1
Introduction to Finite Difference Methods

1.1 A Few Historical Notes

Finite difference (FD)-type discrete approximations can be traced back much earlier than when Gottfried Leibniz[1] and Isaac Newton[2] gave the first descriptions of calculus (in 1684 and 1687, respectively). The introduction of FDs (at first for interpolation) is often attributed to Jost Bürgi,[3] around 1592. Finite difference formulas of high orders of accuracy (especially for integration) were used by James Gregory in 1670.[4] Significant further early FD perspectives were provided by Isaac Newton and later by Brook Taylor[5] in 1715. Finite difference formulas were quite widely used for solving ordinary differential equations (ODEs) in the nineteenth century (notably for problems in fluid mechanics and for planetary orbit calculations). The pioneering work on the use of FD for partial differential equations (PDEs) dates back to the study by Lewis F. Richardson (1911)[6] (on potentially dangerous stresses in the first dam over the Nile in Aswan).

1.2 Finite Difference Formulas

The standard definition of the first derivative

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \tag{1.1}$$

[1] 1646–1716, German mathematician, philosopher, scientist, and diplomat.
[2] 1643–1727, English mathematician, physicist, astronomer, alchemist, and theologian; widely recognized as one of the greatest scientists of all time.
[3] 1552–1632, Swiss mathematician, also a maker of clocks and astronomical instruments.
[4] Described further in Section 7.3.
[5] 1685–1731, English mathematician, of Taylor series fame.
[6] 1881–1953, English mathematician and pioneer of numerical weather forecasting.

is a simple example of an FD formula. Taylor expansion of (1.1) shows that

$$\frac{f(x+h)-f(x)}{h} = f'(x)+\frac{h}{2!}f''(x)+\frac{h^2}{3!}f'''(x)+\cdots = f'(x)+O(h^1), \quad (1.2)$$

that is, the approximation $f'(x) \approx \frac{f(x+h)-f(x)}{h}$ is accurate to *first order* (at the location x). The FD *weights* at the *nodes* x and $x+h$ can be abbreviated as $[-1 \;\; 1] \;/\; h$. We will soon find it convenient to sketch FD *stencils* graphically. In this case:

$$\begin{array}{cc} \circ & \leftarrow \text{ entry for } f', \text{ weight } \{1\}, \\ \square \quad \square & \leftarrow \text{ entries for } f, \text{ weights } \{-\frac{1}{h}, \; \frac{1}{h}\}, \\ \uparrow \quad \uparrow & \\ x \quad x+h & \leftarrow \text{ spatial locations.} \end{array} \quad (1.3)$$

The circle indicates the location in x for a derivative entry, and the squares indicate x-locations for function values (vertical spacing is here of no significance). Each of these locations has a *weight* associated with it. While the simplicity of this approximation is convenient (it uses only two adjacent function values to produce a derivative approximation), its low order of accuracy (first order; exact only for polynomials up through degree one) makes it almost entirely useless for practical computing.

A significantly better approximation to the first derivative is provided by

$$f'(x) = \frac{-\frac{1}{2}f(x-h)+\frac{1}{2}f(x+h)}{h} + O(h^2). \quad (1.4)$$

Figure 1.1 illustrates the two FD approximations (1.2) and (1.4). Although both give the same approximation for the tangent slope in the limit $h \to 0$, it is clear also visually that the second-order approximation is the more accurate one for small but nonzero h.

Taylor expansions provide a helpful means of verifying the order of accuracy if an FD formula is proposed. However, there are numerous more convenient ways to generate such formulas, as described in Section 1.2.1. Since the value of x in (1.2) and (1.4) – and in general for FD formulas – has no influence on the weights, we simplify the notation in Section 1.2.1 by approximating derivatives at $x = 0$.

1.2.1 Some Algorithms for Generating FD Weights

In all but the last of the five methods described in Sections 1.2.1.1–1.2.1.5, the independent variable x can just as well be a complex variable z (with node points distributed in the complex plane rather than along the real axis).

