Interpolation with Radial Basis Functions

In [1]:

```
#keep
import numpy as np
import numpy.linalg as la
import matplotlib.pyplot as pt
%matplotlib inline
```

In [3]:

```
xx = np.linspace(-3, 3, 200)
```

In [5]:

```
np.random.seed(20)
centers = np.random.randn(10)*0.05 + np.linspace(-1.5, 1.5, 10)
centers = np.sort(centers)
centers
```

Out[5]:

```
array([-1.45580534, -1.15687342, -0.81545651, -0.6171631 , -0.220908
3 ,
        0.19465148,  0.54697347,  0.78440928,  1.19182151,  1.520320
72])
```

In [8]:

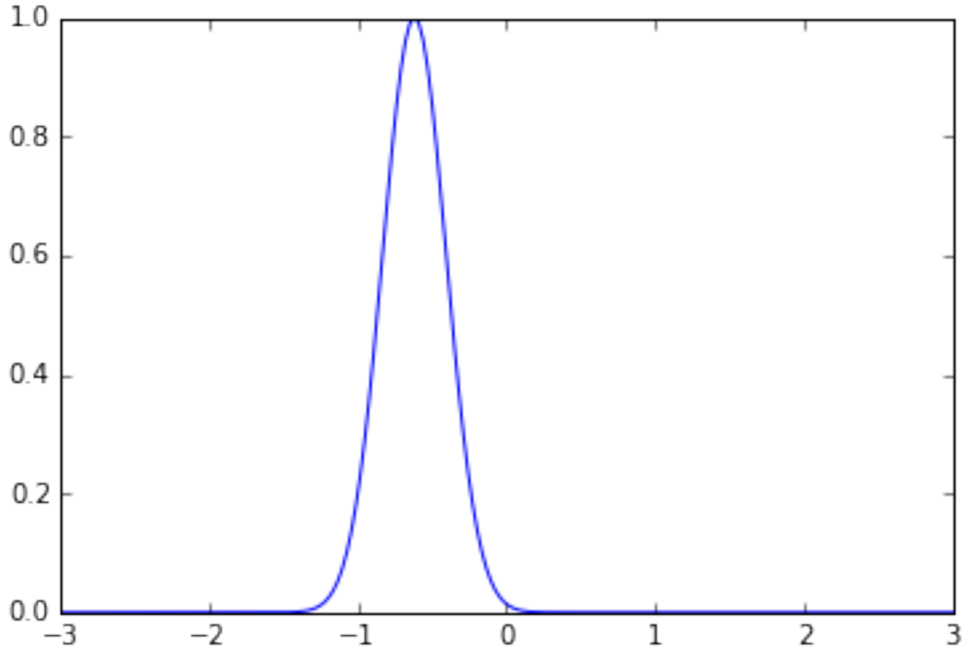
```
radius = 0.3

def radial_basis_function(x, i):
    return np.exp(-(x-centers[i])**2/radius**2)

pt.plot(xx, radial_basis_function(xx, 3))
```

Out[8]:

```
[<matplotlib.lines.Line2D at 0x106c2b128>]
```



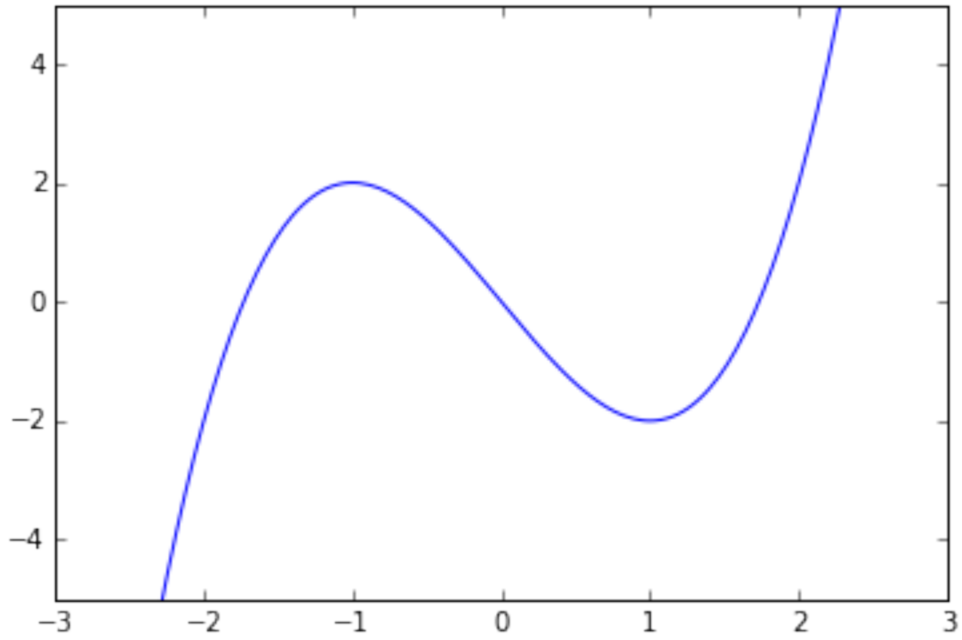
In [11]:

```
def f(x):
    return x**3 - 3*x

pt.plot(xx, f(xx))
pt.ylim([-5,5])
```

