Ordinary Differential Equations – Summary
This is a summary of our work on the initial value problem (IVP)
\[ x'(t) = f(t, x(t)) \]
\[ x(t_0) = x_0 \]
where \( t \) is time, and \( x(t) \) is a vector valued function.
That is
\[
\begin{pmatrix}
  x_1(t) \\
  x_2(t) \\
  \vdots \\
  x_n(t)
\end{pmatrix},
\]
\[
\begin{pmatrix}
  f_1(t, x) \\
  f_2(t, x) \\
  \vdots \\
  f_n(t, x)
\end{pmatrix}
\]

Adams Predictor–Corrector Methods
Get an initial guess using an \( N \)-step, \( N \)th order Adams–Bashforth explicit method
\[
x_{n+1}^{(0)} = x_n + h \sum_{k=1}^{N} \beta_k f_{n+1-k}
\]
where
\[
x_n \approx x(t_n), \quad f_n = f(t_n, x_n) \approx x'(t_n).
\]
The \( \beta_k \) are chosen so that
\[
\int_0^1 t^j dt = \beta_1 0^j + \beta_2 (-1)^j + \cdots + \beta_N (1 - N)^j, \quad j = 1, \ldots, N - 1
\]
When we say \( N \)th order, that means that if we start the IVP at the last step
\[
z'_n(t) = f(t, z_n(t))
\]
\[
z_n(t_n) = x_n
\]
then
\[
z_n(t_{n+1}) - x_{n+1}^{(0)} = \alpha_{N+1} z_n^{(N+1)}(t_n) h^{N+1} + O(h^{N+2}),
\]
Remember this is local truncation error, the error from one step. Global error is $O(h^N)$.

We then use an **Adams–Moulton Implicit Method** of order $N$ called a corrector:

$$
x_{n+1} = x_n + h\hat{\beta}_0 f(t_{n+1}, x_{n+1}) + h \sum_{k=1}^{N-1} \hat{\beta}_k f_{n+1-k} \quad (1)
$$

where the $\hat{\beta}_k$ are chosen so that

$$
\int_0^1 t^j dt = \hat{\beta}_0 1^j + \hat{\beta}_1 0^j + \cdots + \hat{\beta}_{N-1} (2-N)^j, \quad j = 1, \ldots, N-1.
$$

The local truncation error for this method is

$$
z_n(t_{n+1}) - x_{n+1} = \hat{\alpha}_{N+1} z_n^{(N+1)}(t_n) h^{N+1} + O(h^{N+2}).
$$

The constant $|\hat{\alpha}_{N+1}|$ is smaller than $|\alpha_{N+1}|$.

We have to solve the nonlinear equation (1). Newton’s method is an option, but it requires us to produce a Jacobian. Simpler is a fixed point iteration of the form

$$
x_{n+1}^{(k+1)} = x_n + h\hat{\beta}_0 f(t_{n+1}, x_k) + h \sum_{k=1}^{N-1} \hat{\beta}_k f_{n+1-k} \quad (2)
$$

When both the predictor and the corrector are $N$th order, one iteration is enough to get a local truncation error that is asymptotically as good as that for the exact solution to (1). That is,

$$
z_n(t_{n+1}) - x_{n+1}^{(1)} = z_n(t_{n+1}) - x_{n+1} = \hat{\alpha}_{N+1} z_n^{(N+1)}(t_n) h^{N+1} + O(h^{N+2}).
$$

**Example 1** The third order method from last time.

**Adams–Bashforth Predictor**

$$
x_{n+1}^{(0)} = x_n + \frac{h}{12} [23f_n - 16f_{n-1} + 5f_{n-2}]
$$

The local truncation error is

$$
z_n(t_{n+1}) - x_{n+1}^{(0)} = \alpha_4 z_n(t_n) h^4 + O(h^5), \quad \alpha_4 = 5/12.
$$

**Adams–Moulton Corrector**
For $k = 0, 1, \ldots$,

$$x_{n+1}^{(k+1)} = x_n + \frac{5h}{12} f(t_{n+1}, x_{n+1}^{(k)}) + \frac{h}{12} [8f_n - f_{n-1}].$$

For $k = 1$, we have

$$z_n(t_{n+1}) - x_{n+1}^{(1)} = \hat{\alpha}_4 z_n^{(4)}(t_{n+1}) h^4 + O(h^5) = z_n(t_{n+1}) - x_{n+1} + O(h^5).$$

The constant $\hat{\alpha}_4 = -1/12$. Thus one correction step is usually considered to be enough. So we do

$$x_{n+1}^{(1)} = x_n + \frac{5h}{12} f(t_{n+1}, x_{n+1}^{(0)}) + \frac{h}{12} [8f_n - f_{n-1}].$$

and

$$f_{n+1} = f(t_{n+1}, x_{n+1}^{(1)}).$$

We compare the Adams methods to Runge–Kutta methods. A typical method is the Runge–Kutta–Fehlberg (4,5) method. The coefficients are given in previous lectures so I will not give them here again. The general form is

$$k_i = f(t_n + c_i h, x_n + h \sum_{j=1}^{i-1} a_{ij} k_j), \quad i = 1, 2, 3, 4, 5, 6.$$

$$x_{n+1} = x_n + \sum_{i=1}^{5} b_i k_i, \quad \text{using the first 5 stages}$$

$$x_{n+1} = x_n + \sum_{i=1}^{6} \hat{b}_i \hat{k}_i$$

The local truncation error can be estimated using the fact that

$$z_n(t_{n+1}) - x_{n+1} = \hat{x}_{n+1} - x_{n+1} + O(h^6).$$

Thus the highest order term of the error is captured. Adams methods also have easy error estimates (which we did not have time to discuss).

The following is a run down of important aspects of Adams methods and Runge–Kutta methods. Experts could argue for hours on which of the two is more useful. I am an Adams method fan. The MATLAB ode codes are Runge–Kutta methods (ode23, ode45, and others). Octave uses lsode which is based upon Adams methods and a closely related class of methods called backward differentiation formulae.
Adams methods

- No need to use intermediate values in an interval.
- For a given order, fewer function evaluations are necessary.
- Useful methods up to order 7 or 8. (My opinion)
- Changing step size is HARD.
- Changing order is EASY (we did not cover this, but Adams codes are variable order).
- Not self–starting. They are often started with Runge–Kutta methods.
- Even the most basic code is not simple. They are not “black box” methods.

Runge–Kutta methods

- They are self–starting.
- They are very “black box” and easy to code (except for the most sophisticated codes). Easier to teach than Adams methods.
- Changing step size is EASY.
- Changing order is IMPOSSIBLE without choosing a completely different Runge–Kutta method.
- For a given order, large step sizes are possible, but more function evaluations are necessary.
- Useful methods of order up to 6. (my opinion)