3, Ordinary Differential Equations

$$\frac{\mathrm{d}^2 y}{\mathrm{d}x^2} + q(x)\frac{\mathrm{d}y}{\mathrm{d}x} = r(x),$$

- Euler's method
- Runge-Kutta method
- Adaptive stepsize control
- Predictor-corrector method
- Boundary value problems
- Relaxation method

Finite difference approximation

Second-order FD approximation for $u'(x_j)$

$$u'(x_j) \approx \frac{u_{j+1} - u_{j-1}}{2h}$$

in the matrix-vector form (with periodic boundary)

$$\begin{pmatrix} u'_1 \\ \vdots \\ u'_j \\ \vdots \\ u'_N \end{pmatrix} = h^{-1} \begin{pmatrix} 0 & \frac{1}{2} & & & -\frac{1}{2} \\ -\frac{1}{2} & 0 & & & \\ & \ddots & & & \\ & -\frac{1}{2} & 0 & \frac{1}{2} & \\ & & \ddots & & \\ & & & 0 & \frac{1}{2} \\ \frac{1}{2} & & & -\frac{1}{2} & 0 \end{pmatrix} \begin{pmatrix} u_1 \\ \vdots \\ u_j \\ \vdots \\ u_N \end{pmatrix}$$

Finite difference approximation

Taylor's expansion for $u(x)$ at x_{j+1},

$$u(x_{j+1}) = u(x_j) + u'(x_j)(x_{j+1} - x_j) + \frac{u''(x_j)}{2}(x_{j+1} - x_j)^2 + \frac{u'''(x_j)}{3!}(x_{j+1} - x_j)^3 + \ldots,$$

one can approximate $u'(x_j)$ by

$$\begin{aligned} u'(x_j) &= \frac{u(x_{j+1}) - u(x_j)}{x_{j+1} - x_j} - \frac{u''(x_j)}{2}(x_{j+1} - x_j) - \frac{u'''(x_j)}{3!}(x_{j+1} - x_j)^2 + \ldots, \\ &\approx \frac{u(x_{j+1}) - u(x_j)}{x_{j+1} - x_j} + \mathbf{O}(\Delta x), \end{aligned}$$

by the same way the Taylor's expansion for $u(x)$ at x_{j-1},

$$u(x_{j-1}) = u(x_j) + u'(x_j)(x_{j-1} - x_j) + \frac{u''(x_j)}{2}(x_{j-1} - x_j)^2 + \frac{u'''(x_j)}{3!}(x_{j-1} - x_j)^3 + \ldots,$$

one can approximate $u'(x_j)$ by

$$\begin{aligned} u'(x_j) &= \frac{u(x_j) - u(x_{j-1})}{x_j - x_{j-1}} - \frac{u''(x_j)}{2}(x_j - x_{j-1}) + \frac{u'''(x_j)}{3!}(x_j - x_{j-1})^2 + \ldots, \\ &\approx \frac{u(x_j) - u(x_{j-1})}{x_j - x_{j-1}} + \mathbf{O}(\Delta x), \end{aligned}$$

Finite difference approximation

combine

$$
\begin{aligned}
u(x_{j+1}) &= u(x_j) + u'(x_j)\Delta x + \frac{u''(x_j)}{2}(\Delta x)^2 + \frac{u'''(x_j)}{3!}(\Delta x)^3 + \ldots, \\
u(x_{j-1}) &= u(x_j) - u'(x_j)\Delta x + \frac{u''(x_j)}{2}(\Delta x)^2 - \frac{u'''(x_j)}{3!}(\Delta x)^3 + \ldots,
\end{aligned}
$$

one can approximate $u'(x_j)$ by

$$
\begin{aligned}
u'(x_j) &= \frac{u(x_{j+1}) - u(x_{j-1})}{2\Delta x} - \frac{u'''(x_j)}{2 * 3!}(\Delta x)^2 + \ldots, \\
&\approx \frac{u(x_{j+1}) - u(x_{j-1})}{2\Delta x} + \mathbf{O}(\Delta x^2),
\end{aligned}
$$

- 4th-order approximation
- Runge-Kutta method
- differential matrix

By local interpolation

For $j = 1, 2, \ldots, N$:

- Let p_j be the unique polynomial of degree ≤ 2 with $p_j(x_{j-1}) = u_{j-1}$,$p_j(x_j) = u_j$, and $p_j(x_{j+1}) = u_{j+1}$.
- $u'(x_j) = p'_j(x)$.

Then the interpolant p_j is given by

$$p_j(x) = u_{j-1}a_{-1}(x) + u_j a_0(x) + u_{j+1}a_1(x)$$

where

$$\begin{aligned} a_{-1}(x) &= (x - x_j)(x - x_{j+1})/2h^2 \\ a_0(x) &= -(x - x_{j-1})(x - x_{j+1})/2h^2 \\ a_1(x) &= (x - x_{j-1})(x - x_j)/2h^2 \end{aligned}$$

