
    newsysts = systems(sel);
      end
      comp = computer;
      switch  comp
       case 'PCWIN'
    prompt = [sud.ppdir,'\*.ppg'];
       case 'MAC2'
    prompt = [sud.ppdir,':*.ppg'];
       otherwise
    prompt = [sud.ppdir,':/.ppg'];
      end
      [fname,pname] = uiputfile(prompt,'Save the gallery as:');
      ll = length(fname);
      if (ll>4 & strcmp(fname(ll-3:ll),'.ppg'))
    fn = fname;
      else
    fn = [fname, '.ppg'];
      end
      newsyst.name = fn(1:ll-4);
      sud.c.name = fn(1:ll-4);
      set(ppset,'user',sud);

   end  % switch type

   ll = length(newsysts);
   fid = fopen([pname fn],'w');
   ppstring = '%%%% PPLANE file %%%%';
   fprintf(fid,[ppstring,'\n']);
   for k = 1:ll
     fprintf(fid,'\n');
     nstr = newsysts(k).name;
     nstr = strrep(nstr,'''','''''');
     nstr = ['H.name = ''', nstr, ''';\n'];
     fprintf(fid,nstr);
     xname = newsysts(k).xvar;
     xnstr = ['H.xvar = ''', xname];
     xnstr = strrep(xnstr,'\','\\');
     xnstr = [xnstr, ''';\n'];
     fprintf(fid,xnstr);
     yname = newsysts(k).yvar;
     ynstr = ['H.yvar = ''', yname];
     ynstr = strrep(ynstr,'\','\\');
     ynstr = [ynstr, ''';\n'];
     fprintf(fid,ynstr);
     xder = newsysts(k).xder;
     xdstr = ['H.xder = ''', xder];
     xdstr = strrep(xdstr,'\','\\');
     xdstr = [xdstr, ''';\n'];
     fprintf(fid,xdstr);
     yder = newsysts(k).yder;
     ydstr = ['H.yder = ''', yder];
     ydstr = strrep(ydstr,'\','\\');
     ydstr = [ydstr, ''';\n'];
     fprintf(fid,ydstr);

     pname = strrep(newsysts(k).pname,'\','\\');
     pval = strrep(newsysts(k).pval,'\','\\');
     pnl = length(pname);
     pvl = length(pval);
     for kk = 1:6
       if kk <= pnl
     pns = pname{kk};
       else
     pns = '';
       end
       if kk <= pvl
     pvs = pval{kk};
       else
     pvs = '';
       end
       if kk == 1
     pnstr = ['H.pname = {''', pns, ''''];
     pvstr = ['H.pval = {''', pvs, ''''];
       else
     pnstr = [pnstr, ',''',pns, ''''];
```

```
     pvstr = [pvstr, ',''',pvs, ''''];
       end
     end
     pnstr = [pnstr, '};\n'];
     pvstr = [pvstr, '};\n'];


     fprintf(fid,pnstr);
     fprintf(fid,pvstr);
     ftstr = ['H.fieldtype = ''', newsysts(k).fieldtype];
     ftstr - strrep(ftstr,'\','\\');
     ftstr = [ftstr, ''';\n'];
     fprintf(fid,ftstr);
     nstr = ['H.npts = ', num2str(newsysts(k).npts),';\n'];
     fprintf(fid,nstr);
     wind = newsysts(k).wind;
     wstr = ['H.wind = [', num2str(wind),'];\n'];
     fprintf(fid,wstr);


   end
   fclose(fid);

elseif strcmp(action,'loadsyst')  % This loads either a system or a gallery.

   sud = get(gcf,'user');
   pos = get(gcf,'pos');
   wpos = [pos(1),pos(2)+pos(4)+20,300,20];
   waith = figure('pos',wpos,...
          'numb','off',...
          'vis','off',...
          'next','replace',...
          'menubar','none',...
          'resize','off',...
          'createfcn','');
   axes('pos',[0.01,0.01,0.98,0.98],...
    'vis','off');
   xp = [0 0 0 0];
   yp = [0 0 1 1];
   xl = [1 0 0 1 1];
   yl = [0 0 1 1 0];
   patchh = patch(xp,yp,'r','edgecolor','r','erase','none');
   lineh = line(xl,yl,'erase','none','color','k');
   type = input1;
   set(sud.h.gallery,'enable','off');
   if strcmp(type,'default')
     set(waith,'name','Loading the default gallery.','vis','on');
     set(findobj('tag','load default'),'enable','off');
     megall = sud.h.gallery;
     mh = get(megall,'children');
     add = findobj(megall,'tag','add system');
     mh(find(mh == add)) = [];
     delete(mh);
     newsysstruct = get(megall,'user');
     system = sud.system;
     ll = length(system);
```

```
  x = 1/(ll+2);
  xp = [xp(2),x,x,xp(2)];
  set(patchh,'xdata',xp);
  set(lineh,'xdata',xl);
  drawnow;
  sep = 'on';
  for kk = 1:length(system)
    kkk = num2str(kk);
    if kk ==2, sep = 'off';end
    uimenu(megall,'label',system(kk).name,...
       'call',['pplane8(''system'',',kkk,')'],...
       'separator',sep);
  end % for
  set(megall,'user',system);
else
  comp = computer;
  switch  comp
   case 'PCWIN'
    prompt = [sud.ppdir,'\'];
   case 'MAC2'
    prompt = [sud.ppdir,':'];
   otherwise
    prompt = [sud.ppdir,'/'];
  end

  if strcmp(type,'system')
    prompt = [prompt,'*.pps'];
    [fname,pname] = uigetfile(prompt,'Select a system to load.');
  elseif strcmp(type,'gallery')
    prompt = [prompt,'*.ppg'];
    [fname,pname] = uigetfile(prompt,'Select a gallery to load.');
  end  % if strcmp

  if fname == 0
    delete(waith);
    set(sud.h.gallery,'enable','on');
    return;
  end
  set(waith,'name',['Loading ',fname],'vis','on');
  fid = fopen([pname fname],'r');
  sline = fgetl(fid);
  if strcmp(sline,'%% PPLANE file %%')
    date = 'new';
  else
    date = 'old';
  end
  newsysts = {};
  switch date
   case 'old'
    newsysts{1} = sline;
    kk = 1;
   case 'new'
    kk = 0;
  end
  while ~feof(fid)
    kk = kk + 1;
    newsysts{kk} = fgetl(fid);
  end
  fclose(fid);
  newsysts = newsysts([1:kk]);
  false = 0;
  switch date
   case 'old'
    if mod(kk,19)
  false = 1;
    end
   case 'new'
    if mod(kk,11)
  false = 1;
    end
  end %switch date
  if false
    if strcmp(type,'system')
  warndlg(['The file ',fname, ' does not define a proper system.'],...
      'Warning');
    elseif strcmp(type,'gallery')
  warndlg(['The file ',fname, ' does not define a proper gallery.'],...
      'Warning');
    end
    set(sud.h.gallery,'enable','on');
    delete(waith);
    return
  end %if false
  switch date
   case 'old'
    x = 19/(kk+38);
    xp = [xp(2),x,x,xp(2)];
    set(patchh,'xdata',xp);
    set(lineh,'xdata',xl);
    drawnow;
    nnn = kk/19;
    flds = fieldnames(sud.c);
    flds=flds(:);
    for j = 0:(nnn-1)
      newsystemp = newsysts([(j*19+1):(j+1)*19]);
  newsyst.name = newsystemp{1};
  newsyst.xvar = newsystemp{2};
  newsyst.yvar = newsystemp{3};
  newsyst.xder = newsystemp{4};
  newsyst.yder = newsystemp{5};
  newsyst.pname = {newsystemp{6}, newsystemp{7},...
           newsystemp{8}, newsystemp{9},'',''};
  newsyst.pval = {newsystemp{10},...
          newsystemp{11},...
          newsystemp{12},...
          newsystemp{13},'',''};
  newsyst.fieldtype = newsystemp{14};
  newsyst.npts = str2num(newsystemp{15});
  wind = newsystemp(16:19);
      newsyst.wind = [str2num(wind{1}),str2num(wind{2}),...
          str2num(wind{3}),str2num(wind{4})];
  newsysstruct(j+1) = newsyst;
    end  % for j = 0:(nnn-1)
```

```
   case 'new'
    x = 11/(kk+22);
    xp = [xp(2),x,x,xp(2)];
    set(patchh,'xdata',xp);
    set(lineh,'xdata',xl);
    drawnow;
    nnn = kk/11;
    for j = 1:nnn
  for k = 2:11;
    eval(newsysts{(j-1)*11+k});
  end
  newsysstruct(j) = H;
    end

  end %switch date
end  % if strcmp(type,'default') & else
nnn = length(newsysstruct);
ignoresyst = {};
for j = 1:nnn
   x = (j+1)/(nnn+2);
   xp = [xp(2),x,x,xp(2)];
   set(patchh,'xdata',xp);
   set(lineh,'xdata',xl);
   drawnow;
   newsyst = newsysstruct(j);
   sname = newsyst.name;
   sname(find(abs(sname) == 95)) = ' '; % Replace underscores with spaces.
   newsyst.name = sname;
   ignore = pplane8('addgall',newsyst);
   if ignore == -1;
      ignoresyst{length(ignoresyst)+1} = sname;
   end
end % for j = 1:nnn
l = length(ignoresyst);
if l  % There was at least one system which was a dup with a
   % different name.
   if l == 1
      message = {['The system ',ignoresyst{1},'" duplicates a ',...
               'system already in the gallery and was not added.']};
   else
      message = 'The systems ';
      for k = 1:(l-1)
         message = [message,'"',ignoresyst{k},'", '];
      end
      message = {[message,'and "',ignoresyst{l},'" duplicate ',...
               'systems already in the gallery and were not added.']};
   end % if l == 1 & else
   helpdlg(message,'Ignored systems');
end  % if l
if strcmp(type,'system') % Added a system.
   if ignore > 0 % The system was ignored.
      kk = ignore;
   else
      systems = get(sud.h.gallery,'user');
      kk = length(systems);
   end
   pplane8('system',kk);
   end
   if strcmp('type','default')
      pplane8('system',1);
   end
   set(sud.h.gallery,'enable','on');
   x = 1;
   xp = [xp(2),x,x,xp(2)];
   set(patchh,'xdata',xp);
   set(lineh,'xdata',xl);
   drawnow;
   delete(waith);

elseif strcmp(action,'addgall')

  output = 0;
  ppset = findobj('name','pplane8 Setup');
  sud = get(ppset,'user');
  if nargin < 2    % We are adding the current system.

    syst = sud.c;
    snstr = 'Provide a name for this system.';
    sname = inputdlg(snstr,'System name',1,{syst.name});
    if isempty(sname),return;end
    sname = sname{1};
    if ~strcmp(sname,syst.name)
      sud.c.name = sname;
      set(ppset,'user',sud);
      syst.name = sname;
    end

  else  % We have a system coming from a file.
    syst = input1;
    sname = syst.name;
  end
  pnl = length(syst.pname);
  for kk = (pnl+1):6
    syst.pname{kk} = '';
  end
  pvl = length(syst.pval);
  for kk = (pvl+1):6
    syst.pval{kk} = '';
  end

  systems = get(sud.h.gallery,'user');
  ll = length(systems);
  kk = 1;
  while ((kk<=ll) & (~strcmp(sname,systems(kk).name)))
    kk = kk + 1;
  end
  nameflag = (kk<=ll);
  ssyst = rmfield(syst,'name');
  kk = 1;
  while ((kk<=ll) & (~isequal(ssyst,rmfield(systems(kk),'name'))))
    kk = kk + 1;
  end
  systflag = 2*(kk<=ll);
```

```
flag = nameflag + systflag;
switch flag
 case 1  % Same name but different system.
  mh = findobj(sud.h.gallery,'label',sname);
  prompt = {['The system "',sname,'", which you wish to ',...
         'add to the gallery has ',...
             'the same name as a different system ',...
         'already in the gallery.  Please ',...
             'specify the name for the newly added system.'],...
        'Specify the name for the old system.'};
  title = 'Two systems with the same name';
  lineno = 1;
  defans = {sname,sname};
  answer = inputdlg(prompt,title,lineno,defans);
  if isempty(answer),return,end
  sname = answer{1};
  systems(kk).name = answer{2};
  set(mh,'label',answer{2});
  output = kk;
 case 2  % Two names for the same system.
  oldname = systems(kk).name;
  mh = findobj(sud.h.gallery,'label',oldname);

  prompt = {['The system "',sname,'", which you wish to add ',...
             'to the gallery is the same as a system which is ',...
             'already in the gallery with the name "',oldname,'".  ',...
             'Please specify which name you wish to use.']};
  title = 'Two names for the same system.';
  lineno = 1;
  defans = {oldname};
  answer = inputdlg(prompt,title,lineno,defans);
  if isempty(answer),return,end
  systems(kk).name = answer{1};
  set(mh,'label',answer{1});
  output = kk;
 case 3 % Systems and names the same.
  output = -1;
 otherwise
end  % switch
set(sud.h.gallery,'user',systems);
syst.name = sname;
if flag <=1
  switch ll
   case 0
    systems = syst;
    sepstr = 'on';
   case 10
    systems(11) = syst;
    if strcmp(systems(10).name,'square limit set')
  sepstr = 'on';
    else
  sepstr = 'off';
    end
   otherwise
    systems(ll+1) = syst;
    sepstr = 'off';
  end
```

```
    kkk = num2str(ll+1);
    newmenu = uimenu(sud.h.gallery,'label',sname,...
             'call',['pplane8(''system'',',kkk,')'],...
             'separator',sepstr);
    set(findobj('tag','savegal'),'enable','on');
  end
  set(sud.h.gallery,'user',systems);

elseif strcmp(action,'system')

   ppset = findobj('name','pplane8 Setup');
   ud = get(ppset,'user');
   kk = input1;
   if isstr(kk)
      kk = str2num(input1);
   end
   system = get(ud.h.gallery,'user');
   syst = system(kk);
   xname = syst.xvar;
   yname = syst.yvar;
   set(ud.h.xvar,'string',xname);
   set(ud.h.yvar,'string',yname);
   set(ud.h.xder,'string',syst.xder);
   set(ud.h.yder,'string',syst.yder);
   pname = syst.pname;
   pval = syst.pval;
   pnl = length(pname);
   pvl = length(pval);
   for kk = 1:6
     if kk <= pnl
       set(ud.h.pname(kk),'string',pname{kk});
     else
       set(ud.h.pname(kk),'string','');
       syst.pname{kk} = '';
     end
     if kk <= pvl
       set(ud.h.pval(kk),'string',pval{kk});
     else
       set(ud.h.pval(kk),'string','');
       syst.pval{kk} = '';
     end
   end
   ud.o = syst;
   ud.c = syst;
   set(ud.h.twind(1),'string',['The minimum value of ',xname,' = ']);
   set(ud.h.twind(2),'string',['The maximum value of ',xname,' = ']);
   set(ud.h.twind(3),'string',['The minimum value of ',yname,' = ']);
   set(ud.h.twind(4),'string',['The maximum value of ',yname,' = ']);
   for kk = 1:4
     set(ud.h.wind(kk),'string',num2str(syst.wind(kk)));
   end
   set(ud.h.npts,'string',num2str(syst.npts));

   switch syst.fieldtype
   case 'nullclines'
      rval = [1 0 0 0];
   case 'lines'
      rval = [0 2 0 0];
   case 'arrows'
      rval = [0 0 3 0];
   case 'none'
      rval = [0 0 0 4];
   otherwise
      error(['Unknown fieldtype ',ud.o.fieldtype,'.'])
   end

   for i=1:4
      set(ud.h.rad(i),'value',rval(i));
   end
   ud.flag = 0;
   set(ppset,'user',ud);

elseif strcmp(action,'revert')

   ud = get(gcf,'user');
   ud.c = ud.o;
   syst = ud.o;
   xname = syst.xvar;
   yname = syst.yvar;
   set(ud.h.xvar,'string',xname);
   set(ud.h.yvar,'string',yname);
   set(ud.h.xder,'string',syst.xder);
   set(ud.h.yder,'string',syst.yder);
   pname = syst.pname;
   pval = syst.pval;
   pnl = length(pname);
   pvl = length(pval);
   for kk = 1:6
     if kk <= pnl
       nstr = pname(kk);
     else
       nstr = '';
     end
     if kk <= pvl
       vstr = pval(kk);
     else
       vstr = '';
     end
     set(ud.h.pname(kk),'string',nstr);
     set(ud.h.pval(kk),'string',vstr);
   end
   set(ud.h.twind(1),'string',['The minimum value of ',xname,' = ']);
   set(ud.h.twind(2),'string',['The maximum value of ',xname,' = ']);
   set(ud.h.twind(3),'string',['The minimum value of ',yname,' = ']);
   set(ud.h.twind(4),'string',['The maximum value of ',yname,' = ']);
   for kk = 1:4
     set(ud.h.wind(kk),'string',num2str(syst.wind(kk)));
   end
   set(ud.h.npts,'string',num2str(syst.npts));

   switch syst.fieldtype
    case 'lines'
```

```
     rval(1) = 1;
     rval(2) = 0;rval(3) = 0;
    case 'arrows'
     rval(1) = 0;rval(2) = 2;rval(3) = 0;
    case 'none'
     rval(1) = 0;rval(2) = 0;rval(3) = 3;
    otherwise
     error(['Unknown fieldtype ',ud.o.fieldtype,'.'])
   end

   for i=1:3
     set(ud.h.rad(i),'value',rval(i));
   end
   set(gcf,'user',ud);

elseif strcmp(action,'proceed')

   % Proceed connects Setup with the Display window.

   ppset = gcf;
   sud = get(ppset,'user');
   sud.o = sud.c;
   set(ppset,'user',sud);
   he = findobj('name','pplane8 Equilibrium point data');
   hl = findobj('name','pplane8 Linearization');
   close([he;hl]);
   % set([he;hl],'vis','off');

   % Some error checking that has to be done no matter what.

   WINvect = sud.c.wind;
   if any(isnan(WINvect))
      sud.flag = 0;
      set(ppset,'user',sud);
      errmsg = ['One of the entries defining the display window ',...
            'is not a number.'];
      fprintf('\a')
      errordlg(errmsg,'PPLANE error','on');
      return
   end
   xstr = sud.c.xvar;
   if isempty(xstr)
      sud.flag = 0;
      set(ppset,'user',sud);
      errmsg = 'The first dependent variable needs a name.';
      fprintf('\a')
      errordlg(errmsg,'PPLANE error','on');
      return
   end
   ystr = sud.c.yvar;
   if isempty(ystr)
      sud.flag = 0;
      set(ppset,'user',sud);
      errmsg = 'The second dependent variable needs a name.';
      fprintf('\a')
      errordlg(errmsg,'PPLANE error','on');
      return
   end
   if WINvect(2)<= WINvect(1)
      sud.flag = 0;
      set(ppset,'user',sud);
      errmsg = ['The minimum value of ', xstr,...
            ' must be smaller than the maximum value.'];
      fprintf('\a')
      errordlg(errmsg,'PPLANE error','on');
      return
   end
   if WINvect(4)<= WINvect(3)
      sud.flag = 0;
      set(ppset,'user',sud);
      errmsg = ['The minimum value of ', ystr,...
            ' must be smaller than the maximum value.'];
      fprintf('\a')
      errordlg(errmsg,'PPLANE error','on');
      return
   end
   if isnan(sud.c.npts)
      sud.flag = 0;
      set(ppset,'user',sud);
      errmsg = 'The entry for the number of field points is not a number.';
      fprintf('\a')
      errordlg(errmsg,'PPLANE error','on');
      return
   end

   %  sud.flag = 0 if this is the first time through for this equation,
   %  sud.flag = 1 if only the window dimensions or the field data
   %  have been changed.

   % If sud.flag == 1 we only have to update things.

   if (sud.flag == 1)
     Arrflag = sud.c.fieldtype;
     NumbFPts = sud.c.npts;
     ppdisp = findobj('name','pplane8 Display');
     dud = get(ppdisp,'user');
     aud = get(dud.axes,'user');
     wind = sud.c.wind(:);
     if (~all(wind == dud.syst.wind(:)))
       dwind = [wind(1); wind(3); -wind(2); -wind(4)];
       aud.DY = [wind(2)-wind(1); wind(4)-wind(3)];
       aud.cwind = dwind - dud.settings.magn*[aud.DY;aud.DY];
       set(dud.axes,'user',aud);
     end

     arr = dud.arr;
     menull = findobj('tag','null');
     switch Arrflag
      case 'nullclines'
%        set([arr.hx;arr.hy;arr.barrows],'vis','on');
       set([arr.hx;arr.hy],'vis','on');
       set([arr.lines;arr.arrows],'vis','off');
       set(menull,'enable','on','label','Hide nullclines.');
      case 'lines'
       set(arr.lines,'vis','on');
       set([arr.hx;arr.hy;arr.arrows;arr.barrows],'vis','off');
       set(menull,'enable','on','label','Show nullclines.');
      case 'arrows'
       set(arr.arrows,'vis','on');
       set([arr.lines;arr.hx;arr.hy;arr.barrows],'vis','off');
       set(menull,'enable','on','label','Show nullclines.');
      otherwise
       set([arr.hx;arr.hy;arr.lines;arr.arrows;arr.barrows],'vis','off');
       set(menull,'enable','on','label','Show nullclines.');
     end
     dud.syst.fieldtype = Arrflag;
     set(ppdisp,'user',dud);

     if ( (NumbFPts ~= dud.syst.npts) | (any(WINvect ~= dud.syst.wind) ) )
       dud.syst.wind = WINvect;
       dud.syst.npts = NumbFPts;
       set(ppdisp,'user',dud);
       pplane8('dirfield',ppdisp);
     end
     figure(ppdisp);
   else
     sud.flag = 1;
     set(ppset,'user',sud);

     sud = get(ppset,'user');
     %  WINvect = sud.c.wind;

     Xname = sud.c.xvar;
     Yname = sud.c.yvar;
     xderivstr = sud.c.xder;
     yderivstr = sud.c.yder;
     pname = sud.c.pname;
     parav = sud.c.pval;

     % Convert the parameters to their current values.  First remove the
     % blanks.  Also remove the periods inserted by users attempting to
     % make the function array smart.

     xderivstr(find(abs(xderivstr)==32))=[];
     l=length(xderivstr);
     for ( k = fliplr(findstr('.',xderivstr)))
       if (find('*/^' == xderivstr(k+1)))
     xderivstr = [xderivstr(1:k-1), xderivstr(k+1:l)]
       end
       l=l-1;
     end

     yderivstr(find(abs(yderivstr)==32))=[];
     l=length(yderivstr);
     for ( k = fliplr(findstr('.',yderivstr)))
       if (find('*/^' == yderivstr(k+1)))
     yderivstr = [yderivstr(1:k-1), yderivstr(k+1:l)];
       end
       l=l-1;
     end

     for kk = 1:6
       pval = parav{kk};
       if ~isempty(pval)
     pabs = abs(pval);
     kkk = find(pabs==32);
     pval(kkk) = [];
     l = length(pval);
     for ( k = fliplr(findstr('.',pval)))
       if (find('*/^' == pval(k+1)))
         pval = [pval(1:k-1), pval(k+1:l)];
       end
       l=l-1;
     end
     parav{kk} = pval;
       end
     end

     % Build strings for the title.
     txderstr = xderivstr;
     tyderstr = yderivstr;

     kxder = find(abs(txderstr)==42);
     txderstr(kxder)=' '*ones(size(kxder));
     txderstr = strrep(txderstr,'-',' - ');
     txderstr = strrep(txderstr,'+',' + ');
     if (abs(txderstr(1)) == 32)
       txderstr = txderstr(2:length(txderstr));
     end

     kxder = find(abs(tyderstr)==42);
     tyderstr(kxder)=' '*ones(size(kxder));
     tyderstr = strrep(tyderstr,'-',' - ');
     tyderstr = strrep(tyderstr,'+',' + ');
     if (abs(tyderstr(1)) == 32)
       tyderstr = tyderstr(2:length(tyderstr));
     end

     tstr1 = [Xname,' '' = ', txderstr];
     tstr2 = [Yname,' '' = ', tyderstr];
     tstr = str2mat(tstr1,tstr2);
     dud.tstr = tstr;
     pstr1 = {' ';' '};
     pstr2 = {' ';' '};
     pstr3 = {' ';' '};

     pstring = cell(6,1);
     for kk = 1:6
       if ~isempty(parav{kk})
     tpstr = parav{kk};
     kxder = find(abs(tpstr)==42);  % Get rid of *s
     tpstr(kxder)=' '*ones(size(kxder));
     tpstr = strrep(tpstr,'-',' - ');  % Extra spaces
     tpstr = strrep(tpstr,'+',' + ');
```

```
if (abs(tpstr(1)) == 32)  % Get rid of starting space
  tpstr = tpstr(2:length(tpstr));
end
pstring{kk} = [pname{kk},' = ', tpstr];
  else
% pstring{kk} = [pname{kk},' = ',parav{kk}];
pstring{kk} = [pname{kk},' = '];
  end
end

% Get ready to do some error trapping.

SS = warning;
warning off
XxXxXx = WINvect(1) + rand*(WINvect(2)-WINvect(1));
YyYyYy = WINvect(3) + rand*(WINvect(4)-WINvect(3));
err = 0;

% Now we remove the backslashes (\) used to get Greek into the
% labels.
txname = Xname;
tyname = Yname;
xderivstr(find(abs(xderivstr)==92))=[];
yderivstr(find(abs(yderivstr)==92))=[];
Xname(find(abs(Xname)==92))=[];
Yname(find(abs(Yname)==92))=[];

eval([Xname,'=XxXxXx;'],'err = 1;');
if err
  sud.flag = 0;
  set(ppset,'user',sud);
  errmsg = ['"',xstr, '" is not a valid variable name in MATLAB.'];
  fprintf('\a')
  errordlg(errmsg,'PPLANE error','on');
  return
end
err = 0;
eval([Yname,'=YyYyYy;'],'err = 1;');
if err
  sud.flag = 0;
  set(ppset,'user',sud);
  errmsg = ['"',ystr, '" is not a valid variable name in MATLAB.'];
  fprintf('\a')
  errordlg(errmsg,'PPLANE error','on');
  return
end


% Replace the parameters/expressions with their values.

pflag = zeros(1,6);
perr = [];
for kk = 1:6
  if ~isempty(pname{kk})
pn = pname{kk};
```

```
pv = parav{kk};
if isempty(pv)
  perr = [perr, sud.h.pval(kk)];
else
  if isempty(str2num(pv)) % This is an expression.
    tpv = pv;
    tpv(find(abs(tpv)==92))=[];
    err = 0; pval = '';
    eval(['pval = ',tpv,';'],'err=1;');
    if (err | isempty(pval))
      errmsg = ['The expression for ',pn,' is not valid.'];
      fprintf('\a')
      errordlg(errmsg,'pplane8 error','on');
      warning(SS)
      return
    end
  end
  xxderivstr = pplane8('paraeval',pn,pv,xderivstr);
  yyderivstr = pplane8('paraeval',pn,pv,yderivstr);
  if (~strcmp(xxderivstr,xderivstr) | ~strcmp(yyderivstr,yderivstr) )
    pflag(kk) = 1;
    xderivstr = xxderivstr;
    yderivstr = yyderivstr;
  end
end
  end
end

% We have to make the derivative strings array smart.

l = length(xderivstr);
for (k=fliplr(find((xderivstr=='^')|(xderivstr=='*')|(xderivstr=='/'))))
  xderivstr = [xderivstr(1:k-1) '.' xderivstr(k:l)];
  l = l+1;
end

l = length(yderivstr);
for (k=fliplr(find((yderivstr=='^')|(yderivstr=='*')|(yderivstr=='/'))))
  yderivstr = [yderivstr(1:k-1) '.' yderivstr(k:l)];
  l = l+1;
end

% Some more error trapping.

err = 0;res = 1;
eval(['res = ',xderivstr, ';'],'err = 1;');
if err | isempty(res)
   if isempty(perr)
errmsg = ['The first differential equation ',...
      'is not entered correctly.'];
  else
errstr1 = ['The first differential equation ',...
       'does not evaluate correctly.'];
errstr2 = ['At least one of the parameter values is not ',...
       'a number.'];
errmsg = str2mat(errstr1,errstr2);
```

```
     perr
     set(perr,'string','?');
       end
       sud.flag = 0;
       set(ppset,'user',sud);
       fprintf('\a')
       errordlg(errmsg,'PPLANE error','on');
       return;
     end

     err = 0;res = 1;
     eval(['res = ',yderivstr, ';'],'err = 1;');
     if err | isempty(res)
       if isempty(perr)
     errmsg = ['The second differential equation ',...
           'is not entered correctly.'];
       else
     errstr1 = ['The second differential equation ',...
            'does not evaluate correctly.'];
     errstr2 = ['At least one of the parameter values is not ',...
            'a number.'];
     errmsg = str2mat(errstr1,errstr2);
     set(perr,'string','?');
       end
       sud.flag = 0;
       set(ppset,'user',sud);
       fprintf('\a')
       errordlg(errmsg,'PPLANE error','on');
       return;
     end

     % If an old function m-file exists delete it, and then build a new one.

