In [1]:

```
%pylab inline
%config InlineBackend.figure_format = 'retina'
from ipywidgets import interact
```

```
Populating the interactive namespace from numpy and matplotlib
```

Polynomial interpolation

Lagrange polynomials

The Barycentric form is

$$p(x) = \frac{\sum_{j=0}^{n} \frac{w_j y_j}{(x - x_j)}}{\sum_{j=0}^{n} \frac{w_j}{(x - x_j)}}, \qquad w_j = \frac{1}{\prod_{\substack{i=0 \\ i \neq j}}^{n} (x_j - x_i)}$$

Example: interpolation error

Let $f(x) = \sin(x)$ and consider $n = 2$ points at $x_0 = 0$, $x_1 = \pi/4$, and $x_2 = \pi/2$. The polynomial is computed using the Barycentric formula below.

Recall the following error bound we covered in class. Let $f \in C^{n+1}[a, b]$, and let p be the interpolating polynomial for f on distinct nodes $x_0, \ldots, x_n \in [a, b]$. Then, for every $x \in [a, b]$, we have

$$|f(x) - p(x)| \leq \frac{\|f^{(n+1)}\|_\infty}{(n+1)!} \prod_{i=0}^{n} |x - x_i|.$$

Our function is infinitely differentiable, with $\|f'''\|_\infty = 1$. Hence,

$$|\sin(x) - p(x)| \leq \frac{1}{6} \prod_{i=0}^{n} |x - x_i|.$$

The final term in the error bound is

$$x(x - \pi/4)(x - \pi/2) \leq \frac{\pi^3}{96\sqrt{3}}.$$

This gives us the bound

$$\|\sin(x) - p(x)\|_\infty \leq \frac{\pi^3}{576\sqrt{3}}$$

Typesetting math: 100%

In [ ]:

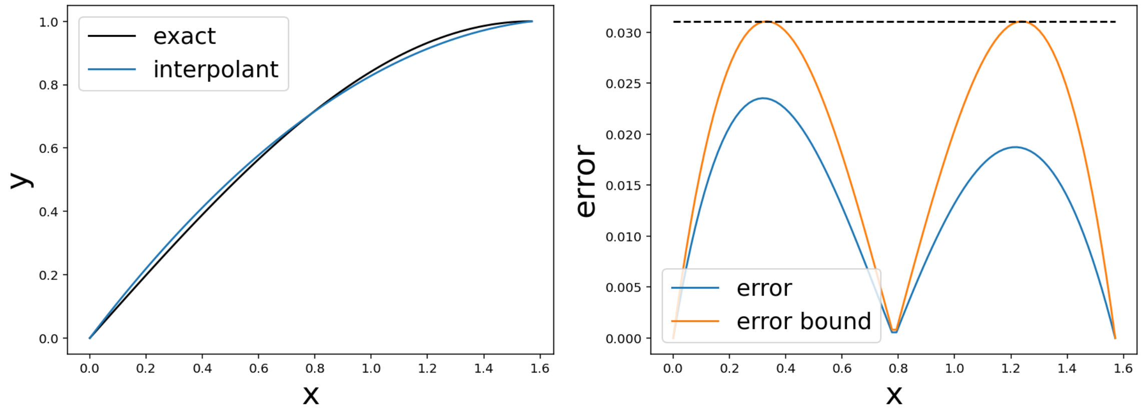
```
def f(x):
    return sin(x)
xnodes = array([0., pi/4., pi/2.])
ynodes = f(xnodes)
weights = array(
    [1/(xnodes[0] - xnodes[1])/(xnodes[0] - xnodes[2]),
    1/(xnodes[1] - xnodes[0])/(xnodes[1] - xnodes[2]),
    1/(xnodes[2] - xnodes[0])/(xnodes[2] - xnodes[1])])

x = linspace(0., pi/2., 100)
yexact = f(x)
bary_numer = zeros(x.size)
bary_denom = zeros(x.size)
replace_point_indices = []
y_replace = []
for j in arange(xnodes.size):
    b = zeros(x.size) ## create an empty array to put values into
    x_not_at_nodes = x[x != xnodes[j]] ## logical indexing: access only values in the
                                       ## array x that are not equal to xnodes[j], result
                                       ## is another array with (possibly) fewer elements
    b[x != xnodes[j]] = weights[j]/(x_not_at_nodes - xnodes[j])
    bary_numer += ynodes[j]*b
    bary_denom += b
    ## below is for the case where a value in the array x is equal to xnodes[j]
    index = where(x == xnodes[j])[0]
    replace_point_indices.extend(index)
    if index.size > 0:
        y_replace.append(ynodes[j])
yinterp = bary_numer/bary_denom
yinterp[replace_point_indices] = y_replace ## replace values where x == xnodes

fig = figure(1, [15, 5])
fig.add_subplot(121)
plot(x, yexact, 'k', label='exact')
plot(x, yinterp, label='interpolant')
xlabel('x', fontsize=24)
ylabel('y', fontsize=24)
legend(fontsize=18)

fig.add_subplot(122)
plot(x, absolute(yexact - yinterp), label='error')
plot(x, 1/6.*absolute(x*(x-pi/2)*(x-pi/4)),  label='error bound')
plot(x, pi**3/(576*sqrt(3))*ones_like(x), '--k')
xlabel('x', fontsize=24)
ylabel('error', fontsize=24);
legend(fontsize=18);
```

