Approximating derivatives:

We are concerned here with approximating the value of a derivative, say \( f'(x^*) \), using the values of the function near \( x^* \). Two examples of such derivative approximation formulas are

\[
f'(x^*) \approx \frac{1}{h} [f(x^* + h) - f(x^*)] \tag{the so-called two-point forward difference formula}
\]

and

\[
f'(x^*) \approx \frac{1}{2h} [f(x^* + h) - f(x^* - h)]
\]

As always, \( h \) represents a small parameter.

Given such a formula, we can analyze the error by taking each term in the formula and calculating its Taylor expansion about \( x = x^* \) (by expansion we mean a Taylor polynomial of a certain degree, with error term, as we’ll see later). We can then obtain various forms and estimates for the error:

- the order of the error, expressed in the form \( O(h^k) \) for some \( k \)
- the leading term in the error, in this case usually expressed in a form such as \( C f^{(k+1)}(x^*) h^k \) for some \( C, k \)
- a bound for the error (i.e. showing that the error cannot be more than such-and-such)
- a mean value form of the error, in this case usually expressed in a form such as \( C f^{(k+1)}(\xi) h^k \), where \( \xi \) is somewhere between \( x^* \) and the values of \( x \) used in the formula.

Here’s how it works in these two examples:

\[
\frac{1}{h} [f(x^* + h) - f(x^*)] = \frac{1}{h} \left\{ \left[ f(x^*) + f'(x^*) h + \frac{1}{2} f''(x^*) h^2 + \ldots \right] - f(x^*) \right\} = f'(x^*) + \frac{1}{2} f''(x^*) h
\]

So that we can write the error as \( e(h) = -\frac{1}{2} f''(x^*) h + \ldots \)

From this calculation, we see that this approximation formula represents an \( O(h) \) method, (where \( O(h) \) refers to the order of the error) with leading error term \( -\frac{1}{2} f''(x^*) h \). For a more precise calculation, we redo our calculation above, replacing \( \frac{1}{2} f''(x^*) h^2 + \ldots \) above by its mean-value form \( \frac{1}{2} f''(\xi) h^2 \), and we obtain finally

\[
e(h) = -\frac{1}{2} f''(\xi) h \text{ which is the mean-value form of the error of this method.}
\]

For the other method

\[
\frac{1}{2h} [f(x^* + h) - f(x^* - h)] = \frac{1}{2h} \left\{ \left[ f(x^*) + f'(x^*) h + \frac{1}{2} f''(x^*) h^2 + \frac{1}{3!} f'''(x^*) h^3 + \ldots \right] \\
- \left[ f(x^*) - f'(x^*) h + \frac{1}{2} f''(x^*) h^2 - \frac{1}{3!} f'''(x^*) h^3 + \ldots \right] \right\} \\
= \frac{1}{2h} \left\{ 2f'(x^*) h + 2 \frac{1}{3!} f'''(x^*) h^3 + O(h^5) \right\} = f'(x^*) + \frac{1}{3!} f'''(x^*) h^2 + O(h^4)
\]

Above, we have noted that the even powers cancel and so the remainder term in
parentheses can be written \( O(h^5) \), and after dividing by \( 2h \), it becomes \( O(h^4) \). Thus this approximation formula, called the two-point central difference formula, is an \( O(h^2) \) method with leading error term \(-\frac{1}{3!}f'''(x^*)h^2\). We can also obtain a mean-value form by replacing the terms \( \frac{1}{3!}f'''(x^*)h^3 \) by their mean-value error form. More specifically, we have

\[
f(x^* + h) = f(x^*) + f'(x^*)h + \frac{1}{2}f''(x^*)h^2 + \frac{1}{3!}f'''(\xi_1)h^3
\]

\[
f(x^* - h) = f(x^*) - f'(x^*)h + \frac{1}{2}f''(x^*)h^2 + \frac{1}{3!}f'''(\xi_2)h^3
\]

from which we obtain

\[
\frac{1}{2h}[f(x^* + h) - f(x^* - h)] = f'(x^*) + \frac{1}{3!}\left[\frac{f'''(\xi_1) + f'''(\xi_2)}{2}\right]h^2
\]