Generalization to fourth-order

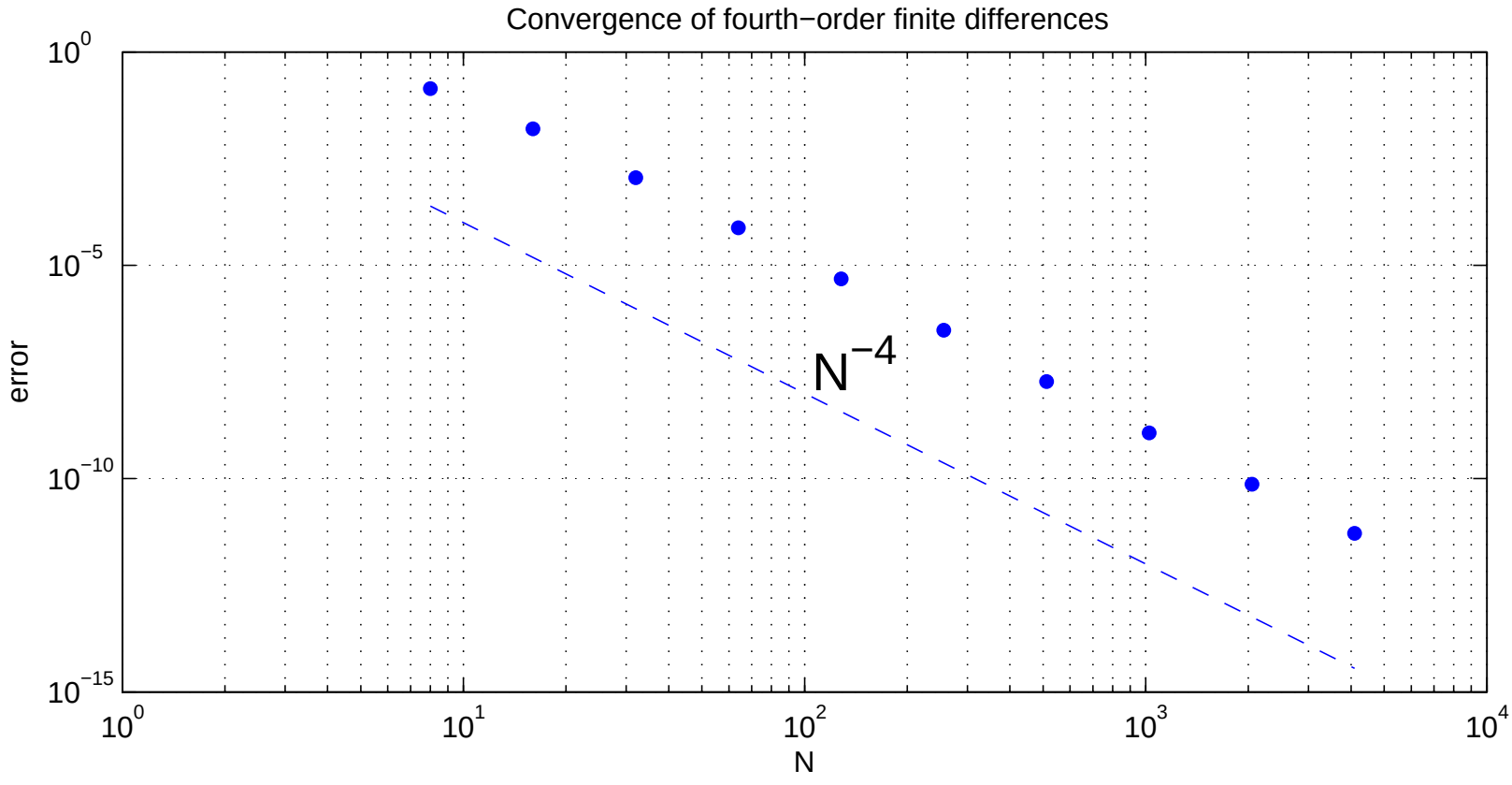
For $j = 1, 2, \ldots, N$:

- Let p_j be the unique polynomial of degree ≤ 2 with $p_j(x_{j\pm1}) = u_{j\pm1}$,$p_j(x_j) = u_j$, and $p_j(x_{j\pm2}) = u_{j\pm2}$.
- $u'(x_j) = p_j'(x)$.

$$\begin{pmatrix} u_1' \\ \vdots \\ \\ u_j' \\ \\ \vdots \\ u_N' \end{pmatrix} = h^{-1} \begin{pmatrix} & & \ddots & & \\ & \ddots & -\frac{1}{12} & \ddots & \\ & \ddots & \frac{2}{3} & \ddots & \\ \frac{1}{12} & -\frac{2}{3} & 0 & \frac{2}{3} & -\frac{1}{12} \\ & \ddots & -\frac{2}{3} & \ddots & \\ & \ddots & \frac{1}{12} & \ddots & \\ & & \ddots & & \end{pmatrix} \begin{pmatrix} u_1 \\ \vdots \\ \\ u_j \\ \\ \vdots \\ u_N \end{pmatrix}$$

Test of the convergence: FD

$$u(x) = e^{\sin(x)}, Du(x) = u'(x).$$



Euler's method

For a first-order differential equation:

$$y'(t) + a\,y(t) = r, \quad \text{with} \quad y(0) = y_0,$$

has the following analytical solution,

$$y(t) = (y_0 - \frac{r}{a})e^{-at} + \frac{r}{a}.$$

Euler's method:

$$\frac{y(t+h) - y(t)}{h} + a\,y(t) = r,$$
$$y(t+h) = [1 - ah]y(t) + hr, \quad \text{with} \quad y(0) = y_0.$$

with error of $\mathbf{O}(h)$

Heun's trapezoidal method

For a first-order differential equation:

$$y'(t) = f(t, y),$$
$$y(t)|_{t_k}^{t_{k+1}} = y(t_{k+1}) - y(t_k) = \int_{t_k}^{t_{k+1}} f(t, y)\mathrm{d}t,$$
$$y(t_{k+1}) = y(t_k) + \int_{t_k}^{t_{k+1}} f(t, y)\mathrm{d}t, \quad \text{with} \quad y(t_0) = y_0.$$

By trapezoidal integration method:

$$y_{k+1} = y_k + \frac{h}{2}[f(t_k, y_k), f(t_{k+1}, y_{k+1})]$$

where y_{k+1} is unknown and can be replaced by $y_{k+1} \approx y_k + hf(t_k, y_k)$.
Heun's method:

$$y_{k+1} = y_k + \frac{h}{2}[f(t_k, y_k), f(t_{k+1}, y_k + hf(t_k, y_k))]$$

with error of $\mathbf{O}(h^2)$

Runge-Kutta method

Second-order Runge-Kutta method:

$$
\begin{aligned}
k_1 &= hf(t_k, y_k), \\
k_2 &= hf(t_k + \frac{1}{2}h, y_k + \frac{1}{2}k_1), \\
y_{k+1} &= y_k + k_2 + \mathbf{O}(h^3).
\end{aligned}
$$

