Numerical Solution of Ordinary Differential Equations

Most fundamental laws of Science are based on models that explain variations in physical properties and states of systems described by **differential equations**.

Several examples of laws appear in C&C PT 7.1 and are applied in Ch. 28 --

- Newton's 2nd law: $\dfrac{dv}{dt} = \dfrac{F}{m}$ velocity(v), force (F), and mass (m)

- Fourier's heat law: $q = -k'\dfrac{dT}{dx}$ heat flux (q), temperature (T), and thermal conductivity (k ')

- Fick's diffusion law $j = -D\dfrac{dc}{dx}$ mass flux (j), concentration (c) and diffusion coefficient (D)

- Faraday's law: $\Delta V_i = L\dfrac{di}{dt}$ voltage drop (ΔV), inductance (L) and current (i)

ODE's Ordinary Differential Equations

Only **one** independent variable, i.e., **x** as in y(x):

$$\frac{d^2y}{dx^2} + \omega^2 y = g(x) \qquad \text{1-dimensional problem in space x}$$

$$\frac{dy}{dt} = f(y,t) \qquad \text{time-dynamics problem}$$

PDE's Partial Differential Equations

More than one independent variable, i.e., **x** and **y** as in T(x,y):

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0 \qquad \text{Laplace Equation}$$

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2} = 0 \qquad \text{Wave Equation } u = u(x,t)$$

Auxiliary Conditions

Because we are integrating an indefinite integral, we need additional information to obtain a unique solution. Note that an **n^{th}** order equations generally requires **n** auxiliary conditions.

- **Initial Value Problem (IV)**

  Information at a single value of the independent variable, typically at the beginning of the interval:

$$y(0) = y_o$$

- **Boundary Value Problem (BV)**

  Information at more that one value of the independent variable, typically at both ends of the interval:

$$y(0) = y_o \ \text{ and } \ y(1) = y_f$$

I. Single-Step Methods for IV Problems (C&C Ch. 25)
   a. Euler
   b. Heun and Midpoint/Improved Polygon
   c. General Runge-Kutta
   d. Adaptive step-size control

II. Stiff ODEs (C&C 26.1)

III. Multi-Step Methods for IV Problems (C&C 26.2)
   a. Non-Self-Starting Heun
   b. Newton-Cotes
   c. Adams

IV. Boundary Value (BV) Problems (C&C Ch. 27.1 & 27.2)
   a. Solution Methods
      1. Shooting Method
      2. Finite Difference Method
   b. Eigenvalue Problems $\{ [A] - \lambda [I] \} \{x\} = \{0\}$

One-Step Methods for IV Problems

Consider the generic first-order initial value (IV) ODE problem:

$$\frac{dy}{dx} = f(x, y)$$ where (x_o, y_o) are given

and we wish to find $y = y(x)$

The independent variable **x**

may be space x,

or time t.

ODE's: One-step methods

We can solve higher-order IV ODE's
by transforming to a set of 1st-order ODE's,

$$\frac{d^2 y}{dx^2} + \frac{dy}{dx} + 5y = 0$$

Let $z = \dfrac{dy}{dx}$ & substitute $\rightarrow$ $\dfrac{dz}{dx} + z + 5y = 0$

Now solve a SYSTEM of two linear, first order ordinary differential equations:

$$\frac{dy}{dx} = z \qquad \text{and} \qquad \frac{dz}{dx} = -z - 5y$$

ODE's: first order IV problem - One-step methods

$$\frac{dy}{dx} = f(x, y)$$

where (x_o, y_o) are given and we wish to find $y = y(x)$.

The basic approach to numerical solution is stepwise:

Start with $(x_o, y_o) \Rightarrow (x_1, y_1) \Rightarrow (x_2, y_2) \Rightarrow$ etc.

Next Value = Previous Value + *slope* × step size

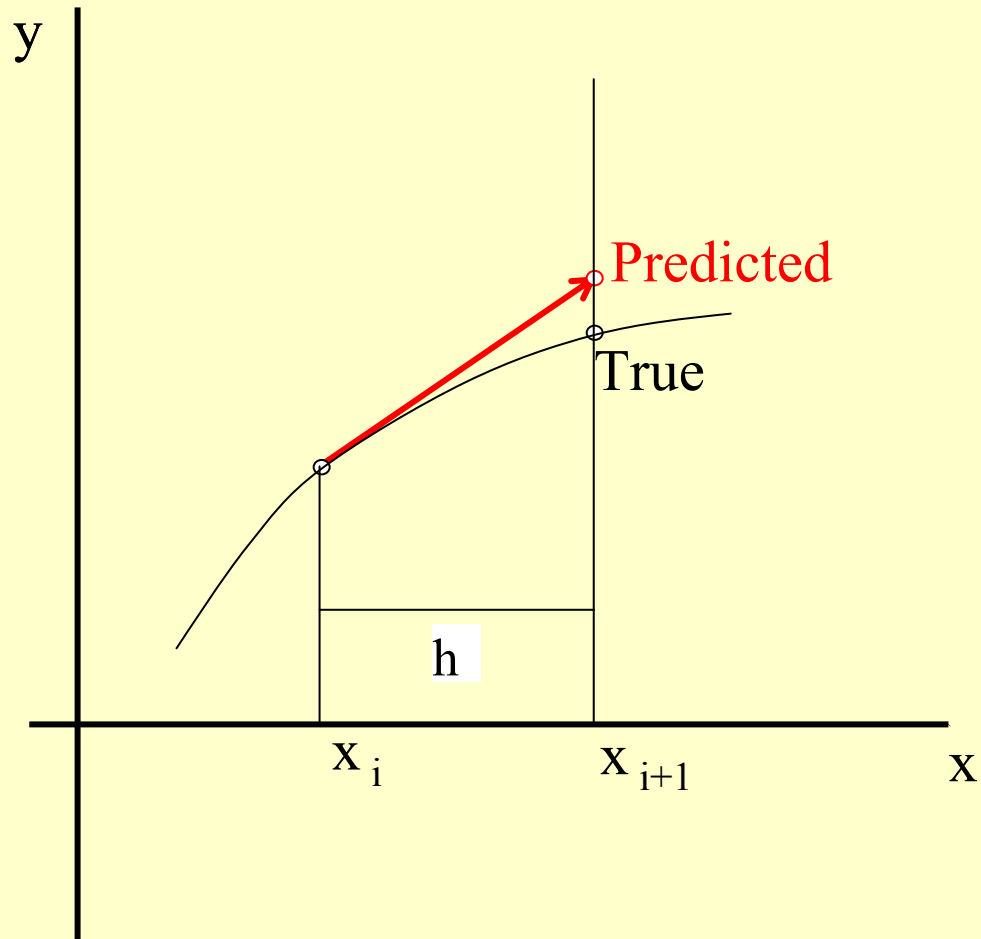$$\mathbf{y_{i+1}} = \mathbf{y_i} + \boldsymbol{\phi_i} \times \mathbf{h}$$

$$h = x_{i+1} - x_i = \text{step size}$$

Key to the various one-step methods is how the *slope* is obtained. This *slope* represents a weighted average of the slope over the entire interval and may <u>not</u> be the tangent at (x_i, y_i)

Using an estimate of the slope to guess $f(x_{i+1})$

Euler's Method

Given (x_i, y_i), need to determine (x_{i+1}, y_{i+1})

$$x_{i+1} = x_i + h \qquad\qquad y_{i+1} = y_i + \phi_i h$$

Estimate the *slope* as $\phi_i = f(x_i, y_i)$

$$y_{i+1} = y_i + f(x_i, y_i) h$$

The slope at the **beginning** of the step is applied across the entire interval

Analysis Local Error for the Euler Method

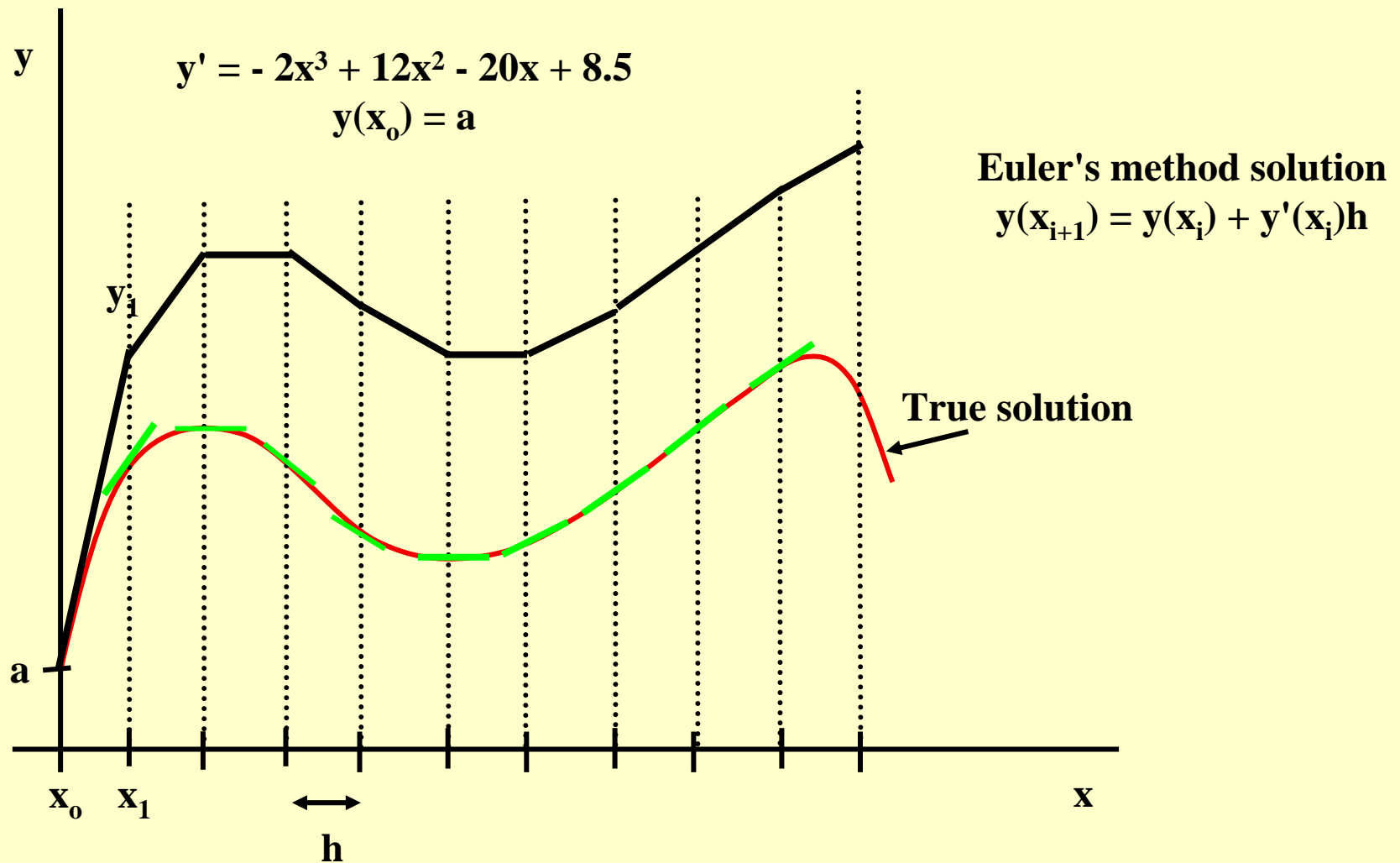Taylor series of true solution: $\quad y_{i+1} = y_i + y_i' h + y_i'' h^2/2 + \dots$

