University of Idaho

Lecture 40

Rayleigh quotient iteration

Note Rayleigh quotient obtains (approximation to) e'value when (approx.) e'vector is known.

Inverse iteration: obtains e'vector from e'value.

Rayleigh quotient iteration: combines both ideas.

idea: update μ

algorithm

1. $v^{(0)}$: given, $\|v^{(0)}\|_2 = 1$, $\lambda^{(0)} = (v^{(0)})^T A v^{(0)}$: % corresponding % Rayleigh quotient
2. for $k = 1, 2, \ldots$
3. solve $(A - \lambda^{(k-1)} I) w = v^{(k-1)}$; % apply $(A - \lambda^{(k-1)} I)^{-1}$
4. $v^{(k)} = w / \|w\|_2$
5. $\lambda^{(k)} = (v^{(k)})^T A v^{(k)}$; % apply Rayleigh quotient

<u>Thm</u> If $v^{(0)}$ is sufficiently close to an e'vector q_J, then

$$\left.\begin{aligned} \|v^{(k+1)} - (\pm q_J)\| &= O(\|v^{(k)} - (\pm q_J)\|^3) \\ |\lambda^{(k+1)} - \lambda_J| &= O(|\lambda^{(k)} - \lambda_J|^3) \end{aligned}\right\} : \quad \underline{\text{cubic convergence}}$$

<u>Pf</u>: omit

<u>ex</u> $A = \begin{pmatrix} 2 & 1 & 1 \\ 1 & 3 & 1 \\ 1 & 1 & 4 \end{pmatrix}$, $\lambda_1 = 5.214319743377$, $v^{(0)} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \cdot \frac{1}{\sqrt{3}}$

k	power method	shifter inverse iteration, $\mu = 5$	Rayleigh quotient iteration
0	5.0	5.0	$\underline{5}.0$
1	5.181818	5.213114	$\underline{5.2}13114$
2	$\underbrace{5.2}_{2}08192$	$\underbrace{5.214312}_{6}617$	$\underbrace{5.2143197431}_{10}84$

<u>Note</u>: each iteration of Rayleigh quotient iteration method triples # of digits of accuracy

For more info: see Numerical Linear Algebra by L. T. Trefethen & D. Bau, pg. 202-210

Back to <u>systems of ODEs</u>

Recall: IVP

$$y' = f(y) \qquad \text{or} \qquad y' = f(t, y) \qquad a \le t \le b$$
$$y(0) = y_0 \qquad\qquad\quad y(a) = y_0$$

For scalar equations (IVPs, rather), we discussed ~~ex~~

- existence, uniqueness, well-posedness
- Euler, modified Euler, Runge-Kutta (explicit), backward Euler (implicit) — one-step methods
- explicit / implicit methods
- one-step or multistep methods (more - later)

Q How can we extend these methods to systems of ODEs?

Def m^{th} order system of 1^{st} order IVPs:

$$u_1' = f_1(t, u_1, u_2, \ldots, u_m)$$

$$u_2' = f_2(t, u_1, u_2, \ldots, u_m)$$

$$\vdots$$

$$u_m' = f_m(t, u_1, u_2, \ldots, u_m)$$

for $a \le t \le b$ with

$$u_1(a) = \alpha_1, \quad u_2(a) = \alpha_2, \quad \ldots, \quad u_m(a) = \alpha_m$$

- Vector form

$$\vec{u}\,' = \vec{f}(t, \vec{u}), \qquad a \le t \le b \qquad \vec{u} = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_m \end{pmatrix} \qquad \vec{f} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_m \end{pmatrix}$$

$$\vec{u}\,'(a) = \vec{\alpha}$$

Then apply numerical method to each component of $\vec{u}$.

How to transfrom n^{th} order DE into a system of 1^{st} order DEs?

Consider a single n^{th} order DE:

$$y^{(n)} = f(t, y, y', \dots, y^{(n-1)})$$

Introduce the dependent variables $y_1, y_2, \dots, y_n$ as follows:

$$y_1 = y, \quad y_2 = y', \quad y_3 = y'', \quad \dots, \quad y_n = y^{(n-1)}$$

<u>Note</u> $y_1' = y' = y_2, \quad y_2' = y'' = y_3, \quad \dots, \quad y_n' = y^{(n)}.$

This yields the system:

$$\left.\begin{aligned} y_1' &= y_2 \\ y_2' &= y_3 \\ &\vdots \\ y_{n-1}' &= y_n \\ y_n' &= f(t, y_1, y_2, \dots, y_n) \end{aligned}\right\} \quad \text{n equations of } 1^{st} \text{ order}$$

Multistep methods (for solving IVPs)

$y' = f(y)$

General 2-step method

$$\alpha_0 u_{n+1} + \alpha_1 u_n + \alpha_2 u_{n-1} = h\left(\beta_0 f(u_{n+1}) + \beta_1 f(u_n) + \beta_2 f(u_{n-1})\right)$$

Note

1. assume $\alpha_0 \neq 0$
2. if $\beta_0 = 0$, the scheme is explicit
   if $\beta_0 \neq 0$, the scheme is implicit (a rootfinding method may be needed to find u_{n+1})

Adams-Bashforth

$$u_{n+1} = u_n + \frac{h}{2}\left(3 f(u_n) - f(u_{n-1})\right)$$

explicit, 2nd order accurate, 1 function evaluation per step

Adams - Moulton

$$u_{n+1} = u_n + \frac{h}{12}\left(5f(u_{n+1}) + 8f(u_n) - f(u_{n-1})\right)$$

implicit, 3^{rd} order accurate

Note

1. There are k-step AB and AM methods.
2. a popular predictor corrector method uses AB as predictor and AM as corrector

leap - frog

$$\frac{u_{n+1} - u_{n-1}}{2h} = f(u_n)$$

$$u_{n+1} = u_{n-1} + 2h f(u_n)$$

explicit, 2^{nd} order accurate, 1 function evaluation per step

BDF : backward differentiation formula (**Gear's method**)

$$\frac{3}{2} u_{n+1} - 2 u_n + \frac{1}{2} u_{n-1} = h f(u_n)$$

explicit, 2nd order accurate, 1 function evaluation per step

Software

adaptive time step: ODE23, ODE45 – Matlab

adaptive order methods