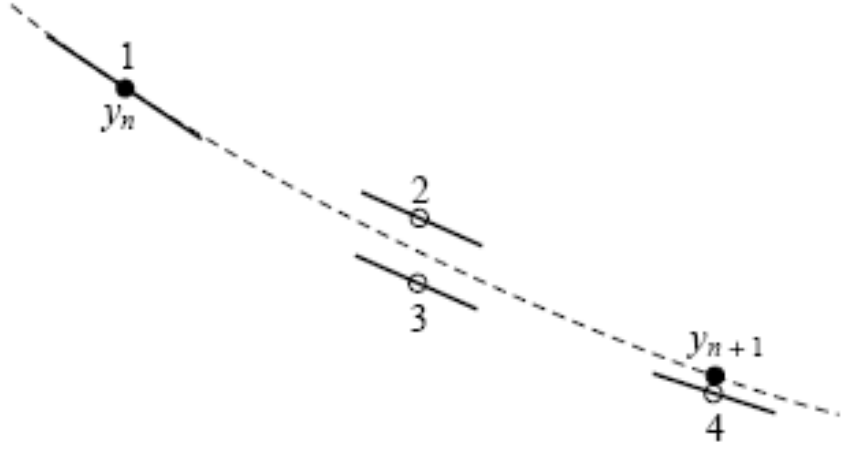
Fourth-order Runge-Kutta method:

$$
\begin{aligned}
k_1 &= hf(t_k, y_k), \\
k_2 &= hf(t_k + \frac{1}{2}h, y_k + \frac{1}{2}k_1), \\
k_3 &= hf(t_k + \frac{1}{2}h, y_k + \frac{1}{2}k_2), \\
k_2 &= hf(t_k + h, y_k + k_3), \\
y_{k+1} &= y_k + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6} + \mathbf{O}(h^5).
\end{aligned}
$$

4th-order Runge-Kutta method

Fourth-order Runge-Kutta method:

$$
\begin{aligned}
k_1 &= hf(t_k, y_k), \\
k_2 &= hf(t_k + \frac{1}{2}h, y_k + \frac{1}{2}k_1), \\
k_3 &= hf(t_k + \frac{1}{2}h, y_k + \frac{1}{2}k_2), \\
k_2 &= hf(t_k + h, y_k + k_3), \\
y_{k+1} &= y_k + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6} + \mathbf{O}(h^5).
\end{aligned}
$$

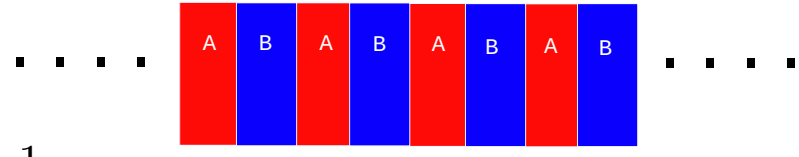


Boundary value problems: 1D Bragg reflector

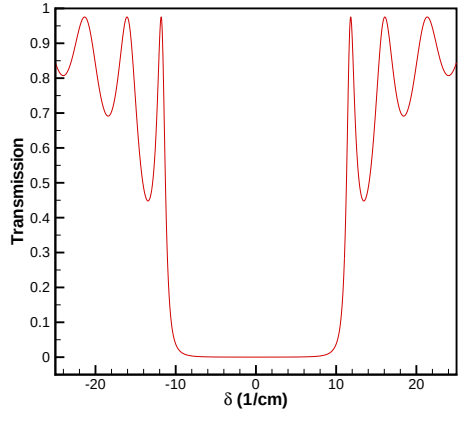
coupled-mode equation:

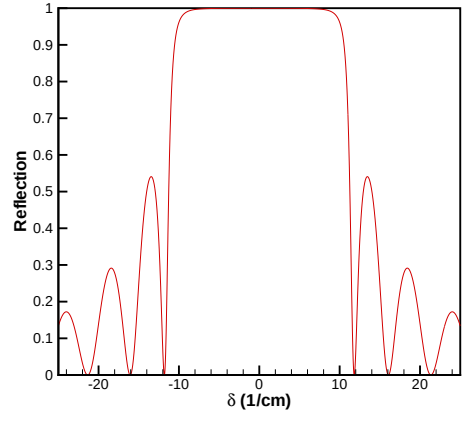
$$\frac{dE_+(z)}{dz} = i\delta E_+(z) + i\kappa E_-(z)$$
$$\frac{dE_-(z)}{dz} = -i\delta E_-(z) - i\kappa^* E_+(z)$$

with the Boundary Condition:



$$E_+(z=0) = 1$$
$$E_-(z=L) = 0$$





Boundary value problems: Homework #1

detuning parameter:

$$\delta = \frac{2\pi n_{\text{eff}}}{\lambda} - \frac{\pi}{\Lambda}$$

where $n_{\text{eff}} = 1.45$, $\Lambda = 0.0005$ [mm]. coupling constant:

$$\kappa(z) = \kappa_0 * \mathbf{F}(z),$$

where $\kappa_0 = 0.5$ [1/mm].