$$\text{where } y_i' = f(x_i, y_i) = f_i$$

Euler rule: $\qquad \hat{y}_{i+1} = y_i + f_i h$

Truncation error:
(local)
$$E_a = y_{i+1} - \hat{y}_{i+1} = y_i'' \frac{h^2}{2} = O(h^2)$$

Global Error: $\quad E_a = G(n,h)\, O(h^2) = O(h)$

Euler's Method: Graphic example

$$y' = -2x^3 + 12x^2 - 20x + 8.5$$
$$y(x_o) = a$$

Euler's method solution
$$y(x_{i+1}) = y(x_i) + y'(x_i)h$$

True solution

y

y_i

a

x_o x_1

h

x

Matlab demo

Error Analysis

1. Truncation Error (Truncating Taylor Series)
   [large step size ➔ large errors]
2. Rounding Error (Machine precision)
   [very small step size ➔ roundoff errors]

Two kinds of Truncation Error:

Local – error within one step due to application of the numerical method

Propagation – error due to previous local errors.

Global Truncation Error = Local + Propagation

Generally if the local truncation error is $O(h^{n+1})$ then, as with numerical quadrature formulas, the global truncation error is $O(h^n)$. (Proof is more difficult.)

General Error Notes:

1. For stable systems, error is reduced by decreasing h
2. If method is $O(h^n)$ globally, then it is exact for $(n-1)^{th}$ order polynomial in x

Improved One-Step Methods
Heun's Method (simple predictor-corrector)

Predict:

$$y_{i+1}{}^{o} = y_i + f(x_{i,}y_i)h$$

Estimate the avg. slope as:

$$\overline{f} = \frac{f(x_i,y_i) + f(x_{i+1},y_{i+1}{}^{o})}{2}$$

Correct:

$$x_{i+1} = x_i + h \qquad y_{i+1} = y_i + \overline{f}h$$

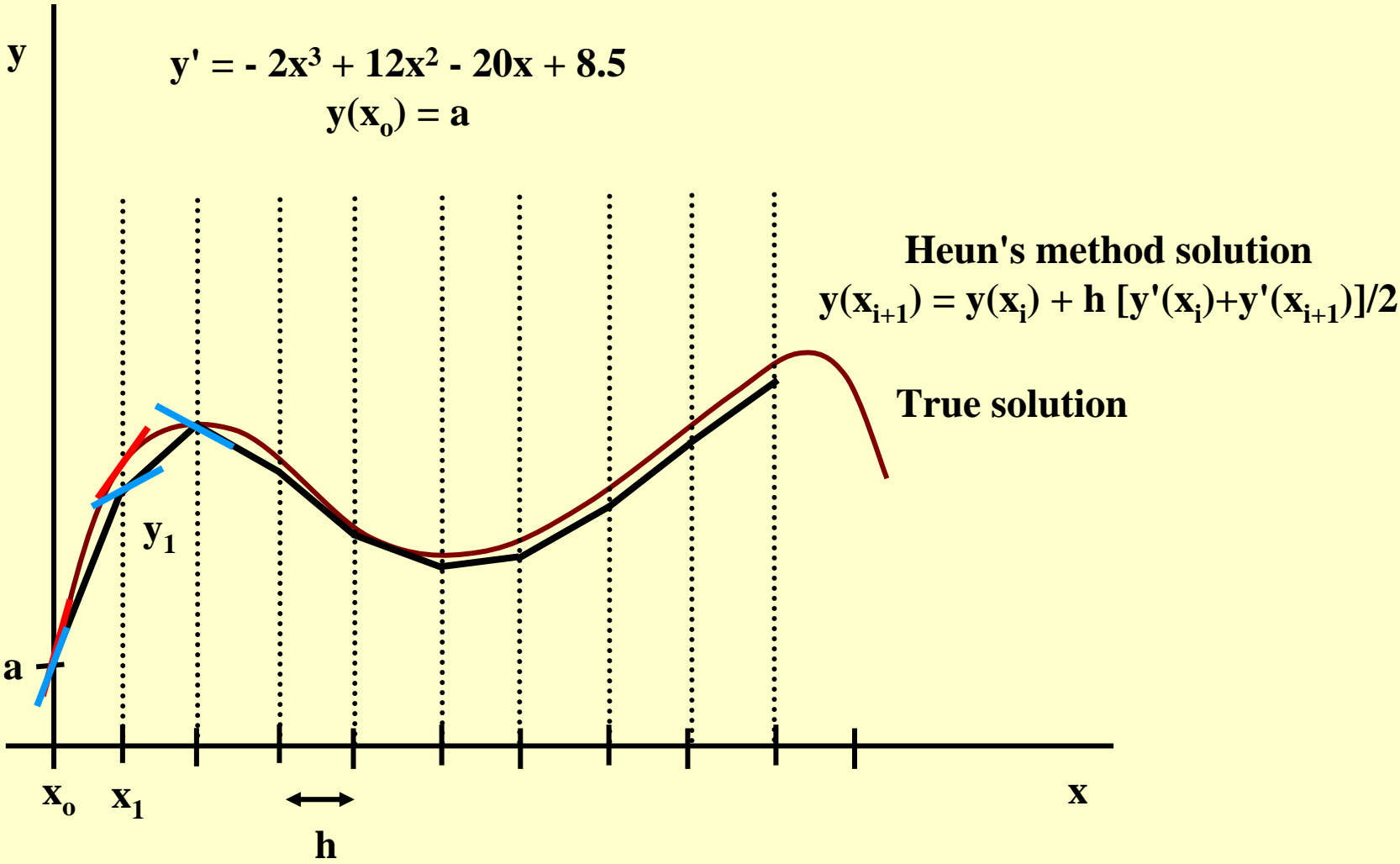**Notes:** 1. Local Error $O\,(h^3)$ and Global Error $O\,(h^2)$

2. Corrector step can be iterated using ε_a as stop criterion

3. If derivative is only f(x) and not f(x,y), then the predictor has no effect and

$$y_{i+1} = y_i + \frac{f(x_i) + f(x_{i+1})}{2}h$$

which is the Trapezoid Rule.

$$y' = - 2x^3 + 12x^2 - 20x + 8.5$$
$$y(x_o) = a$$

Heun's method solution
$$y(x_{i+1}) = y(x_i) + h \, [y'(x_i)+y'(x_{i+1})]/2$$

True solution

y

y_1

a

x_o x_1

h

x

Matlab demo

Midpoint Method *(Improved Polygon, Modified Euler)*

- Predict $y_{i+1/2}$ with a half step:

$$y_{i+1/2} = y_i + f(x_i, y_i)\frac{h}{2}$$

- Estimate the slope:

$$slope = f(x_{i+1/2}, y_{i+1/2}) \quad \text{with} \quad x_{i+1/2} = x_i + \frac{h}{2}$$

- Correct y_{i+1}: $\quad y_{i+1} = y_i + f(x_{i+1/2}, y_{i+1/2})\, h$

Note: Local Error $O(h^3)$ and Global Error $O(h^2)$
Error same order as Heun.

Matlab demo

General Runge-Kutta (RK) Methods

Employing single-step

$$y_{i+1} = y_i + \phi(x_i, y_i, h)\, h$$

Increment function or slope, $\phi(x_i, y_i, h)$, is a weighted average:

$$\phi = a_1 k_{1i} + a_2 k_{2i} + \ldots + a_j k_{ji} + \ldots + a_n k_{ni} \quad (n^{th} \text{ order method})$$

where: a_j = weighting factors (that sum to unity)

         k_{ji} = slope at point x_{ji} such that $x_i < x_{ji} < x_{i+1}$

$$k_{1i} = f(x_i, y_i)$$
$$k_{2i} = f(x_i + p_1 h, \; y_i + q_{11} k_{1i} h)$$
$$k_{3i} = f(x_i + p_2 h, \; y_i + q_{21} k_{1i} h + q_{22} k_{2i} h)$$

$$\vdots$$

$$k_{ni} = f(x_i + p_{n-1} h, \; y_i + q_{n-1,1} k_{1i} h + q_{n-1,2} k_{2i} h + \ldots + q_{n-1,n-1} k_{n-1,i} h)$$

Runge-Kutta (RK) Methods

First-order Runge-Kutta Method: Euler's Method

$$y_{i+1} = y_i + k_1 h \qquad\qquad k_1 = f(x_i, y_i) \quad a_1 = 1$$

Has a global truncation error, $O(h)$

Second-order Runge-Kutta Methods:

