

fhn_snlc

December 18, 2015

0.1 Saddle-nodes of periodic orbits in the FitzHugh-Nagumo model

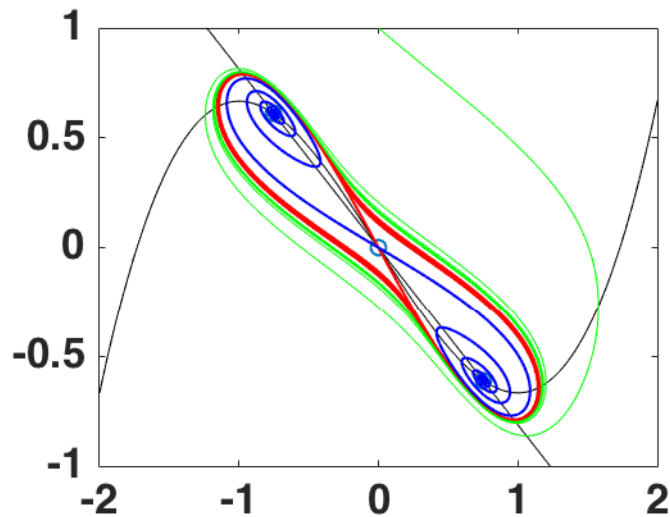
The version of the FitzHugh-Nagumo model used in this notebook is given by the equations

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

Newton's method is used on a return map and its derivative to locate a saddle-node of periodic orbits.

Plot phase portraits for parameters that straddle the bifurcations. Active parameter is e .

```
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])
%
```



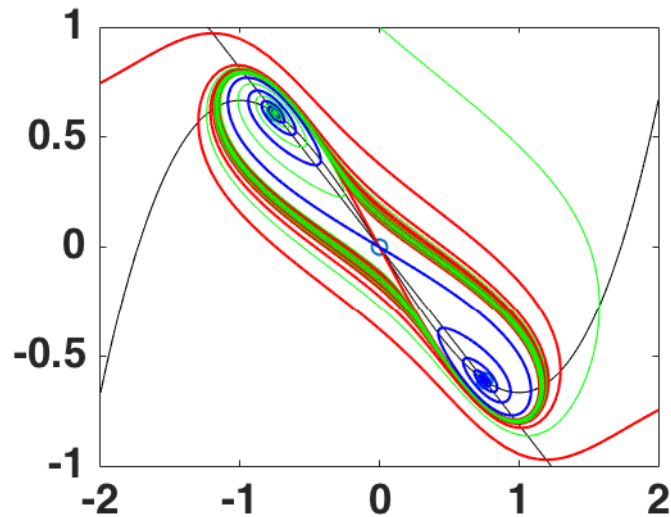
Out[1]: eqm =

```
      0      0
0.7500 -0.6094
-0.7500  0.6094
```

```

In [2]: 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[2]: eqm =

```

```

      0      0
0.7500 -0.6094
-0.7500  0.6094

```

```

Exit of dop853 at x = -2.8247865932943455e+02, step size too small h = -6.0144644841320101e-13
Exit of dop853 at x = -2.8247865932943455e+02, step size too small h = -6.0144644841320101e-13

```

Use bisection to get close to bifurcation. Then use return map to examine details.

```

In [3]: %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])

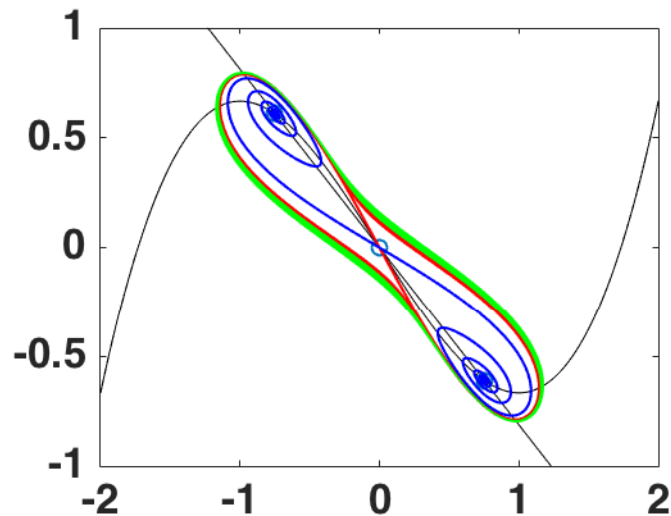
%Plot 100 trajectories to first return with increasing coordinate for
%cross-section that determines linear nullcline
xn = 100;

