Raglan_L1

January 11, 2016

1 NZMRI Raglan Summer School 2016

1.1 Multiparameter Bifurcations of Planar Vector Fields

- Dynamical systems theory studies parameterized vector fields

$$\dot{x} = f(x, \lambda),\ x \in \mathbf{R}^n,\ \lambda \in \mathbf{R}^k$$

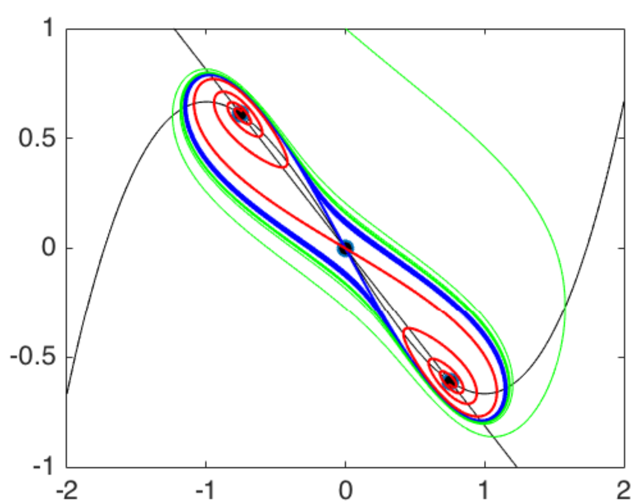
- The example used in this notebook is the FitzHugh-Nagumo model given by the equations

$$\dot{x} = y - x^3/3 + x\dot{y} = -e(ax + b + cy)$$

- Programming is in Matlab. Solutions of initial value problems are computed with dop853 (thanks to Madhu Venkadesan), a program from the Hairer-Wanner collection of initial value solvers.

Here is a phase portrait

In [1]:
```
p=[13/16; 0; 1; 0.544];
eqm = fhn_eq(p)
fhn_nullcline(p);
%Stable and unstable manifolds
[ws1,ws2,wu1,wu2] = fhn2_smflds(eqm(1,:),p,1e-5);
options = dopset('AbsTol',1e-14,'RelTol',1e-12,'MaxStepSize',0.01,'MaxSteps',1e7,'EventTol',1e-
[ts,pts,te,ye,ie,stats] = dop853('fhn2',[0 2000],[0,1],options,p);
plot(pts(:,1),pts(:,2),'g')
axis([-2,2,-1,1])
%
```



```
Out[1]: eqm =

             0         0
        0.7500   -0.6094
       -0.7500    0.6094


        eqm =

             0         0
        0.7500   -0.6094
       -0.7500    0.6094


        V =

        0.9349   -0.6515
       -0.3549    0.7586


        D =

        0.6204         0
             0   -0.1644
```

1.1.1 Goal: Compute phase portraits and map parameter spaces

Identify regions of parameters with same qualitative dynamics

- Topological equivalence is "gold standard"
- Bifurcations occur where qualitative dynamics changes
- Defining equations characterize bifurcations

Saddle-node bifurcations occur when the Jacobian at an equilibrium has a 0 eigenvalue.

Hopf bifurcations occur when the Jacobian at an equilibrium has a pair of pure imaginary eigenvalues.

In the FitzHugh-Nagumo model in the form above, one parameter is redundant. We normalize the family by setting $c = 1$.

The parameter equation for saddle-node bifurcation (tan below) is

$$9b^2 + 4(a-1)^3 = 0$$

For Hopf bifurcation (blue below)

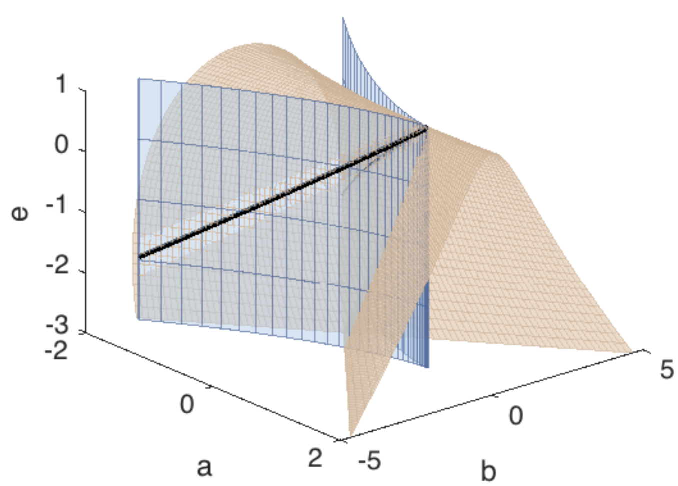
$$9b^2 + (e-1)(3(a-1) - (e-1)^2 = 0$$

To plot these surfaces, we use a parameteric representation which is easier to find than the formulas above.

Takens-Bogdanov bifurcation occurs where there is a zero eigenvalue of multiplicity 2. This is shown as a black curve in the figure below.

```
In [2]: hold on
        u = [-2:0.05:2];
        [x,a]  = meshgrid(u);
```

```
surf(a,-a.*x-x.^3/3+x,1-x.^2, ...
    'FaceAlpha',0.7,'EdgeAlpha',0.7,...
    'FaceColor',[0.9 0.8 0.7],...
    'EdgeColor',[0.8 0.7 0.6]);

t = [-3/2:0.05:3/2];
em = [-3:1];
[e,xm] = meshgrid(em,t);
surf(1-(9/4)^(1/3)*xm.^2,xm.^3,e, ...
'FaceAlpha',0.5,'EdgeAlpha',0.7,...
    'FaceColor',[0.7 0.8 0.9],...
    'EdgeColor',[0.3 0.4 0.6]);

plot3(1-(9/4)^(1/3)*t.^2,t.^3,1-(9/4)^(1/3)*t.^2,'k','LineWidth',2)

xlabel('a');
ylabel('b');
zlabel('e');

view([50 30]);
```



Remarks:

- Both of the bifurcation surfaces above have singularities.
- The saddle-node surface has a line of *cusps*
- The Hopf surface has a *Whitney umbrella*
- Continuation methods compute curves along bifurcation surfaces without explicit formulas
- Little software for manifolds beyond curves: *Multifario* by Michael Henderson

1.1.2 Structurally stable planar vecor fields:

Necessary and sufficient conditions:

- Hyperbolic equilibrium points

- Hyperbolic periodic orbits
- No saddle connections

Kupka-Smale: these conditions are generic (Baire category)

Poincare-Bendixson theorem precludes chaotic dynamics: Forward limit set of bounded trajectory is either a periodic orbit or contains an equilibrium

1.1.3 Codimension one bifurcations of planar vector fields:

- Saddle-nodes of equilibria
- Hopf bifurcations
- Saddle-nodes of periodic orbits
- Saddle connections

Saddle-nodes Defining equations:

$$f(x, \lambda) = 0$$
$$\det(D_x f)(x, \lambda) = 0$$

Regularity of defining equation:

$$\det \begin{pmatrix} D_x f & D_\lambda f \\ D_x(\det(D_x f)) & D_\lambda(\det(D_x f)) \end{pmatrix} \neq 0$$

- Rank deficiency of $(D_x f)$ is 1 with right eigenvector v, left eigenvector w.
- $w \cdot D_\lambda f \neq 0$
- $w \cdot D_{xx} f(v, v) \neq 0$

Eigenvalue is simple if $w \cdot v \neq 0$
Normal form on center manifold: $\dot{x} = x^2 + \lambda$