Assume $a_2 = 1/2$; Heun's w/ Single Corrector: no iteration

$$y_{i+1} = y_i + \left(\tfrac{1}{2} k_1 + \tfrac{1}{2} k_2\right) h \qquad \begin{array}{l} k_1 = f(x_i, y_i) \\ k_2 = f(x_i + h, y_i + hk_1) \end{array}$$

Assume $a_2 = 1$; Midpoint (Improved Polygon)

$$y_{i+1} = y_i + k_2 h \qquad\qquad k_2 = f\left(x_i + \tfrac{1}{2} h, y_i + \tfrac{1}{2} hk_1\right) \quad \text{same } k_1$$

Assume $a_2 = 2/3$; Ralston's Method (minimum bound on E_t)

$$y_{i+1} = y_i + \left(\tfrac{1}{3} k_1 + \tfrac{2}{3} k_2\right) h \quad k_2 = f\left(x_i + \tfrac{3}{4} h, y_i + \tfrac{3}{4} hk_1\right) \quad \text{same } k_1$$

All have global truncation error, $O(h^2)$

Classical Fourth-order Runge-Kutta Method

Classical Fourth-order Runge-Kutta Method

Employing single-step:

$$y_{i+1} = y_i + (1/6) [k_1 + 2k_2 + 2k_3 + k_4] h$$

where

$k_1 = f(x_i , y_i)$ Euler step

$k_2 = f(x_i + h/2 , y_i + k_1h/2)$ Midpoint

$k_3 = f(x_i + h/2 , y_i + k_2h/2)$ Better midpoint

$k_4 = f(x_i + h , y_i + k_3h)$ Full step

1. Global truncation error, $O(h^4)$
2. If the derivative, f, is a function of only x, this reduces to Simpson's 1/3 Rule.

$$y_{i+1} = y_i + h (k_1 + 2k_2 + 2k_3 + k_4)/6 \quad \text{where } k_2 = k_3$$

Matlab demo

Examples of Frequently Encountered, Simple ODE's

1. **Exponential growth**
   unconstrained growth of biological organisms,
   positive feedback electrical systems, and
   chemical reactions generating their own catalyst)

$$\frac{dy}{dt} = \lambda \, y \quad \text{with solution} \quad y(t) = y_0 \, e^{\lambda t}$$

2. **Exponential decay**
   discharge of a capacitor,
   decomposition of material in a river,
   wash-out of chemicals in a reactor, and
   radioactive decay

$$\frac{dy}{dt} = -\lambda \, y \quad \text{with solution} \quad y(t) = y_0 \, e^{-\lambda t}$$

Classical Fourth-order Runge-Kutta Method -- Example

Numerical Solution of the simple differential equation

$y' = +2.77259\, y$ with $y(0) = 1.00$; Solution is $y = \exp(+2.773\, x) = 16^x$

Step sizes vary so that all methods use the same number of
functions evaluations to progress from $x = 0$ to $x = 1$.

x	Exact Solution	Euler	Heun w/o iter	4th-order Runge-Kutta	h * ki for R-K
0.000	1.000	1.000	1.000	1.000	
0.125	1.414	1.347			1.386
0.250	2.000	1.813	1.933		2.347
0.375	2.828	2.442			3.013
0.500	4.000	3.288	3.738	3.945	5.564
0.625	5.657	4.427			5.469
0.750	8.000	5.962	7.227		9.260
0.875	11.31	8.028			11.89
1.000	16.00	10.81	13.97	15.56	21.95
h =		0.125	0.25	0.5	

Classical Fourth-order Runge-Kutta Method – Example (cont.)

x	Exact Solution	Euler	Heun w/o iter	4th-order Runge-Kutta	h * ki for R-K
0.0000	1.0000	1.0000	1.0000		
0.0625	1.1892	1.1733			0.6931
0.1250	1.4142	1.3766	1.4066		0.9334
0.1875	1.6818	1.6151			1.0166
0.2500	2.0000	1.8950	1.9786	1.9985	1.3978
0.3125	2.3784	2.2234			1.3853
0.3750	2.8284	2.6087	2.7832		1.8653
0.4375	3.3636	3.0608			2.0317
0.5000	4.0000	3.5911	3.9149	3.9940	2.7935
0.5625	4.7568	4.2134			2.7684
0.6250	5.6569	4.9436	5.5068		3.7279
0.6875	6.7272	5.8002			4.0604
0.7500	8.0000	6.8053	7.7460	7.9820	5.5829
0.8125	9.5137	7.9846			5.5327
0.8750	11.314	9.3683	10.8958		7.4502
0.9375	13.454	10.992			8.1147
1.0000	16.000	12.896	15.326	15.952	11.157
h =		**0.0625**	**0.125**	**0.25**	Matlab demo

Classical Fourth-order Runge-Kutta Method

In each case, all three RK methods used the same number of function evaluations to move from 0.00 to 1.00.

Which was able to provide the more accurate estimate of y(1)?

Higher-Order ODEs and Systems of Equations (C&C 25.4, p.711)

- An nth order ODE can be converted into a system of n, coupled 1st-order ODEs.
- Systems of first order ODEs are solved just as one solves a single ODE.

Consider the 4th-order ODE:

$$f(x) = y'''' + a(x)\, y''' + b(x)\, y'' + c(x)\, y' + d(x)\, y$$

Let: $y''' = v_3$; $y'' = v_2$; and $y' = v_1$

Write this 4th order ODE as a system of four coupled 1st order ODEs:

$$\frac{d}{dx}\begin{pmatrix} y \\ v_1 \\ v_2 \\ v_3 \end{pmatrix} = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ f(x) - a(x)v_3 - b(x)v_2 - c(x)v_1 - d(x)y \end{pmatrix}$$

$$
\begin{pmatrix}
dy/dx \\
dv_1/dx \\
dv_2/dx \\
dv_3/dx
\end{pmatrix}
=
\begin{pmatrix}
v_1 \\
v_2 \\
v_3 \\
f(x) - a(x)v_3 - b(x)v_2 - c(x)v_1 - d(x)y
\end{pmatrix}
$$

Given the initial conditions @ $x = 0$:

$$y(0); \quad v_1 = y'(0); \quad v_2 = y''(0); \quad v_3 = y'''(0),$$

a numerical scheme can be used to integrate this system forward in

   time.

Example: Harmonically Driven Oscillator

Example: Single-degree-of-freedom, undamped, harmonically driven oscillator (see also C&C 28.4, p. 797)

ODE is 2nd order:

$$m\frac{d^2x}{dt^2} + kx = P(t) \qquad \text{or} \qquad \frac{d^2x}{dt^2} + \omega^2 x = \frac{P(t)}{m}$$

m = mass

k = spring constant

$$\omega = \sqrt{\frac{k}{m}} = \text{circular frequency}$$

P(t) = forcing function = P sin(γt)

$$T = \frac{2\pi}{\omega} = \text{natural period}$$

i.e., harmonically driven

Example: Harmonically Driven Oscillator

$$\frac{d^2x}{dt^2} + \omega^2 x = \frac{P(t)}{m}$$

Initial conditions: $\quad x(0) = 0 \quad$ and $\quad \frac{dx}{dt}(0) = 0$

Analytical solution:

$$x(t) = \frac{-P\gamma}{\omega m(\omega^2 - \gamma^2)}\sin(\omega t) + \frac{P}{m(\omega^2 - \gamma^2)}\sin(\gamma t)$$

$$= \text{(homogeneous soln.)} + \text{(particular soln.)}$$

Example: Harmonically Driven Oscillator

$$\frac{d^2 x}{dt^2} + \omega^2 x = \frac{P(t)}{m}$$

Recast the 2nd-order ODE as two 1st-order ODEs):

$$\frac{dx}{dt} = v = f(t, x, v) \qquad \text{with} \ \ x(0) \ = \ 0$$

$$\frac{dv}{dt} = \frac{P \sin(\gamma t)}{m} - \omega^2 x = g(t, x, v) \qquad \text{with} \ \ v(0) \ = \ 0$$

Example: Harmonically Driven Oscillator – Euler Method

$$\frac{d^2x}{dt^2} + \omega^2 x = \frac{P(t)}{m}$$

$$\frac{dx}{dt} = v = f(t, x, v)$$

$$\frac{dv}{dt} = \frac{P\sin(\gamma t)}{m} - \omega^2 x = g(t, x, v)$$

We can solve these two 1st-order ODE's sequentially.

Let m = 0.25, k = $4\pi^2$, P = 20, γ = 8,

Thus g = 80 sin(8t) - $16\pi^2$x.   Solve from t = 0 until t = 1.

For example, by the Euler method, with h = 0.01:

t	x	v
$t_0 = 0$	$x_0 = 0$	$v_0 = 0$
$t_1 = t_0 + h = 0.01$	$x_1 = x_0 + f(t_0, x_0, v_0)h = 0$	$v_1 = v_0 + g(t_0, x_0, v_0)h = 6.39$
$t_2 = t_1 + h = 0.02$	$x_2 = x_1 + f(t_1, x_1, v_1)h = 0.0639$	$v_2 = v_1 + g(t_1, x_1, v_1)h = 12.75$
...	...	...
$t_n = t_{n-1} + h$	$x_n = x_{n-1} + f(t_{n-1}, x_{n-1}, v_{n-1})h$	$v_n = v_{n-1} + g(t_{n-1}, x_{n-1}, v_{n-1})h$ Matlab demo

Adaptive Step-size Control (C&C 25.5, p. 710)

Goal: with little additional effort estimate (bound) magnitude of local truncation error at each step so that step size can be reduced/increased if local error increases/decreases.

  1. Repeat analysis at each time step with step length h and h/2. Compare results to estimate local error.   (C&C 25.5.1) Use Richardson extrapolation to obtain higher order result.

  2. Use a matched pair of Runge-Kutta formulas of order r and r+1 which use common values of k_i, and yield estimate or bound local truncation error.

C&C 25.5.2 discuss 4th-5th-order Runge-Kutta-Fehlberg pair.