Now \( \frac{f'''(\xi_1) + f'''(\xi_2)}{2} \) lies between the values of \( f''(\xi_1) \) and \( f''(\xi_2) \) and so there is a number \( \eta \) between \( \xi_1 \) and \( \xi_2 \) such that \( \frac{f'''(\xi_1) + f'''(\xi_2)}{2} = f'''(\eta) \), leading us finally to

\[
\frac{1}{2h}[f(x^* + h) - f(x^* - h)] = f'(x^*) + \frac{1}{3!}f'''(\eta)h^2
\]

and an error formula of

\[
e(h) = -\frac{1}{3!}f'''(\eta)h^2,
\]

where \( \eta \) is between \( x^* + h \) and \( x^* - h \).

How to generate formulas:

The formulas for approximating derivatives can all be obtained, at least conceptually, by interpolating the values \( \{f(x_j)\} \), \( j = 0, \ldots, k \), where the \( x_j \) are near \( x^* \), with a polynomial \( p_k(x) \) of degree \( k \), and then approximating \( f'(x^*) \equiv p_k(x^*) \), where the term on the right will become a formula expressed in terms of the data \( \{x_j\} \) and \( \{f(x_j)\} \) for \( j = 0, \ldots, k \). As an example, consider approximating \( f'(x^*) \) by interpolating at

\[
x_0 = x^* - h, \quad x_1 = x^*, \quad x_2 = x^* + h.
\]

We have (being a little clever with the ordering)

\[
p_2(x) = f[x_0] + f[x_0,x_2](x-x_0) + f[x_0,x_2,x_1](x-x_0)(x-x_2)
\]

Noting that the derivative of the last term is zero midway between \( x_0 \) and \( x_2 \) (i.e. where \( x^* \) is located), we have \( p_2'(x^*) = f[x_0,x_2] = \frac{1}{2h}[f(x^* + h) - f(x^* - h)] \). This is just the two-point central difference formula - note that \( x_1 \) "dropped out" due to symmetry. One can go on to analyze the error as is in the textbook. A way to automatically calculate derivative approximation formulas from polynomial interpolation is presented later. However, for now we will take a simpler approach to developing these formulas and determining the leading term of the error.

Developing formulas from exactness:

We are concerned here with formulas of the form

\[
f(x^*) \equiv \frac{1}{h} \left[ c_0f(x_0) + \ldots + c_kf(x_k) \right] = L_h[f]
\]

where \( x_j = x^* + a_jh \) and the \( c_j \) and the \( a_j \) are fixed and independent of \( h \). The symbol \( L_h[f] \) will represent the application of the formula to \( f \) for a given \( h \). We say the formula has **approximation degree** \( m \) if it gives the exact answer whenever \( f(x) \) is a polynomial of degree \( m \) or less, but is not exact for polynomials of degree \( m + 1 \). For instance the formula
Theorem: If the formula $f'(x^*) \approx L_h[f]$ has approximation degree $m$ then

$$L_h[f] = f'(x^*) + C f^{(m+1)}(x^*) h^m + O(h^{m+1})$$

where $C$ is a constant independent of $f$ and $h$.

The value of $C$ can be determined using the special case $h = 1$, $x^* = 0$, $f(x) = x^{m+1}$ on both sides, with the $O(h^{m+1})$ term omitted.

Proof I: If we expand $\frac{1}{h} [c_0 f(x_0) + \ldots + c_k f(x_k)]$ about $x = x^*$ we get an expansion that looks like

$$L_h[f] = \frac{1}{h} [c_0 f(x_0) + \ldots + c_k f(x_k)]
= \frac{1}{h} [C_0 + C_1 f'(x^*) h + C_2 f''(x^*) h^2 + \ldots + C_m f^{(m)}(x^*) h^m + C_{m+1} f^{(m+1)}(x^*) h^{m+1} + O(h^{m+2})$$

The term $O(h^{m+2})$ will involve a number of terms that can be precisely expressed in terms of $f^{(m+1)}(\xi)$ for different values of $\xi$.

