SIAM Rev.
Vol. 40, No. 3, pp. 685–691, September 1998

CALCULATION OF WEIGHTS IN FINITE DIFFERENCE FORMULAS[*]

BENGT FORNBERG[†]

Abstract. The classical techniques for determining weights in finite difference formulas were either computationally slow or very limited in their scope (e.g., specialized recursions for centered and staggered approximations, for Adams–Bashforth-, Adams–Moulton-, and BDF-formulas for ODEs, etc.). Two recent algorithms overcome these problems. For equispaced grids, such weights can be found very conveniently with a two-line algorithm when using a symbolic language such as Mathematica (reducing to one line in the case of explicit approximations). For arbitrarily spaced grids, we describe a computationally very inexpensive numerical algorithm.

Key words. finite differences, weights, interpolation, linear multistep methods

AMS subject classifications. 65D25, 65D05, 65L12, 65M06

PII. S0036144596322507

1. Introduction.

Derivatives of grid-based functions are often approximated by finite difference approximations. In a few special cases (e.g., for explicit approximations on equispaced grids), the optimal weights are known in closed form. Cases 1 and 2 in Table 1 show two such examples. Cases 3 to 6 illustrate more general situations. The purpose of this note is to present two short algorithms for finding the optimal weights in cases such as these (for derivatives of any order, approximated to any level of accuracy). The first algorithm is limited to equispaced grids. When implemented in a symbolic language (such as Mathematica or Maple), it provides the exact weights for both explicit and implicit approximations. The second algorithm works equally well also for irregular grid spacings. Although it is primarily geared towards explicit approximations, a simple extension permits the derivation also of implicit formulas. Since this second algorithm may need to be run a large number of times (e.g., for each location on an irregular grid), it is presented in a numerical language (Fortran). In the special case of approximating the zeroth derivative, it provides a fast computational procedure for polynomial interpolation.

2. Algorithm for equispaced grids.

One example suffices to illustrate both why this algorithm works and how it is used. Consider for example Case 5 from Table 1. Here, we want to find the coefficients which make the stencil

$$b_0 f''(x-h) + b_1 f''(x) + b_2 f''(x+h) \approx c_0 f(x-h) + c_1 f(x) + c_2 f(x+h)$$

accurate for as high degree polynomials as possible. Substituting $f(x) = e^{i\omega x}$ gives

$$-\omega^2 [b_0 e^{-i\omega h} + b_1 + b_2 e^{i\omega h}] e^{i\omega x} \approx [c_0 e^{-i\omega h} + c_1 + c_2 e^{i\omega h}] e^{i\omega x}.$$

The goal is to make the approximation as accurate as possible, if expanded locally

[*]Received by the editors October 10, 1996; accepted for publication May 12, 1997. This research was supported by NSF (DMS-9706919).
http://www.siam.org/journals/sirev/40-3/32250.html

[†]Department of Applied Mathematics, University of Colorado, Boulder, CO 80309-0526 (fornberg@colorado.edu).

Table 1

Some illustrative examples of commonly occurring types of finite difference approximations.