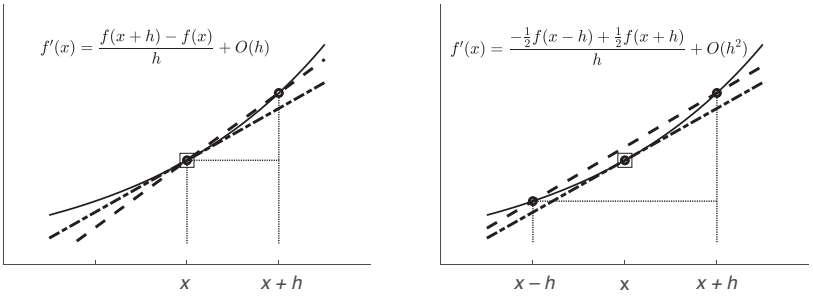


Figure 1.1 Graphical comparison between the first- and second-order approximations for $f'(x)$, as given by (1.2) and (1.4), respectively. The slope of the tangent lines (dash-dotted) corresponds to the true derivative value at the location x, and the slopes of the secant lines (dashed) correspond to the respective approximations.

1.2.1.1 FD Weights by Use of Monomial Test Functions

This algorithm is very flexible. Let L be any derivative operator (such as $\frac{d}{dx}$ and $\frac{d^2}{dx^2}$). We require a stencil with n nodes, located at $x_1, x_2, \ldots, x_n$ (distinct) to be exact for the first n monomials $1, x, x^2, \ldots, x^{n-1}$. This requirement can be written as

$$\begin{bmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_n \\ \vdots & \vdots & & \vdots \\ x_1^{n-1} & x_2^{n-1} & \cdots & x_n^{n-1} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} = \begin{bmatrix} L\,1|_{x=0} \\ L\,x|_{x=0} \\ \vdots \\ L\,x^{n-1}|_{x=0} \end{bmatrix}. \tag{1.5}$$

The weights w_k at nodes x_k, $k = 1, 2, \ldots, n$, can now be obtained by solving this linear system. The matrix is of Vandermonde type – often ill-conditioned, but never singular assuming the x_k are distinct.[7]

Verification of nonsingularity: The result is true for $n = 1$, in which case the matrix is $[1]$. Applying induction over n, assume the result is true up through size $n - 1$, and consider the size n case. Call temporarily $x_n = x$ and the matrix $A(x)$. Expanding along the last column, $\det(A(x))$ becomes a polynomial in x of degree $n - 1$, not identically zero (as, by the induction hypothesis, the coefficient for x^{n-1} is nonzero). All its $n - 1$ roots are accounted for by $x = x_1$, $x = x_2, \ldots, x = x_{n-1}$. It must therefore be nonzero for $x = x_n$.

Example 1.2.1 Create an FD formula that approximates $f''(0)$ based on function values $f(x_1)$ and $f(x_2)$ and first derivative values $f'(x_3)$ and $f'(x_4)$.

[7] A different proof to the one immediately following is given in Section A.2.

Solution The monomial test function algorithm generalizes immediately to cases with mixed types of input data. Enforcing

$$f''(0) \approx w_1 f(x_1) + w_2 f(x_2) + w_3 f'(x_3) + w_4 f'(x_4)$$

to be exact for the functions $1, x, x^2, x^3$ gives the following linear system to solve for the weights:

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ x_1 & x_2 & 1 & 1 \\ x_1^2 & x_2^2 & 2x_3 & 2x_4 \\ x_1^3 & x_2^3 & 3x_3^2 & 3x_4^2 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 2 \\ 0 \end{bmatrix}.$$

The determinant of the coefficient matrix becomes

$$(x_1 - x_2)(x_3 - x_4)(2x_1^2 + 2x_1x_2 + 2x_2^2 - 3x_1x_3 - 3x_2x_3 - 3x_1x_4 - 3x_2x_4 + 6x_3x_4).$$

The first two factors tell that the system is singular in the case of coinciding nodes $x_1 = x_2$ or $x_3 = x_4$. The remaining quadratic form is not positive or negative definite,[8] implying possibilities of additional singularities for certain node locations. □

1.2.1.2 FD Weights by the Method of Exponential Test Functions

This method leads to the same linear system (1.5). However, it simplifies certain tasks (cf. Sections 7.5.1 and 8.3.1). We now apply L to the test function $e^{\xi x}$ rather than to monomials in x. Equating leading Taylor expansion terms in ξ in $\sum_{k=1}^{n} w_k e^{\xi x_k} = L\, e^{\xi x}\big|_{x=0}$ becomes, when written out more explicitly,

$$\left\{ \begin{array}{l} w_1 \left(1 + \xi x_1 + \frac{\xi^2 x_1^2}{2!} + \cdots\right) + \\ w_2 \left(1 + \xi x_2 + \frac{\xi^2 x_2^2}{2!} + \cdots\right) + \\ \qquad \vdots \\ w_n \left(1 + \xi x_n + \frac{\xi^2 x_n^2}{2!} + \cdots\right) \end{array} \right\} = L \left(1 + \xi x + \frac{\xi^2 x^2}{2!} + \cdots\right)\Big|_{x=0},$$

again giving the linear system (1.5).