```

```

%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
end

```



Out[3]: p =

```

0.8125
0
1.0000
0.5444

```

eqm =

```

0 0
0.7500 -0.6094
-0.7500 0.6094

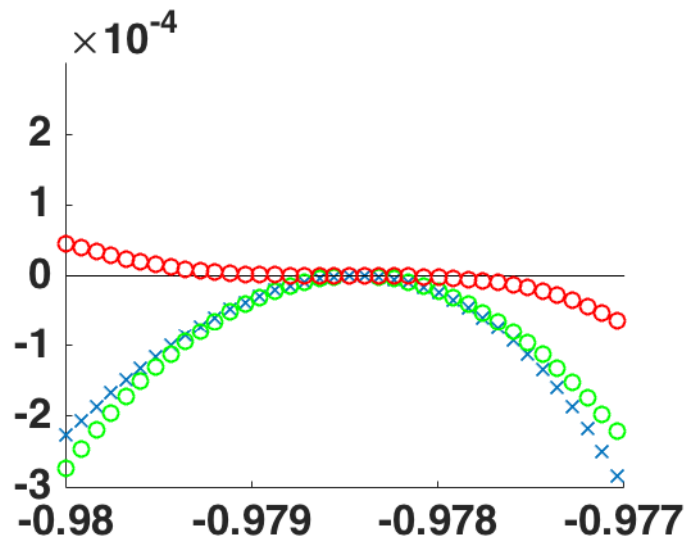
```

Analyze the return data

```

In [4]: %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[4]: resid =

```

```

1.0e-10 *

```

```

0.3646    0.2962

```

```

ind =

```

69 69

```
y0 =  
  
3.6458e-11
```

```
dx0 =  
  
0.0085
```

```
dx0 =  
  
-225.5491
```

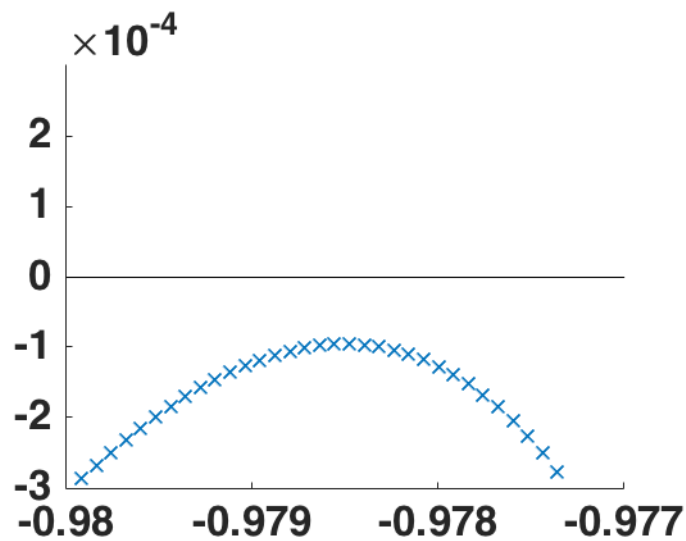
Repeat for parameters without a periodic orbit

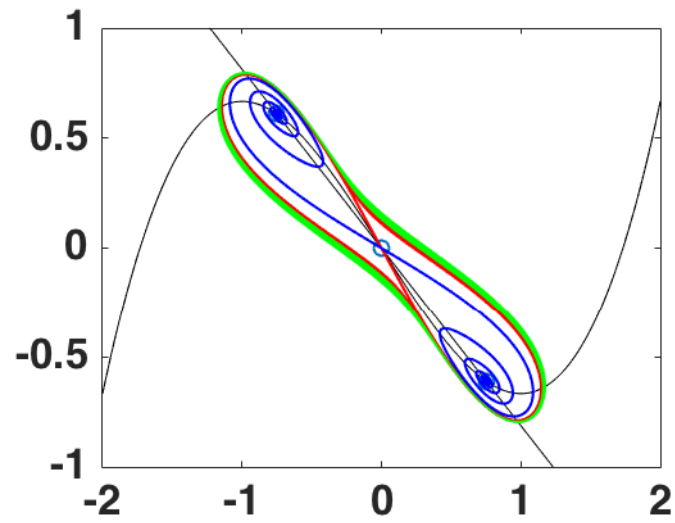
```
In [5]: %The parameters  
p = [13/16; 0; 1; 0.5444]  
axis([-2,2,-1,1])  
%Equilibrium points and nullclines  
eqm = fhn_eq(p)  
fhn_nullcline(p);  
%Stable and unstable manifolds  
[ws1,ws2,wu1,wu2] = fhn2_smflds(eqm(1,:),p,1e-5);  
axis([-2,2,-1,1])  
  
%Plot 100 trajectories to first return with increasing coordinate for  
%cross-section that determines linear nullcline  
xn = 100;  
%x coordinate increment between initial 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_e12',[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[5]: p =

