Diagonalization of matrix with iterative methods

Outline

- power method
- shift and inverse method
- Lanczos/Arnoldi method

Power method

Let's consider standard eigenvalue problem defined by hermitian matrix A which we wish to solve iteratively

$$A\vec{x}_k = \lambda_k \vec{x}_k, \qquad A \in \mathbb{C}^{n\times n}, \, \vec{x}_k \in \mathbb{C}^n, \lambda_k \in \mathbb{R}$$

$$A = A^H$$

all eigenvectors are orthogonal and form complete vector space

$$V = \{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n\}, \qquad \vec{x}_k^H \vec{x}_m = \delta_{k,m} \qquad \Longrightarrow \qquad \bigwedge_{\vec{v}_0 \in \mathbb{C}^n} \vec{v}_0 = \sum_{i=1}^{n} a_i \vec{x}_i$$

What happens to the vector **v₀** if it be multiplied by A many times?

$$\vec{v}_1 = A\vec{v}_0 = \sum_{i=1}^{n} a_i \lambda_i \vec{x}_i$$

$$\vec{v}_m = A^m \vec{v}_0 = \sum_{i=1}^{n} a_i \lambda_i^m \vec{x}_i$$

- each initial contribution a_i*x_i is scaled by (different?) eigenvalue
- coefficients in linear combination are changed every time the vector is multiplied by matrix
- after many multilication one may expect the eigenvectors with the largest eigenvalues will dominate while other shall be overwhelmed

Diagonalization with iterative methods

Let’s assume we may set eigenvalues in a sequence of decreasing values

$$|\lambda_1| > |\lambda_2| > |\lambda_3| > \ldots > |\lambda_n|$$

then we get

$$A\vec{v}_0 = \vec{v}_m = \lambda_1^m \left[a_1 \vec{x}_1 + \sum_{i=2}^{n} \left(\frac{\lambda_i}{\lambda_1} \right)^m a_i \vec{x}_i \right]$$

$$\lim_{m \to \infty} \left(\frac{\lambda_i}{\lambda_1} \right)^m = 0 \qquad \Longrightarrow \qquad \vec{v}_m \approx \lambda_1^m a_1 \vec{x}_1$$

$$\vec{x}_1 \approx \frac{\vec{v}_m}{\lambda_1^m a_1}$$

we don’t know a_1 and λ_1 but we need only normalized vector

$$\boxed{\vec{x}_1 \approx \frac{\vec{v}_m}{\|\vec{v}_m\|_2}}$$

Convergence in power method – Rayleigh quotient

To estimate the eigenvalue related with vector v_m let's first calculate scalar product

$$\vec{v}_m^H \vec{v}_m = \sum_{i=1}^{n} a_i \lambda_i^m \vec{x}_i^H \sum_{j=1}^{n} a_j \lambda_j^m \vec{x}_j$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j \lambda_i^m \lambda_j^m \underbrace{\vec{x}_i^H \vec{x}_j}_{\delta_{i,j}} = \sum_{i=1}^{n} a_i \lambda_i^{2m}$$

and A-scalar product

$$\vec{v}_m^H A \vec{v}_m = \sum_{i=1}^{n} a_i \lambda_i^{2m+1}$$

ratio of both is called **Rayleigh quotient** – it estimates the eigenvalue

$$\boxed{\frac{\vec{v}_m^H A \vec{v}_{m+1}}{\vec{v}_m^H \vec{v}_m} = \lambda_1 + O\left[\left(\frac{\lambda_i}{\lambda_1}\right)^{2m}\right]}$$

Power method allows iteratively find one eigenvector of largest eigenvalues.
But we need more vectors, how to find them?

In order to find another eigenvector we need to restart the algorithm and iterate again, however if we change nothing in it we again get the same eigenvector – it dominates the eigenvalue spectrum.

The remedy for this problem is simple - remove contribution of x_1 from the vector v_m with Gram-Schmidt method.

This must be done in every iteration because numerical errors will generate small but non-zero contributions along removed directions.

