TANA09 Lecture 7

Interpolation

- Introduction. Polynomials.
- Error estimates. Runge's phenomena.
- Application - Equation solving.
- Spline functions and interpolation. End point conditions.
- Error estimate.

Interpolation

Suppose we have a table

x	x_1	x_2	...	x_{n+1}
$f(x)$	f_1	f_2	...	f_{n+1}

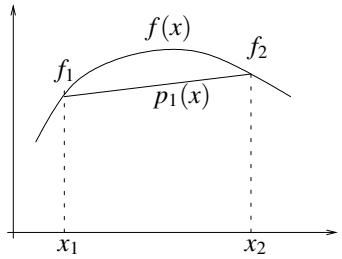
How to estimate $f(x)$ för $x_1 \leq x \leq x_{n+1}$?

Definition A polynomial $p(x)$ *interpolates* a function $f(x)$ at $x_1, x_2, \ldots, x_{n+1}$, if

$$p(x_i) = f(x_i), \qquad i = 1, 2, \ldots, n+1.$$

Questions Degree of the polynomial? How to compute the polynomial? Error estimate?

Linear Interpolation



Theorem The linear polynomial $p_1(x)$ that interpolates *two points* (x_1, f_1) and (x_1, f_2) is given by

$$p_1(x) = f_1 + \frac{x - x_1}{x_2 - x_1}(f_2 - f_1).$$

Error analysis for linear interpolation

Lemma Let the function values f_1 and f_2 have errors $|\Delta f_i| \leq \varepsilon$. If linear interpolation is used we have the error estimate

$$R_{XF} \leq \varepsilon.$$

Theorem Let $p_1(x)$ be the *linear polynomial* that interpolates $f(x)$ at x_1 and x_2. Then

$$R_T = f(x) - p_1(x) = \frac{f''(\xi)}{2}(x - x_1)(x - x_2), \quad x_1 < \xi < x_2.$$

or

$$|R_T| \leq Ch^2, \text{ where } h = x_2 - x_1.$$

Polynomial interpolation

Theorem For $n+1$ points (x_i, f_i) there is a *unique* polynomial $p_n(x)$, of degree n, that *interpolates* the given points, i.e. $p_n(x_i) = f_i$, $i = 1, 2, \ldots, n+1$.

Theorem Let $p_n(x)$ be the polynomial, of degree n, that interpolates $f(x)$ in the points $x_1, x_2, \ldots, x_{n+1}$. Then

$$f(x) - p_n(x) = \frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x - x_1) \cdots (x - x_{n+1}).$$

The function $f(x)$ has to have continuous derivatives. How to organize the computations?

Newton's interpolation formula

Find a polynomial $p_n(x)$ that interpolates the table

x	x_1	x_2	$\ldots$	x_{n+1}
$f(x)$	f_1	f_2	$\ldots$	f_{n+1}

The ansatz

$$p_n(x) = c_0 + c_1(x - x_1) + c_2(x - x_1)(x - x_2) + \ldots + c_n(x - x_1) \cdots (x - x_n),$$

leads to simple calculations. The interpolation conditions give

$$p_n(x_1) = c_0 = f_1,$$

and

$$p_n(x_2) = c_0 + c_1(x_2 - x_1) = f_2, \Longrightarrow c_1 = (f_2 - c_0)/(x_2 - x_1).$$

Every new condition $p_n(x_i) = f_i$ gives one new c_i.

Error estimate

Remark In practice $f''(\xi)$ is unknown. Instead use an *additional point* (x_{n+2}, f_{n+2}) and compute a new interpolating polynomial $p_{n+1}(x)$. Approximate

$$f^{(n+1)}(\xi(x)) \approx p_{n+1}^{(n+1)}(\xi(x)) = n! c_{n+1}.$$

Lemma Let $p_n(x)$ be a polynomial interpolating $f(X)$ at $n+1$ points. We can estimate the error

$$f(x) - p_n(x) = R_T \approx p_{n+1}(x) - p_n(x).$$

Remark If Newtons formula is used it is simple to carry out the calculations.

Example Let $f(x) = \mathrm{e}^{-x/2}\cos(x/7) + (x - 0.1)^2/2$, and suppose we have a table

x	0.0	0.5	0.7
$f(x)$	1.005	0.857	0.881

Make use of the table to estimate $f(x)$, for $x = 0.35$, by linear interpolation and estimate the error.