Now, successively putting $f(x) = x^j$, $j = 0, 1, \ldots, m$ on both sides, setting $x^* = 0$, $h = 1$, and applying the assumed exactness of the formula, we must have, in succession:

$j = 0 : L_h[x^0] = 0 = C_0$

$j = 1 : L_h[x^1] = 1 = C_1$

$j = 2 : L_h[x^2] = 0 = 0 + 2C_2$ so that $C_2 = 0$

and continuing, we find $C_3, \ldots, C_m$ are all zero. Finally, putting $h = 1$, $x^* = 0$, $f(x) = x^{m+1}$ we obtain

$j = m + 1 : L_1 x^{m+1} = c_0 f(a_0) + \ldots + c_k f(a_k) = (m + 1)! C_{m+1}$, where the $O(h^{m+2})$ term vanishes because it involves derivatives of $f(x) = x^{m+1}$ higher than $f^{(m+1)}$, which are identically zero. The value of $C_{m+1}$ can be determined from this equation, since the left hand side is just a number. We obtain finally, $L_h[f] = f'(x^*) + C_{m+1} f^{(m+1)}(x^*) h^m + O(h^{m+1})$

Proof II: Let’s expand $f(x)$ in a Taylor polynomial $T_m(x)$ of degree $m$, about $x = x^*$.

We obtain

$$f(x) = T_m(x) + \frac{1}{(m + 1)!} f^{(m+1)}(x^*)(x-x^*)^{m+1} + \frac{1}{(m + 2)!} f^{(m+2)}(\xi_x)(x-x^*)^{m+2}$$

Now imagine applying the formula to both sides, noting that the formula, applied to a sum of functions, can be obtained by adding the individual results as applied to each term. Now, when the formula is applied to $T_m(x)$ it must give the value

$$L_h[T_m] = T_m(x^*) = f'(x^*)$$

because of exactness; when the formula is applied to $\frac{1}{(m + 1)!} f^{(m+1)}(x^*)(x-x^*)^{m+1}$ the result will be of the form $C_{m+1} f^{(m+1)}(x^*) h^m$ and when applied to the last term, the result is a term of order $O(h^{m+1})$ that involves derivatives
\( f^{(m+2)}(\xi) \) for various \( \xi \). Thus, we have obtained
\[
L_h[f] = f'(x^*) + C_{m+1} f^{(m+1)}(x^*) h^m + O(h^{m+1})
\] and the value of \( C_{m+1} \) can be determined as before.

To summarize: If we have a formula of approximation degree \( m \), we immediately know that the leading term in the error is \( e(h) = C f^{(m+1)}(x^*) h^m + O(h^{m+1}) \). The value of \( C \) can be calculated from the error in the case of \( x^* = 0, h = 1, f(x) = x^{m+1} \) in which case \( e(1) = C f^{(m+1)}(0) = C \cdot (m+1)! \)

(Note: The converse is obvious - if the error satisfies the above, then the formula is exact for all polynomials up to degree \( m \) and if \( C \neq 0 \) then it is exact for no degree higher than \( m \))

