Explorations with the Barycentric Formula for Polynomial Interpolation

Dennis Pence

Kalamazoo, Michigan, U.S.A

July 4, 2014

Other Representations for Polynomials

Students in high school deal almost exclusively with polynomials in the *power form*.

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$

This is fine if you are evaluating the polynomial near *zero* (which they usually are doing) and when the degree is low (usually $n = 2$ or 3). This form is not appropriate for $x = 50$ or 200 or for much higher degrees.

Other Representations for Polynomials

Students in high school deal almost exclusively with polynomials in the *power form*.

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$

This is fine if you are evaluating the polynomial near *zero* (which they usually are doing) and when the degree is low (usually $n = 2$ or 3). This form is not appropriate for $x = 50$ or 200 or for much higher degrees.

In Calculus we introduce the Taylor polynomials, which have a *shifted power form*.

$$\begin{aligned} P_n(x) &= b_0 + b_1 (x-a) + b_2 (x-a)^2 + \cdots + b_n (x-a)^n \\ &\quad \textit{where } b_i = \frac{f^{(i)}(a)}{i!} \end{aligned}$$

Some of my students insist on "multiplying this out" and reversing the order, so that they really believe it is a polynomial.

Other Representations (cont.)

$$\begin{aligned} P_n(x) &= b_0 + b_1(x-a) + b_2(x-a)^2 + \cdots + b_n(x-a)^n \\ & \quad \textit{where } b_i = \frac{f^{(i)}(a)}{i!} \end{aligned}$$

However if the center $x = a$ is not close to zero, we do not want this Taylor polynomial converted to the power form. Since we want to evaluate a this polynomial near the center, this is most accurately and efficiently evaluated in the shifted power form.

Other Representations (cont.)

$$P_n(x) = b_0 + b_1 (x - a) + b_2 (x - a)^2 + \cdots + b_n (x - a)^n$$
$$where \ b_i = \frac{f^{(i)}(a)}{i!}$$

However if the center $x = a$ is not close to zero, we do not want this Taylor polynomial converted to the power form. Since we want to evaluate a this polynomial near the center, this is most accurately and efficiently evaluated in the shifted power form.

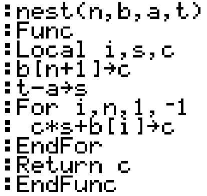
Both the power form and the shifted power form can be efficiently evaluated using *nested multiplication* (sometimes called *Horner's method* or *synthetic division*).

Nested Multiplication

$$P(t) = b_0 + (t-a)\left\{b_1 + (t-a)\left\{b_2 + (t-a)\left\{\cdots + (t-a)\left\{b_n\right\}\cdots\right\}\right\}\right\}$$

Nested Multiplication

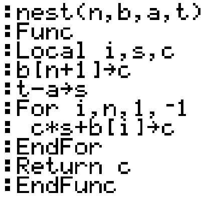
$$P(t) = b_0+(t-a)\left\{b_1+(t-a)\left\{b_2+(t-a)\left\{\cdots+(t-a)\left\{b_n\right\}\cdots\right\}\right\}\right\}$$



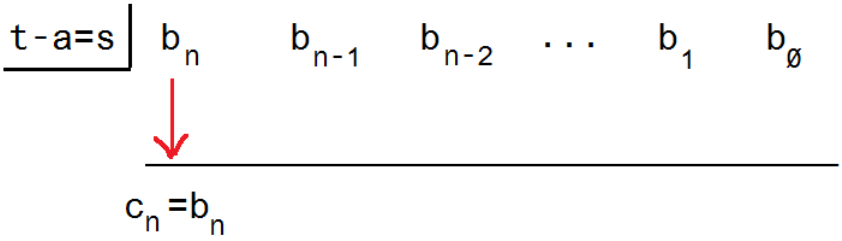
```
:nest(n,b,a,t)
:Func
:Local i,s,c
:b[n+1]→c
:t-a→s
:For i,n,1,⁻1
:  c*s+b[i]→c
:EndFor
:Return c
:EndFunc
```

Nested Multiplication

$$P(t) = b_0 + (t-a)\{b_1 + (t-a)\{b_2 + (t-a)\{\cdots + (t-a)\{b_n\}\cdots\}\}\}$$

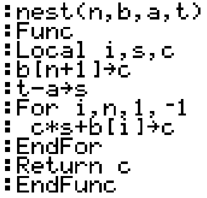


```
:nest(n,b,a,t)
:Func
:Local i,s,c
:b[n+1]→c
:t-a→s
:For i,n,1,-1
: c*s+b[i]→c
:EndFor
:Return c
:EndFunc
```

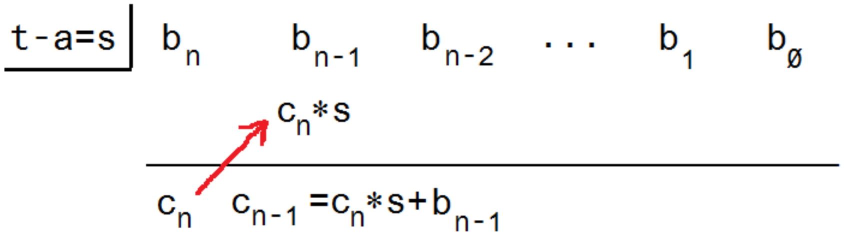


Nested Multiplication (cont.)

$$P(t) = b_0 + (t-a)\{b_1 + (t-a)\{b_2 + (t-a)\{\cdots + (t-a)\{b_n\}\cdots\}\}\}$$

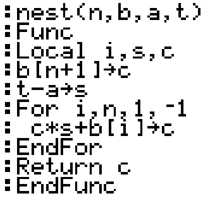


```
:nest(n,b,a,t)
:Func
:Local i,s,c
:b[n+1]→c
:t-a→s
:For i,n,1,-1
:  c*s+b[i]→c
:EndFor
:Return c
:EndFunc
```

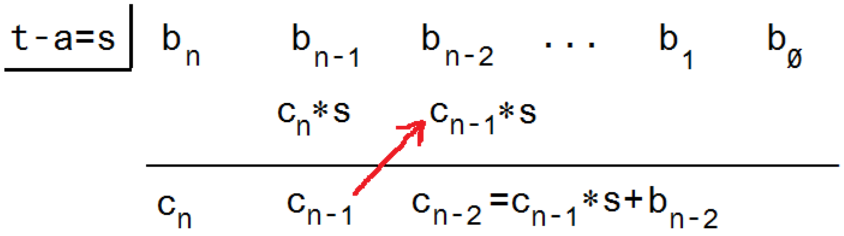


Nested Multiplication (cont.)

