Swift-Hohenberg Numerics - Testing with Chebfun

Edward McDugald

October 18, 2022

1 Operator Splitting for SH

- We have the PDE
$$w_t = -(1+\nabla^2)^2 w + Rw - w^3.$$
- We can break this into a linear part and a non-linear part,
$$W_t = L(w) + NL(w),$$
with
$$\begin{aligned} L(w) &= -(2\nabla^2 + \nabla^4)w \\ NL(w) &= (R-1)w - w^3. \end{aligned}$$
- Basic procedure is as follows:
  (i) Let $A = (-\nabla^4 - 2\nabla^2)$. Consider the pair of PDEs
  $$\begin{aligned} w_t &= Aw \\ w_t &= (R-1)w - w^3. \end{aligned}$$
  (ii) Handling the linear and nonlinear PDEs separately, we get the relations
  $$\begin{aligned} & w(t+\Delta t) = \Delta t A w(t) + w(t) \\ \implies & w(t+\Delta t) \approx e^{A\Delta t} w(t). \end{aligned}$$
  And for the nonlinear part,
  $$w(t+\Delta t) \approx \Delta t \left[(R-1)w(t) - w(t)^3\right] + w(t).$$
  .
  (iii) Apply operator splitting (strang splitting) now

The code is implemented in python as follows:

```
def non_lin_rhs(w,R):
    return (R-1)*w - w**3

def integrateSH(w0,R,dt,nSteps,L):
     """
     :param w0: initial temperature surface
     :param R: bifurcation parameter- can be a constant, or of same shape as w0
     :param dt: time step length
     :param nSteps: number of time steps to take
     :param L: Length of square over which w0 is defined
     :return w0: time evolution of w0 at time 0+dt*nSteps
     Ideally, the size of w0 is fft friendly, ie 2^n x 2^n
     """
     print("Starting time integration of Swift Hohenberg")
     ny, nx = np.shape(w0)
     print("Dimensions of w0:", nx, ny)
     kx = (2.*np.pi/L)*sp.fft.fftfreq(nx,1./nx)
     ky = (2.*np.pi/L)*sp.fft.fftfreq(ny,1./ny)
     Kx, Ky = np.meshgrid(kx,ky)
     fourierLaplacian = -(Kx**2+Ky**2)
     A = -(fourierLaplacian*fourierLaplacian)-2*fourierLaplacian
     for i in range(0,nSteps):
         if i%100 == 0:
             print("step number:",i)
         w1 = np.real(sp.fft.ifft2(np.exp(A*.5*dt)*sp.fft.fft2(w0)))

         #rk4 version
         #k1 = dt*w1
         #k2 = dt*non_lin_rhs(w1+.5*k1, R)
         #k3 = dt*non_lin_rhs(w1+.5*k2, R)
         #k4 = dt*non_lin_rhs(w1+k3, R)
         #w2 = (k1+2*k2+2*k3+k4)/6 + w1

         #fwd euler version
         w2 = dt*((R-1)*w1-w1**3)+w1

         w0 = np.real(sp.fft.ifft2(np.exp(A*.5*dt)*sp.fft.fft2(w2)))
     return w0
```

2 Comparison with Chebfun

2.1 Chebfun vs Python 1

Chebfun 1 code

```
>> dom = [-16 16 -16 16]
>> tspan = [0 10];
>> S = spinop2(dom, tspan)
>> S.lin = @(u) -2*lap(u)-biharm(u)
>> R = .5
>> S.nonlin = @(u) (R-1)*u - u.^3;
>> u0 = 1/20*chebfun2(@(x,y) cos(x) + sin(2*x) + sin(y) + cos(2*y), dom, 'trig');
u0 = u0 + chebfun2(@(x,y) exp(-((x-5*pi).^2 + (y-5*pi).^2)), dom, 'trig');
u0 = u0 + chebfun2(@(x,y) exp(-((x-5*pi).^2 + (y-15*pi).^2)), dom, 'trig');
u0 = u0 + chebfun2(@(x,y) exp(-((x-15*pi).^2 + (y-15*pi).^2)), dom, 'trig');
u0 = u0 + chebfun2(@(x,y) exp(-((x-15*pi).^2 + (y-5*pi).^2)), dom, 'trig');
u0 = u0 + chebfun2(@(x,y) exp(-((x-10*pi).^2 + (y-10*pi).^2)), dom, 'trig');
S.init = u0;
>> u = spin2(S, 256, .1, 'plot', 'off');
>> plot(u), view(0,90), axis equal, axis off
>> saveas(gcf,'/Users/edwardmcdugald/Research/
convection_patterns_matlab/figs/sh_tsts_1018/cf1.pdf');
```

