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

%load_ext sympyprt
```

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

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

Some simulation parameters in a plain python dict:

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

From which we create a `ParameterProvider` instance

In [4]:
```
params = ParameterLoader().load_from_dict(configuration)
print(params)
```

```
====================================
Parameters of the current simulation
------------------------------------
------------------------------------
All parameters provided
------------------------------------
  dt: 0.1
  eps: 0.1
  ncomponents: 1
  potential: quadratic
====================================
```

Create a Hagedorn wavepacket $\Psi$

In [5]:
```
Psi = HagedornWavepacket(params)
```

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

In [6]:
```
Pi = Psi.get_parameters()
Pi
```

Out[6]:
```
(1j, 1.0, 0.0, 0.0, 0.0)
```

In [7]:
```
Pi = list(Pi)
Pi[3] = 0.5
Pi[4] = 1.0
Pi
```

Out[7]:
```
[1j, 1.0, 0.0, 0.5, 1.0]
```

In [8]:
```
Psi.set_parameters(Pi)
```

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

In [9]:
```
Psi.set_coefficient(0,1,1)
```

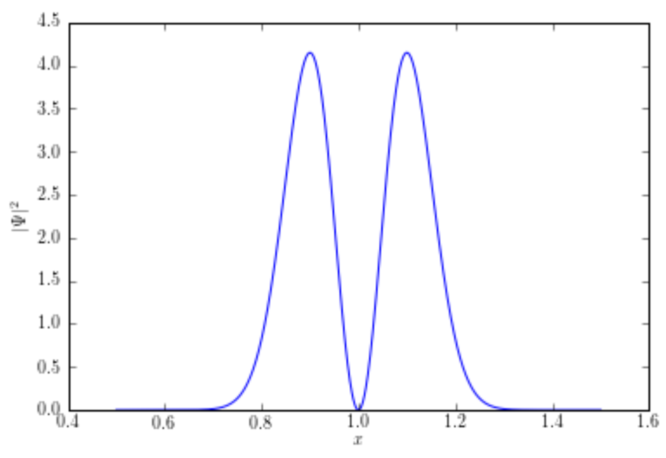
Plot the initial configuration

In [10]:
```
x = linspace(0.5,1.5,1000)
```

In [11]:
```
y = Psi.evaluate_at(x, prefactor=True)[0]
```

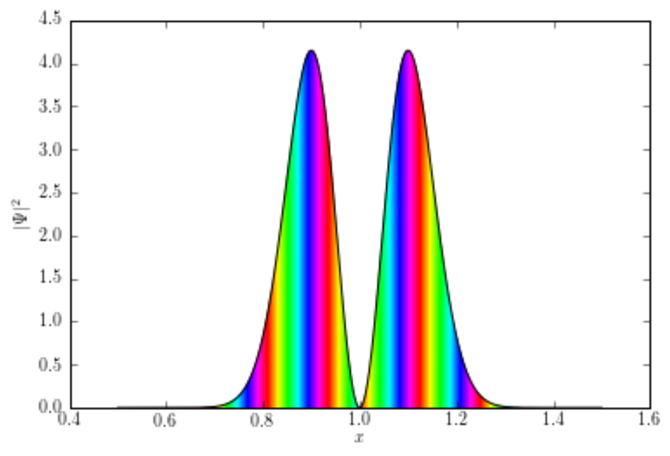
In [12]:
```
plot(x, abs(y)**2)
xlabel(r"$x$")
ylabel(r"$|\Psi|^2$")
```

Out[12]:
```
Text(0,0.5,'$|\\Psi|^2$')
```

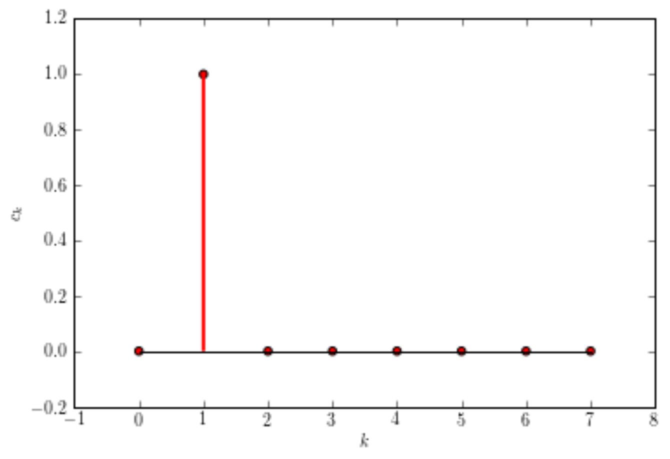


In [13]:
```
from WaveBlocks.Plot import plotcf, stemcf
```

