Chapter 3
function evaluations

CSS 455 Winter 12

Two Important Series

Geometric Series, converges for all |x| < 1.

$$\frac{1}{1-x} = \sum_{k=0}^{\infty} x^k = 1 + x + x^2 + \cdots$$

Exponential series, converges for all x

$$e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$$

$$e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$$

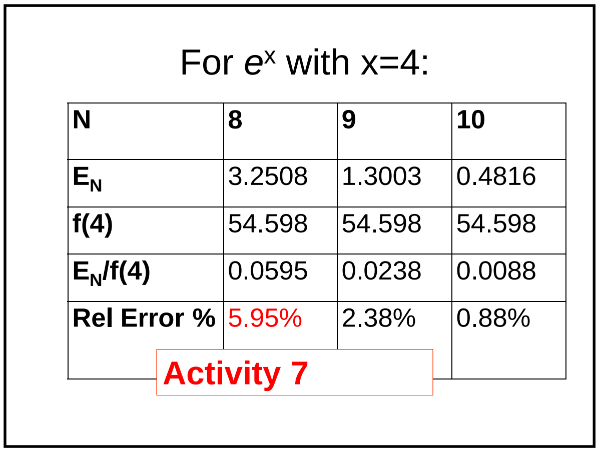
Recall, that for N-term approximation:

$$e^x \approx \sum_{k=0}^{N-1} \frac{x^k}{k!}$$

The Error bound goes like x^N:

$$E_N \le \frac{1}{(N-1)!}\left(\frac{x^N}{N-x}\right)$$

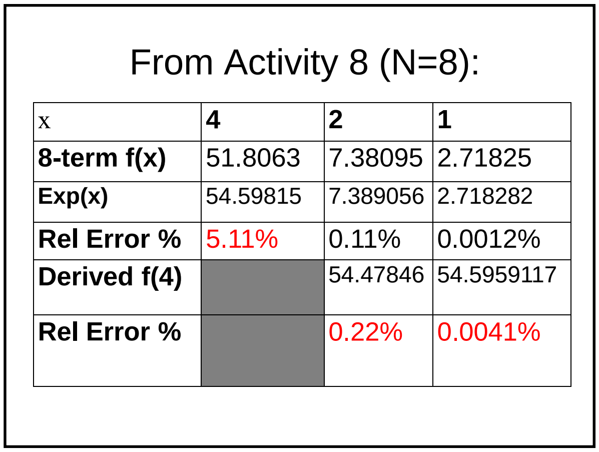
If x is doubled for an 8-term approximation, the error bound goes up by 2^8 (x 256)



For e^x with x=4:

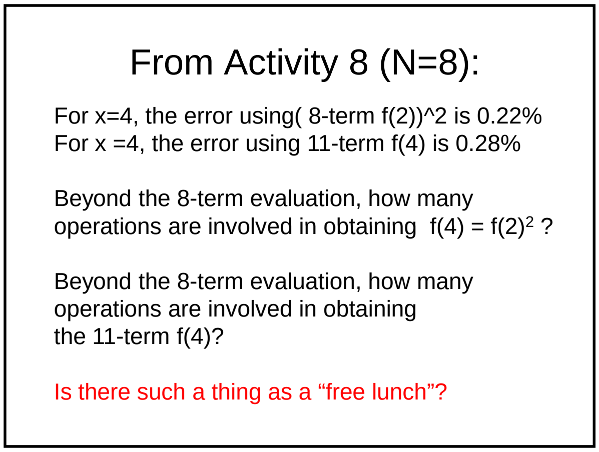
N	8	9	10
E_N	3.2508	1.3003	0.4816
f(4)	54.598	54.598	54.598
E_N/f(4)	0.0595	0.0238	0.0088
Rel Error %	5.95%	2.38%	0.88%

Activity 7



From Activity 8 (N=8):

x	4	2	1
8-term f(x)	51.8063	7.38095	2.71825
Exp(x)	54.59815	7.389056	2.718282
Rel Error %	5.11%	0.11%	0.0012%
Derived f(4)	▨	54.47846	54.5959117
Rel Error %	▨	0.22%	0.0041%



From Activity 8 (N=8):

For x=4, the error using(8-term f(2))^2 is 0.22%
For x =4, the error using 11-term f(4) is 0.28%

Beyond the 8-term evaluation, how many operations are involved in obtaining f(4) = f(2)^2 ?

Beyond the 8-term evaluation, how many operations are involved in obtaining the 11-term f(4)?

Is there such a thing as a “free lunch”?

Evaluate ln2

$$\int \frac{1}{x}\,dx = \ln(x)$$

$$\ln(1-x) = \int\left(\frac{1}{1-x}\right)d(1-x) = -\int \frac{1}{1-x}dx$$

$$= -\int\left(1 + x + x^2 + \ldots\right)dx = -\left(x + \frac{x^2}{2} + \frac{x^3}{3} + \cdots\right)$$

$$= -\sum_{k=0}^{\infty} \frac{x^{k+1}}{k+1}$$

The truncation error in this alternating series is less than the first omitted term. It will take 10^9 s to get the value correct to 9 nal places

? *X is large!*

Evaluate ln2

What size is x here?

$$\ln\left[\frac{1+x}{1-x}\right] = \ln(1+x) - \ln(1-x) \quad \text{for } (|x| < 1)$$

Evaluate ln2

$$\ln\left[\frac{1+x}{1-x}\right] = \left[2x + \frac{2x^3}{3} + \frac{2x^5}{5} + \cdots\right] = 2\left[x + \frac{x^3}{3} + \frac{x^5}{5} + \cdots\right]$$

- What would be the truncation error here after N terms (k=N-1)?

Evaluate ln2

$$E_N = \frac{2}{3}\sum_{k=N}^{\infty}\frac{1}{(2k+1)3^{2k}}$$

Consider this series and get an upper bound on it.
That will yield an upper bound on E$_{N.}$

Evaluate ln2

$$\left\{1+\frac{2N+1}{(2N+3)(3^2)}+\frac{2N+1}{(2N+5)(3^4)}+\cdots\right\}$$

Geometric Series: Eq 3.1, with x=(1/3)$^{2.}$

With that bound for the series:

$$E_N = \left(\frac{2}{3}\right)\left(\frac{1}{3^{2N}}\right)\left(\frac{1}{2N+1}\right)\left\{1+\frac{2N+1}{(2N+3)(3^2)}+\frac{2N+1}{(2N+5)(3^4)}+\cdots\right\}$$

$$E_N \le \left(\frac{2}{3}\right)\left(\frac{1}{3^{2N}}\right)\left(\frac{1}{2N+1}\right)\left(\frac{9}{8}\right)=\left(\frac{3}{4}\right)\left(\frac{1}{3^{2N}}\right)\left(\frac{1}{2N+1}\right)$$

With N=9, $E_N \le 1.02\text{x}10^{-10}$

Converges rapidly because x is now small and powers of it diminish rapidly.

for $x = 1/3$

$$\ln 2 = \frac{2}{3}\left[1+\frac{(1/3)^2}{3}+\frac{(1/3)^4}{5}+\cdots\right]\approx\frac{2}{3}\sum_{k=0}^{8}\frac{1}{(2k+1)3^{2k}}$$

Series for π

In Example 2, the equation arctan(1) = π/4 was used along with a series expansion for arctan(x):

$$\arctan(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \cdots$$

With x = 1, the series converges slowly (10^{16} terms) to yield π to double precision:

$$\frac{\pi}{4} = \arctan(1) = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots$$

How do we know the # of terms?

$$\frac{\pi}{4} = \arctan(1) = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots$$

In an alternating series, with each term smaller than the preceding one, the magnitude of the truncation error is bounded by the first term omitted.

So to get an error less than 10^{-3}, you include the first thousand terms.

To get double precision result (rel error 10^{-16}), you need approximately 10^{16} terms!!

Series for π

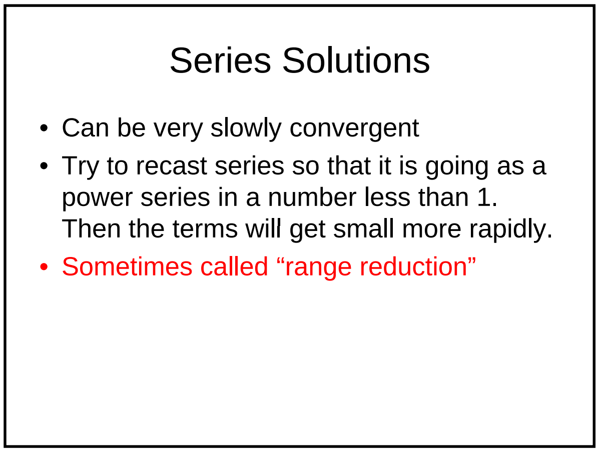
$$\arctan(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \cdots$$

With x = 1/√3 the series converges more rapidly.