python 1 code

```
x = np.linspace(-16,16,256)
y = np.linspace(-16,16,256)
X,Y = np.meshgrid(x,y)
w0 = (1./20.)*(np.cos(X)+np.sin(2*X)+np.sin(Y)+np.cos(2*Y))
w0 = w0 + (np.exp(-((X-5*np.pi)**2 +(Y-5*np.pi)**2)))
w0 = w0 + (np.exp(-((X-5*np.pi)**2 +(Y-15*np.pi)**2)))
w0 = w0 + (np.exp(-((X-15*np.pi)**2 +(Y-15*np.pi)**2)))
w0 = w0 + (np.exp(-((X-15*np.pi)**2 +(Y-5*np.pi)**2)))
w0 = w0 + (np.exp(-((X-10*np.pi)**2 +(Y-10*np.pi)**2)))
dt = .1
R=.5
L=x[len(x)-1]-x[0]
nSteps = 100
W1 = integrateSH(w0,R,dt,nSteps,L)
fig1, ax1 = plt.subplots(nrows=1, ncols=1, figsize=(3,3))
ax1.imshow(W1)
```

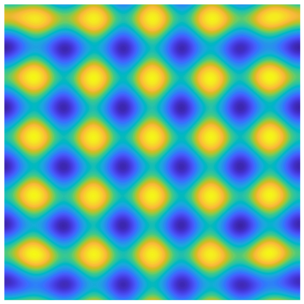

Figure 1: Chebfun

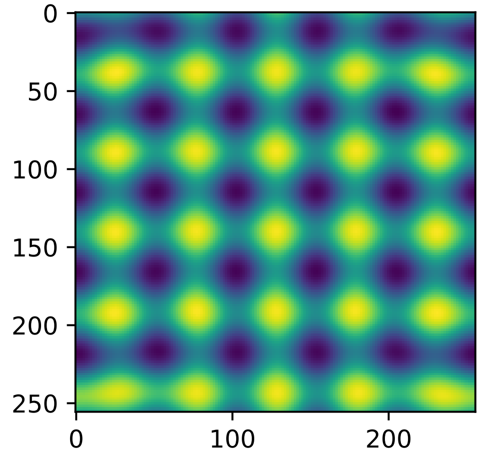

Figure 2: Python

```
plt.savefig("/Users/edwardmcdugald/Research/
convection_patterns/code/figs/sh_num_tsts_1018/mySH1.pdf")
```

2.2 Chebfun vs Python 2

chebfun 2 code

```
dom = [0 20*pi 0 20*pi];
tspan = [0 200];
S = spinop2(dom, tspan);
S.lin = @(u) -2*lap(u) - biharm(u);
r = 1e-2;
S.nonlin = @(u) (-1 + r)*u - u.^3;
u0 = 1/20*chebfun2(@(x,y) cos(x) + sin(2*x) + sin(y) + cos(2*y), dom, 'trig');
u0 = u0 + chebfun2(@(x,y) exp(-((x-5*pi).^2 + (y-5*pi).^2)), dom, 'trig');
u0 = u0 + chebfun2(@(x,y) exp(-((x-5*pi).^2 + (y-15*pi).^2)), dom, 'trig');
u0 = u0 + chebfun2(@(x,y) exp(-((x-15*pi).^2 + (y-15*pi).^2)), dom, 'trig');
u0 = u0 + chebfun2(@(x,y) exp(-((x-15*pi).^2 + (y-5*pi).^2)), dom, 'trig');
u0 = u0 + chebfun2(@(x,y) exp(-((x-10*pi).^2 + (y-10*pi).^2)), dom, 'trig');
S.init = u0;
plot(S.init), view(0,90), axis equal, axis off
```

```
u = spin2(S, 96, 2e-1, 'plot', 'off');
plot(u), view(0,90), axis equal, axis off
>> saveas(gcf,'/Users/edwardmcdugald/Research/
convection_patterns_matlab/figs/sh_tsts_1018/cf2.pdf');
```