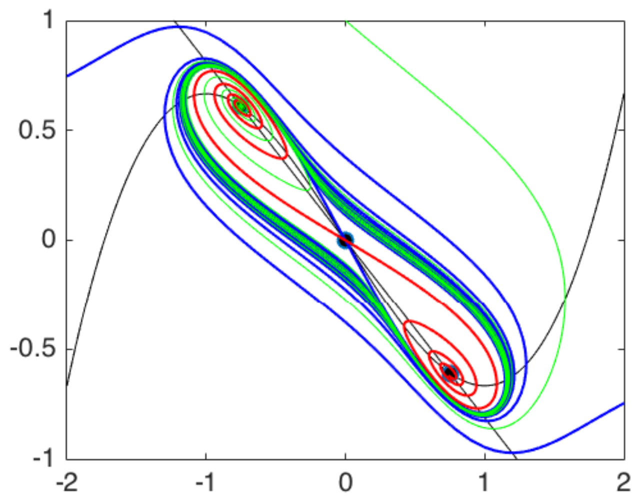
1.1.4 Periodic orbits

Two strategies for computing periodic orbits and stability

- Shooting methods
- Initial value solvers: explicit or implicit
- Cross-sections and return maps: event stopping
- Newton's method for fixed point of return map
- Stability via variational equations or finite differences
- Global methods
- Solve ODE on space of closed curves
- Discretize space of curves and project ODEs
- Collocation (implemented in AUTO)
- Continuous piecewise polynomials
- Solve ODE on inner mesh: superconvergence possible
- Newton's method on large system with structured Jacobians
- Adaptive meshing equidistributes errors
- Jacobians contain stability information

FitzHugh-Nagumo near a saddle-node of periodic orbits

In [3]:
```
p = [13/16; 0; 1; 0.545];
eqm = fhn_eq(p)
fhn_nullcline(p);
%Stable and unstable manifolds
[ws1,ws2,wu1,wu2] = fhn2_smflds(eqm(1,:),p,1e-5);
options = dopset('AbsTol',1e-14,'RelTol',1e-12,'MaxStepSize',0.01,'MaxSteps',1e7,'EventTol',1e-
[ts,pts,te,ye,ie,stats] = dop853('fhn2',[0 2000],[0,1],options,p);
plot(pts(:,1),pts(:,2),'g')
axis([-2,2,-1,1])
```

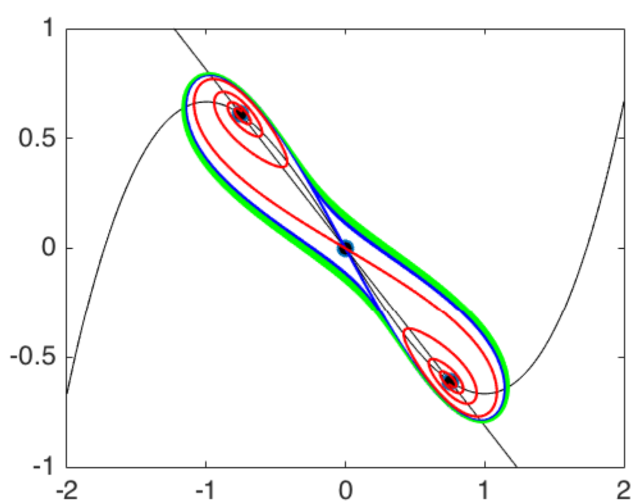


Out[3]:
```
eqm =

         0         0
    0.7500   -0.6094
   -0.7500    0.6094


eqm =

         0         0
    0.7500   -0.6094
   -0.7500    0.6094


V =

    0.9347   -0.6514
   -0.3553    0.7588


D =
```

```
    0.6199         0
         0   -0.1649
```

Use bisection to get close to bifurcation. Then plot return map.

In [4]:
```
%The parameters
p = [13/16; 0; 1; 0.5443639468]
axis([-2,2,-1,1])
%Equilibrium points and nullclines
eqm = fhn_eq(p)
fhn_nullcline(p);
%Stable and unstable manifolds
[ws1,ws2,wu1,wu2] = fhn2_smflds(eqm(1,:),p,1e-5);
axis([-2,2,-1,1])

%Plot 100 trajectories to first return with increasing coordinate for
%cross-section that determines linear nullcline
xn = 100;
%x coordinate increment between intial points
xinc = 8e-5;
%matrix of initial points
xin = -0.984 +8e-5*[1:100];
yin = (-p(1)*xin-p(2))./p(3);
pin = [xin',yin'];
%For output - checking that return is to correct region
pin2= [];
pout = [];
options = dopset('AbsTol',1e-14,'RelTol',1e-12,'MaxStepSize',0.01,'MaxSteps',1e7,'EventTol',1e-
%Compute the trajectories and store initial and final points
for j=1:xn
  [ts,pts,te,ye,ie,stats] = dop853('fhn_el2',[0 2000],pin(j,:),options,p);
  plot(pts(:,1),pts(:,2),'g')
  if (pts(end,1) <-0.75)
      pin2 = [pin2;pin(j,:)];
  pout = [pout;pts(end,:)];
  end
end
```

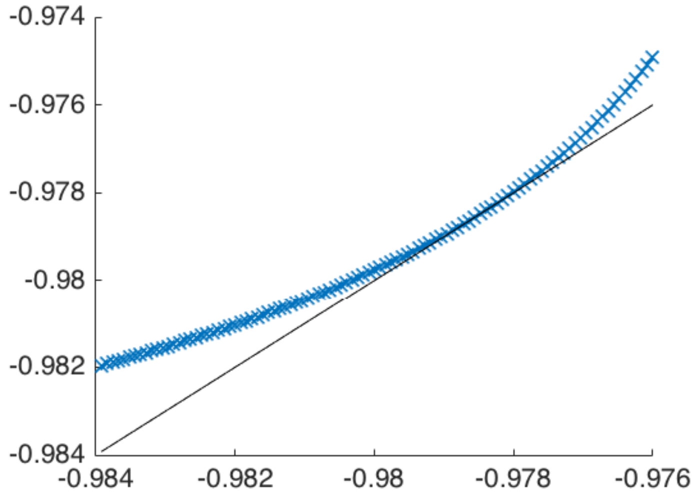


Out[4]: p =

```
    0.8125
         0
    1.0000
    0.5444


eqm =

         0         0
    0.7500   -0.6094
   -0.7500    0.6094


eqm =

         0         0
    0.7500   -0.6094
   -0.7500    0.6094


V =

    0.9348   -0.6515
   -0.3550    0.7587


D =

    0.6202         0
         0   -0.1646
```

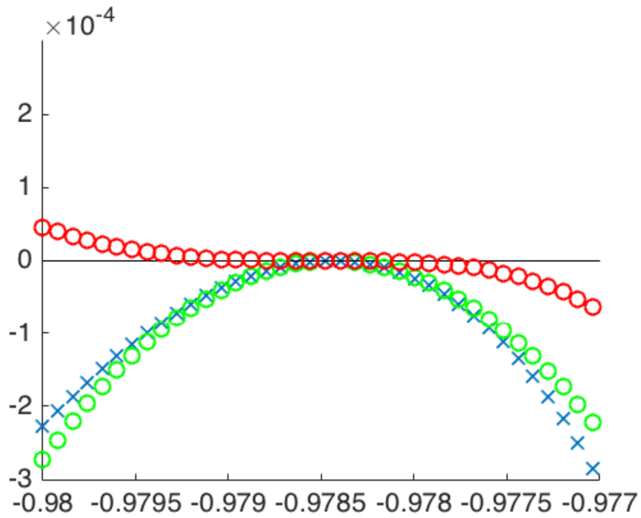
In [5]:

```
%Plot return map
figure(2)
clf
hold on
plot(pin2(:,1),pout(:,1),'x')
plot([pin2(1,1),pin2(end,1)],[pin2(1,1),pin2(end,1)],'k')
```