[8] Seen, for example, by noting that it is linear in x_4 (and x_3). For any choice of x_1, x_2, x_3, it evaluates to zero for $x_4 = \frac{2x_1^2 + 2x_1x_2 + 2x_2^2 - 3x_1x_3 - 3x_2x_3}{3(x_1 + x_2 - 2x_3)}$. More generally, writing it as

$$[x_1, x_2, x_3, x_4] \begin{bmatrix} 2 & 1 & -3/2 & -3/2 \\ 1 & 2 & -3/2 & -3/2 \\ -3/2 & -3/2 & 0 & 3 \\ -3/2 & -3/2 & 3 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix},$$

it is seen from the fact that the (symmetric) matrix has both positive and negative eigenvalues ($\lambda_1 = 6, \lambda_2 = 1, \lambda_3 = 0, \lambda_4 = -3$). However, if $x_3 = x_1$ and $x_4 = x_2$ (then a case of Hermite interpolation, cf. Section 1.4.3), the quadratic form evaluates to $-(x_1 - x_2)^2$, ensuring nonsingularity for $x_1 \neq x_2$.

1.2.1.3 FD Weights by Lagrange's Interpolation Formula

This approach is good for theoretical insight but less convenient to code. Consider, for example, deriving (1.4). The *Lagrange interpolation polynomial*[9] $p(x)$ that takes the values $f(-h), f(0), f(h)$ at $x = -h, 0, h$ is

$$p(x) = \frac{(x-0)(x-h)}{(-h-0)(-h-h)} f(-h) + \frac{(x+h)(x-h)}{(0+h)(0-h)} f(0) + \frac{(x+h)(x+0)}{(h+h)(h+0)} f(h).$$

A quick calculation shows that $p'(0) = \frac{-\frac{1}{2}f(x-h)+\frac{1}{2}f(x+h)}{h}$, in agreement with (1.4).

1.2.1.4 FD Weights by Recursion

This algorithm is computationally very fast and has excellent numerical stability. It follows from Lagrange's interpolation formula – see Fornberg (1988a) for details. We give it here only in the form of the MATLAB code shown in Algorithm 1.1. For example, the statement **weights**$(\mathbf{0}, -\mathbf{2} : \mathbf{2}, \mathbf{6})$ returns the output

```
         0         0    1.0000         0         0
    0.0833   -0.6667         0    0.6667   -0.0833
   -0.0833    1.3333   -2.5000    1.3333   -0.0833
   -0.5000    1.0000         0   -1.0000    0.5000
    1.0000   -4.0000    6.0000   -4.0000    1.0000
         0         0         0         0         0
         0         0         0         0         0
```

The top line gives the weights for interpolation, which here is trivial as the interpolation point coincides with one of the grid points. The next four rows give the optimal weights for five-node approximations to derivatives of orders 1 to 4.[10] The last two rows are all zero, as derivatives of orders 5 and higher cannot be approximated based on just five nodes.

1.2.1.5 FD Weights by Padé-Based Algorithm

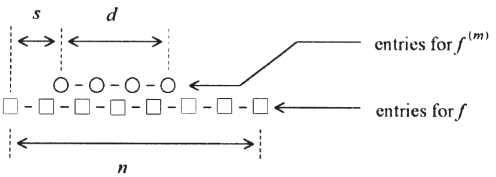
This algorithm was introduced in Fornberg (1998). Its derivation is a straightforward generalization of the special case described in Example 1.2.5. It requires equispaced nodes but is otherwise very general. It readily applies also to *implicit stencils*,[11] and it naturally uses exact rather than floating-point arithmetic. In the display form introduced in (1.3), we consider now stencils of the more general form

[9] Described in Appendix A, with illustration in Figure 1.3.
[10] Thus, lines 2 and 3 match the second row in Tables 1.1 and 1.2, respectively.
[11] Also described as *compact stencils*; see also Sections 3.3.2 and 4.3.1.1.