python 2 code

```
x = np.linspace(0,20*np.pi,96)
y = np.linspace(0,20*np.pi,96)
X,Y = np.meshgrid(x,y)
w0 = (1./20.)*(np.cos(X)+np.sin(2*X)+np.sin(Y)+np.cos(2*Y))
w0 = w0 + (np.exp(-((X-5*np.pi)**2 +(Y-5*np.pi)**2)))
w0 = w0 + (np.exp(-((X-5*np.pi)**2 +(Y-15*np.pi)**2)))
w0 = w0 + (np.exp(-((X-15*np.pi)**2 +(Y-15*np.pi)**2)))
w0 = w0 + (np.exp(-((X-15*np.pi)**2 +(Y-5*np.pi)**2)))
w0 = w0 + (np.exp(-((X-10*np.pi)**2 +(Y-10*np.pi)**2)))
fig, ax = plt.subplots(nrows=1, ncols=1, figsize=(3,3))
ax.imshow(w0)
dt = 2e-1
R= 1e-2
L=x[len(x)-1]-x[0]
nSteps = 1000
W2 = integrateSH(w0,R,dt,nSteps,L)
fig2, ax2 = plt.subplots(nrows=1, ncols=1, figsize=(3,3))
ax2.imshow(W2)
plt.savefig("/Users/edwardmcdugald/Research/
convection_patterns/code/figs/sh_num_tsts_1018/mySH2.pdf")
```

2.3 Chebfun vs Python 3

chebfun 3 code

```
>> dom = [0 20*pi 0 20*pi];
tspan = [0 200];
S = spinop2(dom, tspan);
S.lin = @(u) -2*lap(u) - biharm(u);
>> u0 = 1/20*chebfun2(@(x,y) cos(x) + sin(2*x) + sin(y) + cos(2*y), dom, 'trig');
u0 = u0 + chebfun2(@(x,y) exp(-((x-5*pi).^2 + (y-5*pi).^2)), dom, 'trig');
u0 = u0 + chebfun2(@(x,y) exp(-((x-5*pi).^2 + (y-15*pi).^2)), dom, 'trig');
u0 = u0 + chebfun2(@(x,y) exp(-((x-15*pi).^2 + (y-15*pi).^2)), dom, 'trig');
u0 = u0 + chebfun2(@(x,y) exp(-((x-15*pi).^2 + (y-5*pi).^2)), dom, 'trig');
```

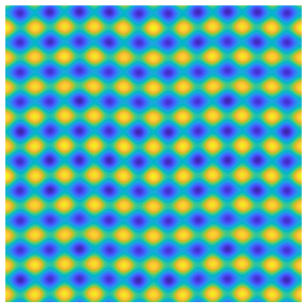

Figure 3: Chebfun

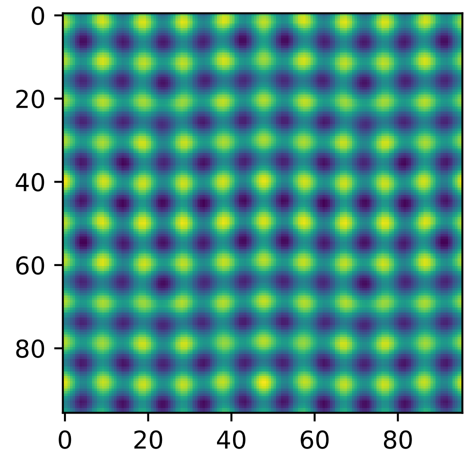

Figure 4: Python

```
u0 = u0 + chebfun2(@(x,y) exp(-((x-10*pi).^2 + (y-10*pi).^2)), dom, 'trig');
S.init = u0;
>> r = 7e-1;
>> S.nonlin = @(u) (-1 + r)*u - u.^3;
>> u = spin2(S, 96, 2e-1, 'plot', 'off');
>> plot(u), view(0,90), axis equal, axis off
>> saveas(gcf,'/Users/edwardmcdugald/Research/
convection_patterns_matlab/figs/sh_tsts_1018/cf3.pdf');
```