     %    if (~strcmp(dfcn,'') & exist(dfcn)==2) delete([dfcn,'.m']);end
     tee = clock;
     tee = ceil(tee(6)*100);
     dfcn=['pptp',num2str(tee)];
     fcnstr = ['function YYyYypr = ',dfcn,'(t,YyYy)\n\n'];
     commstr = '%%%% Created by pplane8\n\n';
     varstr = [Xname,' = YyYy(1,:);', Yname,' = YyYy(2,:);\n\n'];
     lenstr = ['l = length(YyYy(1,:));\n'];
     derstr1 = ['XxXxxpr = ', xderivstr,';\n'];
     derstr2 = ['if (length(XxXxxpr) < l) XxXxxpr =
XxXxxpr*ones(1,l);end\n'];
     derstr3 = ['YyYyypr = ', yderivstr,';\n'];
     derstr4 = ['if (length(YyYyypr) < l) YyYyypr =
YyYyypr*ones(1,l);end\n'];
     derstr5 = 'YYyYypr = [XxXxxpr;YyYyypr];\n';
     ppf = fopen([tempdir,dfcn,'.m'],'w');

     fprintf(ppf,fcnstr);
     fprintf(ppf,commstr);
     fprintf(ppf,varstr);
     fprintf(ppf,lenstr);
     fprintf(ppf,derstr1);
     fprintf(ppf,derstr2);
```

```
 fprintf(ppf,derstr3);
 fprintf(ppf,derstr4);
 fprintf(ppf,derstr5);
 fclose(ppf);



 % Find pplane8 Display if it exists.
 % If pplane8 Display exists, update it.  If it does not build it.

 ppdisp = findobj('name','pplane8 Display');
 if (~isempty(ppdisp))
   figure(ppdisp);
   dud = get(ppdisp,'user');
   dud.syst = sud.c;
   dud.settings = sud.settings;
   dfcnn = dud.function;
 if  exist(dfcnn)==2
   delete([tempdir,dfcnn,'.m']);
 end
 xmstr = [txname,' vs. t'];
 ymstr = [tyname,' vs. t'];
 set(dud.menu(3),'label',xmstr);
 set(dud.menu(5),'label',ymstr);
 menull = findobj('tag','null');
 if ~isempty(menull)
   delete(get(menull,'user'));
   set(menull,'user',[]);
 end
 else
   ppdisp = figure('name','pplane8 Display',...
           'numb','off',...
           'interrupt','on',...
           'visible','off',...
           'tag','pplane8');
   pplane8('figdefault',ppdisp);
   dud = get(ppdisp,'user');
   dud.syst = sud.c;
   switch dud.syst.name
case 'pendulum'
 dud.level = 'omega^2 - 2*cos(theta)';
otherwise
 dud.level = ' ';
   end
   dud.settings = sud.settings;
   dud.egg = sud.egg;
   dud.noticeflag = 1;
   dud.contours = zeros(0,1);
   fs = dud.fontsize;
   ssize = dud.ssize;
   r = ssize/10;
   ppaxw = 437*1.2; % Default axes width
   ppaxh = 315*1.2; % Default axes height
   ppaxl = 45*1.2;  % Default axes left
   buttw = 40*1.2;     % Default button width
   titleh = 45;    % Default title height.  This is changed later.
```

```
  nframeh = 70;   % Default notice frame height
  ppaxb = 4+nframeh+35;
  bottomedge = 38;
  ppdh = bottomedge + nframeh + ppaxh + titleh;
  uni = get(0,'units');
  set(0,'units','pixels');
  ss = get(0,'screensize');
  set(0,'units',uni);
  sw = ss(3);sh = ss(4);
  bottom = 10;
  if r*ppdh > sh -bottom -35;
r = (sh-bottom-35)/ppdh;
fs = 10*r;
lw = 0.5*r;
set(gcf,'defaultaxesfontsize',fs,'defaultaxeslinewidth',lw);
set(gcf,'defaulttextfontsize',fs);
set(gcf,'defaultlinelinewidth',lw);
set(gcf,'defaultuicontrolfontsize',fs*0.9);
  end

  % Set up the bulletin window.

  nframe = uicontrol('style','frame','visible','on');
  nstr = {'More';'than';'five';'lines';'of text'};
  dud.notice = uicontrol('style','text',...
             'horiz','left',...
             'string',nstr,'visible','on');
  ext = get(dud.notice,'extent');
  nframeh = ext(4)+2;

  titleh = r*titleh;
  ppaxl = r*ppaxl;
  ppaxw = r*ppaxw;
  ppaxh = r*ppaxh;
  ppaxb = nframeh+r*bottomedge;
  buttw = r*buttw;
  butth = fs+10*r;
  buttl = ppaxl + ppaxw + 5;
  buttsep = (ppaxh - butth)/2;
  % Set up the coordinate display

  cstr = '(0.99999999, 0,99999999)';
  dud.ccwind = uicontrol('style','text',...
             'horiz','left',...
             'string',cstr,...
             'visible','on');
  cext = get(dud.ccwind,'extent');
  ccwindtxt = uicontrol('style','text',...
            'horiz','left',...
            'string','Cursor position: ',...
            'visible','on');
  cwh = cext(4);
  cww = cext(3);


  % Set up the plot axes.
```

```
dud.axes = axes('units','pix',...
        'position',[ppaxl,ppaxb,ppaxw,ppaxh],...
        'next','add',...
        'box','on',...
        'interrupt','on',...
        'xgrid','on',...
        'ygrid','on',...
        'drawmode','fast',...
        'visible','off',...
        'tag','display axes');

% Set up the title.

dud.title.axes = axes('box','off','xlim',[0 1],'ylim',[0 1],...
          'units','pix','vis','off',...
          'xtick',[-1],'ytick',[-1],...
          'xticklabel','','yticklabel','');

dud.title.eq = text(0.01,0.5,' ','vert','middle');
dud.title.p1 = text(0.75,0.5,' ','vert','middle');
dud.title.p2 = text(0.65,0.5,' ','vert','middle');
dud.title.p3 = text(0.55,0.5,' ','vert','middle');
tstr = {'x_2';'y^2'};
set(dud.title.eq,'string',tstr,'units','pix');
ext = get(dud.title.eq ,'extent');
titleh = ext(4)+15*r;
set(dud.title.eq,'units','data');
taxpos = [ppaxl,ppaxb+ppaxh,ppaxw,titleh];
set(dud.title.axes,'pos',taxpos,'color',get(gcf,'color'));

% Finish the positions.

ppdw = buttl + buttw +5;
ppdh = ppaxb+ppaxh+titleh;
set(nframe,'pos',[10,1,ppdw-20,nframeh]);
set(dud.notice,'pos',[15,1,ppdw-30,nframeh-2],...
       'string',{' ';' ';' ';' ';' '});

ctext = get(ccwindtxt,'extent');
cc1pos = [ppaxl,2+nframeh,ctext(3),cwh];
cc2pos = [ppaxl+ctext(3),2+nframeh,cww,cwh];
set(ccwindtxt,'pos',cc1pos);
set(dud.ccwind,'pos',cc2pos,...
       'string','');

ppdleft = max((sw-ppdw)/2,sw-ppdw-60);
ppdbot = sh - ppdh - 35;
ppdpos = [ppdleft,ppdbot,ppdw,ppdh];
set(ppdisp,'resize','on');
set(ppdisp,'pos',ppdpos);

Arrflag = sud.c.fieldtype;

% Set up the buttons
```

```
       stopstr = 'aud = get(gca,''user'');aud.stop =
4;set(gca,''user'',aud);';

       dbutt(1) = uicontrol('style','push',...
                'pos',[buttl,ppaxb+2*buttsep,buttw,butth],...
                'string','Stop','call',stopstr,...
                'vis','off','tag','stop');

       dbutt(2) = uicontrol('style','push',...
                'pos',[buttl,ppaxb,buttw,butth],...
                'string','Quit',...
                'call','pplane8(''quit'')','visible','off');

       dbutt(3) = uicontrol('style','push',...
                'pos',[buttl,ppaxb+buttsep,buttw,butth],...
                'string','Print',...
                'call','pplane8(''print'')','visible','off');

       dud.butt = dbutt;

       % Menus and Toolbar

       hhsetup = get(0,'showhiddenhandles');
       set(0,'showhiddenhandles','on');

       % Configure the Toolbar.

       fixtb = ['set(gcbo,''state'',''off'');'];

       set(ppdisp,'ToolBar','none');

       % Menus

       tmenu = findobj(ppdisp,'label','&Tools');
       delete(tmenu);

       % Insert menu

       imenu = findobj(gcf,'label','&Insert');
       inschild = get(imenu,'child');
       legitem = findobj(inschild,'label','&Legend');
       colitem = findobj(inschild,'label','&Colorbar');
       delete([legitem,colitem]);

       % File menu

       fmenu = findobj(ppdisp,'label','&File',...
               'pos',1);
       delete(findobj(fmenu,'label','&New Figure'));
       delete(findobj(fmenu,'label','&Open...'));
       delete(findobj(fmenu,'label','&Close'));
       set(findobj(fmenu,'label','&Save'),...
       'pos',1,'separator','off');
       set(findobj(fmenu,'label','Save &As...'),...
       'pos',2);
       set(findobj(fmenu,'label','&Export...'),...
       'pos',3);
       delete(findobj(fmenu,'label','Pre&ferences...'));
       set(findobj(fmenu,'label','Pa&ge Setup...'),'pos',4);
       set(findobj(fmenu,'label','Print Set&up...'),'pos',5);
       set(findobj(fmenu,'label','Print Pre&view...'),'pos',6);
       set(findobj(fmenu,'label','&Print...'),'pos',7);
       merestart = uimenu(fmenu,'label',...
              'Restart pplane8',...
              'call','pplane8(''restart'')',...
              'separator','on');
       mequit = uimenu(fmenu,'label','Quit pplane8',...
               'call','pplane8(''quit'')','separator','off');

       % Edit menu

       emenu = findobj(ppdisp,'label','&Edit',...
               'pos',2);
       menu(2) = uimenu(emenu,'label','Zoom in.',...
            'call','pplane8(''zoomin'')',...
            'pos',1);

       zsqmenu = uimenu(emenu,'label','Zoom in square.',...
            'call','pplane8(''zoominsq'')',...
            'pos',2);

       zbmenu = uimenu(emenu,'label','Zoom back.','call',...
               'pplane8(''zoomback'')',...
               'enable','off','tag','zbmenu',...
               'pos',3);

       medallsol = uimenu(emenu,'label','Erase all solutions.',...
              'call','pplane8(''dallsol'')',...
              'separator','on','pos',4);

       medallep = uimenu(emenu,'label','Erase all equilibrium points.',...
             'call','pplane8(''dallep'')',...
             'separator','off',...
             'pos',5);

       medallics = uimenu(emenu,'label','Erase all marked initial
points.',...
             'call','pplane8(''dallics'')',...
             'separator','off',...
             'pos',6);

       medalllev = uimenu(emenu,'label','Erase all level curves.',...
              'call','pplane8(''dalllev'')',...
              'separator','off',...
              'pos',7);

       medall = uimenu(emenu,'label','Erase all graphics objects.',...
               'call','pplane8(''dall'')',...
```

```
        'separator','off',...
        'pos',8);

medel = uimenu(emenu,'label','Delete a graphics object.',...
       'call','pplane8(''delete'')',...
       'visible','on',...
       'pos',9);

menutext = uimenu(emenu,...
      'label','Enter text on the Display Window.',...
      'call','pplane8(''text'')',...
      'separator','on',...
      'pos',10);
set(findobj(emenu,'label','&Undo'),'separator','on',...
      'pos',11);
set(findobj(emenu,'label','Cu&t'),'pos',12);
set(findobj(emenu,'label','&Copy'),'pos',13);
set(findobj(emenu,'label','&Paste'),'pos',14);
set(findobj(emenu,'label','Clea&r'),'pos',15);
set(findobj(emenu,'label','&Select All'),'pos',16);
set(findobj(emenu,'label','Copy &Figure'),'pos',17);
set(findobj(emenu,'label','Copy &Options'),'pos',18);
set(findobj(emenu,'label','F&igure Properties'),'pos',19);
set(findobj(emenu,'label','&Axes Properties'),'pos',20);
set(findobj(emenu,'label','C&urrent Object Properties'),'pos',21);

% Graph menu

megraph  = uimenu('label','Graph','visible','off','pos',4);
menu(3) = uimenu(megraph,'label',[txname,' vs. t'],...
     'call','pplane8(''plotxy'',1)');

menu(5) = uimenu(megraph,'label',[tyname,' vs. t'],...
     'call','pplane8(''plotxy'',2)');

meplot3 = uimenu(megraph,'label','Both',...
     'call','pplane8(''plotxy'',3)');

meplot4 = uimenu(megraph,'label','3 D',...
     'call','pplane8(''plotxy'',4)');

meplot5 = uimenu(megraph,'label','Composite',...
     'call','pplane8(''plotxy'',5)');

% Solutions menu

solmenu = uimenu('label','Solutions','pos',3);
menukey = uimenu(solmenu,'label','Keyboard input.','call',...
     'pplane8(''kbd'')');

mesev   = uimenu(solmenu,'label','Plot several solutions.',...
     'call','pplane8(''several'')');

meeqpt  = uimenu(solmenu,'label','Find an equilibrium point.',...
     'call','pplane8(''eqpt'')','separator','on');

menu(4) = uimenu(solmenu,...
     'label','List computed equilibrium points.',...
     'call','pplane8(''eqptlist'')');

mestunst= uimenu(solmenu,...
     'label','Plot stable and unstable orbits.',...
     'call','pplane8(''stunst'')','interrupt','on');

meperiod = uimenu(solmenu,...
      'label', 'Find a nearly closed orbit');
periodstr = ['ud = get(gcf,''user'');',...
     'me = gcbo;',...
     'ud.dir = get(me,''user'');',...
     'set(gcf,''user'',ud);',...
     'pplane8(''periodic'');'];
dud.period(1) = uimenu(meperiod,...
           'label','forward',...
           'user',1,...
           'call', periodstr);
dud.period(2) = uimenu(meperiod,...
           'label','backward',...
           'user',-1,...
           'call', periodstr);
dud.period(3) = uimenu(meperiod,...
           'label','in both directions',...
           'user',0,...
           'call', periodstr);

nullcall = ['me = gcbo;',...
    'dud = get(gcf,''user'');',...
    'handx = [dud.arr.hx; dud.arr.hy];',...
    'handr = [dud.arr.hr; dud.arr.hth];',...
    'arron = get(dud.arr.arrows,''vis'');',...
    'lab = get(me,''label'');',...
    'switch lab,',...
    '  case ''Show nullclines.'',',...
    '    set(handx,''vis'',''on'');',...
    '    if strcmp(arron,''on''),',...
    '      set(dud.arr.barrows,''vis'',''off'');',...
    '    else,',...
    '      set(dud.arr.barrows,''vis'',''on'');',...
    '    end,',...
    '    set(handr,''vis'',''off'');',...
    '    set(me,''label'',''Hide nullclines.'');',...
    '    rnh = findobj(''tag'',''rnull'');'...
    '    set(rnh,''label'',''Show polar nullclines.'');',...
    '  case ''Hide nullclines.'',',...
    '    set(handx,''vis'',''off'');',...
    '    set(dud.arr.barrows,''vis'',''off'');',...
    '    set(me,''label'',''Show nullclines.'');',...
    'end'];

rnullcall = ['me = gcbo;',...
     'dud = get(gcf,''user'');',...
```

```
         'handx = [dud.arr.hx; dud.arr.hy];',...
         'handr = [dud.arr.hr; dud.arr.hth];',...
         'lab = get(me,''label'');',...
         'switch lab,',...
         '  case ''Show polar nullclines.'',',...
         '    set(handr,''vis'',''on'');',...
         '    set(handx,''vis'',''off'');',...
         '    set(dud.arr.barrows,''vis'',''off'');',...
         '    set(me,''label'',''Hide polar nullclines.'');',...
         '    nh = findobj(''tag'',''null'');'...
         '    set(nh,''label'',''Show nullclines.'');',...
         '  case ''Hide polar nullclines.'','...
         '    set(handr,''vis'',''off'');',...
         '    set(me,''label'',''Show polar nullclines.'');',...
         'end'];

    menunull = uimenu(solmenu,'label','Show nullclines.',...
          'call',nullcall,'separator','on','tag','null');

    menulevel = uimenu(solmenu,'label','Plot level curves.',...
           'call','pplane8(''level'')',...
           'separator','off','tag','level');

    if dud.egg
  menurnull = uimenu(solmenu,'label','Show polar nullclines.',...
             'call',rnullcall,'separator','off','tag','rnull');
  metest = uimenu(solmenu,'label','Test case',...
          'call','pplane8(''test case'')',...
          'separator','on','tag','testcase');

    end

    % Options menu

    menu(1) = uimenu('label','Options','visible','off','pos',5);
    meset   = uimenu(menu(1),'label','Settings.',...
         'call','pplane8(''settings'')');

    mesolve = uimenu(menu(1),'label','Solver.');

    solverstr = ['ud = get(gcf,''user'');',...
         'me = gcbo;',...
         'meud = get(me,''user'');',...
         'ud.settings.refine = meud.refine;',...
         'ud.settings.tol = meud.tol;',...
         'ud.settings.solver = meud.solver;',...
         'ud.settings.stepsize = meud.stepsize;',...
         'set(ud.solver,''checked'',''off'');',...
         'set(me,''checked'',''on'');',...
         'set(gcf,''user'',ud);',...
         'ppset = findobj(''name'',''pplane8 Setup'');',...
         'sud = get(ppset,''user'');',...
         'sud.settings = ud.settings;',...
         'set(ppset,''user'',sud);',...
         'pplane8(''settings'');'];
```

```
   solver = dud.settings.solver;
   dpset.refine = 8;
   dpset.tol = dud.settings.tol;
   dpset.solver = 'Dormand Prince';
   dpset.stepsize = dud.settings.stepsize;
   dpset.hmax = 0;
   if strcmp(solver,'Dormand Prince')
 dpch = 'on';
   else
 dpch = 'off';
   end
   rk4set.refine = 1;
   rk4set.tol = dud.settings.tol;
   rk4set.solver = 'Runge-Kutta 4';
   rk4set.stepsize = dud.settings.stepsize;
   rk4set.hmax = 0;
   if strcmp(solver,'Runge-Kutta 4')
 rk4ch = 'on';
   else
 rk4ch = 'off';
   end
   ode15sset.refine = 1;
   ode15sset.tol = dud.settings.tol;
   ode15sset.solver = 'ode15s';
   ode15sset.stepsize = dud.settings.stepsize;
   ode15sset.hmax = 0;
   if strcmp(solver,'ode15s')
 ode15sch = 'on';
   else
 ode15sch = 'off';
   end
   ode23sset.refine = 1;
   ode23sset.tol = dud.settings.tol;
   ode23sset.solver = 'ode23s';
   ode23sset.stepsize = dud.settings.stepsize;
   ode23sset.hmax = 0;
   if strcmp(solver,'ode23s')
 ode23sch = 'on';
   else
 ode23sch = 'off';
   end
   ode113set.refine = 1;
   ode113set.tol = dud.settings.tol;
   ode113set.solver = 'ode113';
   ode113set.stepsize = dud.settings.stepsize;
   ode113set.hmax = 0;
   if strcmp(solver,'ode113')
 ode113ch = 'on';
   else
ode113ch = 'off';
   end
   ode23set.refine = 1;
   ode23set.tol = dud.settings.tol;
   ode23set.solver = 'ode23';
   ode23set.stepsize = dud.settings.stepsize;
   ode23set.hmax = 0;
```

```
  if strcmp(solver,'ode23')
ode23ch = 'on';
  else
ode23ch = 'off';
  end
  ode45set.refine = 8;
  ode45set.tol = dud.settings.tol;
  ode45set.solver = 'ode45';
  ode45set.stepsize = dud.settings.stepsize;
  ode45set.hmax = 0;
  if strcmp(solver,'ode45')
ode45ch = 'on';
  else
ode45ch = 'off';
  end
  ode23tset.refine = 1;
  ode23tset.tol = dud.settings.tol;
  ode23tset.solver = 'ode23t';
  ode23tset.stepsize = dud.settings.stepsize;
  ode23tset.hmax = 0;
  if strcmp(solver,'ode23t')
ode23tch = 'on';
  else
ode23tch = 'off';
  end
  ode23tbset.refine = 1;
  ode23tbset.tol = dud.settings.tol;
  ode23tbset.solver = 'ode23tb';
  ode23tbset.stepsize = dud.settings.stepsize;
  ode23tbset.hmax = 0;
  if strcmp(solver,'ode23tb')
ode23tbch = 'on';
  else
ode23tbch = 'off';
  end

  dud.solver(1) = uimenu(mesolve,'label','Dormand Prince',...
             'checked',dpch,...
             'call',solverstr,'user',dpset);

  dud.solver(2) = uimenu(mesolve,'label','Runge-Kutta 4',...
             'checked',rk4ch,...
             'call',solverstr,'user',rk4set);

  dud.solver(3) = uimenu(mesolve,'label','ode45',...
             'checked',ode45ch,...
             'separator','on',...
             'call',solverstr,'user',ode45set);

  dud.solver(4) = uimenu(mesolve,'label','ode23',...
             'checked',ode23ch,...
             'call',solverstr,'user',ode23set);

  dud.solver(5) = uimenu(mesolve,'label','ode113',...
             'checked',ode113ch,...
             'call',solverstr,'user',ode113set);
```

```
dud.solver(6) = uimenu(mesolve,'label','ode15s',...
           'checked',ode15sch,...
           'call',solverstr,'user',ode15sset);

dud.solver(7) = uimenu(mesolve,'label','ode23s',...
           'checked',ode23sch,...
           'call',solverstr,'user',ode23sset);

dud.solver(8) = uimenu(mesolve,'label','ode23t',...
           'checked',ode23tch,...
           'call',solverstr,'user',ode23tset);

dud.solver(9) = uimenu(mesolve,'label','ode23tb',...
           'checked',ode23tbch,...
           'call',solverstr,'user',ode23tbset);

plotch = [
'dud = get(gcf,''user'');',...
'aud = get(dud.axes,''user'');',...
'if aud.plot,',...
'   aud.plot = 0;',...
'   set(gcbo,''label'',''Plot while computing'');',...
'else,',...
'   aud.plot = 1;',...
'   set(gcbo,''label'',''Do not plot while computing'');',...
'end,',...
'set(dud.axes,''user'',aud);'];

medir = uimenu(menu(1),'label','Solution direction.');

directionstr = ['ud = get(gcf,''user'');',...
        'me = gcbo;',...
        'ud.dir = get(me,''user'');',...
        'set(ud.direction,''checked'',''off'');',...
        'set(me,''checked'',''on'');',...
        'set(gcf,''user'',ud);'];

dud.direction(1) = uimenu(medir,'label','Both',...
          'checked','on',...
          'user',0,...
          'call',directionstr);
dud.dir = 0;

dud.direction(2) = uimenu(medir,'label','Forward',...
          'user',1,...
          'call',directionstr);

dud.direction(3) = uimenu(medir,'label','Back',...
          'user',-1,...
          'call',directionstr);


markstr = ['ud = get(gcf,''user'');',...
```

```
          'me = gcbo;',...
          'chkd = get(me,''checked'');',...
          'if strcmp(chkd,''on''),',...
          '  set(me,''checked'',''off'');',...
          '  ud.markflag = 0;',...
          'else,',...
          '  set(me,''checked'',''on'');',...
          '  ud.markflag = 1;',...
          'end,',...
          'set(gcf,''user'',ud);'];

       dud.markflag = 0;
       dud.mark = uimenu(menu(1),'label','Mark initial points.',...
               'checked','off',...
               'call',markstr);

       meexportdata = uimenu(menu(1),'label','Export solution data.',...
                  'call','pplane8(''export'')',...
                  'separator','off','tag','dexp');

       meplot = uimenu(menu(1),'label','Do not plot while computing',...
               'call',plotch,'separator','on');

       menu(6) = uimenu(menu(1),'label','Make the Display Window
inactive.',...
            'call','pplane8(''hotcold'')','separator','on');

       dud.menu = menu;

       % View menu

       set(findobj(gcf,'label','&View'),'pos',6);
       set(findobj(gcf,'label','&Figure Toolbar'),...
       'call','pplane8(''showbar'')');

       set(0,'showhiddenhandles',hhsetup);

       set(gcf,'WindowButtonDownFcn','pplane8(''down'')');
       set(ppdisp,'WindowButtonMotionFcn','pplane8(''cdisp'')');
       hh1 = [dud.axes,dud.title.axes,nframe,dud.notice,dbutt([1 2 3])];
       set(hh1,'units','norm');
       hh2 =
[nframe,dud.notice,dbutt(2:3),dud.axes,dud.menu(1),meplot,megraph];
       set(hh2,'visible','on');

       set(ppdisp,'vis','on');
       dud.printstr = 'print -noui';
     end   % if (~isempty(ppdisp))  & else
     ppdispa = dud.axes;
     axes(ppdispa);
     cla
     xlabel(txname);
     ylabel(tyname);
     % Initialize the window matrix.
```

```
 set(findobj('tag','zbmenu'),'enable','off');
 tstr1 = [txname,' '' = ', txderstr];
 tstr2 = [tyname,' '' = ', tyderstr];
 tstr = str2mat(tstr1,tstr2);
 dud.tstr = tstr;



 k = find(pflag);
 if ~isempty(k)
   lk = length(k);
   switch lk
case 1
 pstr1 = [pstring(k);{' '}];
case 2
 pstr1 = pstring(k);
case 3
 pstr1 = pstring(k([2,3]));
 pstr2 = [pstring(k(1));{' '}];
case 4
 pstr1 = pstring(k([2,4]));
 pstr2 = pstring(k([1,3]));
case 5
 pstr1 = pstring(k([3,5]));
 pstr2 = pstring(k([2,4]));
 pstr3 = [pstring(k(1));{' '}];
case 6
 pstr1 = pstring(k([3,6]));
 pstr2 = pstring(k([2,5]));
 pstr3 = pstring(k([1,4]));
   end
 end
 set(dud.title.eq,'string',tstr);
 set(dud.title.p1,'string',pstr1);
 ext = get(dud.title.p1,'extent');
 pos = get(dud.title.p1,'pos');
 p1 = min(.9,  .93 - ext(3));
 pos(1)  = p1;
 set(dud.title.p1,'pos',pos);
 set(dud.title.p2,'string',pstr2);
 ext = get(dud.title.p2,'extent');
 pos = get(dud.title.p2,'pos');
 p2 = min(.8, p1 - ext(3)-0.02);
 pos(1)  = p2;
 set(dud.title.p2,'pos',pos);
 set(dud.title.p3,'string',pstr3);
 ext = get(dud.title.p3,'extent');
 pos = get(dud.title.p3,'pos');
 pos(1)  = min(.7, p2 - ext(3)-0.02);
 set(dud.title.p3,'pos',pos);

 % Initialize important information as user data.

 dud.function = dfcn;
 dud.solhand = [];  % Handles to solution curves.
 dud.ephand = [];   % Handles to equilibrium points.
```

```
     dud.arr = [];  % Handles for the direction and vector fields.
     dud.eqpts = [];    % Equilibrium point data.
     dud.ics = [];      % Marked initial points.
     dud.wmat = [];
     dud.color = sud.color;
     set(ppdisp,'user',dud);
     if strcmp(dud.syst.fieldtype,'nullclines')
       menunull = findobj('tag','null');
       set(menunull,'label','Hide nullclines.');
     end
     ud.y = zeros(2,1);
     ud.i = 0;
     ud.line = 0;
     wind = dud.syst.wind(:);
     dwind = [wind(1); wind(3); -wind(2); -wind(4)];
     ud.DY = [wind(2)-wind(1); wind(4)-wind(3)];
     ud.cwind = dwind - dud.settings.magn*[ud.DY;ud.DY];
     ud.R = zeros(2,2);
     ud.rr = zeros(2,2);
     ud.perpeps = 0;
     ud.paraeps = 0;
     ud.sinkeps = 0;
     ud.minNsteps = 0;
     ud.turn = zeros(2,10);
     ud.tk = 0;
     ud.stop = 0;
     ud.gstop = 1;
     ud.plot = 1;
     set(dud.axes,'user',ud);
     tc = findobj('tag','testcase');
     if ~isempty(tc)
       if strcmp(dud.syst.name,'default system')
     set(tc,'vis','on');
       else
     set(tc,'vis','off');
       end
     end
     ppkbd = findobj('name','pplane8 Keyboard input','vis','on');
     if ~isempty(ppkbd),pplane8('kbd'),end
     pplevel = findobj('name','pplane8 Level sets');
     if ~isempty(pplevel),delete(pplevel),end
     set(dud.title.axes,'handle','off');
     pplane8('dirfield',ppdisp);
   end % if sud.flag == 1   &  else

elseif strcmp(action,'linear')

   % Initialize linearization window.