Algorithm 1.1 A MATLAB implementation of the weights algorithm in Section 1.2.1.4.

```
function c = weights(z,x,m)
%  Calculates FD weights.
% Input parameters:
%  z Location where approximations are to be accurate,
%  x Row vector with x-coordinates for grid points (distinct but
%    otherwise arbitrarily located),
%  m Highest derivative that we want to find weights for
% Output parameter:
%  c Matrix, size (m+1,length(x)), containing in successive rows the
%    weights for derivatives 0,1,...,m.

n = length(x); c = zeros(m+2,n); c(2,1) = 1; x1 = repmat(x,n,1);
A = x1.'-x1;
b = cumprod([ones(n,1),A],2); rm = cumsum(ones(m+2,n-1))-1;
d = diag(b); d(1:n-1) = d(1:n-1)./d(2:n);
for i = 2:n
   mn = min(i,m+1);
   c(2:mn+1,i) = d(i-1)*(rm(1:mn,1).*c(1:mn,i-1)-(x(i-1)-z)* ...
   c(2:mn+1,i-1));
   c(2:mn+1,1:i-1) = ((x(i)-z)*c(2:mn+1,1:i-1)-rm(1:mn,1:i-1).* ...
   c(1:mn,1:i-1))./(x(i)- x1(1:mn,1:i-1));
end
c(1,:) = [];
end
```



relating weights for the mth derivative at $d + 1$ locations spaced h apart with weights for the function at $n + 1$, also h-spaced locations, with the two sets shifted sideways by s units of h, where s need not be an integer. In symbolic languages, such as Mathematica here, just two lines of code suffice[12]:

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

The following examples illustrate how this algorithm can be used and (Example 1.2.5) also the approach for deriving it.

[12] The present usage of Padé expansions differs significantly from their more common applications to convergence acceleration and numerical analytic continuation of both Taylor and asymptotic expansions; see Section E.3 and, for example, Bender and Orszag (1978), Sections 8.3–8.6; Fornberg and Piret (2020), Section 3.2.9; and Trefethen (2013), Chapter 27.

Example 1.2.2 Find the weights in a stencil of the shape $\square\ \square\ \overset{\circ}{\square}\ \square$ for approximating the second derivative.

Solution For $s = 1,\ d = 0,\ n = 2,\ m = 2$, the algorithm returns

$$\left\{\left\{h^2\right\}, \{1, -2, 1\}\right\},$$

corresponding to the explicit second-order accurate formula for the second derivative

$$f''(x) \approx \{f(x-h) - 2f(x) + f(x+h)\}\,\frac{1}{h^2}\,. \tag{1.6}$$

□

Example 1.2.3 Find the weights in a stencil of the shape $\square\ \overset{\circ}{\square}\ \overset{\circ}{\square}\ \overset{\circ}{\square}$, again for approximating the second derivative.

Solution For $s = 0,\ d = 2,\ n = 2,\ m = 2$, the algorithm returns

$$\left\{\left\{\frac{h^2}{12}, \frac{5h^2}{6}, \frac{h^2}{12}\right\}, \{1, -2, 1\}\right\},$$

corresponding to the compact (implicit) fourth-order accurate formula for the second derivative

$$\begin{aligned}\frac{1}{12}f''(x-h) + \frac{5}{6}f''(x) + \frac{1}{12}f''(x+h)\\ \approx \{f(x-h) - 2f(x) + f(x+h)\}\,\frac{1}{h^2}\,.\end{aligned} \tag{1.7}$$

□

Example 1.2.4 Find the weights in a stencil of the shape $\overset{\circ}{\square}\ \overset{\circ}{\square}\ \overset{\circ}{\square}$ for approximating the first derivative.

Solution For $s = -2,\ d = 2,\ n = 1,\ m = 1$, the output

$$\left\{\left\{\frac{5h}{12}, -\frac{4h}{3}, \frac{23h}{12}\right\}, \{-1, 1\}\right\}$$

is readily rearranged into

$$f(x+h) = f(x) + \frac{h}{12}(23f'(x) - 16f'(x-h) + 5f'(x-2h)). \tag{1.8}$$

We later (in Section 3.2.3) encounter this formula as the third-order Adams–Bashforth method for solving ODEs. □

Example 1.2.5 For Example 1.2.3, obtain the weights by explicitly carrying out the exponential test function approach.