In Example 5, it is recast as:

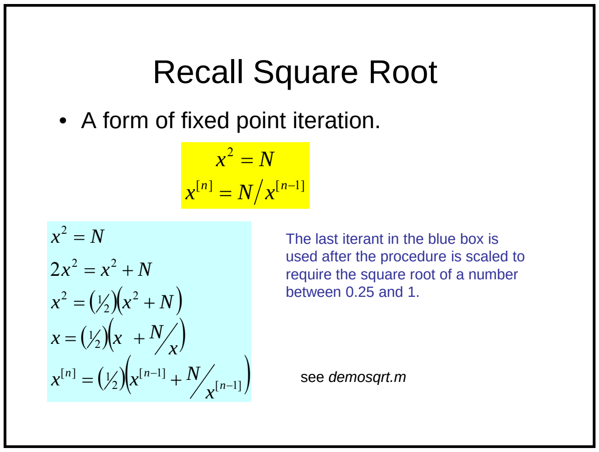
$$\arctan\left(\frac{1}{\sqrt{3}}\right) = \frac{\pi}{6}$$

Now the series has odd powers of {1/√3} and converges rapidly (15 terms for single precision)



Series Solutions

- Can be very slowly convergent
- Try to recast series so that it is going as a power series in a number less than 1. Then the terms will get small more rapidly.
- Sometimes called “range reduction”



Recall Square Root

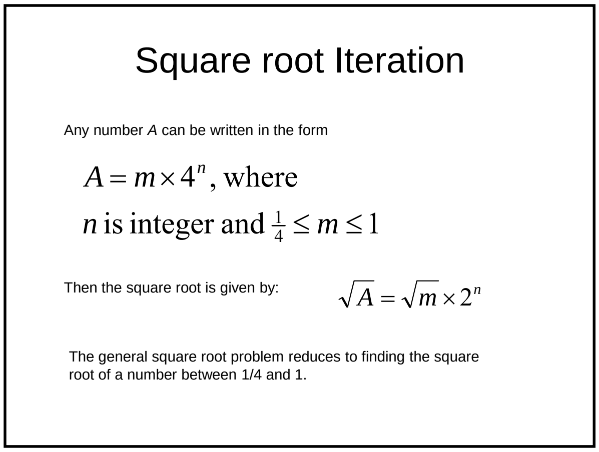
- A form of fixed point iteration.

$$x^2 = N$$
$$x^{[n]} = N / x^{[n-1]}$$

$$x^2 = N$$
$$2x^2 = x^2 + N$$
$$x^2 = (1/2)(x^2 + N)$$
$$x = (1/2)\left(x + N/x\right)$$
$$x^{[n]} = (1/2)\left(x^{[n-1]} + N/x^{[n-1]}\right)$$

The last iterant in the blue box is used after the procedure is scaled to require the square root of a number between 0.25 and 1.

see *demosqrt.m*



Square root Iteration

Any number *A* can be written in the form

$$A = m \times 4^n, \text{ where}$$
$$n \text{ is integer and } \tfrac{1}{4} \le m \le 1$$

Then the square root is given by: $\sqrt{A} = \sqrt{m} \times 2^n$

The general square root problem reduces to finding the square root of a number between 1/4 and 1.

Do not study further in book

- Cordic is general for many mathematical operations (multiply and divide) and many functional evaluations.
- Always involves only shifts and adds
- Can predict the number of terms needed
- Can be efficient in software, but is very well suited to hardware implementations.
- Read the rest of the chapter for examples only.

Polynomial Interpolation

Chapter 4 of Turner

CSS455 Winter 2012

Interpolation

- Given a data set (x_i,y_i), $i = 1,...,n$ seek a function $p(x)$, such that $p(x_i) = y_i$, $i = 1,..., n.$
- (x_i,y_i) could be tabular data or data obtained by evaluation of some underlying function.
- Specified data points are to be fit exactly.

Interpolation

- Interpolations are sometimes expected to give *reasonable* values between the data points as well as fitting them exactly.
- *Approximate* fits with smooth curves *near* the data points are *least squares problems.*

Why interpolation?

- Plotting smooth curve through data.
- Easy evaluation of a more difficult underlying mathematical function.
- Reading “between the lines” of a data table.
- Differentiation or integration of tabular data.

How to express *p(x)*

- Polynomial expansion. (3 term example)

$$p(x_i) = y_i = a + bx_i + cx_i^2$$

$$p(x_1) = y_1 = a + bx_1 + cx_1^2$$

$$p(x_2) = y_2 = a + bx_2 + cx_2^2$$

$$p(x_3) = y_3 = a + bx_3 + cx_3^2$$

Activity 8: (x,y)’s are known (a,b,c) are not.

Express *p(x)* more generally

- Linear combination of basis functions:

$$p(x) = \sum_{j=1}^{n} a_j \varphi_j(x)$$

Recall

$$p(x_i) = y_i = a + bx_i + cx_i^2$$

$$p(x_i) = y_i = a_1 + a_2 x_i + a_3 x_i^2$$

$$p(x_i) = y_i = a_1\varphi_1(x_i) + a_2\varphi_2(x_i) + a_3\varphi_3(x_i)$$

with φ_1(x) = 1; φ_2(x) = x ; and φ_3(x) = x^2

- If the number of data points and the number of basis functions are equal, we can solve a system of linear equations for the {a_j}:

$$p(x_i) = y_i = \sum_{j=1}^{n} a_j \varphi_j(x_i)\text{, for } i = 1,\ldots,n$$

First Equation

$$a_1\varphi_1(x_1) + a_2\varphi_2(x_1) + \cdots + a_n\varphi_n(x_1) = y_1$$

- First element of Matrix – Vector product

$$\begin{pmatrix} \varphi_1(x_1) & \varphi_2(x_1) & \cdots & \varphi_n(x_1) \\ \varphi_1(x_2) & \varphi_2(x_2) & \cdots & \varphi_n(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \varphi_1(x_n) & \varphi_2(x_n) & \cdots & \varphi_n(x_n) \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

First Equation

$$a_1\varphi_1(x_1)+a_2\varphi_2(x_1)+\cdots+a_n\varphi_n(x_1)=y_1$$

- Matrix – Vector product

$$\begin{pmatrix} \varphi_1(x_1) & \varphi_2(x_1) & \cdots & \varphi_n(x_1) \\ \cdots & \vdots & \cdots & \cdots \\ \vdots & \vdots & \ddots & \vdots \\ \cdots & \cdots & \cdots & \cdots \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} y_1 \\ \cdots \\ \vdots \\ \cdots \end{pmatrix}$$

Second Equation

$$a_1\varphi_1(x_2)+a_2\varphi_2(x_2)+\cdots+a_n\varphi_n(x_2)=y_2$$

- Second element of Matrix – Vector product

$$\begin{pmatrix} \varphi_1(x_1) & \varphi_2(x_1) & \cdots & \varphi_n(x_1) \\ \varphi_1(x_2) & \varphi_2(x_2) & \cdots & \varphi_n(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \varphi_1(x_n) & \varphi_2(x_n) & \cdots & \varphi_n(x_n) \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

Second Equation

$$a_1\varphi_1(x_2)+a_2\varphi_2(x_2)+\cdots+a_n\varphi_n(x_2)=y_2$$

- Second element of Matrix – Vector product

$$\begin{pmatrix} \cdots & \cdots & \cdots & \cdots \\ \varphi_1(x_2) & \varphi_2(x_2) & \cdots & \varphi_n(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \cdots & \cdots & \cdots & \cdots \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} \cdots \\ y_2 \\ \vdots \\ \cdots \end{pmatrix}$$

What form for the basis functions?

