Overview  Deferred correction is a general technique for improving the accuracy of numerical methods for differential equations without excessive algebra. Suppose we have a low-order method such as Euler’s method
\[ u_{n+1} = u_n + hf(t_n, u_n), \quad 0 \leq n \leq N - 1, \quad u_0 = y_0, \]
for solving the initial value problem
\[ y'(t) = f(t, y(t)), \quad 0 \leq t \leq T, \quad y(0) = y_0. \]
Construct a continuous function \( u_1(t) \) which interpolates the discrete values \( u_n \) at each \( t_n \). Then there is a continuous error function \( e(t) = y(t) - u_1(t) \) with \( e(t_n) = y(t_n) - u_n = O(h) \) and \( e(0) = 0 \). Differentiate \( e(t) \) and use the differential equation for \( y \) to get
\[ e'(t) = y'(t) - u'_1(t) = f(t, y(t)) - u'_1(t). \]
Since \( y(t) = u_1(t) + e(t) \), we have a differential equation for \( e(t) \):
\[ e'(t) = f(t, u_1(t) + e(t)) - u'_1(t) = g(t, e(t)) \]
where
\[ g(t, e) = f(t, u_1(t) + e) - u'_1(t) \]
is known once we construct \( u_1 \). Since \( e(t) \) satisfies an initial value problem for an ordinary differential equation, we can use Euler’s method to solve for \( e(t) \):
\[ e_{n+1} = e_n + hg(t_n, e_n) = e_n + h \left( f(t_n, u_n + e_n) - u'_1(t_n) \right), \]
with initial condition \( e_0 = 0 \). Euler’s method will approximate \( e(t_n) \) with relative error \( O(h) \), and since \( e(t) = O(h) \) we have
\[ e_n = e(t_n) + O(h^2). \]
Hence we can correct the first-order solution \( u_n \) to a second-order solution \( u_n^2 \) by adding \( e_n \):
\[ u_n^2 = u_n + e_n. \]
If \( u'_1(t_n) \) is sufficiently accurate, then the corrected solution \( u_{n+1}^2 \) will be second-order accurate. The process can be repeated as long as the accuracy holds out, gaining one order of accuracy per iteration. This is global deferred correction, because we construct the first-order solution and then correct it globally to higher order. In practice deferred correction is done locally on some interval \( t_n \leq t \leq t_{n+p} \), the corrected solution is produced on the interval, and only then does the computation proceed from \( u_{n+p} \) to the next interval \([t_{n+p}, t_{n+2p}]\).
**Interpolation**  How to construct an interpolant $u_1(t)$ in practice? Interpolation from $u_n$ and $u_{n+1}$ gives

$$u_1(t) = \frac{t - t_n}{t_{n+1} - t_n} u_{n+1} + \frac{t - t_{n+1}}{t_n - t_{n+1}} u_n$$

and

$$u'_1(t_n) = \frac{u_{n+1} - u_n}{t_{n+1} - t_n} = \frac{u_{n+1} - u_n}{h} = f(t_n, u_n).$$

Then the method becomes

$$u_{n+1}^2 = u_n^2 + h \left( f(t_n, u_n) + f(t_n, u_n^2) - \frac{u_{n+1} - u_n}{h} \right) = u_n^2 + hf(t_n, u_n^2)$$

so we are just running Euler’s method again.

Hence we have to use higher-order interpolation from more data. Now we change notation to avoid an excess of fractions. Our method will now start at $t_n$, compute forward to $t_{n+p}$ for some $p \geq 1$, and correct on the interval $[t_n, t_{n+p}]$ until maximum accuracy is achieved. Then the next interval is processed. Consider the case $p = 2$. Then two steps of Euler’s method from $t_n$ to $t_{n+2}$ gives $u_n$, $u_{n+1}$ and $u_{n+2}$. A quadratic interpolant is

$$u_1(t) = \frac{(t - t_{n+1})(t - t_{n+2})}{(t_n - t_{n+1})(t_n - t_{n+2})} u_{n+1} + \frac{(t - t_n)(t - t_{n+2})}{(t_{n+1} - t_n)(t_n - t_{n+2})} u_{n+1} + \frac{(t - t_n)(t - t_{n+1})}{(t_{n+2} - t_n)(t_{n+2} - t_{n+1})} u_{n+2}.$$

