Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Solution Methods for Eigenproblems

- **eigenproblem:** $K\varphi = \lambda M\varphi$
- **solution methods:**
  - vector iteration method
  - transformation methods
  - polynomial iteration techniques
  - Sturm sequence iteration method

- **eigenproblem:** $K\varphi = \lambda M\varphi$
- **solution methods:**
  - **vector iteration method**
    - inverse iteration
    - forward iteration
    - Rayleigh quotient iteration
    - matrix deflation and Gram-Schmidt orthogonalisation
  - transformation methods
  - polynomial iteration techniques
  - Sturm sequence iteration method

Shifting in Vector Iteration

- How to improve the convergence rate?

→ shifting

$$\left.\begin{aligned}(K-\mu M)\varphi &= \eta M\varphi \\ K\varphi &= \lambda M\varphi\end{aligned}\right\} \quad \eta_i = \lambda_i - \mu, i = 1.....n$$

relation of eigenvalues:

Shifting in Vector Iteration

- Convergence properties:
  - problem in the basis of eigenvectors Φ

    using the transformation $\varphi = \Phi \Psi$

    we obtain the equivalent eigenproblem $\left(\Lambda - \mu I\right)\Psi = \eta \Psi$

Shifting in Vector Iteration

- Convergence properties: **inverse iteration**

iteration vector: $$z_{l+1}^T = \begin{bmatrix} \dfrac{1}{(\lambda_1 - \mu)^l} & \dfrac{1}{(\lambda_2 - \mu)^l} & \cdots & \dfrac{1}{(\lambda_n - \mu)^l} \end{bmatrix}$$

multiplication with $\lambda_i - \mu, \;\; i = j$:

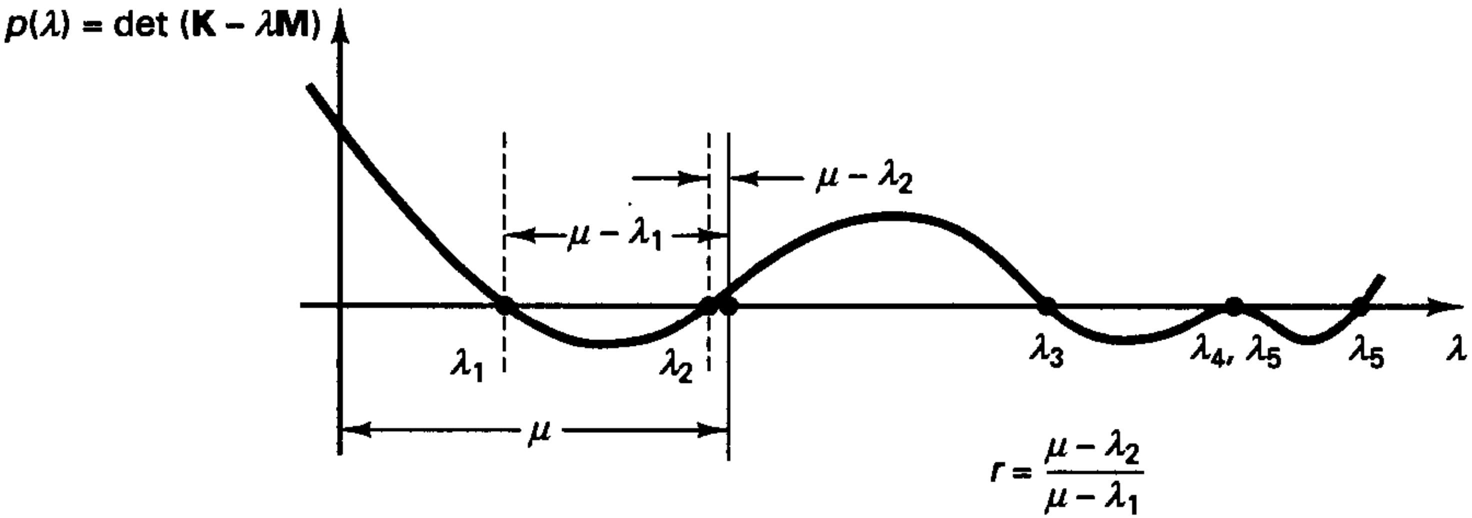
$$\bar{z}_{l+1}^T = \begin{bmatrix} \left(\dfrac{\lambda_j - \mu}{\lambda_1 - \mu}\right)^l & \cdots & \left(\dfrac{\lambda_j - \mu}{\lambda_{j-1} - \mu}\right)^l & 1 & \left(\dfrac{\lambda_j - \mu}{\lambda_{j+1} - \mu}\right)^l & \cdots & \left(\dfrac{\lambda_j - \mu}{\lambda_n - \mu}\right)^l \end{bmatrix}$$

