Higher order & implicit methods for ODEs

- Higher order method using Leap-frog: modified-midpoint & Bulirsch-Stoer methods
- Stiff equations
- Higher order semi-implicit extrapolation method

The modified-midpoint/Gragg method

- leap-frog method has a total error in solution that has only even powers of h

The modified-midpoint/Gragg method

- leap-frog method has a total error in solution that has only even powers of h
- because of time-reversal symmetry, odd single-step error $\epsilon(h) = c_3 h^3 + c_5 h^5 + ...$

The modified-midpoint/Gragg method

- leap-frog method has a total error in solution that has only even powers of h
- because of time-reversal symmetry, odd single-step error $\epsilon(h) = c_3 h^3 + c_5 h^5 + ...$
- total cumulative error is an order worse, so even in h

The modified-midpoint/Gragg method

- leap-frog method has a total error in solution that has only even powers of h
- because of time-reversal symmetry, odd single-step error $\epsilon(h) = c_3 h^3 + c_5 h^5 + \ldots$
- total cumulative error is an order worse, so even in h
- the first Euler step: $x_0 = x(t);\; y_1 = x_0 + (h/2) f(x_0, t)$.

The modified-midpoint/Gragg method

- leap-frog method has a total error in solution that has only even powers of h
- because of time-reversal symmetry, odd single-step error $\epsilon(h) = c_3 h^3 + c_5 h^5 + ...$
- total cumulative error is an order worse, so even in h
- the first Euler step: $x_0 = x(t);\; y_1 = x_0 + (h/2) f(x_0, t)$.
- leap-frog after this: $y_{m+1} = y_m + hf(x_m, t + mh);\; x_{m+1} = x_m + hf(y_{m+1}, t + (m + 1/2)h)$.

The modified-midpoint/Gragg method

- leap-frog method has a total error in solution that has only even powers of h
- because of time-reversal symmetry, odd single-step error $\epsilon(h) = c_3 h^3 + c_5 h^5 + \ldots$
- total cumulative error is an order worse, so even in h
- the first Euler step: $x_0 = x(t);\; y_1 = x_0 + (h/2) f(x_0, t)$.
- leap-frog after this: $y_{m+1} = y_m + h f(x_m, t + mh);\; x_{m+1} = x_m + h f(y_{m+1}, t + (m + 1/2)h)$.
- $x(t + H) = (x_n + y_n + (h/2) f(x_n, t + H))/2$, an average at the end-point; $h = H/n$.

The modified-midpoint/Gragg method

- leap-frog method has a total error in solution that has only even powers of h
- because of time-reversal symmetry, odd single-step error $\epsilon(h) = c_3 h^3 + c_5 h^5 + ...$
- total cumulative error is an order worse, so even in h
- the first Euler step: $x_0 = x(t);\; y_1 = x_0 + (h/2)f(x_0, t)$.
- leap-frog after this: $y_{m+1} = y_m + hf(x_m, t + mh);\; x_{m+1} = x_m + hf(y_{m+1}, t + (m + 1/2)h)$.
- $x(t + H) = (x_n + y_n + (h/2)f(x_n, t + H))/2$, an average at the end-point; $h = H/n$.
- This gives only even terms in the cumulative error with h^2 leading order – Gragg's modified-midpoint method.

Bulirsch-Stoer method

- Richardson extrapolation + the even error terms in Gragg's method.

Bulirsch-Stoer method

- Richardson extrapolation + the even error terms in Gragg's method.
- useful for smooth, non-stiff solutions

Bulirsch-Stoer method

- Richardson extrapolation + the even error terms in Gragg's method.
- useful for smooth, non-stiff solutions
- Choose a time interval H and divide into $h_n = H/n$ subintervals ($n = 1, 2, 3, ...$): gives $R_{1,1},\ R_{2,1},\ R_{3,1},$ ...