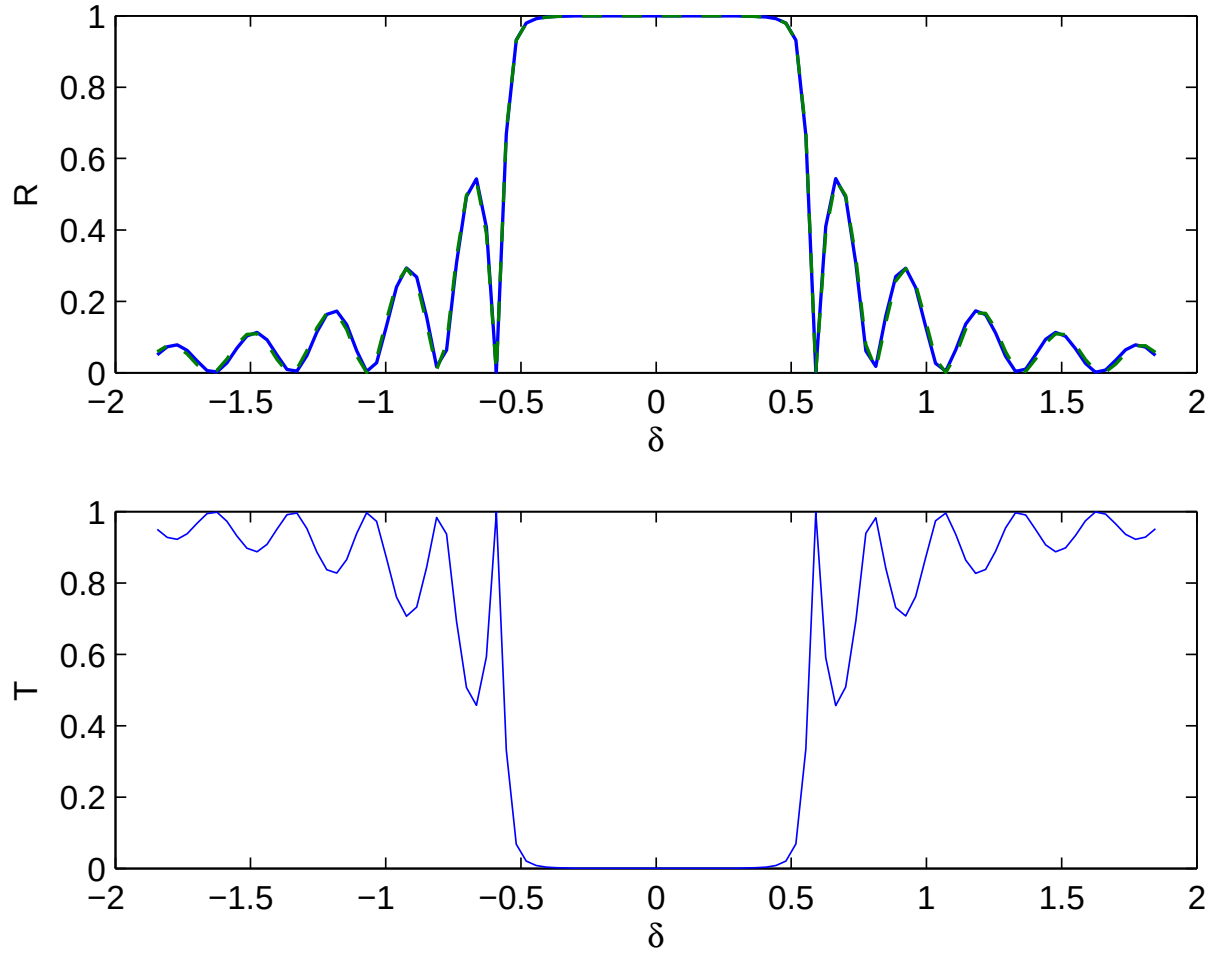
- $\mathbf{F}(z) = 1$, i.e. κ is a constant: calculate
  1. tranmission spectrum: $T \equiv |\frac{E_+(z=L)}{E_+(z=0)}|^2$ v.s. detuning δ;
  2. reflection spectrum: $R \equiv |\frac{E_-(z=0)}{E_+(z=0)}|^2$ v.s. detuning δ.
  3. compare to the analytical solution,

$$R = \frac{\sinh^2 \sqrt{(\kappa L)^2 - (\delta L)^2}}{-\frac{\delta^2}{\kappa^2} + \cosh^2 \sqrt{(\kappa L)^2 - (\delta L)^2}},$$

where $L = 10$[mm] is the grating length, and estimate the absolute error in your simulation.

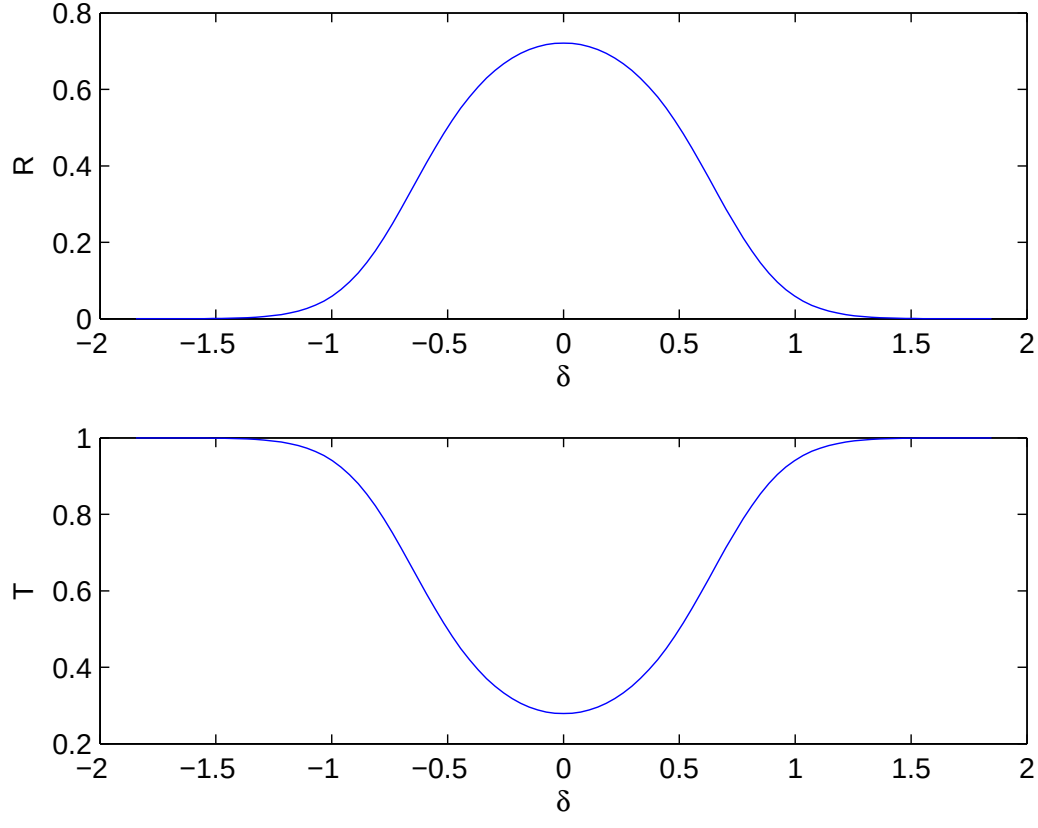
Boundary value problems: Homework #1

detuning windows: $-2.0 < \delta < 2.0$



Boundary value problems: Homework #1

- **F**$(z) = exp[-0.5 * (z - L/2)^2]$, i.e. *apodization*: calculate
  1. tranmission spectrum: $|\frac{E_+(z=L)}{E_+(z=0)}|^2$ v.s. detuning δ;
  2. reflection spectrum: $|\frac{E_-(z=0)}{E_+(z=0)}|^2$ v.s. detuning δ.