Shifting in Vector Iteration

- Convergence properties: **inverse iteration**
  - in the iteration we have $\vec{z}_{l+1} \rightarrow e_j$
  - meaning that to solve the eigenproblem the iteration vector converges to Φ_j
  - furthermore: $\lambda_i = \eta_j + \mu$
  - convergence rate: $r = \max\limits_{p \neq j} \left| \dfrac{\lambda_j - \mu}{\lambda_p - \mu} \right|$

Shifting in Vector Iteration

- Convergence properties: **inverse iteration**
  - convergence rate: $r = \max_{p \neq j} \left| \frac{\lambda_j - \mu}{\lambda_p - \mu} \right|$
  - since λ_j is nearest μ → $\left| \frac{\lambda_j - \mu}{\lambda_{j-1} - \mu} \right|$ or $\left| \frac{\lambda_j - \mu}{\lambda_{j+1} - \mu} \right|$



Shifting in Vector Iteration

- Convergence properties: **inverse iteration**
  - convergence rate of the Rayleigh coefficient:

$$\left|\frac{\lambda_j - \mu}{\lambda_{j-1} - \mu}\right|^2 \quad \text{or} \quad \left|\frac{\lambda_j - \mu}{\lambda_{j+1} - \mu}\right|^2$$

Shifting in Vector Iteration

- Convergence properties: **forward iteration**
  - convergence rate: $r = \max_{p \neq j} \left| \frac{\lambda_p - \mu}{\lambda_j - \mu} \right|$

→ limited convergence rate in forward iteration

→ by means of shifting convergence only to (λ_n, φ_n) or (λ_1, φ_1)

