

```
In [1]: from numpy import *
        from matplotlib.pyplot import *

        %load_ext sympypr
```

This is an example session showing the interactive use of the WaveBlocks simulation packet.

```
In [2]: from WaveBlocks import *
```

Some simulation parameters:

```
In [3]: params = {"eps":0.1, "ncomponents":1, "potential":"quadratic", "dt":0.1, "matrix_exponential":"pade"}
```

Create a Hagedorn wavepacket  $\Psi$

```
In [4]: Phi = HagedornWavepacket(params)
```

Assign the parameter set  $\Pi$  with position 1.0 and momentum 0.5

```
In [5]: Pi = Phi.get_parameters()
        Pi
```

```
Out[5]: (1j, 1.0, 0.0, 0.0, 0.0)
```

```
In [6]: Pi = list(Pi)
        Pi[3] = 0.5
        Pi[4] = 1.0
        Pi
```

```
Out[6]: [1j, 1.0, 0.0, 0.5, 1.0]
```

```
In [7]: Phi.set_parameters(Pi)
```

Set the coefficients such that we start with a  $\phi_1$  packet

```
In [8]: Phi.set_coefficient(0,1,1)
```

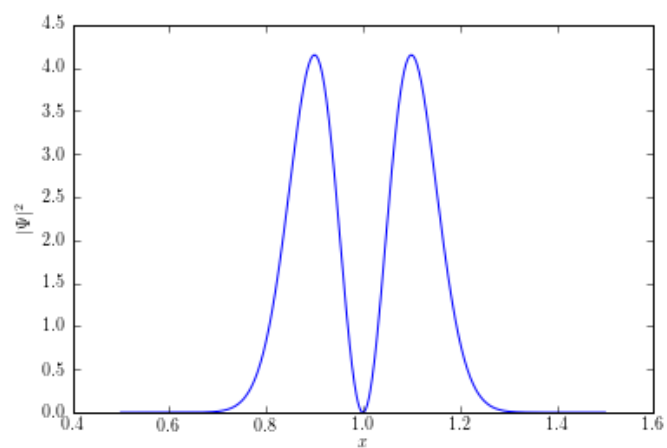
Plot the initial configuration

```
In [9]: x = linspace(0.5,1.5,1000)
```

```
In [10]: y = Phi.evaluate_at(x, prefactor=True)[0]
```

```
In [11]: plot(x, abs(y)**2)
        xlabel(r"$x$")
        ylabel(r"$|\Psi|^2$")
```

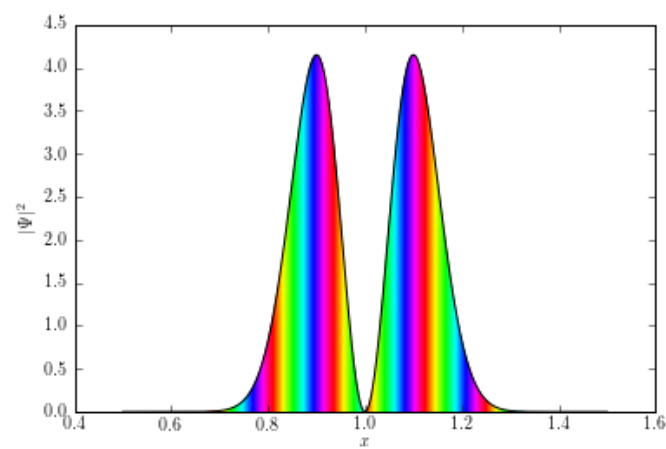
```
Out[11]: Text(0,0.5, '$|\Psi|^2$')
```



```
In [12]: from WaveBlocks.Plot import plotcf, stemcf
```

```
In [13]: plotcf(x, angle(y), abs(y)**2)
xlabel(r"$x$")
ylabel(r"$|\Psi|^2$")
```

Out[13]: Text(0,0.5, '\$|\Psi|^2\$')