Adaptive stepsize control: step doubling

There are two approximate solutions from x to $x + 2h$,

1. one step $2h$:

$$y(x + 2h) = y_1 + (2h)^5\phi + \mathbf{O}(h^6) + \dots$$

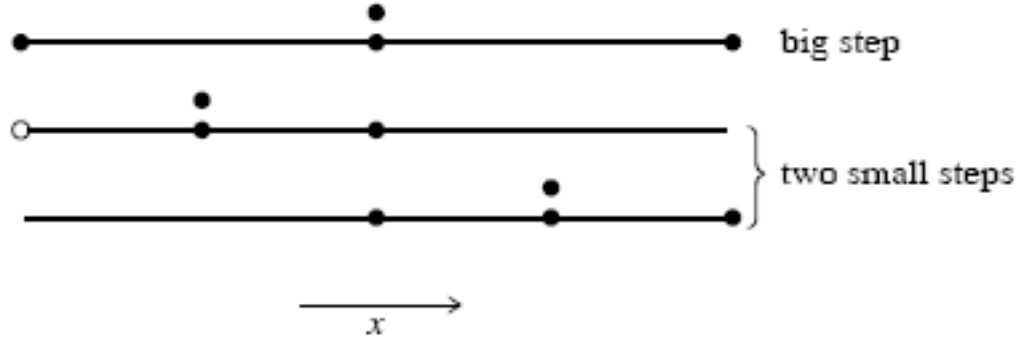
2. 2 steps each of size h:

$$y(x + 2h) = y_1 + 2(h)^5\phi + \mathbf{O}(h^6) + \dots$$

The difference between the two numerical estimates is a convenient *indicator* of truncation error,

$$\Delta \equiv y_2 - y1$$

We can keep a desired degree of accuracy by adjusting h with the cost of 11 evolutions (compared to 8).



Embedded Runge-Kutta formulas

The general form of a fifth-order Runge-Kutta formula is

$$
\begin{aligned}
k_1 &= h\, f(x_n, y_n) \\
k_2 &= h\, f(x_n + a_2 h, y_n + b_{21} k_1) \\
&\cdots \\
k_6 &= h\, f(x_n + a_6 h, y_n + b_{61} k_1 + \cdots + b_{65} k_5)
\end{aligned}
$$

$$
y_{n+1} = y_n + c_1 k_1 + c_2 k_2 + c_3 k_3 + c_4 k_4 + c_5 k_5 + c_6 k_6 + \mathbf{O}(h^6)
$$

The embedded fourth-order formula is

$$
y^*_{n+1} = y_n + c^*_1 k_1 + c^*_2 k_2 + c^*_3 k_3 + c^*_4 k_4 + c^*_5 k_5 + c^*_6 k_6 + \mathbf{O}(h^5)
$$

and the error estimate is

$$
\Delta \equiv y_{n+1} - y^*_{n+1} = \sum_{i=1}^{6} (c_i - c^*_i) k_i
$$

Cash-Karp parameters

Cash-Karp Parameters for Embedded Runga-Kutta Method

i	a_i	b_{ij}					c_i	c_i^*
1							$\frac{37}{378}$	$\frac{2825}{27648}$
2	$\frac{1}{5}$	$\frac{1}{5}$					0	0
3	$\frac{3}{10}$	$\frac{3}{40}$	$\frac{9}{40}$				$\frac{250}{621}$	$\frac{18575}{48384}$
4	$\frac{3}{5}$	$\frac{3}{10}$	$-\frac{9}{10}$	$\frac{6}{5}$			$\frac{125}{594}$	$\frac{13525}{55296}$
5	1	$-\frac{11}{54}$	$\frac{5}{2}$	$-\frac{70}{27}$	$\frac{35}{27}$		0	$\frac{277}{14336}$
6	$\frac{7}{8}$	$\frac{1631}{55296}$	$\frac{175}{512}$	$\frac{575}{13824}$	$\frac{44275}{110592}$	$\frac{253}{4096}$	$\frac{512}{1771}$	$\frac{1}{4}$
$j =$		1	2	3	4	5		

Embedded Runge-Kutta formulas

- Δ scales as h^5.
- If we take a step h_1 and produce an error Δ_1, the next step h_2 is estimated as

$$h_2 = h_1 |\frac{\Delta_2}{\Delta_1}|^2,$$

  where Δ_2 is the desired accuracy.
- If Δ_1 is larger than Δ_2 in magnitude, the equation tells how much to decrease the stepsize.
- If Δ_1 is smaller than Δ_2, the equation tells how much we can safely increase the stepsize for the next step.