python 3 code

```
x = np.linspace(0,20*np.pi,96)
y = np.linspace(0,20*np.pi,96)
X,Y = np.meshgrid(x,y)
w0 = (1./20.)*(np.cos(X)+np.sin(2*X)+np.sin(Y)+np.cos(2*Y))
w0 = w0 + (np.exp(-((X-5*np.pi)**2 +(Y-5*np.pi)**2)))
w0 = w0 + (np.exp(-((X-5*np.pi)**2 +(Y-15*np.pi)**2)))
w0 = w0 + (np.exp(-((X-15*np.pi)**2 +(Y-15*np.pi)**2)))
w0 = w0 + (np.exp(-((X-15*np.pi)**2 +(Y-5*np.pi)**2)))
w0 = w0 + (np.exp(-((X-10*np.pi)**2 +(Y-10*np.pi)**2)))
dt = 2e-1
R= 7e-1
L=x[len(x)-1]-x[0]
```

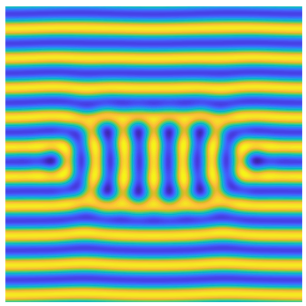

Figure 5: Chebfun

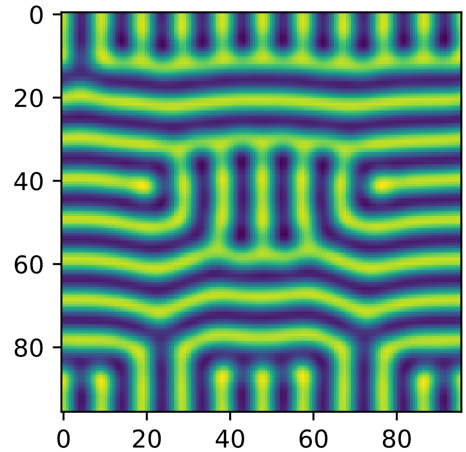

Figure 6: Python

```
nSteps = 1000
W3 = integrateSH(w0,R,dt,nSteps,L)
fig3, ax3 = plt.subplots(nrows=1, ncols=1, figsize=(3,3))
ax3.imshow(W3)
plt.savefig("/Users/edwardmcdugald/Research/
convection_patterns/code/figs/sh_num_tsts_1018/mySH3.pdf")
```

3 Chebfun v Python 4

chebfun 4 code

```
>> dom = [0 20*pi 0 20*pi];
tspan = [0 200];
>> S = spinop2(dom, tspan);
>> S.lin = @(u) -2*lap(u) - biharm(u);
>> u0 = 1/20*chebfun2(@(x,y) cos(x) + sin(2*x) + sin(y) + cos(2*y), dom, 'trig');
u0 = u0 + chebfun2(@(x,y) exp(-((x-5*pi).^2 + (y-5*pi).^2)), dom, 'trig');
u0 = u0 + chebfun2(@(x,y) exp(-((x-5*pi).^2 + (y-15*pi).^2)), dom, 'trig');
u0 = u0 + chebfun2(@(x,y) exp(-((x-15*pi).^2 + (y-15*pi).^2)), dom, 'trig');
u0 = u0 + chebfun2(@(x,y) exp(-((x-15*pi).^2 + (y-5*pi).^2)), dom, 'trig');
u0 = u0 + chebfun2(@(x,y) exp(-((x-10*pi).^2 + (y-10*pi).^2)), dom, 'trig');
```

```
S.init = u0;
>> r = 1e-1;
>> S.nonlin = @(u) (-1 + r)*u - u.^3;
>> u = spin2(S, 100, 2e-1, 'plot', 'off');
plot(u), view(0,90), axis equal, axis off
>> saveas(gcf,'/Users/edwardmcdugald/Research/
convection_patterns_matlab/figs/sh_tsts_1018/cf4.pdf');
```