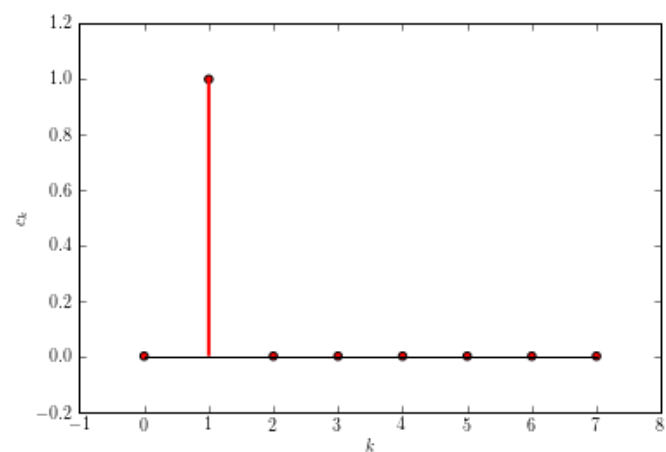
```
In [14]: c = Phi.get_coefficients(component=0)
c = squeeze(c)
```

```
In [15]: c.shape
```

Out[15]: (8,)

```
In [16]: figure(figsize=(6,4))
stemcf(arange(c.shape[0]), angle(c), abs(c)**2)
xlabel(r"$k$")
ylabel(r"$c_k$")
```

Out[16]: Text(0,0.5, '\$c\_k\$')



Set up the potential  $V(x)$  for our simulation. We use a simple harmonic oscillator.

```
In [17]: V = PotentialFactory.create_potential(params)
```

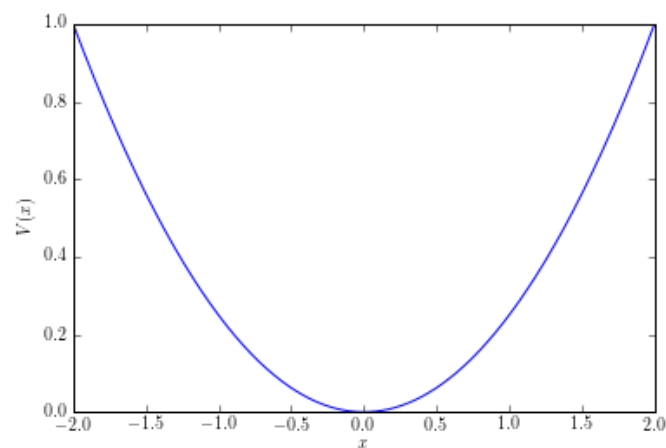
```
In [18]: V.potential
```

Out[18]:  $\frac{1}{4}x^2$

```
In [19]: u = linspace(-2,2,1000)
v = V.evaluate_at(u)[0]
```

```
In [20]: plot(u,v)
xlabel(r"$x$")
ylabel(r"$V(x)$")
```

Out[20]: Text(0,0.5, '\$V(x)\$')



Don't forget to set up the quadratur rule  $(\gamma, \omega)$

```
In [21]: Phi.set_quadrature(None)
```

```
In [22]: Q = Phi.get_quadrature()  
Q
```

```
Out[22]: <WaveBlocks.HomogeneousQuadrature.HomogeneousQuadrature instance at 0xb22712ac>
```