A $2^{nd} - 3^{rd}$ order RK pair yielding local truncation error estimate:

2nd-order Midpoint Method $O(h^2)$:

$$k_1 = f(x_i, y_i); \qquad\qquad k_2 = f(x_i + \tfrac{1}{2}h, \ y_i + \tfrac{1}{2}h_{k1})$$
$$y_{i+1} = y_i + k_2 \, h$$

Third-order RK due to Kutta $O(h^3)$ (C&C 25.3.2)

$$k_3 = f(x_i + h, \ y_i + h(2k_2 - k_1) \,)$$
$$y_{i+1}{}^* = y_i + h(k_1 + 4k_2 + k_3)$$

Estimate of truncation error for midpoint formula is

$$E_i = y_{i+1} - y_{i+1}{}^* = -\, h(k_1 - 2k_2 + k_3)$$

This is a central difference estimate of (const.) $h^3 \, y'''$
which describes the local truncation error for midpoint method.

Use more accurate values $y_{i+1}{}^*$, but E_i provides estimate of the local
error that can be used for step-size control.

Stiff Differential Equations (C&C 26.1, p. 719)

A *stiff* system of ODE's is one involving rapidly changing components together with slowly changing ones. In many cases, the rapidly varying components die away quickly, after which the solution is dominated by the slow ones.

Even simple first-order ODE's can be stiff. C&C gives the example:

$$\frac{dy}{dt} = -1000y + 3000 - 2000e^{-t}$$

With initial conditions $y(0) = 0$ and with solution

$$y(t) = 3 - 0.998e^{-1000t} - 2.002e^{-t}$$

<div align="center">Fast term   Slow term</div>

Stiff Differential Equations

In the solution

$$y(t) = 3 - 0.998e^{-1000t} - 2.002e^{-t}$$

the middle term damps out very quickly and after it does, the slow

solution closely follows the path:

$$y(t) \approx 3 - 2e^{-t}$$

We would need a very small time step $h = \Delta t$ to capture the

behavior of the rapid transient and to preserve a stable and accurate

solution, and this would then make it very laborious to compute the

slowly evolving solution.

Matlab demo

Stiff Differential Equations

A **stability analysis** of this equation provides important insights. For a solution to be *stable* means that errors at any stage of computation are not amplified but are attenuated as computations proceed. To analyze for stability, we consider the homogeneous part of the ODE

$$\frac{dy}{dt} = -ay \qquad \text{with solution} \qquad y(t) = y_0 e^{-at}$$

Euler's method yields

$$y_{i+1} = y_i + \frac{dy_i}{dt} h = y_i - ay_i h = y_i(1 - ah)$$

Assume some small error exists in the initial condition y_0 or in an early stage of the solution. Then we see that after n steps,

$$y_n = y_0(1 - ah)^n \qquad \text{Bounded iff } |(1\text{-}ah)| < 1, \text{ i.e., } h < 2/a$$

Matlab demo

Stiff Differential Equations

Euler's method is known as an ***explicit*** method because the derivative is taken at the known point i. An alternative is to use an ***implicit*** approach, in which the derivative is evaluated at a future step $i+1$. The simplest method of this type is the backward Euler method that yields

$$y_{i+1} = y_i + \frac{dy_{i+1}}{dt} h = y_i - ay_{i+1}h \implies y_{i+1} = \frac{y_i}{1+ah}$$

Because $1/(1+ah)$ will remain bounded for any (positive) value of h, this method is said to be *unconditionally stable*.

Implicit methods always entail more effort than explicit methods, especially for nonlinear equations or sets of ODE's, so the stability is gained at a price. Moreover, accuracy still governs the step size.

Systems of Stiff Differential Equations

What to do?

Need to be aware of the potential for encountering stiff ODE's.

Use special codes for solving stiff differential equations which are generally implicit multistep methods.

For example, MATLAB has some methods specifically designed to solve stiff ODE's, e.g., ode23S.

Predictor-Corrector Multi-Step Methods for ODE's
   - Utilize valuable information at previous points.
   - For increased accuracy, use a predictor that has truncation of same order as corrector. In addition iterate corrector to minimize truncation error and improve stability.

Non-Self Starting (NSS) Heun Method

$O\,(h^3)$-**Predictor** $-\;\; y_{i+1}{}^o = \; y_{i-1}{}^m + f(x_i, y_i{}^m)\, 2h$

> Higher order predictor than the self-starting Heun.
> This is an <u>open</u> integration formula (midpoint).

$O\,(h^3)$-**Corrector** $-\;\; y_{i+1}{}^{j} = y_i{}^m + \dfrac{f(x_i, y_i{}^m) + f(x_{i+1}, y_{i+1}{}^{j-1})}{2} h$

Notes: 1. Corrector is applied iteratively for j=1, ..., m
   2. $y_i{}^m$ is fixed (from previous step iterations)
   3. This is a closed integration formula (Trapezoid)

Multi-step methods

Two general schemes solving ODE's:

1. *Newton-Cotes Formulas*

$$y_{i+1} = \begin{cases} y_{i-n} + h\displaystyle\sum_{k=0}^{n-1} c_k f_{i-k} + O(h^{n+1}) & \text{open predictor} \\[2em] y_{i-n+1} + h\displaystyle\sum_{k=0}^{n-1} \overline{c_k} f_{i-k+1} + O(h^{n+1}) & \text{closed corrector} \end{cases}$$

with c_k from Table 21.4, and $\overline{c}_k$ from Table 21.2 of C&C.

2. *Adams Formulas* (Generally more stable)

$$y_{i+1} = y_i + \begin{cases} h\displaystyle\sum_{k=0}^{n-1} \beta_k f_{i-k} + O(h^{n+1}) & \begin{array}{l}\text{open predictor} \\ \text{Adams - Bashforth}\end{array} \\[2em] h\displaystyle\sum_{k=0}^{n-1} \overline{\beta_k} f_{i-k+1} + O(h^{n+1}) & \begin{array}{l}\text{closed corrector} \\ \text{Adams - Moulton}\end{array} \end{cases}$$

with β_k and $\overline{\beta}_k$ determined from Tables 26.1 and 26.2, respectively.

Popular Multi-Step Methods

1. Heun non-self starting — Newton-Cotes with n=1

2. Milne's — Newton-Cotes with n=3 (but use Hamming's corrector for better stability properties)

3. 4th-Order Adams
   - *Predictor 4th-Order Adams-Bashforth*
   - *Corrector 4th-Order Adams-Moulton*

If predictor and corrector are of same order, we can obtain estimates of the truncation error during computation.

Improving Accuracy and Efficiency (use of modifiers)
  1. provide criterion for step size adjustment (Adaptive Methods)
  2. employ modifiers determined from error analysis

For Non-self-Starting (NSS) Heun:
- *Final Corrector Modifier*

$$E_c = -\frac{1}{5}\left(y_{i+1}^m - \hat{y}_{i+1}^0\right) \qquad y_{i+1}^m \leftarrow y_{i+1}^m - \frac{y_{i+1}^m - \hat{y}_{i+1}^0}{5}$$

- *Predictor Modifier (excluding 1ˢᵗ step)*

$$E_p = \frac{4}{5}\left(y_i^m - \hat{y}_i^0\right) \qquad \hat{y}_{i+1}^0 \leftarrow \hat{y}_{i+1}^0 + \frac{4}{5}\left(y_i^m - \hat{y}_i^0\right)$$

with $\hat{y}_i^0$ = unmodified $\hat{y}_{i+1}^0$ from previous step

$\quad\ \ y_i^m$ = unmodified y_{i+1}^m from previous step

General Predictor-Corrector Schemes

Errors cited are local errors. Global errors order h smaller.

Predictors:

Euler:
$$\hat{y}_{i+1} = y_i + hf(x_i, y_i) + O(h^2)$$

Midpoint (same as Non-Self-Starting Heun):
$$\hat{y}_{i+1} = y_{i-1} + 2hf(x_i, y_i) + O(h^3)$$

Adams-Bashforth 2nd-Order:
$$\hat{y}_{i+1} = y_i + {}^h/_2\{3f(x_i, y_i) - f(x_{i-1}, y_{i-1})\} + O(h^3)$$

Adams-Bashforth 4th-Order:
$$\hat{y}_{i+1} = y_i + {}^h/_{24}\{55f(x_i, y_i) - 59f(x_{i-1}, y_{i-1}) + 37f(x_{i-2}, y_{i-2}) - 9f(x_{i-3}, y_{i-3})\} + O(h^5)$$

Hamming (and Milne's Method):
$$\hat{y}_{i+1} = y_{i-3} + {}^{4h}/_3\{2f(x_i, y_i) - f(x_{i-1}, y_{i-1}) + f(x_{i-2}, y_{i-2})\} + O(h^5)$$

Correctors:

- refine value of y_{i+1} given a $\hat{y}_{i+1}$
- more stable numerically – no spurious solutions which go out of control

Adams-Moulton 2nd-Order *closed (Non-self starting Heun):*

$$y_{i+1} = y_i + {}^h/_2 \{ f(x_{i+1}, \hat{y}_{i+1}) + f(x_i, y_i) \} + O(h^3)$$

Adams-Moulton 4th-Order *closed (relatively stable):*

$$y_{i+1} = y_i + {}^h/_{24} \{ 9f(x_{i+1}, \hat{y}_{i+1}) - 19f(x_i, y_i) - 5f(x_{i-1}, y_{i-1}) + f(x_{i-2}, y_{i-2}) \} + O(h^5)$$

Milne's *(N.C. 3-point, Simpson's 1/3 Rule, <u>not always stable!!!</u>):*

$$y_{i+1} = y_{i-1} + {}^h/_3 \{ f(x_{i+1}, \hat{y}_{i+1}) + 4f(x_i, y_i) + f(x_{i-1}, y_{i-1}) \} + O(h^5)$$

Hamming *(modified Milne, stable):*

$$y_{i+1} = {}^9/_8 y_i - {}^1/_8 y_{i-2} + {}^{3h}/_8 \{ f(x_{i+1}, \hat{y}_{i+1}) + 2f(x_i, y_i) + f(x_{i-1}, y_{i-1}) \} + O(h^5)$$