```
0.8125
0
1.0000
0.5444
```

eqm =

```
0      0
0.7500 -0.6094
-0.7500 0.6094
```

resid =

```
1.0e-04 *
0.9654 0.7844
```

ind =

```
69 69
```

y0 =

```
-9.6541e-05
```

dx0 =

-0.0057

dx0 =

-231.3807

0.1.1 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 x_{inc} , 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 [6]: *% 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[6]: p =

```

0.8125000000000000
0
1.0000000000000000
0.5443639468000000

```

z =

```

-0.9784800000000000    0.7950150000000000

```

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

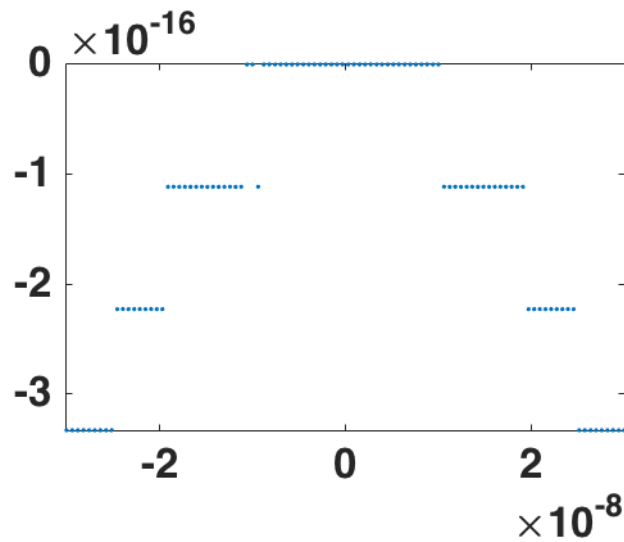
```

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 $1e-8$.

```

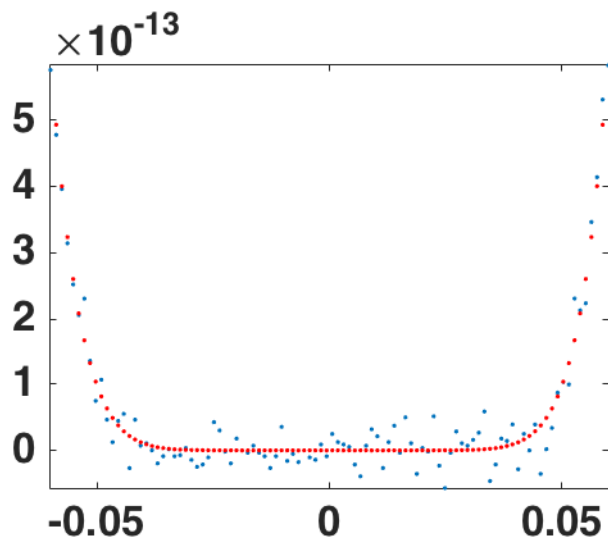
In [7]: xinc = 3e-8;
        x = linspace(pi/3-xinc,pi/3+xinc);
        y = sin(x) - x/2 - sqrt(3)/2+pi/6;
        figure(1)
        plot(x-pi/3,y,'.')
        axis([-xinc,xinc,min(y),max(y)])

```



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 [8]: 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.^4 - 120*u.^3 + 45*u.^2 - 10*u + 1;
        w = (1-u).^10;
        figure(2)
        plot(u-1,v,'b.',u-1,w,'r.')
        axis([-uinc,uinc,min(v),max(v)])
        min(v)
```



```
Out[8]: 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 it as a challenge to implement such an algorithm and embed it into a continuation scheme.

In []: