Reducing a Neuron Model to Normal Form at a Bogdanov-Takens Bifurcation

Joseph P. McKenna

Biomathematics Journal Club, Spring 2016

February 29, 2016

Outline

Outline

Introduction

Goal

Transform the system

$$\begin{aligned}\dot{x}_1 &= -[g_1 n_1(x_1)(x_1 - V_1) + g_2 x_2 (x_1 - V_2) + g_3(x_1 - V_3) - I]/C \\ \dot{x}_2 &= [n_2(x_1) - x_2]/\tau \\ n_i(x) &= [1 + e^{h_i(v_i - x)}]^{-1}, \qquad i \in \{1,2\}\end{aligned}$$

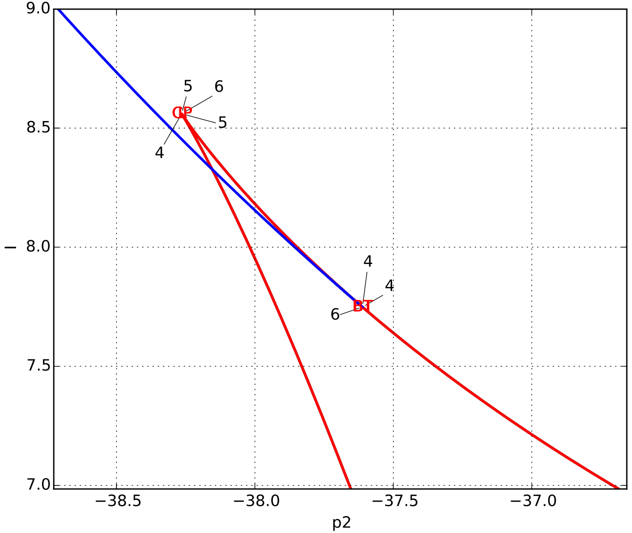
into a ‘simpler’ form while preserving its qualitative behavior near a Bogdanov-Takens (BT) bifurcation.

Where is the BT Bifurcation?

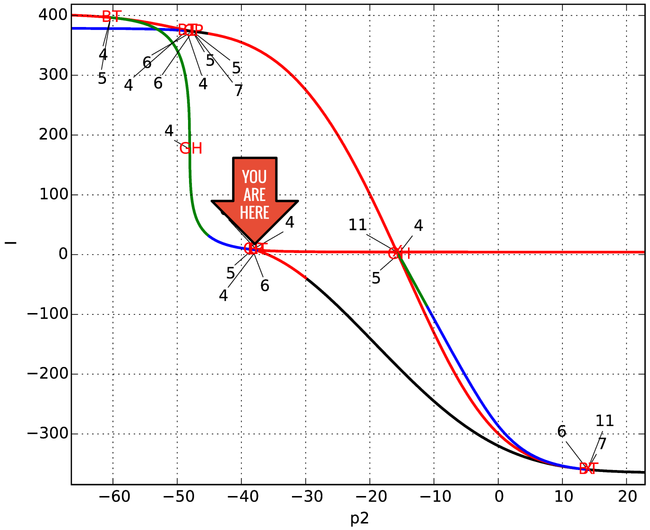
```
import auto
x1 = -100.; p2 = -45.
x2 = 1./(1.+np.exp(h2*(p2-x1)))
I=g1*(x1-V1)*n(1,x1)+g2*x2*(x1-V2)+g3*(x1-V3)
ss = auto.run(e='bif',c='bif',U={'x1':x1,'x2':x2},PAR={'I':I,'p2':p2},STOP=['
    HB2'])
hb = auto.run(ss('HB1'),ICP=['I','p2'],ISW=2,STOP=['BT1'])
```

```
Starting bif ...
  BR    PT  TY  LAB       I              x1             x2
   1     1  EP    1   -1.75377E+02   -1.00000E+02   1.67014E-05
   1   224  HB    2    3.06590E+01   -5.64815E+01   9.14301E-02
   1   567  HB    3    3.69550E+02   -2.40425E+01   9.85102E-01
Starting bif ...
  BR    PT  TY  LAB       I              x1             x2             p2
   2   259  BT    4    7.74871E+00   -5.82119E+01   1.59847E-02   -3.76118E+01
```

Where is the BT Bifurcation?



Where is the BT Bifurcation?



Where is the BT bifurcation?

- Algebraically, at the BT bifurcation, $\mathrm{tr}\, Df, \det Df = 0$

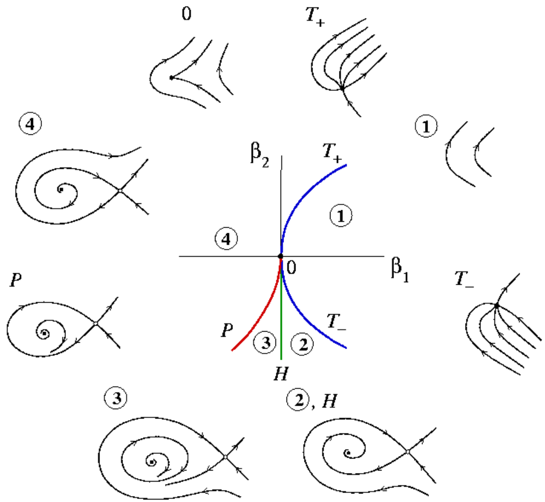
```
DF[0,0] = -(g1*n(1,x0[0]) + g1*dn(1,1,x0[0])*(x0[0]-V1) + g2*x0[1]+g3)/C
DF[0,1] = -g2*(x0[0]-V2)/C
DF[1,0] = dn(2,1,x0[0])/tau
DF[1,1] = -1./tau
tr = DF[0,0]+DF[1,1]
det = DF[0,0]*DF[1,1]-DF[0,1]*DF[1,0]
print tr, det
```

```
-2.30868124618e-08 2.42774838011e-08
```

- Then, at the BT, $Df(0)$ has the form

$$Df(0) = \frac{1}{\tau}\begin{bmatrix} 1 & -\left(\frac{\partial n_2(a)}{\partial x_1}\right)^{-1} \\ \frac{\partial n_2(a)}{\partial x_1} & -1 \end{bmatrix}$$

What is a BT bifurcation?



Outline

Preliminaries

- When injected current is

$$I = g_1 n_1(a)(a - V_1) + g_2 n_2(a)(a - V_2) + g_3(a - V_3),$$

  the system has a steady state at $x_0 = (a, b) = (a, n_2(a))$.
    - Translate by $x \to x + x_0$ so that the steady state is the origin.

- Expand the righthand side of $\dot{x} = f(x)$ in a Taylor series.

- Calculate the Jordan form $J = T^{-1} Df(0) T$ of the linear part $Df(0)x$ at the origin.
    - Transform by $x \to Tx$ so that the linear term of the Taylor series is Jx.

Taylor Series Terms

Multi-index	Derivative at the origin
$(0,0)$	$f_1(0)$ $f_2(0)$
$(1,0)$	$\frac{\partial f_1}{\partial x_1} = -[g_1 n_1(a) + g_1 \frac{\partial n_1}{\partial x_1}(a)(a - V_1) + g_2 b + g_3]/C$ $\frac{\partial f_2}{\partial x_1} = \frac{1}{\tau}\frac{\partial n_2}{\partial x_1}(a)$
$(0,1)$	$\frac{\partial f_1}{\partial x_2} = -g_2(a - V_2)/C$ $\frac{\partial f_2}{\partial x_2} = -\frac{1}{\tau}$
$(1,1)$	$\frac{\partial^2 f_1}{\partial x_2 \partial x_1} = -g_2/C$
$(m,0)$ $m \geq 2$	$\frac{\partial^m f_1}{\partial x_1^m} = -g_1[m\frac{\partial^{m-1} n_1}{\partial x_1^{m-1}}(a) + \frac{\partial^m n_1}{\partial x_1^m}(a)(a - V_1)]/C$ $\frac{\partial^m f_2}{\partial x_1^m} = \frac{1}{\tau}\frac{\partial^m n_2}{\partial x_1^m}(a)$

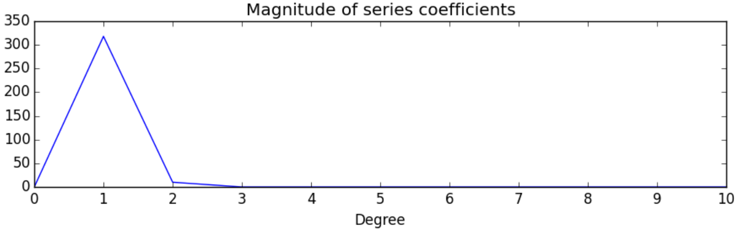
Taylor Series Terms

```
def taylor(k):
    F=[]
    for i in range(k+1):
        F.append(np.zeros([2*i+2]))
        if i==0:
            F[0][0]=-(g1*n(1,x0[0])*(x0[0]-V1)+g2*x0[1]*(x0[0]-V2)+g3*(x0[0]-V3
)-I)/C
            F[0][1]=(n(2,x0[0])-x0[1])/tau
        if i==1:
            F[1][0]=-(g1*n(1,x0[0])+g1*dn(1,1,x0[0])*(x0[0]-V1)+g2*x0[1]+g3)/C
            F[1][1]=-g2*(x0[0]-V2)/C
            F[1][2]=dn(2,1,x0[0])/tau
            F[1][3]=-1./tau
        if i==2:
            F[2][0]=-.5*g1*(2.*dn(1,1,x0[0])+dn(1,2,x0[0])*(x0[0]-V1))/C
            F[2][1]=-g2/C
            F[2][3]=.5*dn(2,2,x0[0])/tau
        if i>=3:
            F[i][0]=-g1*(i*dn(1,i-1,x0[0])+dn(1,i,x0[0])*(x0[0]-V1))/C/np.math.
factorial(i)
            F[i][i+1]=dn(2,i,x0[0])/tau/np.math.factorial(i)
    return F
```

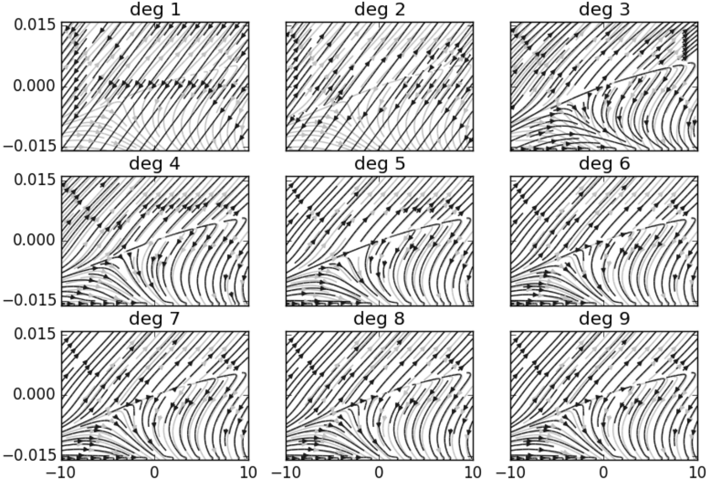
Taylor Series Vector Field

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1(0)}{\partial x_1} & \frac{\partial f_1(0)}{\partial x_2} \\ \frac{\partial f_2(0)}{\partial x_1} & \frac{\partial f_2(0)}{\partial x_2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} \frac{1}{2}\frac{\partial^2 f_1(0)}{\partial x_1^2} & \frac{\partial^2 f_1(0)}{\partial x_1 \partial x_2} & 0 \\ \frac{1}{2}\frac{\partial^2 f_2(0)}{\partial x_1^2} & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1^2 \\ x_1 x_2 \\ x_2^2 \end{bmatrix}$$

$$+ \sum_{m \geq 3} \frac{1}{m!} \begin{bmatrix} \frac{\partial^m f_1(0)}{\partial x_1^m} & 0 & \cdots & 0 \\ \frac{\partial^m f_2(0)}{\partial x_1^m} & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} x_1^m \\ x_1^{m-1} x_2 \\ \vdots \\ x_2^m \end{bmatrix}$$



Taylor Series Vector Field



Jordan Form

- The Jordan decomposition of the linear part of the Taylor series at the origin is $J = T^{-1}Df(0)T$ for

$$T = \begin{bmatrix} 1 & 1 \\ \frac{\partial n_2(a)}{\partial x_1} & 0 \end{bmatrix} \quad \text{and} \quad J = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$

```
T=np.array([[1.,1.],[DF[1,0],0.]])
Tinv=np.linalg.inv(T)
J=np.array([[0.,1.],[0.,0.]])
print np.linalg.norm(J-np.dot(Tinv,np.dot(A1,T)))
```

```
3.3502195888571857e-08
```

Jordan Form

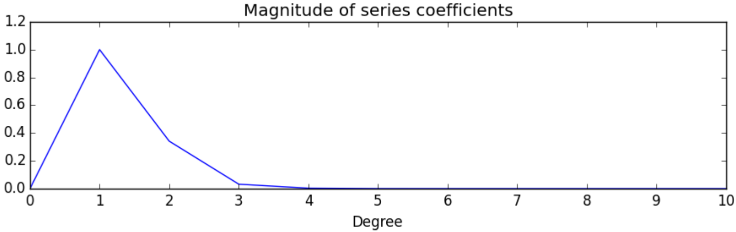
```
def jordan(F,inverse=False):
    k = len(F)-1
    T = np.array([[1.,1.],[DF[1,0],0.]]) if inverse else np.array([[0.,-DF
     [0,1]],[1.,DF[0,1]]])
    powt = np.ones([2,2,k+1])
    for i in range(1,k+1):
        powt[:,:,i] = T*powt[:,:,i-1]
    for k in range (1,k+1):
        TT = np.zeros([k+1,k+1])
        for i in range(k+1):
            c1 = choose(k-i); c2 = choose(i)
            tmp1 = [c1[j]*powt[0,0,k-i-j]*powt[0,1,j] for j in range(k-i+1)]
            tmp2 = [c2[j]*powt[1,0,i-j]*powt[1,1,j] for j in range(i+1)]
            TT[:,i] = np.convolve(tmp1,tmp2)
        F[k][:k+1] = np.dot(TT,F[k][:k+1]); F[k][k+1:] = np.dot(TT,F[k][k+1:])
        tmp = Tinv[0,0]*F[k][:k+1] + Tinv[0,1]*F[k][k+1:]
        F[k][k+1:] = Tinv[1,0]*F[k][:k+1]+Tinv[1,1]*F[k][k+1:]
        F[k][:k+1] = tmp
    return F
print jordan(taylor(10))[1]
print taylor(10)[1]-jordan(jordan(taylor(10)),inverse=True)[1]
```

```
[  0.00000000e+00   1.00000000e+00  -2.42774838e-08  -2.30868125e-08]
[  0.00000000e+00   3.78491734e-07   3.74565136e-12   0.00000000e+00]
```

Taylor Series with Linear Part in Jordan Form

- Transform by $x \to Tx$ so that linear part of the Taylor series is in Jordan form

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ 0 \end{bmatrix} + \sum_{m \geq 2} \frac{1}{m!} T^{-1} f^{(m)}(0)(Tx)^m$$



The Normalization Machine

Main Ideas

- We'll perform a series of smooth, (locally) invertible transformations $x \to \psi(x)$ that annihilate terms in the Taylor series.
- All transformations are linear and can be cast as numerical linear algebra.
- The extent to which the system can be 'simplified' depends on the linear part of the Taylor series.

The Normalization Machine

- How does $x = \psi(\xi)$, $\psi \in \mathsf{Diff}(\mathbb{R}^n)$, transform the o.d.e. $\dot{x} = f(x)$?

$$
\begin{aligned}
\psi'(\xi)\,\dot{\xi} &= \dot{x} && \text{differentiate } x = \psi(\xi) \\
\dot{\xi} &= \psi'(\xi)^{-1}\,\dot{x} && \text{left-multiply by } \psi'(\xi)^{-1} \\
\dot{\xi} &= \psi'(\xi)^{-1}\,f(\psi(\xi)) && \text{substitute } \dot{x} = f(x) = f(\psi(\xi)) \\
\dot{x} &= (\psi')^{-1}\,f \circ \psi(x) && \text{relabel } \xi \leftarrow x \\
&\stackrel{\text{def}}{=} \mathsf{S}_\psi f(x)
\end{aligned}
$$

- By a 'similarity' transformation.

The Normalization Machine

- How does $x = \psi(\xi)$, $\psi \in \mathrm{Diff}(\mathbb{R}^n)$, transform the o.d.e. $\dot{x} = f(x)$?

$$\psi'(\xi)\,\dot{\xi} = \dot{x} \qquad \text{differentiate } x = \psi(\xi)$$

$$\dot{\xi} = \psi'(\xi)^{-1}\,\dot{x} \qquad \text{left-multiply by } \psi'(\xi)^{-1}$$

$$\dot{\xi} = \psi'(\xi)^{-1}\,f(\psi(\xi)) \qquad \text{substitute } \dot{x} = f(x) = f(\psi(\xi))$$

$$\dot{x} = (\psi')^{-1}\,f \circ \psi(x) \qquad \text{relabel } \xi \leftarrow x$$

$$\stackrel{\text{def}}{=} S_\psi f(x)$$

- By a 'similarity' transformation.

The Normalization Machine

- How does $x = \psi(\xi)$, $\psi \in \mathrm{Diff}(\mathbb{R}^n)$, transform the o.d.e. $\dot{x} = f(x)$?

$$
\begin{aligned}
\psi'(\xi)\,\dot{\xi} &= \dot{x} && \text{differentiate } x = \psi(\xi) \\
\dot{\xi} &= \psi'(\xi)^{-1}\,\dot{x} && \text{left-multiply by } \psi'(\xi)^{-1} \\
\dot{\xi} &= \psi'(\xi)^{-1}\,f(\psi(\xi)) && \text{substitute } \dot{x} = f(x) = f(\psi(\xi)) \\
\dot{x} &= (\psi')^{-1}\,f \circ \psi(x) && \text{relabel } \xi \leftarrow x \\
&\stackrel{\text{def}}{=} \mathrm{S}_\psi f(x)
\end{aligned}
$$

- By a ‘similarity’ transformation.

The Normalization Machine

- How does $x = \psi(\xi)$, $\psi \in \mathrm{Diff}(\mathbb{R}^n)$, transform the o.d.e. $\dot{x} = f(x)$?

$$
\begin{aligned}
\psi'(\xi)\,\dot{\xi} &= \dot{x} && \text{differentiate } x = \psi(\xi) \\
\dot{\xi} &= \psi'(\xi)^{-1}\,\dot{x} && \text{left-multiply by } \psi'(\xi)^{-1} \\
\dot{\xi} &= \psi'(\xi)^{-1}\,f(\psi(\xi)) && \text{substitute } \dot{x} = f(x) = f(\psi(\xi)) \\
\dot{x} &= (\psi')^{-1}\,f \circ \psi(x) && \text{relabel } \xi \leftarrow x \\
&\overset{\text{def}}{=} \mathrm{S}_\psi f(x)
\end{aligned}
$$

- By a 'similarity' transformation.

The Normalization Machine

- How does $x = \psi(\xi)$, $\psi \in \mathrm{Diff}(\mathbb{R}^n)$, transform the o.d.e. $\dot{x} = f(x)$?

$$
\begin{aligned}
\psi'(\xi)\,\dot{\xi} &= \dot{x} && \text{differentiate } x = \psi(\xi) \\
\dot{\xi} &= \psi'(\xi)^{-1}\,\dot{x} && \text{left-multiply by } \psi'(\xi)^{-1} \\
\dot{\xi} &= \psi'(\xi)^{-1}\,f(\psi(\xi)) && \text{substitute } \dot{x} = f(x) = f(\psi(\xi)) \\
\dot{x} &= (\psi')^{-1}\,f \circ \psi(x) && \text{relabel } \xi \leftarrow x \\
&\overset{\text{def}}{=} \mathrm{S}_{\psi} f(x)
\end{aligned}
$$

- By a ‘similarity’ transformation.

The Normalization Machine

- How does $x = \psi(\xi)$, $\psi \in \mathrm{Diff}(\mathbb{R}^n)$, transform the o.d.e. $\dot{x} = f(x)$?

$$
\begin{aligned}
\psi'(\xi)\,\dot{\xi} &= \dot{x} && \text{differentiate } x = \psi(\xi) \\
\dot{\xi} &= \psi'(\xi)^{-1}\,\dot{x} && \text{left-multiply by } \psi'(\xi)^{-1} \\
\dot{\xi} &= \psi'(\xi)^{-1}\,f(\psi(\xi)) && \text{substitute } \dot{x} = f(x) = f(\psi(\xi)) \\
\dot{x} &= (\psi')^{-1}\,f \circ \psi(x) && \text{relabel } \xi \leftarrow x \\
&\stackrel{\text{def}}{=} \mathsf{S}_\psi f(x)
\end{aligned}
$$

- By a 'similarity' transformation.

The Normalization Machine

$$
\begin{tikzcd}
\psi \arrow[d, "\mathsf{D}_s"'] & \varphi \arrow[r, "\mathsf{C}_\psi"] \arrow[d, "\mathsf{D}_t"'] & \psi^{-1} \circ \varphi \circ \psi \arrow[d, "\mathsf{D}_t"] \\
g & f \arrow[r, "\mathsf{S}_\psi"] \arrow[dr, "\mathsf{L}_g"'] & (\psi')^{-1} f \circ \psi \arrow[d, "\mathsf{D}_s"] \\
 & & {[f, g]}
\end{tikzcd}
$$

The Normalization Machine

- If f is Lipschitz, $\dot{x} = f(x)$ has a solution, the flow $\varphi^t(x)$.
- Suppose the transformation $\psi(x) = \psi^s(x)$ is smoothly parameterized by $s \in \mathbb{R}$, then it satisfies an o.d.e. $\frac{dx}{ds} = g(x)$, i.e. $\psi^s(x)$ is the flow generated by some vector field $g(x)$
- Suppose $\varphi^0(x) = \psi^0(x) = x$, then

$$\frac{d}{dt}\,\varphi^t(x)\big|_{t=0} = f(x)$$

$$\frac{d}{ds}\,\psi^s(x)\big|_{s=0} = g(x)$$

The Normalization Machine

$$
\begin{tikzcd}
\psi \arrow[d, "\mathsf{D}_s"'] & & \varphi \arrow[r, "\mathrm{C}_\psi"] \arrow[d, "\mathsf{D}_t"'] & \psi^{-1} \circ \varphi \circ \psi \arrow[d, "\mathrm{D}_t"] \\
g & & f \arrow[r, "\mathsf{S}_\psi"] \arrow[dr, "\mathrm{L}_g"'] & (\psi')^{-1}\, f \circ \psi \arrow[d, "\mathrm{D}_s"] \\
& & & [f, g]
\end{tikzcd}
$$

The Normalization Machine

- How can we calculate the righthand side of the new o.d.e. $\dot{x} = \mathsf{S}_\psi f(x)$?
- Using the Lie operator $\mathsf{L}_g f \stackrel{\text{def}}{=} \frac{d}{ds}\, \mathsf{S}_\psi f\big|_{s=0}$.

Theorem

$\mathsf{L}_g f = f'g - g'f$

$$
\begin{aligned}
\frac{d}{ds}\mathsf{S}_\psi f &= \frac{d}{ds}(\psi')^{-1}\ f \circ \psi && \text{def. of } \mathsf{S}_\psi \\
&= (\psi')^{-1}\ f' \circ \psi\ \frac{d}{ds}\psi + \frac{d}{ds}(\psi')^{-1}\ f \circ \psi && \text{prod. \& chain rules} \\
&= (\psi')^{-1}\ f' \circ \psi\ g \circ \psi - \underbrace{(\psi')^{-1}\ g' \circ \psi}_{i.o.u.}\ f \circ \psi && g \text{ generates } \psi
\end{aligned}
$$

$$
\begin{aligned}
\frac{d}{ds}\ \mathsf{S}_\psi f\big|_{s=0} &= f'g - g'f && \text{substitute } s = 0 \\
&\stackrel{\text{def}}{=} [f, g] && \text{'Lie bracket'}
\end{aligned}
$$

The Normalization Machine

- How can we calculate the righthand side of the new o.d.e. $\dot{x} = \mathsf{S}_\psi f(x)$?
- Using the Lie operator $\mathsf{L}_g f \stackrel{\text{def}}{=} \frac{d}{ds}\, \mathsf{S}_\psi f\big|_{s=0}$.

Theorem

$L_g f = f'g - g'f$

$$\begin{aligned}
\frac{d}{ds}\mathsf{S}_\psi f &= \frac{d}{ds}(\psi')^{-1}\ f\circ\psi && \text{def. of } \mathsf{S}_\psi \\
&= (\psi')^{-1}\ f'\circ\psi\ \frac{d}{ds}\psi + \frac{d}{ds}(\psi')^{-1}\ f\circ\psi && \text{prod. \& chain rules} \\
&= (\psi')^{-1}\ f'\circ\psi\ g\circ\psi - \underbrace{(\psi')^{-1}\ g'\circ\psi}_{i.o.u.}\ f\circ\psi && g \text{ generates } \psi \\
\frac{d}{ds}\ \mathsf{S}_\psi f\big|_{s=0} &= f'g - g'f && \text{substitute } s=0 \\
&\stackrel{\text{def}}{=} [f, g] && \text{`Lie bracket'}
\end{aligned}$$

The Normalization Machine

- How can we calculate the righthand side of the new o.d.e. $\dot{x} = \mathsf{S}_\psi f(x)$?
- Using the Lie operator $\mathsf{L}_g f \stackrel{\text{def}}{=} \frac{d}{ds}\, \mathsf{S}_\psi f\big|_{s=0}$.

Theorem

$\mathsf{L}_g f = f' g - g' f$

$$\begin{aligned}
\frac{d}{ds}\mathsf{S}_\psi f &= \frac{d}{ds}(\psi')^{-1}\ f \circ \psi && \text{def. of } \mathsf{S}_\psi \\
&= (\psi')^{-1}\ f' \circ \psi\ \frac{d}{ds}\psi + \frac{d}{ds}(\psi')^{-1}\ f \circ \psi && \text{prod. \& chain rules} \\
&= (\psi')^{-1}\ f' \circ \psi\ g \circ \psi - \underbrace{(\psi')^{-1}\ g' \circ \psi}_{i.o.u.}\ f \circ \psi && g \text{ generates } \psi \\
\frac{d}{ds}\ \mathsf{S}_\psi f\big|_{s=0} &= f' g - g' f && \text{substitute } s = 0 \\
&\stackrel{\text{def}}{=} [f, g] && \text{`Lie bracket'}
\end{aligned}$$

The Normalization Machine

- How can we calculate the righthand side of the new o.d.e. $\dot{x} = \mathsf{S}_\psi f(x)$?
- Using the Lie operator $\mathsf{L}_g f \stackrel{\text{def}}{=} \frac{d}{ds}\, \mathsf{S}_\psi f\big|_{s=0}$.

Theorem

$\mathsf{L}_g f = f'g - g'f$

$$\begin{aligned}
\frac{d}{ds}\mathsf{S}_\psi f &= \frac{d}{ds}(\psi')^{-1}\, f\circ\psi && \text{def. of } \mathsf{S}_\psi \\
&= (\psi')^{-1}\, f'\circ\psi\, \frac{d}{ds}\psi + \frac{d}{ds}(\psi')^{-1}\, f\circ\psi && \text{prod. \& chain rules} \\
&= (\psi')^{-1}\, f'\circ\psi\; g\circ\psi - \underbrace{(\psi')^{-1}\, g'\circ\psi}_{i.o.u.}\; f\circ\psi && g \text{ generates } \psi \\
\frac{d}{ds}\, \mathsf{S}_\psi f\big|_{s=0} &= f'g - g'f && \text{substitute } s=0 \\
&\stackrel{\text{def}}{=} [f,g] && \text{`Lie bracket'}
\end{aligned}$$

The Normalization Machine

- How can we calculate the righthand side of the new o.d.e. $\dot{x} = \mathsf{S}_\psi f(x)$?
- Using the Lie operator $\mathsf{L}_g f \overset{\text{def}}{=} \frac{d}{ds} \mathsf{S}_\psi f|_{s=0}$.

Theorem

$\mathsf{L}_g f = f'g - g'f$

$$\begin{aligned}
\frac{d}{ds}\mathsf{S}_\psi f &= \frac{d}{ds}(\psi')^{-1}\, f\circ\psi && \text{def. of } \mathsf{S}_\psi \\
&= (\psi')^{-1}\, f'\circ\psi\, \frac{d}{ds}\psi + \frac{d}{ds}(\psi')^{-1}\, f\circ\psi && \text{prod. \& chain rules} \\
&= (\psi')^{-1}\, f'\circ\psi\; g\circ\psi - \underbrace{(\psi')^{-1}\, g'\circ\psi}_{i.o.u.}\; f\circ\psi && g \text{ generates } \psi
\end{aligned}$$

$$\begin{aligned}
\frac{d}{ds}\, \mathsf{S}_\psi f|_{s=0} &= f'g - g'f && \text{substitute } s=0 \\
&\overset{\text{def}}{=} [f,g] && \text{‘Lie bracket’}
\end{aligned}$$

The Normalization Machine

- How can we calculate the righthand side of the new o.d.e. $\dot{x} = \mathsf{S}_\psi f(x)$?
- Using the Lie operator $\mathsf{L}_g f \stackrel{\text{def}}{=} \frac{d}{ds}\, \mathsf{S}_\psi f|_{s=0}$.

Theorem

$\mathsf{L}_g f = f'g - g'f$

$$\begin{aligned}
\frac{d}{ds}\mathsf{S}_\psi f &= \frac{d}{ds}(\psi')^{-1}\ f \circ \psi && \text{def. of } \mathsf{S}_\psi \\
&= (\psi')^{-1}\ f' \circ \psi\ \frac{d}{ds}\psi + \frac{d}{ds}(\psi')^{-1}\ f \circ \psi && \text{prod. \& chain rules} \\
&= (\psi')^{-1}\ f' \circ \psi\ g \circ \psi - \underbrace{(\psi')^{-1}\ g' \circ \psi}_{i.o.u.}\ f \circ \psi && g \text{ generates } \psi
\end{aligned}$$

$$\begin{aligned}
\frac{d}{ds}\ \mathsf{S}_\psi f|_{s=0} &= f'g - g'f && \text{substitute } s = 0 \\
&\stackrel{\text{def}}{=} [f, g] && \text{‘Lie bracket’}
\end{aligned}$$

The Normalization Machine

- How can we calculate the righthand side of the new o.d.e. $\dot{x} = \mathsf{S}_\psi f(x)$?
- Using the Lie operator $\mathsf{L}_g f \overset{\text{def}}{=} \frac{d}{ds}\, \mathsf{S}_\psi f|_{s=0}$.