Now retrieve the bare quadrature rule  $(\gamma, \omega)$

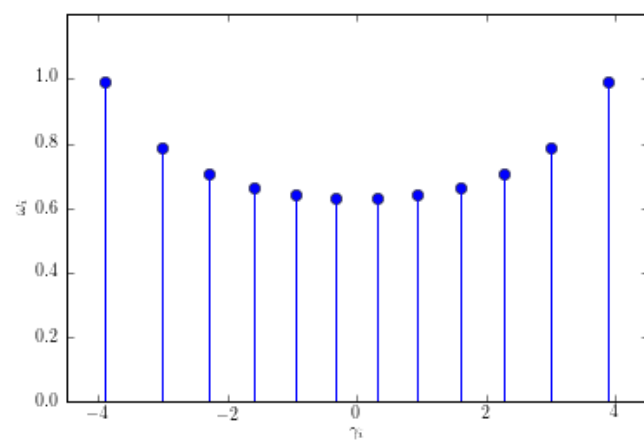
```
In [23]: QR = Q.get_qr()
```

And extract nodes and weights

```
In [24]: g = QR.get_nodes()  
w = QR.get_weights()
```

```
In [25]: figure(figsize=(6,4))  
stem(squeeze(real(g)),squeeze(real(w)))  
xlim(-4.5,4.5)  
ylim(0,1.2)  
xlabel(r"$\gamma_i$")  
ylabel(r"$\omega_i$")
```

```
Out[25]: Text(0,0.5,'$\omega_i$')
```



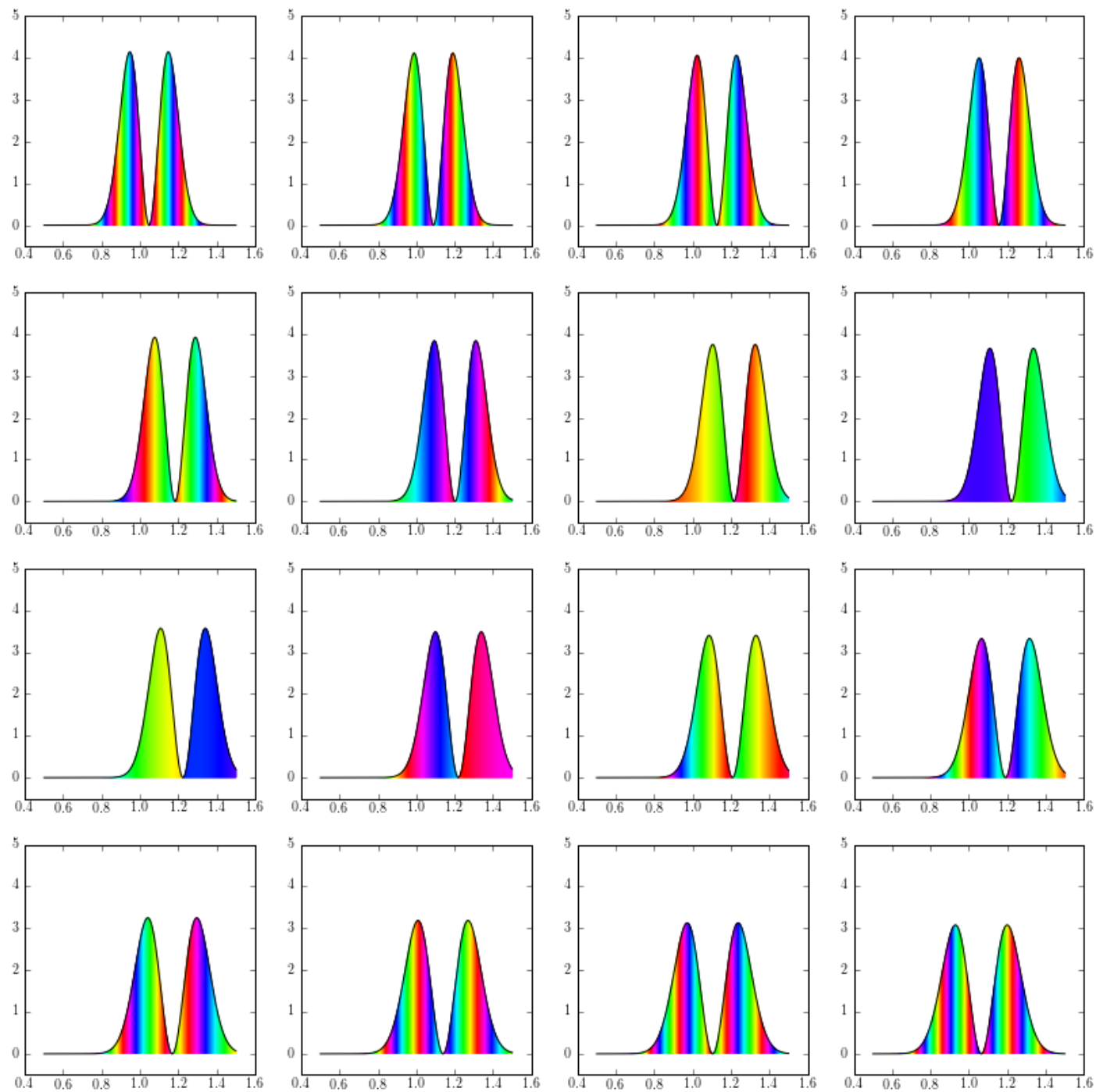
Now construct the time propagator

```
In [26]: P = HagedornPropagator(V, Phi, 0, params)
```