$$P(t) = b_0 + (t-a)\{b_1 + (t-a)\{b_2 + (t-a)\{\cdots + (t-a)\{b_n\}\cdots\}\}\}$$



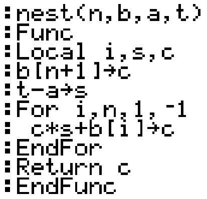
```
:nest(n,b,a,t)
:Func
:Local i,s,c
:b[n+1]→c
:t-a→s
:For i,n,1,-1
:  c*s+b[i]→c
:EndFor
:Return c
:EndFunc
```



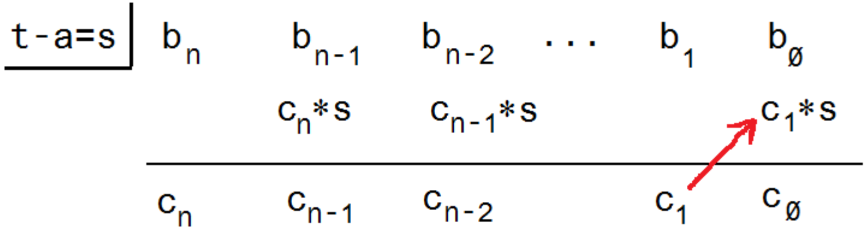
t-a=s
b_n b_{n-1} b_{n-2} ... b_1 $b_Ø$
c_n*s $c_{n-1}*s$
c_n c_{n-1} $c_{n-2}=c_{n-1}*s+b_{n-2}$

Nested Multiplication (cont.)

$$P(t) = b_0 + (t-a)\{b_1 + (t-a)\{b_2 + (t-a)\{\cdots + (t-a)\{b_n\}\cdots\}\}\}$$



```
:nest(n,b,a,t)
:Func
:Local i,s,c
:b[n+1]→c
:t-a→s
:For i,n,1,-1
:  c*s+b[i]→c
:EndFor
:Return c
:EndFunc
```



Polynomial Interpolation

Given any function $f(x)$, perhaps only known via a table, we seek a polynomial of degree at most n that matches this function at $n+1$ points $\{x_i\}$, called the nodes.

Polynomial Interpolation

Given any function $f(x)$, perhaps only known via a table, we seek a polynomial of degree at most n that matches this function at $n+1$ points $\{x_i\}$, called the nodes.
Generally we let $y_i = f(x_i),\ i = 0, 1, \ldots, n$, and we usually express the interpolation conditions in a table.

x_0	x_1	$\cdots$	x_{n-1}	x_n
y_0	y_1	$\cdots$	y_{n-1}	y_n

Polynomial Interpolation

Given any function $f(x)$, perhaps only known via a table, we seek a polynomial of degree at most n that matches this function at $n+1$ points $\{x_i\}$, called the nodes.
Generally we let $y_i = f(x_i),\ i = 0, 1, \ldots, n$, and we usually express the interpolation conditions in a table.

x_0	x_1	$\cdots$	x_{n-1}	x_n
y_0	y_1	$\cdots$	y_{n-1}	y_n

Find $P(x)$ of degree at most n satisfying the conditions

$$P(x_i) = y_i,\ i = 0, 1, 2, \ldots, n$$

Polynomial Interpolation (cont.)

For most of our work, these nodes do not need to be in any order. However, for simplicity we wish to assume that they are ordered, and today the preferred order is in *decreasing order.*

$$x_0 > x_1 > x_2 > \cdots > x_{n-2} > x_{n-1} > x_n$$

Polynomial Interpolation (cont.)

For most of our work, these nodes do not need to be in any order. However, for simplicity we wish to assume that they are ordered, and today the preferred order is in *decreasing order.*

$$x_0 > x_1 > x_2 > \cdots > x_{n-2} > x_{n-1} > x_n$$

The word "interpolate" conveys a sense of "between-ness". Thus we should only be evaluating this interpolating polynomial $P(x)$ for x=values between x_0 and x_n.

Polynomial Interpolation (cont.)

For most of our work, these nodes do not need to be in any order. However, for simplicity we wish to assume that they are ordered, and today the preferred order is in *decreasing order.*

$$x_0 > x_1 > x_2 > \cdots > x_{n-2} > x_{n-1} > x_n$$

The word "interpolate" conveys a sense of "between-ness". Thus we should only be evaluating this interpolating polynomial $P(x)$ for x=values between x_0 and x_n.
However in many applications (especially in business and finance), time is the independent variable x and the data consist of results we have seen up to the current time. Our most important question is what will happen in the future. Technically this is called "extrapolation". But we may very well evaluate the same polynomial $P(x)$ beyond the set of nodes. [Warning: Use caution with extrapolation, and don't go very far beyond the nodes!]

Polynomial Interpolation, Power Form

Using the power form, the $(n+1)$ interpolation conditions give rise to $(n+1)$ equations in the $(n+1)$ coefficients for the polynomial of degree at most n. In matrix notation, this leads to a full matrix, called the *Vandermonde* matrix. Unfortunately these matrices are very *ill-conditioned*, giving us matrix problems that are very difficult to solve. The computational effort is $O\left(n^3\right)$, and even completed in the best way possible, we will have serious concerns about the accuracy of the solutions.

$$\begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ 1 & x_2 & x_2^2 & \cdots & x_2^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

Polynomial Interpolation, Power Form

Using the power form, the $(n+1)$ interpolation conditions give rise to $(n+1)$ equations in the $(n+1)$ coefficients for the polynomial of degree at most n. In matrix notation, this leads to a full matrix, called the *Vandermonde* matrix. Unfortunately these matrices are very *ill-conditioned*, giving us matrix problems that are very difficult to solve. The computational effort is $O\left(n^3\right)$, and even completed in the best way possible, we will have serious concerns about the accuracy of the solutions.

$$\begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ 1 & x_2 & x_2^2 & \cdots & x_2^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

Since it is difficult to find the coefficients and it may be difficult to evaluate the polynomial in this form, no practical computations are done this way!

