Math 311

Numerical Methods

3.2: Divided Differences

NIDD, NFDF, NBDF, Stirling

S. K. Hyde

Burden and Faires, any ed.

Winter 2024

1 Introduction

- The previous section focused on interpolation at one point. (Neville's Method and Lagrange Polynomials).
- Focus of this section is to generate the polynomials themselves.

n^{th} Lagrange Polynomial in divided difference form

$P(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \cdots + a_n(x - x_0)(x - x_1) \cdots (x - x_{n-1})$, where the constants a_i are solved for.

- We know $P_n(x_k) = f(x_k)$. Let's solve for a_0. If we plug x_0 into $P_n(x)$ we get

$$P_n(x_0) = a_0 = f(x_0) \qquad \Longrightarrow \qquad \boxed{a_0 = f(x_0)}$$

- Let's continue to a_1. Plugging x_1 into $P_n(x)$ yields

$$\begin{aligned} P_n(x_1) = a_0 + a_1(x_1 - x_0) &= f(x_1) \\ f(x_0) + a_1(x_1 - x_0) &= f(x_1) \end{aligned} \quad \Longrightarrow \quad \boxed{a_1 = \frac{f(x_1) - f(x_0)}{x_1 - x_0}}$$

- Let's find a_2. Plugging x_2 into $P_n(x)$ yields

$$P_n(x_2) = a_0 + a_1(x_2 - x_0) + a_2(x_2 - x_0)(x_2 - x_1) = f(x_2)$$

- At this point, it becomes more difficult (but possible) to do the algebra involved.
- However, if we introduce a new notation, it will become a lot easier

Divided Difference Notations

We define a series of recursively generated divided differences beginning with

$$\begin{aligned} f[x_i] &= f(x_i) \\ f[x_i, x_{i+1}] &= \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i} \\ f[x_i, x_{i+1}, x_{i+2}] &= \frac{f[x_{i+1}, x_{i+2}] - f[x_i, x_{i+1}]}{x_{i+2} - x_i} \\ &\vdots \qquad\qquad \vdots \\ f[x_i, x_{i+1}, \cdots, x_{i+k}] &= \frac{f[x_{i+1}, x_{i+2}, \cdots, x_{i+k}] - f[x_i, x_{i+1}, x_{i+k-1}]}{x_{i+k} - x_i} \end{aligned}$$

- With this, we can solve for each of the a_k's easier. So, for example

$$\begin{aligned}
P_2(x) = f(x_2) &= a_0 + a_1(x_2 - x_0) + a_2(x_2 - x_0)(x_2 - x_1) \\
f[x_2] &= f[x_0] + f[x_0, x_1](x_2 - x_0) + a_2(x_2 - x_0)(x_2 - x_1) \\
f[x_2] - f[x_0] - f[x_0, x_1](x_2 - x_0) &= a_2(x_2 - x_0)(x_2 - x_1) \\
\underbrace{f[x_2] - f[x_1]}_{f[x_1,x_2](x_2-x_1)} + \underbrace{f[x_1] - f[x_0]}_{f[x_0,x_1](x_1-x_0)} - f[x_0, x_1](x_2 - x_0) &= a_2(x_2 - x_0)(x_2 - x_1) \\
f[x_1, x_2](x_2 - x_1) + f[x_0, x_1](x_1 - x_0) - f[x_0, x_1](x_2 - x_0) &= a_2(x_2 - x_0)(x_2 - x_1) \\
f[x_1, x_2](x_2 - x_1) - f[x_0, x_1](x_2 - x_1) &= a_2(x_2 - x_0)(x_2 - x_1)
\end{aligned}$$