Theorem

$\mathsf{L}_g f = f'g - g'f$

$$
\begin{aligned}
\frac{d}{ds}\mathsf{S}_\psi f &= \frac{d}{ds}(\psi')^{-1}\ f \circ \psi && \text{def. of } \mathsf{S}_\psi \\
&= (\psi')^{-1}\ f' \circ \psi\ \frac{d}{ds}\psi + \frac{d}{ds}(\psi')^{-1}\ f \circ \psi && \text{prod. \& chain rules} \\
&= (\psi')^{-1}\ f' \circ \psi\ g \circ \psi - \underbrace{(\psi')^{-1}\ g' \circ \psi}_{i.o.u.}\ f \circ \psi && g \text{ generates } \psi
\end{aligned}
$$

$$
\begin{aligned}
\frac{d}{ds}\ \mathsf{S}_\psi f|_{s=0} &= f'g - g'f && \text{substitute } s = 0 \\
&\overset{\text{def}}{=} [f, g] && \text{`Lie bracket'}
\end{aligned}
$$

The Normalization Machine

- How can we calculate the righthand side of the new o.d.e. $\dot{x} = \mathsf{S}_\psi f(x)$?
- Using the exponential of the Lie operator $e^{\mathsf{L}_g} f$.

Theorem

$S_\psi f = e^{L_g} f$

$$\frac{d}{ds}\mathsf{S}_\psi f = \frac{d}{ds}(\psi')^{-1} f \circ \psi = (\psi')^{-1}\, \mathsf{L}_g f \circ \psi \qquad \text{last slide}$$

$$\frac{d^j}{ds^j}\mathsf{S}_\psi f = (\psi')^{-1}\, \mathsf{L}_g^j f \circ \psi \qquad \text{iteration}$$

$$\frac{d^j}{ds^j}\, \mathsf{S}_\psi f\big|_{s=0} = \mathsf{L}_g^j f \qquad \text{substitute } s = 0$$

$$\mathsf{S}_\psi f = \left(I + \mathsf{L}_g + \frac{1}{2!}\mathsf{L}_g^2 + \cdots \right) f \qquad \text{Taylor series}$$

The Normalization Machine

- How can we calculate the righthand side of the new o.d.e. $\dot{x} = \mathsf{S}_\psi f(x)$?
- Using the exponential of the Lie operator $e^{\mathsf{L}_g} f$.

Theorem

$S_\psi f = e^{L_g} f$

$$\frac{d}{ds}\mathsf{S}_\psi f = \frac{d}{ds}(\psi')^{-1} f \circ \psi = (\psi')^{-1}\, \mathsf{L}_g f \circ \psi \qquad \text{last slide}$$

$$\frac{d^j}{ds^j}\mathsf{S}_\psi f = (\psi')^{-1}\, \mathsf{L}_g^j f \circ \psi \qquad \text{iteration}$$

$$\frac{d^j}{ds^j}\, \mathsf{S}_\psi f\Big|_{s=0} = \mathsf{L}_g^j f \qquad \text{substitute } s = 0$$

$$\mathsf{S}_\psi f = \left(I + \mathsf{L}_g + \frac{1}{2!}\mathsf{L}_g^2 + \cdots \right) f \qquad \text{Taylor series}$$

The Normalization Machine

- How can we calculate the righthand side of the new o.d.e. $\dot{x} = \mathsf{S}_\psi f(x)$?
- Using the exponential of the Lie operator $e^{\mathsf{L}_g} f$.

Theorem

$S_\psi f = e^{L_g} f$

$$\frac{d}{ds}\mathsf{S}_\psi f = \frac{d}{ds}(\psi')^{-1} f \circ \psi = (\psi')^{-1}\, \mathsf{L}_g f \circ \psi \qquad \text{last slide}$$

$$\frac{d^j}{ds^j}\mathsf{S}_\psi f = (\psi')^{-1}\, \mathsf{L}_g^j f \circ \psi \qquad \text{iteration}$$

$$\frac{d^j}{ds^j}\, \mathsf{S}_\psi f\Big|_{s=0} = \mathsf{L}_g^j f \qquad \text{substitute } s = 0$$

$$\mathsf{S}_\psi f = \left(I + \mathsf{L}_g + \frac{1}{2!}\mathsf{L}_g^2 + \cdots \right) f \qquad \text{Taylor series}$$

The Normalization Machine

- How can we calculate the righthand side of the new o.d.e. $\dot{x} = \mathsf{S}_\psi f(x)$?
- Using the exponential of the Lie operator $e^{\mathsf{L}_g} f$.

Theorem

$S_\psi f = e^{L_g} f$

$$\frac{d}{ds}\mathsf{S}_\psi f = \frac{d}{ds}(\psi')^{-1} f \circ \psi = (\psi')^{-1}\ \mathsf{L}_g f \circ \psi \qquad \text{last slide}$$

$$\frac{d^j}{ds^j}\mathsf{S}_\psi f = (\psi')^{-1}\ \mathsf{L}_g^j f \circ \psi \qquad \text{iteration}$$

$$\frac{d^j}{ds^j}\ \mathsf{S}_\psi f\Big|_{s=0} = \mathsf{L}_g^j f \qquad \text{substitute } s = 0$$

$$\mathsf{S}_\psi f = \left(I + \mathsf{L}_g + \frac{1}{2!}\mathsf{L}_g^2 + \cdots\right) f \qquad \text{Taylor series}$$

The Normalization Machine

- How can we calculate the righthand side of the new o.d.e. $\dot{x} = \mathsf{S}_\psi f(x)$?
- Using the exponential of the Lie operator $e^{\mathsf{L}_g} f$.

Theorem

$S_\psi f = e^{L_g} f$

$$\frac{d}{ds}\mathsf{S}_\psi f = \frac{d}{ds}(\psi')^{-1} f \circ \psi = (\psi')^{-1}\ \mathsf{L}_g f \circ \psi \qquad \text{last slide}$$

$$\frac{d^j}{ds^j}\mathsf{S}_\psi f = (\psi')^{-1}\ \mathsf{L}_g^j f \circ \psi \qquad \text{iteration}$$

$$\frac{d^j}{ds^j}\ \mathsf{S}_\psi f\big|_{s=0} = \mathsf{L}_g^j f \qquad \text{substitute } s = 0$$

$$\mathsf{S}_\psi f = \left(I + \mathsf{L}_g + \frac{1}{2!}\mathsf{L}_g^2 + \cdots \right) f \qquad \text{Taylor series}$$

The Normalization Machine

- How can we calculate the righthand side of the new o.d.e. $\dot{x} = \mathsf{S}_\psi f(x)$?
- Using the exponential of the Lie operator $e^{\mathsf{L}_g} f$.