Polynomial Interpolation, Newton Form

Virtually all numerical analysis textbooks recommend the *Newton form*.

$$\begin{aligned} P(x) &= c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1) + \cdots \\ &\quad + c_n(x - x_0)(x - x_1)\cdots(x - x_{n-1}) \end{aligned}$$

Polynomial Interpolation, Newton Form

Virtually all numerical analysis textbooks recommend the *Newton form*.

$$\begin{aligned} P(x) &= c_0 + c_1 (x - x_0) + c_2 (x - x_0)(x - x_1) + \cdots \\ &\quad + c_n (x - x_0)(x - x_1) \cdots (x - x_{n-1}) \end{aligned}$$

Thus

$$\begin{aligned} p_1(x) &= c_0 + c_1 (x - x_0) \quad \text{will interpolate at nodes } x_0, x_1 \\ p_2(x) &= c_0 + c_1 (x - x_0) + c_2 (x - x_0)(x - x_1) \quad \text{at nodes } x_0, x_1, x_2 \\ p_3(x) &= p_2(x) + c_3 (x - x_0)(x - x_1)(x - x_2) \quad \text{nodes } x_0, x_1, x_2, x_3 \end{aligned}$$

Polynomial Interpolation, Newton Form (cont.)

This leads to an lower triangular matrix in the equations for the coefficients.

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 1 & (x_1 - x_0) & 0 & \cdots & 0 \\ 1 & (x_2 - x_0) & (x_2 - x_0)(x_2 - x_1) & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ 1 & (x_n - x_0) & (x_n - x_0)(x_n - x_1) & \cdots & \prod_{j=0}^{n-1}(x_n - x_j) \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}$$

Polynomial Interpolation, Newton Form (cont.)

This leads to an lower triangular matrix in the equations for the coefficients.

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 1 & (x_1 - x_0) & 0 & \cdots & 0 \\ 1 & (x_2 - x_0) & (x_2 - x_0)(x_2 - x_1) & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ 1 & (x_n - x_0) & (x_n - x_0)(x_n - x_1) & \cdots & \prod_{j=0}^{n-1}(x_n - x_j) \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}$$

We use a *divided difference table* to efficiently solve for the coefficients. We can also generalize our nested multiplication (changing the object factored out) to efficiently evaluate this Newton form.

Polynomial Interpolation, Lagrange Form

Mostly for theoretical purposes, we create a custom basis for the space of polynomials of degree at most n consisting of the following,

$$\varphi_i(x) = \frac{\prod_{j=0,\, j\neq i}^{n} (x - x_j)}{\prod_{j=0,\, j\neq i}^{n} (x_i - x_j)}, \quad i = 0, 1, 2, \ldots, n$$

with the feature that for each i we have

$$\varphi_i\left(x_i\right) = 1, \;\; \varphi_i\left(x_j\right) = 0, \quad \text{for all}\, j \neq i.$$

Polynomial Interpolation, Lagrange Form

Mostly for theoretical purposes, we create a custom basis for the space of polynomials of degree at most n consisting of the following,

$$\varphi_i(x) = \frac{\prod_{j=0,\, j\neq i}^{n} (x - x_j)}{\prod_{j=0,\, j\neq i}^{n} (x_i - x_j)}, \quad i = 0, 1, 2, \ldots, n$$

with the feature that for each i we have

$$\varphi_i(x_i) = 1, \;\; \varphi_i(x_j) = 0, \quad \text{for all}\, j \neq i.$$

Then with no matrix problem to solve at all, we get that

$$P(x) = y_0\varphi_0(x) + y_1\varphi_1(x) + y_2\varphi_2(x) + \cdots + y_n\varphi_n(x).$$

Polynomial Interpolation, Lagrange Form

Mostly for theoretical purposes, we create a custom basis for the space of polynomials of degree at most n consisting of the following,

$$\varphi_i(x) = \frac{\prod_{j=0,\, j\neq i}^{n} (x - x_j)}{\prod_{j=0,\, j\neq i}^{n} (x_i - x_j)}, \quad i = 0, 1, 2, \ldots, n$$

with the feature that for each i we have

$$\varphi_i(x_i) = 1, \ \ \varphi_i(x_j) = 0, \quad \text{for all}\, j \neq i.$$

Then with no matrix problem to solve at all, we get that

$$P(x) = y_0\varphi_0(x) + y_1\varphi_1(x) + y_2\varphi_2(x) + \cdots + y_n\varphi_n(x).$$

But we always thought it was too much trouble to efficiently evaluate this form!

Evaluation of the Lagrange Form

First we can note that we can pre-compute the denominator-part of these Lagrange basis polynomials, and these will be called the *weights*.

$$w_i = \frac{1}{\prod_{j=0,\, j\neq i}^{n} (x_i - x_j)}, \;\; i = 0, 1, 2, \ldots, n$$

Further we create what we call the *nodal polynomial* with all of the factors involving the nodes.

$$\varphi(x) = \prod_{j=0}^{n} (x - x_j)$$

Evaluation of the Lagrange Form

First we can note that we can pre-compute the denominator-part of these Lagrange basis polynomials, and these will be called the *weights*.

$$w_i = \frac{1}{\prod_{j=0,\, j\neq i}^{n} (x_i - x_j)}, \;\; i = 0, 1, 2, \ldots, n$$

Further we create what we call the *nodal polynomial* with all of the factors involving the nodes.

$$\varphi(x) = \prod_{j=0}^{n} (x - x_j)$$

Then we get the *first barycentric formula*.

$$\begin{aligned} P(x) &= y_0\varphi_0(x) + y_1\varphi_1(x) + y_2\varphi_2(x) + \cdots + y_n\varphi_n(x) \\ &= \varphi(x) \sum_{i=0}^{n} \frac{y_i w_i}{(x - x_i)} \end{aligned}$$

Evaluation of the Lagrange Form (cont.)

$$P(x) = \varphi(x) \sum_{i=0}^{n} \frac{y_i w_i}{(x - x_i)}$$

But these interpolating polynomials are unique, and so the polynomial interpolating the function $f(x) = 1$ must in fact be the constant polynomial $P(x) = 1$. Thus

$$1 = \varphi(x) \sum_{i=0}^{n} \frac{w_i}{(x - x_i)}.$$

Evaluation of the Lagrange Form (cont.)

$$P(x) = \varphi(x) \sum_{i=0}^{n} \frac{y_i w_i}{(x - x_i)}$$

But these interpolating polynomials are unique, and so the polynomial interpolating the function $f(x) = 1$ must in fact be the constant polynomial $P(x) = 1$. Thus

$$1 = \varphi(x) \sum_{i=0}^{n} \frac{w_i}{(x - x_i)}.$$

So now we divide our “general” first barycentric formula by 1 to get the second barycentric formula. Notice we can eliminate the nodal polynomial.