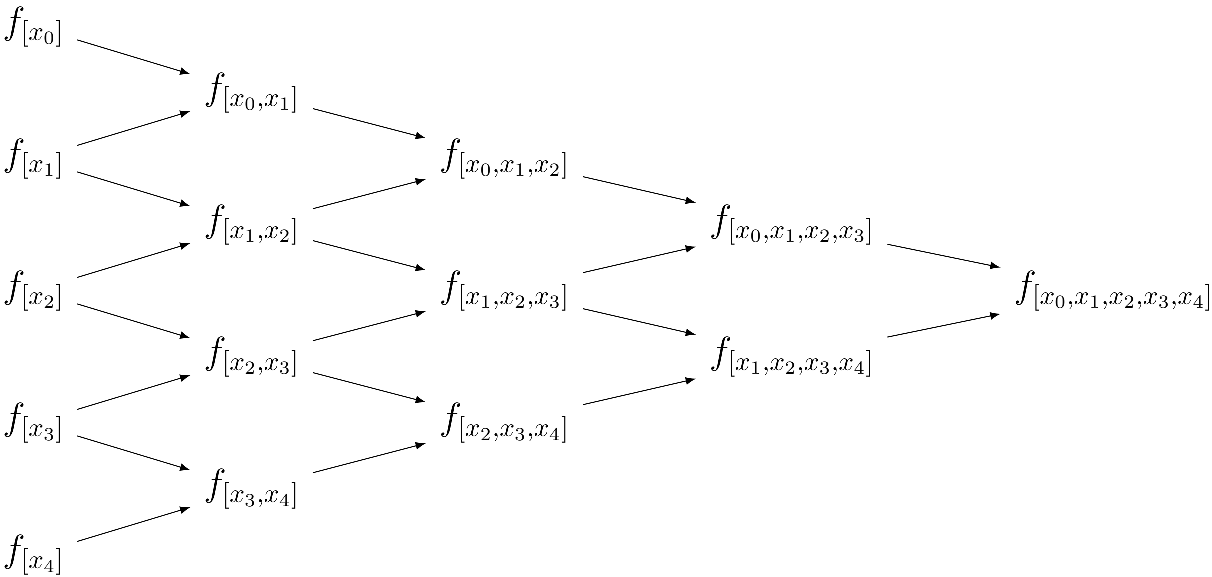
$$\begin{aligned}
a_2 &= \frac{f[x_1, x_2](x_2 - x_1) - f[x_0, x_1](x_2 - x_1)}{(x_2 - x_0)(x_2 - x_1)} \\
a_2 &= \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0}
\end{aligned}$$

$$\boxed{f[x_0, x_1, x_2] = \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0}}$$

- So the polynomial can be written as:

$$
\begin{aligned}
P_n(x) &= a_0 + \sum_{k=1}^{n} a_k (x - x_0)(x - x_1)\cdots(x - x_{k-1}) \\
&= f[x_0] + \sum_{k=1}^{n} f[x_0, x_1, \cdots, x_k](x - x_0)(x - x_1)\cdots(x - x_{k-1})
\end{aligned}
$$

- You can create a Divided Difference Table that looks like this:



- Note that the polynomial coefficients follow the top numbers in the table. All the other numbers are only there to create all the numbers at the top.

X	y	First DD	Second DD	Third DD	Fourth DD
x_0	$f[x_0]$				
		$f[x_0, x_1]$			
x_1	$f[x_1]$		$f[x_0, x_1, x_2]$		
		$f[x_1, x_2]$		$f[x_0, x_1, x_2, x_3]$	
x_2	$f[x_2]$		$f[x_1, x_2, x_3]$		$f[x_0, x_1, x_2, x_3, x_4]$
		$f[x_2, x_3]$		$f[x_1, x_2, x_3, x_4]$	
x_3	$f[x_3]$		$f[x_2, x_3, x_4]$		
		$f[x_3, x_4]$			
x_4	$f[x_4]$				

Newton's Interpolary Divided-Difference Formula

To obtain the divided-difference coeficients of the interpolatory polynomial $P(x)$ on the $(n+1)$ distinct numbers, x_0, $x_1, \cdots, x_n$ for the function $f(x)$:

- Input: numbers $x_0, x_1, \cdots, x_n$, plus $f(x_0), f(x_1), \cdots f(x_n)$ as the first column of the matrix F $(F_{0,0}, F_{1,0}, \cdots, F_{n,0})$
- Output: The numbers $F_{0,0}, F_{1,1}, \cdots, F_{n,n}$, where $P(x) = \sum_{i=0}^{n} F_{i,i} \prod_{j=0}^{i-1} (x - x_j)$.

  Step 1 :

  for $(i = 1 \text{ to } n)$

  for $(j = 1, 2, \cdots, i)$ Set $F_{i,j} = \dfrac{F_{i,j-1} - F_{i-1,j-1}}{x_i - x_{i-j}}$.

  Step 2 output$(F_{0,0}, F_{1,0}, \cdots, F_{n,0})$; STOP.