Typesetting math: 100%



Stability of interpolation in the $n \to \infty$ limit

Lebesgue constant

$$\Lambda_n = \max_{a \le x \le b} \sum_{j=0}^{n} |L_j(x)|.$$

Theorem: Stability

Let $x_0, x_1, \dots x_n$ be distinct nodes, and suppose $p(x)$ and $\hat{p}(x)$ are polynomials of degree at most n satisfying $p(x_j) = y_j$ and $\hat{p}(x_j) = \hat{y}_j$, $j = 0, 1, \dots, n$. If

$$|y_j - \hat{y}_j| \le \delta, \qquad j = 0, 1, \dots, n,$$

then

$$\|p - \hat{p}\|_\infty \le \Lambda_n \delta.$$

Example: Gaussian function using uniform nodes

$$f(x) = e^{-\frac{x^2}{2}}$$

The Lebesgue constant Λ_n for uniform nodes is bounded from below by

$$\Lambda_n \ge \frac{2^n}{4n^2}.$$

Notice this is exponentially growing!

In [2]:

```
from scipy.special import binom ## this imports a function to compute binomial coeficients

def bary_weights_uniform(n): ## homework problem 2 asks you to derive this formula
    j = arange(n+1)
    return (-1)**j*binom(n, j)

def f(x):
    return exp(-x**2/2)


@interact(n=(2, 100, 1))
def plot_fn(n=2):
    xnodes = linspace(-3., 3., n+1)
    ynodes = f(xnodes)
    w = bary_weights_uniform(n)

    x = linspace(-3., 3., 200) + 1e-5 # I am adding a small number so that values are
                                      # not the same as the nodes

    bary = w[:, None]/(x[None, :] - xnodes[:, None]) ## you should use loops in your
                                                     ## homework assignment
    yinterp = (bary*ynodes[:, None]).sum(axis=0)/bary.sum(axis=0)

    figure(1, [7, 5])
    plot(x, f(x), 'k', label='exact')
    plot(xnodes, ynodes, 'kx')
    plot(x, yinterp, label='interp')
    legend()
    show()
```

```
interactive(children=(IntSlider(value=2, description='n', min=2), Output()), _dom_cl
asses=('widget-interact',)…
```

In [3]:

```
bary_weights_uniform(3)
```

Out[3]:

```
array([ 1., -3.,  3., -1.])
```

Now with Chebyshev nodes

Chebyshev nodes are given by $x_j = -\cos(j\pi/n)$. The Lebesgue constant Λ_n for Chebyshev nodes is $$\Lambda_n = O(\log(n)).$$