Stiff sets of equations

Stiff occurs when there are two or more very different scales of the independent variable which the dependent variables are changing. For example:

$$\begin{aligned} u' &= 998u + 1998v \\ v' &= -999u - 1999u \end{aligned}$$

with boundary conditions

$$u(0) = 1 \qquad v(0) = 0$$

By means of the transformation

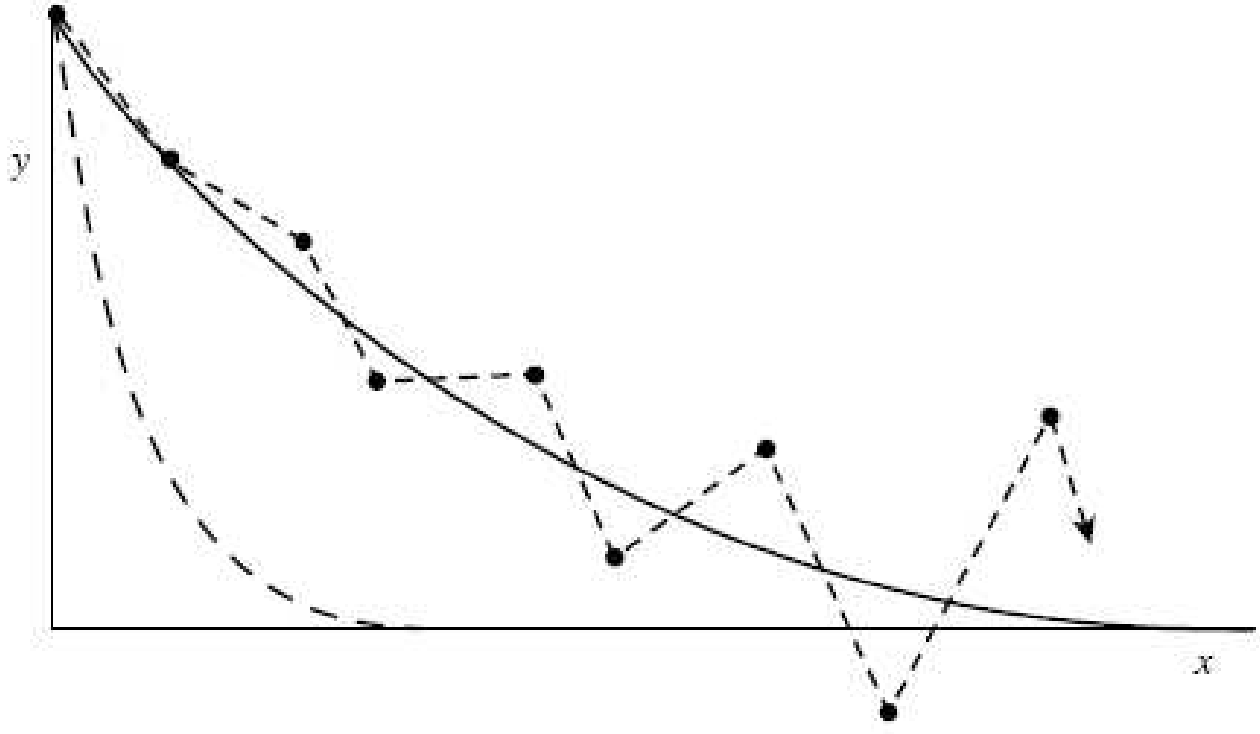
$$u = 2y - z \qquad v = -y + z$$

we find the solution

$$u = 2e^{-x} - e{-1000x}$$
$$u = -e^{-x} + e{-1000x}$$

Stiff sets of equations

If we integrated the system with any of the methods given so far, the presence of the e^{-1000x} term would require a stepsize $h \ll 1/1000$ for the method to be stable.



To cure this problem, one may use *implicit integration* methods.

Implicit integration methods

Consider the single equation,

$$y' = -cy$$

where $c > 0$ is a constant. The explicit (or forward) Euler scheme for integrating this equation with stepsize h is

$$y_{n+1} = y_n + hy'_n = (1 - ch)y_n.$$

Clearly this method is unstable if $h > 2/c$, for then $|y_n| \to \infty$ as $n \to \infty$.
The simplest cure is to resort *implicit* differencing, backward Euler scheme,

$$y_{n+1} = y_n + hy'_{n+1}$$

or

$$y_{n+1} = \frac{y_n}{1 + ch}$$

This method is absolutely stable: even as $h \to \infty$, $y_{n+1} \to 0$, which is the true equilibrium solution.

Implicit integration methods for sets of linear equations

For sets of linear equations

$$\mathbf{Y}' = -\mathbf{C} \cdot \mathbf{Y}$$

where **C** is a positive definite matrix. Explicit differencing gives

$$\mathbf{Y}_{n+1} = (\mathbf{1} - \mathbf{C}h) \cdot \mathbf{Y}_n.$$

Now a matrix $\mathbf{A}^n$ tends to zero as $n \to \infty$ only if the largest eigenvalue of **A** has magnitude less than unity. Thus $\mathbf{Y}_n$ is bounded as $n \to \infty$ only if the largest eigenvalue of $\mathbf{1} - \mathbf{C}h$ is less than 1, or

$$h < \frac{2}{\lambda_{max}}$$

where λ_{max} is the largest eigenvalue of **C**. Implicit differencing gives

$$\mathbf{Y}_{n+1} = \mathbf{Y}_n + h\mathbf{Y}'_{n+1}, \quad \text{or} \quad \mathbf{Y}_{n+1} = (\mathbf{1} + \mathbf{C}h)^{-1} \cdot \mathbf{Y}_n$$

Thus the method is stable for all stepsizes h. The penalty we pay for this stability is that we are required to invert a matrix at each step.

Semi-implicit integration methods

For a nonlinear system

$$\mathbf{Y}' = \mathbf{F}(\mathbf{Y}),$$

where **F** is a nonlinear operator, and the implicit differencing gives

$$\mathbf{Y}_{n+1} = \mathbf{Y}_n + h\mathbf{F}(\mathbf{Y}_{n+1})$$

In general this is some nasty set of nonlinear equations that hs to be solved iteratively at each step.
Suppose we try linearizing the equations, as in Newton's method:

$$\mathbf{Y}_{n+1} = \mathbf{Y}_n + h[\mathbf{F}(\mathbf{Y}_n) + \frac{\partial \mathbf{F}}{\partial \mathbf{Y}}|_{\mathbf{Y}_n} \cdot (\mathbf{Y}_{n+1} - \mathbf{Y}_n)]$$

where $\partial\mathbf{F}/\partial\mathbf{Y}$ is the Jacobian matrix. Then

$$\mathbf{Y}_{n+1} = \mathbf{Y}_n + h[\mathbf{1} - h\frac{\partial \mathbf{F}}{\partial \mathbf{Y}}]^{-1} \cdot \mathbf{F}(\mathbf{Y}_n)$$

Solving implicit methods by Linearization is called a "semi-implicit" method.