- This algorithm is fairly easy to implement. However, there are the same issues as for Neville's method since R does not allow arrays that start at 0.
- It's pretty easy to fix:

```
##########################################################################
#
# Newton's Interpolatory Divided Difference Formula -
#   This returns the coefficients of the interpolatory polynomial P where
#   f is stored in table form as A, which contains the x -values and f(x-values)
#      in the first and second columns.
#   It will also evaluate the polynomial at a certain point.
#  written by Scott Hyde
##########################################################################

nidd = function(A,xs) {
    ## Inputs
    ##   A = table of nodes with function values
    ##   xs = value to interpolate from the table
    ##
    ## Note that n below is really n+1 from algorithm. This is because R does
    ##  not use 0 as an index, so everything has to be increased by one (except
    ##  when subtracting i-j, which needs to be increased by 1 as well
    n=dim(A)[1]
    x=A[,1]
    F=matrix(NA,n,n)
    F[,1]=A[,2]

    for (i in 2:n) {
      for (j in 2:i) {
        F[i,j] = (F[i,j-1]-F[i-1,j-1])/(x[i]-x[i-j+1])
      }
    }
    ## The coefficients of the Newton Interpolary Divided Difference Formula are
    ##   the diagonal entries of F
    coef=diag(F)
```

```
    ## The next two lines use the coefficients to figure out the interpolation.
    ## First line creates a vector of the product of x-xj, then the second
    ##   finds the dot product of them.
    xvec=c(1,cumprod(xs-x[-length(x)]))
    interp=sum(coef*xvec)

    ## Next, it names the columns appropriately.
    names(coef)=paste("a",0:(n-1),sep="")
    dimnames(F)=list(x,c("f(x)",paste(ordinal(1:(n-1)),"DD")))
    return(list(table=F,coef=coef,interp=interp))
}

```

MVT applied to $f[x_0, x_1, \cdots, x_n]$

Suppose that $f \in C^n[a,b]$ and $x_0, x_1, \ldots, x_n$ are distginct numbers in $[a,b]$. Then a number ξ in (a,b) exists with

$$f[x_0, x_1, \cdots, x_n] = \frac{f^{(n)}(\xi)}{n!}$$

The theorem uses the Generalized Rolle's Theorem in the proof.

1.1 Arranging $x_0, x_1, \ldots, x_n$ to have EQUAL spacing

- We now want to apply the theory we have to equal spacing of the x values.
- This was historically done because most tables of numbers were equally spaced
- We will reformulate

$$P_n(x) = f[x_0] + \sum_{k=1}^{n} f[x_0, x_1, \cdots, x_k](x - x_0)(x - x_1)\cdots(x - x_{k-1})$$

  with equal spacing.

- So let $h = x_{i+1} - x_i$. We can then reformulate:

$$\begin{aligned} x &= x_0 + sh && \text{(the point to interpolate)} \\ x_i &= x_0 + ih && \text{(any general } x_i\text{)} \\ x - x_i &= (s - i)h && \text{(subtract the two)} \end{aligned}$$

- Now, we can change $(x - x_0)(x - x_1)\cdots(x - x_{k-1})$ into

$$(sh)(s-1)h(s-2)h\cdots(s-(k-1))h = s(s-1)(s-2)\cdots(s-k+1)h^k$$

- For convenience, we will redefine the binomial coefficient for non-integer values of s:

$$\binom{s}{k} = \frac{s(s-1)\cdots(s-k+1)}{k!}, \qquad \text{(where } s \in \mathbb{R}\text{)}$$

- Therefore, we have

$$(x-x_0)(x-x_1)\cdots(x-x_{k-1}) = \binom{s}{k} k! h^k$$

- This gives Newton's Forward Divided Difference Formula (form 1):

Newton's Forward Divided Difference Formula (form 1)

$$P_n(x) = P_n(x_0 + sh) = \sum_{k=0}^{n} f[x_0, x_1, \cdots, x_k] \binom{s}{k} k! h^k$$

- If we use Aitken's Δ^2 operator, we can make a similification to the notation.
- First, note that $\Delta f(x_0) = f(x_1) - f(x_0)$. This means that

$$f[x_0, x_1] = \frac{f(x_1) - f(x_0)}{x_1 - x_0} = \frac{\Delta f(x_0)}{h}$$

- And

$$f[x_0, x_1, x_2] = \frac{f[x_1, x_2] - f[x_0, x_1])}{x_2 - x_0} = \frac{\frac{\Delta f(x_1)}{h} - \frac{\Delta f(x_0)}{h}}{2h} = \frac{\Delta^2 f(x_0)}{2h^2}$$

- In general,

$$f[x_0, x_1, \cdots, x_k] = \frac{1}{k!h^k}\Delta^k f(x_0)$$

- So we can reformulate NFDF

Newton's Forward Divided-Difference Formula (form 2)

$$P_n(x) = P_n(x_0 + sh) = \sum_{k=0}^{n} \binom{s}{k} \Delta^k f(x_0)$$

- You can also reorder the indices from the back to the front. (e.g. $x_n, x_{n-1}, \cdots, x_0$)
- In this case, we get

$$P_n(x) = f[x_n] + \sum_{k=1}^{n} f[x_n, x_{n-1}, \cdots, x_0](x - x_n)(x - x_{n-1}) \cdots (x - x_1)$$