Predictor-Corrector solution

Example: Undamped, harmonically driven oscillator

$$m\frac{d^2x}{dt^2} + kx = P(t)$$

m = mass

$$\omega = \sqrt{\frac{k}{m}} = \text{circular frequency}$$

k = spring constant

P(t) = forcing function $= P\sin(\gamma t)$ $T = \dfrac{2\pi}{\omega}$ = natural period

γ = driving frequency

Initial conditions: $x(0) = 0$ and $\dfrac{dx}{dt}(0) = 0$

Analytical solution:

$$x(t) = \frac{-P\gamma}{\omega m(\omega^2 - \gamma^2)}\sin(\omega t) + \frac{P}{m(\omega^2 - \gamma^2)}\sin(\gamma t)$$

Predictor-Corrector solution: **Undamped, harmonic oscillator**

$$m\frac{d^2x}{dt^2} + kx = P(t)$$

Recast 2nd-order ODE as two 1st-order ODE's:

$$\frac{dx}{dt} = v = f(t, x, v) \qquad\qquad \text{with } x(0) = 0$$

$$\frac{dv}{dt} = \frac{P\sin(\gamma t)}{m} - \omega^2 x = g(t, x, v) \qquad \text{with } v(0) = 0$$

Let m = 0.25, k = $4\pi^2$, P = 20, γ = 8,

thus: **g = 80 sin(8t) - 16π^2x**

Use 4th-ord. Adams Method to solve from t=0 until t=1 w/ h=0.1

Predictor-Corrector solution: **Undamped, harmonic oscillator**

$$m\frac{d^2x}{dt^2} + kx = P(t)$$

First, we need start-up values. Use 4th-order RK to get 4 points:

t	x(t)	y(t)
0.0	0.0000	0.0000
0.1	0.1038	2.6234
0.2	0.5324	5.1401
0.3	0.8692	0.4949

Predictor-Corrector solution: **Undamped, harmonic oscillator**

Second, the Adams-Bashforth 4th-order predictor is:

$$y_{i+1} = y_i + \frac{h}{24}\{55\,f(x_i,y_i) - 59\,f(x_{i-1},y_{i-1}) + 37\,f(x_{i-2},y_{i-2}) - 9\,f(x_{i-3},y_{i-3})\} + O(h^5)$$

Example :

We apply this predictor to both x and v:

x(0.4) = x(0.3) + [55(0.4949) - 59(5.1401) + 37(2.6234) - 9(0.0)] =

  = **0.1235**

v(0.4) = v(0.3) + (0.1/24) [55 g(0.3, 0.8692, 0.4949)

    − 59 g(0.2, 0.5324, 5.1401)

    + 37 g(0.1, 0.1038, 2.6234) − 9 g(0,0,0)] = **− 11.2492**

    where dv/dt = g(t, x, v) = 80 sin 8t − $16\pi^2$ x

Predictor-Corrector solution: **Undamped, harmonic oscillator**

Third, the Adams-Moulton 4th-order corrector is:

$$y_{i+1} = y_i + \frac{h}{24}\left\{9\,f(x_{i+1},y_{i+1}) + 19\,f(x_i,y_i) - 5\,f(x_{i-1},y_{i-1}) + f(x_{i-2},y_{i-2})\right\} + O(h^5)$$

We apply the corrector to **both** x and v to obtain the 1st iteration:

x(0.4) = x(0.3) + (0.1/24) [9 (**-11.2492**) + 19(0.4949)

$$- 5(5.1401) + (2.6234)] = 0.8169$$

v(0.4) = v(0.3) + (0.1/24) [9 g(0.4, **0.1235**, **-11.2492**)

$$+ 19 \, g(0.3,...) \, ...] = -6.7435$$

where dv/dt = g(t, x, v) = 80 sin 8t – $16\pi^2$ x

We then can **iterate** until convergence:

x(0.4) = 0.8169, 0.842862, 0.843847, 0.843834,...

v(0.4) = -6.7435, -10.84973, -11.00371, -11.00949,...

Do the Second and Third tasks for each successive step.

Numerical Methods for ODE's

Advantages of Runge-Kutta [such as $O(h^4)$ formula]

    · simple to apply

    · self-starting (single-step)

    · easy to change step size

    · always stable (all h)

    · can use matched pairs of different order to estimate local
      truncation error

Advantages of Predictor-Correctors
[such as Adams-Moulton Formula of $O(h^4)$]

    ● without iteration is twice as efficient as Runge-Kutta 4th-Order

    ● local truncation error is easily estimated from difference so we
      can adjust step sizes and apply modifiers

For same step sizes, error terms are reasonably similar.

Summary

Single-step   Multi-step (predictor-corrector)
Euler   Non-self-starting Heun
Heun   Milne (Newton-Cotes n=3)
Mid-point   Adams (Adams-Bashford, Adams-Moulton)
RK4

Stiffness - stability

Adaptive methods – local error estimates – modifiers
Half step
RK Fehlberg

Outline of our Study of ODE's

I. Single-Step Methods for I.V. Problems (C&C Ch. 25)

    a. Euler

    b. Heun and Improved Polygon

    c. General Runge-Kutta

    d. Adaptive step-size control

II. Stiff ODEs (C&C 26.1)

III. Multi-Step Methods for I.V. Problems (C&C 26.2)

    a. Non-Self-Starting Heun

    b. Newton-Cotes

    c. Adams

IV. Boundary Value Problems (C&C Ch. 27.1 & 27.2)

    a. Solution Methods

        1. Shooting Method

        2. Finite Difference Method

    b. Eigenvalue Problems $\{ [A] - l[I] \} \{x\} = \{0\}$

ODE's – Boundary Value Problems

Recall: Unique solution to an nth-order ODE
requires n given conditions

Conditions for a 2nd-order ODE (requires 2 given conditions):

Initial value problem

$$y(x_o) = y_o$$
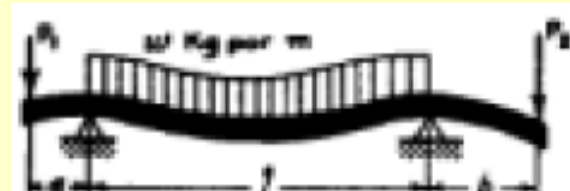$$y'(x_o) = y'_o$$

Boundary value problem

$$y(x_o) = y_o$$
$$y(x_f) = y_f$$

$$y(0) = y'(0) = 0, \; y''(l) = y'''(l) = 0.$$

Beam Equation:

$$EIy^{IV)}(x) = -p(x)$$

$$y(0) = y(l) = 0, \; y''(0) = y''(l) = 0$$

Beam Equation

Sailboat mast deflection problem

$$\frac{d^4 v}{dz^4} = \frac{f(z)}{EI} \qquad \textbf{(Euler-Bernoulli Law of Bending)}$$
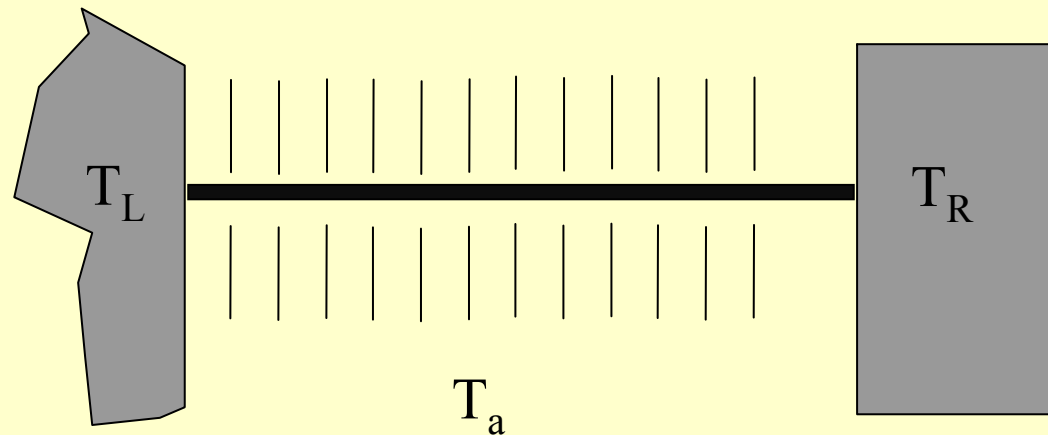
at the base of mast: $v(0) = 0$; $v'(0) = 0$;

at the top of mast: $v''(L) = 0$; $v'''(L) = 0$

where:

E = Modulus of Elasticity (material property)

I = Moment of Inertia (geometry of material)

$f(z)$ = wind pressure at height z

$v(x)$ = deflection (displacement) from vertical

z = height

$E\,I\,v''$ = moment

$E\,I\,v'''$ = shear (applied force)

ODE's – Boundary Value Problem   (C&C 27.1)

$$\frac{d^2T}{dx^2} = h'(T - T_a)$$

Solve for the steady-state
temperature distribution in the rod
For a given T_L and $T_R < T_L$

ODE's – Boundary Value Problem – Shooting Method

$$\text{Solve} \ = \ \frac{d^2 y}{dx^2} = f(x, y, y') \qquad \text{given } y(x_o) = y_o \text{ and } y(x_f) = y_f$$

1. Convert 2nd-Order ODE to two 1st-Order ODE's

$$\frac{dy}{dx} = z \qquad\qquad \frac{dz}{dx} = f(x, y, z)$$

2. Given initial condition, $y(x_o) = y_o$,
         estimate (best guess) the initial condition $z(x_o) = z_o^{(1)}$

3. Solve ODE using the assumed initial values from $x = x_o$ to $x = x_f$
         using stepping methods. Because we **estimated** the initial
         conditions for $z(x_o)$; will find $y(x_f) \neq y_f$

ODE's – Boundary Value Problem – Shooting Method

4. Re-estimate initial condition $z(x_o) = z_o^{(2)}$, and again solve the assumed initial value ODE from $x = x_o$ to $x = x_f$.

   Because we estimated the I.C. $z(x_o)$, $y(x_f) \neq y_f$.