   % Linear takes the information from the EQPT window and initializes
   % the Linear Display window if it already exists.  If the Linear
   % Display window does not exist, it builds one.

   ppeqpt = gcf;

   % Get the information from pplane8 Equilibrium ...
```

```
sud = get(ppeqpt,'user');
WINvect = [-1 1 -1 1];
jac = sud.jac;
type = sud.type;
vectors = sud.vectors;
ppdisp = findobj('name','pplane8 Display');
pdud = get(ppdisp,'user');
settings = pdud.settings;
system = sud.system;

% Find pplane8 Linearization if it exists.
% If pplane8 Linearization exists, update it.  If it does not build it.

pplin= findobj('name','pplane8 Linearization');
if (~isempty(pplin))
   figure(pplin);
   dud = get(pplin,'user');
   dud.syst = system;
   dud.settings = settings;
   dfcn = dud.function;
else
   pplin = figure('name','pplane8 Linearization',...
      'numb','off',...
      'interrupt','on',...
      'visible','off',...
      'tag','pplane8');
   pplane8('figdefault',pplin);
   dud = get(pplin,'user');
   dud.dir = 0;
   dud.syst = system;
   dud.settings = settings;
   dfcn = '';
   fs = dud.fontsize;
   ssize = dud.ssize;
   r = ssize/10;
   ppaxw = 300;  % Default axes width
   ppaxh = 300;  % Default axes height
   ppaxl = 45;   % Default axes left
   buttw = 40;     % Default button width
   titleh = 45;    % Default title height
   ppaxb = 35;
   ppdh = ppaxb + ppaxh + titleh;
   uni = get(0,'units');
   set(0,'units','pixels');
   ss = get(0,'screensize');
   set(0,'units',uni);
   sw = ss(3);sh = ss(4);
   if r*ppdh > sh -80;
      r = (sh-80)/ppdh;
      fs = fs*r;
      lw = 0.5*r;
      set(gcf,'defaultaxesfontsize',fs,'defaultaxeslinewidth',lw);
      set(gcf,'defaulttextfontsize',fs);
      set(gcf,'defaultlinelinewidth',lw);
      set(gcf,'defaultuicontrolfontsize',fs*0.9);
      end

      axpos = r*[ppaxl,ppaxb,ppaxw,ppaxh];

      % Set up the plot axes.

      dud.axes = axes('units','pix',...
         'position',axpos,...
         'next','add',...
         'box','on',...
         'interrupt','on',...
         'xgrid','on',...
         'ygrid','on',...
         'drawmode','fast',...
         'visible','off',...
         'tag','linear axes');

      % Set up the title.

      dud.title.axes = axes('box','off','xlim',[0 1],'ylim',[0 1],...
         'units','pix','vis','off','xtick',[-1],'ytick',[-1],...
         'xticklabel','','yticklabel','');

      dud.title.eq = text(0.07,0.5,' ','vert','middle');
      dud.title.p1 = text(0.75,0.5,' ','vert','middle');
      dud.title.p2 = text(0.65,0.5,' ','vert','middle');
      tstr = {'x_2';'y^2'};
      set(dud.title.eq,'string',tstr,'units','pix');
      ext = get(dud.title.eq ,'extent');
      titleh = ext(4)+15*r;
      set(dud.title.eq,'units','data');
      taxpos = r*[ppaxl,ppaxb+ppaxh,ppaxw,titleh];
      set(dud.title.axes,'pos',taxpos,'color',get(gcf,'color'));

      % Finish the positions.

      %         buttw = r*buttw;
      %         butth = fs+10*r;
      %         buttl = ppaxl + ppaxw + 5;
      ppdw = r*(ppaxl + ppaxw + 35);
      ppdh = r*(ppaxb+ppaxh)+titleh;

      ppdleft = sw-ppdw-60;
      ppdbot = 60;
      ppdpos = [ppdleft,ppdbot,ppdw,ppdh];
      set(pplin,'resize','on');
      set(pplin,'pos',ppdpos);

      Arrflag = system.fieldtype;

      axpos = r*[ppaxl,ppaxb,ppaxw,ppaxh];

      stopstr = 'aud = get(gca,''user'');aud.stop =
4;set(gca,''user'',aud);';
```

```
      stoppos = [axpos(1)+axpos(3)-r*10, axpos(2)+axpos(4)+r*10,
r*40,fs+10*r];

      dud.butt  = uicontrol('style','push',...
         'pos',stoppos,...
         'string','Stop','call',stopstr,'vis','off','tag','stop');

      hhsetup = get(0,'showhiddenhandles');
      set(0,'showhiddenhandles','on');

      % Configure the Toolbar.

      set(pplin,'ToolBar','none');

      tmenu = findobj(gcf,'label','&Tools');
      delete(tmenu);

      % File menu

      fmenu = findobj(gcf,'label','&File');
      delete(findobj(fmenu,'label','&New Figure'));
      delete(findobj(fmenu,'label','&Open...'));
      delete(findobj(fmenu,'label','Pre&ferences...'));
      set(findobj(fmenu,'label','&Close'),'pos',1);
      set(findobj(fmenu,'label','&Save'),'pos',2,...
            'separator','off');
      set(findobj(fmenu,'label','Save &As...'),'pos',3);
      set(findobj(fmenu,'label','&Export...'),...
      'pos',4);
      delete(findobj(fmenu,'label','Pre&ferences...'));
      set(findobj(fmenu,'label','Pa&ge Setup...'),'pos',5);
      set(findobj(fmenu,'label','Print Set&up...'),'pos',6);
      set(findobj(fmenu,'label','Print Pre&view...'),'pos',7);
      set(findobj(fmenu,'label','&Print...'),'pos',8);

      mequit = uimenu(fmenu,'label','Quit pplane8',...
      'call','pplane8(''quit'')','separator','on','pos',9);

      % View menu

      set(findobj(gcf,'label','&Figure Toolbar'),...
      'call','pplane8(''showbar'')');

      % Solutions menu

      solmenu = uimenu('label','Solutions','pos',3);

      menukey = uimenu(solmenu,'label','Keyboard input.','call',...
         'pplane8(''kbd'')','vis','on');

      mesev   = uimenu(solmenu,'label','Plot several solutions.',...
         'call','pplane8(''several'')');
      fundcall = ['dud = get(gcf,''user'');',...
          'col = dud.color.sep;',...
          'vect = dud.vectors;',...
          'h = zeros(1,2);',...
          'h(1) = plot(2*[vect(1,1),-vect(1,1)],2*[vect(2,1),-
vect(2,1)]);',...
          'h(2) = plot(2*[vect(1,2),-vect(1,2)],2*[vect(2,2),-
vect(2,2)]);',...
          'set(h,''color'',col);',...
          'dud.solhand = [dud.solhand;h(:)];'...
          'set(gcf,''user'',dud);'];

      dud.menu = uimenu(solmenu,'call',fundcall);

      markstr = ['ud = get(gcf,''user'');',...
         'me = gcbo;',...
         'chkd = get(me,''checked'');',...
         'if strcmp(chkd,''on''),',...
         '  set(me,''checked'',''off'');',...
         '  ud.markflag = 0;',...
         'else,',...
         '  set(me,''checked'',''on'');',...
         '  ud.markflag = 1;',...
         'end,',...
         'set(gcf,''user'',ud);'];

      dud.markflag = pdud.markflag;
      chkd = 'off';
      if dud.markflag
    chkd = 'on';
      end
      dud.mark = uimenu(solmenu,'label','Mark initial points.',...
            'checked',chkd,...
            'call',markstr);
      % Edit menu

      emenu = findobj(gcf,'label','&Edit');

      medallsol = uimenu(emenu,'label','Erase all solutions.',...
             'call','pplane8(''dallsol'')',...
             'pos',1);

      medallics = uimenu(emenu,'label','Erase all marked initial points.',...
             'call','pplane8(''dallics'')',...
             'separator','off',...
             'pos',2);

      medall = uimenu(emenu,'label','Erase all graphics objects.',...
               'call','pplane8(''dall'')',...
               'separator','off',...
               'pos',3);
      medel = uimenu(emenu,'label','Delete a graphics object.',...
             'call','pplane8(''delete'')',...
             'visible','on',...
             'pos',4);

      menutext = uimenu(emenu,'label','Enter text on the Display Window.',...
            'call','pplane8(''text'')',...
            'pos',5);
      set(findobj(emenu,'label','&Undo'),'separator','on',...
            'pos',6);
      set(findobj(emenu,'label','Cu&t'),'pos',7);
      set(findobj(emenu,'label','&Copy'),'pos',8);
      set(findobj(emenu,'label','&Paste'),'pos',9);
      set(findobj(emenu,'label','Clea&r'),'pos',10);
      set(findobj(emenu,'label','&Select All'),'pos',11);
      set(findobj(emenu,'label','Copy &Figure'),'pos',12);
      set(findobj(emenu,'label','Copy &Options'),'pos',13);
      set(findobj(emenu,'label','F&igure Properties'),'pos',14);
      set(findobj(emenu,'label','&Axes Properties'),'pos',15);
      set(findobj(emenu,'label','C&urrent Object Properties'),'pos',16);

      % Options menu

      optmenu = uimenu('label','Options','visible','off');

      mehc = uimenu(optmenu,'label','Make the Display Window inactive.',...
         'call','pplane8(''hotcold'')','separator','on');

      set(0,'showhiddenhandles',hhsetup);


set(pplin,'WindowButtonDownFcn',['pplane8(''down'',',num2str(pplin),')']);
      hh1 = [dud.axes,dud.title.axes];
      set(hh1,'units','norm');
      set(dud.axes,'visible','on');
      set(pplin,'vis','on');
      dud.printstr = 'print';
   end   % if (~isempty(pplin))  & else
   dud.ics = [];

   switch type
   case 1
      vstr = 'on';
      lstr = 'Plot stable and unstable orbits.';
   case {2,3}
      vstr = 'on';
      lstr = 'Plot fundamental modes.';
   otherwise
      vstr = 'off';
      lstr='';
   end
   set(dud.menu,'label',lstr,'vis',vstr);

   pplina = dud.axes;
   dud.vectors = sud.vectors;
   axes(pplina);
   cla
   xlabel('u');
```

```
ylabel('v');

% The title strings.

tstr = {'u'' = A u + B v';'v'' = C u + D v'};
dud.tstr = tstr;
pstr2 = {['A = ',num2str(jac(1,1))];['C = ',num2str(jac(2,1))]};
pstr1 = {['B = ',num2str(jac(1,2))];['D = ',num2str(jac(2,2))]};

set(dud.title.eq,'string',tstr);
set(dud.title.p1,'string',pstr1);
ext = get(dud.title.p1,'extent');
pos = get(dud.title.p1,'pos');
p1 = min(.9, .93 - ext(3));
pos(1) = p1;
set(dud.title.p1,'pos',pos);
set(dud.title.p2,'string',pstr2);
ext = get(dud.title.p2,'extent');
pos = get(dud.title.p2,'pos');
pos(1) = min(.8, p1 - ext(3)-0.02);
set(dud.title.p2,'pos',pos);

if (~strcmp(dfcn,'') & exist(dfcn)==2) delete([tempdir,dfcn,'.m']);end
tee = clock;
tee = ceil(tee(6)*100);
dfcn=['pptp',num2str(tee)];
fcnstr = ['function ypr = ',dfcn,'(t,y)\n\n'];
commstr = '%%%% Created by pplane8\n\n';
astr = ['A = [',num2str(jac(1,1)),',',num2str(jac(1,2)),';',...
      num2str(jac(2,1)),',',num2str(jac(2,2)),'];\n'];
dstr = 'ypr = A * y;';
ppf = fopen([tempdir,dfcn,'.m'],'w');
fprintf(ppf,fcnstr);
fprintf(ppf,commstr);
fprintf(ppf,astr);
fprintf(ppf,dstr);
fclose(ppf);

% Initialize important information as user data.

dud.function = dfcn;
dud.solhand = [];    % Handles to solution curves.
dud.ephand = []; % Handles to equilibrium points.
dud.contours = [];
dud.arr = [];        % Handles for the direction and vector
% fields.
dud.eqpts = [];      % Equilibrium point data.
dud.notice = 0;
dud.syst.wind = [-1 1 -1 1]';
dud.color = sud.color;
ud.y = zeros(2,1);
ud.i = 0;
ud.line = 0;
wind = [-1 1 -1 1]';
dwind = [wind(1); wind(3); -wind(2); -wind(4)];
ud.DY = [wind(2)-wind(1); wind(4)-wind(3)];
```

```
   ud.cwind = dwind - dud.settings.magn*[ud.DY;ud.DY];
   ud.R = zeros(2,2);
   ud.rr = zeros(2,2);
   ud.perpeps = 0;
   ud.paraeps = 0;
   ud.sinkeps = 0;
   ud.turn = zeros(2,10);
   ud.tk = 0;
   ud.stop = 0;
   ud.gstop = 1;
   ud.plot = 1;
   dud.ephand = plot(0,0,...
            'color',dud.color.eqpt,...
            'markersize',20,...
            'marker','.');
   set(dud.axes,'user',ud);
   set(pplin,'user',dud);
   ppkbd = findobj('name','pplane8 Keyboard input','vis','on');
   if ~isempty(ppkbd),pplane8('kbd'),end
   pplane8('dirfield',pplin);

elseif strcmp(action,'dirfield')

   % 'dirfield' computes and plots the field elements.  This is the entry
   % point both from 'display' and from later commands that require the
   % recomputation of the field elements.

   % Find pplane8 Display and get the user data.

   disph = input1;   % This could be ppdisp or pplin.

   dud = get(disph,'user');
   color = dud.color;

   dfcn = dud.function;
   ppdispa = dud.axes;
   WINvect = dud.syst.wind;
   settings = dud.settings;
   notice = dud.notice;
   if notice
      nstr = get(notice,'string');
      nstr(1:4)=nstr(2:5);
      nstr{5,1} = 'Computing the field elements.';
      set(notice,'string',nstr);

      % Augment the window matrix

      wmat = dud.wmat;
      wrows = size(wmat,1);
      wflag = 0;
      for k = 1:wrows
         if wmat(k,:)==WINvect
            wflag = 1;
         end
      end
      if wflag == 0
         wmat = [wmat;WINvect];
         dud.wmat = wmat;
      end
      if wrows
    hhsetup = get(0,'showhiddenhandles');
    set(0,'showhiddenhandles','on');
    hhh = findobj('tag','zbmenu');
    set(hhh,'enable','on');
    set(0,'showhiddenhandles',hhsetup);
      end
   end


   Xmin = WINvect(1);
   Xmax = WINvect(2);
   Ymin = WINvect(3);
   Ymax = WINvect(4);

   N = dud.syst.npts;
   if strcmp(get(disph,'name'),'pplane8 Linearization')
      N = floor(3*N/4);
   end
   deltax=(Xmax - Xmin)/(N-1);
   deltay=(Ymax - Ymin)/(N-1);

   % Set up the display window.

   Dxint=[Xmin-deltax,Xmax+deltax];
   Dyint=[Ymin-deltay,Ymax+deltay];

   % Set up the original mesh.

   XXXg=Xmin + deltax*[0:N-1];
   YYYg=Ymin + deltay*[0:N-1];

   [Xx,Yy]=meshgrid(XXXg,YYYg);

   % Calculate the line and vector fields.

   Xx=Xx(:);Yy=Yy(:);
   Ww = zeros(size(Xx));
   Ww = feval(dfcn,0,[Xx';Yy']);
   Vv = Ww(1,:) + Ww(2,:)*sqrt(-1);
   Vv = Vv.';
   Arrflag = dud.syst.fieldtype;


   mgrid = Xx+Yy.*sqrt(-1); % mgrid = mgrid(:);
   zz=Vv.';
   sc = min(deltax,deltay);

   arrow=[-1,1].';
   zzz=sign(zz);
```

```
scale = sqrt((real(zzz)/deltax).^2+(imag(zzz)/deltay).^2);
ww = (zzz == 0);
scale = scale + ww;
aa1 = 0.3*arrow*(zzz./scale)+ones(size(arrow))*(mgrid.');
[r,c] = size(aa1);
aa1=[aa1;NaN*ones(1,c)];
aa1=aa1(:);

arrow = [0,1,.7,1,.7].' + [0,0,.25,0,-.25].' * sqrt(-1);
zz=sign(zz).*((abs(zz)).^(1/3));
scale = 0.9*sc./max(max(abs(zz)));
aa2 = scale*arrow*zz +ones(size(arrow))*(mgrid.');
[r,c] = size(aa2);
aa2=[aa2;NaN*ones(1,c)];
aa2=aa2(:);
axes(ppdispa);

arr = dud.arr;   % Delete the old field data.
if isstruct(arr)
   hand = [arr.hx;arr.hy;arr.lines;arr.arrows;arr.barrows];
   delete(hand);
end
NN = N;
N = 50; k = ceil(N/NN);
deltax=(Xmax - Xmin)/(N-1);
deltay=(Ymax - Ymin)/(N-1);

% Set up the original mesh.

XXXg=Xmin + deltax*[-k:N+k];
YYYg=Ymin + deltay*[-k:N+k];

[Xx,Yy]=meshgrid(XXXg,YYYg);

% Calculate the line and vector fields.

Xxx=Xx(:);Yyy=Yy(:);
Ww = zeros(size(Xxx));
Ww = feval(dfcn,0,[Xxx';Yyy']);
DX = Ww(1,:)';
DY = Ww(2,:)';
minx = min(DX);
maxx = max(DX);
miny = min(DY);
maxy = max(DY);
DR = Xxx.*DX + Yyy.*DY;
DTheta = Xxx.*DY - Yyy.*DX;
minr = min(DR);
maxr = max(DR);
minth = min(DTheta);
maxth = max(DTheta);
DX = reshape(DX,N+2*k+1,N+2*k+1);
DY = reshape(DY,N+2*k+1,N+2*k+1);
DR = reshape(DR,N+2*k+1,N+2*k+1);
DTheta = reshape(DTheta,N+2*k+1,N+2*k+1);
```

```
if minx < 0 & 0 < maxx
   [Cx,hx] = contour(Xx,Yy,DX,[0,0],'--');
   set(hx,'visible','off','color',color.xcline,'linestyle','--');
else
   hx = zeros(0,1);
end
if miny < 0 & 0 < maxy
   [Cy,hy] = contour(Xx,Yy,DY,[0,0],'--');
   set(hy,'visible','off','color',color.ycline,'linestyle','--');
else
   hy = zeros(0,1);
end
if minr < 0 & 0 < maxr
   [Cr,hr] = contour(Xx,Yy,DR,[0,0],'--');
   set(hr,'visible','off','color',color.xcline,'linestyle','--');
else
   hr = zeros(0,1);
end
if minth < 0 & 0 < maxth
   [Cth,hth] = contour(Xx,Yy,DTheta,[0,0],'--');
   set(hth,'visible','off','color',color.ycline,'linestyle','--');
else
   hth = zeros(0,1);
end
arrh1 = plot(real(aa1),imag(aa1),'color',color.arrows,'visible','off');
arrh2 = plot(real(aa2),imag(aa2),'color',color.arrows,'visible','off');

% We plot both the line field and the vector field.  Then we
% control which is seen by manipulating the visibility.
Xmin = Xmin - k*deltax;
Xmax = Xmax + k*deltax;
Ymin = Ymin - k*deltay;
Ymax = Ymax + k*deltay;
Zz = (Xx-Xmin).*(Xmax-Xx).*(Yy-Ymin).*(Ymax-Yy).*abs(DX).*abs(DY);
Zzc = Zz(2:N+k+2,2:N+k+2);
Xxc = Xx(2:N+k+2,2:N+k+2);
Yyc = Yy(2:N+k+2,2:N+k+2);
DXc = DX(2:N+k+2,2:N+k+2);
DYc = DY(2:N+k+2,2:N+k+2);
Zzl = Zz(1:N+k+1,2:N+k+2);
Zzr = Zz(3:N+k+3,2:N+k+2);
Zzd = Zz(2:N+k+2,1:N+k+1);
Zzu = Zz(2:N+k+2,3:N+k+3);
Zzdl = Zz(1:N+k+1,1:N+k+1);
Zzdr = Zz(3:N+k+3,1:N+k+1);
Zzul = Zz(1:N+k+1,3:N+k+3);
Zzur = Zz(3:N+k+3,3:N+k+3);
kk = find((Zzc>Zzl)&(Zzc>Zzr)&(Zzc>Zzd)&(Zzc>Zzu)...
   &(Zzc>Zzdl)&(Zzc>Zzdr)&(Zzc>Zzul)&(Zzc>Zzur));
Xxx = Xxc(kk);
Yyy = Yyc(kk);
DXx = DXc(kk);
DYy = DYc(kk);
ll = length(hx);
Xxx = Xxx(:);
Yyy = Yyy(:);
DXx = DXx(:);
```

```
DYy = DYy(:);
if ~exist('both')
   Xxx = zeros(0,1);
   Yyy = zeros(0,1);
   DXx = zeros(0,1);
   DYy = zeros(0,1);
end
for j = 1:ll
   xd = get(hx(j),'xdata');xd = xd(:);
   yd = get(hx(j),'ydata');yd = yd(:);
   zxd = feval(dfcn,0,[xd';yd']);
   yxd = zxd(2,:);yxd = yxd(:);
   ayxd = abs(yxd).*(xd-Xmin).*(Xmax-xd).*(yd-Ymin).*(Ymax-yd);
   NNN = length(ayxd);
   ayxdc = ayxd(2:NNN-1);
   yxdc = yxd(2:NNN-1);
   ayxdl = ayxd(1:NNN-2);
   ayxdr = ayxd(3:NNN);
   lll = find((ayxdc>ayxdl)&(ayxdc>ayxdr));
   Xx = xd(lll+1);
   Yy = yd(lll+1);
   Xxx = [Xxx;Xx(:)];
   Yyy = [Yyy;Yy(:)];
   yxdcp = yxdc(lll);
   yxdcp = yxdcp(:);
   DXx = [DXx;zeros(size(yxdcp))];
   DYy = [DYy;yxdcp];
end
ll = length(hy);
for j = 1:ll
   xd = get(hy(j),'xdata');xd = xd(:);
   yd = get(hy(j),'ydata');yd = yd(:);
   zxd = feval(dfcn,0,[xd';yd']);
   yxd = zxd(1,:);yxd = yxd(:);
   ayxd = abs(yxd).*(xd-Xmin).*(Xmax-xd).*(yd-Ymin).*(Ymax-yd);
   NNN = length(ayxd);
   ayxdc = ayxd(2:NNN-1);
   yxdc = yxd(2:NNN-1);
   ayxdl = ayxd(1:NNN-2);
   ayxdr = ayxd(3:NNN);
   lll = find((ayxdc>ayxdl)&(ayxdc>ayxdr));
   Xx = xd(lll+1);
   Yy = yd(lll+1);
   Xxx = [Xxx;Xx(:)];
   Yyy = [Yyy;Yy(:)];
   yxdcp = yxdc(lll);
   yxdcp = yxdcp(:);
   DXx = [DXx;yxdcp];
   DYy = [DYy;zeros(size(yxdcp))];
end
mgrid = Xxx(:) + Yyy(:)*sqrt(-1);
Vv = DXx(:) + DYy(:)*sqrt(-1);
Vv = sign(Vv);       %.*((abs(Vv)).^(1/3));
sc = (N/20)*min(deltax,deltay);
aa3 = sc*arrow*(Vv.') +ones(size(arrow))*(mgrid.');
[r,c] = size(aa3);
aa3 = [aa3;NaN*ones(1,c)];
```