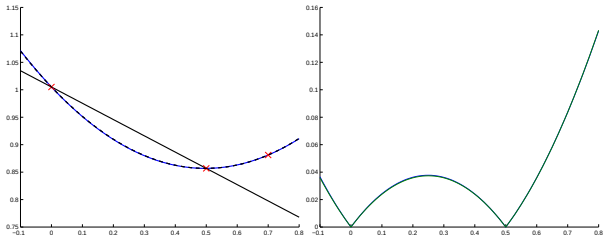
Question How to do this using Matlab? What if we want to use quadratic interpolation?

Find polynomials $p_1(x)$ and $p_2(x)$ by

```
>> x=[0 0.5 0.7]; , y=[ 1.005 0.857 0.881 ];
>> p1 = polyfit( x(1:2) , y(1:2) , 1 );
>> p2 = polyfit( x(1:3) , y(1:3) , 2 );
```

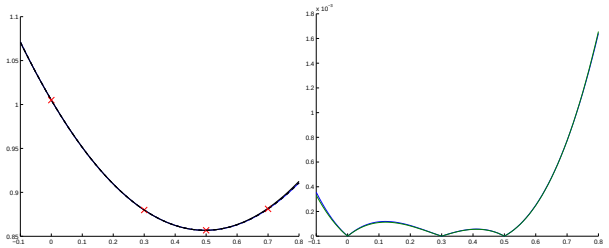
Comnpute the polynomials on a dense grid and plot by

```
>> xx=-0.1:0.01:0.8; y1=polyval(p1,xx); plot(xx,y1)
```



Left $p_1(x)$, $p_2(x)$ and $f(x)$. **Right** Error $|f(x) - p_1(x)|$ and $R_T \approx |p_2(x) - p_1(x)|$. Here $|R_T| \leq 0.038$.

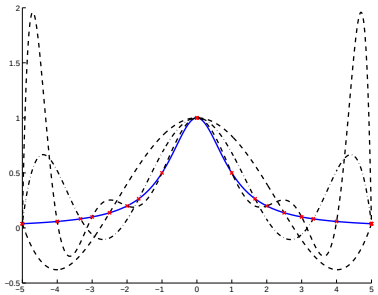
Quadratic interpolation require 3 points for $p_2(x)$ and one more for estimating R_T.



Left $p_2(x)$, $p_3(x)$ and $f(x)$. **Right** Error $|f(x) - p_2(x)|$ and $R_T \approx |p_3(x) - p_2(x)|$. Here $|R_T| \leq 1.2 \cdot 10^{-4}$.

What if we increse the degree of the polynomial?

Runge's phenomena



Polynomials of degree $n = 4$, 6, and10 that interpolates $f(x) = 1/(1+x^2)$.

The maximnum error grows as the degree increses. In practice only low degree polynomials are used.

Application - Equation solving

Problem Find the root x^* to the equation $f(x) = 0$. In an iterative method we have a sequence $x_0, x_1, x_2, \ldots, x_n$ and want the next iterate x_{n+1}. How to compute?

Method Find a quadratic polynomial $p(x)$ interpolating the table

x	x_{n-2}	x_{n-1}	x_n
$f(x)$	f_{n-2}	f_{n-1}	f_n

Compute the two roots x_i^*, $i = 1, 2$ of $p(x)$. Pick x_{n+1} as the root with the smallest function value $f(x_i^*)$.

In Matlab

```
x=[ 0 1 2];fvals=f(x);

while abs(fvals(end))>10^-15
  p=polyfit(x(end-2:end),fvals(end-2:end),2);
  r=roots(p);fr=f(r);
  if abs(fr(1))<abs(fr(2))
      x=[x,r(1)];fvals=[fvals,fr(1)];
  else
      x=[x,r(2)];fvals=[fvals,fr(2)];
  end
  fprintf(1,'%16.14f %8.2e\n', x(end),fvals(end))
end
```

Remark A quadratic polynomial has two roots. Two function evaluations in each step. Very fast convergence but a step can fail.

Example Solve $f(x) = \cos(x/2) + e^{-x/5} - x/2 - 4x^2 = 0$.

k	x_k	$f(x_k)$
0	0	$2.00 \cdot 10^{0}$
1	1	$-2.80 \cdot 10^{0}$
2	2	$-1.58 \cdot 10^{1}$
3	0.61749871787530	$2.57 \cdot 10^{-3}$
4	0.61794339080406	$2.20 \cdot 10^{-6}$
5	0.61794377127614	$-1.98 \cdot 10^{-12}$
6	0.61794377127579	$4.44 \cdot 10^{-16}$

Remark Looks like quadratic convergence! How to avoid two function evaluations?