Theorem

$S_\psi f = e^{L_g} f$

$$\frac{d}{ds}\mathsf{S}_\psi f = \frac{d}{ds}(\psi')^{-1} f \circ \psi = (\psi')^{-1}\, \mathsf{L}_g f \circ \psi \qquad \text{last slide}$$

$$\frac{d^j}{ds^j}\mathsf{S}_\psi f = (\psi')^{-1}\, \mathsf{L}_g^j f \circ \psi \qquad \text{iteration}$$

$$\frac{d^j}{ds^j}\, \mathsf{S}_\psi f\big|_{s=0} = \mathsf{L}_g^j f \qquad \text{substitute } s = 0$$

$$\mathsf{S}_\psi f = \left(I + \mathsf{L}_g + \frac{1}{2!}\mathsf{L}_g^2 + \cdots \right) f \qquad \text{Taylor series}$$

The Normalization Machine

$$
\begin{tikzcd}
\psi \arrow[d, "\mathsf{D}_s"'] & \varphi \arrow[r, "\mathrm{C}_\psi"] \arrow[d, "\mathsf{D}_t"'] & \psi^{-1} \circ \varphi \circ \psi \arrow[d, "\mathrm{D}_t"] \\
g & f \arrow[r, "\mathsf{S}_\psi"] \arrow[dr, "\mathsf{L}_g"'] & (\psi')^{-1}\, f \circ \psi \arrow[d, "\mathsf{D}_s"] \\
 & & {[f, g]}
\end{tikzcd}
$$

The Normalization Machine

Recap

- $\dot{x} = f(x) \sim f_1(x) + f_2(x) + \cdots$ (Taylor series) where each f_j is a homogenous vector field of degree j
- If g generates the flow ψ, then the substitution $x \to \psi(x)$ transforms the o.d.e $\dot{x} = f(x)$ to

$$\begin{aligned}\dot{x} &= \mathsf{S}_\psi f(x) \sim e^{\mathsf{L}_g} f(x) \\ &= (I + \mathsf{L}_g + \cdots)(f_1(x) + f_2(x) + \cdots)\end{aligned}$$

- $\mathsf{L}_g f_i = f_i' g - g' f_i$

The Normalization Algorithm

- If g_j generates ψ_j and $\deg(g_j) = j$, then the substitution $x \to \psi_j(x)$ transforms the o.d.e. $\dot{x} = f(x)$ up to degree j as

$$\begin{aligned}\dot{x} \sim & (I + \mathsf{L}_j + \cdots)(f_1 + f_2 + \cdots + f_j + \cdots) \\ = & f_1 + \cdots + f_{j-1} + f_j + \mathsf{L}_j f_1 + \cdots\end{aligned}$$

  denoting L_{g_j} by L_j

- Define $h_j = f_j + \mathsf{L}_j f_1$, then

$$\begin{aligned} h_j &= f_j + f_1' g_j - g_j' f_1 \\ \mathcal{L} g_j &= f_j - h_j \end{aligned}$$

  where $\mathcal{L}(\cdot) \stackrel{\text{def}}{=} [\cdot, f_1]$ is the 'Fundamental Lie operator for normal form theory.'

The Normalization Algorithm

- At the j^{th} step of normalization, $\mathcal{L}g_j = f_j - h_j$
  - set h_j to the projection of f_j into $\overline{\text{im}\,\mathcal{L}}$
  - solve $\mathcal{L}g_j = f_j - h_j$ for g_j
  - calculate L_j and the rest of the transformation $\dot{x} = \left(I + \mathsf{L}_j + \frac{1}{2}\mathsf{L}_j^2 + \cdots\right)(f_1 + f_2 + f_3 + \cdots)$
- The full transformation $x \to \cdots \circ \psi_4 \circ \psi_3 \circ \psi_2(x)$ modifies the o.d.e. $\dot{x} = f(x)$ to

$$\dot{x} = f_1(x) + h_2(x) + h_3(x) + h_4(x) + \cdots$$

The Normalization Algorithm

$$\mathcal{L}\begin{bmatrix} g_1(x) \\ g_2(x) \end{bmatrix} = \begin{bmatrix} \frac{\partial g_1}{\partial x_1} & \frac{\partial g_1}{\partial x_2} \\ \frac{\partial g_2}{\partial x_1} & \frac{\partial g_2}{\partial x_2} \end{bmatrix} \begin{bmatrix} x_2 \\ 0 \end{bmatrix} - \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} g_1(x) \\ g_2(x) \end{bmatrix}$$

- $\mathcal{L}$ acting on quadratic vector fields can be represented as a matrix with nontrivial left eigenvectors to 0:

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 2 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

```
print lie(2)
```

```
[[ 0.   0.   0.  -1.   0.   0.]
 [ 2.   0.   0.   0.  -1.   0.]
 [ 0.   1.   0.   0.   0.  -1.]
 [ 0.   0.   0.   0.   0.   0.]
 [ 0.   0.   0.   2.   0.   0.]
 [ 0.   0.   0.   0.   1.   0.]]
```

- In general, a basis for $\overline{\text{im}\mathcal{L}_j}$ is $\{e_{j+2}, je_1 + e_{j+3}\}$, 'inner product style'

The Normalization Algorithm

- Determine h_j and solve $\mathcal{L}g_j = f_j - h_j$ for g_j

```
def generator(j,F,style='inner_product'):
    g = np.zeros(2*j+2); h = np.zeros(2*j+2)
    img = np.zeros(2*j+2)
    if style == 'inner_product':
        tmp = (j*F[j][0]+F[j][j+2])/float(1.+j**2)
        h[0] = j*tmp; h[j+1] = F[j][j+1]; h[j+2] = tmp
        img = F[j]-h
        for i in range(j+1,2*j+1):
            g[i] = img[i+1]/float(2*j-i+1.)
        for i in range(j-1):
            g[i] = (img[i+1]+g[i+j+2])/float(j-i)
        g[j] = 0.
        g[j-1] = .5*img[j]
        g[2*j+1] = -g[j-1]
    return g,h
print generator(2,jordan(taylor(10)))
```

```
array([ 0.15470054,  0.04840153,  0.        ,  0.06069202,  0.11579495,
       -0.04840153])
array([ 0.15749509,  0.        ,  0.        ,  0.08433663,  0.07874754,  0.
        ])
```