This strategy shall be applied to any subsequent eigenvector, so we may construct general projector that removes contributions from previously found (k-1) vectors

$$\boxed{P_k = I - \sum_{p=1}^{k-1} \vec{x}_p \vec{x}_p^H}$$

Gram-Schmidt orthogonalization is recommended for large sparse matrix problems.
If we operate on small matrices we may utilize the spectral form of matrix and remove information about these vectors i.e. reduce the rank of matrix one by one (Hotelling method)

$$A_1 = A = \sum_{i=1}^{n} \lambda_i^n \vec{x}_i \vec{x}_i^H$$

$$\vec{v}_0 \quad \rightarrow \quad \vec{v}_m = A_1^m v_0 \quad \rightarrow \quad \left\{ \vec{x}_1 = \frac{\vec{v}_m}{\|\vec{v}_m\|_2}, \lambda_1 = \frac{\vec{v}_m^H A_1 \vec{v}_m}{\vec{v}_m^H \vec{v}_m} \right\}$$

$$A_2 = A_1 - \lambda_1 \vec{x}_1 \vec{x}_1^H$$

$$\vec{v}_0 \quad \rightarrow \quad \vec{v}_m = A_2^m v_0 \quad \rightarrow \quad \left\{ \vec{x}_2 = \frac{\vec{v}_m}{\|\vec{v}_m\|_2}, \lambda_2 = \frac{\vec{v}_m^H A_2 \vec{v}_m}{\vec{v}_m^H \vec{v}_m} \right\}$$

$$A_3 = A_2 - \lambda_2 \vec{x}_2 \vec{x}_2^H$$

$$\vdots$$

Algorithm of power method with Gram-Schmidt orthogonalization

```
input: A, $\varepsilon$, $K_{val}$, ITMAX
for k=1 to $K_{val}$ by 1 do
    !initialize: $\vec{v} \leftarrow$ random vector, $\lambda_{old} \leftarrow$ large number
    for m=1 to ITMAX by 1 do
        !update vector and calculate Rayleigh quotient
            $\vec{v} \leftarrow A\vec{v}$
            $\lambda \leftarrow \frac{\vec{v}^H A\vec{v}}{\vec{v}^H \vec{v}}$
        !perform Gram–Schmidt orthogonalization
            $\vec{V} \leftarrow \left(I - \sum_{i=1}^{m-1} \vec{x}_i \vec{x}_i^H\right) \vec{v}$
        !normalize vector
            $\vec{v} \leftarrow \frac{\vec{v}}{\|\vec{v}\|_2}$
        !check convergence
        if $\left|\frac{\lambda - \lambda_{old}}{\lambda_{old}}\right| < \varepsilon$ then
            break
        end if
        !save $\lambda$ for next iteration
            $\lambda_{old} \leftarrow \lambda$
    end do
    if $m <$ ITMAX then
        save: $\lambda_k \leftarrow \lambda$, $\vec{x}_k \leftarrow \vec{v}$
    else
        STOP – lack of convergence of k–eigenvalue
    end if
end do
```

Shift and inverse method

Power method as well as other iterative schemes find the eigenvectors pinned to eigenvalues from top part of matrix spectrum but e.g. in physcis problems we often need eigenvalues from the bottom part.
The use of basic algorithm of iterative method would then prohibit this task.
In order to find the solution let's consider general EVP

First, assume the point of interest in eigenvalue spectrum – we will seek eigenvalues nearest to this point

$$\sigma \in \mathbb{C}$$

then shift both sides by **σBx**

$$A\vec{x} = \lambda B\vec{x} \quad /(-\sigma\, B\vec{x})$$

for standard EVP:

$$B = I$$

and transform GEVP to EVP form

$$(A - \sigma B)\vec{x} = (\lambda - \sigma)B\vec{x}, \qquad (A - \sigma)^{-1} \cdot /$$

$$\vec{x} = (\lambda - \sigma)(A - \sigma B)^{-1}B\vec{x} \qquad / \cdot \frac{1}{\lambda - \sigma}$$

$$\underbrace{\frac{1}{\lambda - \sigma}}_{\mu}\vec{x} = \underbrace{(A - \sigma B)^{-1}B}_{C}\,\vec{x}$$