```
   aa3=aa3(:);
   arrh3 = plot(real(aa3),imag(aa3),'color',color.narrows,'visible','off');

   switch Arrflag
   case 'nullclines'
%      set([hx;hy;arrh3],'vis','on');
set([hx;hy],'vis','on');
   case 'lines'
      set(arrh1,'visible','on');
   case 'arrows'
      set(arrh2,'visible','on');
   end
   dud.arr.lines = arrh1;      % Save the handles for later use.
   dud.arr.arrows = arrh2;     % Save the handles for later use.
   dud.arr.hx = hx;
   dud.arr.hy = hy;
%   dud.arr.barrows = arrh3;
   dud.arr.barrows = [];
   dud.arr.hr = hr;
   dud.arr.hth = hth;

   menull = findobj('tag','null');
   mernull = findobj('tag','rnull');
   if strcmp(get(menull,'label'),'Delete nullclines.')
      set([hx;hy],'vis','on');
   elseif strcmp(get(mernull,'label'),'Delete polar nullclines.')
      set([hr;hth],'vis','on');
   end
   if notice
      nstr = get(notice,'string');
      nstr(1:4) = nstr(2:5);
      nstr{5,1} = 'Ready.';
      set(notice,'string',nstr);
   end
   set(disph,'user',dud);
   axis([Dxint,Dyint]);

elseif strcmp(action,'hotcold')

   % 'hotcold' is the callback for the menu selection that makes the
   % Display Window active or inactive.

   ppdisp = gcf;
   dud = get(ppdisp,'user');
   nstr = get(dud.notice,'string');
   nstr(1:4) = nstr(2:5);
   mehc = dud.menu(6);
   if (findstr(get(mehc,'label'),'inactive'))
      set(ppdisp,'WindowButtonDownFcn',' ');
      set(mehc,'label','Make the Display Window active.');
      nstr{5,1} = 'The Display Window is not active.';
      set(dud.notice,'string',nstr);
   else
      set(ppdisp,'WindowButtonDownFcn','pplane8(''down'')');
      set(mehc,'label','Make the Display Window inactive.');
      nstr{5,1} = 'The Display Window is active.';
      set(dud.notice,'string',nstr);
   end

elseif strcmp(action,'down')

  % 'down' is the Window Button Down call.  It starts the computation of
  % solutions from a click of the mouse.

  disph = gcf;
  seltype = get(disph,'selectiontype');
  if strcmp(seltype,'alt')
    pplane8('zoom');
    return
  elseif strcmp(seltype,'extend')
    pplane8('zoomsqd');
    return
  end
  dud = get(disph,'user');
  ax = dud.axes;
  ch = findobj('type','uicontrol','enable','on');
  set(ch,'enable','inactive');
  wbdf = get(disph,'WindowbuttonDownFcn');
  set(disph,'WindowbuttonDownFcn','');
  axes(ax);
  initpt = get(ax,'currentpoint');
  initpt = initpt(1,[1,2]);
  if dud.markflag
    h = plot(initpt(1),initpt(2),'.k');
    dud.ics = [dud.ics,h];
    set(disph,'user',dud);
  end
  pplane8('solution',initpt,disph);
  set(disph,'WindowbuttonDownFcn',wbdf);
  %  set([ch;mh],'enable','on');
  set(ch,'enable','on');
  notice = dud.notice;
  if notice
    nstr = get(notice,'string');
    nstr(1:4) = nstr(2:5);
    nstr{5,1} = 'Ready.';
    set(notice,'string',nstr)
  end

elseif strcmp(action,'several')

   % 'several' allows the user to pick several initial points at once.
   % This is not needed in X-windows, but it is on the Macintosh.

   disph = gcf;
   ch = findobj('type','uicontrol','enable','on');
   %  mh = findobj('type','uimenu','enable','on');
   set(ch,'enable','inactive');
   %  set(mh,'enable','off')
   wbdf = get(disph,'WindowbuttonDownFcn');
   set(disph,'WindowbuttonDownFcn','');
   dud = get(disph,'user');
   notice = dud.notice;
   if notice
      nstr = get(notice,'string');
      nstr(1:4) = nstr(2:5);
      nstr{5,1} = 'Pick initial points with the mouse. Enter "Return" when 
finished.';
      set(notice,'string',nstr)
   end
%    [X,Y]=ppginput;
    [X,Y]=ginput;
  NN = length(X);
   for k = 1:NN
      initpt = [X(k),Y(k)];
      pplane8('solution',initpt,disph);
   end
   if notice
      nstr = get(notice,'string');
      nstr(1:4) = nstr(2:5);
      nstr{5,1} = 'Ready.';
      set(notice,'string',nstr);
   end
   set(ch,'enable','on');
   %  set([ch;mh],'enable','on');
   set(disph,'WindowbuttonDownFcn',wbdf);

elseif strcmp(action,'test case')

   tic
   ppdisp = gcf;
   for k = 1:10
      initpt = k*[-.2 .2];
      pplane8('solution',initpt,ppdisp);
      initpt = k*[-.2 -.2];
      pplane8('solution',initpt,ppdisp);
   end
   pplane8('solution',[1,-1],ppdisp);
   dud = get(gcf,'user');
   nstr = get(dud.notice,'string');
   nstr(1:4) = nstr(2:5);
   nstr{5,1} = 'Ready.';
   set(dud.notice,'string',nstr)
   toc

elseif strcmp(action,'solution')

   % 'solution' effects the computation and (erasemode == none) plotting of
   % solutions.  It also stores the data as appropriate.

   disph = input2;
   dud = get(disph,'user');
   tcol = dud.color.temp;
   pcol = dud.color.orb;
   notice = dud.notice;
   initpt = input1(:);
   dfcn = dud.function;
   ppdispa = dud.axes;
```

```
settings = dud.settings;
ptstr = [' (',num2str(initpt(1),2), ', ', num2str(initpt(2),2), ')'];
refine = settings.refine;
ssize = settings.stepsize;
tol = settings.tol;
ud = get(dud.axes,'user');
%  rtol = tol;
atol = tol*ud.DY*1e-4';


if length(initpt)  == 2
  AA = -1e6;
  BB = 1e6;
  switch dud.dir
   case 0
    intplus = [0, BB];
    intminus = [0, AA];
   case -1
    intplus = [0, 0];
    intminus = [0, AA];
   case 1
    intplus = [0, BB];
    intminus = [0, 0];
  end

else
  intplus = [initpt(3),initpt(5)];
  intminus = [initpt(3),initpt(4)];
  initpt = initpt([1:2]);

end
stopbutt = findobj(disph,'tag','stop');

set(stopbutt,'vis','on','enable','on');

solver = settings.solver;
switch solver
 case 'Dormand Prince'
  solh = @ppdp45;
  opt = disph;
 case 'Runge-Kutta 4'
  solh = @pprk4;
  opt = disph;
 case 'ode45'
  solh = @ode45;
  opt = odeset('OutputFcn',@ppout,'Refine',refine,...
           'RelTol',tol,'Abstol',atol);
 case 'ode23'
  solh = @ode23;
  opt = odeset('OutputFcn',@ppout,'Refine',refine,...
           'RelTol',tol,'Abstol',atol);
 case 'ode113'
  solh = @ode113;
  opt = odeset('OutputFcn',@ppout,'Refine',refine,...
           'RelTol',tol,'Abstol',atol);
 case 'ode15s'
```

```
  solh = @ode15s;
  opt = odeset('OutputFcn',@ppout,'Refine',refine,...
           'RelTol',tol,'Abstol',atol);
 case 'ode23s'
  solh = @ode23s;
  opt = odeset('OutputFcn',@ppout,'Refine',refine,...
           'RelTol',tol,'Abstol',atol);
 case 'ode23t'
  solh = @ode23t;
  opt = odeset('OutputFcn',@ppout,'Refine',refine,...
           'RelTol',tol,'Abstol',atol);
 case 'ode23tb'
  solh = @ode23tb;
  opt = odeset('OutputFcn',@ppout,'Refine',refine,...
           'RelTol',tol,'Abstol',atol);
end

exist(dfcn);
dfh = str2func(dfcn);
cflag = 0;

if intplus(2)>intplus(1)
  cflag = cflag + 1;
  if notice
    nstr = get(notice,'string');
    nstr(1:4) = nstr(2:5);
    nstr{5} = ['The forward orbit from',ptstr];
    set(notice,'string',nstr);
  end
  drawnow

  [tp,xp] = feval(solh,dfh,intplus,initpt,opt);
  aud = get(ppdispa,'user');
  hnew1 = aud.line;
end

if intminus(2) < intminus(1)
  cflag = cflag + 2;
  if notice
    nstr = get(notice,'string');
    nstr(1:4) = nstr(2:5);
    nstr{5} = ['The backward orbit from',ptstr];
    set(notice,'string',nstr);
  end
  drawnow
  [tm,xm] = feval(solh,dfh,intminus,initpt,opt);
  aud = get(ppdispa,'user');
  hnew2 = aud.line;

  set(stopbutt,'vis','off');
end    % if intminus(2) < intminus(1)



% Store the trajectory.
```

```
   switch cflag
   case 1 % positive only
      set(hnew1,'xdata',xp(:,1),'ydata',xp(:,2),'zdata',tp,'color',pcol);
      set(hnew1,'erase','normal');
      dud.solhand = [dud.solhand;hnew1];
   case 2 % negative only
      x = flipud(xm);
      t = flipud(tm);
      set(hnew2,'xdata',x(:,1),'ydata',x(:,2),'zdata',t,'color',pcol);
      set(hnew2,'erase','normal');
      dud.solhand = [dud.solhand;hnew2];

   case 3 % both directions
      x = flipud(xm);
      t = flipud(tm);
      x=[x;xp];
      t=[t;tp];
      delete(hnew1);
      set(hnew2,'xdata',x(:,1),'ydata',x(:,2),'zdata',t,'color',pcol);
      set(hnew2,'erase','normal');
      dud.solhand = [dud.solhand;hnew2];
   end   % switch cflag
   set(disph,'user',dud);

elseif strcmp(action,'kcompute')

   % 'kcompute' is the call back for the Compute
   % button on the pplane8 Keyboard figure.

   compute = 1;
   kh = get(gcf,'user');
   ppdisp = kh.fig;
   if (isempty(ppdisp))
      pplane8('confused');
   end
   dud = get(ppdisp,'user');
   ppdispa = dud.axes;
   aud = get(ppdispa,'user');
   ppset = findobj('name','pplane8 Setup');
   sud = get(ppset,'user');
   ch = findobj('type','uicontrol','enable','on');
   set(ch,'enable','inactive');
   set(ppdisp,'WindowbuttonDownFcn','');
   xstr = get(kh.xval,'string');
   ystr = get(kh.yval,'string');
   pnameh = sud.h.pname;
   pvalh = sud.h.pval;
   pflag = zeros(1,4);
   perr = [];
   for kk = 1:6;
     pn = char(get(pnameh(kk),'string'));
     pv = char(get(pvalh(kk),'string'));
     if ~isempty(pn)
       if isempty(pv)
     perr = pvalh(kk);
```

```
    else
  xstr = pplane8('paraeval',pn,pv,xstr);
  ystr = pplane8('paraeval',pn,pv,ystr);
    end
  end
end
xvalue = str2num(xstr);
yvalue = str2num(ystr);


if get(kh.spec,'value')
   tzero = str2num(get(kh.tval,'string'));
   t0 = str2num(get(kh.t0,'string'));
   tf = str2num(get(kh.tf,'string'));
   initpt = [xvalue,yvalue,tzero,t0,tf];
   str1 = 'Values must be entered for all of the entries.';
   if (length(initpt) ~= 5)
      warndlg({str1},'Illegal input');
      compute = 0;
   elseif tf <= t0
      warndlg({'The final time of the computation interval';...
            'must be smaller than the initial time.'},'Illegal input');
      compute = 0;
   elseif (tzero < t0) | (tzero > tf)
 str2 = 'The initial time must be in the computation interval.';
      warndlg(str2,'Illegal input');
      compute = 0;
   end
   aud.gstop = 0;
   set(ppdispa,'user',aud);
else
   initpt = [xvalue,yvalue];
   if (length(initpt) ~= 2)
 warndlg({str1},'Illegal input');
 compute = 0;
   end
end  % if get(kh.spec,'value')

if compute
  if dud.markflag
    figure(ppdisp)
    h = plot(initpt(1),initpt(2),'.k');
    dud.ics = [dud.ics,h];
    set(ppdisp,'user',dud);
  end
  pplane8('solution',initpt,ppdisp);
end
if dud.notice
  nstr = get(dud.notice,'string');
  nstr(1:4) = nstr(2:5);
  nstr{5} = 'Ready.';
  set(dud.notice,'string',nstr);
end
%  set([ch;mh],'enable','on');
set(ch,'enable','on');
set(ppdisp,'WindowbuttonDownFcn','pplane8(''down'')');
```

```
   aud.gstop = 1;
   set(ppdispa,'user',aud);


elseif strcmp(action,'kbd')

   % 'kbd' is the callback for the Keyboard Input menu selection.  It
   % sets up the pplane8 Keyboard figure which allows accurate input of
   % initial values using the keyboard.

   ppdisp = gcf;  % The figure to be plotted in.
   dud = get(ppdisp,'user');
   Xname = dud.syst.xvar;
   Yname = dud.syst.yvar;
   xnstr = ['The initial value of ',Xname,' = '];
   ynstr = ['The initial value of ',Yname,' = '];
   tnstr = 'The initial value of t = ';
   ppkbd = findobj('name','pplane8 Keyboard input');
   if  ~isempty(ppkbd)
     delete(ppkbd);
   end

   ppkbd = figure('name','pplane8 Keyboard input',...
          'vis','off',...
          'numb','off','tag','pplane8');

   pplane8('figdefault',ppkbd);
   set(ppkbd,'menubar','none');

   kbd.fr1 = uicontrol('style','frame');

   kbd.fr2 = uicontrol('style','frame');

   kbd.fr3 = uicontrol('style','frame');

   dname = get(ppdisp,'name');
   kbd.which = uicontrol('style','text','horiz','center',...
         'string',['Data for ',dname]);

   kbd.inst = uicontrol('style','text','horiz','left',...
         'string','Enter the initial conditions:');

   kbd.xname = uicontrol('style','text',...
             'horiz','right','string',xnstr);

   kbd.xval = uicontrol('style','edit',...
            'string','','call','');

   kbd.yname = uicontrol('style','text',...
             'horiz','right',...
             'string',ynstr);

   kbd.yval = uicontrol('style','edit',...
            'string','');
```

```
kbd.tname = uicontrol('style','text',...
          'horiz','right',...
          'string',tnstr);

kbd.tval = uicontrol('style','edit');

kbd.t0 = uicontrol('style','edit');

kbd.tf = uicontrol('style','edit');

kbd.t = uicontrol('style','text','string','<= t<= ');

kbd.spec = uicontrol('style','check','horiz','center',...
         'string','Specify a computation interval.');

kbd.comp = uicontrol('style','push',...
         'string','Compute','call','pplane8(''kcompute'')');

kbd.close = uicontrol('style','push',...
          'string','Close','call','set(gcf,''vis'',''off'')');
kbd.fig = ppdisp;
left = 5; varl = 70; % buttw = 60;
frsep = 1;
nudge = 3;
xex = get(kbd.xname,'extent');
ht = xex(4)+nudge;
yex = get(kbd.yname,'extent');
nwdth = max(xex(3),yex(3)) + nudge;
varl = varl*ht/19;
fr1bot = 2*left + ht;
fr1ht = 4*nudge + 3*ht;
frw = 2*nudge + nwdth + varl;
fr1wind = [left,fr1bot,frw,fr1ht];
set(kbd.fr1,'pos',fr1wind);
tnb = fr1bot + nudge;
nl = left+nudge;
vl = nl + nwdth;
tvwind = [vl,tnb,varl,ht];
tnwind = [nl,tnb,nwdth,ht];
set(kbd.tname,'pos',tnwind);
set(kbd.tval,'pos',tvwind);
intext = get(kbd.t,'extent');
tw = intext(3);
margin = (frw -tw - 2*varl)/2;
t0l = left+margin;
intbot = tnb +ht + nudge;
t0wind = [t0l,intbot,varl,ht];
set(kbd.t0,'pos',t0wind);
tl = t0l + varl;
tfl = tl + tw;
twind = [tl,intbot,tw,ht];
tfwind = [tfl,intbot,varl,ht];
set(kbd.t,'pos',twind);
set(kbd.tf,'pos',tfwind);
```

```
specb = intbot + ht + nudge;
specw = frw - 2*nudge;
specwind = [nl,specb,specw,ht];
set(kbd.spec,'pos',specwind);
fr2bot = fr1bot + fr1ht +frsep;
ynb = fr2bot + nudge;
xnb = ynb + ht;
xnwind = [nl,xnb,nwdth,ht];
ynwind = [nl,ynb,nwdth,ht];
xvwind = [vl,xnb,varl,ht];
yvwind = [vl,ynb,varl,ht];
instb = xnb + ht + nudge;
instw = nwdth + varl;
instwind = [nl,instb,instw,ht];
fr2ht = 4*nudge + 3*ht;
fr3bot = fr2bot + fr2ht+frsep;
fr3ht = 2*nudge + ht;
frw = 2*nudge + nwdth + varl;
whichbot = fr3bot + nudge;
whichw = frw - 2*nudge;
whichwind = [nl,whichbot,whichw,ht];
fr3wind = [left,fr3bot,frw,fr3ht];
fr2wind = [left,fr2bot,frw,fr2ht];
figw = 2*left + frw;
fight = 3*left + ht + fr1ht + fr2ht + fr3ht;
figwind = [30,300,figw,fight];
buttw = (frw-left)/2;
closel = left;
compl = 2*left+buttw;
clwind = [closel,left,buttw,ht];
compwind = [compl,left,buttw,ht];
set(ppkbd,'pos',figwind);
set(kbd.fr2,'pos',fr2wind);
set(kbd.fr3,'pos',fr3wind);
set(kbd.which,'pos',whichwind);
set(kbd.inst,'pos',instwind);
set(kbd.xname,'pos',xnwind);
set(kbd.yname,'pos',ynwind);
set(kbd.xval,'pos',xvwind);
set(kbd.yval,'pos',yvwind);
set(kbd.comp,'pos',compwind);
set(kbd.close,'pos',clwind);
speccall = [
    'ud = get(gcf,''user'');',...
    'if get(gcbo,''value''),',...
    '  set([ud.t0,ud.t,ud.tf,ud.tname,ud.tval],''enable'',''on'');',...
    'else,',...
    '  set([ud.t0,ud.t,ud.tf,ud.tname,ud.tval],''enable'',''off'');',...
    'end'];

set(kbd.spec,'call',speccall);
set(ppkbd,'resize','on');
set(findobj(ppkbd,'type','uicontrol'),'units','normal');

set([kbd.tval,kbd.t0],'string','0');
set(kbd.spec,'value',0);
```

```
   set(ppkbd,'user',kbd,'vis','on');
   set([kbd.t0,kbd.t,kbd.tf,kbd.tname,kbd.tval],'enable','off')
   set(findobj(ppkbd,'type','uicontrol'),'units','normal');
   edith = findobj(ppkbd,'style','edit');
   set(edith,'backgroundcolor','w');
   figure(ppkbd)

elseif strcmp(action,'eqpt')

  % Find and classify equilibrium points.

  ppdisp = findobj('name','pplane8 Display');
  dud = get(ppdisp,'user');
  col = dud.color.eqpt;
  dfcn = dud.function;
  dbutt = dud.butt;
  menu = dud.menu;
  ppdispa =dud.axes;
  Dx = get(ppdispa,'xlim');
  Dy = get(ppdispa,'ylim');
  DY = [Dx(2)-Dx(1);Dy(2)-Dy(1)];

  epsilon = 1e-4;
  nstr = get(dud.notice,'string');
  nstr(1:4) = nstr(2:5);
  nstr{5} = 'Choose an approximation with the mouse.';
  set(dud.notice,'string',nstr);

%  z0 = ppginput(1);
  z0 = ginput(1);

  z = pplane8('newton',z0,dfcn);

  flag = z(:,4);
  A = z(:,2:3);
  B = real(A);
  k = find(abs(B)<1e-8);
  B(k) = zeros(size(k));
  C = imag(A);
  k = find(abs(C)<1e-8);
  C(k) = zeros(size(k));
  A = B+C*sqrt(-1);
  z = z(:,1);

  if (~flag | norm((z-z0')./DY) > 1/5)
    nstr(1:4) = nstr(2:5);
    nstr{5} = ['There is not an equilibrium point near (',...
           num2str(z0(1)),', ', num2str(z0(2)), ').  Ready'];
    set(dud.notice,'string',nstr);
    return
  end
  zero =find(abs(z) < epsilon);
  z(zero) = zeros(size(zero));
  D=det(A); T=trace(A);
  if (D<-epsilon)
```

```
    string1 = ['There is a saddle point at (',...
           num2str(z(1)),', ', num2str(z(2)), ').'];
    string2 = '';
    type = 1;
  elseif(D>epsilon)
    if (T*T>4*D+epsilon)
      if (T<0)
    string1 = ['There is a nodal sink at (',...
           num2str(z(1)),', ', num2str(z(2)), ').'];
    string2 = '';
    type = 2;
      else
    string1 = ['There is a nodal source at (',...
           num2str(z(1)),', ', num2str(z(2)), ').'];
    string2 = '';
    type = 3;
      end
    elseif(T*T<4*D-epsilon)
      if(T<-epsilon)
    string1 = ['There is a spiral sink at (',...
           num2str(z(1)),', ', num2str(z(2)), ').'];
    string2 = '';
    type = 4;
      elseif(T>epsilon)
    string1 = ['There is a spiral source at (',...
           num2str(z(1)),', ', num2str(z(2)), ').'];
    string2 = '';
    type = 5;
      else
    string1 = ['There is a spiral equilibrium point at (',...
           num2str(z(1)),', ', num2str(z(2)), ').'];
    string2 = 'Its specific type has not been determined.';
    type = 6;
      end
    else
      if (T>epsilon)
    string1 = ['There is a source at (',...
           num2str(z(1)),', ', num2str(z(2)), ').'];
    string2 = 'Its specific type has not been determined.';
    type = 7;
      elseif (T<-epsilon)
    string1 = ['There is a sink at (',...
           num2str(z(1)),', ', num2str(z(2)), ').'];
    string2 = 'Its specific type has not been determined.';
    type = 8;
      else
    string1 = ['There is an equilibrium point at (',...
           num2str(z(1)),', ', num2str(z(2)), ').'];
    string2 = 'Its specific type has not been determined.';
    type = 9;
      end
    end
  else
    string1 = ['There is an equilibrium point at (',...
           num2str(z(1)),', ', num2str(z(2)), ')'];
    string2 = 'Its specific type has not been determined.';
    type = 9;
  end

  [V,D] = eig(A);
  EqPtList = dud.eqpts;
  infostr{1,1} = string1;
  infostr{2,1} = string2;
  infostr{3,1} = ' ';
  infostr{4,1} = 'The Jacobian is:';
  infostr{5,1} = ['      ',num2str(A(1,1)),'   ',num2str(A(1,2))];
  infostr{6,1} = ['      ',num2str(A(2,1)),'   ',num2str(A(2,2))];
  infostr{7,1} = '';
  infostr{8,1} = 'The eigenvalues and eigenvectors are:';
  infostr{9,1} = ['     ',num2str(D(1,1)),'    (',num2str(V(1,1)),', 
',num2str(V(2,1)),')'];
  infostr{10,1} = ['     ',num2str(D(2,2)),'    (',num2str(V(1,2)),', 
',num2str(V(2,2)),')'];

  k=1; l=size(EqPtList,1);
  while ((k <= l))
    if (norm((EqPtList(k,1:2)-z')./DY') <= 1e-3)
      break;
    end
    k = k+1;
  end
  if (k > l)
    EqPtList = [EqPtList;z',type];
    dud.eqpts = EqPtList;
    newh = plot(z(1),z(2),...
        'color',col,...
        'markersize',20,...
        'marker','.','Erasemode','none');
    dud.ephand = [dud.ephand;newh];
    set(newh,'Erasemode','normal');
    drawnow
  end
  ppeqpt = findobj('name','pplane8 Equilibrium point data');

   if (isempty(ppeqpt))
     ppeqpt = figure('vis','off','resize','on',...
            'name','pplane8 Equilibrium point data',...
            'numb','off','tag','pplane8');
     pplane8('figdefault',ppeqpt);
     set(ppeqpt,'menubar','none');

     ud.frame = uicontrol('style','frame');
     ud.eptext = uicontrol('style','text','string',infostr,'hor','left');
     ud.goaway = uicontrol('style','push','string','Go away',...
              'call','close');
     ud.display = uicontrol('style','push',...
               'string','Display the linearization',...
               'call','pplane8(''linear'')');
   else
     figure(ppeqpt);
     ud = get(ppeqpt,'user');
     set(ppeqpt,'vis','off');
     set(findobj(ppeqpt,'type','uicontrol'),'units','pixels');
     set(ud.eptext,'string',infostr);
   end
   ud.jac = A;
   ud.vectors = V;
   ud.type = type;
   ud.system = dud.syst;
   ud.settings = dud.settings;
   ud.color = dud.color;
   set(ppeqpt,'user',ud);
   left = 5; nudge = 3;
   ext = get(ud.eptext,'extent');
   n = size(infostr,1);
   ht = ext(4)/n+2*nudge;
   frbot = 3*left + 2*ht;
   txtbot = frbot + nudge;
   txtl = left + nudge;
   twind = [txtl,txtbot,ext(3)+nudge,ext(4)];
   frw = ext(3) + 3*nudge;
   frh = ext(4) + 2*nudge;
   frwind = [left,frbot,frw,frh];
   figw = frw + 2*left;
   figh = frh + 4*left + 2*ht;
   uni = get(0,'units');
   set(0,'units','pixels');
   ss = get(0,'screensize');
   set(0,'units',uni);
   sh = ss(4);
   figbot = sh - figh - 350;
   figwind = [30,figbot,figw,figh];
   buttw = frw - 4;
   buttl = left + 2;
   closewind = [buttl,left,buttw,ht];
   dispb = 2*left + ht;
   dispwind = [buttl,dispb,buttw,ht];
   set(ppeqpt,'pos',figwind);
   set(ud.frame,'pos',frwind);
   set(ud.eptext,'pos',twind);
   set(ud.goaway,'pos',closewind);
   set(ud.display,'pos',dispwind);
   set(findobj(ppeqpt,'type','uicontrol'),'units','normal');
   set(ppeqpt,'vis','on');
   set(get(ppeqpt,'child'),'vis','on');
   nstr(1:4) = nstr(2:5);
   nstr{5} = 'Ready.';
   set(dud.notice,'string',nstr);
   set(ppdisp,'user',dud);

elseif strcmp(action,'stunst')

   ppdisp = findobj('name','pplane8 Display');
   dud = get(ppdisp,'user');
   ecol = dud.color.eqpt;
   scol = dud.color.sep;
   ppdispa = dud.axes;
   aud = get(ppdispa,'user');
   DY = aud.DY;
   settings = dud.settings;
   dfcn = dud.function;
   EqPtList = dud.eqpts;
   Stop = norm(DY)*1e-4;