```
In [14]: plotcf(x, angle(y), abs(y)**2)
         xlabel(r"$x$")
         ylabel(r"$|\Psi|^2$")

Out[14]: Text(0,0.5,'$|\\Psi|^2$')
```



```
In [15]: c = Psi.get_coefficients(component=0)
         c = squeeze(c)

In [16]: c.shape

Out[16]: (8,)

In [17]: figure(figsize=(6,4))
         stemcf(arange(c.shape[0]), angle(c), abs(c)**2)
         xlabel(r"$k$")
         ylabel(r"$c_k$")

Out[17]: Text(0,0.5,'$c_k$')
```

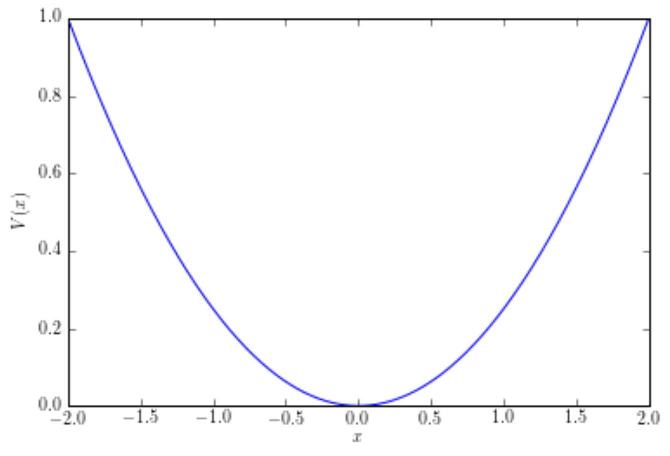


Set up the potential $V(x)$for our simulation. We use a simple harmonic oscillator.

```
In [18]: V = PotentialFactory().create_potential(params)

In [19]: V.potential
```

Out[19]: $\frac{1}{4}x^2$

```
In [20]: u = linspace(-2,2,1000)
         v = V.evaluate_at(u)[0]

In [21]: plot(u,v)
         xlabel(r"$x$")
         ylabel(r"$V(x)$")

Out[21]: Text(0,0.5,'$V(x)$')
```



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

```
In [22]: Psi.set_quadrature(None)
```

```
In [23]: Q = Psi.get_quadrature()
         Q
```

```
Out[23]: <WaveBlocks.HomogeneousQuadrature.HomogeneousQuadrature instance at 0xba1f18c>
```

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

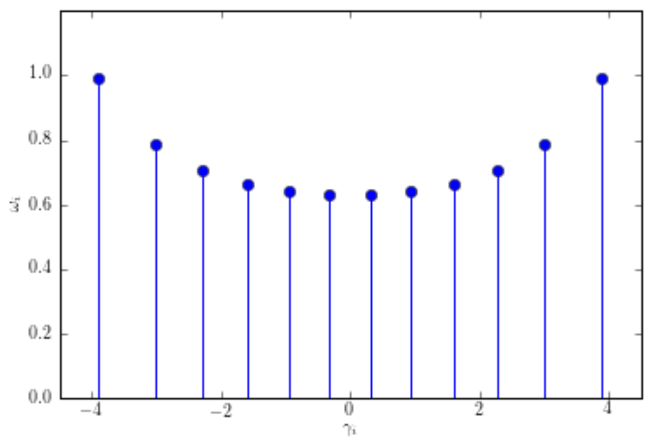
```
In [24]: QR = Q.get_qr()
```

And extract nodes and weights