Analyze the return data

In [6]:
```
%Compute derivatives for return
%difference of x-coordinate endpoints
xd = pin2(:,1) -pout(:,1);
%Plot x-coordinate differences vs. initial values of x
figure(2)
clf
hold on
plot(pin2(:,1),xd,'x')
plot([pin2(1,1),pin2(end,1)],[0,0],'k')
%Closest return to a fixed point
[resid,ind] = min(abs(pin2-pout))
ind = ind(1);
%Compute first and second derivatives of x-coordinate endpoint differences
x0 = pin2(ind,1);
y0 = xd(ind,1)
dx0 =  (xd(ind+1)-xd(ind-1))/(2*xinc)
dxx0 = (xd(ind+1)-2*xd(ind)+xd(ind-1))/(xinc^2)
%Quadratic fit to return
qfit = y0 + dx0*(pin2(:,1)-x0) + dxx0*(pin2(:,1)-x0).^2/2;
%Plot quadratic green
plot(pin2(:,1),qfit,'go')
%Plot residual red
plot(pin2(:,1),xd-qfit,'ro')
axis([-0.98,-0.977,-3e-4,3e-4])
```

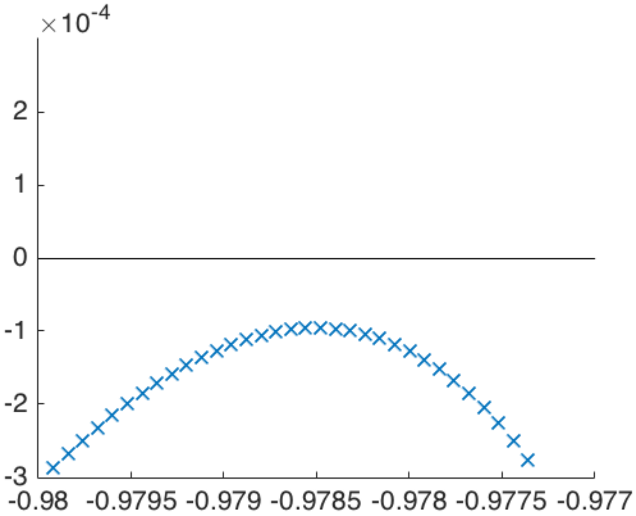


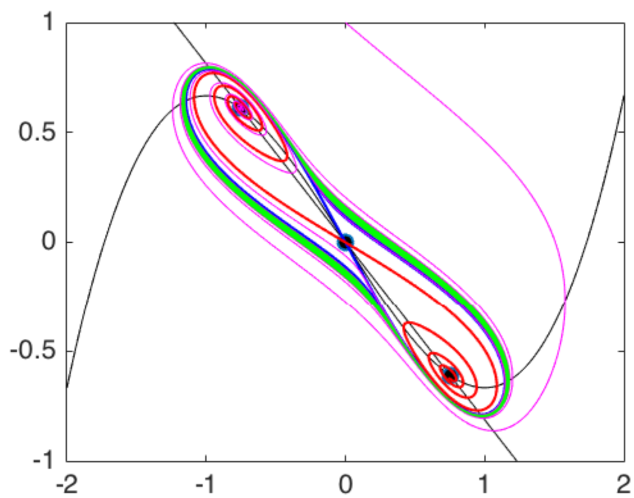

```
Out[6]: resid =

           1.0e-10 *

            0.3646    0.2962


        ind =

            69    69


        y0 =

           3.6458e-11


        dx0 =

            0.0085


        dxx0 =

         -225.5491
```

Repeat for parameters without a periodic orbit

```
In [7]: %The parameters
        p = [13/16; 0; 1; 0.5444]
        axis([-2,2,-1,1])
        %Equilibrium points and nullclines
        eqm = fhn_eq(p)
        fhn_nullcline(p);
```

```
%Stable and unstable manifolds
[ws1,ws2,wu1,wu2] = fhn2_smflds(eqm(1,:),p,1e-5);
axis([-2,2,-1,1])

%Plot one trajectory without stopping
[ts,pts,te,ye,ie,stats] = dop853('fhn2d',[0 2000],[0,1],options,p);
plot(pts(:,1),pts(:,2),'m','Linewidth',0.5)

%Plot 100 trajectories to first return with increasing coordinate for
%cross-section that determines linear nullcline
xn = 100;
%x coordinate increment between intial points
xinc = 8e-5;
%matrix of initial points
xin = -0.984 +8e-5*[1:100];
yin = (-p(1)*xin-p(2))./p(3);
pin = [xin',yin'];
%For output - checking that return is to correct region
pin2= [];
pout = [];
options = dopset('AbsTol',1e-14,'RelTol',1e-12,'MaxStepSize',0.01,'MaxSteps',1e7,'EventTol',1e-
%Compute the trajectories and store initial and final points
for j=1:xn
  [ts,pts,te,ye,ie,stats] = dop853('fhn_el2',[0 2000],pin(j,:),options,p);
  plot(pts(:,1),pts(:,2),'g')
  if (pts(end,1) <-0.75)
      pin2 = [pin2;pin(j,:)];
  pout = [pout;pts(end,:)];
  end
end
%Compute derivatives for return
%difference of x-coordinate endpoints
xd = pin2(:,1) -pout(:,1);
%Plot x-coordinate differences vs. initial values of x
figure(2)
clf
hold on
plot(pin2(:,1),xd,'x')
plot([pin2(1,1),pin2(end,1)],[0,0],'k')
%Closest return to a fixed point
[resid,ind] = min(abs(pin2-pout))
ind = ind(1);
%Compute first and second derivatives of x-coordinate endpoint differences
x0 = pin2(ind,1);
y0 = xd(ind,1)
dx0 =  (xd(ind+1)-xd(ind-1))/(2*xinc)
dxx0 = (xd(ind+1)-2*xd(ind)+xd(ind-1))/(xinc^2)
%Quadratic fit to return
qfit = y0 + dx0*(pin2(:,1)-x0) + dxx0*(pin2(:,1)-x0).^2/2;
%Plot quadratic green
%plot(pin2(:,1),qfit,'go')
%Plot residual red
%plot(pin2(:,1),xd-qfit,'ro')
axis([-0.98,-0.977,-3e-4,3e-4])
```






```
Out[7]: p =

    0.8125
         0
    1.0000
    0.5444


eqm =

         0         0
    0.7500   -0.6094
   -0.7500    0.6094
```

```
eqm =

         0         0
    0.7500   -0.6094
   -0.7500    0.6094


V =

    0.9348   -0.6515
   -0.3551    0.7587


D =

    0.6202         0
         0   -0.1646


resid =

   1.0e-04 *

    0.9654    0.7844


ind =

    69    69


y0 =

  -9.6541e-05


dx0 =

   -0.0057


dxx0 =

 -231.3807
```