Bulirsch-Stoer method

- Richardson extrapolation + the even error terms in Gragg's method.
- useful for smooth, non-stiff solutions
- Choose a time interval H and divide into $h_n = H/n$ subintervals $(n = 1, 2, 3, ...)$: gives $R_{1,1},\ R_{2,1},\ R_{3,1},$ ...
- Combine them using Richardson extrapolation

$$
\begin{array}{lccccccc}
n=1: & R_{1,1} & & & & & & \\
 & & \searrow & & & & & \\
n=2: & R_{2,1} & \rightarrow & R_{2,2} & & & & \\
 & & \searrow & & \searrow & & & \\
n=3: & R_{3,1} & \rightarrow & R_{3,2} & \rightarrow & R_{3,3} & & \\
 & & \searrow & & \searrow & & \searrow & \\
n=4: & R_{4,1} & \rightarrow & R_{4,2} & \rightarrow & R_{4,3} & \rightarrow & R_{4,4} \\
 & \underbrace{\phantom{R_{4,1}}}_{\substack{\text{Modified}\\ \text{midpoint}}} & & \multicolumn{5}{c}{\underbrace{\phantom{R_{4,2} \rightarrow R_{4,3} \rightarrow R_{4,4}}}_{\text{Richardson extrapolation}}}
\end{array}
$$

Bulirsch-Stoer method

- $x(t+H) = R_{1,1} + c_1 h_1^2 + \mathcal{O}(h_1^4) = R_{2,1} + c_1 h_2^2 + \mathcal{O}(h_2^4)$

Bulirsch-Stoer method

- $x(t+H) = R_{1,1} + c_1 h_1^2 + \mathcal{O}(h_1^4) = R_{2,1} + c_1 h_2^2 + \mathcal{O}(h_2^4)$
- leading order error in $R_{2,1}$, $c_1 h_2^2 = (R_{2,1} - R_{1,1})/3$

Bulirsch-Stoer method

- $x(t+H) = R_{1,1} + c_1 h_1^2 + \mathcal{O}(h_1^4) = R_{2,1} + c_1 h_2^2 + \mathcal{O}(h_2^4)$
- leading order error in $R_{2,1}$, $c_1 h_2^2 = (R_{2,1} - R_{1,1})/3$
- $x(t+H) = R_{2,1} + (R_{2,1} - R_{1,1})/3 + \mathcal{O}(h_2^4)$

Bulirsch-Stoer method

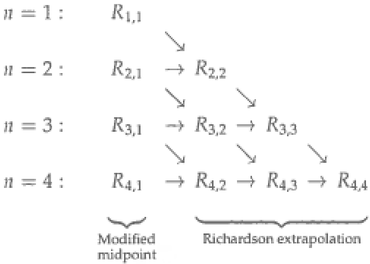
- $x(t+H) = R_{1,1} + c_1 h_1^2 + \mathcal{O}(h_1^4) = R_{2,1} + c_1 h_2^2 + \mathcal{O}(h_2^4)$
- leading order error in $R_{2,1}$, $c_1 h_2^2 = (R_{2,1} - R_{1,1})/3$
- $x(t+H) = R_{2,1} + (R_{2,1} - R_{1,1})/3 + \mathcal{O}(h_2^4)$
- a higher order estimate $R_{2,2} = R_{2,1} + (R_{2,1} - R_{1,1})/3$

Bulirsch-Stoer method

- $x(t+H) = R_{1,1} + c_1 h_1^2 + \mathcal{O}(h_1^4) = R_{2,1} + c_1 h_2^2 + \mathcal{O}(h_2^4)$
- leading order error in $R_{2,1}$, $c_1 h_2^2 = (R_{2,1} - R_{1,1})/3$
- $x(t+H) = R_{2,1} + (R_{2,1} - R_{1,1})/3 + \mathcal{O}(h_2^4)$
- a higher order estimate $R_{2,2} = R_{2,1} + (R_{2,1} - R_{1,1})/3$
- proceeding further in this way:
  $R_{n,m+1} = R_{n,m} + \frac{R_{n,m} - R_{n-1,m}}{[n/(n-1)]^{2m} - 1}$; impose error tolerance to choose n

Bulirsch-Stoer method

- $x(t+H) = R_{1,1} + c_1 h_1^2 + \mathcal{O}(h_1^4) = R_{2,1} + c_1 h_2^2 + \mathcal{O}(h_2^4)$
- leading order error in $R_{2,1}$, $c_1 h_2^2 = (R_{2,1} - R_{1,1})/3$
- $x(t+H) = R_{2,1} + (R_{2,1} - R_{1,1})/3 + \mathcal{O}(h_2^4)$
- a higher order estimate $R_{2,2} = R_{2,1} + (R_{2,1} - R_{1,1})/3$
- proceeding further in this way: $R_{n,m+1} = R_{n,m} + \frac{R_{n,m} - R_{n-1,m}}{[n/(n-1)]^{2m} - 1}$; impose error tolerance to choose n
- typically n shouldn't be more than 10 or so