- Polynomials in *x*.
- Piecewise polynomials. Sections of data are fit and then the fits are pieced together
- Trigonometric functions (Fourier): *cos(jx), sin(jx), etc.*
- Straight lines between neighbors
- Exponentials

$$e^{\pm jx^2}, e^{\pm jx},$$

Polynomials for basis functions

- The fit of an (*n-1*) degree polynomial to *n* data points **is unique.** The resulting polynomial does not depend upon the form of the polynomial basis functions.
- The numeric conditioning of the problem **depends strongly** on the choice of polynomial basis. The problem can be very poorly conditioned for high degree polynomials.

$$\{1, x, x^2\}$$

$$\{5, x-1, 2x^2+x\}$$

Monomials - Vandermonde

- Basis set is {1, *x*, x^2, x^3,…, x^{n-1}} for interpolation of *n* data points.
- The system of equations is:

$$\begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{n-1} \\ 1 & x_3 & x_3^2 & \cdots & x_3^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^{n-1} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix}$$

Monomials - Vandermonde

$$\begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{n-1} \\ 1 & x_3 & x_3^2 & \cdots & x_3^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^{n-1} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix}$$

- For this case, the matrix is full and provides difficult numeric challenges in many cases.
- Approach taken by built-in *polyfit.*

Activity 9: Part I

Activity 9: part I

- Find the polynomial interpolant using monomials for the following data set:

i	0	1	2
x_i	-2	0	1
y_i or $f(x_i)$	-27	-1	0

Activity 9: part I

- Find the polynomial interpolant using monomials for the following data set:

p(x) = -1 + 5x -4x^2
p(-1) = -1 -5 -4 = -10

i	0	1	2
x_i	-2	0	1
y_i or $f(x_i)$	-27	-1	0

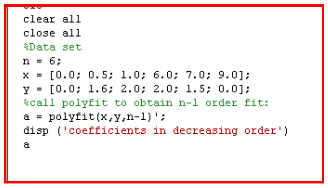
Monomials - Vandermonde

- Consider the six experimental points given below:

x	0.0	0.5	1.0	6.0	7.0	9.0
y	0.0	1.6	2.0	2.0	1.5	0.0

Set up and solve using *polyfit*

Matlab Solution (*polydemo.m)*



```
clc
clear all
close all
%Data set
n = 6;
x = [0.0; 0.5; 1.0; 6.0; 7.0; 9.0];
y = [0.0; 1.6; 2.0; 2.0; 1.5; 0.0];
%call polyfit to obtain n-1 order fit:
a = polyfit(x,y,n-1)';
disp ('coefficients in decreasing order')
a
```

```
a =
  0.0057
 -0.1348

  1.1208
 -3.8559
  4.8643
 -0.0000
```

x^5 coefficient

x^3 coefficient

x^0 coefficient

Evaluation of Interpolant

$$f(x) = a_1 + a_2 x + a_3 x^2 + ... + a_n x^{n-1}$$

- For each value of *x*, must evaluate a set of powers of *x*. This should be done efficiently.
- One idea: for a particular value of *x* accumulate the polynomial from term to term by multiplication, avoiding the exponentiations.
- Note: *polyfit* returned the coefficients in reverse order: a_1 *is the coefficient of* x^{n-1}*.*

Evaluation of Interpolant

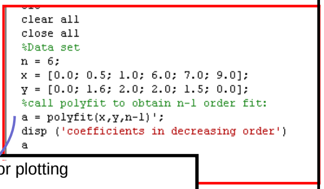
$$f(x) = a_1 + a_2 x + a_3 x^2 + ... + a_n x^{n-1}$$

- Note: *polyfit* returned the coefficients in reverse order: a_1 *is the coefficient of* x^{n-1}*.*

$$f(x) = a_n + a_{n-1} x + a_{n-2} x^2 + ... + a_1 x^{n-1}$$

Polyval is a built-in function that takes the coefficients in the reverse order provided by *polyfit* and a vector x of input values and returns a vector of y-values obtained from the polynomial interpolant.

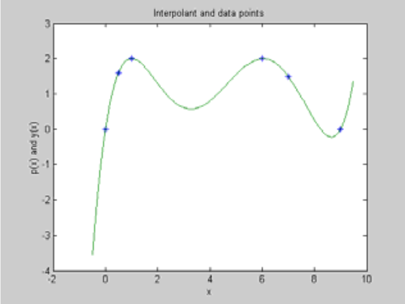
$$f(x) = a_n + a_{n-1} x + a_{n-2} x^2 + ... + a_1 x^{n-1}$$



```
clc
clear all
close all
%Data set
n = 6;
x = [0.0; 0.5; 1.0; 6.0; 7.0; 9.0];
y = [0.0; 1.6; 2.0; 2.0; 1.5; 0.0];
%call polyfit to obtain n-1 order fit:
a = polyfit(x,y,n-1)';
disp ('coefficients in decreasing order')
a
```

```
% evaluation of interpolant for plotting
xlo=min(x)-0.5;
xhi=max(x)+0.5;
x2 = linspace(xlo,xhi,120)';
f3 = polyval(a,x2);
figure
plot(x,y,'*',x2,f3)
title('Interpolant and data points using polyval')
xlabel ('X')
ylabel ('p(x) and y(x)')
```

$$f(x) = a_n + a_{n-1}x + a_{n-2}x^2 + \ldots + a_1 x^{n-1}$$

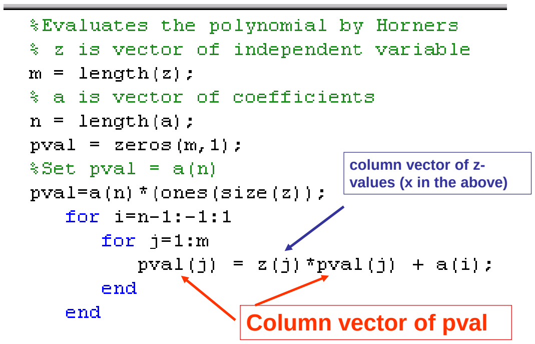


Horner’s Rule

$$f(x) = a_1 + a_2 x + a_3 x^2 + a_4 x^3 + a_5 x^4 + a_6 x^5$$
$$f(x) = ((((a_6 x + a_5)x + a_4)x + a_3)x + a_2)x + a_1$$

- The first form requires nine multiplications and five additions.
- The nested form requires five multiplications and five additions.

$$f(x) = ((((a_6 x + a_5)x + a_4)x + a_3)x + a_2)x + a_1$$



```
%Evaluates the polynomial by Horners
% z is vector of independent variable
m = length(z);
% a is vector of coefficients
n = length(a);
pval = zeros(m,1);
%Set pval = a(n)
pval=a(n)*(ones(size(z));
   for i=n-1:-1:1
      for j=1:m
         pval(j) = z(j)*pval(j) + a(i);
      end
   end
```

column vector of z-values (x in the above)

Column vector of pval

$$f(x) = ((((a_6 x + a_5)x + a_4)x + a_3)x + a_2)x + a_1$$

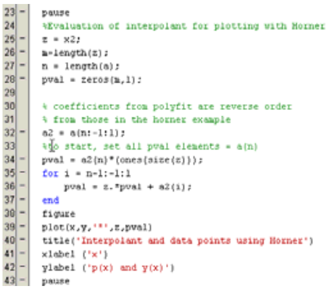
```
%Evaluates the polynomial by Horners
% z is vector of independent variable
m = length(z);
% a is vector of coefficients
n = length(a);
pval = zeros(m,1);
%Set pval = a(n)
pval=a(n)*(ones(size(z));
   for i=n-1:-1:1
      pval = z.*pval + a(i);
   end
```

column vector of z-values

Pointwise multiply

Column vector of pval

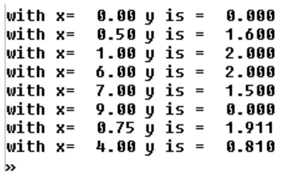
$$f(x) = a_1 + a_2 x + a_3 x^2 + a_4 x^3 + a_5 x^4 + a_6 x^5$$
$$f(x) = ((((a_6 x + a_5)x + a_4)x + a_3)x + a_2)x + a_1$$



```
pause
%Evaluation of interpolant for plotting with Horner
z = x2;
m=length(z);
n = length(a);
pval = zeros(m,1);

% coefficients from polyfit are reverse order
% from those in the horner example
a2 = a(n:-1:1);
%To start, set all pval elements = a(n)
pval = a2(n)*(ones(size(z)));
for i = n-1:-1:1
    pval = z.*pval + a2(i);
end
figure
plot(x,y,'*',z,pval)
title('Interpolant and data points using Horner')
xlabel ('x')
ylabel ('p(x) and y(x)')
pause
```

x	0.0	0.5	1.0	6.0	7.0	9.0
y	0.0	1.6	2.0	2.0	1.5	0.0



```
with x=  0.00 y is =  0.000
with x=  0.50 y is =  1.600
with x=  1.00 y is =  2.000
with x=  6.00 y is =  2.000
with x=  7.00 y is =  1.500
with x=  9.00 y is =  0.000
with x=  0.75 y is =  1.911
with x=  4.00 y is =  0.810
>>
```

- Interpolant reproduces original data exactly.
- Also evaluated at two additional points.

Monomial Basis Functions

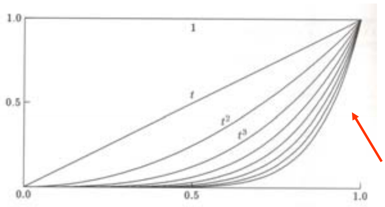


FIGURE 7.1
Monomial basis functions.

Functional behavior can be hard to describe in this region, since the basis functions are so similar.

Other Polynomial Interpolants?