   %  Plot the stable and unstable orbits at a saddle point.

   nstr = get(dud.notice,'string');
   nstr(1:4) = nstr(2:5);
   nstr{5} = 'Choose a saddle point with the mouse.';
   set(dud.notice,'string',nstr);
%    z0 = ppginput(1);
   z0 = ginput(1);
   z = zeros(2,1);
   z = pplane8('newton',z0,dfcn);

   flag = z(:,4);
   A = z(:,2:3);
   k = find(abs(A)<1e-8);
   A(k) = zeros(size(k));
   z = z(:,1);

   if (~flag | norm((z-z0')./DY)> 1/5)
      nstr(1:4) = nstr(2:5);
      nstr{5} = ['There is not an equilibrium point near (',...
            num2str(z0(1),2),', ', num2str(z0(2),2), ').'];
      set(dud.notice,'string',nstr);
      return
   end
   zero = find(abs(z) < 1e-4);
   z(zero) = zeros(size(zero));
   D=det(A);
   if (D>=0)
      nstr(1:4) = nstr(2:5);
      nstr{5} = ['The equilibrium point at (',...
            num2str(z(1),2),', ', num2str(z(2),2),...
            ') is not a saddle point.'];
      set(dud.notice,'string',nstr);
      return
   end
   if isempty(EqPtList)
      EqPtList = [z',1];
      dud.eqpts = EqPtList;
   else
      k=1; l=size(EqPtList,1);
      while ((k <= l))
         if (norm(EqPtList(k,1:2)-z') <= Stop)
            break;
         end
         k = k+1;
      end
      if (k > l)
         EqPtList = [EqPtList;z',1];
         dud.eqpts = EqPtList;
      end
   end
```

```
nstr(1:4) = nstr(2:5);
nstr{5} = ['Plotting from the saddle point at (',...
      num2str(z(1),2),', ', num2str(z(2),2),').'];
set(dud.notice,'string',nstr);
[V,L] = eig(A);
[L,I] = sort(diag(L));

magn = settings.magn;
refine = settings.refine;
ssize = settings.stepsize;
solver = settings.solver;
tol = settings.tol;
offset = norm(DY)/1000;
lm=abs(L);
lm=lm/max(lm);
lm = (max([lm,0.2*ones(size(lm))]'))';
offs = offset./lm;
ud = get(dud.axes,'user');
atol = tol*ud.DY*1e-4';
options = odeset('OutputFcn',@ppout,'Refine',refine,...
    'RelTol',tol,'Abstol',atol);
stopbutt = findobj('tag','stop');
set(stopbutt,'vis','on','enable','on');
VV=V(:,I(1));
newhand = zeros(4,1);
w = z + offs(1)*VV;
nstr = get(dud.notice,'string');
nstr(1:4) = nstr(2:5);
nstr{5} = 'The first stable trajectory';
set(dud.notice,'string',nstr);
int = [0,-1e6];

solver = settings.solver;
switch solver
 case 'Dormand Prince'
  solh = @ppdp45;
  opt = ppdisp;
 case 'Runge-Kutta 4'
  solh = @pprk4;
  opt = ppdisp;
 case 'ode45'
  solh = @ode45;
  opt = odeset('OutputFcn',@ppout,'Refine',refine,...
           'RelTol',tol,'Abstol',atol);
 case 'ode23'
  solh = @ode23;
  opt = odeset('OutputFcn',@ppout,'Refine',refine,...
           'RelTol',tol,'Abstol',atol);
 case 'ode113'
  solh = @ode113;
  opt = odeset('OutputFcn',@ppout,'Refine',refine,...
           'RelTol',tol,'Abstol',atol);
 case 'ode15s'
  solh = @ode15s;
  opt = odeset('OutputFcn',@ppout,'Refine',refine,...
           'RelTol',tol,'Abstol',atol);
    case 'ode23s'
     solh = @ode23s;
     opt = odeset('OutputFcn',@ppout,'Refine',refine,...
              'RelTol',tol,'Abstol',atol);
    case 'ode23t'
     solh = @ode23t;
     opt = odeset('OutputFcn',@ppout,'Refine',refine,...
              'RelTol',tol,'Abstol',atol);
    case 'ode23tb'
     solh = @ode23tb;
     opt = odeset('OutputFcn',@ppout,'Refine',refine,...
              'RelTol',tol,'Abstol',atol);
   end
   exist(dfcn);
   dfh = str2func(dfcn);
   cflag = 0;
   [tp,Xp] = feval(solh,dfh,int,w,opt);



%   [tp,Xp] = ppdp45(dfcn,[0,-1e6],w,ppdisp);
   set(stopbutt,'enable','off');
   aud = get(ppdispa,'user');
   newhand(1) = aud.line;
   X1=[z';Xp];
   x1 = [X1,[NaN;tp]];
   set(newhand(1),'xdata',x1(:,1),'ydata',x1(:,2),'zdata',x1(:,3));

   w = z - offs(1)*VV;
   nstr = get(dud.notice,'string');
   nstr(1:4) = nstr(2:5);
   nstr{5} = 'The second stable trajectory';
   set(dud.notice,'string',nstr);
   set(stopbutt,'enable','on');
   [tp,Xp] = feval(solh,dfh,int,w,opt);

%   [tp,Xp] = ppdp45(dfcn,[0,-1e6],w,ppdisp);
   set(stopbutt,'enable','off');

   aud = get(ppdispa,'user');
   newhand(2) = aud.line;

   X2=[z';Xp];
   x2 = [X2,[NaN;tp]];
   set(newhand(2),'xdata',x2(:,1),'ydata',x2(:,2),'zdata',x2(:,3));

   VV=V(:,I(2));
   w = z + offs(2)*VV;
   nstr = get(dud.notice,'string');
   nstr(1:4) = nstr(2:5);
   nstr{5} = 'The first unstable trajectory';
   set(dud.notice,'string',nstr);
   set(stopbutt,'enable','on');
   int = [0,1e6];
   [tp,Xp] = feval(solh,dfh,int,w,opt);
   set(stopbutt,'enable','off');

   aud = get(ppdispa,'user');
   newhand(3) = aud.line;
   X3=[z';Xp];
   x3 = [X3,[NaN;tp]];
   set(newhand(3),'xdata',x3(:,1),'ydata',x3(:,2),'zdata',x3(:,3));

   w = z - offs(2)*VV;
   nstr = get(dud.notice,'string');
   nstr(1:4) = nstr(2:5);
   nstr{5} = 'The second unstable trajectory';
   set(dud.notice,'string',nstr);
   set(stopbutt,'enable','on');
   [tp,Xp] = feval(solh,dfh,int,w,opt);

   set(stopbutt,'vis','off');

   aud = get(ppdispa,'user');
   newhand(4) = aud.line;
   X4=[z';Xp];
   x4 = [X4,[NaN;tp]];
   set(newhand(4),'xdata',x4(:,1),'ydata',x4(:,2),'zdata',x4(:,3));

   eqpt = plot(z(1),z(2),...
           'color',ecol,...
           'markersize',20,...
           'marker','.','Erasemode','normal');
   dud.solhand = [dud.solhand;newhand(:)];
   dud.ephand = [dud.ephand;eqpt];
   set(newhand,'color',scol);
   set(newhand,'erase','normal');
   nstr = get(dud.notice,'string');
   nstr(1:4) = nstr(2:5);
   nstr{5} = 'Ready.';
   set(dud.notice,'string',nstr);
   set(gcf,'user',dud);

elseif strcmp(action,'zoomin')

   % 'zoomin' is the callback for the Zoomin menu item.  It allows the
   % user to pick a new display rectangle.

   set(gcf,'WindowButtonDownFcn','pplane8(''zoom'')',...
      'WindowButtonUpFcn','1;','inter','on');
   set(gca,'inter','on');
   dud = get(gcf,'user');
   nstr = get(dud.notice,'string');
   nstr(1:4) = nstr(2:5);
   nstr{5} = ['Pick a new display rectangle by clicking and ',...
         'dragging the mouse, or by clicking on a point.'];
   set(dud.notice,'string',nstr);

elseif strcmp(action,'zoom')
```

```
   disph = gcf;
   dud = get(disph,'user');
   axh = dud.axes;
   aud = get(axh,'user');
   DY = aud.DY;
   w = dud.syst.wind;
   q1 = get(disph,'currentpoint');
   p1 = get(axh,'currentpoint');
   p1 = p1(1,1:2);
   rbbox([q1 0 0],q1);
   p2 = get(axh,'currentpoint');
   p2 = p2(1,1:2);
   if all(abs(p1'-p2')>0.01*DY)
      a = [p1;p2];
      a = [min(a);max(a)];
      DY = (a(2,:) - a(1,:))';
   else
      DY = DY/4;
      a(1) = max(w(1),p1(1)-DY(1));
      a(2) = min(w(2),p1(1)+DY(1));
      a(3) = max(w(3),p1(2)-DY(2));
      a(4) = min(w(4),p1(2)+DY(2));
      DY(1) = a(2) - a(1);
      DY(2) = a(4) - a(3);
   end
   WINvect = a(:)';
   dud.syst.wind = WINvect;
   aud.DY = DY;
   dwind = [WINvect(1); WINvect(3); -WINvect(2); -WINvect(4)];
   aud.cwind = dwind - dud.settings.magn*[aud.DY;aud.DY];
   set(axh,'user',aud);
   set(disph,'user',dud);
   set(disph,'WindowButtonDownFcn','pplane8(''down'')',...
      'WindowButtonUpFcn','');
   pplane8('dirfield',disph);
   ppset = findobj('name','pplane8 Setup');
   if isempty(ppset)
      pplane8('confused');
   else
      sud = get(ppset,'user');
      sud.c.wind = WINvect;
      sud.o.wind = WINvect;
      set(sud.h.wind(1),'string',num2str(WINvect(1)));
      set(sud.h.wind(2),'string',num2str(WINvect(2)));
      set(sud.h.wind(3),'string',num2str(WINvect(3)));
      set(sud.h.wind(4),'string',num2str(WINvect(4)));
      set(ppset,'user',sud);
   end

elseif strcmp(action,'showbar')

  sbfig = gcbf;
  domymenu('menubar','toggletoolbar',sbfig);
  hhset = get(0,'showhiddenhandles');
  set(0,'showhiddenhandles','on');
  state = get(sbfig,'toolbar');
  if strcmp(state,'figure')
    name = get(sbfig,'name');
    fixtb = ['set(gcbo,''state'',''off'');'];
    set(findobj(sbfig,'tooltipstr','Print'),...
    'clickedcallback','pplane8(''print'');');
    switch name
     case 'pplane8 Display'
      set(findobj(sbfig,'tooltipstr','Zoom Out'),...
      'clickedcallback',['pplane8(''zoomback'');' fixtb]);
      set(findobj(sbfig,'tooltipstr','Zoom In'),...
      'clickedcallback',['pplane8(''zoomin'');' fixtb]);
      delete(findobj(sbfig,'tooltipstr','Rotate 3D'));

     case 'pplane8 Linearization'
      delete(findobj(sbfig,'tooltipstr','Zoom Out'));
      delete(findobj(sbfig,'tooltipstr','Zoom In'));
      delete(findobj(sbfig,'tooltipstr','Rotate 3D'));
     otherwise
      delete(findobj(sbfig,'tooltipstr','Zoom Out'));
      delete(findobj(sbfig,'tooltipstr','Zoom In'));
      fud = get(sbfig,'user');
      switch fud.type
       case {4, 5}
    set(findobj(sbfig,'tooltipstr','Rotate 3D'),'vis','on');
       otherwise
    set(findobj(sbfig,'tooltipstr','Rotate 3D'),'vis','off');
      end

    end
    sbmh = findobj(sbfig,'label','Show &Toolbar');
    set(sbmh,'label','Hide &Toolbar','checked','off');
    uni = get(0,'units');
    ss = get(0,'screensize');
    funit = get(sbfig,'units');
    set(sbfig,'units',uni);
    sw = ss(3);sh = ss(4);
    fpos = get(sbfig,'pos');
    if fpos(2)+fpos(4)>sh-40;
      fpos(2) = sh - fpos(4) -70;
      set(sbfig,'pos',fpos);
    end
    set(sbfig,'units',funit);

  else
    sbmh = findobj(sbfig,'label','Hide &Toolbar');
    set(sbmh,'label','Show &Toolbar','checked','off');
  end
  set(0,'showhiddenhandles',hhset)

elseif strcmp(action,'dall')

   % 'dall' is the callback for the Erase all graphics objects.

   disph = gcf;
   dud = get(disph,'user');
   notice = dud.notice;
   kk = find(ishandle(dud.solhand));
   delete(dud.solhand(kk));
   dud.solhand = [];
   kk = find(ishandle(dud.ephand));
   delete(dud.ephand(kk));
   dud.ephand = [];
   dud.eqpts = [];
   kk = find(ishandle(dud.contours));
   delete(dud.contours(kk));
   dud.contours = [];
   kk = find(ishandle(dud.ics));
   delete(dud.ics(kk));
   dud.ics = [];
   if notice
      set(dud.butt(1),'enable','off');
   end
   set(disph,'user',dud);


elseif strcmp(action,'dallsol')

   % 'dallsol' is the callback for the Erase all solutions option.

   disph = gcf;
   dud = get(disph,'user');
   notice = dud.notice;
   kk = find(ishandle(dud.solhand));
   delete(dud.solhand(kk));
   dud.solhand = [];
   if notice
      set(dud.butt(1),'enable','off');
   end
   set(disph,'user',dud);

elseif strcmp(action,'dallep')

   % 'dallep' is the callback for the Erase all equilibrium points option.

   disph = gcf;
   dud = get(disph,'user');
   notice = dud.notice;
   kk = find(ishandle(dud.ephand));
   delete(dud.ephand(kk));
   dud.ephand = [];
   dud.eqpts = [];
   if notice
      set(dud.butt(1),'enable','off');
   end
   set(disph,'user',dud);

elseif strcmp(action,'dalllev')

   % 'dalllev' is the callback for the Erase all level curves option.

   disph = gcf;
   dud = get(disph,'user');
   notice = dud.notice;
   kk = find(ishandle(dud.contours));
   delete(dud.contours(kk));
   dud.contours = [];
   if notice
      set(dud.butt(1),'enable','off');
   end
   set(disph,'user',dud);

elseif strcmp(action,'dallics')

   % 'dallics' is the callback for the Erase all marked initial cond.

   disph = gcf;
   dud = get(disph,'user');
   notice = dud.notice;
   kk = find(ishandle(dud.ics));
   delete(dud.ics(kk));
   dud.ics = [];
   if notice
      set(dud.butt(1),'enable','off');
   end
   set(disph,'user',dud);

elseif strcmp(action,'newton')

   % Newton's method to find equilibrium points.

   Iterlimit = 50;

   Iter=0;

   % The increment for calculating approximate derivatives.

   h=.000001;

   dfcn = input2;

   zInit = input1;

   zNext=zInit(:);
   functionf = zeros(2,1);
   functionf=feval(dfcn,0,zNext);
   functionf=functionf(:);

   % Allow for large/small solutions.

   errorlim = norm(functionf,inf)*0.000001;

   while ( (norm(functionf,inf) > errorlim) & (Iter < Iterlimit)    )
      Iter = Iter + 1;

      % Now we calculate the jacobian.
      for jjw=1:2
         sav = zNext(jjw);
         zNext(jjw) = zNext(jjw) + h;
         functionfhj = feval(dfcn,0,zNext);
         functionfhj = functionfhj(:);
         Jacobian(:,jjw) = (functionfhj-functionf)/h;
         zNext(jjw) = sav;
      end
      zNext = zNext - Jacobian\(functionf);
      functionf = feval(dfcn,0,zNext);
      functionf=functionf(:);
   end

   if Iter > Iterlimit - 1
      fLag=[0;0];
   else
      fLag=[1;1];
      for j=1:2
         sav = zNext(j);
         zNext(j) = zNext(j) + h;
         functionfhj = feval(dfcn,0,zNext);
         functionfhj = functionfhj(:);
         Jacobian(:,j) = (functionfhj-functionf)/h;
         zNext(j) = sav;
      end
   end
   output = [zNext,Jacobian,fLag];

elseif strcmp(action,'paraeval')

   %    Replace a parameter with its value in string form.

   para = deblank(input1);
   value = input2;
   value = ['(',value,')'];
   str = input3;
   ll = length(str);
   lp = length(para);
   lv = length(value);

   if strcmp(para,str)
      str = value
   elseif (ll >= lp+1)
      k = findstr(para,str);

      lk = length(k);
      lopstr = '(+-*/^';
      ropstr = ')+-*/^';
      s = [];
      pos = 1;
      for jj = 1:lk
    if (((k(jj) == 1)|(find(lopstr == str(k(jj)-1))))...
        &((k(jj)+lp-1 == ll)|(find(ropstr == str(k(jj) + lp))))
      s = [s,str(pos:(k(jj)-1)),value];
      pos = k(jj)+lp;
    end
      end
      str = [s,str(pos:ll)];
   end
   output = str;


elseif strcmp(action,'settings')

   % 'settings' is the call back for the Settings menu option.  It sets
   % up the pplane8 Settings window, which allows the user to interactively
   % change several parameters that govern the behaviour of the program.

   dud = get(gcf,'user');
   data.settings = dud.settings;
   solver = dud.settings.solver;
   left = 2; nudge = 3; varl = 60;
   setfig = findobj('name','pplane8 Settings');
   if ~isempty(setfig);
     delete(setfig);
   end

   setfig = figure('name','pplane8 Settings',...
           'numb','off',...
           'tag','pplane8','vis','off');

   pplane8('figdefault',setfig);
   set(setfig,'menubar','none');

   setfr = uicontrol('style','frame');
   speedfr = uicontrol('style','frame');

   cwfr = uicontrol('style','frame');
   ss = uicontrol('style','text','horiz','center',...
      'string',['Settings for ',dud.settings.solver]);

   nstepcall =[
      'data = get(gcf,''user'');',...
         'ss = max(round(str2num(get(data.nstep,''string''))),0);',...
         'data.settings.refine = ss;',...
         'set(data.nstep,''string'',num2str(ss));',...
         'set(gcf,''user'',data);'];

   nstep = uicontrol('style','text','horiz','left',...
      'string','Number of plot steps per computation step:  ');
   data.nstep = uicontrol('style','edit','call',nstepcall,...
      'string',num2str(data.settings.refine));

   if strcmp(solver,'Runge-Kutta 4')
     rtolstr = 'Step size: ';
     rtolentry = num2str(data.settings.stepsize);
     rtolcall =['data = get(gcf,''user'');',...
       'data.settings.stepsize = str2num(get(data.rtol,''string''));',...
       'set(data.rtol,''string'',num2str(data.settings.stepsize));',...
       'set(gcf,''user'',data);'];
   else
     rtolstr = 'Relative error tolerance: ';
     rtolentry = num2str(data.settings.tol);
     rtolcall =['data = get(gcf,''user'');',...
       'data.settings.tol = str2num(get(data.rtol,''string''));',...
       'set(data.rtol,''string'',num2str(data.settings.tol));',...
       'set(gcf,''user'',data);'];
   end
   rtol = uicontrol('style','text','horiz','left',...
       'string',rtolstr);
   data.rtol = uicontrol('style','edit','call',rtolcall,...
      'string',rtolentry);

   kk = 1+2*data.settings.magn;
   magcall = ['data = get(gcf,''user'');',...
         'mag =     (str2num(get(data.mag,''string''))-1)/2;',...
         'data.settings.magn = max(0,mag);',...
         'set(gcf,''user'',data);'];

    speedcall = ['data = get(gcf,''user'');',...
         'me = data.speed;',...
         'val = round(get(me,''value''));',...
         'set(me,''value'',val);',...
         'set(data.sp.val,''string'',num2str(val));',...
         'data.settings.speed = val;',...
         'set(gcf,''user'',data);'];
    data.speed = uicontrol('style','slider',...
               'string','Steps per second.',...
               'min',2,...
               'max',100,...
               'value',data.settings.speed,...
               'sliderstep',[1/98,10/98],...
               'call',speedcall);

    data.sp.min = uicontrol('style','text','string',' 2',...
                'horiz','left');
    data.sp.max = uicontrol('style','text','string','100 ',...
                'horiz','right');
    data.sp.val = uicontrol('style','text',...
                'string',num2str(data.settings.speed),...
                'horiz','center');

    pps = uicontrol('style','text',...
            'string','Solution steps per second.');




   cw1 = uicontrol('style','text','horiz','left',...
      'string','The calculation window is');
   data.mag = uicontrol('style','edit','call',magcall,...
      'string',num2str(kk));
   cw2 = uicontrol('style','text','horiz','left',...
      'string',' times as large as the ');
```

```
cw3 = uicontrol('style','text','horiz','left',...
   'string','display window.');

gob = uicontrol('style','push','string','Go Away','call','close');

chb = uicontrol('style','push',...
   'string','Change settings',...
   'call','pplane8(''setchange'')');

frsep = 1;
ext = get(nstep,'extent');
ht = ext(4)+nudge;
stw = ext(3);        % = nstepw = rtolw
varl = varl*ht/19;
cwfrb = 2*left + ht;
cwfrh = 2*nudge + 2*ht;
speedfrb = cwfrb + cwfrh + frsep;
speedfrh = 4*nudge + 3*ht;
setfrb = speedfrb + speedfrh + frsep;
setfrh = 3*nudge + 3*ht;
figh = setfrb + setfrh + left;
bb = left;
cw3b = cwfrb + nudge;
cw2b = cw3b + ht;    % = cw1b = cweb
rtolb = setfrb + nudge;
rtolbb = rtolb;
if strcmp(solver,'Runge-Kutta 4')
  rtolbb = rtolbb +ht/2;
end
nstepb = rtolb + ht;
ssb = nstepb + ht + nudge;
ssw = stw + varl;
spb1 = speedfrb+2*nudge;
spb2 = spb1+ht;
spb3 = spb2+ht+nudge;
sptw = varl;
sl = left + nudge;
sptl1 = sl;
sptsep = (ssw-3*sptw-sl)/2;
sptl2 = sptl1 + sptw + sptsep;
sptl3 = sptl2 + sptw + sptsep;

cw1ext = get(cw1,'extent');
cw1w = cw1ext(3);
cw2ext = get(cw2,'extent');
cw2w = cw2ext(3);
cwew = 40*ht/19;
frw = 2*nudge + ssw;
figw = frw + 2*left;
buttw = figw/3;
sep = figw/9;
sl = left + nudge;
gobl = sep;
chbl = 2*sep + buttw;

sunit = get(0,'units');
```

```
   set(0,'units','pix');
   ssize = get(0,'screensize');
   figb = ssize(4) - figh - 50;

   set(setfig,'pos',[20,figb,figw,figh]);
   set(setfr,'pos',[left,setfrb,frw,setfrh]);
   set(speedfr,'pos',[left,speedfrb,frw,speedfrh]);
   set(cwfr,'pos',[left,cwfrb,frw,cwfrh]);
   set(ss,'pos',[sl,ssb,ssw,ht]);
   set(nstep,'pos',[sl,nstepb,stw,ht]);
   set(data.nstep,'pos',[sl+stw,nstepb,varl,ht]);
   set(rtol,'pos',[sl,rtolbb,stw,ht]);
   set(data.rtol,'pos',[sl+stw,rtolbb,varl,ht]);

   set(data.speed,'pos',[sl,spb1,ssw,ht],...
          'backgroundcolor',0.6*[1 1 1],...
          'foregroundcolor','r');
   set(data.sp.min,'pos',[sptl1,spb2,sptw,ht]);
   set(data.sp.max,'pos',[sptl3,spb2,sptw,ht]);
   set(data.sp.val,'pos',[sptl2,spb2,sptw,ht]);
   set(pps,'pos',[sl,spb3,ssw,ht]);

   set(cw1,'pos',[sl,cw2b,cw1w,ht]);
   set(cw2,'pos',[sl+cw1w+cwew,cw2b,cw2w,ht]);
   set(cw3,'pos',[sl,cw3b,ssw,ht]);
   set(data.mag,'pos',[sl+cw1w,cw2b,cwew,ht]);
   set(gob,'pos',[gobl,bb,buttw,ht]);
   set(chb,'pos',[chbl,bb,buttw,ht]);


   set(setfig,'user',data);
   set(setfig,'units','normal');
   set(findobj(setfig,'type','uicontrol'),'units','normal');
   set(setfig,'vis','on','resize','on');
   ch = get(setfig,'child');
   set([nstep,data.nstep],'vis','off')
   if strcmp(solver,'Runge-Kutta 4')
     ch = setdiff(ch,[nstep,data.nstep]);
   end
   edith = findobj(setfig,'style','edit');
   set(edith,'backgroundcolor','w');
   set(ch,'vis','on');


elseif strcmp(action,'setchange')

   % 'setchange' is the callback for the Change button
   % on the pplane8 Settings window.

   data = get(gcf,'user');
   settings = data.settings;
   ppdisp = findobj('name','pplane8 Display');
   dud = get(ppdisp,'user');
   if isempty(ppdisp)
      pplane8('confused');
   else
      dud.settings = settings;
      set(ppdisp,'user',dud);
      WINvect = dud.wmat;
      WINvect = WINvect(size(WINvect,1),:);
      dwind = [WINvect(1); WINvect(3); -WINvect(2); -WINvect(4)];
      aud = get(dud.axes,'user');
      aud.cwind = dwind - dud.settings.magn*[aud.DY;aud.DY];
      set(dud.axes,'user',aud);
   end
   ppset = findobj('name','pplane8 Setup');
   if isempty(ppset)
      pplane8('confused');
   else
      sud = get(ppset,'user');
      sud.settings = settings;
      set(ppset,'user',sud);
   end
   close


elseif strcmp(action,'delete')

   % 'delete' is the callback for the Delete a graphics object selection
   % on the menu.

   disph = gcf;
   dud = get(disph,'user');
   arr = dud.arr;
   lv = get(arr.lines,'vis');
   av = get(arr.arrows,'vis');
   if ~isempty(arr.hx)
      nv = get(arr.hx(1),'vis');
   elseif ~isempty(arr.hy)
      nv = get(arr.hx(1),'vis');
   else
      nv = zeros(1,0);
   end
   if ~isempty(arr.barrows)
      bv = get(arr.barrows,'vis');
   else
      bv = zeros(1,0);
   end
   handles = [arr.lines;arr.arrows;arr.hx;arr.hy;arr.barrows];
   set(handles,'vis','off');
   oldcall = get(disph,'WindowButtonDownFcn');
   set(disph,'WindowButtonDownFcn','');
   trjh = dud.solhand;
   notice = dud.notice;
   if notice   % Display window
      nstr = get(notice,'string');
      nstr(1:4) = nstr(2:5);
      nstr{5} = 'Select a graphics object with the mouse.';
      set(notice,'string',nstr);
   end
%   ppginput(1);
```