Out[11]:

```
(-5, 5)
```



Let's build a Vandermonde matrix at the centers:

In [15]:

```
nodes = centers

V = np.array([
    radial_basis_function(nodes, i)
    for i in range(len(centers))
    ]).T
```

Find the coefficients:

In [16]:

```
coeffs = la.solve(V, f(nodes))
```

Find the interpolant:

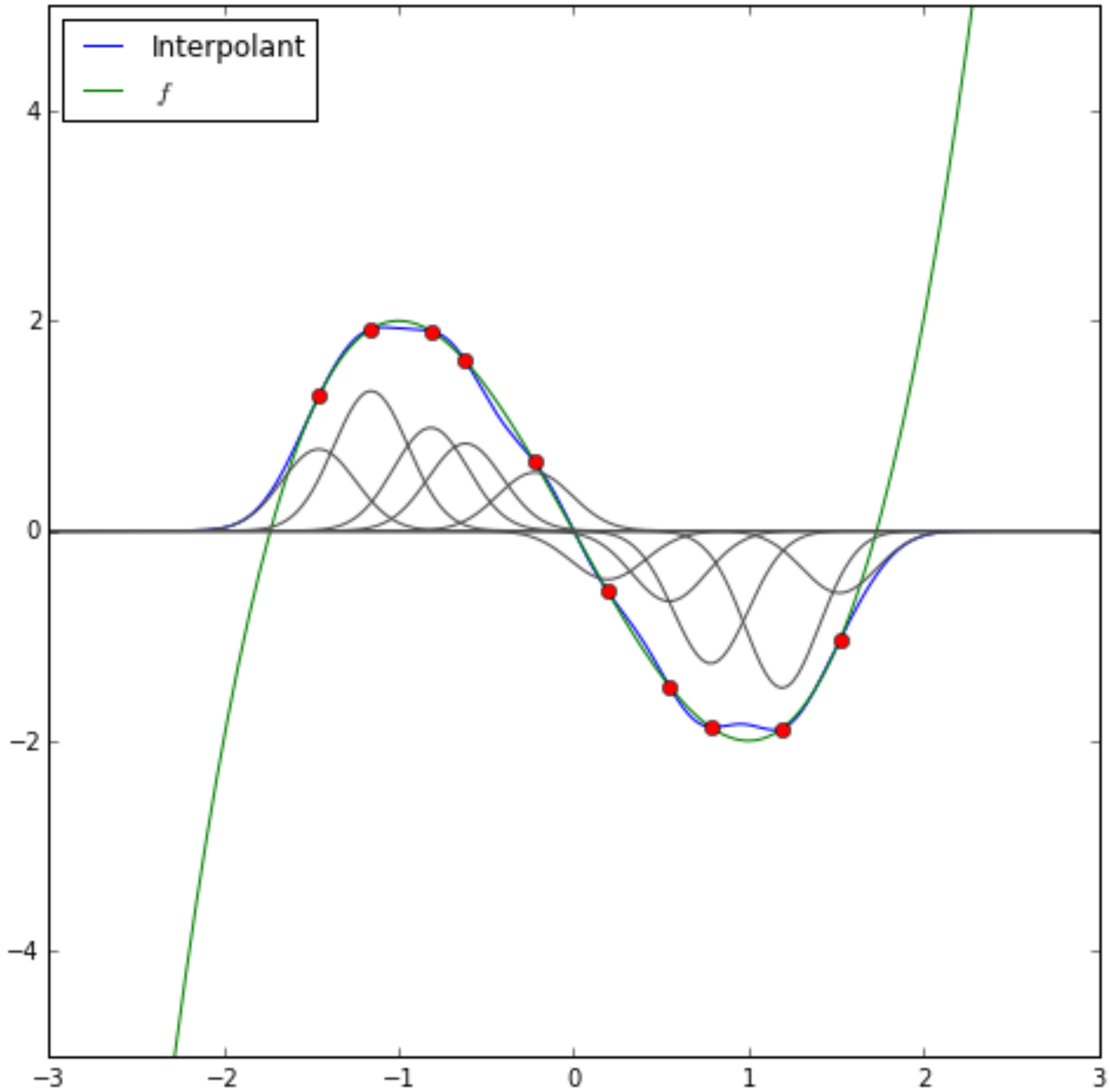
In [19]:

```
interpolant = 0
for i in range(len(centers)):
    interpolant += coeffs[i] * radial_basis_function(xx, i)

pt.figure(figsize=(8,8))
pt.ylim([-5,5])
pt.plot(xx, interpolant, label="Interpolant")
pt.plot(xx, f(xx), label="$f$")
####
## Look at the basis functions here
#for i in range(len(centers)):
#    pt.plot(xx, coeffs[i] * radial_basis_function(xx, i), '-', color='0.3')
pt.plot(centers, f(centers), "o")
pt.legend(loc="best")
```

Out[19]:

```
<matplotlib.legend.Legend at 0x107221fd0>
```



- Play around with the radius of the RBFs
- Play with node placement