Now, how can we derive such formulas of a given approximation degree? Consider the following: Let \( a_0, \ldots, a_k \) be any fixed numbers. Suppose we interpolate a general function \( f(x) \) at \( x = a_0, \ldots, a_k \) with a polynomial of degree \( k \), denoted \( p_k(x) \). Then if \( f(x) \) is any polynomial of degree \( k \) or less we must have \( f'(0) = p'_k(x) \) since
\[
p_k(x) = f(x) \text{ under those circumstances. This gives us a formula}
\]
\[
f'(0) \cong c_0 f(a_0) + \ldots + c_k f(a_k) \text{ that is exact for any polynomial } f \text{ of degree } k \text{ or less. In fact, you can obtain this form directly by differentiating the Lagrange form}
\]
\[
p_k(x) = f(a_0)c_0(x) + f(a_1)c_1(x) + \ldots + f(a_k)c_k(x),
\]
\( \text{obtaining } c_j = \ell_j(0). \) We then let \( m \) be the highest degree such that this formula is exact for any polynomial up to degree \( m \). The value of \( m \) can be determined by directly trying \( f(x) = x^{k+1}, x^{k+2} \) until the formula fails to be exact. Next, we claim that for a formula \( f'(0) \cong c_0 f(a_0) + \ldots + c_k f(a_k) \) exact for any polynomial up to degree \( m \), the so-called scaled formula
\[
f'(0) \cong \frac{1}{h} [c_0 f(a_0 h) + \ldots + c_k f(a_k h)]
\]
is exact for any polynomial \( f \) of degree \( m \) or less. Examining the expression
\[
c_0 f(a_0 h) + \ldots + c_k f(a_k h),
\]
we can consider this as applying our original formula to the function \( F(t) = f(ht) \), namely \( F'(0) \cong c_0 F(a_0) + \ldots + c_k F(a_k) = c_0 f(a_0 h) + \ldots + c_k f(a_k h) \). Now if \( f(x) = p(x) \) is a polynomial then \( F(t) \) is a polynomial in \( t \) and so by exactness
\[
h f'(0) = F'(0) = c_0 F(a_0) + \ldots + c_k F(a_k) = c_0 f(a_0 h) + \ldots + c_k f(a_k h)
\]
This shows that the formula \( f'(0) \cong \frac{1}{h} [c_0 f(a_0 h) + \ldots + c_k f(a_k h)] \) is exact as described. Finally, it is a simple matter to translate the formula to any \( x = x^* \) instead of \( x = 0 \), so that we have
\[
f'(x^*) \cong \frac{1}{h} [c_0 f(x^* + a_0 h) + \ldots + c_k f(x^* + a_k h)]
\]
is exact up to degree \( m \), and no higher.

Example: Let’s analyze the central difference formula \( f'(x^*) \cong \frac{1}{2h} [f(x^* + h) - f(x^* - h)] \)
\( . \) The approximation degree can be obtained from the unscaled version with \( x^* = 0, h = 1 \), namely \( f'(0) \cong \frac{1}{2} [f(1) - f(-1)]. \) We see that this formula is exact for
\( f(x) = 1, x, x^2 \) but if \( f(x) = x^3 \) the error is \( e(1) = f'(0) - \frac{1}{2} [f(1) - f(-1)] = 0 - \frac{1}{2} [1 - (-1)] = -1 \) and the formula is not exact in that case.

So the approximation degree is 2. The error \( e(h) \) then satisfies \( e(h) = C f''(x^*) h^2 + O(h^3). \) Now we determine \( C \) with \( f(x) = x^3, x^* = 0, h = 1 : \)
\[ e(1) = C f'''(0) = (3!)C \text{ so } C = \frac{1}{3!} \text{ and } e(h) = -\frac{1}{3!} f'''(x^*) h^2 + O(h^3). \]

There is one more useful observation to make: If we want to make the formula \( f(0) \cong c_0 f(a_0) + \ldots + c_k f(a_k) \) exact for all polynomials of degree \( k \) or less, we can simply put \( f(x) = x^j, j = 0, \ldots, k \) on both sides and obtain a square system of linear equations for the \( c_j \). We know this system has a solution (namely the one described above that could be obtained by polynomial interpolation). Moreover the solution must be unique - setting \( f(x) \equiv (x-a_0) \cdots (x-a_{j-1})(x-a_{j+1}) \cdots (x-a_k) \), and applying exactness, we must have \( f'(0) = c_j f(a_j) \), which fixes the value of each \( c_j \). The system of linear equations has a very simple structure and is easy to solve in MATLAB. There are no issues of efficiency or roundoff error to deal with as long as the number of points is reasonably small, as is the case in practice.