$$P(x) = \frac{\sum_{i=0}^{n} \frac{y_i w_i}{(x - x_i)}}{\sum_{i=0}^{n} \frac{w_i}{(x - x_i)}}$$

Evaluation of the Lagrange Form (cont.)

$$P(x) = \frac{\sum_{i=0}^{n} \frac{y_i w_i}{(x-x_i)}}{\sum_{i=0}^{n} \frac{w_i}{(x-x_i)}}$$

One final simplification we sometimes use is this. Since the weights are used in both the numerator and the denominator, we can scale the weights by any constant without altering the resulting quotient.

$$\widetilde{w}_i = K w_i, \text{ for any constant } K \neq 0, \; i = 0, 1, 2, \ldots, n$$

An Example Using NASA Data

I like to give my students the following problem which was actually handed to me one summer over 30 years ago when I worked at the Goddard Spaceflight Center:

- Given the data below, approximate the time T when the geodetic latitude $\theta = -80^\circ$.

θ (in deg)	-79.789	-80.387	-80.818	-81.058	-81.091
T (in sec)	144	120	96	72	48

- Try first an interpolating polynomial of degree at most $n = 4$. When you do not like the answer that you get (the NASA worker handing me the task did not!) suggest something else.

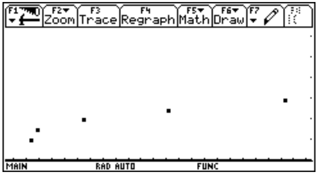
An Example Using NASA Data

I like to give my students the following problem which was actually handed to me one summer over 30 years ago when I worked at the Goddard Spaceflight Center:

- Given the data below, approximate the time T when the geodetic latitude $\theta = -80^\circ$.

θ (in deg)	-79.789	-80.387	-80.818	-81.058	-81.091
T (in sec)	144	120	96	72	48

- Try first an interpolating polynomial of degree at most $n = 4$. When you do not like the answer that you get (the NASA worker handing me the task did not!) suggest something else.



NASA Problem via Newton Form

θ (in deg)	-79.789	-80.387	-80.818	-81.058	-81.091
T (in sec)	144	120	96	72	48

$$\begin{aligned} P(\theta) \quad &= \quad 144 + (\theta + 79.789)\,\{40.1338 + (\theta + 80.387)\,\{-15.1124 \\ &\quad + (\theta + 80.818)\,\{40.1352 + (\theta + 81.058)\,\{2403.87\}\}\}\ \} \end{aligned}$$

Then evaluating at $\theta = -80.$ gives the following.

$$P(-80.) = 303.965$$

NASA Problem via Barycentric Formula

θ (in deg)	-79.789	-80.387	-80.818	-81.058	-81.091
T (in sec)	144	120	96	72	48

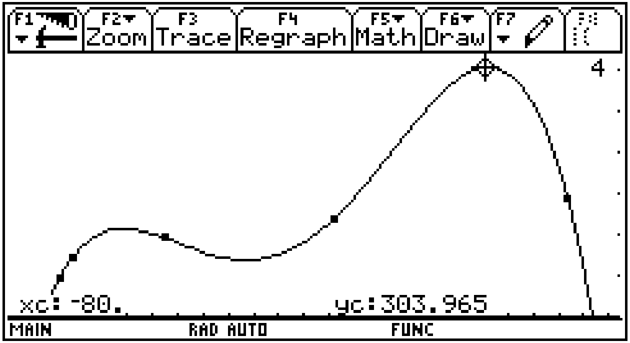
$$P(\theta) = \frac{\frac{144(.983583)}{\theta+79.789} - \frac{120(8.21346)}{\theta+80.387} + \frac{96(34.4139)}{\theta+80.818} - \frac{72(148.283)}{\theta+81.058} + \frac{48(121.009)}{\theta+81.091}}{\frac{(.983583)}{\theta+79.789} - \frac{(8.21346)}{\theta+80.387} + \frac{(34.4139)}{\theta+80.818} - \frac{(148.283)}{\theta+81.058} + \frac{(121.009)}{\theta+81.091}}$$

Then evaluating at $\theta = -80.$ gives the following.

$$P(-80.) = 303.965$$

NASA Problem Graph of $P(\theta)$

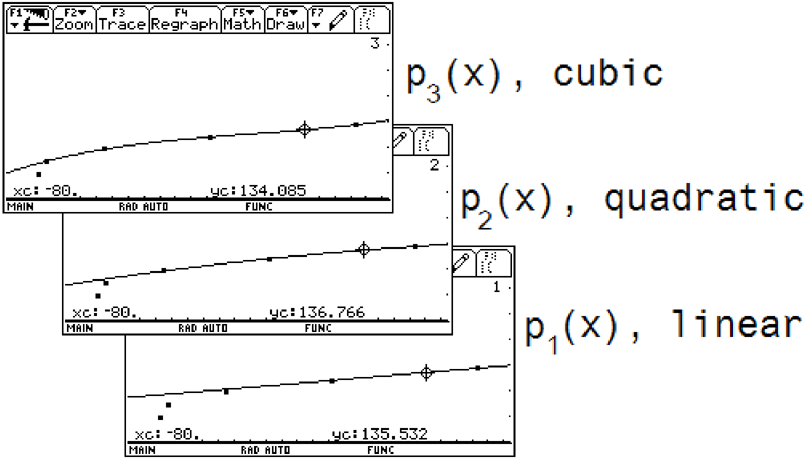
θ (in deg)	-79.789	-80.387	-80.818	-81.058	-81.091
T (in sec)	144	120	96	72	48



xc: -80. yc:303.965

NASA Problem Alternatives

θ (in deg)	-79.789	-80.387	-80.818	-81.058	-81.091
T (in sec)	144	120	96	72	48

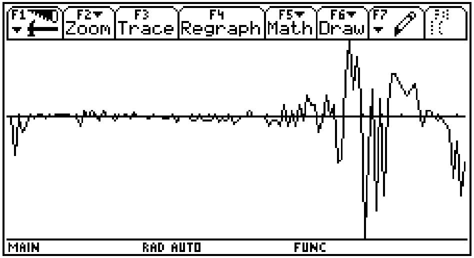


```
xc: -80.          yc:134.085
MAIN        RAD AUTO        FUNC
```

```
xc: -80.          yc:136.766
MAIN        RAD AUTO        FUNC
```

```
xc: -80.          yc:135.532
MAIN        RAD AUTO        FUNC
```