- Using equal spacing (like before) yields

$$P_n(x) = f[x_n] + shf[x_{n-1}, x_n] + s(s+1)h^2 f[x_{n-2}, x_{n-1}, x_n] + s(s+1)\cdots(s+n-1)h^n f[x_0, \cdots, x_n]$$

Backward Difference operator

$$\nabla p_n = p_n - p_{n-1} \qquad \nabla^k p_n = \nabla(\nabla^{k-1} p_n)$$

This makes $f[x_{n-1}, x_n] = \dfrac{\nabla f(x_n)}{n}$, $\quad f[x_{n-2}, x_{n-1}, x_n] = \dfrac{\nabla^2 f(x_n)}{2h^2}$

and in general $f[x_{n-k}, \cdots, x_n] = \dfrac{\nabla^k f(x_n)}{k! h^k}$

- which means that

$$P_n(x) = f[x_n] + s\nabla f(x_n) + \frac{s(s+1)}{2}\nabla^2 f(x_n) + \cdots + \frac{s(s+1)\cdots(s+n-1)}{n!}\nabla^n f(x_n)$$

- Note that the binomial coefficient idea doesn't seem to work because the terms are increasing (instead of decreasing).

- But it really still works! Here's the trick:

$$\frac{s(s+1)\cdots(s+k-1)}{k!} = (-1)^k \frac{(-1)^k s(s+1)\cdots(s+k-1)}{k!} \qquad \text{(mult by 1 trick)}$$

$$= (-1)^k \frac{-s(-s-1)(-s-2)\cdots(-s-k+1)}{k!}$$

(distribute the -1's)

$$= (-1)^k \binom{-s}{k} \qquad \text{(recognize the binomial pattern!)}$$

Newton's Backward Divided-Difference Formula (both forms)

$$P_n(x) = f[x_n] + \sum_{k=1}^{n} \frac{s(s+1)\cdots(s+k-1)}{k!} \nabla^k f(x_n)$$

$$P_n(x) = \sum_{k=0}^{n} (-1)^k \binom{-s}{k} \nabla^k f(x_n)$$

- Note that we now have 4 different formulas for the polynomial that interpolates all the points.

- Each gives the SAME polynomial, but they are used in different places, depending on the situation.
- The NIDD should be used when there is not equal spacing.
- The NFDF should be used when the value of x is close to x_0
- The NBDF should be used when the value of x is close to x_n
- Stirlings formula should be when the value of x is near the center of the table.
- Stirling is part of a category of methods called "centered-difference formulas".
- Difference in notation too. Choose x_0 to be near the point being approximated.
- Call the ones directly below x_0 as x_1, x_2, etc.
- Label the ones above x_0 as x_{-1}, x_{-2}, etc.
- It uses the coefficients in the middle of the table (see Table 3.9 in book)
- Depending on the value of n (odd or even), you use a different formula. Here is the summary of it:

Stirling's Formula ($n = 2m + 1$)

If $n = 2m + 1$ (is odd), then

$$P_n(x) = P_{2m+1}(x) = f[x_0] + \frac{sh}{2}(f[x_{-1}, x_0] + f[x_0, x_1]) + s^2h^2 f[x_{-1}, x_0, x_1]$$
$$+ \frac{s(s^2-1)h^3}{2}(f[x_{-1}, x_0, x_1, x_2] + f[x_{-2}, x_{-1}, x_0, x_1])$$
$$+ s^2(s^2-1)(s^2-4)\cdots(s^2-(m-1)^2)h^{2m} f[x_{-m}, \cdots, x_m]$$
$$+ \frac{s(s^2-1)\cdots(s^2-m^2)h^{2m+1}}{2}(f[x_{-m}, \cdots, x_{m+1}] + f[x_{-m-1}, \cdots, x_m])$$

Stirling's Formula ($n = 2m$)

If $n = 2m$ (is even), then

$$P_n(x) = P_{2m}(x) = f[x_0] + \frac{sh}{2}(f[x_{-1}, x_0] + f[x_0, x_1]) + s^2h^2 f[x_{-1}, x_0, x_1]$$
$$+ \frac{s(s^2-1)h^3}{2}(f[x_{-1}, x_0, x_1, x_2] + f[x_{-2}, x_{-1}, x_0, x_1])$$
$$+ s^2(s^2-1)(s^2-4)\cdots(s^2-(m-1)^2)h^{2m} f[x_{-m}, \cdots, x_m]$$