Example: Develop an approximation formula for \( f'(x^*) \) of maximum approximation degree that uses the points \( x^* - 2h, x^* - h, x^*, x^* + h, x^* + 2h \). Develop the leading term in the error \( e(h) \).

Solution: We begin with the formula

\[ f'(0) \cong c_0 f(-2) + c_1 f(-1) + c_2 f(0) + c_3 f(1) + c_4 f(2) \]

Successively setting \( f(x) = x^j, j = 0, 1, \ldots, 4 \) and applying exactness, we obtain the system (written in suggestive form)

\[
\begin{align*}
x^0 &: \quad 0 = 1c_0 + 1c_1 + 1c_2 + 1c_3 + 1c_4 \\
x^1 &: \quad 1 = (-2)^1c_0 + (-1)^1c_1 + (0)^1c_2 + (1)^1c_3 + (2)^1c_4 \\
x^2 &: \quad 0 = (-2)^2c_0 + (-1)^2c_1 + (0)^2c_2 + (1)^2c_3 + (2^2)c_4 \\
&\vdots \\
x^4 &: \quad 0 = (-2)^4c_0 + (-1)^4c_1 + (0)^4c_2 + (1)^4c_3 + (2^4)c_4
\end{align*}
\]

The structure of the system is clear. If \( a = [-2 -1 0 1 2] \) denotes a row array as in MATLAB, we have:

\[
\begin{bmatrix}
0 \\
1 \\
0 \\
\vdots \\
0
\end{bmatrix} =
\begin{bmatrix}
a^0 \\
\vdots \\
a^j \\
\vdots \\
a^k
\end{bmatrix}
\begin{bmatrix}
c
\end{bmatrix}
\]

We solve in MATLAB:

```matlab
>> a=-2:2; A=ones(size(a)); for j=1:4, A=[A;a.^j]; end
>> b=zeros(size(a)); b=b'; b(2)=1;
>> [b A];
ans =
0  1  1  1  1
1 -2 -1  0  1  2
0  4  1  0  1  4
0 -8 -1  0  1  8
0 16  1  0  1  16
```
\[ \begin{align*}
\mathbf{c} &= \mathbf{A}^{-1}\mathbf{b} \\
&= \begin{bmatrix}
0.0833 \\
-0.6667 \\
0 \\
0.6667 \\
-0.0833
\end{bmatrix}
\end{align*} \]

This gives us the values of the coefficients. Now for the approximation degree. It is at least 4. The error when \( f(x) = x^5 \) can be easily calculated now in MATLAB:

\[ \begin{align*}
\texttt{er} &= 0 - (\mathbf{a}^T \mathbf{c}) \\
&= \begin{bmatrix}
4
\end{bmatrix}
\end{align*} \]

So the approximation degree is in fact \( m = 4 \). The error \( e(h) \) now satisfies

\[ e(h) = C f^{(5)}(x^*) h^4 + O(h^5) \]

We solve for \( C \) by plugging \( x^* = 0, h = 1, f(x) = x^5 \) on both sides and solving for \( C \). Since \( e(1) = 4 \) has already been calculated, we see that \( C = \frac{4}{5!} = \frac{1}{30} \) and so

\[ e(h) = \frac{1}{30} f^{(5)}(x^*) h^4 + O(h^5). \]

We now experimentally check the error with MATLAB.