5. Interpolate or extrapolate to find the "correct" $z(x_o) = z_o$

   Given y_f, $(y_f^{(1)}, z_0^{(1)})$, and $(y_f^{(2)}, z_0^{(2)})$:

$$z_o = z_o^{(1)} + \frac{z_o^{(2)} - z_o^{(1)}}{y_f^{(2)} - y_f^{(1)}}\left(y_f - y_f^{(1)}\right)$$

[This equation is <u>not</u> in C&C in general form]

If ODE is linear, z_o is the correct solution

If ODE in nonlinear, iterate until $y_f^{(h)} \approx y_f$

Boundary Value Problem: Classical Shooting Method

Example: Single-degree-of-freedom, undamped, harmonically driven oscillator (see also C&C 28.4, p. 797)

ODE is 2nd order:

$$m \frac{d^2x}{dt^2} + kx = P(t) \qquad \text{or} \qquad \frac{d^2x}{dt^2} + \omega^2 x = \frac{P(t)}{m}$$

m = mass

k = spring constant

$$\omega = \sqrt{\frac{k}{m}} = \text{circular frequency}$$

P(t) = forcing function $= P \sin(\gamma t)$ $T = \frac{2\pi}{\omega}$ = natural period

Now, we are given **boundary** conditions:

$$x(0) = 0 \quad \text{and} \quad x(1) = 0.5$$

(that is, the initial condition $v = dx/dt$ at $t = 0$ is unknown, v_0)

Boundary Value Problem: Classical Shooting Method

Use the Shooting Method by employing a 4th-order RK

$$\frac{d^2x}{dt^2} + \omega^2 x = \frac{P(t)}{m}$$

Recast the 2nd-order ODE as two 1st-order ODEs:

$$\frac{dx}{dt} = v = f(t, x, v) \qquad \text{with } x(0) = 0$$

$$\frac{dv}{dt} = \frac{P\sin(\gamma t)}{m} - \omega^2 x = g(t, x, v) \qquad \text{with } v(0) = v_0$$

Let $m = 0.25$, $k = 4\pi^2$, $P = 20$, $\gamma = 8$,

and thus $g(t, x, v) = 80\sin(8t) - 16\pi^2 x$.

Boundary Value Problem: Classical Shooting Method

Use 4th-ord. RK Method to solve from t=0 until t=1 w/ h=0.1.

1. Guess v(0) = 1.0

2. Find x(1) = **0.91687** (Note: not equal to 0.5)

3. Guess v(0) = 100

4. Find x(1) = **0.09332**

5. Because our ODE is linear, we can always interpolate ➔ yields the required initial value.

$$v(0) = \frac{100 - 1.0}{0.09332 - 0.91687}(0.5 - 0.91687) = \textbf{51.113}$$

6. With v(0) = 51.113, check for x(1) = 0.5000   **OK**

Finite Difference Method

- The shooting method is inefficient for higher-order equations with several boundary conditions.

- Finite Difference method has advantage of being direct (not iterative) for linear problems, but requires the solution of simultaneous algebraic equations.
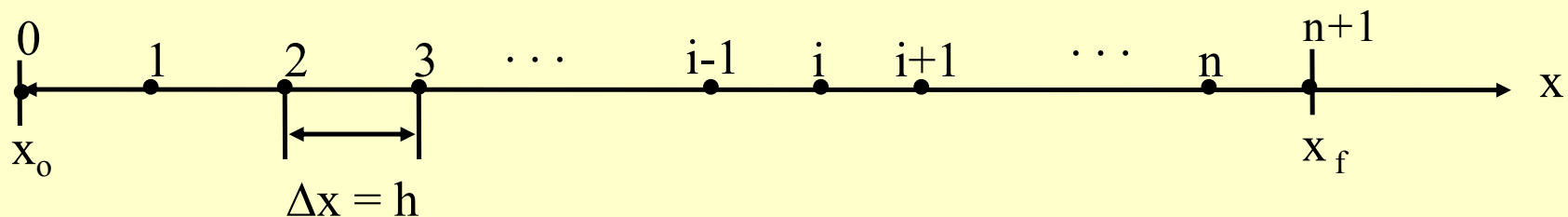
Finite Difference Method

Approach:
1. Divide domain to obtain n interior discrete points
       (usually evenly spaced @ h).
2. Write a finite difference expression for the ODE at each
      interior point.
3. Use known values of y at $x = x_o$ and $x = x_f$

4. Set up n linear equations with n unknowns. System is banded
     and often symmetric, so solve with an efficient method.

   Note: If higher-order FDD equations and/or centered
          differences are used, you may need to employ
          imaginary or "phantom" points outside of domain to
          express B.C.'s (together with appropriate FD versions
          of B.C.'s).

Finite Difference Method

Sailboat mast deflection problem

$$\frac{d^4 v}{dz^4} = \frac{f(z)}{EI} \quad \textbf{(Euler-Bernoulli Law of Bending)}$$

at the base of mast: $v(0) = 0;\; v'(0) = 0;$

at the top of mast: $v''(L) = 0;\; v'''(L) = 0$

where:
- E = Modulus of Elasticity (material property)
- I = Moment of Inertia (geometry of material)
- $f(z)$ = wind pressure at height z
- $v(x)$ = deflection (displacement) from vertical
- z = height
- $E\,I\,v''$ = moment
- $E\,I\,v'''$ = shear (applied force)

Finite Difference Method

Sailboat mast deflection problem: $\dfrac{d^4 v}{dz^4} = v'''' = \dfrac{f(z)}{EI}$

at the base (z = 0): $v(0) = 0;\ v'(0) = 0;$

at the top (z = L): $v''(L) = 0;\ v'''(L) = 0$

$$v'''' = \frac{d^4 v}{dx^4} \approx \frac{v_{i-2} - 4v_{i-1} + 6v_i - 4v_{i+1} + v_{i+2}}{h^4} \quad \text{[dist. load,f(z)]/EI}$$

$$v''' = \frac{d^3 v}{dx^3} \approx \frac{-v_{i-2} + 2v_{i-1} - 2v_{i+1} + v_{i+2}}{2h^3} \quad \text{(shear)/EI}$$

$$v'' = \frac{d^2 v}{dx^2} \approx \frac{v_{i-1} - 2v_i + v_{i+1}}{h^2} \quad \text{(bending moment)/EI}$$

$$v' = \frac{dv}{dx} \approx \frac{v_{i-1} - v_{i+1}}{2h} \quad \text{(slope)}$$
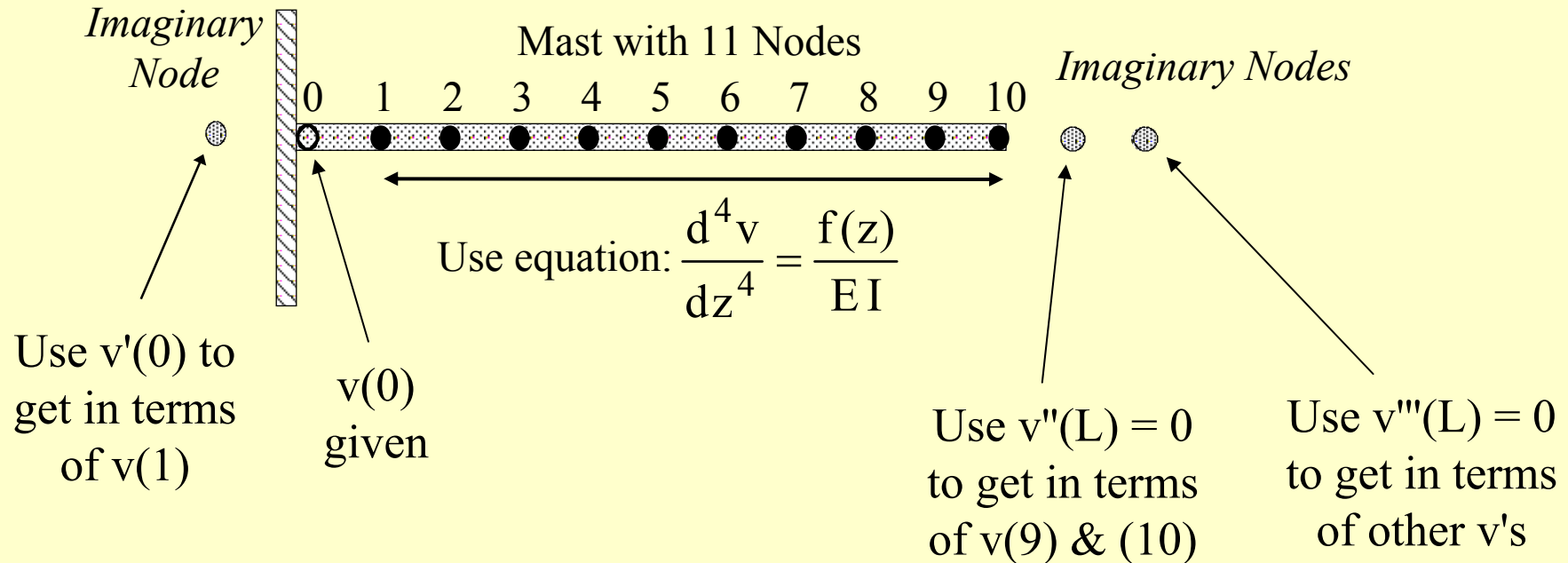
Finite Difference Method

Sailboat mast deflection problem: $\dfrac{d^4v}{dz^4} = \dfrac{f(z)}{EI}$

at the base: $v(0) = 0;\ v'(0) = 0;$

at the top: $v''(L) = 0;\ v'''(L) = 0$

Imaginary Node

Mast with 11 Nodes

Imaginary Nodes

0 1 2 3 4 5 6 7 8 9 10

Use equation: $\dfrac{d^4v}{dz^4} = \dfrac{f(z)}{EI}$

Use v'(0) to get in terms of v(1)

v(0) given

Use v"(L) = 0 to get in terms of v(9) & (10)