$$\begin{aligned} P(\theta) &= 144 + (\theta + 79.789)\{40.1338 + (\theta + 80.387)\{-15.1124 \\ &\quad + (\theta + 80.818)\{40.1352 + (\theta + 81.058)\{2403.87\}\}\}\ \} \\ P(\theta) &= \frac{\frac{144(.983583)}{\theta+79.789} - \frac{120(8.21346)}{\theta+80.387} + \frac{96(34.4139)}{\theta+80.818} - \frac{72(148.283)}{\theta+81.058} + \frac{48(121.009}{\theta+81.091}}{\frac{(.983583)}{\theta+79.789} - \frac{(8.21346)}{\theta+80.387} + \frac{(34.4139)}{\theta+80.818} - \frac{(148.283)}{\theta+81.058} + \frac{(121.009)}{\theta+81.091}} \end{aligned}$$



$$-81.22 \le \theta \le -79.66,\ -2.3E-10 \le y \le 1.4E-10$$

Comparison of Computations

TI-89 Programs

```
:coef(n,x,y)
:Func
: Local i,a,t,j
:For i,1,n+1
:   y[i]→a[i]
:EndFor
:For j,1,n
: For i,n,j,-1
:   (a[i+1]-a[i])/(x[i+1]-x[i-j+1])→a[i+1]
: EndFor
:EndFor
```

Divided Difference, Newton

```
:wbary(n,xx)
:Func
: Local i,ww,zz,temp,j
:For i,1,n+1
:   1→ww[i]
:   1→zz[i]
:EndFor
:For i,1,n
: For j,1,i
:   xx[j]-xx[i+1]→zz[j]
: EndFor
: For j,1,i
:   ww[j]/(zz[j])→ww[j]
: EndFor
: 1→temp
: For j,1,i
:   -temp*zz[j]→temp
: EndFor
: 1/temp→ww[i+1]
:EndFor
:Return ww
:EndFunc
```

Weights Computed, Bary.

```
:eval(n,x,a,t)
:Func
:Local answ,i
: a[n+1]→answ
: For i,n,1,-1
:   answ*(t-x[i])+a[i]→answ
: EndFor
:Return answ
:EndFunc
```

Nested Multiplication,
Newton Form

```
:baryeval(n,x,y,w,t)
:Func
:Local num,den,m,i,s
:0→num
:0→den
:n+1→m
:For i,1,m
: w[i]/(t-x[i])→s
: num+y[i]*s→num
: den+s→den
:EndFor
:num/den→s
:Return s
:EndFunc
```

Barycentric Formula

Special Choices of Nodes

For many special choices of nodes, the *weights* in the barycentric formula are known (and no computation is needed). For simplicity, assume the interval is $[-1, 1]$.

- Equally spaced nodes: the normalized weights can be chosen as

$$x_i = 1 - \frac{2j}{n}, \ \widetilde{w}_i = (-1)^j \left(\begin{array}{c} n \\ i \end{array} \right), \ i = 0, 1, 2, \ldots, n$$

Special Choices of Nodes

For many special choices of nodes, the *weights* in the barycentric formula are known (and no computation is needed). For simplicity, assume the interval is $[-1, 1]$.

- Equally spaced nodes: the normalized weights can be chosen as

$$x_i = 1 - \frac{2j}{n}, \ \widetilde{w}_i = (-1)^j \left(\begin{array}{c} n \\ i \end{array} \right), \ i = 0, 1, 2, \ldots, n$$

- Chebyshev nodes of the first kind (zeros of a Chebyshev polynomial):

$$\begin{aligned} x_i &= \cos\left(\frac{(2i+1)\,\pi}{2n+2}\right), \ \widetilde{w}_i = (-1)^j \sin\left(\frac{(2i+1)\,\pi}{2n+2}\right), \\ & i = 0, 1, 2, \ldots, n \end{aligned}$$

Special Choices of Nodes (cont.)

- Chebyshev nodes of the second kind (extrema of a Chebyshev polynomial):

$$\begin{aligned}
x_i &= \cos\left(\frac{j\pi}{n}\right), \ i = 0, 1, 2, \ldots, n \\
\widetilde{w_0} &= \left(\frac{1}{2}\right) \\
\widetilde{w_i} &= (-1)^j, \ i = 1, 2, \ldots, (n-1) \\
\widetilde{w_n} &= (-1)^n \left(\frac{1}{2}\right) \\
\frac{1}{K} &= 2^{n-1} \prod_{j=1}^{2n-1} \sin\left(\frac{j\pi}{2n}\right)
\end{aligned}$$

Barycentric Formula for Chebyshev Nodes (2nd)

$$P(x) = \frac{\frac{y_0}{2(x-x_0)} + \sum_{i=1}^{n-1} \frac{y_i(-1)^i}{(x-x_i)} + \frac{y_n(-1)^n}{2(x-x_n)}}{\frac{1}{2(x-x_0)} + \sum_{i=1}^{n-1} \frac{(-1)^i}{(x-x_i)} + \frac{(-1)^n}{2(x-x_n)}}$$

where

$$\begin{aligned} x_i &= \cos\left(\frac{j\pi}{n}\right), \; y_i = f(x_i) \\ & i = 0, 1, 2, \ldots, n \end{aligned}$$

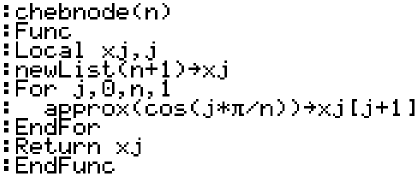
TI-89 Implementation

```
:chebnode(n)
:Func
:Local xj,j
:newList(n+1)→xj
:For j,0,n,1
:  approx(cos(j*π/n))→xj[j+1]
:EndFor
:Return xj
:EndFunc
```

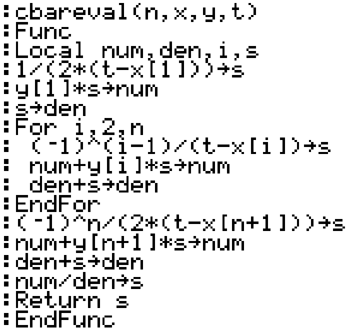
```
:chebwt(n)
:Func
:Local wj,j
:newList(n+1)→wj
:wj .+ 1→wj
:For j,1,n,2
:   ⁻1→wj[j+1]
:EndFor
:wj[1]/2→wj[1]
:wj[n+1]/2→wj[n+1]
:Return wj
:EndFunc
```

```
:baryeval(n,x,y,w,t)
:Func
:Local num,den,m,i,s
:0→num
:0→den
:n+1→m
:For i,1,m
: w[i]/(t-x[i])→s
: num+y[i]*s→num
: den+s→den
:EndFor
:num/den→s
:Return s
:EndFunc
```