Let's take the simple example of \( f(x) = \frac{1}{1+x} \), at \( x^* = 0.3 \) and \( h = 0.1 \). We have

\[ f^{(5)}(x) = \frac{-5!}{(1+x)^5} \]

\[ \begin{align*}
\texttt{f} &= \text{inline}('1/(1+x)'), \quad x = 0.3, h = 0.1; \\
\texttt{y} &= \texttt{f(x+h*a)}; \quad \texttt{df} = \texttt{y*c/h} \\
\texttt{df} &= -0.59163059163059
\end{align*} \]

Next we calculate the exact value and the error and compare with the leading term:

\[ \begin{align*}
\texttt{dfexact} &= -1/(1+x)^2 \\
\texttt{dfexact} &= -0.59171597633136 \\
\texttt{er} &= \texttt{dfexact} - \texttt{df} \\
\texttt{er} &= -8.538470076868698e-005 \\
\texttt{d5f} &= \texttt{factorial(5)./(1+x)^6} \quad \texttt{%the fifth derivative at x} \\
\texttt{d5f} &= -24.8614532396040 \\
\texttt{d5f*(h^4)/30} \quad \texttt{%calculation of leading term}
\end{align*} \]
ans =
-8.287048441320135e-005

The agreement is quite good. The smaller the value of $h$ the better the agreement will be. We will now perform the same operation on a sequence of successively halved values of $h$. We’ll do this using a loop. After that we calculate the error and the leading term (error estimate).

```
>> h=.1; H=[]; D=[]; for n=1:10, y=f(x+a*h); df=y*c/h;
D=[D;df]; H=[H;h]; h=h/2; end;
>> [H D]
ans =
0.10000000000000 -0.59163059163059
0.05000000000000 -0.59171075837743
0.02500000000000 -0.59171565201902
0.01250000000000 -0.59171595608995
0.00625000000000 -0.59171597506671
0.00312500000000 -0.59171597625231
0.00156250000000 -0.59171597632636
0.00078125000000 -0.59171597633096
0.00039062500000 -0.59171597633110
0.00019531250000 -0.59171597633124
```

```
>> er=dfexact-D;
>> erest=d5f*(H.^4)/30;
>> [H D er erest]
ans =
0.10000000000000 -0.59163059163059 -0.00000008538470077 -0.00008287048441
0.05000000000000 -0.59171075837743 -0.0000000521795394 -0.000000517940528
0.02500000000000 -0.59171565201902 -0.000000032431234 -0.000000032371283
0.01250000000000 -0.59171595608995 -0.00000002024141 -0.00000002023205
0.00625000000000 -0.59171597506671 -0.0000000126465 -0.0000000126450
0.00312500000000 -0.59171597625231 -0.000000007905 -0.000000007903
0.00156250000000 -0.59171597632636 -0.000000005050 -0.000000005049
0.00078125000000 -0.59171597633096 -0.0000000040 -0.0000000040
0.00039062500000 -0.59171597633110 -0.0000000026 -0.0000000026
0.00019531250000 -0.59171597633124 -0.0000000012 -0.0000000012
```

An important point is to observe the behavior of the error in column 3. Since $h$ is halved each time, the error should decrease by a factor of $(1/2)^4$ since this is an $O(h^4)$ method. We check this:

```
>> er(2:end)./er(1:end-1)
ans =
0.061111111110150
0.06215316282503
0.06241331655073
0.06247845458847
0.06251020545042
0.06327066000798
```
The discrepancies observed in the last few entries, here and in the leading term estimate of the error above, is the result of roundoff error in evaluating the formula, for example when $h = .000195$ we will be losing about three decimal places of accuracy in applying the formula and the error due to roundoff is considerably larger than the approximation error of the formula.