Case	Description	Example of finite difference formula	Applicable method	Error level of sample formula; Comment
Explicit approximations				
1	centered, regular grid	$f''(x) \approx [-\frac{1}{12}f(x-2h) + \frac{4}{3}f(x-h) - \frac{5}{2}f(x) + \frac{4}{3}f(x+h) - \frac{1}{12}f(x+2h)]/h^2$	$1^\dagger$, 2	$O(h^4)$
2	staggered, regular grid	$f'(x) \approx [\frac{1}{24}f(x-\frac{3}{2}h) - \frac{9}{8}f(x-\frac{1}{2}h) + \frac{9}{8}f(x+\frac{1}{2}h) - \frac{1}{24}f(x+\frac{3}{2}h)]/h$	$1^\dagger$, 2	$O(h^4)$
3	one-sided, regular grid	$f'(x+h) \approx [\frac{1}{4}f(x-3h) - \frac{4}{3}f(x-2h) + 3f(x-h) - 4f(x) + \frac{25}{12}f(x+h)]/h$	$1^\dagger$, 2	$O(h^4)$; example of backward differentiation (BDF) method for ODEs
4	partly one-sided, irregular grid	$f'''(\frac{1}{2}) \approx \frac{13}{10}f(0) - \frac{5832}{1105}f(\frac{1}{3}) + \frac{42}{5}f(1) - \frac{177}{20}f(2) + \frac{288}{65}f(\frac{7}{2}) - \frac{1}{1340}f(6)$	2	$O(h^3)$ if grid spacing proportional to h
Implicit approximations				
5	centered, regular grid	$1f''(x-h) + 10f''(x) + 1f''(x+h) \approx [12f(x-h) - 24f(x) + 12f(x+h)]/h^2$	1, 2^*	$O(h^4)$; example of Collatz' "Mehrstellenverfahren"
6	one-sided, regular grid	$-\frac{3}{8}f'(x-3h) + \frac{37}{24}f'(x-2h) - \frac{59}{24}f'(x-h) + \frac{55}{24}f'(x) \approx [-f(x) + f(x+h)]/h$	1, 2^*	$O(h^4)$; example of Adams-Bashforth method for ODEs (Adams–Moulton methods are obtained similarly)

Notes: $1^\dagger$ indicates that also the briefer Taylor expansion method is applicable.
2^* denotes method 2 with "fictitious point" extension.

around $h = 0$. Canceling the factors $e^{i\omega x}$, and substituting $e^{i\omega h} = \xi$, i.e., $i\omega h = \ln \xi$, gives

$$\left\{\frac{\ln \xi}{h}\right\}^2 \left[\frac{b_0}{\xi} + b_1 + b_2\xi\right] \approx \left[\frac{c_0}{\xi} + c_1 + c_2\xi\right];$$

$$\left\{\frac{\ln \xi}{h}\right\}^2 \approx \frac{c_0 + c_1\xi + c_2\xi^2}{b_0 + b_1\xi + b_2\xi^2}.$$

At this point, we want the best possible accuracy when expanded around $\xi = 1$. Padé approximation of $((\ln \xi)/h)^2$ around $\xi = 1$, to order $[2, 2]$, will offer this

$$\left\{\frac{\ln \xi}{h}\right\}^2 \approx \frac{(\xi-1)^2}{h^2(1 + (\xi - 1) + \frac{1}{12}(\xi-1)^2)} = \frac{12 - 24\xi + 12\xi^2}{h^2(1 + 10\xi + 1\xi^2)}.$$

Similar Padé approximations have been used in the past for analysis of linear multistep methods; see for example [5]. (Padé coefficients can be calculated rapidly from Taylor-coefficients; see [1].)

In Mathematica (after having loaded the Padé package through `<<Calculus‘Pade‘`), we get the desired weights by executing the two statements

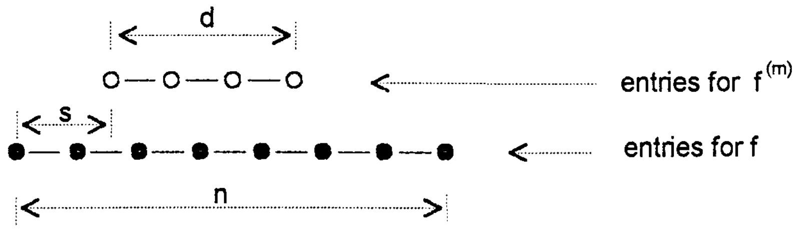

FIG. 1. *Schematic illustration of how the notation used in the Padé weight algorithm relates to a stencil shape; here shown in a staggared case with* $s = \frac{3}{2}$, $d = 3$, *and* $n = 7$.

```
t=Pade[Log[x]^2,{x,1,2,2}];
{CoefficientList[Denominator[t],x],CoefficientList[Numerator[t],x]/h^2}
```

For more general finite difference approximations, we similarly use

```
t=Pade[x^s*Log[x]^m,{x,1,n,d}];
{CoefficientList[Denominator[t],x],CoefficientList[Numerator[t],x]/h^m}
```

where

m	order of derivative to be approximated,
s	number of grid intervals in between the leftmost derivative and function entries; select the sign "+" if the former is to the right of the latter, else the sign "−,"
d	number of grid intervals in between the left- and rightmost derivative entries, and
n	number of grid intervals in between the left- and rightmost function entries;

cf. Figure 1.