TI-89 Implementation (cont.)



```
:chebnode(n)
:Func
:Local xj,j
:newList(n+1)→xj
:For j,0,n,1
:  approx(cos(j*π/n))→xj[j+1]
:EndFor
:Return xj
:EndFunc
```



```
:cbareval(n,x,y,t)
:Func
:Local num,den,i,s
:1/(2*(t-x[1]))→s
:y[1]*s→num
:s→den
:For i,2,n
: (⁻1)^(i-1)/(t-x[i])→s
: num+y[i]*s→num
: den+s→den
:EndFor
:(⁻1)^n/(2*(t-x[n+1]))→s
:num+y[n+1]*s→num
:den+s→den
:num/den→s
:Return s
:EndFunc
```

Berrut-Trefethen Example

J.-P. Berrut & L. N. Trefethen, "Barycentric Lagrange Interpolation," SIAM Review, 2004 demonstrate with Matlab code for the following function.

$$f(x) = |x| + \frac{1}{2}x - x^2, \ -1 \leq x \leq 1$$

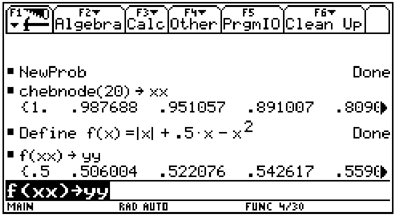
Let us start with Chebyshev nodes (2nd) with $n = 20$.

Berrut-Trefethen Example

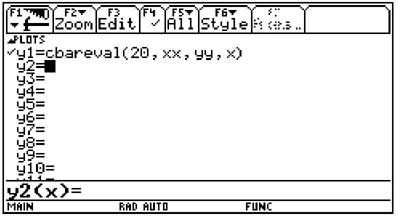
J.-P. Berrut & L. N. Trefethen, "Barycentric Lagrange Interpolation," SIAM Review, 2004 demonstrate with Matlab code for the following function.

$$f(x) = |x| + \frac{1}{2}x - x^2, \ -1 \le x \le 1$$

Let us start with Chebyshev nodes (2nd) with $n = 20$.



```
■ NewProb                                   Done
■ chebnode(20) → xx
  {1.   .987688   .951057   .891007   .8090▶
■ Define f(x)=|x| + .5·x - x^2              Done
■ f(xx) → yy
  {.5   .506004   .522076   .542617   .5590▶
f(xx)→yy
```

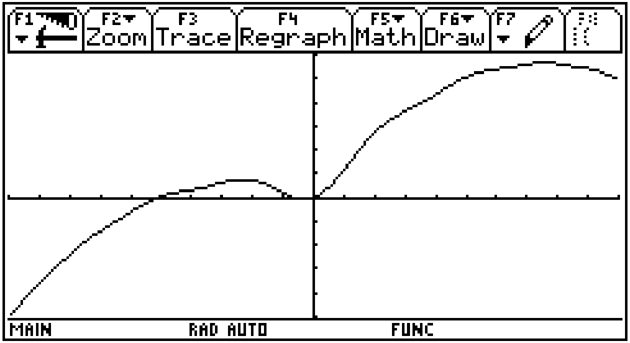


```
PLOTS
✓y1=cbareval(20,xx,yy,x)
 y2=
 y3=
 y4=
 y5=
 y6=
 y7=
 y8=
 y9=
 y10=
y2(x)=
```

Berrut-Trefethen Example (cont.)

$$f(x) = |x| + \frac{1}{2}x - x^2, \; -1 \le x \le 1$$

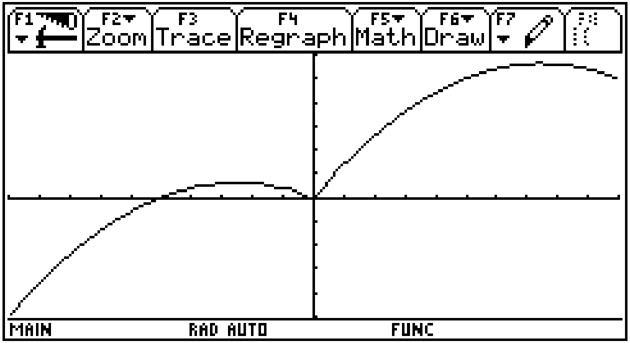
Chebyshev nodes (2nd), $n = 20$.



Berrut-Trefethen Example (cont.)

$$f(x) = |x| + \frac{1}{2}x - x^2, \ -1 \leq x \leq 1$$

Chebyshev nodes (2nd), $n = 60$.



Runge's Example

If we translate Runge's original example (on the interval $[-5, 5]$) to our standard interval, it becomes the following.

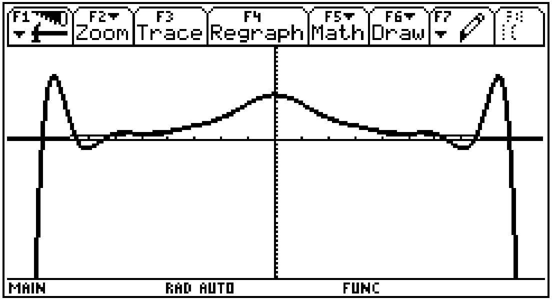
$$f(x) = \frac{1}{1 + 25x^2}$$

Runge's Example

If we translate Runge's original example (on the interval $[-5, 5]$) to our standard interval, it becomes the following.

$$f(x) = \frac{1}{1 + 25x^2}$$

Equally spaced nodes, $n = 16$.



$$-1 \quad \leq \quad x \leq 1, \ -3 \leq y \leq 2$$

$$|\text{max error}| \approx 14.4$$

Runge's Example (cont.)

Runge's original example.

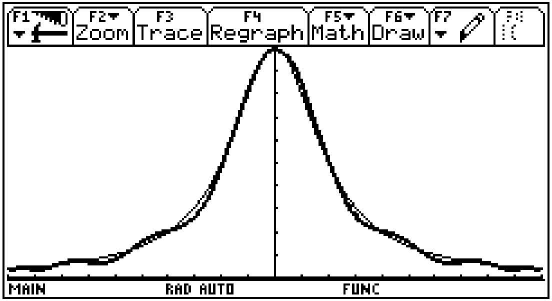
$$f(x) = \frac{1}{1 + 25x^2}$$

Runge's Example (cont.)