Typesetting math: 100%

In [ ]:

```
from scipy.special import binom ## this imports a function to compute binomial coeficients

def bary_weights_cheb(n):
    j = arange(n+1)
    d = ones(n+1)
    d[0] = 0.5
    d[-1] = 0.5
    return (-1)**j*d

def f(x):
    return exp(-x**2/2.)


@interact(n=(2, 100, 1))
def plot_fn(n=2):
    xnodes = -3*cos(arange(n+1)*pi/n)
    ynodes = f(xnodes)
    w = bary_weights_cheb(n)

    x = linspace(-3., 3., 200) + 1e-5 # I am adding a small number so that values are
                                      # not the same as the nodes

    bary = w[:, None]/(x[None, :] - xnodes[:, None]) ## you should use loops in your
                                                     ## homework assignment
    yinterp = (bary*ynodes[:, None]).sum(axis=0)/bary.sum(axis=0)

    figure(1, [7, 5])
    plot(x, f(x), 'k', label='exact')
    plot(xnodes, ynodes, 'kx')
    plot(x, yinterp, label='interp')
    legend()
    show()
```

```
interactive(children=(IntSlider(value=2, description='n', min=2), Output()), _dom_cl
asses=('widget-interact',)…
```

Example: The Runge function is particularly sensitive to the Runge phenomenon

$$f(x) = \frac{1}{1 + 25x^2}$$ The nth derivative grows near the end points of the interval $[-1, 1]$ and is exponentially growing as $n\to\infty$. The exponential growth of the derivative and the Lebesgue constant (for uniform nodes) combine to make large errors at the end points.

Typesetting math: 100%

In [ ]:

```
from scipy.special import binom

def bary_weights_uniform(n):
    j = arange(n+1)
    return (-1)**j*binom(n, j)

def f_runge(x):
    return 1/(1 + 25.*x**2)


@interact(n=(2, 100, 1))
def plot_fn(n=2):
    xnodes = linspace(-1., 1., n+1)
    ynodes = f_runge(xnodes)
    w = bary_weights_uniform(n)

    x = linspace(-1., 1., 200) + 1e-5 # I am adding a small number so that values are
                                      # not the same as the nodes

    bary = w[:, None]/(x[None, :] - xnodes[:, None]) ## you should use loops in your
                                                      ## homework assignment
    yinterp = (bary*ynodes[:, None]).sum(axis=0)/bary.sum(axis=0)

    figure(1, [7, 5])
    plot(x, f_runge(x), 'k', label='exact')
    plot(xnodes, ynodes, 'kx')
    plot(x, yinterp, label='interp')
    legend(loc='upper right', fontsize=18)
    show()
```

```
interactive(children=(IntSlider(value=2, description='n', min=2), Output()), _dom_cl
asses=('widget-interact',)…
```

Let's try the same function with Chebyshev nodes instead

Chebyshev nodes $x_j = -\cos(j\pi/n)$

Typesetting math: 100%

In [ ]:

```
from scipy.special import binom

def bary_weights_cheb(n):
    j = arange(n+1)
    d = ones(n+1)
    d[0] = 0.5
    d[-1] = 0.5
    return (-1)**j*d

def f_runge(x):
    return 1/(1 + 25.*x**2)


@interact(n=(2, 100, 1))
def plot_fn(n=2):
    xnodes = -cos(arange(n+1)*pi/n)
    ynodes = f_runge(xnodes)
    w = bary_weights_cheb(n)

    x = linspace(-1., 1., 200) + 1e-5 # I am adding a small number so that values are
                                      # not the same as the nodes

    bary = w[:, None]/(x[None, :] - xnodes[:, None]) ## you should use loops in your
                                                     ## homework assignment
    yinterp = (bary*ynodes[:, None]).sum(axis=0)/bary.sum(axis=0)

    figure(1, [7, 5])
    plot(x, f_runge(x), 'k', label='exact')
    plot(xnodes, ynodes, 'kx')
    plot(x, yinterp, label='interp')
    legend(loc='upper right')
    show()
```

```
interactive(children=(IntSlider(value=2, description='n', min=2), Output()), _dom_cl
asses=('widget-interact',)…
```

In [ ]:

```
```

Typesetting math: 100%