1.1.5 Newton iteration

Now we want to use Newton iteration to obtain a more accurate value of the parameter at the saddle-node of limit cycles. The linear nullcline is chosen as a cross-section. (Any periodic orbit must cross both nullclines.) Three initial points are chosen on the nullcline, separated by distances proportional to xinc, defined above. The value of the return map σ at these points is computed, together with a centered difference estimate of σ' at the middle point. The defining equations for a saddle-node of limit cycles are that $\sigma(x) = x$ and

$\sigma'(x) = 1$

In [8]:
```
% Starting parameters and initial point
format long
p0 = [13/16; 0; 1; 0.5443639468];
z0 = [-0.97848,0.795015];
p = p0
z = z0
% Finite difference increments
xinc = 3e-4;
pinc = 1e-5;
% maximum number of Newton iterations
nsteps = 20;
resid = [];
for j=1:nsteps
% Trajectories for computing x derivatives
options = dopset('AbsTol',1e-14,'RelTol',1e-12,'MaxStepSize',0.01,'MaxSteps',1e7,'EventTol',1e-
[ts,pts,te,ye,ie,stats] = dop853('fhn_el2',[0 2000],z,options,p);
zout = pts(end,:);
% Residual of sigma(x) - x
xd = zout(1)-z(1);
[ts,pts,te,ye,ie,stats] = dop853('fhn_el2',[0 2000],z+[xinc,-p(1)*xinc/p(3)],options,p);
zr = pts(end,:);
xr = zr(1) - z(1) - xinc;
[ts,pts,te,ye,ie,stats] = dop853('fhn_el2',[0 2000],z-[xinc,-p(1)*xinc/p(3)],options,p);
zl = pts(end,:);
xl = zl(1) - z(1) + xinc;
% Residual of sigma'(x) - 1
dx = (xr-xl)./(2*xinc);
% Accumulate residuals
resid = [resid;[xd,dx]];
% Convergence test
if abs(xd) < 1e-10 && abs(dx) < 1e-10
    j=j
    resid
    z
    p
    return
end
% sigma''(x) for Jacobian
dxx = (zr(1)-2*zout(1)+zl(1))/(xinc^2);
%
% Increment parameter p(4)
pp = p+pinc*[0;0;0;1];
% Compute trajectories for parameters pp
[ts,pts,te,ye,ie,stats] = dop853('fhn_el2',[0 2000],z,options,pp);
zoutp = pts(end,:);
xdp = zoutp(1)-z(1);
[ts,pts,te,ye,ie,stats] = dop853('fhn_el2',[0 2000],z+[xinc,-p(1)*xinc/p(3)],options,pp);
zrp = pts(end,:);
xrp = zrp(1) - z(1) - xinc;
[ts,pts,te,ye,ie,stats] = dop853('fhn_el2',[0 2000],z-[xinc,-p(1)*xinc/p(3)],options,pp);
zlp = pts(end,:);
xlp = zlp(1) - z(1) + xinc;
```

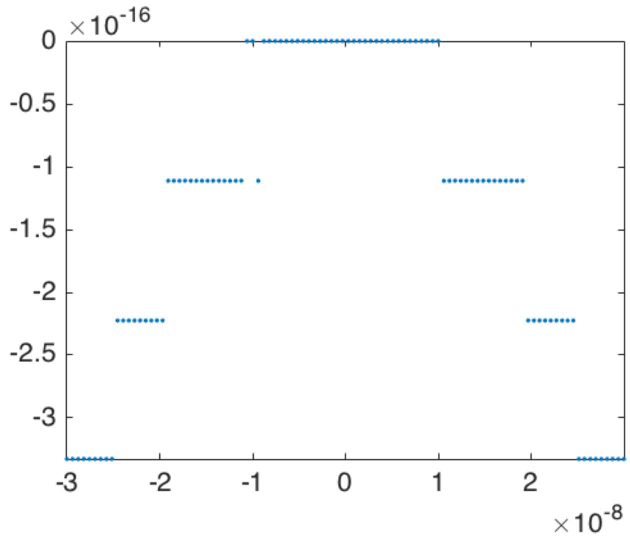
```
dxp = (xrp-xlp)./(2*xinc);
% Derivatives with respect to p(4)
dpar = (xdp-xd)/pinc;
dxpar = (dxp-dx)/pinc;
%Jacobian for defining function (\sigma - id,\sigma' - 1)
jac = [[dx,dpar];[dxx,dxpar]];
%
% Newton step
newt_delta = jac\[xd;dx];
xpnew = [z(1);p(4)] - newt_delta;
monitor = [xd,dx,newt_delta(2)]
% Update z and p
z = [xpnew(1),(-p(1)*xpnew(1)-p(2))/p(3)];
p = [p(1:3);xpnew(2)];
end
p0(4)-p(4)
```

Out[8]:
```
p =

   0.812500000000000
                   0
   1.000000000000000
   0.544363946800000


z =

  -0.978480000000000   0.795015000000000


monitor =

  -0.000000000036458  -0.007128927614822  -0.000000083933687


monitor =

   1.0e-04 *

   0.000662913762772   0.638614027224970   0.000247311646362


monitor =

   1.0e-07 *

   0.004601613534660  -0.740449543765292   0.001717175267417


monitor =

   1.0e-09 *

  -0.000253908005732   0.539458343447319  -0.000094750367979
```

```
j =

     5


resid =

  -0.000000000036458  -0.007128927614822
   0.000000066291376   0.000063861402722
   0.000000000460161  -0.000000074044954
  -0.000000000000254   0.000000000539458
   0.000000000000008  -0.000000000005476



z =

  -0.978448834514833   0.794989678043302



p =

   0.812500000000000
                   0
   1.000000000000000
   0.544364005830899
```

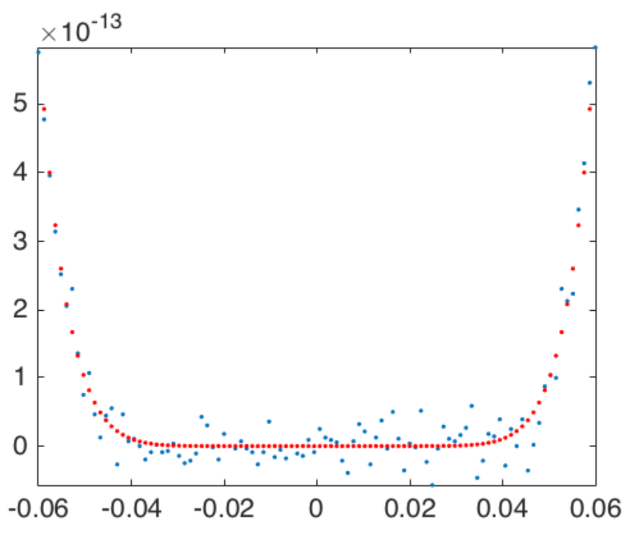
One problem with this Newton iteration is that numerical identification of *where* a function has an extreme value is difficult. Here are two examples that illustrate this difficulty. In the first, we plot computed values of the function $f(x) = \sin(x) - x/2$ and see that the discretization of floating point values makes the function appear constant over an interval of length comparable to 1e-8.

```
In [9]: xinc = 3e-8;
        x = linspace(pi/3-xinc,pi/3+xinc);
        y = sin(x) - x/2 - sqrt(3)/2+pi/6;
        figure(1)
        plot(x-pi/3,y,'.')
        axis([-xinc,xinc,min(y),max(y)])
```



The second example writes $f(u) = (1-u)^{10}$ in its expanded form. The plot shows that cancellation in the round-off evaluation of different terms can produce "noisy" values while evaluation of the function in the unexpanded form does not.