Runge's original example.

$$f(x) = \frac{1}{1 + 25x^2}$$

Chebyshev nodes (2nd), $n = 16$.



$$-1 \quad \leq \quad x \leq 1, \ -0 \leq y \leq 1$$

$$|\mathrm{max\ \ error}| \approx 0.03672$$

Barycentric Formula Details

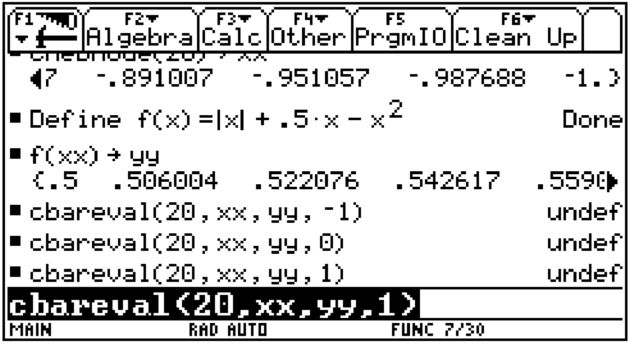
Technically we have replaced a polynomial $P(x)$ of degree at most n with a rational function which is not defined at the nodes $\{x_i\}$. We really should have the following.

$$P(x) = \begin{cases} \dfrac{\sum_{i=0}^{n} \frac{y_i w_i}{(x-x_i)}}{\sum_{i=0}^{n} \frac{w_i}{(x-x_i)}} & \text{if } x \neq x_i \text{ for any } i \\ y_j & \text{if } x = x_j \text{ for some } j \end{cases}$$

Barycentric Formula Details

Technically we have replaced a polynomial $P(x)$ of degree at most n with a rational function which is not defined at the nodes $\{x_i\}$. We really should have the following.

$$P(x) = \begin{cases} \dfrac{\sum_{i=0}^{n} \frac{y_i w_i}{(x-x_i)}}{\sum_{i=0}^{n} \frac{w_i}{(x-x_i)}} & \text{if } x \neq x_i \text{ for any } i \\ y_j & \text{if } x = x_j \text{ for some } j \end{cases}$$



```
 ◀7   -.891007   -.951057   -.987688   -1.}
■ Define f(x)=|x| + .5·x - x²               Done
■ f(xx) → yy
 {.5   .506004   .522076   .542617   .559▶
■ cbareval(20,xx,yy,-1)                    undef
■ cbareval(20,xx,yy,0)                     undef
■ cbareval(20,xx,yy,1)                     undef
cbareval(20,xx,yy,1)
MAIN            RAD AUTO          FUNC 7/30
```

Barycentric Formula Details

Technically we have replaced a polynomial $P(x)$ of degree at most n with a rational function which is not defined at the nodes $\{x_i\}$. We really should have the following.

$$P(x) = \begin{cases} \dfrac{\sum_{i=0}^{n} \frac{y_i w_i}{(x-x_i)}}{\sum_{i=0}^{n} \frac{w_i}{(x-x_i)}} & \text{if } x \neq x_i \text{ for any } i \\ y_j & \text{if } x = x_j \text{ for some } j \end{cases}$$

Barycentric Formula Details

Technically we have replaced a polynomial $P(x)$ of degree at most n with a rational function which is not defined at the nodes $\{x_i\}$. We really should have the following.

$$P(x) = \begin{cases} \dfrac{\sum_{i=0}^{n} \frac{y_i w_i}{(x-x_i)}}{\sum_{i=0}^{n} \frac{w_i}{(x-x_i)}} & \text{if } x \neq x_i \text{ for any } i \\ y_j & \text{if } x = x_j \text{ for some } j \end{cases}$$

This potential "divide by zero" situation causes a problem in some platforms. Berrut and Trefethen used Matlab code that did not need to worry about this. When Matlab comes upon a "divide by zero" situation, it simply assigns "NaN" to indicate "Not a Number." (GNU Octave does this as well). If this happens in one or more components of a vector, Matlab and GNU Octave both will continue to compute all of the rest of the components. Then when you plot, points labeled "NaN" will simply be skipped.

Barycentric Formula Details (cont.)

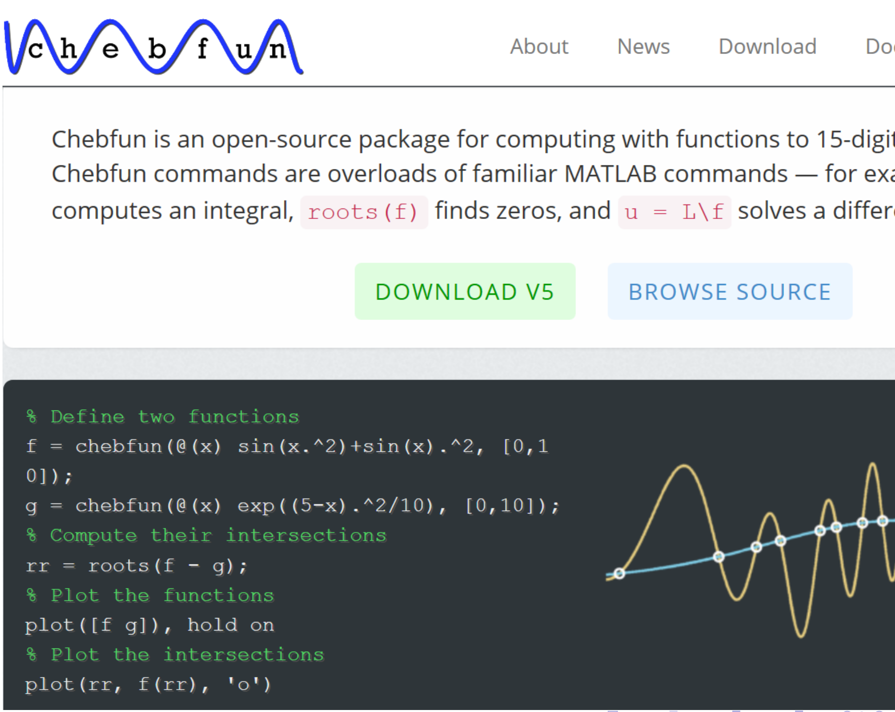
$$P(x) = \begin{cases} \dfrac{\sum_{i=0}^{n} \frac{y_i w_i}{(x-x_i)}}{\sum_{i=0}^{n} \frac{w_i}{(x-x_i)}} & \text{if } x \neq x_i \text{ for any } i \\ y_j & \text{if } x = x_j \text{ for some } j \end{cases}$$

Scilab (and most programming languages) will abort the whole process if a "divide by zero" occurs, and the other components of the vector will not be computed. Thus you need to "check to see if x equals one of the nodes or not" before you evaluate the barycentric formula.