Propagate for 16 timesteps and plot each state

```
In [27]: fig = figure(figsize=(14,14))

for i in xrange(16):
    P.propagate()
    ynew = P.get_wavepackets().evaluate_at(x, prefactor=True)[0]
    ax = subplot(4,4,i+1)
    plotcf(x, angle(ynew), abs(ynew)**2, axes=ax)
    ax.set_ylim((-0.5, 5))
```

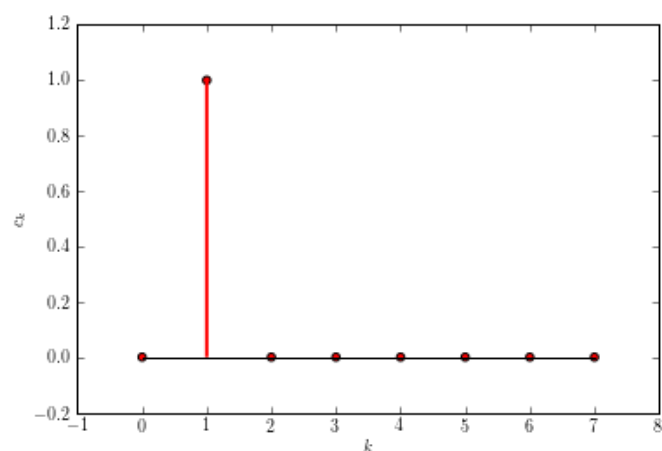


Look at the coefficients  $C$  again

```
In [28]: cnew = Phi.get_coefficients(component=0)
cnew = squeeze(cnew)
```

```
In [29]: figure(figsize=(6,4))
stemcf(arange(8), angle(cnew), abs(c)**2)
xlabel(r"$k$")
ylabel(r"$c_k$")
```

```
Out[29]: Text(0,0.5,'$c_k$')
```