In [10]:
```
uinc = 0.06;
u = linspace(1-uinc,1+uinc);
v = u .^ 10 - 10 * u .^ 9 + 45 * u .^ 8 - 120 * u .^ 7 + 210 * u .^ 6 - 252 * u .^ 5 + 210 * u
w = (1-u).^10;
figure(2)
plot(u-1,v,'.',u-1,w,'r.')
axis([-uinc,uinc,min(v),max(v)])
min(v)
```



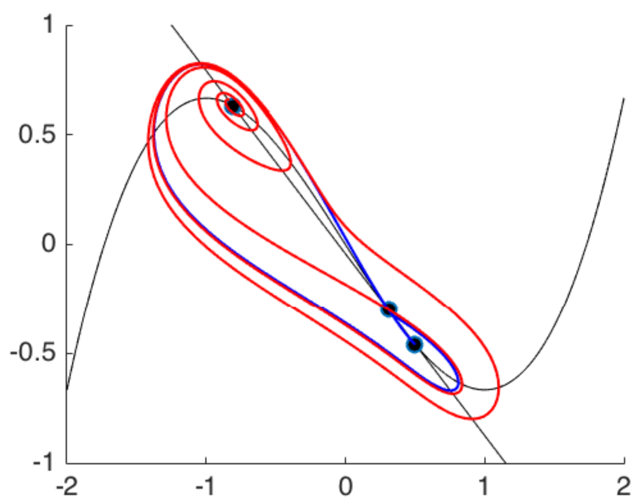
Out[10]:
```
ans =

      -5.684341886080801e-14
```

These examples suggest that it is hardly possible to locate precisely where the return map $\sigma(z)$ of the FitzHugh-Nagumo model has an extreme point for $\sigma(z) - z$. However, it does seem feasible to estimate accurately how the extreme value depends upon the parameter e. With this in mind, we leave at as a challenge to implemnt such an algorithm and embed it into a continuation scheme.

1.1.6 Homoclinic orbit computation and continuation

Strategy: apply secant method to intersections of stable and unstable manifolds with cross-section

In [11]:
```
p = [1, 0.05, 1.2, 0.388,1,-1]
p0 = p;
figure(1)
clf
hold on
eqm = fhn_eq(p);
fhn_nullcline(p);
[ws1,ws2,wu1,wu2] = fhn2_smflds(eqm(3,:),p,1e-5);
options = dopset('AbsTol',1e-14,'RelTol',1e-12,'EventTol',1e-14,'Events',1);
[ts,pts,te,ye,ie,stats] = dop853('fhn_el',[0 2000],[1,0],options,p);
axis([-2,2,-1,1])
```



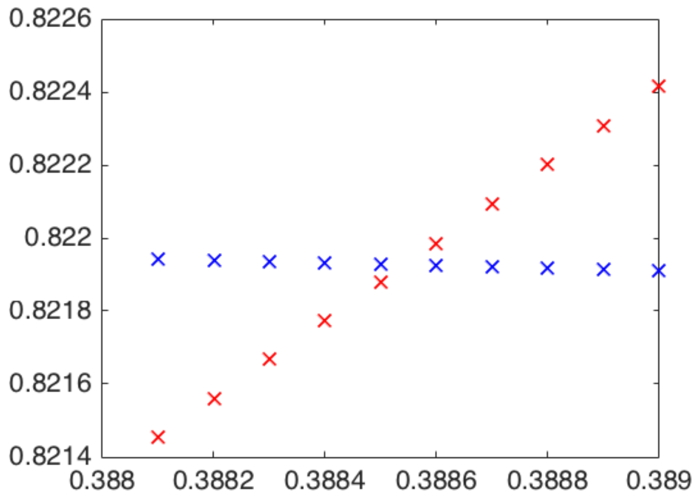
Out[11]:
```
p =

  Columns 1 through 3

   1.000000000000000   0.050000000000000   1.200000000000000

  Columns 4 through 6

   0.388000000000000   1.000000000000000  -1.000000000000000


eqm =
```

```
  -0.809016994374948   0.632514161979123
   0.500000000000001  -0.458333333333334
   0.309016994374947  -0.299180828645789


V =

   0.928501058429452  -0.717725457351455
  -0.371329751697044   0.696326193582896


D =

   0.504584610913027                   0
                   0  -0.065676113725553
```

Illustrate transversal crossing of manifold intersections as parameter e is varied

In [12]:
```
np = 10;
pinc = 1e-4;
vinc = 1e-7;
sdata = zeros(np,2);
udata = zeros(np,2);
p4 = p(4)+pinc*[1:np];
for j= 1:np
    p(4) = p4(j);;
    eqm = fhn_eq(p);
    jac = fhn2_jac(eqm(1,:),p);
    if jac(1,1)*jac(2,2) - jac(1,2)*jac(2,1) < 0
        saddle = eqm(1,:);
    else
        jac = fhn2_jac(eqm(2,:),p);
        if jac(1,1)*jac(2,2) - jac(1,2)*jac(2,1) < 0
        saddle = eqm(2,:);
        else
        jac = fhn2_jac(eqm(3,:),p);
        saddle = eqm(3,:);
        end
    end
    [V,D] = eig(jac);
    if D(1,1) < 0
        sind = 1;
    else
        sind = 2;
    end
    uind = 3-sind;
    p(6) = -1;
    xs = saddle+vinc*sign(V(2,sind))*[V(1,sind),V(2,sind)];
    [ts1,ws1,te,ye,ie,stats] = dop853('fhn_el3',[0 -2000],xs,options,p);
    p(6) = 1;
    xu = saddle-vinc*sign(V(2,uind))*[V(1,uind),V(2,uind)];
    [tu1,wu1,te,ye,ie,stats] = dop853('fhn_el3',[0 100],xu,options,p);
    sdata(j,:) = ws1(end,:);
    udata(j,:) = wu1(end,:);
```

```
    end
    plot(p4,udata(:,2),'bx',p4,sdata(:,2),'rx');
```



Function that measures the difference between the ends of the stable and unstable manifolds.

```
function [sud] = hom_diff( p)
%Compute difference of values for intersections of
%stable and unstable manifolds of saddle in the FitzHugh-Nagumo model
    vinc = 1e-7;
    options = dopset('AbsTol',1e-14,'RelTol',1e-12,'EventTol',1e-14,'Events',1);
eqm = fhn_eq(p);
    jac = fhn2_jac(eqm(1,:),p);
    if jac(1,1)*jac(2,2) - jac(1,2)*jac(2,1) < 0
        saddle = eqm(1,:);
    else
        jac = fhn2_jac(eqm(2,:),p);
        if jac(1,1)*jac(2,2) - jac(1,2)*jac(2,1) < 0
        saddle = eqm(2,:);
        else
        jac = fhn2_jac(eqm(3,:),p);
        saddle = eqm(3,:);
        end
    end
    [V,D] = eig(jac);
    if D(1,1) < 0
        sind = 1;
    else
        sind = 2;
    end
    uind = 3-sind;
    p(6) = -1;
    xs = saddle+vinc*sign(V(2,sind))*[V(1,sind),V(2,sind)];
    [ts1,ws1,te,ye,ie,stats] = dop853('fhn_el3',[0 -2000],xs,options,p);
    p(6) = 1;
    xu = saddle-vinc*sign(V(2,uind))*[V(1,uind),V(2,uind)];
```

```
    [tu1,wu1,te,ye,ie,stats] = dop853('fhn_el3',[0 100],xu,options,p);
    sud = ws1(end,:)-wu1(end,:);
end
```