Algorithm 1.2 A MATLAB implementation of the weights algorithm in Section 1.2.1.5.

```
function [w_f,w_der] = weights_Pade(s,d,n,m)
% Input parameters (type double or symbolic); d,n,m must be integers;
% If s is non-integer, symbolic form for s ensures exact arithmetic.
%  s,d,n  Define stencil shape
%      m  Order of derivative approximated
% Output parameters (symbolic variables)
%    w_f  Weight(s) for function values
%  w_der  Weight(s) for derivative values
syms x h;  [N,D] = numden(pade(x^s*log(x)^m,x,1,'Order',[n,d]));
w_f = fliplr(coeffs(N,'All'));  w_der = fliplr(coeffs(D,'All'))*h^m;
end
```

Solution The approximation should be of the form

$$b_{-1}f''(x-h)+b_0f''(x)+b_1f''(x+h)\approx c_{-1}f(x-h)+c_0f(x)+c_1f(x+h).$$

With $f=e^{\xi x}$, this becomes

$$\xi^2\left(b_{-1}e^{-\xi h}+b_0+b_1e^{\xi h}\right)e^{\xi x}\approx\left(c_{-1}e^{-\xi h}+c_0+c_1e^{\xi h}\right)e^{\xi x}.$$

After canceling $e^{\xi x}$ and substituting $e^{\xi h}=s$ (i.e., $\xi=\frac{1}{h}\log s$), this can be written as

$$\left\{\frac{1}{h}\log s\right\}^2\approx\frac{c_{-1}+c_0s+c_1s^2}{b_{-1}+b_0s+b_1s^2}. \tag{1.9}$$

This relation should be as accurate as possible for $\xi\to 0$, that is, for $s\to 1$. The Padé expansion (cf. Section E.3) of $\left\{\frac{1}{h}\log s\right\}^2$ around $s=1$ with numerator and denominator degrees both equal to 2 becomes

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

Equating coefficients with (1.9) gives (1.7). □

The MATLAB code in Algorithm 1.2 implements this Padé-based algorithm using the Symbolic Toolbox. For example, to compute the weights in the second row in Table 4.1, the statement

$$\texttt{[w\_f, w\_der] = weights\_Pade(sym(3/2), 0, 3, 1)}$$

produces the output `w_f` $=[1,-27,27,-1]$, `w_der` $=24*$`h`.

Table 1.1 *Weights for centered FD approximations of the first derivative on an equispaced grid (omitting the factor 1/h).*

Order													
				Weights		First derivative							
2						$-\frac{1}{2}$	0	$\frac{1}{2}$					
4					$\frac{1}{12}$	$-\frac{2}{3}$	0	$\frac{2}{3}$	$-\frac{1}{12}$				
6				$-\frac{1}{60}$	$\frac{3}{20}$	$-\frac{3}{4}$	0	$\frac{3}{4}$	$-\frac{3}{20}$	$\frac{1}{60}$			
8			$\frac{1}{280}$	$-\frac{4}{105}$	$\frac{1}{5}$	$-\frac{4}{5}$	0	$\frac{4}{5}$	$-\frac{1}{5}$	$\frac{4}{105}$	$-\frac{1}{280}$		
10		$-\frac{1}{1260}$	$\frac{5}{504}$	$-\frac{5}{84}$	$\frac{5}{21}$	$-\frac{5}{6}$	0	$\frac{5}{6}$	$-\frac{5}{21}$	$\frac{5}{84}$	$-\frac{5}{504}$	$\frac{1}{1260}$	
⋮		↓	↓	↓	↓	↓	⋮	↓	↓	↓	↓	↓	
Limit	⋯	$-\frac{1}{5}$	$\frac{1}{4}$	$-\frac{1}{3}$	$\frac{1}{2}$	−1	0	1	$-\frac{1}{2}$	$\frac{1}{3}$	$-\frac{1}{4}$	$\frac{1}{5}$	⋯

1.2.2 Some Tables of FD Formulas

Especially with the recursion and Padé algorithms (in Sections 1.2.1.4 and 1.2.1.5, respectively), it is very easy to generate tables of FD weights. Four examples are given in Tables 1.1–1.4.[13]

We can make a number of observations from these relating to increasing orders of accuracy:

i. For centered FD approximations, the weights converge to well-defined limits – derived later in Section 2.1 for an arbitrary-order derivative.

ii. As seen for the first and second derivatives in Tables 1.1 and 1.2, and for a general-order derivative in (2.1), the weights for centered FD approximations decay in magnitude with distance k from the stencil center at the rate of $O(1/k)$ for odd-order derivatives, and $O(1/k^2)$ for those of even order. These slow decay rates are in some contexts problematic, since analytically, a derivative is a local property of a function and should not depend heavily on distant data.[14]

iii. For one-sided approximations, the weights diverge rapidly with increasing orders of accuracy (as seen in Tables 1.3 and 1.4). Ways to understand and control this will be discussed in several contexts later in this book (including Sections 1.4, 5.3.7, and F.3).