```
ginput(1);
objh = get(disph,'currentobject');
typ = get(objh,'type');
axh = dud.axes;
hh = get(dud.title.axes,'children');
hh = [hh;get(axh,'title');get(axh,'xlabel');get(axh,'ylabel')];
if notice   % Display window
  eph = dud.ephand;
  levh = dud.contours;
  eqpt = dud.eqpts;
  ics = dud.ics;
  nstr(1:4) = nstr(2:5);
  if strcmp(typ,'line')
    eqk = find(eph == objh);
    if ~isempty(eqk);  % An equilibrium point.
  if ~isempty(eqk)
    if size(eqpt,1) == 1;
      dud.eqpts = [];
    else
      dud.eqpts = [eqpt(1:eqk-1,:);eqpt(eqk+1:size(eqpt,1),:)];
    end
  end
    end
    dud.ics = setdiff(dud.ics,objh);
    dud.solhand = setdiff(dud.solhand,objh);
    dud.ephand = setdiff(dud.ephand,objh);
    dud.contours = setdiff(dud.contours,objh);
    delete(objh);
    nstr{5} = 'Ready.';

  elseif strcmp(typ,'text') & ~ismember(objh,hh)
    dud.contours = setdiff(dud.contours,objh)';
    delete(objh);
    nstr{5} = 'Ready.';

  else
    nstr{5} = 'The object you selected cannot be deleted.';

  end
  set(notice,'string',nstr);
else     % Linearization window
  if strcmp(typ,'line')
    eqk = find(dud.ephand == objh);
    if isempty(eqk);  % Npt an equilibrium point.
  dud.solhand = setdiff(dud.solhand,objh)';
  dud.ics = setdiff(dud.ics,objh);
  delete(objh);
    end
  elseif ~ismember(objh,hh)
    delete(objh);
  end
end

set(arr.lines,'vis',lv);
set(arr.arrows,'vis',av);
set([arr.hx;arr.hy],'vis',nv);
```

```
   set(arr.barrows,'vis',bv);
   set(disph,'user',dud);
   set(disph,'WindowButtonDownFcn','pplane8(''down'')');

elseif strcmp(action,'text')

   ppdisp = gcf;
   oldcall = get(ppdisp,'WindowButtonDownFcn');
   set(ppdisp,'WindowButtonDownFcn','');
   prompt = ['Enter the text here. Then choose ',...
         'the location in the Display Window.'];
   txtstr = inputdlg(prompt,'Text entry');
   if ~isempty(txtstr)
      txtstr = txtstr{1};
      figure(ppdisp);
      gtext(txtstr);
   end
   set(ppdisp,'WindowButtonDownFcn',oldcall);

elseif strcmp(action,'plotxy')   % Start a graph

  type = input1;

  ppdisp =gcf;
  dud = get(ppdisp,'user');

  handles = dud.solhand;
  printstr = dud.printstr;
  exsol=0;
  lll = length(handles);

  if (lll)   % Are there any orbits?
    kk=0;
    while (kk < lll) & (exsol == 0)
      kk = kk + 1;
      xd = get(handles(kk),'xdata');
      if (length(xd) > 7)
    exsol = 1;
      end
    end
  end
  if (exsol == 0)
    nstr = get(dud.notice,'string');
    nstr(1:3) = nstr(3:5);
    nstr{4} = 'There are no solution curves.';
    nstr{5} = 'Ready.';
    set(dud.notice,'string',nstr);
    return
  end

  oldcall = get(ppdisp,'WindowButtonDownFcn');
  set(ppdisp,'WindowButtonDownFcn','');
  arr = dud.arr;
  lv = get(arr.lines,'vis');
  av = get(arr.arrows,'vis');
```

```
if ~isempty(arr.hx)
  nv = get(arr.hx(1),'vis');
elseif ~isempty(arr.hy)
  nv = get(arr.hy(1),'vis');
else
  nv = zeros(1,0);
end
if ~isempty(arr.barrows)
  bv = get(arr.barrows,'vis');
else
  bv = zeros(1,0);
end
handles = [arr.lines;arr.arrows;arr.hx;arr.hy;arr.barrows];
set(handles,'vis','off');

nstr = get(dud.notice,'string');
nstr(1:4) = nstr(2:5);
nstr{5} = 'Select a solution curve with the mouse.';
set(dud.notice,'string',nstr);
% ppginput(1);
  ginput(1);
objh = get(ppdisp,'currentobject');
if isempty(objh)
  solflag = 0;
elseif ~(strcmp(get(objh,'type'),'line'))
  solflag = 0;
else
  t=get(objh,'zdata');
  if length(t)<10
    solflag = 0;
  else
    solflag = 1;
  end
end
if solflag == 0
  nstr = get(dud.notice,'string');
  nstr(1:4) = nstr(2:5);
  nstr{5} = 'This object is not a solution curve.';
  set(dud.notice,'string',nstr);
  set(ppdisp,'WindowButtonDownFcn',oldcall);
  set(arr.lines,'vis',lv);
  set(arr.arrows,'vis',av);
  set([arr.hx;arr.hy],'vis',nv);
  return
end
set(arr.lines,'vis',lv);
set(arr.arrows,'vis',av);
set([arr.hx;arr.hy],'vis',nv);
set(arr.barrows,'vis',bv);
t=get(objh,'zdata');
x=get(objh,'xdata');
y=get(objh,'ydata');
hh = pplane8('plotxyfig',type,gcf);
set(ppdisp,'WindowButtonDownFcn',oldcall);
aud = get(hh(2),'user');
ud = get(hh(1),'user');
ud.data = [t;x;y];
```

```
  set(hh(1),'user',ud);
  pplane8('plxy',aud.rad,hh(1));

elseif strcmp(action,'plotxyfig')  % Build the graph figure.

  type = input1;
  cfig = input2;  % This is ppdisp or a ppxy
  ppdisp = findobj('name','pplane8 Display');
  dud = get(ppdisp,'user');
  ppdispa = dud.axes;
  xstring = get(get(ppdispa,'XLabel'),'string');
  ystring = get(get(ppdispa,'YLabel'),'string');
  dud = get(ppdisp,'user');

  ppxy = figure('number','off',...
        'tag','pplane8','visible','off');
  name = ['pplane8 t-plot # ',int2str(ppxy)];
  ud.color = dud.color;
  ud.type = 0;
  set(ppxy,'name',name,'user',ud);
  pplane8('figdefault',ppxy);

  % Menus and Toolbar

  hhsetup = get(0,'showhiddenhandles');
  set(0,'showhiddenhandles','on');

  % Configure the Toolbar.

  set(ppxy,'ToolBar','none');

  % Menus

  emenu = findobj(gcf,'label','&Edit');
  tmenu = findobj(gcf,'label','&Tools');
  delete(tmenu)

  % File menu

  fmenu = findobj(gcf,'label','&File');
  delete(findobj(fmenu,'label','&New Figure'));
  delete(findobj(fmenu,'label','&Open...'));
  set(findobj(fmenu,'label','&Close'),'pos',1);
  set(findobj(fmenu,'label','&Save'),...
      'pos',2,'separator','off');
  set(findobj(fmenu,'label','Save &As...'),...
      'pos',3);
  set(findobj(fmenu,'label','&Export...'),...
      'pos',4);
  delete(findobj(fmenu,'label','Pre&ferences...'));
  set(findobj(fmenu,'label','Pa&ge Setup...'),'pos',4);
  set(findobj(fmenu,'label','Print Set&up...'),'pos',5);
  set(findobj(fmenu,'label','Print Pre&view...'),'pos',6);
  set(findobj(fmenu,'label','&Print...'),'pos',7);
  qmenu = uimenu(fmenu,'label','Quit pplane8',...
         'call','pplane8(''quit'')',...
         'separator','on');

  % Insert menu

  imenu = findobj(gcf,'label','&Insert');
  inschild = get(imenu,'child');
  legitem = findobj(inschild,'label','&Legend');
  colitem = findobj(inschild,'label','&Colorbar');
  delete([legitem,colitem]);

  % View menu

  set(findobj(gcf,'label','&Figure Toolbar'),...
      'call','pplane8(''showbar'')');

  set(0,'showhiddenhandles',hhsetup);

  fs = get(ppxy,'defaultaxesfontsize');
  r = fs/10;
  axw = 437*.8; % Default axes width
  axh = 315*.8; % Default axes height
  axl = 45; % Default axes left
  buttw = 40;     % Default button width
  titleh = 45;    % Default title height
  legw = 90;     % Default legend width
  axb = 35;       % Default axes bottom
  sep = 10;
  fh = axb + axh + titleh;  % Default figure height.
  fw = axl + axw + sep + legw + sep; % Default figure width.
  uni = get(0,'units');
  set(0,'units','pixels');
  ss = get(0,'screensize');
  set(0,'units',uni);
  sw = ss(3);sh = ss(4);
  if r*fh > sh -80;
    r = (sh-80)/fh;
  end
  if r*fw > sw - 50
    r = (sw - 50)/fw;
  end
  fs = 10*r;
  lw = 0.5*r;
  set(ppxy,'defaultaxesfontsize',fs,'defaultaxeslinewidth',lw);
  set(ppxy,'defaulttextfontsize',fs);
  set(ppxy,'defaultlinelinewidth',lw);
  set(ppxy,'defaultuicontrolfontsize',fs*0.9);
  axw = r*axw;
  axh = r*axh;
  axl = r*axl;
  legw = r*legw;
  axb = r*axb;
  sep = r*sep;

  % The legend.
```

```
leg = axes('units','pix',...
       'box','on',...
       'drawmode','fast',...
       'nextplot','add',...
       'xtick',[-1],'ytick',[-1],...
       'xticklabel','','yticklabel','',...
       'xlim',[0,1],'ylim',[0,1],...
       'clipping','on','visible','off');

set(leg,'user',str2mat(xstring,ystring));
axes(leg);
xh = text(0,0,xstring,'units','norm','visible','off',...
      'parent',leg);
yh = text(0,0,ystring,'units','norm','visible','off',...
      'parent',leg);
set(xh,'units','pix')
set(yh,'units','pix')
xext = get(xh,'extent');
yext = get(yh,'extent');
set(xh,'units','norm')
set(yh,'units','norm')
th = max(xext(4),yext(4))+3*r;
axhn = max(axh, 11*th + 4 + 6*r);  % Nudge the axes height if needed.
axw = axw*axhn/axh;
axh = axhn;
frsep = (axh - 11*th - 4 - 6*r)/4;

legh = 2*(th+3*r);
legb = axb + axh - legh;  %frsep*4 + th*9 +4;
legl = axl + axw + sep;


% If the legend text is too big, make the legend and the figure larger.

legpos = [legl+1, legb, legw, legh];
set(leg,'pos',legpos);
xext = get(xh,'extent');
yext = get(yh,'extent');
tw = max(xext(3),yext(3))+0.1;

rrrr = 0.5;
if (tw > rrrr)
  rrr = tw/rrrr;
  legw = legw*rrr;
  tw = rrrr;
end
legpos = [legl+1, legb, legw, legh];
set(leg,'pos',legpos);
fw = axl + axw + sep + legw + sep; % Figure width.

tx = min(1-tw,2/3);
set(xh,'pos',[tx,2/3]);
set(yh,'pos',[tx,1/3]);
```

```
line('xdata',[0.1,tx-0.1],'ydata',[2/3,2/3],'linestyle','-',...
     'color',ud.color.tx,'vis','off');
line('xdata',[0.1,tx-0.1],'ydata',[1/3,1/3],'linestyle','--',...
     'color',ud.color.ty,'vis','off');

% Set up the title.

tax = axes('box','off',...
       'xlim',[0 1],'ylim',[0 1],...
       'units','pix','vis','off',...
       'xtick',[-1],'ytick',[-1],...
       'xticklabel','','yticklabel','');
titb = axb + axh;
titl = axl;
titw = axw;
titeq = text(0.01,0.5,dud.tstr,'vert','middle');
set(titeq,'units','pix');
ext = get(titeq ,'extent');
set(titeq,'units','data');
titleh = ext(4)+6*r;
p1 = get(dud.title.p1,'string');
p2 = get(dud.title.p2,'string');
p3 = get(dud.title.p3,'string');
pp1 = text(0.85,0.5,p1,'vert','middle');
ext = get(pp1,'extent');
pp1l = 1 - ext(3);
set(pp1,'pos',[pp1l,0.5]);
pp2 = text(0.7,0.5,p2,'vert','middle');
ext = get(pp2,'extent');
pp2l = pp1l - ext(3)-0.05;
set(pp2,'pos',[pp2l,0.5]);
pp3 = text(0.7,0.5,p3,'vert','middle');
ext = get(pp3,'extent');
pp3l = pp2l - ext(3)-0.05;
set(pp3,'pos',[pp3l,0.5]);

fh = axb + axh + titleh;  % Figure height.

pos = get(cfig,'pos');
fl = max(0,pos(1)-30);
fb = max(20,pos(2)-30);
fpos = [fl,fb,fw,fh];
set(ppxy,'pos',fpos);

set(tax,'pos',[titl,titb,titw,titleh]);

set(tax,'color',get(ppxy,'color'));


% The t-plot axes.

axpos = [axl,axb,axw,axh];

axy = axes('units','pix','pos',axpos,'box',...
       'on','xgrid','on','ygrid','on','next','add',...
       'drawmode','fast');

output = [ppxy,axy];



% The position of the close button

bbot = axb;
bwid = legw;
bleft = legl;
bpos = [bleft,bbot,bwid,th];

% The position of the print button.

pbbot = bbot + th + frsep;
pbpos = [bleft, pbbot, legw, th];

% The position of the crop button.
cbbot = pbbot + th + frsep;
cbpos = [bleft, cbbot, legw, th];

% The positions of the radio frame and its elements.

frbot = cbbot + th + frsep;
frleft = legl;
frht = 6*th+4;
frpos = [frleft,frbot,legw,frht];

frtitleft = frleft+2;
frtitwid = legw-4;
frtitpos = [frtitleft, frbot+5*th, frtitwid, th];

radleft = frleft + 2;
radwid = legw - 4;
radbot5 = frbot + 2;
radbot4 = radbot5 + th;
radbot3 = radbot4 + th;
radbot2 = radbot3 + th;
radbot1 = radbot2 + th;
pcall = [
    'ppdisp = findobj(''name'',''pplane8 Display'');',...
    'dud = get(ppdisp,''user'');',...
    'eval(dud.printstr)'
    ];

but = uicontrol('style','push',...
        'pos',bpos,...
        'string','Go away',...
        'call','close');

pbut = uicontrol('style','push',...
         'pos',pbpos,...
         'string','Print',...
         'call',pcall);

cbut = uicontrol('style','push',...
         'pos',cbpos,...
         'string','Crop',...
         'call','pplane8(''crop'')');

pframe = uicontrol('style','frame','pos',frpos);

pfrtitle = uicontrol('style','text','pos',frtitpos,...
             'string','Graph','horizon','center');

rval1 = ~(type -1);
rval2 = (~(type -2))*2;
rval3 = (~(type -3))*3;
rval4 = (~(type -4))*4;
rval5 = (~(type -5))*5;

rad(1) = uicontrol('style','radio',...
           'pos',[radleft radbot1 radwid th],...
           'string',xstring,'value',rval1);

rad(2) = uicontrol('style','radio',...
           'pos',[radleft radbot2 radwid th],...
           'string',ystring,'value',rval2,'max',2);

rad(3) = uicontrol('style','radio',...
           'pos',[radleft radbot3 radwid th],...
           'string','Both','value',rval3,'max',3);

rad(4) = uicontrol('style','radio',...
           'pos',[radleft radbot4 radwid th],...
           'string','3 D','value',rval4,'max',4);

rad(5) = uicontrol('style','radio',...
           'pos',[radleft radbot5 radwid th],...
           'string','Composite','value',rval5,'max',5);

for i=1:5
  set(rad(i),'user',[rad(:,[1:(i-1),(i+1):5]),leg,axy]);
end

callrad = [
    'me = gcbo;',...
    'vv = get(me,''value'');'...
    'mm = get(me,''max'');'...
    'if vv ,'...
    ' hand = get(me,''user'');',...
    ' set(hand(1:4),''value'',0);',...
    ' pplane8(''plxy'',me,gcf);'...
    'end, '...
    'set(me,''value'',mm);'...
    'axy = gca;',...
    'aud = get(axy,''user'');',...
```

```
      'aud.rad = me;',...
      'set(axy,''user'',aud);'];


  set(rad,'call',callrad);

  set(findobj(ppxy,'type','axes'),'units','normal');
  set(findobj(ppxy,'type','uicontrol'),'units','normal');
  set(ppxy,'resize','on');

  set(ppxy,'visible','on');

  nstr = get(dud.notice,'string');
  nstr(1:4) = nstr(2:5);
  nstr{5} = 'Ready.';
  set(dud.notice,'string',nstr);
  aud.h = [];
  aud.int = [0 0];
  aud.crop = cbut;
  aud.rad = rad(type);
  aud.leg = leg;
  set(axy,'user',aud);
  set(cbut,'user',axy);

elseif strcmp(action,'plxy')

   radbut = input1;
   fig = input2;
   fud = get(fig,'user');
   figure(fig)
   tbh = findobj(allchild(fig),'flat','type','uitoolbar');
   r3dh = findobj(tbh,'tooltipstr','Rotate 3D');
   ud = get(radbut,'user');
   axy = ud(6);
   axis('auto');
   delete(get(axy,'children'));
   leg = ud(5);
   legch = get(leg,'children');
   type = get(radbut,'max');
   fud.type = type;
   set(fig,'user',fud);
   if type == 3
      set(leg,'visible','on');
      set(legch,'visible','on');
   else
      set(leg,'visible','off');
      set(legch,'visible','off');
   end

   aud = get(axy,'user');
   aud.h = [];
   set(axy,'user',aud);
   strings = get(leg,'user');
   xstring = deblank(strings(1,:));
   ystring = deblank(strings(2,:));
```

```
ud = get(fig,'user');
data = ud.data;
color = ud.color;

t = data(1,:);
x = data(2,:);
y = data(3,:);
tmin = min(t);
tmax = max(t);
xmin = min(x);
xmax = max(x);
ymin = min(y);
ymax = max(y);
et = max((tmax-tmin)/20,1e-4);
ex = max((xmax-xmin)/20,1e-4);
ey = max((ymax-ymin)/20,1e-4);
tmin1 = tmin - et;
tmax1 = tmax + et;
xmin = xmin - ex;
xmax = xmax + ex;
ymin = ymin - ey;
ymax = ymax + ey;

axes(axy);
switch type
 case 1
   plot(t,x,'color',color.tx,'linestyle','-');
   view(2);
   axis([tmin,tmax,xmin,xmax])
   xlabel('t')
   ylabel(xstring)
   set(r3dh,'vis','off')
   set(fig,'WindowButtonDownFcn','pplane8(''plxybdf'')');
 case 2
   plot(t,y,'color',color.ty,'linestyle','-');
   view(2);
   axis([tmin,tmax,ymin,ymax])
   xlabel('t')
   ylabel(ystring)
   set(r3dh,'vis','off')
   rotate3d off
   set(fig,'WindowButtonDownFcn','pplane8(''plxybdf'')');
 case 3
   plot(t,x,'color',color.tx,'linestyle','-');
   plot(t,y,'color',color.ty,'linestyle','--');
   axis([tmin,tmax,min(xmin,ymin),max(xmax,ymax)])
   view(2);
   xlabel('t')
   ylabel([xstring,' and ',ystring])
   set(r3dh,'vis','off')
   rotate3d off
   set(fig,'WindowButtonDownFcn','pplane8(''plxybdf'')');
 case 4
   plot3(x,y,t,'color',color.tx,'linestyle','-');
   view([-30,20]);
   axis([xmin,xmax,ymin,ymax,tmin1,tmax1]);
```

```
      zlabel('t')
      xlabel(xstring);
      ylabel(ystring);
      set(r3dh,'vis','on')
      rotate3d off
      set(fig,'WindowButtonDownFcn','');
    otherwise
      ax = [xmin,xmax,ymin,ymax,tmin1,tmax1];
      view([-30,20]);
      axis(ax);
      plot3(x,y,ax(5)*ones(size(x)),'color',color.ty,'linestyle','-');
      plot3(x,ax(4)*ones(size(y)),t,'color',color.ty,'linestyle','-');
      plot3(ax(2)*ones(size(x)),y,t,'color',color.ty,'linestyle','-');
      plot3(x,y,t,'color',color.tx,'linestyle','-');

      zlabel('t')
      xlabel(xstring);
      ylabel(ystring);
      set(r3dh,'vis','on')
      rotate3d off
      set(fig,'WindowButtonDownFcn','');
   end
   if (type < 4)
      set(aud.crop,'vis','on','enable','off');
   else
      set(aud.crop,'vis','off');
   end
   if type < 4
      set(fig,'windowbuttondownfcn','pplane8(''plxybdf'')');
   elseif type == 5
      set(fig,'windowbuttondownfcn',' ');
   end

elseif strcmp(action,'crop')

   cb = gcbo;
   axy = get(cb,'user');
   aud = get(axy,'user');
   delete(aud.h);
   set(aud.crop,'enable','off');
   aud.h = [];
   set(axy,'user',aud);
   ppxy = gcf;
   ud = get(ppxy,'user');
   data = ud.data;
   color = ud.color;
   t = data(1,:);
   x = data(2,:);
   y = data(3,:);
   int = aud.int;
   k = find((t>=int(1)) & (t<=int(2)));
   t = t(k);
   x = x(k);
   y = y(k);
   type = get(aud.rad,'max');
   hh = pplane8('plotxyfig',type,ppxy);
   ud = get(hh(1),'user');
   ud.data = [t;x;y];
   set(hh(1),'user',ud);
   aud = get(hh(2),'user');
   pplane8('plxy',aud.rad,hh(1))

elseif strcmp(action,'plxybdf')

   dispa = gca;
   dispf = gcf;
   set(dispf,'windowbuttonmotionfcn','pplane8(''plxybmf'')',...
      'windowbuttonupfcn','pplane8(''plxybuf'')',...
      'inter','on');
   aud = get(dispa,'user');
   delete(aud.h);
   aud.h = [];
   set(aud.crop,'enable','off');
   point = get(dispa,'currentpoint');
   aud.start = point(1,1);
   aud.finish = point(1,1);
%  aud.centh = plot(point(1,1),point(1,2),'or','erase','xor');
   aud.h = plot(point(1,1),point(1,2),'--g',...
      'erase','xor','vis','off');
   set(dispa,'user',aud);

elseif strcmp(action,'plxybmf')

   dispa = gca;
   aud = get(dispa,'user');
   point = get(dispa,'currentpoint');
   finish = point(1,1);
   start = aud.start;
   xlim = get(dispa,'xlim');
   ylim = get(dispa,'ylim');

   if abs(finish - start)>0.05*(xlim(2)-xlim(1));
      set(aud.h,'xdata',[start,start,NaN,finish,finish],...
         'ydata',[ylim,NaN,ylim],...
         'vis','on');
   end
   aud.finish = finish;
   set(gca,'user',aud);

elseif strcmp(action,'plxybuf')

   dispa = gca;
   dispf = gcf;
   xlim = get(dispa,'xlim');
   ylim = get(dispa,'ylim');
   aud = get(dispa,'user');
   set(gcf,'windowbuttonmotionfcn','',...
      'windowbuttonupfcn','',...
      'inter','on');
   start = aud.start;
   finish = aud.finish;
   if abs(finish - start)>0.05*(xlim(2)-xlim(1));
      set(aud.h,'erase','normal','vis','on');
      set(aud.crop,'enable','on');
      aud.int = [min(start,finish),max(start,finish)];
   else
      set(aud.h,'vis','off');
   end
   set(gca,'user',aud);


elseif strcmp(action,'print')

   dud = get(gcf,'user');
   nstr = get(dud.notice,'string');
   nstr(1:4) = nstr(2:5);
   nstr{5} = 'Preparing to print the pplane8 Display Window.  Please be
patient.';
   set(dud.notice,'string',nstr);

   nstr(1:4) = nstr(2:5);
   nstr{5} = 'Printing the pplane8 Display Window.';
   set(dud.notice,'string',nstr);
   set(gcf,'pointer','watch');
   eval(dud.printstr);
   nstr(1:4) = nstr(2:5);
   nstr{5} = 'Ready.';
   set(dud.notice,'string',nstr);
   set(gcf,'pointer','arrow');

elseif strcmp(action, 'eqptlist')

   % The labels for equilibrium points.

   disp(' ');
   EqPtType = ['Saddle point.            ';
      'Nodal sink.              ';
      'Nodal source.            ';
      'Spiral sink.             ';
      'Spiral source.           ';
      'Spiral equilibrium point.';
      'Source.                  ';
      'Sink.                    ';
      'Unspecified.             ';];
   dud = get(gcf,'user');
   EqPtList = dud.eqpts;
   if isempty(EqPtList)
      disp('No equilibrium points have been computed.'),disp(' ')
   else
      disp('The following equilibrium points have been calculated:')
      disp(' ')
      L = size(EqPtList,1);
      eqpttext = cell(L,1);
      for k = 1:L
         disp([sprintf('(%6.4f, %6.4f)\t',EqPtList(k,1),EqPtList(k,2)),...
               EqPtType(EqPtList(k,3),:)])
      end
      disp(' ')
   end

elseif strcmp (action,'zoomback')

   disph = gcf;
   dud = get(disph,'user');
   axh = dud.axes;
   Xname = dud.syst.xvar;
   Yname = dud.syst.yvar;
   wmat = dud.wmat;
   WINvect = dud.syst.wind;

   NN = size(wmat,1);

   wch = 0;j=0;
   while wch == 0
      j = j+1;
      if WINvect == wmat(j,:)
         wch = j;
      end
   end
   winstr = cell(1,NN);
   for j = 1:NN
      a = num2str(wmat(j,1));
      b = num2str(wmat(j,2));
      c = num2str(wmat(j,3));
      d = num2str(wmat(j,4));
      winstr{j} = ['  ',a,' < ',Xname,' < ',b,'   &   ',c,' < ', Yname,' <
',d];
   end