Barycentric Formula Details (cont.)

$$P(x) = \begin{cases} \dfrac{\sum_{i=0}^{n} \frac{y_i w_i}{(x-x_i)}}{\sum_{i=0}^{n} \frac{w_i}{(x-x_i)}} & \text{if } x \neq x_i \text{ for any } i \\ y_j & \text{if } x = x_j \text{ for some } j \end{cases}$$

Scilab (and most programming languages) will abort the whole process if a "divide by zero" occurs, and the other components of the vector will not be computed. Thus you need to "check to see if x equals one of the nodes or not" before you evaluate the barycentric formula.
This is the main difficulty porting some of the algorithms in the Chebfun package consisting of open source code for Mathlab implementing these and many other ideas related polynomial interpolation at Chebyshev nodes. (A URL for this appears in the References.)

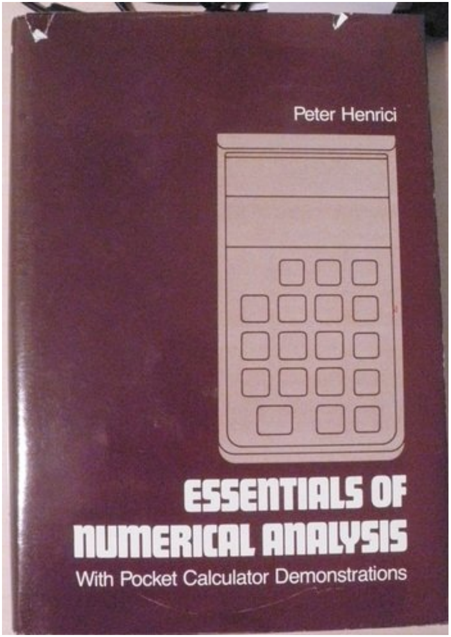


Chebfun is an open-source package for computing with functions to 15-digit
Chebfun commands are overloads of familiar MATLAB commands — for exa
computes an integral, `roots(f)` finds zeros, and `u = L\f` solves a differ

```
% Define two functions
f = chebfun(@(x) sin(x.^2)+sin(x).^2, [0,1
0]);
g = chebfun(@(x) exp((5-x).^2/10), [0,10]);
% Compute their intersections
rr = roots(f - g);
% Plot the functions
plot([f g]), hold on
% Plot the intersections
plot(rr, f(rr), 'o')
```

References

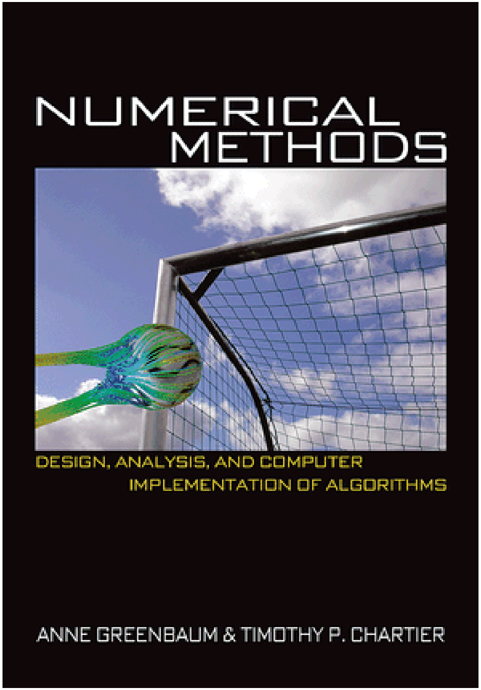
- H. E. Salzer (1972), Lagrange Interpolation at Cheyshev points $x_{n,v} = \cos\left(v\pi/n\right),\ v = 0(1)n$; some unnoted advantages, *Computer J.* 15, 156-159.
- P. Henrici (1982), *Essentials of Numerical Analysis: with Pocket Calculator Demonstrations*, Wiley, New York.
- J.-P. Berrut & L. N. Trefethen (2004), Barycentric Lagrange Interpolation, *Siam Rev.* 46, 501-517.
- A. Greenbaum & T. P. Chartier (2012), *Numerical Methods: Design, Analysis, and Computer Implementation of Algorithms,* Princeton University Press.
- L. N. Trefethen (2013), *Approximation Theory and Approximation Practice*, SIAM, Philadelphia.
  - `http://www.maths.ox.ac.uk/chebfun`



Peter Henrici

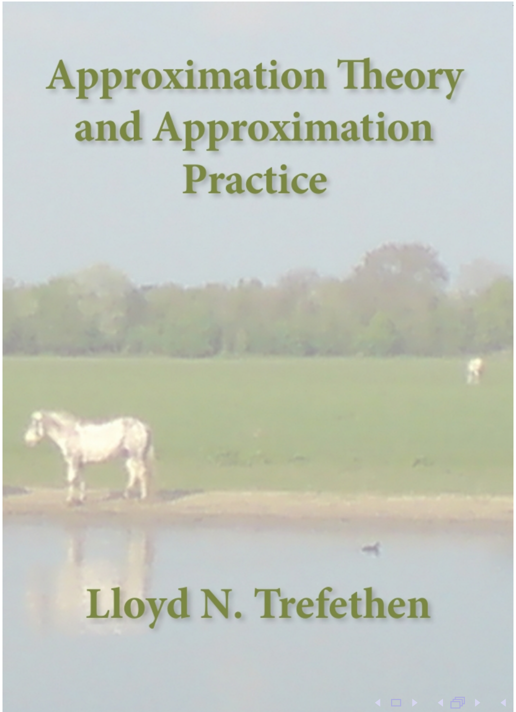
ESSENTIALS OF NUMERICAL ANALYSIS

With Pocket Calculator Demonstrations


NUMERICAL METHODS

DESIGN, ANALYSIS, AND COMPUTER IMPLEMENTATION OF ALGORITHMS

ANNE GREENBAUM & TIMOTHY P. CHARTIER



Approximation Theory and Approximation Practice

Lloyd N. Trefethen