→ to achieve highest convergence rates in both we need to choose

$\mu = (\lambda_1 + \lambda_{n-1})/2$ resp. $\mu = (\lambda_2 + \lambda_n)/2$

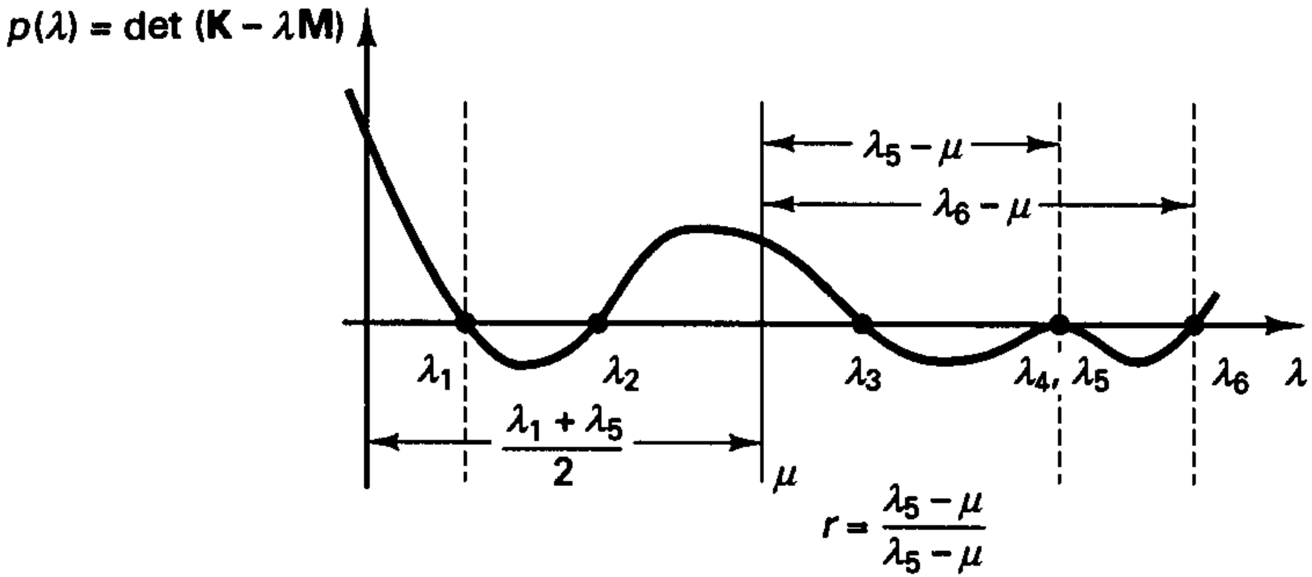
Shifting in Vector Iteration

- Convergence properties: **forward iteration**

  - corresponding convergence rates:

$$\left| \frac{\lambda_{n-1} - \dfrac{\lambda_1 + \lambda_n}{2}}{\lambda_n - \dfrac{\lambda_1 + \lambda_{n-1}}{2}} \right|$$

$$\left| \frac{\lambda_2 - \dfrac{\lambda_2 + \lambda_n}{2}}{\lambda_1 - \dfrac{\lambda_2 + \lambda_n}{2}} \right|$$



  - much higher convergence rate with shifting in inverse iteration

Rayleigh Quotient Iteration

- Improving of convergence rate in inverse iteration by shifting → but how to choose the appropiate shift?
- one possibility: Rayleigh quotient as shhift value

Rayleigh Quotient Iteration

- we assume a starting iteration vector x_1, hence $y_1 = Mx_1$, a starting shift $p(\bar{x}_1)$ (usually 0) and then evaluate for *k=1, 2, ….* :

$$[\mathbf{K} - \rho(\overline{\mathbf{x}}_k)\mathbf{M}]\overline{\mathbf{x}}_{k+1} = \mathbf{y}_k$$

$$\overline{\mathbf{y}}_{k+1} = \mathbf{M}\,\overline{\mathbf{x}}_{k+1}$$

$$\rho(\overline{\mathbf{x}}_{k+1}) = \frac{\overline{\mathbf{x}}_{k+1}^T\,\mathbf{y}_k}{\overline{\mathbf{x}}_{k+1}^T\,\overline{\mathbf{y}}_{k+1}} + \rho(\overline{\mathbf{x}}_k)$$

$$\mathbf{y}_{k+1} = \frac{\overline{\mathbf{y}}_{k+1}}{(\overline{\mathbf{x}}_{k+1}^T\,\overline{\mathbf{y}}_{k+1})^{1/2}}$$

where now $\mathbf{y}_{k+1} \rightarrow \mathbf{M}\boldsymbol{\phi}_i$ and $\rho(\overline{\mathbf{x}}_{k+1}) \rightarrow \lambda_i$ as $k \rightarrow \infty$

- eigenvalue λ_i and corr. eigenvector φ_i to which the iteration converges depend on starting iteration vector x_1 and initial shift $p(\bar{x}_1)$

Matrix Deflation

- inverse iteration converges to λ_1 and φ_1, forward iteration to λ_n and φ_n
- methods can also employed with shifting to calculate other eigenvalues and corresponding eigenvectors
- assuming that we have calculated a specific eigenpair (λ_k, φ_k) and that we require the solution of another eigenpair
- to ensure that we do not coverge again to λ_k and φ_k we need to deflate either the matrices or the iteration vectors