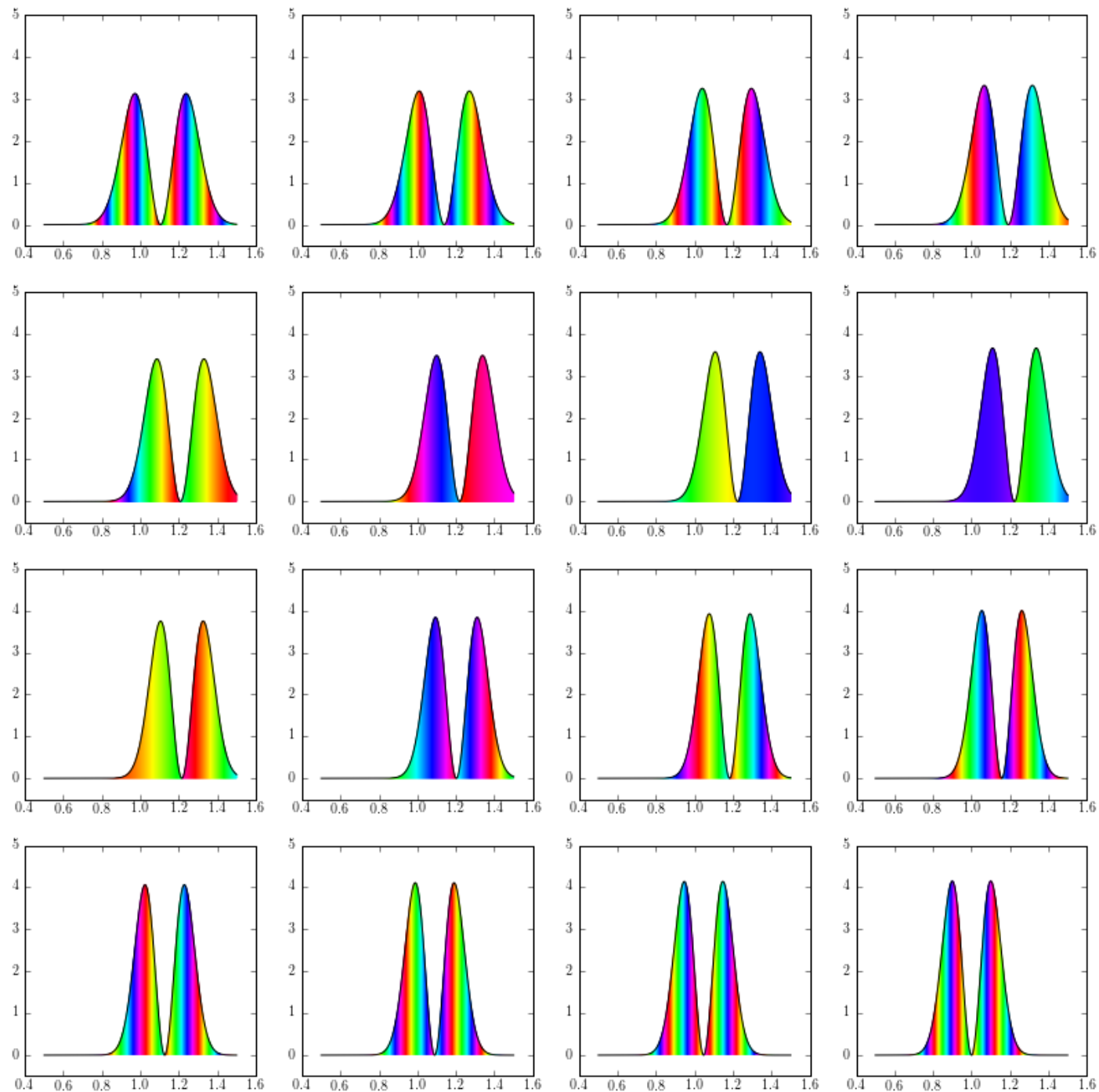
We see that the packet  $\Psi$  is still a  $\phi_1$

Now we go back in time ...

```
In [30]: params["dt"] *= -1
```

```
In [31]: Pinv = HagedornPropagator(V, Phi, 0, params)
```

```
In [32]: fig = figure(figsize=(14,14))  
  
for i in xrange(16):  
    Pinv.propagate()  
    ynew = Pinv.get_wavepackets().evaluate_at(x, prefactor=True)[0]  
    ax = subplot(4,4,i+1)  
    plotcf(x, angle(ynew), abs(ynew)**2, axes=ax)  
    ax.set_ylim((-0.5,5))
```



```
In [33]: Phi.get_parameters()
```

```
Out[33]: ((-2.08166817117e-17+1j), (1-1.17961196366e-16j), 3.29597460436e-17, 0.5, 1.0)
```

We see that the propagation is reversible up to machine precision!