Next define function that iterates secant method on hom_diff to locate parameters with a homoclinic orbit

```
function [ pout, sud ] = fhn2d_hom_e( p)
%Secant method iteration to homoclinic orbit in the FitzHugh-Nagumo model
%as the parameter $e$ varies
nit = 10;
pinc = 1e-3;
pd = zeros(nit,3);
sud = hom_diff(p);
pd(1,:) = [p(4), sud];
p(4) = p(4) + pinc;
sud = hom_diff(p);
pd(2,:) = [p(4), sud];
for j= 3:nit
    p(4) = pd(j-1,1) - pd(j-1,2)*(pd(j-1,1) - pd(j-2,1))/(pd(j-1,2) - pd(j-2,2));
    sud = hom_diff(p);
    pd(j,:) = [p(4),sud];
    if max(abs(sud)) < 1e-13
        pout = p;
        iter = j;
        break
    end
end
end
```

Continue fhn2d_hom_e to find parameter curve with homoclinic orbits with a and e active parameters.

```
In [13]: p = [1, 0.05, 1.2, 0.388,1,-1];
         na = 10;
         ainc = -1e-3;
         pdata = zeros(na,6);
         resid = zeros(na,1);
         [pout,sud] = fhn2d_hom_e(p);
         pdata(1,:) = pout;
         resid(1) = max(abs(sud));
         p(1) = p(1) + ainc;
         [pout,sud] = fhn2d_hom_e(p);
         pdata(2,:) = pout;
         resid(2) = max(abs(sud));
         for k = 3:na;
             p = 2*pdata(k-1,:)-pdata(k-2,:);
             [pout,sud] = fhn2d_hom_e(p);
             resid(k) = max(abs(sud));
             pdata(k,:) = pout;
         end
         format long
         ae = [pdata(:,1),pdata(:,4),resid]

Out[13]: ae =
```

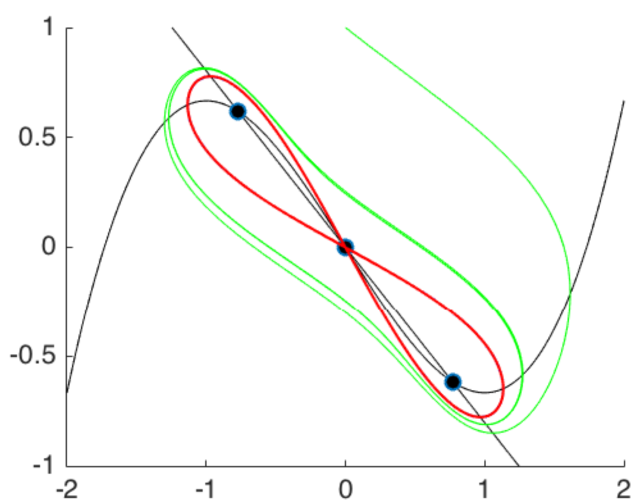
```
1.000000000000000   0.388544920636919   0.000000000000015
0.999000000000000   0.386675161501377   0.000000000000085
0.998000000000000   0.384787727350473   0.000000000000026
0.997000000000000   0.382883608019431   0.000000000000069
0.996000000000000   0.380963607397934   0.000000000000003
0.995000000000000   0.379028385925058   0.000000000000025
0.994000000000000   0.377078491308955   0.000000000000027
0.993000000000000   0.375114381233732   0.000000000000004
0.992000000000000   0.373136440483494   0.000000000000077
0.991000000000000   0.371144994074477   0.000000000000036
```

1.1.7 Double homoclinic orbit and perturbations

In the parameter space of a generic, two parameter family of planar vector fields, two curves of "small" homoclinic orbits cross tranversally at a double homoclinic bifurcation. In addition, there are two curves of "large homoclinic" orbits that terminate at this point.

Display phase portrait of a double homoclinic orbit.

```
In [14]: p = [0.4015410733057, 0.0, 0.5, 1];
         figure(1)
         clf
         hold on
         eqm = fhn_eq(p)
         fhn_nullcline(p);
         [ws1,ws2,wu1,wu2] = fhn2_smflds(eqm(1,:),p,1e-5);
         options = dopset('AbsTol',1e-14,'RelTol',1e-12,'EventTol',1e-14,'Events',1);
         [ts,pts,te,ye,ie,stats] = dop853('fhn2d',[0 2000],[0,1],options,p);
         plot(pts(:,1),pts(:,2),'g')
         [ts,pts,te,ye,ie,stats] = dop853('fhn2d',[0 -500],eqm(2,:)-[0,1e-5],options,p);
         %plot(pts(:,1),pts(:,2),'g')
         [ts,pts,te,ye,ie,stats] = dop853('fhn2d',[0 -500],eqm(3,:)-[0,1e-5],options,p);
         %plot(pts(:,1),pts(:,2),'g')
         axis([-2,2,-1,1])
```

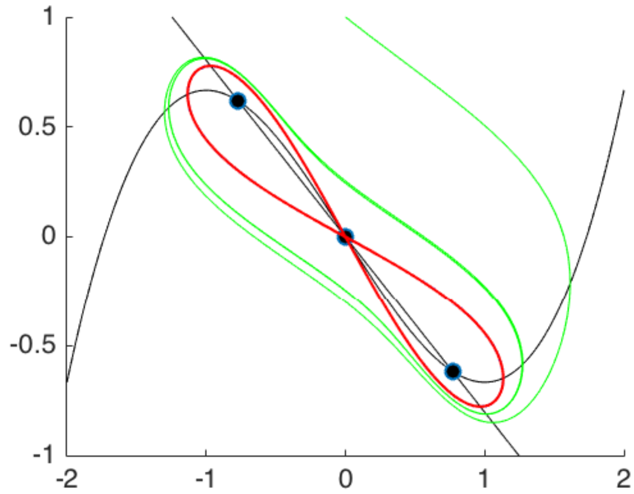


Out[14]: eqm =

```
                    0                    0
    0.768604944146081   -0.617252908440970
   -0.768604944146081    0.617252908440970


eqm =

                    0                    0
    0.768604944146081   -0.617252908440970
   -0.768604944146081    0.617252908440970


V =

    0.944210187498426   -0.655789982917242
   -0.329343470899588    0.754943374237699


D =

    0.651196867752354                    0
                    0   -0.151196867752354
```