Matrix Deflation

- standard eigenproblem: $K\varphi = \lambda\varphi$
- stable matrix deflation can be carried out by finding an orthogonal matrix **P** whose first column is the calculated eigenvector φ_k

  writing $\mathbf{P} = [\boldsymbol{\phi}_k, \mathbf{p}_2, \ldots, \mathbf{p}_n]$

  we need to have $\boldsymbol{\phi}_k^T \mathbf{p}_i = 0$ for $i = 2, \ldots, n.$

  It then follows $\mathbf{P}^T\mathbf{K}\mathbf{P} = \begin{bmatrix} \lambda_k & \mathbf{0} \\ \mathbf{0} & \mathbf{K}_1 \end{bmatrix}$

  $\mathbf{P}^T\mathbf{K}\mathbf{P}$ has the same eigenvalues as **K**, and therefore $\mathbf{K}_1$ must have all eigenvalues of **K** except λ_k

Gram-Schmidt Orthogonalisation

- other possibility: deflation of iteration vector

→ basis: the iteration vector must not be orthogonal the required eigenvector

→ conversely, if the iteration vector is orthogonalised to the eigenvectors allready calculated, convergence to these eigenvectors is eliminated

→ Gram-Schmidt method

Gram-Schmidt Orthogonalisation

eigenproblem: $K\varphi = \lambda M\varphi$

assuming that we have calculated the eigenvectors φ_1, φ_2, … , φ_m and that we want to **M**-orthogonalise x_1 to these eigenvectors

$$\tilde{\mathbf{x}}_1 = \mathbf{x}_1 - \sum_{i=1}^{m} \alpha_i \boldsymbol{\phi}_i$$

we obtain for the coefficients α_i

$$\alpha_i = \boldsymbol{\phi}_i^T \mathbf{M} \mathbf{x}_1; \qquad i = 1, \ldots, m$$

in inverse iteration $\widetilde{x}_1$ is now the starting iteration vector instead of x_1

Example 11.4, p. 898

eigenproblem: $K\varphi = \lambda M\varphi$ *tol = 10⁻⁶*

evaluating λ_4 and φ_4 using <u>forward</u> iteration

$$\mathbf{K} = \begin{bmatrix} 5 & -4 & 1 & 0 \\ -4 & 6 & -4 & 1 \\ 1 & -4 & 6 & -4 \\ 0 & 1 & -4 & 5 \end{bmatrix}; \qquad \mathbf{M} = \begin{bmatrix} 2 & & & \\ & 2 & & \\ & & 1 & \\ & & & 1 \end{bmatrix}$$

starting iteration vector:

$$\mathbf{x}_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Example 11.4, p. 898

k	$\bar{\mathbf{x}}_{k+1}$	$\bar{\mathbf{y}}_{k+1}$	$\rho(\bar{\mathbf{x}}_{k+1})$	$\mathbf{y}_{k+1}$	$\dfrac{\lvert\lambda_4^{(k+1)} - \lambda_4^{(k)}\rvert}{\lambda_4^{(k+1)}}$
1	1	6	5.93333	2.1909	—
	−0.5	−1		−0.3651	
	−1	−11		−4.0166	
	2	13.5		4.9295	
2	1.0954	2.1909	8.57887	0.3345	0.3084
	−0.1826	15.5188		2.3694	
	−4.0166	−41.9921		−6.4112	
	4.9295	40.5315		6.1882	
3	0.1672	−10.3137	10.15966	−1.1372	0.1556
	1.1847	38.2720		4.2198	
	−6.4112	−67.7914		−7.4745	
	6.1882	57.7704		6.3696	
8	−1.1285	−24.2083	10.63838	−2.2756	0.00003304
	2.7044	57.7298		5.4267	
	−7.7481	−82.4222		−7.7478	
	5.9969	63.6811		5.9861	
9	−1.1378	−24.2902	10.63844	−2.2833	0.000005584
	2.7133	57.8086		5.4340	
	−7.7478	−82.4224		−7.7476	
	5.9861	63.6351		5.9816	
10	−1.1416	−24.3237	10.63845	−2.2864	0.0000009437
	2.7170	57.8405		5.4369	
	−7.7476	−82.4219		−7.7476	
	5.9816	63.6157		5.9798	