```
In [25]: g = QR.get_nodes()
         w = QR.get_weights()
```

```
In [26]: 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[26]: Text(0,0.5,'$\\omega_i$')
```



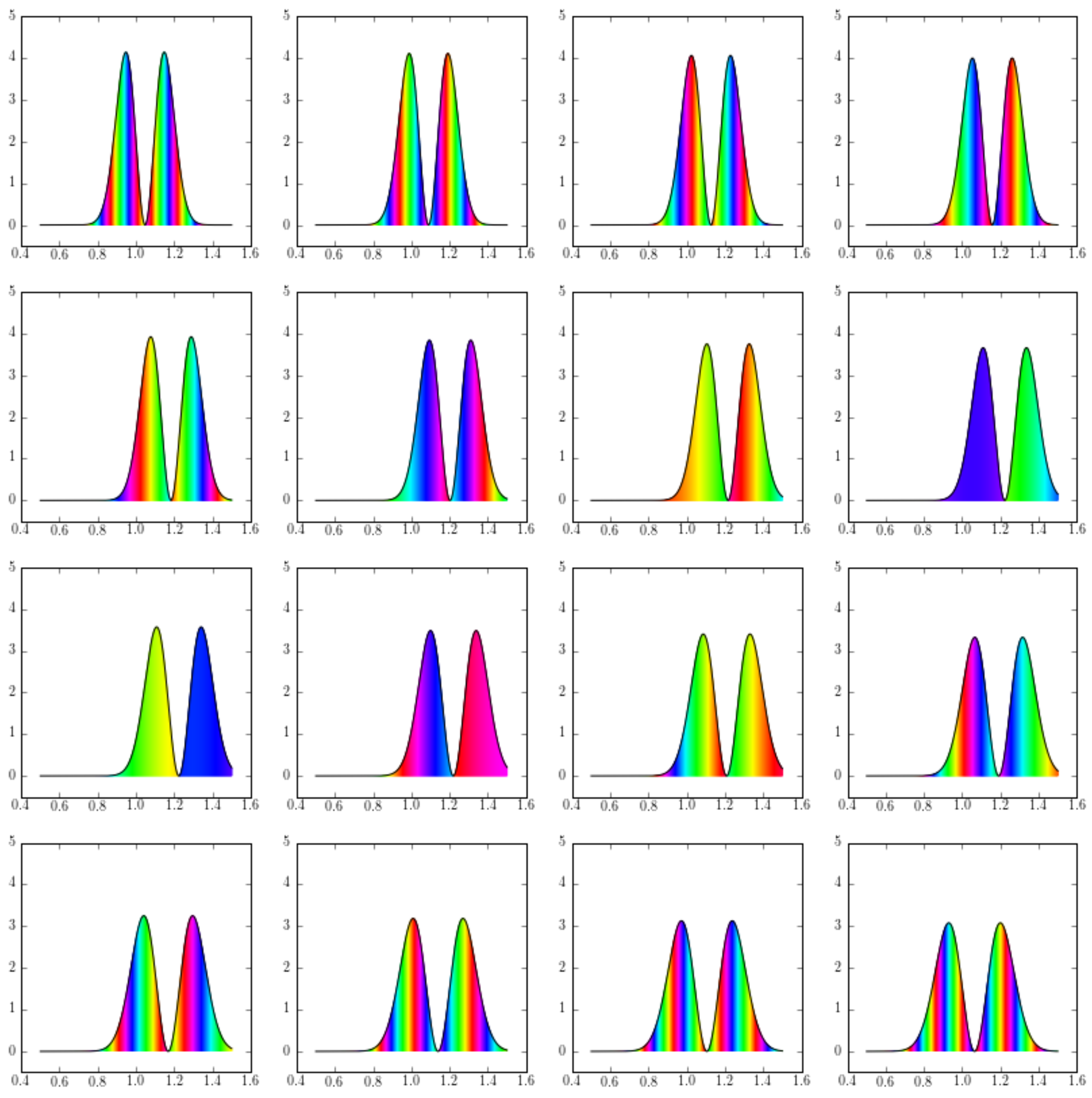
Now construct the time propagator

```
In [27]: P = HagedornPropagator(V, Psi, 0, params)
```

```
Warning: parameter 'matrix_exponential' not found, now trying global defaults!
Warning: parameter 'basis_size' not found, now trying global defaults!
Warning: parameter 'arnoldi_steps' not found, now trying global defaults!
```

Propagate for 16 timesteps and plot each state

```
In [28]: 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))
```

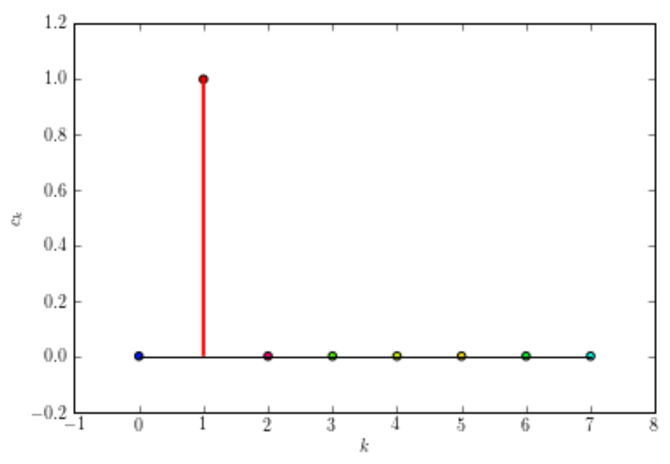


Look at the coefficients $c$ again

```
In [29]: cnew = Psi.get_coefficients(component=0)
         cnew = squeeze(cnew)
```

```
In [30]: figure(figsize=(6,4))
         stemcf(arange(cnew.shape[0]), angle(cnew), abs(cnew)**2)
         xlabel(r"$k$")
         ylabel(r"$c_k$")
```