Investigate parameter values near the double homoclinic point

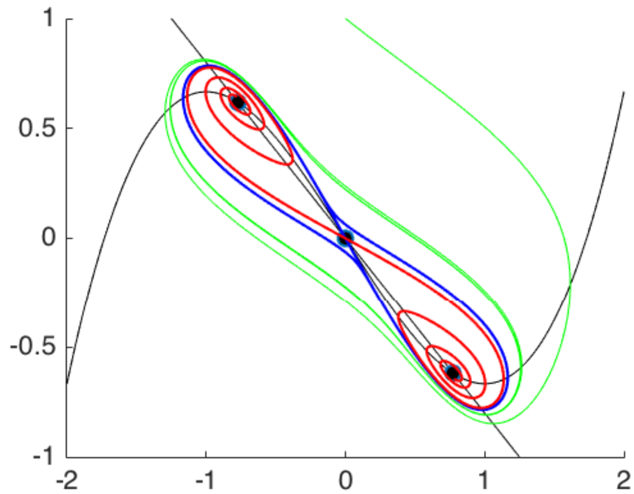
In [15]:
```
p = [0.4015410733058, 0.0, 0.5, 1];
figure(1)
clf
hold on
eqm = fhn_eq(p)
fhn_nullcline(p);
[ws1,ws2,wu1,wu2] = fhn2_smflds(eqm(1,:),p,1e-5);
options = dopset('AbsTol',1e-14,'RelTol',1e-12,'EventTol',1e-14,'Events',1);
[ts,pts,te,ye,ie,stats] = dop853('fhn2d',[0 2000],[0,1],options,p);
plot(pts(:,1),pts(:,2),'g')
[ts,pts,te,ye,ie,stats] = dop853('fhn2d',[0 -500],eqm(2,:)-[0,1e-5],options,p);
%plot(pts(:,1),pts(:,2),'g')
[ts,pts,te,ye,ie,stats] = dop853('fhn2d',[0 -500],eqm(3,:)-[0,1e-5],options,p);
%plot(pts(:,1),pts(:,2),'g')
axis([-2,2,-1,1])
```



Out[15]:
```
eqm =

                   0                   0
   0.768604944145690  -0.617252908440810
  -0.768604944145690   0.617252908440810


eqm =

                   0                   0
   0.768604944145690  -0.617252908440810
  -0.768604944145690   0.617252908440810


V =

   0.944210187498389  -0.655789982917283
  -0.329343470899693   0.754943374237664


D =

   0.651196867752230                   0
                   0  -0.151196867752230
```

In [16]:
```
p = [0.4, 0.0, 0.5, 1];
figure(1)
clf
hold on
eqm = fhn_eq(p)
fhn_nullcline(p);
[ws1,ws2,wu1,wu2] = fhn2_smflds(eqm(1,:),p,1e-5);
options = dopset('AbsTol',1e-14,'RelTol',1e-12,'EventTol',1e-14,'Events',1);
[ts,pts,te,ye,ie,stats] = dop853('fhn2d',[0 2000],[0,1],options,p);
```

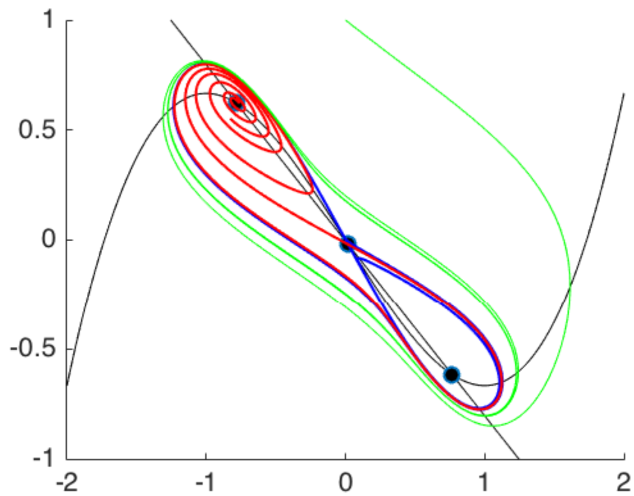
```
plot(pts(:,1),pts(:,2),'g')
[ts,pts,te,ye,ie,stats] = dop853('fhn2d',[0 -500],eqm(2,:)-[0,1e-5],options,p);
%plot(pts(:,1),pts(:,2),'g')
axis([-2,2,-1,1])
```



Out[16]:
```
eqm =

                      0                     0
    0.774596669241483  -0.619677335393187
   -0.774596669241483   0.619677335393187


eqm =

                      0                     0
    0.774596669241483  -0.619677335393187
   -0.774596669241483   0.619677335393187


V =

    0.944771725580764  -0.655168273806749
   -0.327729135938728   0.755482979951954


D =

    0.653112887414927                     0
                    0  -0.153112887414927
```

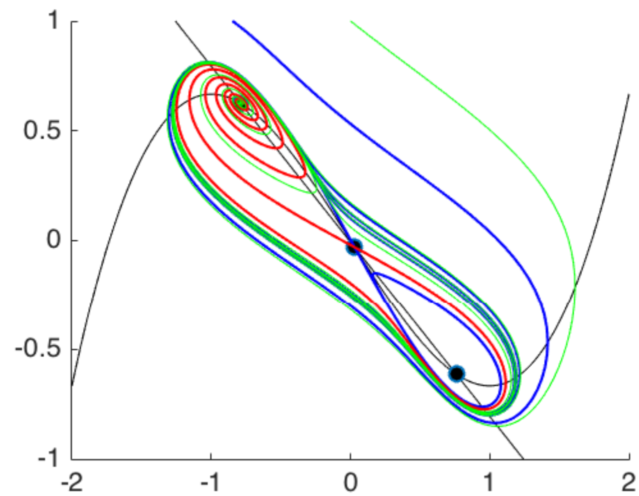
In [17]:
```
p = [0.4, 0.002, 0.5, 1];
figure(1)
clf
hold on
```

```
eqm = fhn_eq(p)
fhn_nullcline(p);
[ws1,ws2,wu1,wu2] = fhn2_smflds(eqm(3,:),p,1e-5);
options = dopset('AbsTol',1e-14,'RelTol',1e-12,'EventTol',1e-14,'Events',1);
[ts,pts,te,ye,ie,stats] = dop853('fhn2d',[0 2000],[0,1],options,p);
plot(pts(:,1),pts(:,2),'g')
[ts,pts,te,ye,ie,stats] = dop853('fhn2d',[0 -500],eqm(2,:)-[0,1e-5],options,p);
%plot(pts(:,1),pts(:,2),'g')
axis([-2,2,-1,1])
```



Out[17]:
```
eqm =

  -0.784409417029790   0.623527533623832
   0.764396056958460  -0.615516845566768
   0.020013360071329  -0.020010688057063


eqm =

  -0.784409417029790   0.623527533623832
   0.764396056958460  -0.615516845566768
   0.020013360071329  -0.020010688057063


V =

   0.944721268484968  -0.655354120706123
  -0.327874556609923   0.755321770157265


D =

   0.652539896054106                   0
                   0  -0.152940430635451
```