The MATLAB calculation can be done even more efficiently using matrix algebra - we create a matrix in which each row represents the function values $f(x + h * a)$ for a given $h$, and each column represents the values at the different $h$ values. The matrix $F$ is created in which $(F)_{ij} = f(x + a_j * h_i)$ and then $F * c$ gives, as a column, the array of formula calculations for the different $h$'s. The basic matrix operation is:

$$H = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_n \end{bmatrix}, \quad a = \begin{bmatrix} a_0 & \cdots & a_k \end{bmatrix}, \quad H * a = \begin{bmatrix} h_1 a_0 & \cdots & h_1 a_k \\ h_2 a_0 & \cdots & h_2 a_k \\ \vdots & \cdots & \vdots \\ h_n a_0 & \cdots & h_n a_k \end{bmatrix}$$

and then:

$$F = f(x + H * a)$$

>> H=1./2.^(0:10)’; F=f(x+H*a); D=F*c./H; [H D]

ans =

0.10000000000000 -0.59163059163059
0.05000000000000 -0.59171075837743
0.02500000000000 -0.59171565201902
0.01250000000000 -0.59171595608995
0.00625000000000 -0.59171597506671
0.00312500000000 -0.59171597625231
0.00156250000000 -0.59171597632636
0.00078125000000 -0.59171597633096
0.00039062500000 -0.59171597633110
0.00019531250000 -0.59171597633124
0.00009765625000 -0.59171597633025

One last comment - examining the coefficient in $c$, we can write the formula as:

$$f'(0) \approx \frac{1}{12} f(-2h) - \frac{2}{3} f(-h) + \frac{2}{3} f(h) - \frac{1}{12} f(2h) = \frac{4}{3} \left[ \frac{f(h) - f(-h)}{2} \right] - \frac{1}{3} \left[ \frac{f(2h) - f(-2h)}{2(2h)} \right]$$

which can be considered as a combination of two applications of the two-point central difference formula, one at $2h$ and the other at $h$. This combination will come up again in Richardson extrapolation.

Higher derivatives: For a derivative $f^{(i)}(x^*)$, we can similarly obtain formulas of the form
\[ f^{(i)}(x^*) \approx \frac{1}{h^i} [c_0 f(x^* + a_0 h) + \ldots + c_k f(x^* + a_k h)] \]

and the same techniques of Taylor expansions, polynomial interpolation, and exactness apply. The only modification we need to make is in the error term: If the formula above has approximation degree \( m \), then the error has the form 
\[
e(h) = C f^{(m+1)}(x^*) h^{m+1-j} + O(h^{m+2-j})
\]
where \( C \) can be determined from the case of \( f(x) = x^{m+1}, h = 1, x^* = 0 \) without the \( O(h^{m+2-j}) \) term.

Example: Consider a formula 
\[
f''(x^*) \approx \frac{1}{h^2} [c_0 f(x^* - 2h) + c_1 f(x^* - h) + c_2 f(x^*) + c_3 f(x^* + h) + c_4 f(x^* + 2h)]
\]

We start with the unscaled version 
\[
f''(0) \equiv c_0 f(-2) + c_1 f(-1) + c_2 f(0) + c_3 f(1) + c_4 f(2)
\]

We formulate exactness using MATLAB and solve for the coefficients - note that here we are taking \( f(x) = x^j \) and calculating \( f''(0) \).

\[
>> a=-2:2; A=ones(size(a)); for j=1:4, A=[A;a.^j]; end
>> b=zeros(size(a)); b=b'; b(3)=2;
\]
\[
>> [b A]
\]
\[
ans =
0 1 1 1 1
0 -2 -1 0 1 2
2 4 1 0 1 4
0 -8 -1 0 1 8
0 16 1 0 1 16
\]
\[
>> format rat; c=A\b
\]
\[
c =
-1/12
4/3
-5/2
4/3
-1/12
\]
\[
>> format long
\]
\[
>> 0-(a.^5)*c
\]
\[
ans =
4.440892098500626e-016
\]
\[
>> 0-(a.^6)*c
\]
\[
ans =
8.000000000000000
\]

Note that this formula has approximation degree \( m = 5 \) as calculated above - exactness must hold at least up to \( m = 4 \) and exactness for \( m = 5 \) clearly holds from the symmetry of the coefficients. The calculation above shows that the formula is not exact for degree \( 6 \) - the error in the case of \( x^6 \) is calculated above as \( 8 \). Finally, the leading error term is 
\[
e(h) \approx C f^{(6)}(x^*) h^4
\]
where 
\[
C = 8/6! = \frac{1}{90}.
\]