- Recall, the requirement is: $p(x_i) = y_i = \sum_{j=0}^{n} a_j \varphi_j(x_i)$, for $i = 0,\ldots,n$

•The set of $\{x_i, y_i\}$ is given.

•The set $\{\varphi_j\}$must be identified

•The set of $\{a_j\}$are found to define the particular interpolant

•Consider:

$$p(x_i) = y_i = \sum_{j=0}^{n} a_j l_j(x_i)\text{, for } i = 0,\ldots,n$$

$$l_j(x_k) = \delta_{jk} = \begin{cases} 1 \text{ if } \mathrm{j} = \mathrm{k} \\ 0 \text{ if } \mathrm{j} \neq \mathrm{k} \end{cases}$$

$$p(x) = \sum_{j=0}^{n} a_j l_j(x)\text{, for all x}$$

What form the A matrix?

$$\begin{pmatrix} \varphi_1(x_1) & \varphi_2(x_1) & \cdots & \varphi_n(x_1) \\ \varphi_1(x_2) & \varphi_2(x_2) & \cdots & \varphi_n(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \varphi_1(x_n) & \varphi_2(x_n) & \cdots & \varphi_n(x_n) \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

$$l_j(x_k) = \delta_{jk} = \begin{cases} 1 \text{ if } \mathrm{j} = \mathrm{k} \\ 0 \text{ if } \mathrm{j} \neq \mathrm{k} \end{cases}$$

$$p(x_i) = y_i = \sum_{j=0}^{n} a_j l_j(x_i)\text{, for } i = 0,\ldots,n$$

$$l_j(x_k) = \delta_{jk} = \begin{cases} 1 \text{ if } j = k \\ 0 \text{ if } j \neq k \end{cases}$$

- What form for the $\{l_j(x_i)\}$?
- For the case, $n = 4$, $j = 2$, try*:*

$$l_2(x) = \frac{(x-x_0)(x-x_1)(x-x_3)(x-x_4)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)(x_2-x_4)}$$

If $x = x_2$, $l_2(x) = 1$

If $x = x_3$, $l_2(x_3) = 0$

This definition meets the requirement.

Notice the denominator has only x_2 factors and the numerator has *none.*

$$l_j(x) = \prod_{k \neq j} \frac{(x - x_k)}{(x_j - x_k)}$$

Example: Find the Lagrange Interpolating Polynomial for:

i	0	1	2
x_i	-2	0	1
y_i or $f(x_i)$	-27	-1	0

$$l_0(x) = \frac{(x-0)(x-1)}{(-2-0)(-2-1)} = \frac{(x)(x-1)}{6}$$

$$l_1(x) = \frac{(x-(-2))(x-1)}{(0-(-2))(0-1)} = \frac{(x+2)(x-1)}{-2}$$

$$l_2(x) = \frac{(x-(-2))(x-0)}{(1-(-2))(1-0)} = \frac{(x+2)(x)}{3}$$

$$l_j(x) = \prod_{k \neq j} \frac{(x - x_k)}{(x_j - x_k)}$$

Activity 9: Part II

Example: Find the Lagrange Interpolating Polynomial for:

i	0	1	2
x_i	-2	0	1
y_i or $f(x_i)$	-27	-1	

Recall, the A matrix was a unit matrix here.

$$l_0(x) = \frac{(x)(x-1)}{6}$$

$$l_1(x) = \frac{(x+2)(x-1)}{-2}$$

$$l_2(x) = \frac{(x+2)(x)}{3}$$

$$p(x) = \sum_{j=0}^{n} y_j l_j(x)$$

$$= (-27)\left(\frac{x(x-1)}{6}\right) + (-1)\left(\frac{(x+2)(x-1)}{-2}\right) + (0)\left(\frac{(x+2)x}{3}\right)$$

$$p(-1) = -9 - 1 + 0 = -10$$

$$p(x) = -4x^2 + 5x - 1$$

The Same Polynomial!!

Lagrangian Functions

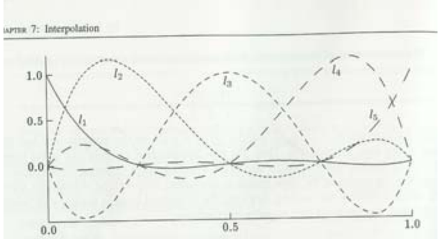


FIGURE 7.2
Lagrange basis functions.

Newton:

Each basis function is of *(j-1)* power, but it is not a simple monomial of that power.

$$\varphi_j(x) = \prod_{k=1}^{j-1}(x - x_k)$$

$$\varphi_1(x) = 1$$

$$\varphi_2(x) = (x - x_1)$$

$$\varphi_3(x) = (x - x_1)(x - x_2)$$

$$\varphi_n(x) = (x - x_1)(x - x_2)\cdots(x - x_{n-1})$$

The x_i values are the independent data set variables.

Heath Fig 7.3

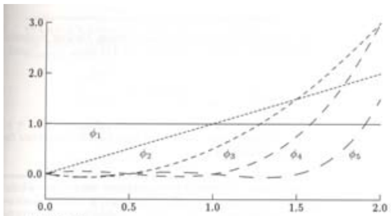
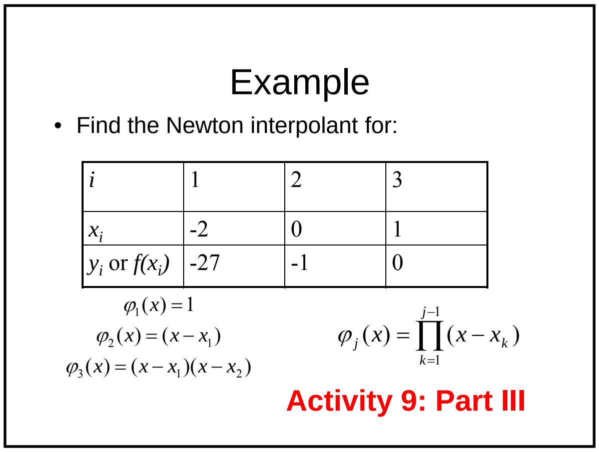


FIGURE 7.3
Newton basis functions.

The five data points (x_1,…,x_5) are evenly spaced here.



Example

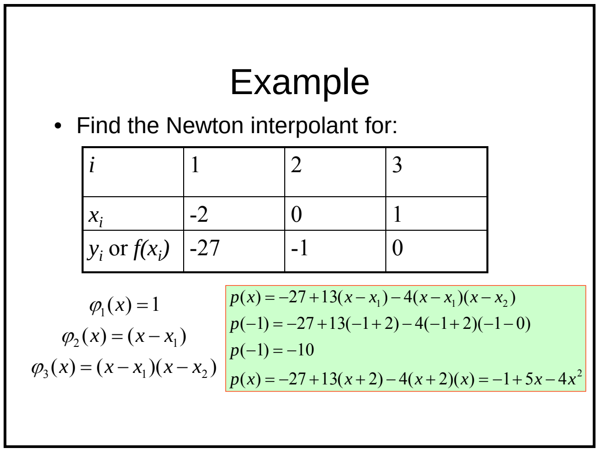
- Find the Newton interpolant for:

i	1	2	3
x_i	-2	0	1
y_i or $f(x_i)$	-27	-1	0

$$\varphi_1(x) = 1$$
$$\varphi_2(x) = (x - x_1)$$
$$\varphi_3(x) = (x - x_1)(x - x_2)$$

$$\varphi_j(x) = \prod_{k=1}^{j-1}(x - x_k)$$

Activity 9: Part III



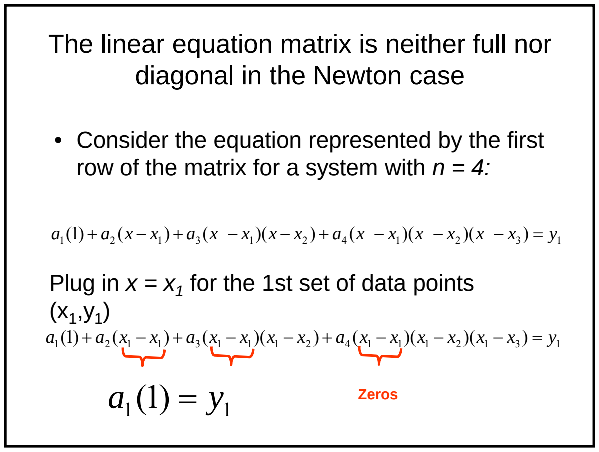
Example

- Find the Newton interpolant for:

i	1	2	3
x_i	-2	0	1
y_i or $f(x_i)$	-27	-1	0

$$\varphi_1(x) = 1$$
$$\varphi_2(x) = (x - x_1)$$
$$\varphi_3(x) = (x - x_1)(x - x_2)$$

$$p(x) = -27 + 13(x - x_1) - 4(x - x_1)(x - x_2)$$
$$p(-1) = -27 + 13(-1 + 2) - 4(-1 + 2)(-1 - 0)$$
$$p(-1) = -10$$
$$p(x) = -27 + 13(x + 2) - 4(x + 2)(x) = -1 + 5x - 4x^2$$