Method Let $y_i = f(x_i)$ and $x_i = f^{-1}(y_i)$. Find a polynomial quadratic polynomial $p(y)$ interpolating

y	f_{n-2}	f_{n-1}	f_n
$f^{-1}(y)$	x_{n-2}	x_{n-1}	x_n

Pick $x_{n+1} = p(0)$. In Matlab

```
x=[ 0 1 2];fvals=f(x);
while abs(fvals(end))>10^-15
   p=polyfit(fvals(end-2:end),x(end-2:end),2);
   x=[x,polyval(p,0)];fvals=[fvals,f(x(end))];
   fprintf(1,'%16.14f %8.2e\n', x(end),fvals(end))
end
```

Remark One function evaluation in each step. The interpolation step doesn't always work. This is the main method for `fzero`.

Example Solve $f(x) = \cos(x/2) + e^{-x/5} - x/2 - 4x^2 = 0$ using *Inverse quadratic interpolation*

k	x_k	$f(x_k)$
0	0	$2.00 \cdot 10^{0}$
1	1	$-2.80 \cdot 10^{0}$
2	2	$-1.58 \cdot 10^{1}$
3	0.45769147717309	$8.20 \cdot 10^{-1}$
4	0.59042717372728	$1.56 \cdot 10^{-1}$
5	0.62070311273328	$-1.60 \cdot 10^{-2}$
6	0.61795896942350	$-8.77 \cdot 10^{-5}$
7	0.61794377007804	$6.91 \cdot 10^{-9}$
8	0.61794377127579	$2.37 \cdot 10^{-16}$

Remark Still very fast convergence!

Lagrange Interpolation

Definition Let $x_j, j = 1, \ldots, n$ be distinct points. The Lagrange basis polynomial is

$$\ell_i(x) = \Pi_{j \neq i} \frac{x - x_j}{x_i - x_j}.$$

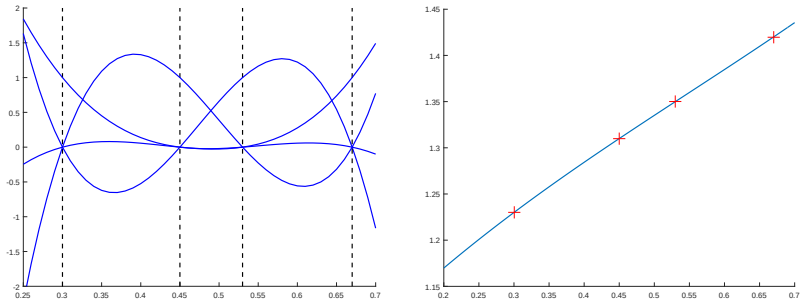
Remark We have $\ell_i(x_i) = 1$ and $\ell_i(x_j) = 0$ for $i \neq j$.

Lemma The unique polynomial of degree $n-1$ that interpolates $f(x)$ in the points x_j, $i = 1, 2, \ldots, n$ is

$$p(x) = \sum_{i=1}^{n} f(x_i)\ell_i(x).$$

Example Find the unique polynomial interpolating the table

x_i	0.30	0.45	0.53	0.67
f_i	1.23	1.31	1.35	1.42



Results The Lagrange basis polynomials of degree 3 and the interpolating polynomial. The error is $\mathcal{O}(h^4)$.

Question How to generalize to $2D$? Errors in the used function values?

Lemma Let $p(x)$ be an interpolating polynomial of degree $n-1$. If the approximate function values $\bar{f}_i$ are used then

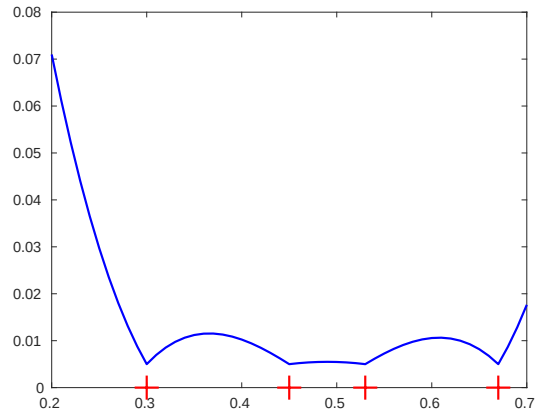
$$|\Delta p(x)| \leq \sum_{i=1}^{n} |\ell_i(x)||\Delta f_i|,$$

where $\ell_i(x)$ are the Lagrange basis polynomials.