[13] A simple relation between the entries in Tables 1.1 and 1.2 is given in Section 6.3.3.
[14] This comment does not apply to FD in the complex plane (cf. Chapter 6).

Table 1.2 *Weights for centered FD approximations of the second derivative on an equispaced grid (omitting the factor $1/h^2$).*

Order			Weights			Second derivative						
2					1	−2	1					
4				$-\frac{1}{12}$	$\frac{4}{3}$	$-\frac{5}{2}$	$\frac{4}{3}$	$-\frac{1}{12}$				
6			$\frac{1}{90}$	$-\frac{3}{20}$	$\frac{3}{2}$	$-\frac{49}{18}$	$\frac{3}{2}$	$-\frac{3}{20}$	$\frac{1}{90}$			
8		$-\frac{1}{560}$	$\frac{8}{315}$	$-\frac{1}{5}$	$\frac{8}{5}$	$-\frac{205}{72}$	$\frac{8}{5}$	$-\frac{1}{5}$	$\frac{8}{315}$	$-\frac{1}{560}$		
10	$\frac{1}{3150}$	$-\frac{5}{1008}$	$\frac{5}{126}$	$-\frac{5}{21}$	$\frac{5}{3}$	$-\frac{5269}{1800}$	$\frac{5}{3}$	$-\frac{5}{21}$	$\frac{5}{126}$	$-\frac{5}{1008}$	$\frac{1}{3150}$	
⋮	↓	↓	↓	↓	↓	⋮	↓	↓	↓	↓	↓	
Limit ⋯	$\frac{2}{5^2}$	$-\frac{2}{4^2}$	$\frac{2}{3^2}$	$-\frac{2}{2^2}$	$\frac{2}{1^2}$	$-\frac{\pi^2}{3}$	$\frac{2}{1^2}$	$-\frac{2}{2^2}$	$\frac{2}{3^2}$	$-\frac{2}{4^2}$	$\frac{2}{5^2}$	⋯

Table 1.3 *Weights for one-sided FD approximations of the first derivative on an equispaced grid (omitting the factor $1/h$).*

Order			Weights		First derivative						
1	−1	1									
2	$-\frac{3}{2}$	2	$-\frac{1}{2}$								
3	$-\frac{11}{6}$	3	$-\frac{3}{2}$	$\frac{1}{3}$							
4	$-\frac{25}{12}$	4	−3	$\frac{4}{3}$	$-\frac{1}{4}$						
5	$-\frac{137}{60}$	5	−5	$\frac{10}{3}$	$-\frac{5}{4}$	$\frac{1}{5}$					
6	$-\frac{49}{20}$	6	$-\frac{15}{2}$	$\frac{20}{3}$	$-\frac{15}{4}$	$\frac{6}{5}$	$-\frac{1}{6}$				
7	$-\frac{363}{140}$	7	$-\frac{21}{2}$	$\frac{35}{3}$	$-\frac{35}{4}$	$\frac{21}{5}$	$-\frac{7}{6}$	$\frac{1}{7}$			
8	$-\frac{761}{280}$	8	−14	$\frac{56}{3}$	$-\frac{35}{2}$	$\frac{56}{5}$	$-\frac{14}{3}$	$\frac{8}{7}$	$-\frac{1}{8}$		
9	$-\frac{7129}{2520}$	9	−18	28	$-\frac{63}{2}$	$\frac{126}{5}$	−14	$\frac{36}{7}$	$-\frac{9}{8}$	$\frac{1}{9}$	
10	$-\frac{7381}{2520}$	10	$-\frac{45}{2}$	40	$-\frac{105}{2}$	$\frac{252}{5}$	−35	$\frac{120}{7}$	$-\frac{45}{8}$	$\frac{10}{9}$	$-\frac{1}{10}$

1.3 Errors When Applying FD Formulas

Two types of errors arise when applying FD approximations to a function:

1. **Truncation errors:** The FD formula has an error of size $O(h^p)$, where p is the approximation's order of accuracy.
2. **Rounding errors:** Typical double precision accuracy in a computer has a relative error level of around 10^{-16}.