The linear equation matrix is neither full nor diagonal in the Newton case

- Consider the equation represented by the first row of the matrix for a system with *n = 4:*

$$a_1(1) + a_2(x - x_1) + a_3(x - x_1)(x - x_2) + a_4(x - x_1)(x - x_2)(x - x_3) = y_1$$

Plug in *x* = *x₁* for the 1st set of data points (x_1,y_1)

$$a_1(1) + a_2\underbrace{(x_1 - x_1)}_{} + a_3\underbrace{(x_1 - x_1)}_{}(x_1 - x_2) + a_4\underbrace{(x_1 - x_1)}_{}(x_1 - x_2)(x_1 - x_3) = y_1$$

$$a_1(1) = y_1$$

Zeros

The linear equation matrix is neither full nor diagonal

- Consider the equation represented by the second row of the matrix for a system with *n = 4:*

$$a_1(1)+a_2(x-x_1)+a_3(x-x_1)(x-x_2)+a_4(x-x_1)(x-x_2)(x-x_3)=y_2$$

Plug in $x = x_2$ for the 2nd set of data points (x_2,y_2)

$$a_1(1)+a_2(x_2-x_1)+a_3(x_2-x_1)\underbrace{(x_2-x_2)}+a_4(x_2-x_1)\underbrace{(x_2-x_2)}(x_2-x_3)=y_2$$

$$a_1(1)+a_2(x_2-x_1)=y_2$$

Zeros

The linear equation matrix is neither full nor diagonal

- Consider the equation represented by the third row of the matrix for a system with *n = 4:*

$$a_1(1)+a_2(x-x_1)+a_3(x-x_1)(x-x_2)+a_4(x-x_1)(x-x_2)(x-x_3)=y_3$$

Plug in $x = x_3$ for the third set of data points (x_3,y_3)

$$a_1(1)+a_2(x_3-x_1)+a_3(x_3-x_1)(x_3-x_2)+a_4(x_3-x_1)(x_3-x_2)\underbrace{(x_3-x_3)}=y_3$$

$$a_1(1)+a_2(x_3-x_1)+a_3(x_3-x_1)(x_3-x_2)=y_3$$

Zeros

- The problem is better conditioned because the magnitudes of individual terms are similar due to the shifting.

$$a_1(1)+a_2(x-x_1)+a_3(x-x_1)(x-x_2)+a_4(x-x_1)(x-x_2)(x-x_3)=y_3$$

•The linear equations matrix is lower triangular rather than full.

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 1 & x_2-x_1 & 0 & \cdots & 0 \\ 1 & (x_3-x_1) & (x_3-x_1)(x_3-x_2) & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & (x_n-x_1) & (x_n-x_1)(x_n-x_2) & \cdots & (x_n-x_1)(x_n-x_2)\ldots(x_n-x_{n-1}) \end{bmatrix}$$

- The problem is better conditioned because the magnitudes of individual terms are similar due to the shifting.

$$a_1(1)+a_2(x-x_1)+a_3(x-x_1)(x-x_2)+a_4(x-x_1)(x-x_2)(x-x_3)=y_3$$

•The linear equations matrix is lower triangular rather than full.

- The solution of the system takes fewer operations because the equations are simpler. (n^2 instead of n^3)
- The interpolant can also be evaluated most efficiently by a nested algorithm (Horner)

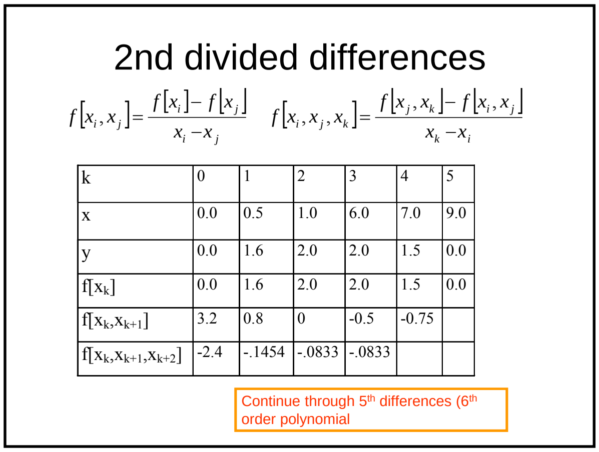
Newton Method

- Newton coefficients can be solved by divided differences
- **Activity 10**

$$\begin{aligned}&a_1(1)+a_2(x-x_0)+a_3(x-x_0)(x-x_1)+\\&+a_4(x-x_0)(x-x_1)(x-x_2)+\cdots\\&+a_6(x-x_0)(x-x_1)(x-x_2)(x-x_3)(x-x_4)=y\end{aligned}$$

In this section, the author numbers data points from k=0 to k=N. Above, N=5.

k	0	1	2	3	4	5
x	0.0	0.5	1.0	6.0	7.0	9.0
y	0.0	1.6	2.0	2.0	1.5	0.0

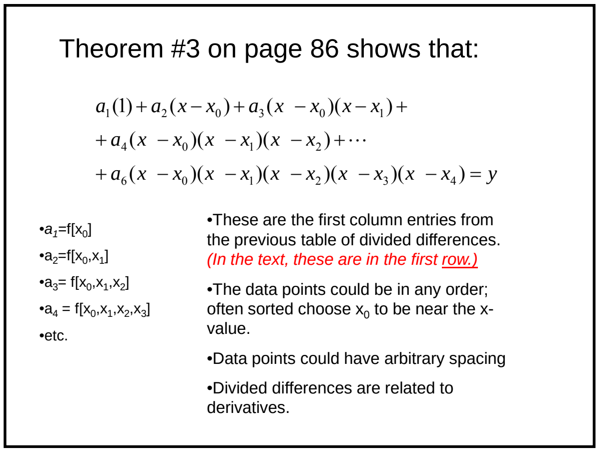


2nd divided differences

$$f[x_i,x_j]=\frac{f[x_i]-f[x_j]}{x_i-x_j} \qquad f[x_i,x_j,x_k]=\frac{f[x_j,x_k]-f[x_i,x_j]}{x_k-x_i}$$

k	0	1	2	3	4	5
x	0.0	0.5	1.0	6.0	7.0	9.0
y	0.0	1.6	2.0	2.0	1.5	0.0
$f[x_k]$	0.0	1.6	2.0	2.0	1.5	0.0
$f[x_k,x_{k+1}]$	3.2	0.8	0	-0.5	-0.75	
$f[x_k,x_{k+1},x_{k+2}]$	-2.4	-.1454	-.0833	-.0833		

Continue through 5th differences (6th order polynomial

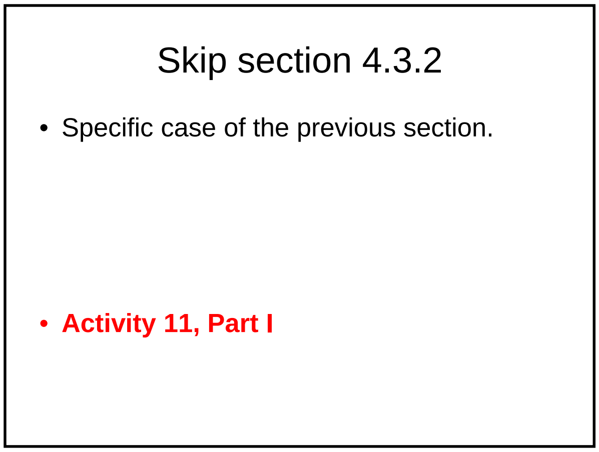


Theorem #3 on page 86 shows that:

$$a_1(1)+a_2(x-x_0)+a_3(x-x_0)(x-x_1)+$$
$$+a_4(x-x_0)(x-x_1)(x-x_2)+\cdots$$
$$+a_6(x-x_0)(x-x_1)(x-x_2)(x-x_3)(x-x_4)=y$$

- a_1=f[x_0]
- a_2=f[x_0,x_1]
- a_3= f[x_0,x_1,x_2]
- a_4 = f[x_0,x_1,x_2,x_3]
- etc.

- These are the first column entries from the previous table of divided differences. *(In the text, these are in the first row.)*
- The data points could be in any order; often sorted choose x_0 to be near the x-value.
- Data points could have arbitrary spacing
- Divided differences are related to derivatives.