Stiff equations

- for higher than first order ODEs, one can have stiff equations; e.g., chemical, nuclear reactions

Stiff equations

- for higher than first order ODEs, one can have stiff equations; e.g., chemical, nuclear reactions
- dependent variable changes on very different timescales

Stiff equations

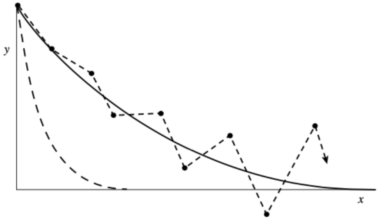
- for higher than first order ODEs, one can have stiff equations; e.g., chemical, nuclear reactions
- dependent variable changes on very different timescales
- with explicit method $\Delta t \leq 2/\lambda_{\max}$; for stability must resolve fastest variation even after the mode decays

Stiff equations

- for higher than first order ODEs, one can have stiff equations; e.g., chemical, nuclear reactions
- dependent variable changes on very different timescales
- with explicit method $\Delta t \le 2/\lambda_{\max}$; for stability must resolve fastest variation even after the mode decays
- Example: $u' = 998u + 1998v$; $v' = -999u - 1999v$ with $u(0) = 1$ and $v(0) = 0$

Stiff equations

- for higher than first order ODEs, one can have stiff equations; e.g., chemical, nuclear reactions
- dependent variable changes on very different timescales
- with explicit method $\Delta t \le 2/\lambda_{\max}$; for stability must resolve fastest variation even after the mode decays
- Example: $u' = 998u + 1998v$; $v' = -999u - 1999v$ with $u(0) = 1$ and $v(0) = 0$
- Analytic solution: $u = 2e^{-x} - e^{-1000x}$; $v = -e^{-x} + e^{-1000x}$; one term decays 1000 times faster!



Implicit methods for stiff equations

- $\mathbf{y}' = \mathbf{f}(\mathbf{y})$

Implicit methods for stiff equations

- $\mathbf{y}' = \mathbf{f}(\mathbf{y})$
- implicit differencing gives $\mathbf{y}_{n+1} = \mathbf{y}_n + h\mathbf{f}(\mathbf{y}_{n+1})$, an implicit nonlinear equation

Implicit methods for stiff equations

- $\mathbf{y}' = \mathbf{f}(\mathbf{y})$
- implicit differencing gives $\mathbf{y}_{n+1} = \mathbf{y}_n + h\mathbf{f}(\mathbf{y}_{n+1})$, an implicit nonlinear equation
- linearizing, as in Newton's method with one step:

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\left[\mathbf{f}(\mathbf{y}_n) + \left.\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right|_{\mathbf{y}_n} \cdot (\mathbf{y}_{n+1} - \mathbf{y}_n)\right],$$

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\left[\mathbf{I} - h\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right]^{-1} \cdot \mathbf{f}(\mathbf{y})_n$$

Implicit methods for stiff equations

- $\mathbf{y}' = \mathbf{f}(\mathbf{y})$
- implicit differencing gives $\mathbf{y}_{n+1} = \mathbf{y}_n + h\mathbf{f}(\mathbf{y}_{n+1})$, an implicit nonlinear equation
- linearizing, as in Newton's method with one step:

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\left[\mathbf{f}(\mathbf{y}_n) + \left.\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right|_{\mathbf{y}_n} \cdot (\mathbf{y}_{n+1} - \mathbf{y}_n)\right],$$

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\left[\mathbf{I} - h\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right]^{-1} \cdot \mathbf{f}(\mathbf{y})_n$$