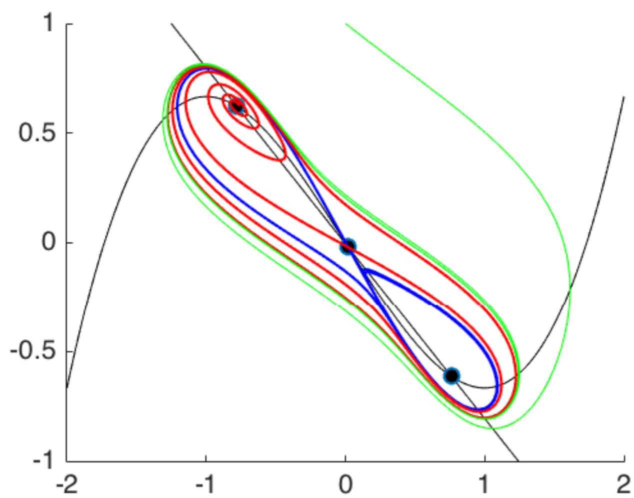
In [18]:
```
p = [0.4, 0.003, 0.5, 1];
figure(1)
clf
hold on
eqm = fhn_eq(p)
fhn_nullcline(p);
[ws1,ws2,wu1,wu2] = fhn2_smflds(eqm(3,:),p,1e-5);
options = dopset('AbsTol',1e-14,'RelTol',1e-12,'EventTol',1e-14,'Events',1);
[ts,pts,te,ye,ie,stats] = dop853('fhn2d',[0 2000],[0,1],options,p);
plot(pts(:,1),pts(:,2),'g')
[ts,pts,te,ye,ie,stats] = dop853('fhn2d',[0 -500],eqm(2,:)-[0,1e-5],options,p);
%plot(pts(:,1),pts(:,2),'g')
axis([-2,2,-1,1])
```



Out[18]:
```
eqm =

  -0.789182123084507   0.625345698467606
   0.759136919361091  -0.613309535488873
   0.030045203723416  -0.030036162978733


eqm =

  -0.789182123084507   0.625345698467606
   0.759136919361091  -0.613309535488873
   0.030045203723416  -0.030036162978733


V =

   0.944657881104889  -0.655587336179838
  -0.328057140855708   0.755119357877034
```

```
D =

   0.651821147548666                   0
                   0  -0.152723861815448
```

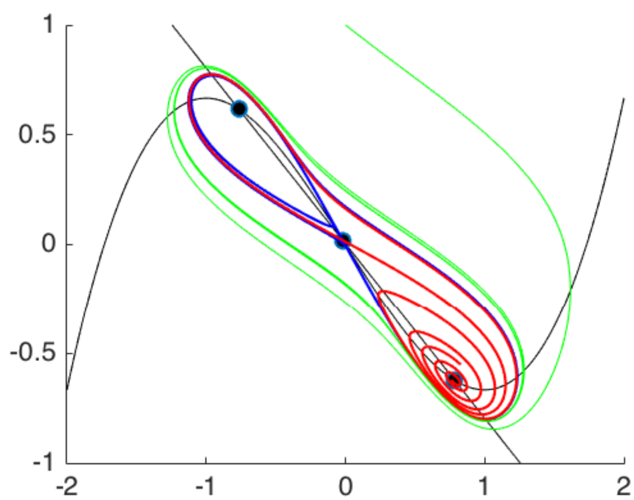
In [19]:
```
p = [0.401, 0.002, 0.5, 1];
figure(1)
clf
hold on
eqm = fhn_eq(p)
fhn_nullcline(p);
[ws1,ws2,wu1,wu2] = fhn2_smflds(eqm(3,:),p,1e-5);
options = dopset('AbsTol',1e-14,'RelTol',1e-12,'EventTol',1e-14,'Events',1);
[ts,pts,te,ye,ie,stats] = dop853('fhn2d',[0 2000],[0,1],options,p);
plot(pts(:,1),pts(:,2),'g')
[ts,pts,te,ye,ie,stats] = dop853('fhn2d',[0 -500],eqm(2,:)-[0,1e-5],options,p);
%plot(pts(:,1),pts(:,2),'g')
axis([-2,2,-1,1])
```



Out[19]:
```
eqm =

  -0.780623043704398   0.622059681050927
   0.760407114541793  -0.613846505862518
   0.020215929162605  -0.020213175188409


eqm =

  -0.780623043704398   0.622059681050927
   0.760407114541793  -0.613846505862518
   0.020215929162605  -0.020213175188409


V =
```

```
    0.944355933313491  -0.655761435480458
   -0.328925327719842   0.754968171340096


D =

    0.651284797324125                   0
                    0  -0.151693481116033
```

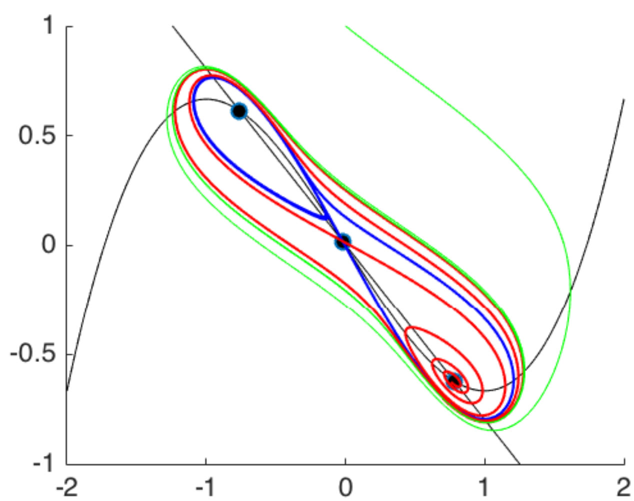
In [20]:
```
p = [0.4, -0.002, 0.5, 1];
figure(1)
clf
hold on
eqm = fhn_eq(p)
fhn_nullcline(p);
[ws1,ws2,wu1,wu2] = fhn2_smflds(eqm(3,:),p,1e-5);
options = dopset('AbsTol',1e-14,'RelTol',1e-12,'EventTol',1e-14,'Events',1);
[ts,pts,te,ye,ie,stats] = dop853('fhn2d',[0 2000],[0,1],options,p);
plot(pts(:,1),pts(:,2),'g')
[ts,pts,te,ye,ie,stats] = dop853('fhn2d',[0 -500],eqm(2,:)-[0,1e-5],options,p);
%plot(pts(:,1),pts(:,2),'g')
axis([-2,2,-1,1])
```



Out[20]:
```
eqm =

    0.784409417029790  -0.623527533623832
   -0.764396056958460   0.615516845566768
   -0.020013360071329   0.020010688057063


eqm =

    0.784409417029790  -0.623527533623832
```

```
  -0.764396056958460   0.615516845566768
  -0.020013360071329   0.020010688057063


V =

   0.944721268484968  -0.655354120706123
  -0.327874556609923   0.755321770157265


D =

   0.652539896054106                   0
                   0  -0.152940430635451
```

In [21]:
```
p = [0.401, -0.002, 0.5, 1];
figure(1)
clf
hold on
eqm = fhn_eq(p)
fhn_nullcline(p);
[ws1,ws2,wu1,wu2] = fhn2_smflds(eqm(3,:),p,1e-5);
options = dopset('AbsTol',1e-14,'RelTol',1e-12,'EventTol',1e-14,'Events',1);
[ts,pts,te,ye,ie,stats] = dop853('fhn2d',[0 2000],[0,1],options,p);
plot(pts(:,1),pts(:,2),'g')
[ts,pts,te,ye,ie,stats] = dop853('fhn2d',[0 -500],eqm(2,:)-[0,1e-5],options,p);
%plot(pts(:,1),pts(:,2),'g')
axis([-2,2,-1,1])
```



Out[21]:
```
eqm =

   0.780623043704398  -0.622059681050927
  -0.760407114541793   0.613846505862518
  -0.020215929162605   0.020213175188409
```

```
eqm =

   0.780623043704398  -0.622059681050927
  -0.760407114541793   0.613846505862518
  -0.020215929162605   0.020213175188409


V =

   0.944355933313491  -0.655761435480458
  -0.328925327719842   0.754968171340096


D =

   0.651284797324125                   0
                   0  -0.151693481116033
```

```
In [ ]:
```