Coefficients by polynomial interpolation:

As noted previously, a direct way to exactness in a formula of the form 
\[
f'(0) \equiv c_0 f(a_0) + \ldots + c_k f(a_k)
\]
is to interpolate \( f(x) \) at \( x = a_0, \ldots, a_k \) with a polynomial \( p_k(x) \) and then to write \( f'(0) \equiv p_k'(0) \), where the expression on the right will have the desired form. This method is obviously exact if \( f \) is a polynomial of degree \( k \) or less since in that case \( p_k(x) = f(x) \) at all \( x \). This may seem like a messy proposition to calculate, but
it’s not that bad. First, if we write \( p_k(x) \) in Lagrange form, 
\[ p_k(x) = f(a_0)\ell_0(x) + \ldots + f(a_k)\ell_k(x) \]
then we have 
\[ p_k'(0) = f(a_0)\ell'_0(0) + \ldots + f(a_k)\ell'_k(0) \].
The calculation of \( \ell'_j(0) \) can be effected in MATLAB by converting \( \ell_j(x) \) to pure power form and then easily calculating the desired derivative at \( x = 0 \) from the coefficients - in this case we just take the coefficient of \( x \) in \( \ell_j(x) \) as the value of \( \ell'_j(0) \). The conversion of a general polynomial in Newton form to its pure power form is done in the MATLAB program Newtconvert.m. The MATLAB program derivapprox.m then uses this idea to calculate the correct coefficients for the formula 
\[ f^{(j)}(0) \approx c_0 f(a_0) + \ldots + c_k f(a_k) \] to be exact up to degree \( k \), where \( j \) can be any order of derivative.

Richardson extrapolation:
We apply Richardson extrapolation to the example above, namely \( f'(x^*) \) with 
\[ x^* = 0.3, f(x) = \frac{1}{1+x} \]
\[ \gg f = \text{inline}('1./(1+x)'); x = 0.3; h = 0.1; \]
\[ \gg a = [-1 1]; c = [-0.5 0.5]; \]
\[ \gg H = 0.1 ./ 2. ^ (0:10)'; F = f(x + H * a); D = F * c ./ H; \]
\[ \text{ans} = \]
\[
\begin{array}{cccc}
0.10000000000000 & -0.59523809523810 \\
0.05000000000000 & -0.59259259259259 \\
0.02500000000000 & -0.59171939556789 \\
0.01250000000000 & -0.59171598967008 \\
0.00625000000000 & -0.59171597633136 \\
0.00312500000000 & -0.59171597633136 \\
0.00156250000000 & -0.59171597633136 \\
0.00078125000000 & -0.59171597633136 \\
0.00039062500000 & -0.59171597633136 \\
0.00019531250000 & -0.59171597633136 \\
0.00009765625000 & -0.59171597633136 \\
\end{array}
\]
\[ \gg m = 11; w = 4; \text{for } j = 2 : m, D(j : m, j) = (w * D(j : m, j - 1) - D(j - 1 : m - 1, j - 1)) / (w - 1); w = 4 * w; \text{end} \]
\[ \gg D(1:5, 1:5) \]
\[ \text{ans} = \]
Columns 1 through 4
\[-0.59523809523810 \quad 0 \quad 0 \quad 0 \]
\[-0.59259259259259 \quad 0 \quad 0 \quad 0 \]
\[-0.59171939556789 \quad 0 \quad 0 \quad 0 \]
\[-0.59171598967008 \quad 0 \quad 0 \quad 0 \]
\[-0.59171597633136 \quad 0 \quad 0 \quad 0 \]
\[ \gg \text{dfexact} \]
dfexact =
\[-0.59171597633136 \]
We observe that we obtain virtually the exact answer, 14 significant digits, using only
the first five entries of the first column of $D$, in which the most accurate approximation had only about 4 significant digits.