In Maple, the appropriate package is loaded by the command `with (numapprox):`, and the full code for the general case becomes

```
t:=pade (x^s*ln(x)^m,x=1,[n,d]):
coeff (expand (denom(t)),x,i)      $i=0..d;
coeff (expand (numer(t)),x,i)/h^m  $i=0..n;
```

The three first test cases in Table 1 are explicit. The Padé denominator is then of degree $d = 0$ and a simpler Taylor expansion can be used instead (this special case was noted in [2]). No packages need now be preloaded, and the code reduces to one single line, as follows:

In Mathematica:
```
CoefficientList[Normal[Series[x^s*Log[x]^m,{x,1,n}]/h^m],x]
```

In Maple:
```
coeff(expand(convert(taylor(x^s*ln(x)^m,x=1,n+1),polynom)),
     x,i)/h^m $i=0..n;
```

To apply this algorithm to the test cases in Table 1, we choose the parameters as shown in Table 2.

3. Algorithm for arbitrarily spaced grids.

The following algorithm was first outlined in pseudocode in 1988 [3]:

TABLE 2
Input parameters to the Mathematica or Maple algorithms for the different test cases.

Case	`m =`	`s =`	`n =`	`d =`	Comment
1	2	2	4	0	Padé or Taylor
2	1	3/2	3	0	-”-
3	1	4	4	0	-”-
4	irregular grid - the algorithm is not applicable				- - - - - - - - - -
5	2	0	2	2	Padé only
6	1	−3	1	3	-”-

Given:

z	location at which we want to approximate the derivative (may but need not be a grid point),
$x_0, x_1, \ldots, x_n$	grid point locations which the stencil is to extend over (distinct, otherwise arbitrary),
n	one less than the number of grid points, and
m	highest derivative for which weights are sought,

`subroutine weights` returns

$c_0^k, c_1^k, \ldots, c_n^k$ the optimal weights in approximations of the form

$$\left[\frac{d^k f}{dx^k}\right]_{x=z} \approx \sum_{i=0}^{n} c_i^k f(x_i), \qquad k = 0, 1, \ldots, m.$$

```
      subroutine weights (z,x,n,nd,m,c)
c---
c---   Input Parameters
c---        z           location where approximations are to be accurate,
c---        x(0:nd)      grid point locations, found in x(0:n)
c---        n           one less than total number of grid points; n must
c---                    not exceed the parameter nd below,
c---        nd          dimension of x- and c-arrays in calling program
c---                    x(0:nd) and c(0:nd,0:m), respectively,
c---        m           highest derivative for which weights are sought,
c---   Output Parameter
c---        c(0:nd,0:m)  weights at grid locations x(0:n) for derivatives
c---                    of order 0:m, found in c(0:n,0:m)
c---
      implicit real*8 (a-h,o-z)
      dimension x(0:nd),c(0:nd,0:m)
      c1 = 1.0d0
      c4 = x(0)-z
      do 10 k=0,m
         do 10 j=0,n
   10       c(j,k) = 0.0d0
      c(0,0) = 1.0d0
      do 50 i=1,n
         mn = min(i,m)
         c2 = 1.0d0
         c5 = c4
         c4 = x(i)-z
         do 40 j=0,i-1
            c3 = x(i)-x(j)
            c2 = c2*c3
            if (j.eq.i-1) then
               do 20 k=mn,1,-1
20                c(i,k) = c1*(k*c(i-1,k-1)-c5*c(i-1,k))/c2
               c(i,0) = -c1*c5*c(i-1,0)/c2
            endif
            do 30 k=mn,1,-1
30             c(j,k) = (c4*c(j,k)-k*c(j,k-1))/c3
40          c(j,0) = c4*c(j,0)/c3
50       c1 = c2
      return
      end
```

The following is a brief summary of its derivation.