   [sel,ok] = listdlg('liststring',winstr,...
      'selectionmode','single',...
      'listsize',[400,200],...
      'initialvalue',wch,...
      'name','pplane8 Zoomback',...
      'promptstring','Select a rectangle:',...
      'OKString','Zoom');

   if (~isempty(sel))
      WINvect = wmat(sel,:);
      dud.syst.wind = WINvect;
      set(gcf,'user',dud);
      ppset = findobj('name','pplane8 Setup');
      sud = get(ppset,'user');
      set(disph,'user',dud);
      set(sud.h.wind(1),'string',num2str(WINvect(1)));
      set(sud.h.wind(2),'string',num2str(WINvect(2)));
      set(sud.h.wind(3),'string',num2str(WINvect(3)));
      set(sud.h.wind(4),'string',num2str(WINvect(4)));
      sud.c.wind = WINvect;
      sud.o.wind = WINvect;
      set(ppset,'user',sud);
      aud = get(axh,'user');
      aud.DY = [WINvect(2) - WINvect(1);WINvect(4) - WINvect(3)];
      dwind = [WINvect(1); WINvect(3); -WINvect(2); -WINvect(4)];
      aud.cwind = dwind - dud.settings.magn*[aud.DY;aud.DY];
      set(axh,'user',aud);
      pplane8('dirfield',disph);
   end

elseif strcmp(action,'figdefault')

  fig = input1;
  set(fig,'CloseRequestFcn','pplane8(''closefcn'')');
  ppset = findobj('name','pplane8 Setup');
  sud = get(ppset,'user');
  ud = get(fig,'user');
  ud.ssize = sud.ssize;
  fs = sud.fontsize;
  ud.fontsize = fs;
  style = sud.style;
  set(fig,'defaulttextfontsize',fs);
  set(fig,'defaultaxesfontsize',fs);
  set(fig,'defaultuicontrolfontsize',9*fs/10)
  lw = 0.5*fs/10;
  set(fig,'defaultaxeslinewidth',lw)
  set(fig,'defaultlinelinewidth',lw)
  set(fig,'defaultaxesfontname','helvetica')
  set(fig,'defaultaxesfontweight','normal')

  switch style
   case 'black'
    % if isunix | isvms, gamma = 0.5; else gamma = 0.0; end
    whitebg(fig,[0,0,0])
    if isunix | isvms
      fc = [.35 .35 .35];
    else
      fc = [.2 .2 .2];
    end
    set(fig,'color',fc);
    set(fig,'defaultaxescolor',[0 0 0])
    % whitebg(fig,brighten([.2 .2 .2],gamma))
    set(fig,'defaultaxescolor',[0 0 0])
    set(fig,'defaultaxescolororder', ...
        1-[0 0 1;0 1 0;1 0 0;0 1 1;1 0 1;1 1 0;.25 .25 .25]) % ymcrgbw
    % set(fig,'colormap',brighten(jet(64),gamma))
    set(fig,'colormap',jet(64))
    set(fig,'defaultsurfaceedgecolor',[0 0 0]);
   case 'white'
    whitebg(fig,[1 1 1])
    set(fig,'color',[.8 .8 .8])
    set(fig,'defaultaxescolor',[1 1 1])
    set(fig,'defaultaxescolororder', ...
        [0 0 1;0 .5 0;1 0 0;0 .75 .75;.75 0 .75;.75 .75 0;.25 .25 .25]) %
bgrymck
    set(fig,'colormap',jet(64))
    set(fig,'defaultsurfaceedgecolor',[0 0 0])

   case 'display'
```

```
    whitebg(fig,[1 1 1])
    set(fig,'defaultaxescolor',[1 1 1])
    set(fig,'defaultaxescolororder', ...
        [0 0 1;0 .5 0;1 0 0;0 .75 .75;.75 0 .75;.75 .75 0;.25 .25 .25]) %
bgrymck
    set(fig,'colormap',jet(64))
    set(fig,'defaultsurfaceedgecolor',[0 0 0])
    set(fig,'color',[1 1 1]*240/255);
    set(fig,'defaultuicontrolbackgroundcolor',[1 1 1]*220/255);
    set(fig,'defaultaxesfontweight','bold')
    set(fig,'defaulttextfontweight','bold')
    set(fig,'defaultaxeslinewidth',1)
    set(fig,'defaultlinelinewidth',1)

   case 'bw'
    whitebg(fig,[0 0 0])
    set(fig,'color',[0 0 0])
    set(fig,'defaultaxescolor',[0 0 0])
    set(fig,'defaultaxescolororder', ...
        [1 1 1])
    set(fig,'colormap',[1 1 1;0 0 0])
    set(fig,'defaultsurfaceedgecolor',[0 0 0])

  end
  set(fig,'user',ud);

elseif strcmp(action,'zoominsq')

   % 'zoominsq' is the callback for the Zoom in square menu item.

   set(gcf,'WindowButtonDownFcn','pplane8(''zoomsqd'')',...
      'WindowButtonUpFcn','1;','inter','on');
   set(gca,'inter','on');
   dud = get(gcf,'user');
   nstr = get(dud.notice,'string');
   nstr(1:4) = nstr(2:5);
   nstr{5} = ['Pick a center and ',...
         'drag the mouse, or just click on a center.'];
   set(dud.notice,'string',nstr);

elseif strcmp(action,'zoomsqd')

   ppdispa = gca;
   aud = get(ppdispa,'user');
   point = get(ppdispa,'currentpoint');
   aud.center = point(1,[1,2]);
   aud.centh = plot(point(1,1),point(1,2),'or','erase','xor');
   aud.box = plot(point(1,1),point(1,2),'--r','erase','xor');
   set(ppdispa,'user',aud);
   set(gcf,'windowbuttonmotionfcn','pplane8(''zoomsqm'')',...
      'windowbuttonupfcn','pplane8(''zoomsqu'')');


elseif strcmp(action,'zoomsqm')
```

```
   ppdispa = gca;
   aud = get(ppdispa,'user');
   point = get(ppdispa,'currentpoint');
   point = point(1,[1,2])';
   un = get(ppdispa,'units');
   set(ppdispa,'units','pix');
   w = get(ppdispa,'pos'); w = w([3,4]);w = w(:);
   set(ppdispa,'units',un);
   cent = aud.center(:);
   lam = max(abs(point - cent)./w);
   v = lam*w;
   data = cent(:,[1 1 1 1 1]) + [v,[-v(1);v(2)], -v, [v(1);-v(2)],v];
   set(aud.box,'xdata',data(1,:),'ydata',data(2,:));

elseif strcmp(action,'zoomsqu')

   disph = gcf;
   dud = get(disph,'user');
   axh = dud.axes;
   aud = get(axh,'user');
   hand = [aud.centh;aud.box];
   set(hand,'erase','normal');
   delete(hand);
   DY = aud.DY(:);
   un = get(axh,'units');
   set(axh,'units','pix');
   w = dud.syst.wind;
   point = get(axh,'currentpoint');
   point = point(1,[1,2]);point = point(:);
   cent = aud.center(:);
   ww = get(axh,'pos'); ww = ww([3,4]); ww = ww(:);
   lamb = max(abs(point-cent)./ww);
   v = lamb*ww;
   if ~all(v > 0.01*DY)
      points = [w([1 1 2 2]);w([3 4 3 4])];
      for j=1:4
         lambs(j) = max(abs(points(:,j)-cent)./ww);
      end
      lamb = min(lambs);
      v = lamb*ww/4;
   end
   p1 = cent + v; p2 = cent - v;
   a = [p2';p1'];
   a = [min(a);max(a)];
   DY = (a(2,:) - a(1,:))';
   WINvect = a(:)';
   dud.syst.wind = WINvect;
   aud.DY = DY;
   dwind = [WINvect(1); WINvect(3); -WINvect(2); -WINvect(4)];
   aud.cwind = dwind - dud.settings.magn*[aud.DY;aud.DY];
   set(axh,'units',un,'user',aud);
   set(disph,'user',dud);
   set(disph,'WindowButtonDownFcn','pplane8(''down'')',...
      'WindowButtonMotionFcn','pplane8(''cdisp'')',...
      'WindowButtonUpFcn','');
   pplane8('dirfield',disph);
   ppset = findobj('name','pplane8 Setup');
   if isempty(ppset)
      pplane8('confused');
   else
      sud = get(ppset,'user');
      sud.c.wind = WINvect;
      sud.o.wind = WINvect;
      set(sud.h.wind(1),'string',num2str(WINvect(1)));
      set(sud.h.wind(2),'string',num2str(WINvect(2)));
      set(sud.h.wind(3),'string',num2str(WINvect(3)));
      set(sud.h.wind(4),'string',num2str(WINvect(4)));
      set(ppset,'user',sud);
   end

elseif strcmp(action,'level')

  ppdisp = gcf;
  dud = get(ppdisp,'user');
  lfcn = dud.level;
  Xname = dud.syst.xvar;
  Yname = dud.syst.yvar;
  Xname(find(abs(Xname)==92))=[];  % Remove \s if any.
  Yname(find(abs(Yname)==92))=[];
  pplevel = findobj('name','pplane8 Level sets');
  if ~isempty(pplevel)
    delete(pplevel)
  end
  pplevel = figure('name','pplane8 Level sets',...
         'vis','off',...
         'numb','off','tag','pplane8');

  pplane8('figdefault',pplevel);
  set(pplevel,'menubar','none');

  lev.fr1 = uicontrol('style','frame');
  lev.fr2 = uicontrol('style','frame');

  inst1str = ['Enter the function in terms of the variables ',...
          Xname, ' and ', Yname,':'];

  lev.inst1 = uicontrol('style','text','horiz','left',...
            'string',inst1str);

  lev.lfcn = uicontrol('style','edit','horiz','center',...
               'string',lfcn,'call','',...
               'background','w');

  lev.ch(3) = uicontrol('style','radio','horiz','center',...
            'min',0,'max',3,...
            'value',0,...
            'vis','on',...
            'string','Let pplane8 decide.');

  lev.ch(2) = uicontrol('style','radio','horiz','center',...
            'min',0,'max',2,...
            'value',0,...
            'string','Select a point in the Display Window.');

  lev.inst2 = uicontrol('style','text','horiz','left',...
            'string',['Choose one of the following ways to',...
            ' choose level value(s):']);

  lev.ch(1)  = uicontrol('style','radio','horiz','center',...
             'min',0,'max',1,...
             'value',0,...
             'string','Enter a vector of level values.');

  lev.rhs = uicontrol('style','edit','horiz','center',...
              'string',' ','call','');

  lev.proc = uicontrol('style','push',...
              'string','Proceed',...
              'call','pplane8(''levcomp'')');

  lev.close = uicontrol('style','push',...
            'string','Close',...
            'call','close');

  for i=1:3
    set(lev.ch(i),'user',lev.ch(:,[1:(i-1),(i+1):3]));
  end

  callrad = [
      'me = get(gcf,''currentobject'');',...
      'kk = get(me,''max'');',...
      'col = get(me,''backg'');',...
      'set(get(me,''user''),''value'',0),',...
      'set(me,''value'',kk);',...
      'ud = get(gcf,''user'');',...
      'if kk == 1,',...
      '   set(ud.rhs,''enable'',''on'',''backg'',''w'');',...
      'else,',...
      '   set(ud.rhs,''enable'',''off'',''backg'',col);',...
      'end,'];

  set(lev.ch,'call',callrad);

  left = 2; varl = 300; buttw = 60;
  nudge = 3;
  tab = 15;
  lines1 = 5;
  lines2 = 2;
  xex = get(lev.inst1,'extent');
  ht = xex(4)+nudge;
  frw = varl + 2*tab+ 2*nudge;
  fr1bot = 2*left + ht;
  fr1ht = lines1*(nudge + ht) + nudge;
  fr2bot = fr1bot + fr1ht;
  fr2ht = lines2*(nudge + ht) + nudge;
  vbot = fr1bot + nudge;
```

```
  ch1bot = vbot + nudge + ht;
  ch2bot = ch1bot + nudge + ht;
  ch3bot = ch2bot + nudge + ht;
  inst2bot = ch3bot + nudge + ht;
  fbot =  fr2bot + nudge;
  inst1bot = fbot + nudge + ht;
  fleft = left + nudge + tab;
  instleft = left + nudge;
  chleft = instleft + tab;
  vleft = chleft + tab;
  vw = (frw - 2*vleft);
  fr1wind = [left,fr1bot,frw,fr1ht];
  fr2wind = [left,fr2bot,frw,fr2ht];
  inst1wind = [instleft,inst1bot,varl,ht];
  inst2wind = [instleft,inst2bot,varl,ht];
  fwind = [fleft,fbot,varl,ht];
  ch1wind = [chleft,ch1bot,varl,ht];
  ch2wind = [chleft,ch2bot,varl,ht];
  ch3wind = [chleft,ch3bot,varl,ht];
  vwind = [vleft,vbot,vw,ht];
  figw = 2*left + frw;
  fight = 3*left + ht + fr1ht + fr2ht;
  figwind = [40, 300, figw, fight];
  buttw = frw/2;
  sep = (figw - 2*buttw)/3;
  closel = sep;
  procl = 2*sep+buttw;
  clwind = [closel,left,buttw,ht];
  procwind = [procl,left,buttw,ht];
  set(pplevel,'pos',figwind);
  set(lev.fr1,'pos',fr1wind);
  set(lev.fr2,'pos',fr2wind);
  set(lev.inst1,'pos',inst1wind);
  set(lev.inst2,'pos',inst2wind);
  set(lev.ch(1),'pos',ch1wind);
  set(lev.ch(2),'pos',ch2wind);
  set(lev.ch(3),'pos',ch3wind);
  set(lev.rhs,'pos',vwind);
  set(lev.lfcn,'pos',fwind);
  set(lev.proc,'pos',procwind);
  set(lev.close,'pos',clwind);
  set(lev.ch(3),'value',3);
  set(lev.rhs,'enable','off');

  child = get(pplevel,'children');
  set(pplevel,'vis','on','user',lev);
  set(child,'vis','on');

elseif strcmp(action,'levcomp')

  pplevel = gcf;
  ud = get(pplevel,'user');
  ppdisp = findobj('name','pplane8 Display');
  dud = get(ppdisp,'user');
  ppset = findobj('name','pplane8 Setup');
  sud = get(ppset,'user');
```

```
 ch = ud.ch;
val = zeros(1,3);
for kk = 1:3
  val(kk) = get(ch(kk),'value');
end
KK = max(val);
lfcn = get(ud.lfcn,'string');
l=length(lfcn);
for ( k = fliplr(findstr('.',lfcn)))
  if (find('*/^' == lfcn(k+1)))
    lfcn = [lfcn(1:k-1), lfcn(k+1:l)];
  end
  l=l-1;
end
pnameh = sud.h.pname;
pvalh = sud.h.pval;
pflag = zeros(1,4);
perr = [];
lfcn(find(abs(lfcn)==32))=[];
for kk = 1:4;
  pn = get(pnameh(kk),'string');
  pv = get(pvalh(kk),'string');
  if ~isempty(pn)
    pn(find(abs(pn)==92))=[];
    if isempty(pv)
  perr = pvalh(kk);
    else
  pv(find(abs(pv)==32))=[];
  lfcn = pplane8('paraeval',pn,pv,lfcn);
    end
  end
end
l = length(lfcn);
for (k=fliplr(find((lfcn=='^')|(lfcn=='*')|(lfcn=='/'))))
  lfcn = [lfcn(1:k-1) '.' lfcn(k:l)];
  l = l+1;
end
WINvect = dud.syst.wind;
XxXxXx = WINvect(1) + rand*(WINvect(2)-WINvect(1));
YyYyYy = WINvect(3) + rand*(WINvect(4)-WINvect(3));
Xname = dud.syst.xvar;
Yname = dud.syst.yvar;
Xname(find(abs(Xname)==92))=[];  % Remove \s if any.
Yname(find(abs(Yname)==92))=[];
err = 0;res = 1;
eval([Xname,'=XxXxXx;'],'err = 1;');
eval([Yname,'=YyYyYy;'],'err = 1;');
eval(['res = ',lfcn, ';'],'err = 1;');
if err | isempty(res)
  errmsg = 'The function does not evaluate correctly.';
  fprintf('\a')
  errordlg(errmsg,'PPLANE error','on');
  return;
end

Xmin = WINvect(1);
Xmax = WINvect(2);
```

```
  Ymin = WINvect(3);
  Ymax = WINvect(4);
  N = 50; k = 4;
  deltax=(Xmax - Xmin)/(N-1);
  deltay=(Ymax - Ymin)/(N-1);
  XXXg=Xmin + deltax*[-k:N+k];
  YYYg=Ymin + deltay*[-k:N+k];

  [Xx,Yy]=meshgrid(XXXg,YYYg);
  Xxx=Xx(:);Yyy=Yy(:);
  Ww = zeros(size(Xxx));
  eval([Xname,'=Xxx;'],'err = 1;');
  eval([Yname,'=Yyy;'],'err = 1;');
  eval(['Ww = ',lfcn, ';'])

  KKK = 3; %# of significant figures.

  switch KK
   case 1  % vector input
    rhs = get(ud.rhs,'string');
    rhs = str2num(rhs);

   case 2  % mouse input
    figure(ppdisp);
%      [XX,YY] = ppginput(1);
    [XX,YY] = ginput(1);
    figure(pplevel);
    eval([Xname,'=XX;'],'err = 1;');
    eval([Yname,'=YY;'],'err = 1;');
    eval(['rhs = ',lfcn, ';'],'err = 1');
    LL = ceil(log10(abs(rhs)));
    rhs = round(10^(KKK-LL)*rhs);
    rhs = 10^(LL-KKK)*rhs;

   case 3  % pplane8 input
    Www = Ww;
    kkk = find(isnan(Www));
    Www(kkk) = [];
    kkk = find(imag(Www));
    Www(kkk) = [];
    MM = max(Www);
    mm = min(Www);
    LL = ceil(log10(MM-mm));
    NN = 7;  % Number of curves
    rhs = mm+(1:NN).^2*(MM-mm)/NN^2;
    rhs = round(10^(KKK-LL)*rhs);
    rhs = 10^(LL-KKK)*rhs;

  end

  Ww = reshape(Ww,N+2*k+1,N+2*k+1);
  lrhs = length(rhs);
  if lrhs == 0
    return
  elseif lrhs == 1
```

```
    rhs = [rhs,rhs];
  end

  figure(ppdisp);
  [Cm,hcont] = contour(Xx,Yy,Ww,rhs,'--');
  hlabel = clabel(Cm,hcont);
%  set(hlabel,'fontsize',dud.fontsize,...
%        'color',dud.color.level,...
%        'rotation',0);
  set(hlabel,'fontsize',dud.fontsize,...
         'color',[1,0,0],...
         'rotation',0);
  set(hcont,'visible','on',...
        'color',dud.color.level,...
        'linestyle',':');
  dud.contours = [dud.contours ;hcont;hlabel];
  set(ppdisp,'user',dud);

elseif strcmp(action,'restart')

   ppset = findobj('name','pplane8 Setup');
   oldfiles = dir('pptp*.m');
   for k = 1:length(oldfiles)
      fn = oldfiles(k).name;
      fid = fopen(fn,'r');
      ll = fgetl(fid);
      ll = fgetl(fid);
      ll = fgetl(fid);
      fclose(fid);
      if strcmp(ll,'%% Created by pplane8')
         delete(fn)
      end
   end
   h = findobj('tag','pplane8');
   delete(setdiff(h,ppset));
   sud = get(ppset,'user');
   sud.flag = 0;
   set(ppset,'user',sud);
   figure(ppset)

elseif strcmp(action,'quit')

  ppset = findobj('name','pplane8 Setup');
  sud = get(ppset,'user');
  if sud.remtd
    rmpath(tempdir);
  end
  oldfiles = dir([tempdir,'pptp*.m']);
  for k = 1:length(oldfiles)
    fn = [tempdir,oldfiles(k).name];
    fid = fopen(fn,'r');
    ll = fgetl(fid);
    ll = fgetl(fid);
    ll = fgetl(fid);
    fclose(fid);
    if strcmp(ll,'%% Created by pplane8')
```

```
      delete(fn)
    end
  end
  h = findobj('tag','pplane8');
  delete(h);

elseif strcmp(action,'closefcn')

   fig = gcf;
   name = get(fig,'name');
   if strcmp(name,'pplane8 Setup') | strcmp(name,'pplane8 Display')
      quest = ['Closing this window will cause all pplane8 ',...
            'windows to close, and pplane8 will stop.  ',...
            'Do you want to quit pplane8?'];
      butt = questdlg(quest,'Quit pplane8?','Quit','Cancel','Quit');
      if strcmp(butt,'Quit')
         pplane8('quit');
      end
   elseif strcmp(name,'pplane8 Linearization')
      dud = get(fig,'user');
      fcn = dud.function;
      if (exist(fcn)==2) delete([fcn,'.m']);end
      delete(findobj('label',name));
      delete(fig);
   else
      delete(findobj('label',name));
      delete(fig);
   end

elseif strcmp(action,'confused')

   tstring = 'pplane8 is totally confused';
   qstring = {['You will have to restart pplane8 from '...
            'the beginning in order to ',...
            'do anything new.  However, it might be possible '...
            'to save the current system ',...
            'or the gallery to make your restart easier, '...
            'or it may be possible to ',...
            'print out a figure, if the appropriate '...
            'figures are visible.  In such a case ',...
            'it would be best to do nothing now.'];
      'What do you want to do?'};
   bstr1 = 'Quit and restart pplane8.';
   bstr2 = 'Just quit pplane8.';
   bstr3 = 'Do nothing.';
   answer = questdlg(qstring,tstring,bstr1,bstr2,bstr3,bstr1);
   if strcmp(answer,bstr1)
      delete(findobj('tag','pplane8'));
      pplane8;return
   elseif strcmp(answer,bstr2)
      delete(findobj('tag','pplane8'));
      return
   else
      return
   end
```

```
elseif strcmp(action,'export')

  % export is the callback for the Export solution data item in the
  % Options menu.

  disph = gcf;
  dud = get(disph,'user');
  arr = dud.arr;
  lv = get(arr.lines,'vis');
  av = get(arr.arrows,'vis');
  if ~isempty(arr.hx)
    nv = get(arr.hx(1),'vis');
  elseif ~isempty(arr.hy)
    nv = get(arr.hx(1),'vis');
  else
    nv = zeros(1,0);
  end
  if ~isempty(arr.barrows)
    bv = get(arr.barrows,'vis');
  else
    bv = zeros(1,0);
  end
  handles = [arr.lines;arr.arrows;arr.hx;arr.hy;arr.barrows];

  set(handles,'vis','off');
  oldcall = get(disph,'WindowButtonDownFcn');
  set(disph,'WindowButtonDownFcn','');
  trjh = dud.solhand;
  notice = dud.notice;
  switch length(trjh)
   case 0
    if notice
      nstr = get(notice,'string');
      nstr(1:3) = nstr(3:5);
      nstr{4} = 'There are no solutions.';
      nstr{5} = 'Ready.';
      set(notice,'string',nstr);
    end
    th = [];

   case 1
    th = trjh;
   otherwise
    if notice
      nstr = get(notice,'string');
      nstr(1:4) = nstr(2:5);
      nstr{5} = 'Select a solution with the mouse.';
      set(notice,'string',nstr);
    end
%    ppginput(1);
    ginput(1);
    th = get(disph,'currentobject');
  end
  if isempty(th)
    if notice
      nstr = get(notice,'string');
      nstr(1:3) = nstr(3:5);
      nstr{4} = 'The item selected is not a solution.';
      nstr{5} = 'Ready.';
      set(notice,'string',nstr);
    end
  else
    vars = evalin('base','who');
    no = 1;
    kk = 0;
    while no
      kk = kk + 1;
      vstr = ['ppdata',num2str(kk)];
      if ~any(strcmp(vars,vstr))
    no = 0;
      end
    end
    yname = dud.syst.yvar;
    if abs(yname(1)) == 92
      yname = yname(2:length(yname));
    end
    xname = dud.syst.xvar;
    if abs(xname(1)) == 92
      xname = xname(2:length(xname));
    end
    tname = 't';
    tval = get(th,'zdata');
    tval = tval(:);
    xval = get(th,'xdata');
    xval = xval(:);
    yval = get(th,'ydata');
    yval = yval(:);
    ivstr = struct(tname,tval,xname,xval,yname,yval);
    assignin('base',vstr,ivstr);
    if notice
      nstr = get(notice,'string');
      nstr(1:3) = nstr(3:5);
      nstr{4} = ['The data has been exported as the structure ',...
         vstr,' with fields ', tname, ', ', xname, ', and ', yname,'.'];
      nstr{5} = 'Ready.';
      set(notice,'string',nstr);
    end


  end



  set(arr.lines,'vis',lv);
  set(arr.arrows,'vis',av);
  set([arr.hx;arr.hy],'vis',nv);
  set(arr.barrows,'vis',bv);
  set(disph,'user',dud);
  set(disph,'WindowButtonDownFcn','pplane8(''down'')');
```

```
elseif strcmp(action,'cdisp')

  [ppcbo,ppdisp] = gcbo;
  dud = get(ppdisp,'user');
  cp = get(ppdisp,'currentpoint');
  fpos = get(ppdisp,'pos');
  ppax = dud.axes;
  xd = get(ppax,'xlim');
  yd = get(ppax,'ylim');
  apos = get(ppax,'pos');
  xp = xd(1) + (cp(1) - apos(1)*fpos(3))*(xd(2)-xd(1))/(apos(3)*fpos(3));
  yp = yd(1) + (cp(2) - apos(2)*fpos(4))*(yd(2)-yd(1))/(apos(4)*fpos(4));
  str = ['(',num2str(xp,3),', ',num2str(yp,3),')'];
  set(dud.ccwind,'string',str);

elseif strcmp(action,'periodic')

  % Find a periodic orbit.