Use v'"(L) = 0 to get in terms of other v's

Finite Difference Method

$$\frac{d^4 v}{dz^4} = \frac{f(z)}{EI}$$

@ pt. i: $\quad v_{i-2} - 4\,v_{i-1} + 6\,v_i - 4\,v_{i+1} + v_{i+2} = h^4\,f(x_i)/EI$

Using $h = L/10$, write the FD equations at points 1, 2, 9 and 10:

@ i = 1 $\quad v_{-1} - 4\,v_0 + 6\,v_1 - 4\,v_2 + v_3 = h^4\,f(x_1)/EI$ $\qquad$ (1)

@ i = 2 $\quad v_0 - 4\,v_1 + 6\,v_2 - 4\,v_3 + v_4 = h^4\,f(x_2)/EI$ $\qquad$ (2)

@ i = 9 $\quad v_7 - 4\,v_8 + 6\,v_9 - 4\,v_{10} + v_{11} = h^4\,f(x_9)/EI$ $\qquad$ (3)

@ i = 10 $\quad v_8 - 4\,v_9 + 6\,v_{10} - 4\,v_{11} + v_{12} = h^4\,f(x_{10})/EI$ $\qquad$ (4)

We now need to eliminate v_{-1}, v_0, v_{11}, and v_{12}

with the four boundary conditions

Finite Difference Method

$$\frac{d^4 v}{dz^4} = \frac{f(z)}{EI}$$

Eliminating v_{-1}, v_0, v_{11}, and v_{12} with the four Boundary conditions

$v(0) = 0$: $\qquad\qquad\qquad\qquad\qquad\qquad v_0 = 0$

$v'(0) = 0$: $\qquad\qquad \dfrac{v_{-1} - v_1}{2h} = 0 \qquad\qquad v_{-1} = v_1$

$v''(L) = 0$: $\qquad\qquad \dfrac{v_9 - 2v_{10} + v_{11}}{h^2} = 0 \qquad\qquad v_{11} = 2v_{10} + v_9$

$v'''(L) = 0$: $\qquad \dfrac{-v_8 + 2v_9 - 2v_{11} + v_{12}}{2h^3} = 0 \qquad v_{12} = v_8 - 4v_9 + 4v_{10}$

Finite Difference Method

$$\frac{d^4 v}{dz^4} = \frac{f(z)}{EI}$$

We substitute these into equations (1) to (4) to eliminate the unknowns at the "imaginary" points

$v_0 = 0$ $v_{-1} = v_1$

$v_{11} = 2v_{10} + v_9$ $v_{12} = v_8 - 4v_9 + 4v_{10}$

@ i = 1 $7 v_1 - 4 v_2 + v_3 = h^4 f(x_1) / EI$ (1a)

@ i = 2 $- 4 v_1 + 6 v_2 - 4 v_3 + v_4 = h^4 f(x_2) / EI$ (2a)

@ i = 9 $v_7 - 4 v_8 + 5 v_9 - 2 v_{10} = h^4 f(x_9) / EI$ (3a)

@ i = 10 $v_8 - 4 v_9 + 2 v_{10} = h^4 f(x_{10}) / EI$ (4a)

Finite Difference Method

$$
\begin{bmatrix}
7 & -4 & 1 & & & & & & & \\
-4 & 6 & -4 & 1 & & & & & & \\
1 & -4 & 6 & -4 & 1 & & & & & \\
& 1 & -4 & 6 & -4 & 1 & & & & \\
& & 1 & -4 & 6 & -4 & 1 & & & \\
& & & 1 & -4 & 6 & -4 & 1 & & \\
& & & & 1 & -4 & 6 & -4 & 1 & \\
& & & & & 1 & -4 & 6 & -4 & 1 \\
& & & & & & 1 & -4 & 5 & -2 \\
& & & & & & & 1 & -4 & 2
\end{bmatrix}
\begin{Bmatrix}
v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \\ v_8 \\ v_9 \\ v_{10}
\end{Bmatrix}
= \frac{h^4 f}{EI}
\begin{Bmatrix}
f(x_1) \\ f(x_2) \\ f(x_3) \\ f(x_4) \\ f(x_5) \\ f(x_6) \\ f(x_7) \\ f(x_8) \\ f(x_9) \\ f(x_{10})
\end{Bmatrix}
$$

Finite Difference Method

Once we have solved for all the v_i, we can obtain secondary results such as bending moments and shear forces by substituting the finite-divided-difference operators and the values of the v_i into such equations as:

$$M = EI \ v''$$
$$V = EI \ v'''$$

For more refined results, we can use a smaller h and more segments.

Merits of Different Numerical Methods for ODE Boundary Value Problems

- **Shooting method**

    Conceptually simple and easy.

    Inefficient for higher-order systems w/ many boundary conditions.

    May not converge for nonlinear problems.

    Can blow up for bad guess of initial conditions.

- **Finite Difference method**

    Stable

    Direct (not iterative) for linear problems.

   Requires solution of simultaneous algebraic eqns.

    More complex.

    FD better suited for eigenvalue problems.

Engineering Applications of Eigenvalue Problems

Natural periods and modes of vibration of structures and dynamical systems.
   (C&C Ex. 27.4, p. 761)

Application: determination of the natural frequencies of a system

Example: A mass-spring system with three identical (frictionless) masses connected by three springs with different spring constants.
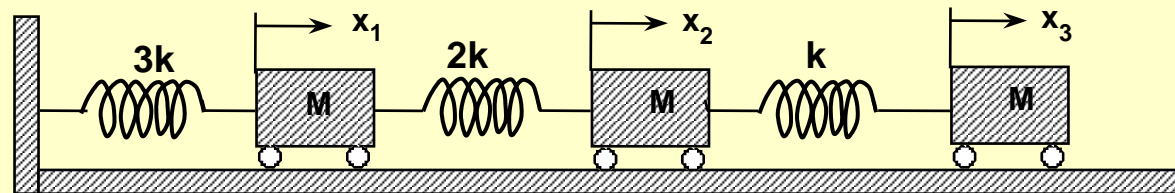
The displacement of each spring is measured relative to its own local coordinate system with an origin at the spring's equilibrium position.

$$m\frac{d^2x_1}{dt^2} = -3kx_1 - 2k(x_1 - x_2)$$

$$m\frac{d^2x_2}{dt^2} = -2k(x_2 - x_1) - k(x_2 - x_3)$$

$$m\frac{d^2x_3}{dt^2} = -k(x_3 - x_2)$$

Engineering Applications of Eigenvalue Problems

If one assumes that x has the form:

$$x_i = a_i \cos \omega t \; ; \qquad\qquad d^2x/dt^2 = - \omega^2 a_i \cos \omega t$$

Then, with $\lambda = m \omega^2 / k$, the governing equations become:

$$-\frac{m\omega^2}{k} a_1 = -5a_1 + 2a_2 \quad ==> \quad 5a_1 - 2a_2 = \lambda a_1$$

$$-\frac{m\omega^2}{k} a_2 = 2a_1 - 3a_2 + a_3 \quad ==> \quad -2a_1 + 3a_2 - a_3 = \lambda a_2$$

$$-\frac{m\omega^2}{k} a_3 = a_2 - a_3 \quad ==> \quad -a_2 + a_3 = \lambda a_3$$

Engineering Applications of Eigenvalue Problems

$$5a_1 - 2a_2 = \lambda a_1$$
$$-2a_1 + 32a_2 - a_3 = \lambda a_2 \quad \Longrightarrow \quad \begin{bmatrix} 5 & -2 & 0 \\ -2 & 3 & -1 \\ 0 & -1 & 1 \end{bmatrix} \begin{Bmatrix} a_1 \\ a_2 \\ a_3 \end{Bmatrix} = \lambda \begin{Bmatrix} a_1 \\ a_2 \\ a_3 \end{Bmatrix}$$
$$-a_1 + a_2 = \lambda a_3$$

$$[\mathbf{A}] \qquad \{\mathbf{x}\} = \lambda \quad \{\mathbf{x}\}$$

$$\text{or} \qquad \begin{bmatrix} 5-\lambda & -2 & 0 \\ -2 & 3-\lambda & -1 \\ 0 & -1 & 1-\lambda \end{bmatrix} \begin{Bmatrix} a_1 \\ a_2 \\ a_3 \end{Bmatrix} = 0$$

$$[\mathbf{A} - \mathbf{I}\,\lambda] \qquad \{\mathbf{x}\} = \mathbf{0}$$

Engineering Applications of Eigenvalue Problems

For a non-trivial solution:

$$\det [\mathbf{A} - \mathbf{I}\lambda] = 0 \quad \rightarrow \quad \text{find} \quad \det \begin{bmatrix} 5-\lambda & -2 & 0 \\ -2 & 3-\lambda & -1 \\ 0 & -1 & 1-\lambda \end{bmatrix} = 0$$

The determinant yields a cubic equation for λ:

$$(5 - \lambda)\,[(3 - \lambda)(1 - \lambda) - (-1)(-1)] - 2\,[(-1)(0) - (-2)(1 - \lambda)] = 0$$

$$(5 - \lambda)\,[2 - 4\lambda + \lambda^2] - 2\,[2 - 2\lambda] = 0$$

$$6 - 18\lambda + 9\lambda^2 - \lambda^3 = 0$$

The three solutions of the cubic equation are the three eigenvalues:

$$\lambda_1 = 6.29 \qquad\qquad \lambda_2 = 2.29 \qquad\qquad \lambda_3 = 0.42$$

Fast oscillation   Slow oscillation

Engineering Applications of Eigenvalue Problems

The corresponding eigenvectors $\{a_i\}$ are found by solving:

$$\begin{bmatrix} 5-\lambda_i & -2 & 0 \\ -2 & 3-\lambda_i & -1 \\ 0 & -1 & 1-\lambda_i \end{bmatrix} \begin{Bmatrix} a_{1i} \\ a_{2i} \\ a_{3i} \end{Bmatrix} = 0$$

Because this is a homogeneous equation, we can only find the relative values of the a_i's. For $\lambda_1 = 6.29$

$$\begin{bmatrix} 5-6.29 & -2 & 0 \\ -2 & 3-6.29 & -1 \\ 0 & -1 & 1-6.29 \end{bmatrix} \begin{Bmatrix} a_{1i} \\ a_{2i} \\ a_{3i} \end{Bmatrix} = 0$$

$$\frac{a_2}{a_1} = -0.645 \qquad \frac{a_3}{a_1} = 0.122$$

Engineering Applications of Eigenvalue Problems

$$\frac{a_2}{a_1} = -0.645 \qquad \frac{a_3}{a_1} = 0.122$$

Only the relative values of the a_i's are significant. Thus setting $a_1 = 1.00$ we have:

<table>
<tr><td>**Fast**</td><td></td><td>**Slow**</td></tr>
<tr><td>for $\lambda = 6.29$</td><td>for $\lambda_2 = 2.29$</td><td>for $\lambda_2 = 0.42$</td></tr>
</table>

$$\{a\}_1 = \left\{ \begin{array}{c} 1.000 \\ -0.645 \\ 0.122 \end{array} \right\} \qquad \{a\}_2 = \left\{ \begin{array}{c} 0.74 \\ 1.000 \\ -0.77 \end{array} \right\} \qquad \{a\}_2 = \left\{ \begin{array}{c} 0.25 \\ 0.55 \\ 1.000 \end{array} \right\}$$

Engineering Applications of Eigenvalue Problems

Thus, there are **3 possible natural frequencies**.