Example 11.4, p. 898

$$\lambda_n \doteq \rho(\overline{\mathbf{x}}_{l+1}) \qquad \phi_n \doteq \frac{\overline{\mathbf{x}}_{l+1}}{(\overline{\mathbf{x}}_{l+1}^T \mathbf{y}_l)^{1/2}}$$

$$\lambda_4 \doteq 10.63845 \qquad \phi_4 \doteq \begin{bmatrix} -0.10731 \\ 0.25539 \\ -0.72827 \\ 0.56227 \end{bmatrix}$$

Example 11.5, p. 901

eigenproblem: $K\varphi = \lambda M\varphi$ *tol* = 10^{-6}

evaluating λ_1 and φ_1 using <u>inverse</u> iteration

after 3 iterations we get:

$$\lambda_1 \doteq 0.09654; \qquad \boldsymbol{\phi}_1 \doteq \begin{bmatrix} 0.3126 \\ 0.4955 \\ 0.4791 \\ 0.2898 \end{bmatrix}$$

Example 11.5, p. 901

Now imposing a shift of μ = *10*, we obtain:

$$\mathbf{K} - \mu\mathbf{M} = \begin{bmatrix} -15 & -4 & 1 & 0 \\ -4 & -14 & -4 & 1 \\ 1 & -4 & -4 & -4 \\ 0 & 1 & -4 & -5 \end{bmatrix}$$

Example 11.5, p. 901

Using inverse iteration on the problem (**K**-μ**M**)φ = η**M**φ, we obtain convergence after 6 iterations with

$$\rho(\bar{\mathbf{x}}_7) = 0.6385; \qquad \mathbf{x}_7 = \begin{bmatrix} -0.1076 \\ 0.2556 \\ -0.7283 \\ 0.5620 \end{bmatrix}$$

Example 11.5, p. 901

Using the shift, we know that $\mu + p(\bar{x}_7)$ is an approximation to an eigenvalue and x_7 is an approximation to the corresponding eigenvector

Comparing with the reults from 11.4, we find

$$\lambda_4 \doteq \mu + \rho(\mathbf{x}_7) \doteq 10.6385; \qquad \boldsymbol{\phi}_4 \doteq \mathbf{x}_7$$

Example 11.8, p. 908

Now we want to calculate an appropriate starting iteration vector, using Gram-Schmidt orthogonalisation:

$$\tilde{\mathbf{x}}_1 = \mathbf{x}_1 - \sum_{i=1}^{m} \alpha_i \boldsymbol{\phi}_i \quad \Longrightarrow \quad \tilde{\mathbf{x}}_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} - \alpha_1 \boldsymbol{\phi}_1 - \alpha_4 \boldsymbol{\phi}_4$$

$$\alpha_i = \boldsymbol{\phi}_i^T \mathbf{M} \mathbf{x}_1 \quad \Longrightarrow \quad \alpha_1 = \boldsymbol{\phi}_1^T \mathbf{M} \mathbf{x}_1$$

$$\alpha_4 = \boldsymbol{\phi}_4^T \mathbf{M} \mathbf{x}_1$$

Example 11.8, p. 908

Substituting for **M**, φ_1 and φ_4 leads to:

$$\alpha_1 = 2.385$$

$$\alpha_4 = 0.1299$$

and

$$\tilde{\mathbf{x}}_1 = \begin{bmatrix} 0.2683 \\ -0.2149 \\ -0.04812 \\ 0.2358 \end{bmatrix}$$