The Normalization Algorithm

- Determine the matrix form of L_j from the generator g_j

```
def lie(j,g=np.array([0,1,0,0])):
    k = len(g)/2-1
    g_ = [sp.symbols('g%d'%(i+1)) for i in range(2*k+2)]
    G_ = sp.Matrix([[sum([g_[i]*x_[0]**(k-i)*x_[1]**i for i in range(k+1)])],
                    [sum([g_[i+k+1]*x_[0]**(k-i)*x_[1]**i for i in range(k+1)])
     ]])
    DG_ = sp.Matrix([[sp.diff(G_[0],x_[0]),sp.diff(G_[0],x_[1])],
                     [sp.diff(G_[1],x_[0]),sp.diff(G_[1],x_[1])]])
    bas_ = vf_basis(j)
    Dbas_ = [sp.Matrix([[sp.diff(bas_[i][0],x_[0]),sp.diff(bas_[i][0],x_[1])],
             [sp.diff(bas_[i][1],x_[0]),sp.diff(bas_[i][1],x_[1])]]) for i in
     range(2*j+2)]
    L = sp.zeros(2*(j+k),2*j+2)
    for i in range(2*j+2):
        bracket = sp.expand(Dbas_[i]*G_-DG_*bas_[i])
        L[:j+k,i] = [bracket[0].coeff(x_[0]**(j+k-1-l)*x_[1]**l) for l in range
     (j+k)]
        L[j+k:,i] = [bracket[1].coeff(x_[0]**(j+k-1-l)*x_[1]**l) for l in range
     (j+k)]
    for i in range(2*k+2):
        L = L.subs(g_[i],g[i])
    return np.array(L).astype(float)
```

The Normalization Algorithm

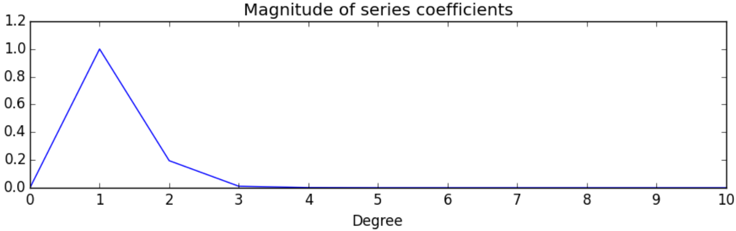
- Apply the transformation e^{L_j} to the righthand side of the o.d.e. for $j = 2, 3, 4, \ldots$

```
def normalize(F):
    k = len(F)-1
    h = (k+1)*[0]; h[0:2] = F[0:2]
    for j in range(2,k+1):
        u,h[j] = generator(j,F); dF = []
        for i in range(j,k+1):
            l = (i-1)/(j-1); m = i-l*(j-1)
            dF.append(np.dot(lie(m,u),F[m])/float(l))
            for n in range(1,l):
                dF[-1] += F[m+n*(j-1)]
                dF[-1] = np.dot(lie(m+n*(j-1),u),dF[-1])/float(l-n)
        for i in range(len(dF)):
            F[j+i] += dF[i]
    return h
normalize(jordan(taylor(10)))
```

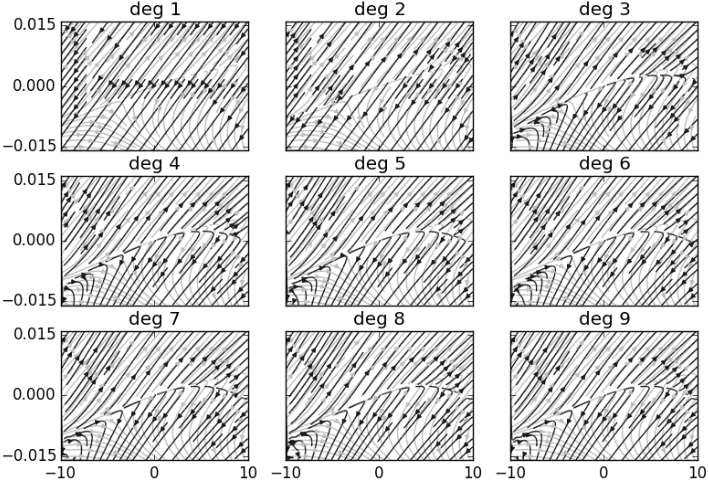
Outline

The Normalized System with Linear Term in Jordan Form

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ 0 \end{bmatrix} + \begin{bmatrix} a_{00}^{(2)} & 0 & 0 \\ a_{10}^{(2)} & a_{11}^{(2)} & 0 \end{bmatrix} \begin{bmatrix} x_1^2 \\ x_1 x_2 \\ x_2^2 \end{bmatrix} + \begin{bmatrix} a_{00}^{(3)} & 0 & 0 & 0 \\ a_{10}^{(3)} & a_{11}^{(3)} & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1^3 \\ x_1^2 x_2 \\ x_1 x_2^2 \\ x_2^3 \end{bmatrix} + \cdots$$

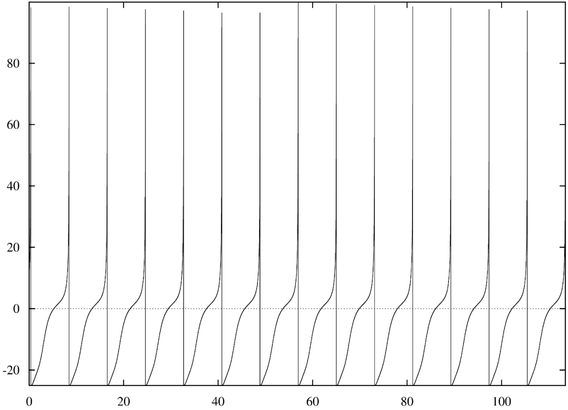


The Normalized System Vector Field



The Normalized System Simulation

- Simulation of the normalized system truncated at degree 3, with resetting condition $x_1 \leftarrow -25$ and $x_2 \leftarrow -.025$ if $x_1 > 100$.



TODO

- Compute the unfolding/bifurcation diagram of the normalized system
- Use optimization to approximate the location of the BT bifurcation more accurately
- Investigate the geometric properties of the normal form
  - The flow of each h_i commutes with the flow of $J^T \Rightarrow [J^T x, h_i(x)] = 0$ (Is my code working?)
  - In some cases the stable, unstable, and center manifolds can be calculated at the same time as normalization
- Convert the system to normal form at other bifurcations