Recall that the frequency is defined as $\omega = \sqrt{\dfrac{\lambda k}{m}}$

$$[A] \; \{x\} = \lambda \; \{x\}$$

$$[A - I\lambda] \; \{x\} = 0$$

$$\det [A - I\lambda] = 0 ==> \text{find } \lambda$$

- If $[A]$ is n x n, there are n eigenvalues λ_i
  and n eigenvectors $\{x\}_i$

- If $[A]$ is symmetric, the eigenvectors are orthogonal:

$$\{x\}_i^T \{x\}_j = 0 \qquad\qquad \text{if} \quad j \neq i$$
$$= 1 \qquad\qquad \text{if} \quad j = i$$

POWER METHOD *for Eigenvector Analysis*

(An iterative approach for solving for eigenvalues & eigenvectors)

Assume that a vector $\{y\}$ can be expressed as a linear combination of the eigenvectors:

$$\{y\} = b_1\{x_1\} + b_2\{x_2\} + \ldots + b_n\{x_n\} = \sum_{i=1}^{n} b_i\{x_i\}$$

Multiplying the above equation by [A] yields:

$$[A]\{y\} = \sum_{i=1}^{n} b_i[A]\{x_i\} = \sum_{i=1}^{n} b_i \lambda_i\{x_i\}$$

Ordering the λ_i so that $\qquad |\lambda_1| > |\lambda_2| > |\lambda_3| > \ldots > |\lambda_n|$

Then $[A]\{y\} = \lambda_1\left(b_1\{x_1\} + \sum_{i=2}^{n} b_i\left(\dfrac{\lambda_i}{\lambda_1}\right)\{x_i\}\right)$ where $\left|\dfrac{\lambda_i}{\lambda_1}\right| < 1$

POWER METHOD for Eigenvector Analysis

Multiplying the equation by [A] again, we have:

$$[A]^2 \{y\} = \lambda_1^2 \left(b_1\{x_1\} + \sum_{i=2}^{n} b_i \left(\frac{\lambda_i}{\lambda_1}\right)^2 \{x_i\} \right)$$

Repeating this process m times:

$$[A]^m \{y\} = \lambda_1^m \left(b_1\{x_1\} + \sum_{i=2}^{n} b_i \left(\frac{\lambda_i}{\lambda_1}\right)^m \{x_i\} \right)$$

Since $\left(\dfrac{\lambda_i}{\lambda_1}\right) ==> 0$ for all $i \neq 1$ as $m ==> \infty$,

$$\text{then} \quad [A]^m \{y\} \blacktriangleright \lambda_1^m b_1\{x_1\}$$

Since we can only know the relative values of the elements of $\{x\}$, we may normalize $\{x\}$. If $\{x\}$ is normalized such that the largest element is equal to $\{1\}$, then is the first eigenvalue, λ_1

POWER METHOD Numerical Example

$$\begin{bmatrix} 3 & 7 & 9 \\ 9 & 4 & 3 \\ 9 & 3 & 8 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \lambda \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix}$$

Find the maximum eigenvector. Initial guess: $\{y\}^T = \{1 \ \ 1 \ \ 1\}$

$$[A]\{y\} = \begin{bmatrix} 3 & 7 & 9 \\ 9 & 4 & 3 \\ 9 & 3 & 8 \end{bmatrix} \begin{Bmatrix} 1 \\ 1 \\ 1 \end{Bmatrix} = \begin{Bmatrix} 19 \\ 16 \\ 20 \end{Bmatrix} ==> 20 \qquad \begin{Bmatrix} 0.95 \\ 0.80 \\ 1.00 \end{Bmatrix}$$

Approx. eigen-
eigenvalue vector

POWER METHOD Numerical Example

2ⁿᵈ iteration

$$[A]^2 \{y\} = \begin{bmatrix} 3 & 7 & 9 \\ 9 & 4 & 3 \\ 9 & 3 & 8 \end{bmatrix} \begin{Bmatrix} 0.95 \\ 0.80 \\ 1.00 \end{Bmatrix} = 18.95 \begin{Bmatrix} 0.92084 \\ 0.77836 \\ 1.00 \end{Bmatrix}$$

Approximate error, ε_a:

$$\max_j \left| \frac{(e_{i,j} - e_{i-1,j})}{e_{i,j}} \right| * 100\% = \left| \frac{0.92084 - 0.95}{0.92084} \right| * 100\% = 4.2\%$$

3ʳᵈ iteration

$$[A]^3 \{y\} = \begin{bmatrix} 3 & 7 & 9 \\ 9 & 4 & 3 \\ 9 & 3 & 8 \end{bmatrix} \begin{Bmatrix} 0.92084 \\ 0.77836 \\ 1.00 \end{Bmatrix} = 18.623 \begin{Bmatrix} 0.92420 \\ 0.77331 \\ 1.00 \end{Bmatrix}$$

$$\varepsilon_a = \left| \frac{0.92084 - 0.95}{0.92084} \right| * 100\% = 0.75\%$$

POWER METHOD Numerical Example

After 10 iterations:

$$\lambda = 18.622 \qquad \{x\} = \begin{Bmatrix} 0.92251 \\ 0.77301 \\ 1.00 \end{Bmatrix} \qquad \varepsilon_a = 0.00033\%$$

Engineering Applications of Eigenvalue Problems

1. **Natural periods & modes of vibration** of dynamic systems and structures. Boundary-value problem from separation of variables:

$$m * a = \text{force} \Rightarrow m(x)\frac{\partial\, y(x,t)}{\partial\, t^2} = k(x)\frac{\partial^2 y}{\partial\, x^2}$$

If one assumes $y(x,t) = \exp(i\omega t)\, u(x)$

then displacement function $u(x)$ satisfies ODE:

$$-\omega^2\, u(x) = [k(x)/m(x)]\, d^2u/dx^2$$

which may be written: $a(x)\, d^2u/dx^2 + \omega^2\, u(x) = 0$

With a FD approximation of $u_{,xx} = d^2u/dx^2$ this becomes:

$$[A]\,\{u_i\} = -\omega^2\,\{u_i\}$$

Engineering Applications of Eigenvalue Problems

2. **Buckling loads and modes of structures (C&C 27.2.3, p. 762)**

   Deflection of vertically loaded beam with horizontally constrained end:

   $$EI \; y'' - Py = 0; \;\; y(0) = 0; \;\; y(L) = 0.$$

   Is there deflection?

3. Directions and values of principal stresses.

4. Directions and values of principal moments of inertia.

5. Condition numbers linear systems of equations.

6. Stability criteria for numerical solution of PDE's.

7. Other problems in which "principal values" are sought.

Engineering Applications of Eigenvalue Problems

Matrix Form of Eigenvalue Problem

$$[A] \{x\} = \lambda \{x\}$$

➔ $([A] - \lambda [I]) \{x\} = 0$ *but want* $\{x\} \neq 0$

Computation of Eigenvalues and Eigenvectors

1. Power and Inverse Power method for largest and smallest eigenvalues. (C&C 27.2.5, p. 767)

2. Direct numerical algorithms: Jacobi, Given, Householder, and QR factorization. (C&C 27.2.6, p. 770)

3. Use of characteristic polynomial for "toy" problems. (C&C 27.2.4, p. 765)

Power Method for the largest Eigenvalue

Compute $x_{t+1} = A \, x_t \; / \; \| x_t \|$

for larger and larger t to estimate largest $\lambda(A)$.

Always works for Symmetric A.

Eigenvalue Problems: Power Method

Inverse Power Method for the smallest Eigenvalue

Compute $x_{t+1} = B\, x_t \;/\; \| x_t \|$

for larger and larger t to estimate largest $\lambda(B)$.

Here $B = A^{-1}$. Eigenvalues of B are inverse of those of A.

BUT WE DO NOT COMPUTE A^{-1} ! (Don't do it.)

Instead use LU decomposition of A.

Engineering Applications of Eigenvalue Problems

Shifting method for any Eigenvalue:

Suppose we want to compute eignvector whose eigenvalue is about δ.

Let $B = (A - \delta I)^{-1}$ and compute $x_{t+1} = B x_t / \| x_t \|$

for larger and larger t to estimate largest $\lambda[(A - \delta I)^{-1}]$.

This will compute the eignvalue *nearest* to δ !

But we DO NOT COMPUTE INVERSE. Use LU decomposition.

If A has eigenvector-value pairs (e_i, λ_i),

$(A - \delta I) e_i = A e_i - \delta I e_I = \lambda_i e_i - \delta e_i = (\lambda_i - \delta) e_i$

Thus shifted matrix has same eignvectors e_i as A with shifted eigenvalues $(\lambda_i - \delta)$.

Hence $(A - \delta I)^{-1}$ has eigenvalues $(\lambda_i - \delta)^{-1}$.