Example Let $p(x)$ interpolate the table

x_i	0.30	0.45	0.53	0.67
f_i	1.23	1.31	1.35	1.42

Estimate the maximum error due to f_i being correctly rounded.



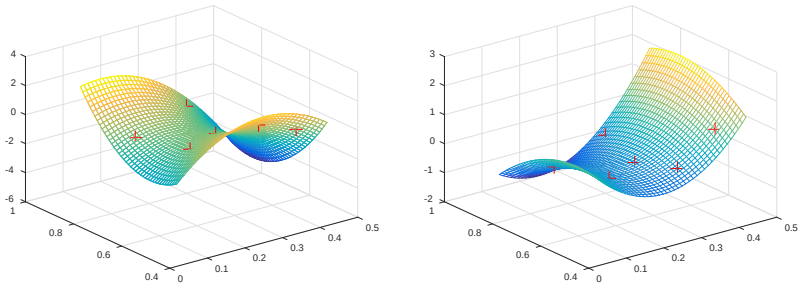
The error due to $|\Delta f_i| \leq 0.5 \cdot 10^{-2}$. Estimated using the Lagrange interpolation formula.

Remark The error is exactly $|\Delta f_i|$ at the interpolation points x_i.

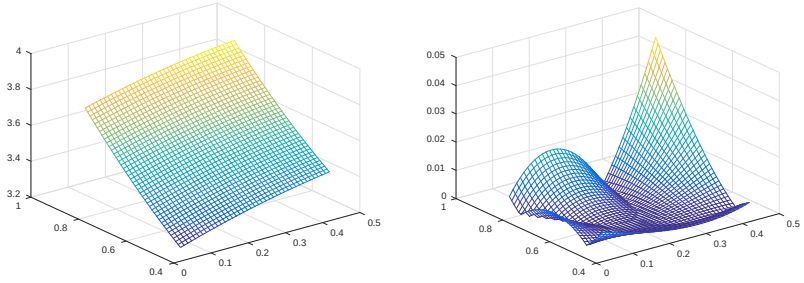
Example We have $m = 6$ points (x_i, y_j) and want to find the interpolating polynomial $u_I(x, y)$.

Introduce basis functions ℓ_i such that $\ell_i(x_j, y_j) = 1$ if $i = j$ and zero otherwise. Then the interpolant is

$$u_I(x) = \sum_{i=1}^{m} u(x_i, x_i)\ell_i(x).$$



Two basis functions $\ell_2(x, y)$ and $\ell_3(x, y)$. We use $m = 6$ interpolation points and basis functions $1, x, y, xy, x^2, y^2$.



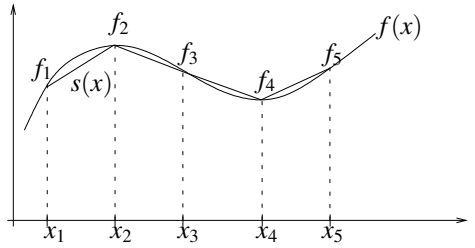
The interpolant $u_I(x, y)$ and the error $|u_I - u|$, where $u = 3 + x\sqrt{y} + \cos(2x)y^2$ was interpolated using $m = 6$ points.

Remark Really simple and flexible method. The error terms are obtained by comparing with a Taylor series expansion.

Spline interpolation

Problem A function $f(x)$ is known at certain *nodes* $x_1, x_2, \ldots, x_n$. How can we find an approximation $s(x) \approx f(x)$ on $[x_1, x_n]$?

Solution Use linear interpolation on *each* subinterval $[x_i, x_{i+1}]$.



The theory for linear interpolation holds.