Given the data values u_i at the locations x_i, $i = 0,\, 1, \ldots, n$, the Lagrange interpolation polynomial based on the first $j+1$ function values $u_i = u(x_i)$, $i = 0, 1, \ldots, j$ becomes

$$p_j(x) = \sum_{i=0}^{j} L_{i,j}(x) u_i, j = 0, 1, \ldots, n,$$

where

$$L_{i,j}(x) = \frac{(x-x_0)\cdots(x-x_{i-1})(x-x_{i+1})\cdots(x-x_j)}{(x_i-x_0)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_j)}. \tag{1}$$

Assuming for simplicity that we want the approximations to be accurate at $x = 0$, we get

$$\left.\frac{d^k u(x)}{dx^k}\right|_{x=0} \approx \left.\frac{d^k p_j(x)}{dx^k}\right|_{x=0} = \sum_{i=0}^{j} \left.\frac{d^k L_{i,j}(x)}{dx^k}\right|_{x=0} \cdot u_i = \sum_{i=0}^{j} c_{i,j}^k \cdot u_i.$$

(The second subscript for $c_{i,j}^k$ denotes the stencil width; we abbreviate $c_{i,n}^k$ as c_i^k). By Taylor's formula

$$L_{i,j}(x) = \sum_{k=0}^{j} \left.\frac{d^k L_{i,j}(x)}{dx^k}\right|_{x=0} \cdot \frac{x^k}{k!} = \sum_{k=0}^{j} c_{i,j}^k \frac{x^k}{k!}, \tag{2}$$

i.e., the weights $c_{i,j}^k$ can be read off from the Taylor coefficients of $L_{i,j}(x)$. Equation (1) implies the recursion relations

$$L_{i,j}(x) = \frac{(x-x_j)}{(x_i-x_j)} L_{i,j-1}(x)$$

and

$$L_{j,j}(x) = \left\{ \frac{\Pi_{\nu=0}^{j-2}(x_{j-1}-x_\nu)}{\Pi_{\nu=0}^{j-1}(x_j-x_\nu)} \right\} (x-x_{j-1}) L_{j-1,j-1}(x).$$

TABLE 3
Input parameters to `subroutine weights` *for the different test cases.*

Case	`z =`	`x(0:*) =`	`n =`	`nd≥`	`m =`
1	0	$-2h,\ -h,\ 0,\ h,\ 2h$	4	4	2
2	0	$-\frac{3}{2}h,\ -\frac{1}{2}h,\ \frac{1}{2}h,\ \frac{3}{2}h$	3	3	1
3	h	$-3h,\ -2h,\ -h,\ 0,\ h$	4	4	1
4	$\frac{1}{2}$	$0,\ \frac{1}{3},\ 1,\ 2,\ \frac{7}{2},\ 6$	5	5	3
5	implicit approximation - use "fictitious point" extension				
6	-"-				

Substitution of the Taylor series (2) into these leads to the following recursion relations for its coefficients:

$$c^k_{i,j} = \frac{1}{x_j - x_i}\left(x_j c^k_{i,j-1} - k c^{k-1}_{i,j-1}\right), \tag{3}$$

$$c^k_{j,j} = \left\{ \frac{\Pi_{\nu=0}^{j-2}(x_{j-1} - x_\nu)}{\Pi_{\nu=0}^{j-1}(x_j - x_\nu)} \right\} \left(k c^{k-1}_{j-1,j-1} - x_{j-1} c^k_{j-1,j-1}\right). \tag{4}$$

Starting from the trivial $c^0_{0,0} = 1$ (and assuming any undefined weights to be zero), all the required weights $c^k_{i,j}$ follow recursively from (3) and (4). (The ratios of the products in (4) are also easily evaluated recursively.)