Predictor-corrector method

For the ODE, $y' = f(x, y)$, from x_n to x,

$$y(x) = y_n + \int_{x_n}^{x} f(x_1, y)\mathrm{d}x_1$$

- In a single step method, like Runge-Kutta, the value y_{n+1} at x_{n+1} depends only on y_n.
- In a multistep method, we approximate $f(x, y)$ by a polynomial passing through *several* previous points x_n, x_{n-1}, ... and possibly also through x_{n+1}. The result is

$$y_{n+1} = y_n + h[\beta_0 f(x_{n+1}, y_{n+1}) + \beta_1 f(x_n, y_n) + \beta_2 f(x_{n-1}, y_{n-1}) + \cdots]$$

  If $\beta_0 = 0$, the method is explicit; otherwise it is implicit.

- Predictor step: for y_{n+1}, two methods suggest, *functional iteration* and *Newton's method*.
  1. take some initial guess for y_{n+1},
  2. insert it into the right-hand side to get an updated value of y_{n+1} until converges.
  3. use some *explicit* formula for the initial guess.

Adams-Bashforth-Moulton schemes

- Corrector step:
  1. In the predictor step we are essentially *extrapolating* the polynomial fit to the derivative from the previous points to the new point x_{n+1}.
  2. Then we doing the integral in a Simpson-like manner from x_n to x_{n+1}.

$$\int_{x_{k-1}}^{x_{k+1}} f(x)\mathrm{d}x \approx \frac{\Delta x}{3}[f(x_{k-1}) + 4f(x_k) + f(x_{k+1})] + \mathbf{O}(\Delta x^5),$$

  3. The subsequent Simpson-like integration, using the prediction step's value of y_{n+1} to *interpolate* the derivative, is called the corrector step.

- Adams-Bashforth-Moulton schemes
  1. Predictor:

$$y_{n+1} = y_n + \frac{h}{12}[23f(x_n, y_n) - 16f(x_{n-1}, y_{n-1}) + f(x_{n-2}, y_{n-2}] + \mathbf{O}(h^4)$$

  2. Corrector:

$$y_{n+1} = y_n + \frac{h}{12}[5f(x_{n+1}, y_{n+1}) + 8f(x_n, y_n) - f(x_{n-1}, y_{n-1})] + \mathbf{O}(h^4)$$

Boundary value problem

For the differential equations:

$$\frac{\mathrm{d}y_i(x)}{\mathrm{d}x} = g_i(x, y_1, y_2, \ldots, y_N), \quad i = 1, 2, \ldots, N$$

At x_1, the solutions is supposed to satisfy

$$B_{1j}(x_1, y1, y_2, \ldots, y_N) = 0, \quad j = 1, \ldots, n_1,$$

while at x_2, it is supposed to satisfy

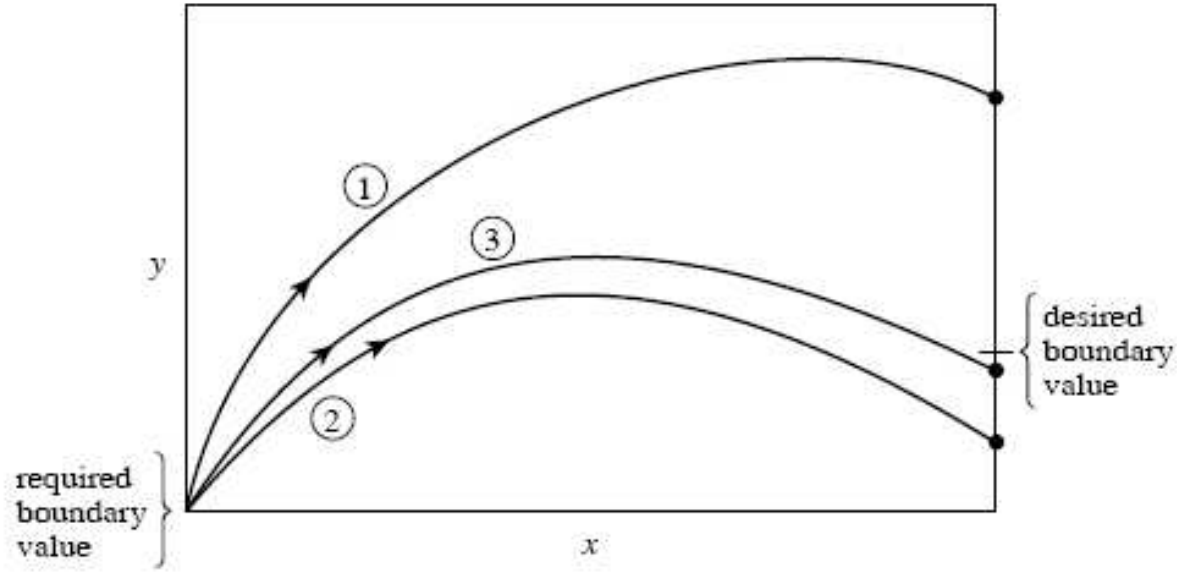
$$B_{2k}(x_2, y1, y_2, \ldots, y_N) = 0, \quad k = 1, \ldots, n_2,$$

n_1 boundary conditions at the starting point x_1, and a remaining set of $n_2 = N - n_1$ boundary conditions at the final point x_2.

Boundary value problem: shooting method

For there are $n_2 = N - n_1$ *freely specifiable* starting values. Let us imagine that these freely specifiable values are the components of a vector **V** that lives in a vector space of dimension n_2.

- Given a particular **V**, a particular $\mathbf{y}(x_1)$ is thus generated.
- A $\mathbf{y}(x_2)$ can be integrated by the ODEs to x_2 as an initial value problem.
- At x_2, let us define a *discrepancy vector* **F**, also of dimension n_2, whose component measure how far we are from satisfying the n_2 boundary conditions at x_2.