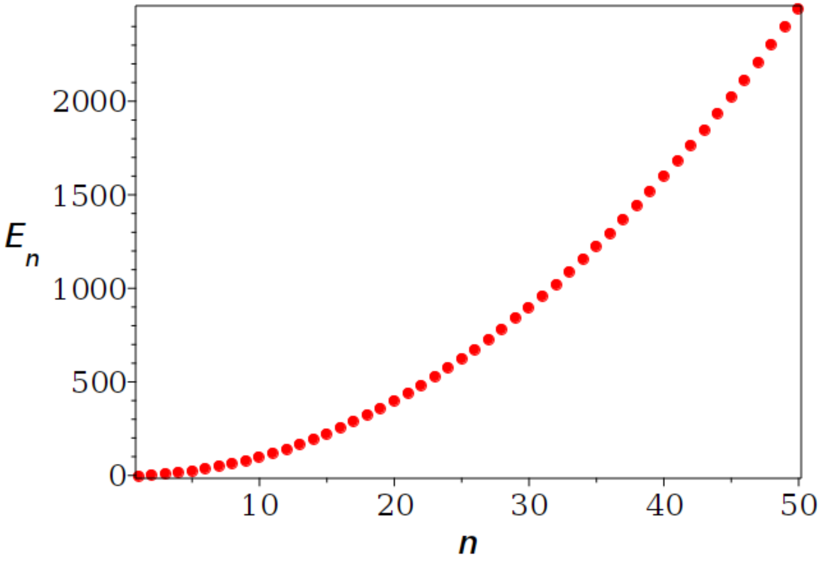
$$\boxed{C\vec{x} = \mu\vec{x}}$$

we change the matrix and its eigenvalues but eigenvectors remain the same

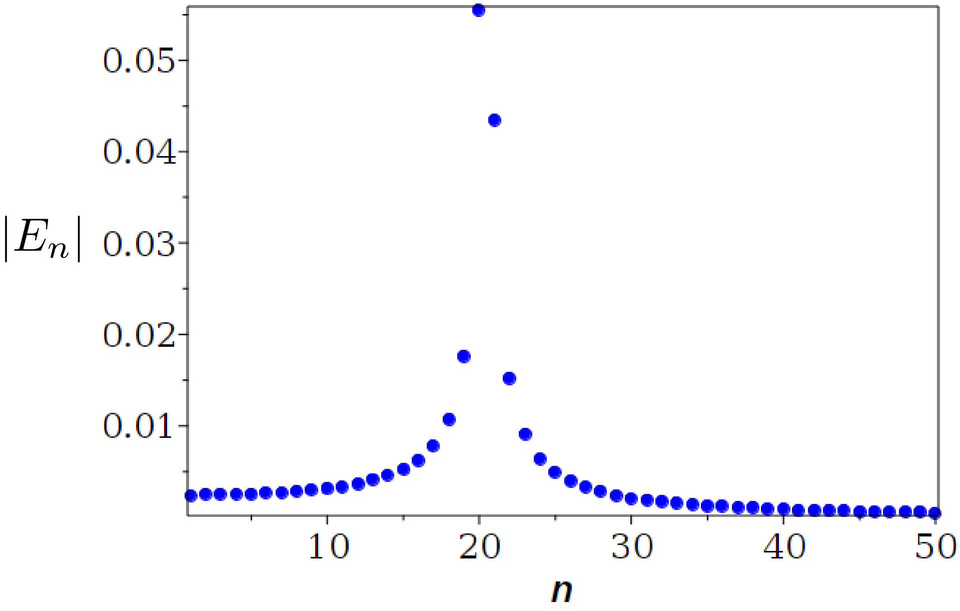
Example: transformation of eigenspectrum