Then two steps of Euler approximate the errors

$$e_{n+1} = h \left( f(t_n, u_n) - u'_1(t_n) \right),$$

and

$$e_{n+2} = e_{n+1} + h \left( f(t_{n+1}, u_{n+1} + e_{n+1}) - u'_1(t_{n+1}) \right),$$

where we have set $e_n = 0$. (We cannot correct accumulated error at or before $t_n$ so we don’t do anything about it.) After computing $e_{n+1}$ and $e_{n+2}$, we can correct $u_{n+1}$ and $u_{n+2}$ to second-order accuracy by

$$u_{n+1}^2 = u_{n+1} + e_{n+1}$$

and

$$u_{n+2}^2 = u_{n+2} + e_{n+2}.$$  

The computations which produce the second-order solution $u^2$ from the first-order solution $u^1 = u$ constitute one pass of ”deferred correction.”

**IDEC**  Iterated deferred correction consists of several passes, each of which promotes an order-$k$ solution $u_{n+j}^k$ to an order-$k + 1$ solution on an interval $t_n \leq t \leq t_n + ph = t_{n+p}$. It can also be viewed as a Runge-Kutta method with stepsize $ph$ for $u_{n+p}^p$. (But it is worth noting that $u_{n+j}^p$ is also order-$p$ accurate for $1 \leq j \leq p$. This contrasts with extrapolation methods which produce a highly accurate solution only at the final point of each timestep.)  

For each pass $k = 1$ through $p - 1$, we construct the degree-$p$ polynomial $u_k(t)$ interpolating $u_n^k$ through $u_{n+p}^k$. Then we solve the error equation

$$e'(t) = f(t, u(t) + e(t)) - u_k(t), \quad t_n \leq t \leq t_{n+p},$$
with initial conditions $e(t_n) = 0$, by Euler’s method:

$$e_{n+j+1} = e_{n+j} + h \left( f(t_{n+j}, u_{n+j}^k + e_{n+j}) - u_k'(t_{n+j}) \right), \quad 0 \leq j \leq p - 1,$$

with initial condition $e_n = 0$. The corrected solution is

$$u_{n+j}^{k+1} = u_{n+j}^k + e_{n+j}$$

for $0 \leq j \leq p - 1$, and approximates the exact solution to accuracy $O(h^{k+1})$. The maximum possible accuracy achievable with degree-$p$ interpolants $u_k$ is $O(h^p)$, as in the second-order method derived above by quadratic interpolation.

Deferred correction builds $u_{n+p}$ from stages where $f$ is evaluated at various arguments depending on previous stages, and is therefore an explicit Runge-Kutta method with stepsize $ph$. For example, take $p = 2$. Two steps of Euler’s method generate two stages

$$k_1 = f(t_n, u_n)$$
$$k_2 = f(t_{n+1}, u_n + hk_1) = f(t_n + c_22h, u_n + 2ha_21k_1)$$

where $c_2 = a_{21} = 1/2$. In terms of these stages we compute first-order accurate solutions

$$u_{n+1}^1 = u_n + hk_1$$
$$u_{n+2}^1 = u_{n+1}^1 + hk_2 = u_n + h(k_1 + k_2).$$

Then two steps of Euler’s method for the error equation yield

$$e_{n+1} = h \left( f(t_n, u_n) - u_1'(t_n) \right) = hk_1 - hu_1'(t_n)$$

since $e_n = 0$ and

$$e_{n+2} = e_{n+1} + h \left( f(t_{n+1}, u_{n+1}^1 + e_{n+1}) - u_1'(t_{n+1}) \right).$$

Computation of $e_{n+1}$ uses the previously computed stage $k_1$, but $e_{n+2}$ generates a new stage

$$k_3 = f(t_{n+1}, u_{n+1}^1 + e_{n+1})$$

Explicitly, there are dimensionless coefficients $d_{ij}$

$$hu_1'(t_n) = d_{00}u_n^1 + d_{01}u_{n+1}^1 + d_{02}u_{n+2}^1$$

and

$$hu_1'(t_{n+1}) = d_{10}u_n^1 + d_{11}u_{n+1}^1 + d_{12}u_{n+2}^1$$

where $d_{ij} = hL_j'(t_{n+i})$ in terms of the Lagrange basis polynomials $L_j$ for interpolating at $t_n$ through $t_{n+2}$. These constants can be computed by Fornberg’s method or some algebra. Since derivatives of constants vanish, $d_{i0}u_n + d_{i1}u_n + d_{i2}u_n = 0$ for each $i$ and