```
Out[30]: Text(0,0.5,'$c_k$')
```



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

Now we go back in time ...

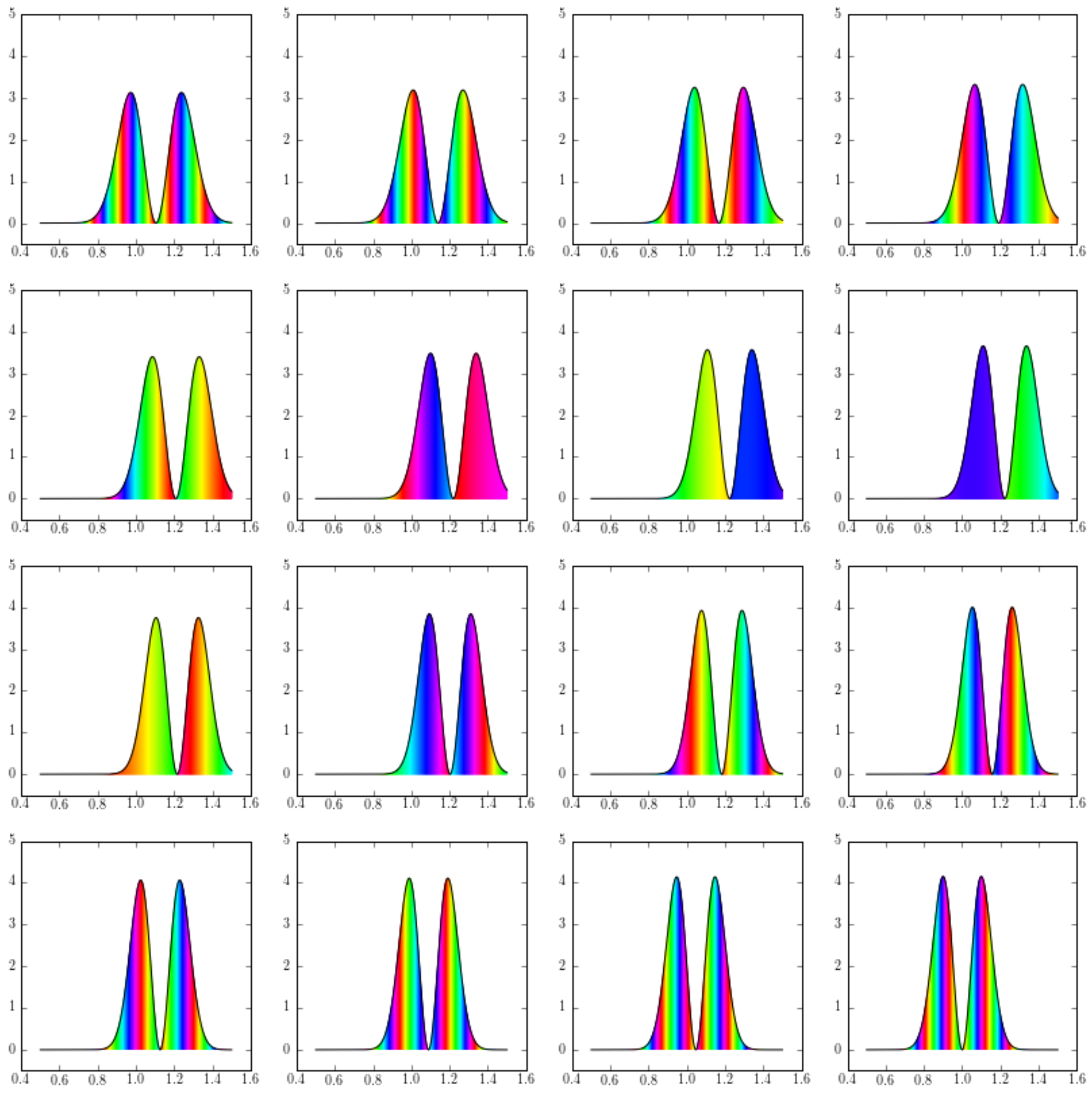
```
In [31]: params["dt"] *= -1
In [32]: print(params)
```

```
====================================
Parameters of the current simulation
------------------------------------
------------------------------------
All parameters provided
------------------------------------
  arnoldi_steps: 20
  dt: -0.1
  eps: 0.1
  matrix_exponential: arnoldi
  ncomponents: 1
  potential: quadratic
====================================
```

```
In [33]: Pinv = HagedornPropagator(V, Psi, 0, params)
```

```
Warning: parameter 'basis_size' not found, now trying global defaults!
```

```
In [34]: 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 [35]: Psi.get_parameters()
```

```
Out[35]: ((-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!