Linear splines

Definition A function $s(x)$ is an *interpolating linear spline* with *nodes* $x_1, \ldots, x_n$ is

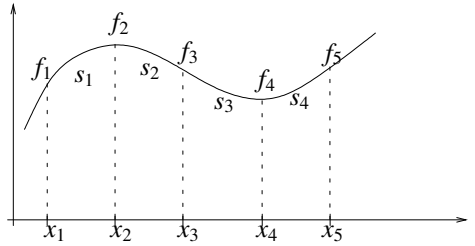
1. $s(x)$ is *continuous* on $[x_1, x_n]$.
2. $s(x)$ is a straight line on each subinterval $[x_i, x_{i+1}]$.
3. $s(x)$ interpolates $f(x)$ at the nodes, i.e. $s(x_i) = f(x_i)$.

Theorem For an *interpolating linear spline* it holds that

$$|f(x) - s(x)| \leq \frac{M}{8} h^2,$$

where $|f''(x)| \leq M$ och $h = \max |x_{i+1} - x_i|$.

Cubic splines



Definition A function $s(x)$ is an *interpolating cubic spline* with *nodes* $x_1, \ldots, x_n$ if

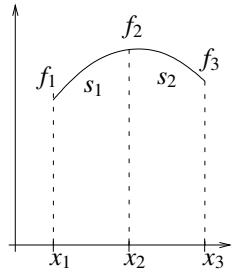
1. $s(x)$, $s'(x)$, och $s''(x)$ are *continuous* on $[x_1, x_n]$.
2. $s(x)$ is a cubic polynomial on each $[x_i, x_{i+1}]$.
3. $s(x)$ interpolates $f(x)$ at the nodes, i.e. $s(x_i) = f(x_i)$.

Example Find the *cubic* spline $s(x)$ interpolating the table

x	0	1	2
$f(x)$	1	2.5	2

with *end point conditions* $f'(0) = 1$ and $f'(2) = -1$.

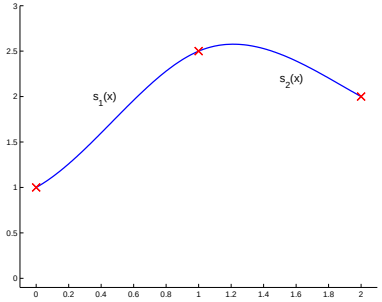
Solution Find polynomials $s_1(x)$ och $s_2(x)$.



The coefficients of s_1 and s_2 are found by solving a linear system of equations.

The resulting spline is

$$s(x)=\begin{cases} s_1(x)=1.0+1.00(x-0)+1.75(x-0)^2-1.25(x-0)^3, & 0\le x\le 1,\\ s_1(x)=2.5+0.75(x-1)-2.00(x-1)^2+0.75(x-1)^3, & 1\le x\le 2, \end{cases}$$



Two continuous derivatives and correct slopes at the end points.

End point conditions

Theorem A cubic spline $s(x)$, interpolating $f(x)$ at nodes $x_1,\dots,x_n$, is uniqely determined if we provide two *end point conditions*.

Prove by comparing the number of unknowns with the number of conditions.

We have the options

Natural conditions $s''(x_1)=s''(x_n)=0$.
Correct conditions $s'(x_1)=f'(x_1)$ and $s'(x_n)=f'(x_n)$.
Periodic conditions $s'(x_1)=s'(x_n)$.

Error estimate

Theorem If *correct end point conditions* are used then

$$|s(x)-f(x)|\le \frac{5}{384}Mh^4,$$

where $h=\max|x_{i+1}-x_i|$ and $M=\max|f^{(4)}(x)|$.

Exempel Approximate $f(x)=1/(1+x^2)$, on $[-5,5]$, with a cubic spline and correct end point conditions. How does the error depend on the number of nodes?

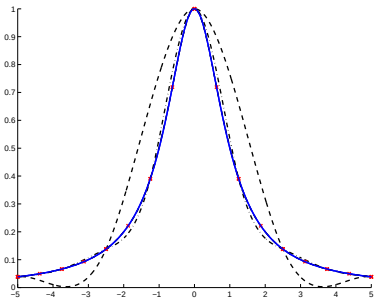
N	5	9	17
h	1/4	1/8	1/16
Felet	0.2714	0.0561	0.0037

The error behave as Ch^4.

In Matlab `csape` computes an interpolating cubic spline.

```
>> pp = csape( x , y , 'complete' , [ d1 , d2 ] );
```

where d_1 and d_2 are numerical values for the derivatives $f'(x_1)$ and $f'(x_n)$. Compute the values of the spline using `ppval`.



Plot the function $f(x)=1/(1+x^2)$ and the cubic splines $s(x)$, for $N=5$, 9, and 17 nodes.