Boundary value problem: shooting method

- At x_1, a particular initial condition is given

$$y_i(x_1) = y_i(x_1; V_1, \ldots, V_{n2}), \quad i = 1, \ldots, N$$

- The boundary condition at x_2,

$$B_{2k}(x_2, y1, y_2, \ldots, y_N) = 0, \quad k = 1, \ldots, n_2,$$

becomes

$$B_{2k}(x_2, \mathbf{y}) = \mathbf{F}_k, \quad k = 1, \ldots, n_2.$$

- The problem becomes to find the root of the equation,

$$B_{2k}(x_2, \mathbf{V}) = 0, \quad k = 1, \ldots, n_2.$$

for **V**.

Newton-Raphson method

- Newton-Raphson mouthed for root finding,

$$\text{Given} \quad F_i(x_1, x_2, \ldots, x_N) = 0 \quad i = 1, 2, \ldots, N$$

$$\mathbf{F}(\mathbf{x} + \delta\mathbf{x}) = \mathbf{F}(\mathbf{x}) + \mathbf{J} \cdot \delta\mathbf{x} + \mathbf{O}(\delta\mathbf{x}^2) = 0,$$

where the Jacobian matrix **J**:

$$J_{ij} \equiv \frac{\partial F_i}{\partial x_j}.$$

and the new solution $\mathbf{x}_{new} = \mathbf{x}_{old} + \delta\mathbf{x}$, with

$$\mathbf{J} \cdot \delta\mathbf{x} = -\mathbf{F}.$$

- For the boundary condition at x_2, we have a correction term for **V**

$$\mathbf{J} \cdot \delta\mathbf{V} = -\mathbf{F}.$$

- Then adding the correction back until it converges,

$$\mathbf{V}^{new} = \mathbf{V}^{old} + \delta\mathbf{V}.$$

Boundary value problem: finite difference method

Second-order FD approximation for $u'(x_j)$

$$u'(x_j) \approx \frac{u_{j+1} - u_{j-1}}{2h}$$

in the matrix-vector form (with periodic boundary)

$$\begin{pmatrix} u'_1 \\ \vdots \\ u'_j \\ \vdots \\ u'_N \end{pmatrix} = h^{-1} \begin{pmatrix} 0 & \frac{1}{2} & & & -\frac{1}{2} \\ -\frac{1}{2} & 0 & & & \\ & & \ddots & & \\ & -\frac{1}{2} & 0 & \frac{1}{2} & \\ & & \ddots & & \\ & & & 0 & \frac{1}{2} \\ \frac{1}{2} & & & -\frac{1}{2} & 0 \end{pmatrix} \begin{pmatrix} u_1 \\ \vdots \\ u_j \\ \vdots \\ u_N \end{pmatrix}$$

Finite difference approximation

$$\frac{\mathrm{d}u(x)}{\mathrm{d}x} = x,$$

Second-order FD approximation for $u'(x_j)$

$$u'(x_j) \approx \frac{u_{j+1} - u_{j-1}}{2h}$$

in the matrix-vector form (with periodic boundary)

$$\begin{pmatrix} u'_1 \\ \vdots \\ u'_j \\ \vdots \\ u'_N \end{pmatrix} = h^{-1} \begin{pmatrix} 0 & \frac{1}{2} & & & & -\frac{1}{2} \\ -\frac{1}{2} & 0 & & & & \\ & & \ddots & & & \\ & & -\frac{1}{2} & 0 & \frac{1}{2} & \\ & & & \ddots & & \\ & & & & 0 & \frac{1}{2} \\ \frac{1}{2} & & & & -\frac{1}{2} & 0 \end{pmatrix} \begin{pmatrix} u_1 \\ \vdots \\ u_j \\ \vdots \\ u_N \end{pmatrix} = \begin{pmatrix} x_1 \\ \vdots \\ x_j \\ \vdots \\ x_N \end{pmatrix}$$

Jacobi iteration

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{b}, \quad \begin{bmatrix} 3 & 2 \\ 1 & 2 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix},$$

Jacobi's method for iteration

$$\begin{bmatrix} x_{1,k+1} \\ x_{2,k+1} \end{bmatrix} = \begin{bmatrix} 0 & -2/3 \\ -1/2 & 0 \end{bmatrix} \cdot \begin{bmatrix} x_{1,k} \\ x_{2,k} \end{bmatrix} + \begin{bmatrix} 1/3 \\ -1/2 \end{bmatrix},$$

$$\mathbf{x}_{k+1} = \tilde{\mathbf{A}} \cdot \mathbf{x}_k + \tilde{\mathbf{b}},$$

Gauss-Seidel iteration

Gauss-seidel iteration:

$$\begin{aligned} x_{1,k+1} &= -\frac{2}{3}x_{2,k} + \frac{1}{3}, \\ x_{2,k+1} &= -\frac{1}{2}x_{1,k+1} - \frac{1}{2}. \end{aligned}$$

for $m = 1, 2, \ldots, N$,

$$\begin{aligned} x_m^{(k+1)} &= -\sum_{n=1}^{m-1} \frac{a_{mn}}{a_{mm}} x_n^{(k+1)} - \sum_{n=m+1}^{N} \frac{a_{mn}}{a_{mm}} x_n^{(k)} + \frac{b_m}{a_{mm}}, \\ &= \frac{b_m - \sum_{n=1}^{m-1} a_{mn} x_n^{(k+1)} - \sum_{n=m+1}^{N} a_{mn} x_n^{(k)}}{a_{mm}} \end{aligned}$$

converge more fast!

Relaxation technique

relaxation technique:

$$x_m^{(k+1)} = (1-\omega)x_m^{(k)} + \omega \frac{b_m - \sum_{n=1}^{m-1} a_{mn} x_n^{(k+1)} - \sum_{n=m+1}^{N} a_{mn} x_n^{(k)}}{a_{mm}}$$

with $0 < \omega < 2$

- $1 < \omega < 2$: SOR, Successive Over-Relaxation,
- $0 < \omega < 1$: successive under-relaxation.

Relaxation method