$$hu_1'(t_{n+i}) = h(d_{i1} + d_{i2})k_1 + hd_{i2}k_2.$$
Hence
\[ e_{n+1} = h((1 - d_{01} - d_{02})k_1 - d_{02}k_2) \]
and
\[ k_3 = f(t_{n+1}, u_n + h((2 - d_{01} - d_{02})k_1 - d_{02}k_2)). \]
Thus
\[ e_{n+2} = e_{n+1} + h(k_3 - u'(t_{n+1}) = h ((1 - d_{01} - d_{02} - d_{11} - d_{12})k_1 - (d_{02} + d_{12})k_2 + k_3). \]
Adding it all up,
\[ u_{n+2}^2 = u_{n+1}^1 + e_{n+2} = u_n + h(k_1 + k_2) + e_{n+2} \]
or
\[ u_{n+2}^2 = u_n + h((2 - d_{01} - d_{02} - d_{11} - d_{12})k_1 + (1 - d_{02} - d_{12})k_2 + k_3). \]
Hence the Butcher array is
\[
\begin{array}{cccc}
0 & 0 & 0 & 0 \\
1/2 & 1/2 & 0 & 0 \\
1/2 & (2 - d_{01} - d_{02})/2 & -d_{02}/2 & 0 \\
(2 - d_{01} - d_{02} - d_{11} - d_{12})/2 & (1 - d_{02} - d_{12})/2 & 1/2 \\
\end{array}
\]
Since
\[ d = \frac{1}{2} \begin{bmatrix} -3 & 4 & -1 \\ -1 & 0 & 1 \\ 1 & -4 & 3 \end{bmatrix}, \]
we have the Butcher array
\[
\begin{array}{cccc}
0 & 0 & 0 & 0 \\
1/2 & 1/2 & 0 & 0 \\
1/2 & 1/4 & 1/4 & 0 \\
0 & 1/2 & 1/2 \\
\end{array}
\]
Since
\[ \sum_{i=1}^{s} b_i = 1 \]
the method is at least first-order accurate, and since
\[ \sum_{i=1}^{s} b_i \sum_{j=1}^{s} a_{ij} = \frac{1}{2}, \]
the method is at least second-order accurate. Since
\[ \sum_{i=1}^{s} b_i \left( \sum_{j=1}^{s} a_{ij} \right)^2 = \frac{1}{16} \neq \frac{1}{3} \sum_{i=1}^{s} b_i \sum_{j=1}^{s} a_{ij} \sum_{k=1}^{s} a_{jk} = \frac{1}{16} \neq \frac{1}{6}, \]
the method is exactly second-order accurate as expected.
Error estimate  IDEC produces an error estimate as well as several approximate solution values, since the error is approximated during the correction process. Thus the values of $e_{n+j}$ during the final pass can be used as a pessimistic estimate for the error in the final corrected solution values $u^p_{n+j}$.

Fixed point equivalent  IDEC can also be viewed as a fixed point iteration for the solution of an implicit Runge-Kutta method. The iteration updates $p$-vectors $u^k_{n+j} \rightarrow u^{k+1}_{n+j}$ by adding $e_{n+j}$. Hence if it converges as $k \rightarrow \infty$ then the limit $u_{n+j}$ is determined by setting $e_{n+j} = 0$ or

$$0 = 0 + hf(t_{n+j}, u_{n+j}) - hu'(t_{n+j}).$$

In terms of the dimensionless differentiation coefficients $d_{jk}$ such that

$$hu'(t_{n+j}) = \sum_{k=0}^{p} d_{jk}u_{n+k},$$

the limit $u$ must satisfy

$$\sum_{k=0}^{p} d_{jk}u_{n+k} = hf(t_{n+j}, u_{n+j})$$

for $1 \leq j \leq p$. So far this does not look like a Runge-Kutta method. However,

$$\sum_{k=0}^{p} d_{jk} = 0$$

since differentiating a constant gives 0. Hence

$$\sum_{k=0}^{p} d_{jk}u_{n} = 0$$

so $u$ must satisfy

$$\sum_{k=0}^{p} d_{jk}(u_{n+k} - u_n) = hf(t_{n+j}, u_{n+j}) = hk_j$$

where we have recognized the $p$ stages $k_j$ of a Runge-Kutta method. The $k = 0$ term is $d_{j0}(u_n - u_n) = 0$, giving a square linear system of equations