The following are some notes regarding the implementation of the algorithm:

- Although the algorithm can be shown to be numerically robust, *applying* a difference stencil to smooth data can lead to cancellation of significant digits (especially when approximating high derivatives).
- A call to `weights` to obtain the coefficients for the mth derivative returns also the coefficients for the kth derivative, $k = 0,\ 1, \ldots, m$. These are "byproducts," available at no additional computational cost.
- The code can be modified to return all the data above also for stencils which extend only over $x_0,\ x_1, \ldots, x_j,\ j = 0,\ 1, \ldots, n$—still at no additional cost [4]. (This version is actually slightly shorter than the present one; c is then a 3-D array, and less care needs to be taken to prevent premature overwritings within it.)
- Calling `weights` with $m = 0$ gives the weights for polynomial interpolation at a cost (to leading order) of $2n^2$ floating point operations. This can be compared to $1.5n^2$ operations to obtain Newton's divided differences and $2.5n^2$ operations for interpolation using the well-known algorithms by Aitken and Neville. Particularly large savings are realized if several functions are to be interpolated on the same grid; the weights can then be reused at a cost of only $2n$ operations per case (this situation will arise already for a single data function if it is given on a 2-D or 3-D Cartesian grid).

To apply `subroutine weights` to the examples in Table 1, we call it with input parameters as seen in Table 3.

4. Extension to implicit formulas. Again, one case suffices to illustrate the idea (described in [4] under "fictitious gridpoints" as a tool to incorporate boundary information with pseudospectral methods). To obtain the weights in Case 5 (assuming $h = 1$ to keep the notation simple), let the grid points be located at $x = -1,\ 0,\ 1$. Next, we introduce two more points *anywhere*, say at $x = 2,\ 3$ (requiring only that the five points are distinct, and preferably not too closely clustered). The top section in

Table 4
Generation of weights for Case 5 *using* `subroutine weights` *with two fictitious points.*

$f''(-1)$	$f''(0)$	$f''(1)$		$f(-1)$	$f(0)$	$f(1)$	$f(2)$	$f(3)$
1			$=$	$\frac{35}{12}$	$-\frac{26}{3}$	$\frac{19}{2}$	$-\frac{14}{3}$	$\frac{11}{12}$
	1		$=$	$\frac{11}{12}$	$-\frac{5}{3}$	$\frac{1}{2}$	$\frac{1}{3}$	$-\frac{1}{12}$
		1	$=$	$-\frac{1}{12}$	$\frac{4}{3}$	$-\frac{5}{2}$	$\frac{4}{3}$	$-\frac{1}{12}$
Adding multiples 1, 10, 1 of the three rows above give								
1	10	1	$=$	12	-24	12	0	0

Table 4 is obtained by three separate calls to `subroutine weights`. These three lines can then be linearly combined to produce zero weights at the fictitious points—thus producing the desired stencil.

Compared to the "method of undetermined coefficients" (in this case, requiring a formula $1 \cdot f''(-1) + \alpha_1 f''(0) + \alpha_2 f''(1) - \alpha_3 f(-1) - \alpha_4 f(0) - \alpha_5 f(1) = 0$ to be exact for $f = 1,\ x,\ x^2,\ x^3,\ x^4$ and then solving a linear system for α_i, $i = 1, \ldots, 5$), the fictitious point approach leads to smaller and typically better conditioned systems. Very general implicit formulas can be obtained in this manner; different derivatives can be present at same or different irregularly spaced points, etc.

Acknowledgment. Discussions with Professor André Weideman are gratefully acknowledged.

REFERENCES

[1] C. M. Bender and S. A. Orszag, *Advanced Mathematical Methods for Scientists and Engineers*, McGraw–Hill, New York, 1978.

[2] J. K. Cohen and D. R. De Baun, *Discrete approximation of linear functionals*, Mathematica Journal, 2, Issue 2 (1992), pp. 62–65.

[3] B. Fornberg, *Generation of finite difference formulas on arbitrarily spaced grids*, Math. Comput., 51 (1988), pp. 699–706.

[4] B. Fornberg, *A Practical Guide to Pseudospectral Methods*, Cambridge University Press, Cambridge, UK, 1996.

[5] A. Iserles, *First Course in Numerical Analysis of Differential Equations*, Cambridge University Press, Cambridge, UK, 1996.