Skip section 4.3.2

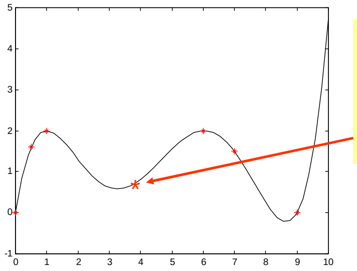
- Specific case of the previous section.

- **Activity 11, Part I**

Placement of data points.

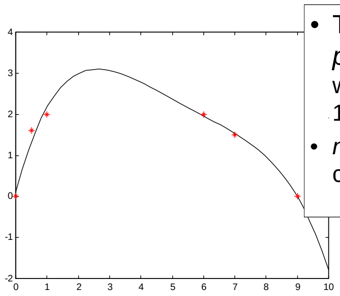
- Evenly spaced data points can be used to speed up fitting and evaluation. See discussion of finite difference method in text for evenly spaced points.
- Unevenly spaced points can sometimes improve description.

Plot Interpolant f(x) vs x.



What do you think about the value of f(4.0)?

"Built in" Matlab Functions



- Try *polyfit/polyval* with *n* reduced by 1.
- *n* = length(x)-1 call polyfit(x,y,n-1)

Which do you like better?

- Simple *monomials* x^{j-1} can be improved as a basis by shifting and scaling

$$\left(\frac{x-c}{d}\right)^{j-1}$$

Example

- Census data for nine years.
- To be fit with 8th order polynomial with shifting and scaling.

Data =

1900	76212168
1910	92228496
1920	106021537
1930	123202624
1940	132164569
1950	151325798
1960	179323175
1970	203302031
1980	226542199

Shifting and Scaling

- Four polynomials

$\varphi_j(t) = t^{j-1}$

$\varphi_j(t) = (t-1900)^{j-1}$

$\varphi_j(t) = (t-1940)^{j-1}$

$\varphi_j(t) = [(t-1940)/40]^{j-1}$

1) For t^j

Cond1 = Inf

2) For (t-1900)^j

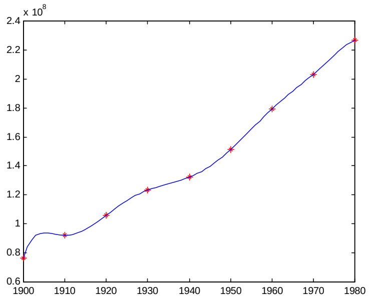
Cond2 = 5.9730e+015

3) For (t-1940)^j

Cond3 = 9.3155e+012

4) for ((t-1940)/40

Cond4 = 1.6054e+003



•What about interpolation between values?

•What about extrapolation beyond end points?

Other Polynomials

- Lagrange: Each basis function is of *(j-1)* order.
- Orthogonal Polynomials: the basis functions are orthogonal to each other in some sense. Legendre:

$$1, \quad t, \quad \frac{3t^2-1}{2}, \quad \frac{5t^3-3t}{2}, \quad \frac{35t^4-30t^2+3}{8},$$

Piecewise Interpolations

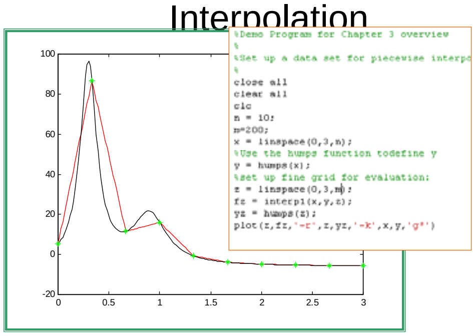
CSS455
Winter 2011

Piecewise Polynomials

- Linear fit. Straight lines connect adjacent data points.
- Each segment has two coefficients (slope and intercept). They are used to make the adjacent functions continuous at their endpoints.
- The derivatives are not continuous, resulting in “kinks” at each data point.
- Interpolation is easy. For a point *z* between the points (x_i,y_i) and (x_{i+1},y_{i+1}):

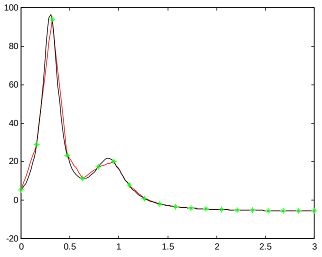
$$y(z) \approx y(x_i) + \left[\frac{y(x_{i+1}) - y(x_i)}{x_{i+1} - x_i}\right](z - x_i)$$

Example of Linear Interpolation



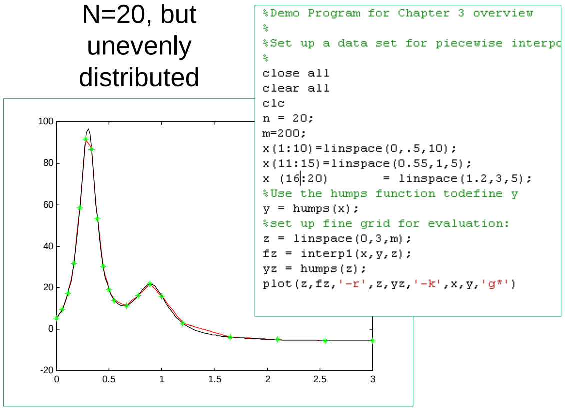
```
%Demo Program for Chapter 3 overview
%
%Set up a data set for piecewise interpo
%
close all
clear all
clc
n = 10;
m=200;
x = linspace(0,3,n);
%Use the humps function todefine y
y = humps(x);
%set up fine grid for evaluation:
z = linspace(0,3,m);
fz = interp1(x,y,z);
yz = humps(z);
plot(z,fz,'-r',z,yz,'-k',x,y,'g*')
```

Increase from n=10 to n=20 points



This is much better, but the points need to be concentrated on the left hand side of the graph for maximum efficiency.

N=20, but
unevenly
distributed



```
%Demo Program for Chapter 3 overview
%
%Set up a data set for piecewise interpo
%
close all
clear all
clc
n = 20;
m=200;
x(1:10)=linspace(0,.5,10);
x(11:15)=linspace(0.55,1,5);
x (16:20)        = linspace(1.2,3,5);
%Use the humps function todefine y
y = humps(x);
%set up fine grid for evaluation:
z = linspace(0,3,m);
fz = interp1(x,y,z);
yz = humps(z);
plot(z,fz,'-r',z,yz,'-k',x,y,'g*')
```

Linear Interpolations

- Easy.
- First derivative is discontinuous at each data point, where the curve has kinks.
- Second derivative may be infinite at those points.
- Accuracy may not be sufficient unless we use large numbers of well placed data points.

Piecewise Polynomials

- Each segment is fit with cubic polynomial (four constants to choose, 4*n* in all).

$$p_k(x) = a_k + b_k(x - x_k) + c_k(x - x_k)^2 + d_k(x - x_k)^3$$

Activity 11, Part II.

Cubic Hermite “pchip”

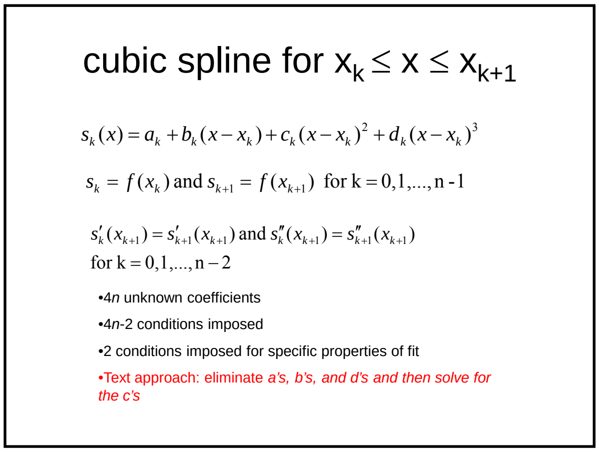
- The interpolating function and its first derivative are continuous. (*3 constants*)
- The second derivative is piecewise linear and is probably not continuous; there may be jumps at nodes.
- Can be chosen to preserve both the shape of the data and monotonicity. *(provided by choice of slopes.*
  n constants)

Cubic Hermite “pchip”

- On intervals where the data is monotonic, so is the interpolant.
- At points where the data has a local extremum, so does interpolant.

Piecewise Polynomials

- **Cubic Spline.** Each data interval [x_k, x_{k+1}] is fit with a cubic polynomial (4 coefficients).
- In addition to fitting the data, it is required that the function be twice continuously differentiable. First and second derivatives of $f(x)$ must be equal at the data points (x_i).
- For interior segments, this fixes all four parameters.
- Each end segment has one free parameter.