- a matrix equation

Implicit methods for stiff equations

- $\mathbf{y}' = \mathbf{f}(\mathbf{y})$
- implicit differencing gives $\mathbf{y}_{n+1} = \mathbf{y}_n + h\mathbf{f}(\mathbf{y}_{n+1})$, an implicit nonlinear equation
- linearizing, as in Newton's method with one step:
  $$\mathbf{y}_{n+1} = \mathbf{y}_n + h\left[\mathbf{f}(\mathbf{y}_n) + \left.\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right|_{\mathbf{y}_n} \cdot (\mathbf{y}_{n+1} - \mathbf{y}_n)\right],$$
  $$\mathbf{y}_{n+1} = \mathbf{y}_n + h\left[\mathbf{I} - h\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right]^{-1} \cdot \mathbf{f}(\mathbf{y})_n$$
- a matrix equation
- solving implicit equations by linearization known as semi-implicit method

Implicit methods for stiff equations

- $\mathbf{y}' = \mathbf{f}(\mathbf{y})$
- implicit differencing gives $\mathbf{y}_{n+1} = \mathbf{y}_n + h\mathbf{f}(\mathbf{y}_{n+1})$, an implicit nonlinear equation
- linearizing, as in Newton's method with one step:

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\left[\mathbf{f}(\mathbf{y}_n) + \left.\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right|_{\mathbf{y}_n} \cdot (\mathbf{y}_{n+1} - \mathbf{y}_n)\right],$$

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\left[\mathbf{I} - h\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right]^{-1} \cdot \mathbf{f}(\mathbf{y})_n$$

- a matrix equation
- solving implicit equations by linearization known as semi-implicit method
- for explicit dependence on independent variable, add one more equation

$$\begin{pmatrix} \mathbf{y} \\ x \end{pmatrix}' = \begin{pmatrix} \mathbf{f} \\ 1 \end{pmatrix}$$

Implicit methods for stiff equations

- $\mathbf{y}' = \mathbf{f}(\mathbf{y})$
- implicit differencing gives $\mathbf{y}_{n+1} = \mathbf{y}_n + h\mathbf{f}(\mathbf{y}_{n+1})$, an implicit nonlinear equation
- linearizing, as in Newton's method with one step:

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\left[\mathbf{f}(\mathbf{y}_n) + \left.\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right|_{\mathbf{y}_n} \cdot (\mathbf{y}_{n+1} - \mathbf{y}_n)\right],$$

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\left[\mathbf{I} - h\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right]^{-1} \cdot \mathbf{f}(\mathbf{y})_n$$

- a matrix equation
- solving implicit equations by linearization known as semi-implicit method
- for explicit dependence on independent variable, add one more equation

$$\begin{pmatrix} \mathbf{y} \\ x \end{pmatrix}' = \begin{pmatrix} \mathbf{f} \\ 1 \end{pmatrix}$$

- error control by trying h and $2h$ and estimating error

Implicit methods for stiff equations

- $\mathbf{y}' = \mathbf{f}(\mathbf{y})$
- implicit differencing gives $\mathbf{y}_{n+1} = \mathbf{y}_n + h\mathbf{f}(\mathbf{y}_{n+1})$, an implicit nonlinear equation
- linearizing, as in Newton's method with one step:

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\left[\mathbf{f}(\mathbf{y}_n) + \left.\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right|_{\mathbf{y}_n} \cdot (\mathbf{y}_{n+1} - \mathbf{y}_n)\right],$$

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\left[\mathbf{I} - h\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right]^{-1} \cdot \mathbf{f}(\mathbf{y})_n$$

- a matrix equation
- solving implicit equations by linearization known as semi-implicit method
- for explicit dependence on independent variable, add one more equation

$$\begin{pmatrix} \mathbf{y} \\ x \end{pmatrix}' = \begin{pmatrix} \mathbf{f} \\ 1 \end{pmatrix}$$

- error control by trying h and $2h$ and estimating error
- semi-implicit extrapolation schemes like BS available for

Other methods

- multipoint predictor-corrector, requires dependent variable at multiple points; e.g., Adams-Bashforth-Moulton

Other methods

- multipoint predictor-corrector, requires dependent variable at multiple points; e.g., Adams-Bashforth-Moulton
- sophisticated, higher order methods built on ideas presented here

Other methods

- multipoint predictor-corrector, requires dependent variable at multiple points; e.g., Adams-Bashforth-Moulton
- sophisticated, higher order methods built on ideas presented here
- black-box solvers are based on such sophisticated schemes