python 4 code

```
x = np.linspace(0,20*np.pi,100)
y = np.linspace(0,20*np.pi,100)
X,Y = np.meshgrid(x,y)
w0 = (1./20.)*(np.cos(X)+np.sin(2*X)+np.sin(Y)+np.cos(2*Y))
w0 = w0 + (np.exp(-((X-5*np.pi)**2 +(Y-5*np.pi)**2)))
w0 = w0 + (np.exp(-((X-5*np.pi)**2 +(Y-15*np.pi)**2)))
w0 = w0 + (np.exp(-((X-15*np.pi)**2 +(Y-15*np.pi)**2)))
w0 = w0 + (np.exp(-((X-15*np.pi)**2 +(Y-5*np.pi)**2)))
w0 = w0 + (np.exp(-((X-10*np.pi)**2 +(Y-10*np.pi)**2)))
dt = 2e-1
R= 1e-1
L=x[len(x)-1]-x[0]
nSteps = 1000
W4 = integrateSH(w0,R,dt,nSteps,L)
fig4, ax4 = plt.subplots(nrows=1, ncols=1, figsize=(3,3))
ax4.imshow(W4)
plt.savefig("/Users/edwardmcdugald/Research/convection_patterns/code/figs/sh_num_tsts_10

nSteps = 10000
W4b = integrateSH(w0,R,dt,nSteps,L)
fig4b, ax4b = plt.subplots(nrows=1, ncols=1, figsize=(3,3))
ax4b.imshow(W4b)
plt.savefig("/Users/edwardmcdugald/Research/convection_patterns/code/figs/sh_num_tsts_10
```

ToDo: Compare Results with different initial value function ToDo: Compare results using RK4 instead of Fwd Euler ToDo: Add results of ellipse (with sigmoid)

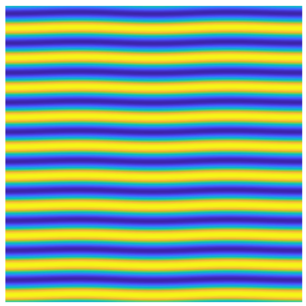

Figure 7: Chebfun

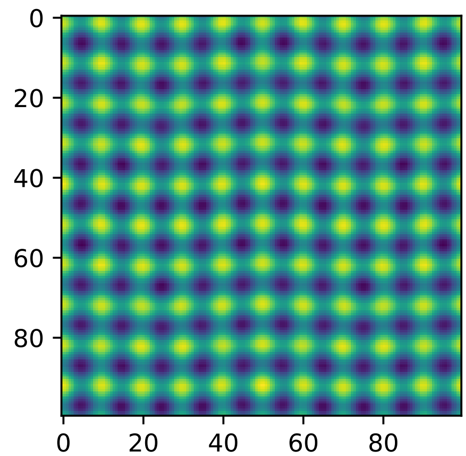

Figure 8: Python (using 1000 steps)

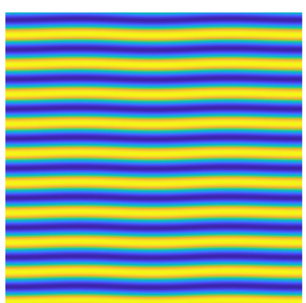

Figure 9: Chebfun

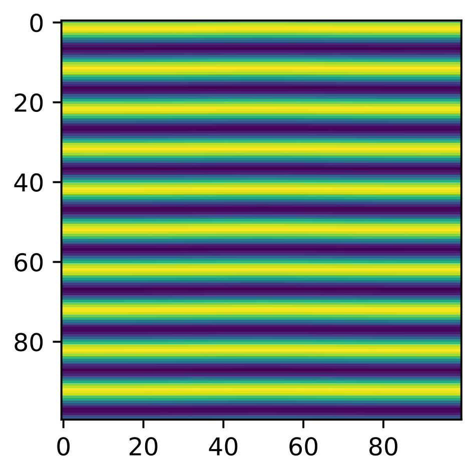

Figure 10: Python (using 10000 steps)