  ppdisp = findobj('name','pplane8 Display');
  dud = get(ppdisp,'user');
  dfcn = dud.function;
  direction = dud.dir;
  notice = dud.notice;
  settings = dud.settings;
  refine = settings.refine;
  tol = settings.tol;
  AA = -1e6;
  BB = 1e6;
  switch direction
   case 0
    intplus = [0, BB];
    intminus = [0, AA];
   case -1
    intplus = [0, 0];
    intminus = [0, AA];
   case 1
    intplus = [0, BB];
    intminus = [0, 0];
  end
  ppdispa = dud.axes;
  aud = get(ppdispa,'user');
  DY = aud.DY;
  aud.plot = 0;
  atol = tol*DY*1e-4';
  set(ppdispa,'user',aud);
  nstr = get(dud.notice,'string');
  nstr(1:4) = nstr(2:5);
  nstr{5} = 'Choose a starting point with the mouse.';
  set(dud.notice,'string',nstr);
%  z00 = ppginput(1);
  z00 = ginput(1);
  z0 = z00;
  ptstr = [' (',num2str(z0(1),2), ', ', num2str(z0(2),2), ')'];
  solver = dud.settings.solver;
  opt = odeset('OutputFcn',@ppout,'Refine',refine,...
           'RelTol',tol,'Abstol',atol);
  switch solver
   case 'Dormand Prince'
    solh = @ppdp45;
    opt = ppdisp;
   case 'Runge-Kutta 4'
    solh = @pprk4;
    opt = ppdisp;
   case 'ode45'
    solh = @ode45;
   case 'ode23'
    solh = @ode23;
   case 'ode113'
    solh = @ode113;
   case 'ode15s'
    solh = @ode15s;
   case 'ode23s'
    solh = @ode23s;
   case 'ode23t'
    solh = @ode23t;
   case 'ode23tb'
    solh = @ode23tb;
  end
  exist(dfcn);
  dfh = str2func(dfcn);
  dud.noticeflag = 0;   % Notices only from here.
  set(ppdisp,'user',dud);
  if intplus(2)>intplus(1)
    if notice
      nstr = get(notice,'string');
      nstr(1:4) = nstr(2:5);
      nstr{5} = 'The forward orbit ';
      set(notice,'string',nstr);
    end
    drawnow
    [tp,xp] = feval(solh,dfh,intplus,z0,opt);
    aud = get(ppdispa,'user');
    set(aud.line,'erase','normal')
    delete(aud.line);
    aud.line = [];
    z0 = aud.y;
    stop = aud.stop;
    nstr = get(dud.notice,'string');
    switch stop
     case 1
      nstr{5} = [nstr{5}, ' left the computation window.'];
     case 2
      zz = aud.zz;
      ystr = ['(',num2str(zz(1),2), ', ', num2str(zz(2),2),').'];
      nstr{5} = [nstr{5}, ' --> a possible eq. pt. near ',ystr];
     case 3
      nstr{5} = [nstr{5}, ' --> a closed orbit'];
     case 4
      nstr{5} = [nstr{5}, ' was stopped by the user.'];
     case 5
      ystr = ['(',num2str(y(1),2), ', ', num2str(y(2),2),').'];
      nstr(1:3) = nstr(2:4);
      nstr{4} = [nstr{5},' experienced a failure at ',ystr];
      nstr{5} = 'Problem is singular or tolerances are too large.';
    end
    set(dud.notice,'string',nstr);
    drawnow

    if stop==3  % periodic orbit found.
      % [tp,xp] = feval(solh,dfh,intplus,z0,opt);
      rr = sqrt(((xp(:,1) - z0(1))/DY(1)).^2 + ((xp(:,2) - z0(2))/DY(2)).^2);
      kk = find(rr < 0.01);
      LL = length(kk);
      kmax = max(kk);
      kkk = max(kk(find(kk<kmax-30)));
      T = tp(kmax) - tp(kkk);
      aud.gstop = 0;
      set(ppdispa,'user',aud);
      kkk = 0;
      NN = 1;
      dz0 = feval(dfh,0,z0);
      while abs(NN)>0.0001*abs(T) & kkk < 10
    [tt,yy] = feval(solh,dfh,[0,T],z0,opt);
    yy = yy(length(tt),:);
    dyy = feval(dfh,0,yy');
    NN = ((yy-z0')*dz0)/(dyy'*dz0);
    T = T - NN;
    kkk = kkk +1;
      end
      nstr{5} = [nstr{5}, ' of period ', num2str(abs(T),3), '.'];
      set(dud.notice,'string',nstr);
      drawnow
      [tp,xp] = feval(solh,dfh,[0 3*T],z0,opt);
      aud.gstop = 1;
      set(ppdispa,'user',aud);
      hnew = plot(xp(:,1),xp(:,2),'color','b');
      set(hnew,'zdata',tp);
      solhand = [dud.solhand;hnew];
      dud.solhand = solhand;
    end
    drawnow
  end
  if intminus(2)<intminus(1)
    z0 = z00;
    if notice
      nstr = get(notice,'string');
      nstr(1:4) = nstr(2:5);
      nstr{5} = 'The backward orbit ';
      set(notice,'string',nstr);
    end
    drawnow
    [tp,xp] = feval(solh,dfh,intminus,z0,opt);
    aud = get(ppdispa,'user');
    set(aud.line,'erase','normal')
    delete(aud.line);
    aud.line = [];
    z0 = aud.y;
    stop = aud.stop;
    nstr = get(dud.notice,'string');
    switch stop
     case 1
      nstr{5} = [nstr{5}, ' left the computation window.'];
     case 2
      zz = aud.zz;
      ystr = ['(',num2str(zz(1),2), ', ', num2str(zz(2),2),').'];
      nstr{5} = [nstr{5}, ' --> a possible eq. pt. near ',ystr];
     case 3
      nstr{5} = [nstr{5}, ' --> a nearly closed orbit.'];
     case 4
      nstr{5} = [nstr{5}, ' was stopped by the user.'];
     case 5
      ystr = ['(',num2str(y(1),2), ', ', num2str(y(2),2),').'];
      nstr(1:3) = nstr(2:4);
      nstr{4} = [nstr{5},' experienced a failure at ',ystr];
      nstr{5} = 'Problem is singular or tolerances are too large.';
    end
    set(dud.notice,'string',nstr);
    drawnow
    if stop==3  % periodic orbit found.
      z0 = aud.y;
      rr = sqrt(((xp(:,1) - z0(1))/DY(1)).^2 + ((xp(:,2) - z0(2))/DY(2)).^2);
      kk = find(rr < 0.01);
      LL = length(kk);
      kmax = max(kk);
      kkk = max(kk(find(kk<kmax-30)));
      T = tp(kmax) - tp(kkk);
      aud.gstop = 0;
      set(ppdispa,'user',aud);
      kkk = 0;
      NN = 1;
      dz0 = feval(dfh,0,z0);
      while abs(NN)>0.0001*abs(T) & kkk < 10
    [tt,yy] = feval(solh,dfh,[0,T],z0,opt);
    yy = yy(length(tt),:);
    dyy = feval(dfh,0,yy');
    NN = ((yy-z0')*dz0)/(dyy'*dz0);
    T = T - NN;
    kkk = kkk +1;
      end
      nstr{5} = [nstr{5}, ' of period ', num2str(abs(T),3), '.'];
      set(dud.notice,'string',nstr);
      drawnow
      [tp,xp] = feval(solh,dfh,[0 3*T],z0,opt);
      aud.gstop = 1;
      set(ppdispa,'user',aud);
      hnew = plot(xp(:,1),xp(:,2),'color','b');
      set(hnew,'zdata',tp);
      solhand = [dud.solhand;hnew];
      dud.solhand = solhand;
    end
    drawnow
  end
  dud.noticeflag = 1;
  dud.dir = 0;
  set(ppdisp,'user',dud);
  aud.plot = 1;
```

```
  set(ppdispa,'user',aud);

  if notice
    nstr = get(notice,'string');
    nstr(1:4) = nstr(2:5);
    nstr{5} = 'Ready';
    set(notice,'string',nstr);
  end
  drawnow


end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [tout,yout] = ppdp45(dfcn,tspan,y0,disph)

% PPDP45 is an implementation of the explicit Runge-Kutta (4,5) which
%        is described in Chapters 5 & 6 of John Dormand's book,
%        Numerical Methods for Differential Equations.
%
%        This is the same algorithm used in ODE45, part of the new MATLAB
%        ODE Suite.  Details are to be found in The MATLAB ODE Suite,
%        L. F. Shampine and M. W. Reichelt, SIAM Journal on Scientific
%        Computing, 18-1, 1997.


% Input the user data.

  dud = get(disph,'user');
  dispha = dud.axes;
  ud = get(dispha,'user');
  refine = dud.settings.refine;
  tol = dud.settings.tol;
  gstop = ud.gstop;
  plotf = ud.plot;
  DY = ud.DY;
  col = dud.color.temp;
  speed = dud.settings.speed;
  slow = (speed < 100);

  % Initialize the stopping criteria.
  if gstop

    % Initialize detection of closed orbits & limit cycles.
    % Choose a random direction & initialize the search for orbit maxima in
    % that direction.

    theta = rand*2*pi;
    R = [cos(theta), sin(theta); -sin(theta),cos(theta)];
    qq = R*(y0(:));
    rr = [qq,qq];
    z = DY(1) + sqrt(-1)*DY(2);
    w=exp(i*theta);
    a1 = w*z;
    a2 = w*(z');
    a=max(abs(real([a1,a2])));
    b=max(abs(imag([a1,a2])));
    perpeps = a*0.00001; % a/2000;  % We want to be real
            % close in this direction.
    paraeps = b/100;    % We do not have to be
            % so careful in this direction.
    tk = 0;
    turn = zeros(2,10);

    % The test for an equilibrium point.

    sinkeps = 0.005/refine;

    %  The compute window.

    cwind = ud.cwind;
  end
  % The stop button.

  stop = 0;
  ud.stop = 0;

  % Set the the line handle.

  ph = plot([y0(1),y0(1)],[y0(2),y0(2)],...
        'color',col,...
        'erase','none',...
        'parent',dispha);
  ud.line = ph;
  set(dispha,'UserData',ud);

  % Initialize the loop.

  t0 = tspan(1);
  tfinal = tspan(2);
  tdir = sign(tfinal - t0);
  t = t0;
  y = y0(:);

  % By default, hmax is 1/10 of the interval.
  hmax = min(abs(0.1*(tfinal-t)),1);

  rDY = DY(:,ones(1,refine));
  steps = 0;
  block = 120;
  tout = zeros(block,1);
  yout = zeros(block,2);

  N = 1;
  tout(N) = t;
  yout(N,:) = y.';
```

```
% Initialize method parameters.
pow = 1/5;
%  C = [1/5; 3/10; 4/5; 8/9; 1; 1];
%  Not needed because the sytem is autonomous.
A = [
    1/5         3/40    44/45   19372/6561      9017/3168
    0           9/40    -56/15  -25360/2187     -355/33
    0           0       32/9    64448/6561      46732/5247
    0           0       0       -212/729        49/176
    0           0       0       0               -5103/18656
    0           0       0       0               0
    0           0       0       0               0
    ];
bhat = [35/384 0 500/1113 125/192 -2187/6784 11/84 0]';
% E = bhat - b.
E = [71/57600; 0; -71/16695; 71/1920; -17253/339200; 22/525; -1/40];
if refine > 1
  sigma = (1:refine-1) / refine;
  S = cumprod(sigma(ones(4,1),:));
  bstar = [
  1       -183/64     37/12       -145/128
  0       0       0       0
  0       1500/371    -1000/159   1000/371
  0       -125/32     125/12      -375/64
  0       9477/3392   -729/106    25515/6784
  0       -11/7       11/3        -55/28
  0       3/2     -4      5/2
      ];
  bstar = bstar*S;

end

f = zeros(2,7);
f0 = feval(dfcn,t,y);

mm = max(abs(f0./DY));
absh = hmax;
if mm
  absh = min(absh,1/(100*mm));
end

f(:,1) = f0;
minNsteps =20;

% THE MAIN LOOP

tic
while ~stop

  % hmin is a small number such that t+h is distinquishably
  % different from t if abs(h) > hmin.
  hmin = 16*eps*abs(t);
  absh = min(hmax, max(hmin, absh));
  h = tdir * absh;
```

```
% LOOP FOR ADVANCING ONE STEP.
while stop~=5
  % hC= h * C;
  hA = h * A;

  f(:,2) = feval(dfcn, t, y + f*hA(:,1));
  f(:,3) = feval(dfcn, t, y + f*hA(:,2));
  f(:,4) = feval(dfcn, t, y + f*hA(:,3));
  f(:,5) = feval(dfcn, t, y + f*hA(:,4));
  f(:,6) = feval(dfcn, t, y + f*hA(:,5));
  tn = t + h;
  yn = y + f*h*bhat;
  f(:,7) = feval(dfcn, tn, yn);

  % Estimate the error.
  err = abs(h * f * E);
  alpha = (2*max(err./((abs(y)+abs(yn)+DY*1e-3)*tol)))^pow;
  if alpha < 1         % Step is OK
break
  else
if absh <= hmin % This keeps us out of an infinite loop.
  stop = 5;
  break;
end

absh = max(hmin,0.8*absh / alpha);
h = tdir * absh;
  end  % if alpha < 1
end  % while stop~=5
steps = steps + 1;


oldN = N;
N = N + refine;
if N > length(tout)
  tout = [tout; zeros(block,1)];
  yout = [yout; zeros(block,2)];
end
if refine > 1             % computed points, with refinement
  j = oldN+1:N-1;
  tout(j) = t + h*sigma';
  yout(j,:) = (y(:,ones(length(j),1)) + f*h*bstar).';
  tout(N) = tn;
  yout(N,:) = yn.';
else               % computed points, no refinement
  tout(N) = tn;
  yout(N,:) = yn.';
end


% Update stop.   Maybe the stop button has been pressed.

ud = get(dispha,'user');
stop = max(ud.stop,stop);
```

```
if gstop
  % Are we out of the compute window?
  yl = yout(N,:).';
  if any([yl;-yl] - cwind < 0);
stop = 1;
  end

  % If the step in the phase plane is small we assume there is a sink.
  if (steps > minNsteps)
yy = yout(oldN:N,:).';
dyy = yy(:,1:refine) - yy(:,2:(refine+1));
dyy = dyy./rDY;
MMf = min(sqrt(sum(dyy.^2)));
if (MMf<=sinkeps*absh);
  z0 = yy(:,refine+1);
  zz = pplane8('newton',z0,dfcn);
  zz = zz(:,1);
  ud.zz = zz;
  nz = norm((zz-z0)./DY);
  if nz <= 0.01;
    stop = 2;
    ud.y = z0;
  end
  minNsteps = minNsteps + 20;
end

  end

  % We look for a maximum in the randomly chosen direction.  If
  % we find one, we compare it with previously found maxima.  If
  % our new one is close to an old one we stop.  If not, we
  % record the on.

  jj = oldN+1:N;
  yy = yout(jj,:).';
  rrr = R*yy;
  v = [rr,rrr];
  rr = v(:,[refine+1,refine+2]);   % Use this next time.
  [m,ii] = max(v(1,:));
  if( 1< min(ii) & max(ii)<refine+2 )  % If the max is in the middle.
kk=0;
while ( (kk<tk) & (~stop) )
  kk = kk+1;
  if ((abs(v(1,ii)-turn(1,kk))<perpeps) &...
      (abs(v(2,ii)-turn(2,kk))<paraeps) )
    z0 = yy(:,refine);
    ud.y = z0;
    zz = pplane8('newton',z0,dfcn);
    zz = zz(:,1);
    ud.zz = zz;
    nz = norm((zz-z0)./DY);
    if nz <= 0.015;
      stop = 2;
    else
      stop = 3;
```

```
   end
  end
 end
 tk = tk + 1;
 if tk > size(turn,2)
  turn = [turn,zeros(2,10)];
 end
 turn(:,tk+1) = v(:,ii);
  end
 elseif (abs(tn-tfinal) < hmin)
  stop = 6;
 end  % if gstop
 if plotf
  ii = oldN:N;
  set(ph,'Xdata',yout(ii,1),'Ydata',yout(ii,2));
  drawnow
 end  % if plotf

 % Compute a new step size.
 absh = max(hmin,0.8*absh / max( alpha,0.1));
 absh = min(absh,tdir*(tfinal - tn));
 h = absh*tdir;
 % Advance the integration one step.
 t = tn;
 y = yn;
 f(:,1) = f(:,7);     % Already evaluated
                    % dfcn(tnew,ynew)

 if slow
  ttt= N/(speed*refine);
  tt = toc;
  while tt < ttt
 tt = toc;
  end
 end

end  % while ~stop
ud.stop = stop;
set(dispha,'user',ud);
tout = tout(1:N);
yout = yout(1:N,:);
if dud.notice & dud.noticeflag
 nstr = get(dud.notice,'string');

 switch stop
  case 1
   nstr{5} = [nstr{5}, ' left the computation window.'];
  case 2
   ystr = ['(',num2str(zz(1),2), ', ', num2str(zz(2),2),').'];
   nstr{5} = [nstr{5}, ' --> a possible eq. pt. near ',ystr];
  case 3
   nstr{5} = [nstr{5}, ' --> a nearly closed orbit.'];
  case 4
   nstr{5} = [nstr{5}, ' was stopped by the user.'];
  case 5
   ystr = ['(',num2str(y(1),2), ', ', num2str(y(2),2),').'];
   nstr(1:3) = nstr(2:4);
```

```
      nstr{4} = [nstr{5},' experienced a failure at ',ystr];
      nstr{5} = 'Problem is singular or tolerances are too large.';
    end
    set(dud.notice,'string',nstr);
    drawnow
  end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [tout,yout] = pprk4(dfcn,tspan,y0,disph)

% PPRK4  is an implementation of the fourth order Runge-Kutta method.

% Input the user data.

dud = get(disph,'user');
dispha = dud.axes;
ud = get(dispha,'user');
refine = dud.settings.refine;
tol = dud.settings.tol;
gstop = ud.gstop;
ssize = dud.settings.stepsize;
plotf = ud.plot;
DY = ud.DY;
col = dud.color.temp;
speed = dud.settings.speed;
slow = (speed < 100);

% Initialize the stopping criteria.

%  The compute window.

if gstop

   % Initialize detection of closed orbits & limit cycles.
   % Choose a random direction & initialize the search for orbit maxima in
   % that direction.

   theta = rand*2*pi;
   R = [cos(theta), sin(theta); -sin(theta),cos(theta)];
   qq = R*(y0(:));
   rr = [qq,qq];
   z = DY(1) + sqrt(-1)*DY(2);
   w=exp(i*theta);
   a1 = w*z;
   a2 = w*(z');
   a=max(abs(real([a1,a2])));
   b=max(abs(imag([a1,a2])));
   perpeps = a*0.00001; % We want to be real
            % close in this direction.
   paraeps = b/100; % We do not have to be
            % so careful in this direction.
   tk = 0;
   turn = zeros(2,10);

   % The test for an equilibrium point.

   sinkeps = 0.0001;

   %  The compute window.

   cwind = ud.cwind;
 end

% The stop button.

stop = 0;
ud.stop = 0;

% Set the the line handle.

ph = plot([y0(1),y0(1)],[y0(2),y0(2)],'color',col,...
   'erase','none',...
   'parent',dispha);
ud.line = ph;
set(dispha,'UserData',ud);

% Initialize the loop.

t0 = tspan(1);
tfinal = tspan(2);
tdir = sign(tfinal - t0);
t = t0;
y = y0(:);

h = ssize*tdir;

steps = 0;
block = 120;
tout = zeros(block,1);
yout = zeros(block,2);
N = 1;
tout(N) = t;
yout(N,:) = y.';
minNsteps =20;

% The main loop
tic
while ~stop
  if abs(t - tfinal) < ssize
    h = tfinal - t;
  end

  % Compute the slope
  s1 = feval(dfcn,t,y); s1=s1(:);
  s2 = feval(dfcn, t + h/2, y + h*s1/2); s2=s2(:);
  s3 = feval(dfcn, t + h/2, y + h*s2/2); s3=s3(:);
```

```
s4 = feval(dfcn, t + h, y + h*s3); s4=s4(:);

t = t + h;
y = y + h*(s1 + 2*s2 + 2*s3 +s4)/6;

if N >= length(tout)
  tout = [tout;zeros(block,1)];
  yout = [yout;zeros(block,2)];
end
oldN = N;
N = N + 1;
tout(N) = t;
yout(N,:) = y.';
steps = steps + 1;

% Update stop.   Maybe the stop button has been pressed.

ud = get(dispha,'user');
stop = max(ud.stop,stop);

if gstop
  % Are we out of the compute window?
  yl = yout(N,:).';
  if any([yl;-yl] - cwind < 0);
    stop = 1;
  end

  % If the step in the phase plane is small we assume there is a sink.
  if (steps > minNsteps)
    yy = yout(N-1:N,:).';
    dyy = yy(:,1) - yy(:,2);
    dyy = dyy./DY;
    MMf = sqrt(sum(dyy.^2));
    if (MMf<=sinkeps)
  z0 = yy(:,refine+1);
  zz = pplane8('newton',z0,dfcn);
  zz = zz(:,1);
  ud.zz = zz;
  nz = norm((zz-z0)./DY);
  if nz <= 0.01;
    stop = 2;
    ud.y = z0;
  end
  minNsteps = minNsteps + 20;
    end
  end

  % We look for a maximum in the randomly chosen direction.  If
  % we find one, we compare it with previously found maxima.  If
  % our new one is close to an old one we stop.  If not, we
  % record the on.

  yy = yout(N,:).';
  rrr = R*yy;
  v = [rr,rrr];
```

```
    rr = v(:,[2,3]);   % Use this next time.
    [m,ii] = max(v(1,:));
    if( 1< min(ii) & max(ii)<3 )  % If the max is in the middle.
      kk=0;
      while ( (kk<tk) & (~stop) )
    kk = kk+1;
    if ((abs(v(1,ii)-turn(1,kk))<perpeps) &...
          (abs(v(2,ii)-turn(2,kk))<paraeps) )
      z0 = yy(:,refine);
      ud.y = z0;
      zz = pplane8('newton',z0,dfcn);
      zz = zz(:,1);
      ud.zz = zz;
      nz = norm((zz-z0)./DY);
      if nz <= 0.015;
        stop = 2;
      else
        stop = 3;
      end
    end
      end
      tk = tk + 1;
      if tk > size(turn,2)
    turn = [turn,zeros(2,10)];
      end
      turn(:,tk+1) = v(:,ii);
    end
  end  % if gstop
  if (abs(t-tfinal) < 0.001*ssize)
    stop = 6;
  end

  if plotf
    nn = (N-1):N;
    set(ph,'Xdata',yout(nn,1),'Ydata',yout(nn,2));
    drawnow
  end  % if plotf
  if slow
    ttt= N/(speed);
    tt = toc;
    while tt < ttt
      tt = toc;
    end
  end

end  % while ~stop

ud.stop = stop;
set(dispha,'user',ud);
tout = tout(1:N);
yout = yout(1:N,:);
if dud.notice & dud.noticeflag
   nstr = get(dud.notice,'string');

   switch stop
   case 1
```

```
      nstr{5} = [nstr{5}, ' left the computation window.'];
   case 2
      ystr = ['(',num2str(zz(1),2), ', ', num2str(zz(2),2),').'];
      nstr{5} = [nstr{5}, ' --> a possible eq. pt. near ',ystr];
   case 3
      nstr{5} = [nstr{5}, ' --> a nearly closed orbit.'];
   case 4
      nstr{5} = [nstr{5}, ' was stopped by the user.'];
   case 5
      ystr = ['(',num2str(y(1),2), ', ', num2str(y(2),2),').'];
      nstr(1:3) = nstr(2:4);
      nstr{4} = [nstr{5},' experienced a failure at ',ystr];
      nstr{5} = 'Problem is singular or tolerances are too large.';
   end
   set(dud.notice,'string',nstr);
end
drawnow

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function output = ppout(t,y,flag,varargin)

% PPOUT  is the output function for pplane8.

%  Copywright (c)  John C. Polking, Rice University
%  Last Modified: January 21, 2001

output = 0;
ppdisp = findobj(get(0,'child'),'flat','name','pplane8 Display'); %gca;   %
dud = get(ppdisp,'user');
ppdispa = dud.axes;
dfcn = dud.function;
ud = get(ppdispa,'user');
stop = ud.stop;
gstop = ud.gstop;
col = dud.color.temp;
plotf = ud.plot;
speed = dud.settings.speed;
DY = ud.DY;
slow = (speed < 100);

if (nargin < 3) | (isempty(flag))
    if stop
        output = 1;
    vers = version;
    vers = str2num(vers(1:3));
    if vers<6.5
      feval(@ppout,t,y,'done');
    end
    else
        L = length(t);
        if gstop
            % Update stop.   Are we out of the compute window?
            yl = y(:,L);
            if any([yl;-yl] - ud.cwind < 0);
        stop = 1;
    end

    % If the derivative function is small we assume there is a
    % sink.

minNsteps = ud.minNsteps;
    if (ud.i > minNsteps)
        yy = [ud.y,y];
        dyy = yy(:,1:L) - yy(:,2:(L+1));
        rDY = DY(:,ones(1,L));
        dyy = dyy./rDY;
        MMf = min(sqrt(sum(dyy.^2)));
        if (MMf<=ud.sinkeps*abs(t(1) - t(L)))
  z0 = yy(:,L+1);
  zz = pplane8('newton',z0,dfcn);
  zz = zz(:,1);
  ud.zz = zz;
  nz = norm((zz-z0)./DY);
  if nz <= 0.01;
    ud.zz = zz;
    stop = 2;
  end
  ud.minNsteps = minNsteps + 20;
end
    else
        ud.i = ud.i + 1;
    end

    % We look for a maximum in the randomly chosen direction.  If
    % we find one, we compare it with previously found maxima.  If
    % our new one is close to an old one we stop.  If not, we
    % record the position.

    rr = ud.R*y;

    v = [ud.rr,rr];
    ud.rr = v(:,[L+1,L+2]);
    [m,ii] = max(v(1,:));
    %ii = ii(1);
    if( 1< min(ii) & max(ii)<L+2 )
        kk=0;
        turn = ud.turn;
        perpeps = ud.perpeps;
        paraeps = ud.paraeps;
        tk = ud.tk;
        while ( (kk<tk) & (~stop) )
            kk = kk+1;
            if ((abs(v(1,ii)-turn(1,kk))<perpeps) &...
                    (abs(v(2,ii)-turn(2,kk))<paraeps) )
      z0 = y(:,L);
      zz = pplane8('newton',z0,dfcn);
      zz = zz(:,1);
      ud.zz = zz;
      nz = norm((zz-z0)./DY);
      if nz <= 0.002;
```

```
            ud.zz = zz;
            stop = 2;
              else
                        stop = 3;
              end
                    end
                end
                ud.tk = tk + 1;
                if tk >= size(turn,2)
                    ud.turn = [turn,zeros(2,10)];
                end
                ud.turn(:,tk+1) = v(:,ii);
            end
        end
        output = 0;
        ud.stop = stop;
        yold = ud.y;
        ud.y = y(:,L);
        set(ppdispa,'user',ud);
    if slow
      ttt = clock;
      newtime = (24*ttt(4)+ttt(5))*60 + ttt(6);
      ctime = ud.ctime;
      N = ud.i;
      while newtime < ctime + N/speed
        ttt = clock;
        newtime = (24*ttt(4)+ttt(5))*60 + ttt(6);
      end
        end
        % Finally we plot the newest line segment.
        if plotf
      set(ud.line,'Xdata',[yold(1),y(1,:)],'Ydata',[yold(2),y(2,:)]);
      drawnow
    end
    end

else
  switch(flag)
   case 'init'                        % ppout(tspan,y0,'init')
    if slow
      ttt = clock;
      ctime = (24*ttt(4)+ttt(5))*60 + ttt(6);
      ud.ctime = ctime;
    end
        ud.y = y(:);
        ud.i = 1;

        % Set the the line handle.
        figure(ppdisp);
        ud.line = plot([ud.y(1),ud.y(1)],[ud.y(2),ud.y(2)],...
            'color',col,'erase','none');

        if gstop
            % Chose a random direction & initialize the search for orbit
            % maxima in that direction.
            theta = rand*2*pi;
            ud.R = [cos(theta), sin(theta); -sin(theta),cos(theta)];
            qq = ud.R*y;
            ud.rr = [qq,qq];
            z = ud.DY(1) + sqrt(-1)*ud.DY(2);
            w = exp(i*theta);
            r = abs(z);
            a1 = w*z;
            a2 = w*(z');
            a = max(abs(real([a1,a2])));
            b = max(abs(imag([a1,a2])));
            ud.perpeps = a*0.00001;;    % We want to be real
            % close in this direction.
            ud.paraeps = b/100; % We do not have to be
            % too careful in this direction.
            ud.sinkeps = 0.005/dud.settings.refine;
        ud.minNsteps = 20;
            ud.tk = 0;
            ud.turn = zeros(2,10);
        end
        output = 0;
        ud.stop = 0;
        set(ppdispa,'UserData',ud);

    case 'done'         % ppn6(t,y,'done');
     if dud.noticeflag
       nstr = get(dud.notice,'string');
       if ~isempty(y)
     set(ud.line,'Xdata',[ud.y(1),y(1,:)],'Ydata',[ud.y(2),y(2,:)]);
       end
       switch stop
    case 1
     nstr{5} = [nstr{5}, ' left the computation window.'];
    case 2
     yy = ud.zz;
     ystr = ['(',num2str(yy(1),2), ', ', num2str(yy(2),2),').'];
     nstr{5} = [nstr{5}, ' --> a possible eq. pt. near ',ystr];
    case 3
     nstr{5} = [nstr{5}, ' --> a nearly closed orbit.'];
    case 4
     nstr{5} = [nstr{5}, ' was stopped by the user.'];
       end
       set(dud.notice,'string',nstr);
       drawnow
       output = 1;
     end
  end
end
```