$$\sum_{k=1}^{p} d_{jk}(u_{n+k} - u_n) = hk_j$$

for the differences $u_{n+k} - u_n$. Let $c_{ij}$ be the elements of the inverse matrix $C = D^{-1}$ to the square $p \times p$ matrix $D$ with elements $d_{ij}$ for $1 \leq i, j \leq p$. (As the inverse of part of a differentiation matrix, $C$ is actually the integration matrix which takes $p$ values $v_{n+j}$ of a degree $p-1$ polynomial $v(t)$ to $p$ values $w_{n+j}$ of the degree-$p$ polynomial

$$w(t) = \int_{t_n}^{t} v(s)ds$$
with \( w(t_n) = 0 \). Then applying the inverse extracts the updates

\[
  u_{n+k} - u_n = h \sum_{j=1}^p c_{kj} k_j
\]

and we can rewrite \( k_j \) as

\[
  k_j = f(t_{n+j}, u_{n+j}) = f(t_{n+j}, u_n + ph \sum_{r=1}^p (c_{jr}/p) k_r).
\]

We recognize the matrix \( A = C/p \) of the Butcher array for a \( p \)-stage implicit Runge-Kutta method with step size \( ph \), and for \( j = p \) we recognize the row vector \( b^T \) as the last row of \( A \).

Question: What is the order of accuracy of this implicit Runge-Kutta method? In other words, what is the \( q \) such that the local truncation error

\[
  \tau = \frac{y_{n+p} - y_n}{ph} - \sum_{i=1}^p b_i k_i
\]

is exactly \( O(h^q) \) as \( h \to 0 \)? Here the stages \( k_i \) are the exact solutions of a system of implicit equations

\[
  k_i = f(t_{n+i}, y_n + ph \sum_{j=1}^p a_{ij} k_j).
\]

for \( 1 \leq i \leq p \). It is easier to answer this question if we eliminate the stages and write the implicit Runge-Kutta method in the primitive variables \( u_n \) through \( u_{n+p} \); Multiplying by \( D \) gives the previous equations

\[
  \frac{1}{h} \sum_{k=1}^p d_{jk}(u_{n+k} - u_n) = f(t_{n+j}, u_{n+j}).
\]

In this form, the local truncation error satisfies

\[
  \frac{1}{h} \sum_{k=1}^p d_{jk}(y_{n+k} - y_n) - f(t_{n+j}, y_{n+j}) = \tau_{n+j}.
\]

The sum on the left-hand side is the derivative \( Y'(t_{n+j}) \) of the degree-\( p \) polynomial \( Y(t) \) which interpolates \( y_n \) through \( y_{n+p} \), so we know

\[
  y'(t_{n+j}) - Y'(t_{n+j}) = \frac{y^{(p+1)}(\xi)}{p!} \prod_{i \neq j}(t_{n+j} - t_{n+i}) = O(h^p).
\]

Since \( y \) satisfies the differential equation

\[
  y'(t_{n+j}) = f(t_{n+j}, y_{n+j}),
\]

we have

\[
  \tau_{n+j} = O(h^p)
\]
and the method is order-$p$ accurate as expected.

Now we can recognize IDEC as a fixed point iteration for solving the vector equation $F(x) = 0$ where $x$ is the $p$-vector $x = (u_{n+1}, u_{n+2}, \ldots, u_{n+p})^T$ and $F(x) = 0$ is equivalent to the implicit Runge-Kutta method:

$$F(x)_j = \frac{1}{h} \sum_{k=1}^{p} d_{jk} (x_k - u_n) - f(t_{n+j}, x_j), \quad 1 \leq j \leq p.$$ 

Eliminating the correction $e = u^{k+1} - u^k$ from

$$e_{n+j+1} = e_{n+j} + h \left( f(t_{n+j}, u_{n+j}^k + e_{n+j}) - u'_k(t_{n+j}) \right), \quad 0 \leq j \leq p - 1,$$

gives a triangular system of nonlinear equations

$$u_{n+j+1}^{k+1} - u_{n+j}^{k+1} - hf(t_{n+j}, u_{n+j}^{k+1}) = u_{n+j+1}^{k} - u_{n+j}^{k} - hu'_k(t_{n+j})$$

for $0 \leq j \leq p - 1$. Solving this triangular system for $u^{k+1}$ defines the vector iteration function $G$ in $u^{k+1} = G(u^k)$. If the iteration converges as $k \to \infty$, then the differences cancel and the implicit Runge-Kutta method is satisfied.

Questions: Does the fixed point iteration actually converge? Is the convergence rate $O(h)$ so that IDEC gains one order of accuracy per pass, up to order $O(h^p)$?