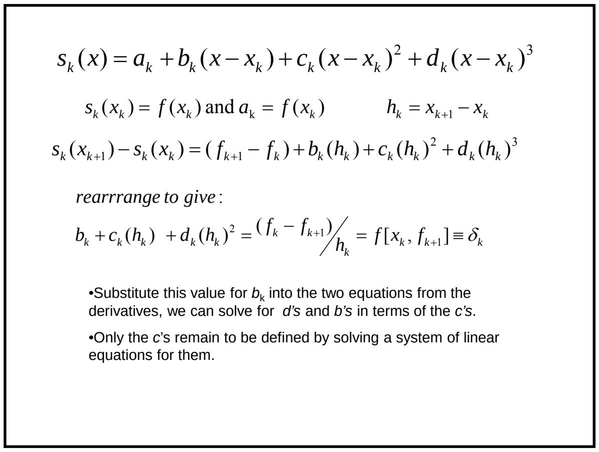
cubic spline for $x_k \le x \le x_{k+1}$

$$s_k(x) = a_k + b_k(x - x_k) + c_k(x - x_k)^2 + d_k(x - x_k)^3$$

$$s_k = f(x_k) \text{ and } s_{k+1} = f(x_{k+1}) \text{ for k} = 0,1,\ldots,\text{n - 1}$$

$$s'_k(x_{k+1}) = s'_{k+1}(x_{k+1}) \text{ and } s''_k(x_{k+1}) = s''_{k+1}(x_{k+1})$$
$$\text{for k} = 0,1,\ldots,\text{n} - 2$$

- 4*n* unknown coefficients
- 4*n*-2 conditions imposed
- 2 conditions imposed for specific properties of fit
- Text approach: eliminate *a's, b's, and d's and then solve for the c's*



$$s_k(x) = a_k + b_k(x - x_k) + c_k(x - x_k)^2 + d_k(x - x_k)^3$$

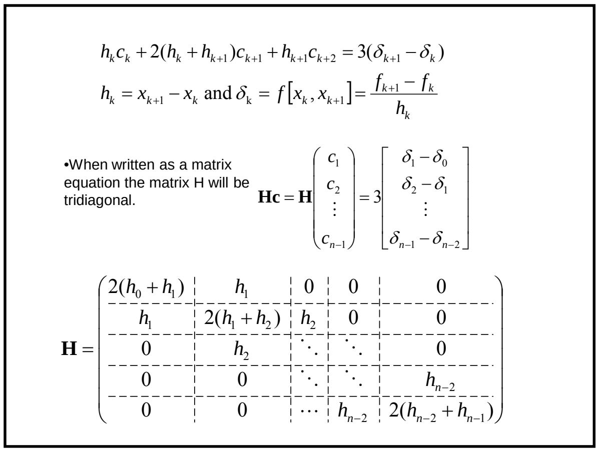
$$s_k(x_k) = f(x_k) \text{ and } a_k = f(x_k) \qquad h_k = x_{k+1} - x_k$$

$$s_k(x_{k+1}) - s_k(x_k) = (f_{k+1} - f_k) + b_k(h_k) + c_k(h_k)^2 + d_k(h_k)^3$$

rearrrange to give:

$$b_k + c_k(h_k) + d_k(h_k)^2 = {(f_k - f_{k+1})}\big/{h_k} = f[x_k, f_{k+1}] \equiv \delta_k$$

- Substitute this value for b_k into the two equations from the derivatives, we can solve for *d's* and *b's* in terms of the *c's*.
- Only the *c*'s remain to be defined by solving a system of linear equations for them.



$$h_k c_k + 2(h_k + h_{k+1})c_{k+1} + h_{k+1}c_{k+2} = 3(\delta_{k+1} - \delta_k)$$

$$h_k = x_{k+1} - x_k \text{ and } \delta_\text{k} = f\left[x_k, x_{k+1}\right] = \frac{f_{k+1} - f_k}{h_k}$$

- When written as a matrix equation the matrix H will be tridiagonal.

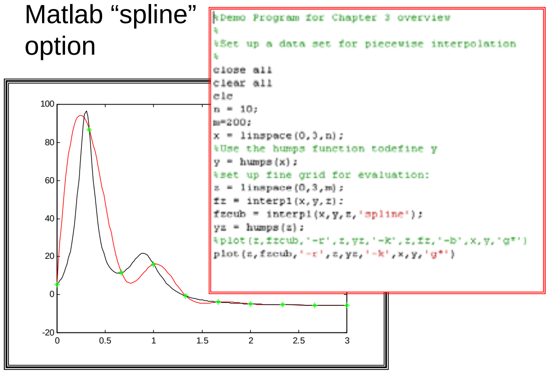
$$\mathbf{Hc} = \mathbf{H}\begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_{n-1} \end{pmatrix} = 3\begin{bmatrix} \delta_1 - \delta_0 \\ \delta_2 - \delta_1 \\ \vdots \\ \delta_{n-1} - \delta_{n-2} \end{bmatrix}$$

$$\mathbf{H} = \left(\begin{array}{c:c:c:c:c} 2(h_0 + h_1) & h_1 & 0 & 0 & 0 \\ \hdashline h_1 & 2(h_1 + h_2) & h_2 & 0 & 0 \\ \hdashline 0 & h_2 & \ddots & \ddots & 0 \\ \hdashline 0 & 0 & \ddots & \ddots & h_{n-2} \\ \hdashline 0 & 0 & \cdots & h_{n-2} & 2(h_{n-2} + h_{n-1}) \end{array}\right)$$

Cubic Spline Polynomials

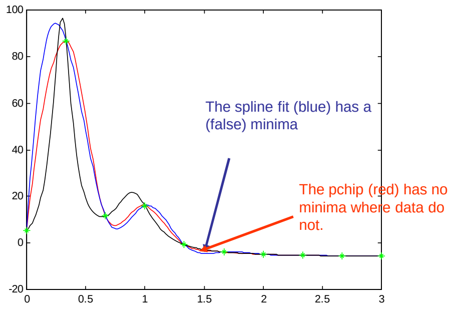
- Remaining parameters used for various other constraints: slopes at ends of intervals, periodic conditions, etc.
- **Not-a-Knot:** set end segment splines to be same as adjacent ones**.** (default Matlab mode with *spline* or *interp1)*
- **Complete:** specify the derivative at end points.*(can be done with spline function)*
- **Natural:** set second derivatives at end point equal to zero**.*(***can be done with spline fn)*

Matlab "spline" option



```
%Demo Program for Chapter 3 overview
%
%Set up a data set for piecewise interpolation
%
close all
clear all
clc
n = 10;
m=200;
x = linspace(0,3,n);
%Use the humps function todefine y
y = humps(x);
%set up fine grid for evaluation:
z = linspace(0,3,m);
fz = interp1(x,y,z);
fzcub = interp1(x,y,z,'spline');
yz = humps(z);
%plot(z,fzcub,'-r',z,yz,'-k',z,fz,'-b',x,y,'g*')
plot(z,fzcub,'-r',z,yz,'-k',x,y,'g*')
```

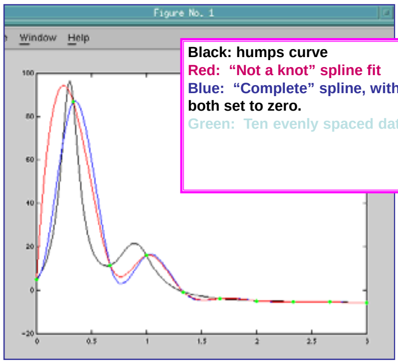
Matlab "cubic or pchip" option (red)



The spline fit (blue) has a (false) minima

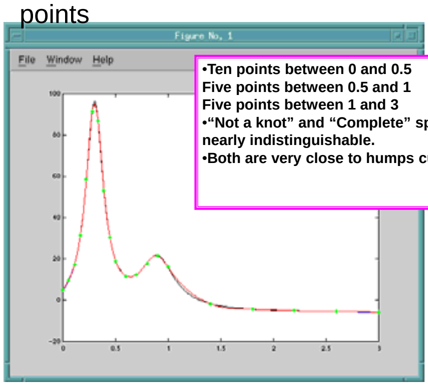
The pchip (red) has no minima where data do not.

Spline with different end conditions



Black: humps curve
Red: "Not a knot" spline fit
Blue: "Complete" spline, with endpoint slopes both set to zero.
Green: Ten evenly spaced data points

Spline with 20 unevenly spaced points



•Ten points between 0 and 0.5
Five points between 0.5 and 1
Five points between 1 and 3
•"Not a knot" and "Complete" splines are nearly indistinguishable.
•Both are very close to humps curve.

How to use PP form

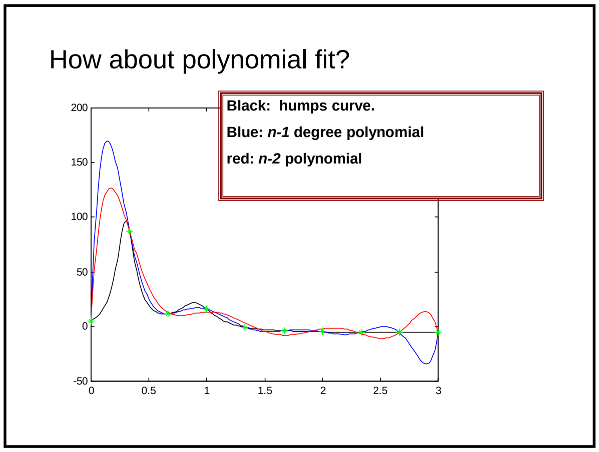
```
%set up fine grid for evaluation:
z = linspace(0,3,m);
yz = humps(z);

% fz3 = interp1(x,y,z,'spline'); %not-a-knot cubic spline
PP3 = interp1(x,y,'spline','pp'); %not-a-knot cubic spline
fz3 = ppval (PP3,z);

% fz1 = spline(x,[ 300 y 0],z); %complete spline
PP1 = spline(x,[ 300 y 0]); %complete spline
fz1 = ppval(PP1,z);
plot(z,yz,'-k',z,fz1,'-b',x,y,'g*')

% fz2 = spline(x,[ 0 y 0],z); %natural spline
PP2 = spline(x,[ 0 y 0]); %natural spline
fz2 = ppval(PP2,z);
```

PP forms can be conveniently saved.

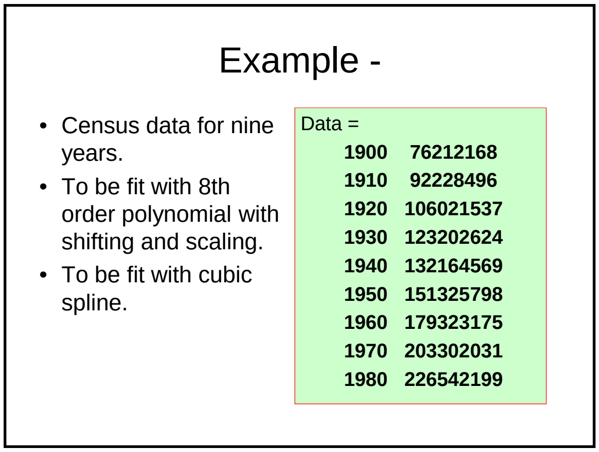


How about polynomial fit?

Black: humps curve.

Blue: *n-1* degree polynomial

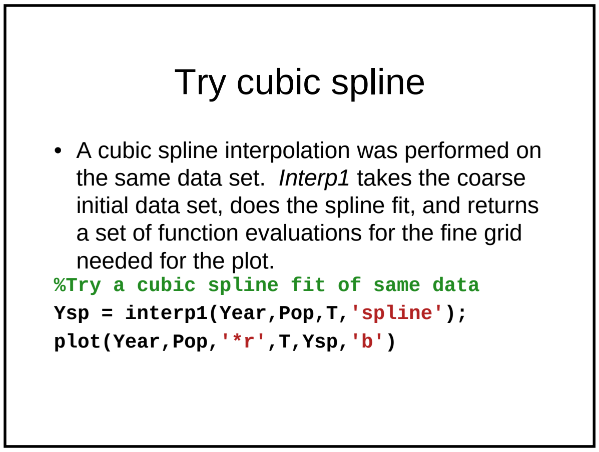
red: *n-2* polynomial



Example -

- Census data for nine years.
- To be fit with 8th order polynomial with shifting and scaling.
- To be fit with cubic spline.

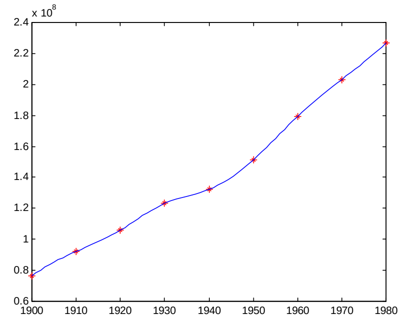
```
Data =
      1900    76212168
      1910    92228496
      1920   106021537
      1930   123202624
      1940   132164569
      1950   151325798
      1960   179323175
      1970   203302031
      1980   226542199
```



Try cubic spline

- A cubic spline interpolation was performed on the same data set. *Interp1* takes the coarse initial data set, does the spline fit, and returns a set of function evaluations for the fine grid needed for the plot.

```
%Try a cubic spline fit of same data
Ysp = interp1(Year,Pop,T,'spline');
plot(Year,Pop,'*r',T,Ysp,'b')
```



- •What about interpolation between data?
- •What about beyond the endpoints?

Extrapolation is <u>always</u> dangerous

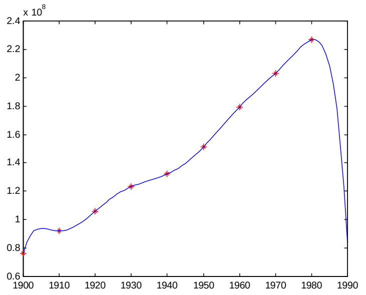
- Evaluation of the polynomial fit and the spline fit for *t* = 1990 and comparison with actual 1990 census figure.

Actual 1990 population was 248,709,873

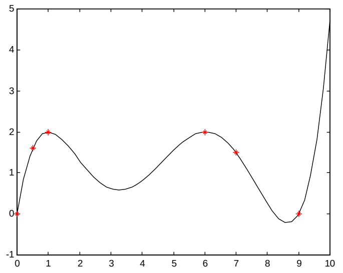
Predicted 1990 population by polynomial fit was 82,749,141

Predicted 1990 population by cubic spline fit was 256,915,297

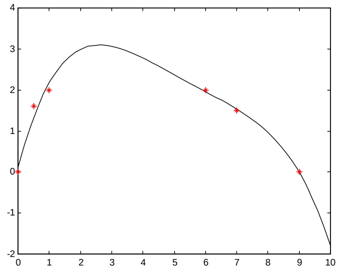
- •What happened with polynomial fit?
- •Extend the plot beyond the initial data set.



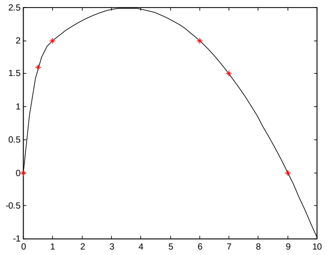
Polnomial: Test Data



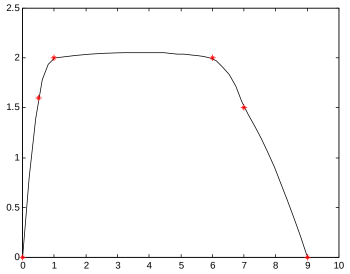
Polynomial, with reduced order



spline fit



cubic pchip fit



Cubic Hermite vs Cubic Spline

- Cubic Hermite only requires continuous function and first derivative.
- If we require derivative to be continuous, we have *n* free parameters to set.
- This allows adaptation to pleasing shapes, monotonicity, etc.

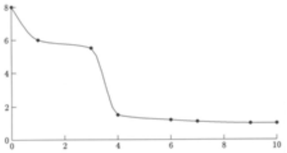


FIGURE 7.8
Monotone cubic Hermite interpolation of monotonic data.

interp1 with 'pchip'

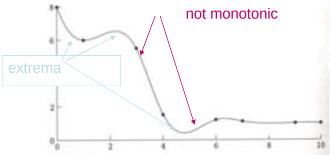
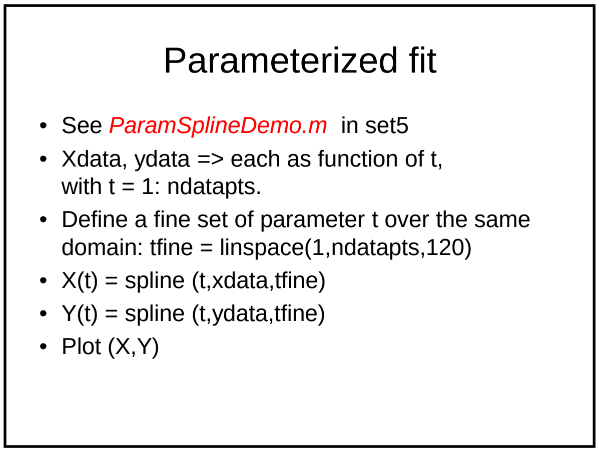


FIGURE 7.9
Cubic spline interpolation of monotonic data.

interp1 with 'spline' or direct use of 'spline' function.



Parameterized fit

- See *ParamSplineDemo.m* in set5
- Xdata, ydata => each as function of t, with t = 1: ndatapts.
- Define a fine set of parameter t over the same domain: tfine = linspace(1,ndatapts,120)
- X(t) = spline (t,xdata,tfine)
- Y(t) = spline (t,ydata,tfine)
- Plot (X,Y)