$$E_n = n^2 - 3$$

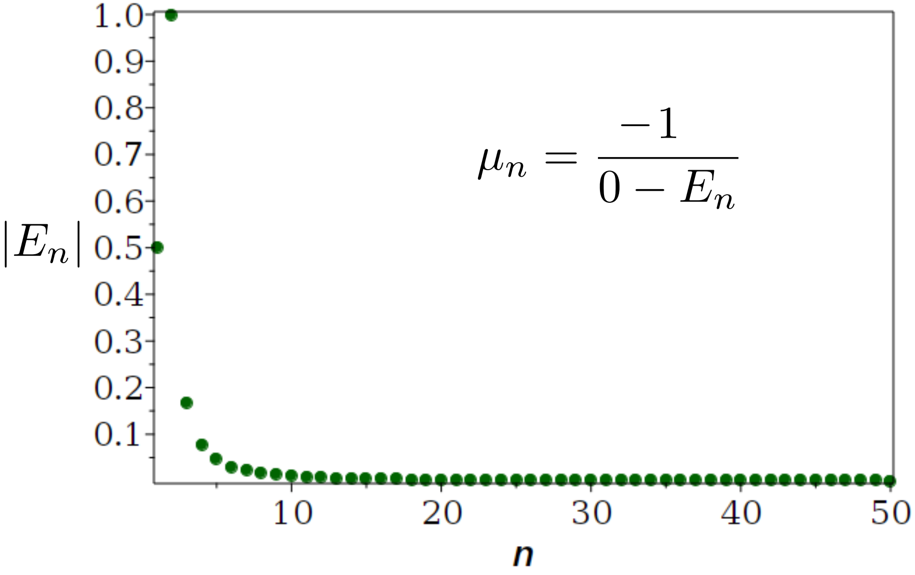
original eigenspectrum



$E_n = n^2 - 3$



$$\mu_n = \frac{-1}{415 - E_n}$$



$$\mu_n = \frac{-1}{0 - E_n}$$

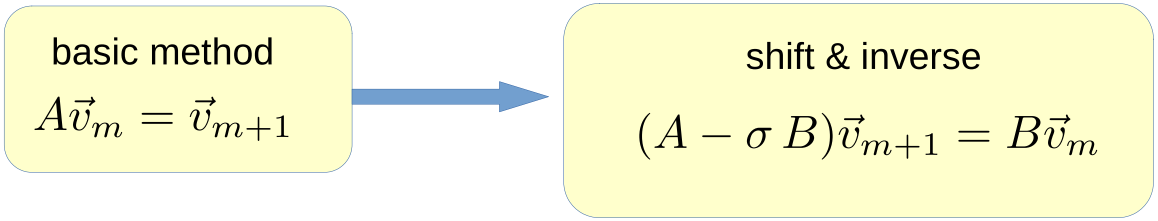
$$\mu_n = +\frac{1}{E_n}$$

Diagonalization with iterative methods

With shift-inverse modification we may use power method (and any other one) to find eigenvectors in interesting part of eigenvalue spectrum.

In each iteration we need to solve system of linear equations (SLE), this can be done by means of one of decomposition methods (LU, LL^T, LDL^T) or iteratively (SOR, CG, etc.)

$$C\vec{v}_m = (A - \sigma)^{-1} B\vec{v}_m = \vec{v}_{m+1}$$



basic method

$$A\vec{v}_m = \vec{v}_{m+1}$$

shift & inverse

$$(A - \sigma\, B)\vec{v}_{m+1} = B\vec{v}_m$$

Once we find the eigenvector v_m → x_m and eigenvalue μ_m the latter must be transformed back

$$\mu = \frac{1}{\lambda - \sigma} \qquad \Longrightarrow \qquad \lambda = \sigma + \frac{1}{\mu}$$

Remark: power method is simple but very inefficient, many iterations are needed to refine single eigenvector from unwanted contributions, moreover, calculations of subsequent eigenvalues requires conducting Gram-Schmidt orthogonalization in every iteration what may significantly elongate the time of computations

Lanczos method

- this method is used get the partial diagonalization of large sparse hermitian matrix
- firstly, the sparse matrix is iteratively aproximated by two other matrices:
  orthogonal Q and tridiagonal T
- secondly, tridiagonal matrix is diagonalized with direct methods (e.g. QR decomposition) and eigenvectors for sparse matrix are reconstructed

The method utilizes the properties of **Krylov subspace** generated with vector **q** and matrix **A**

$$\begin{array}{rcl} \vec{q} & \in & \mathbb{C}^n \\ K_m(\vec{q}, A) & = & span[\vec{q}, A\vec{q}, \ldots, A^{m-1}\vec{q}] \\ m & \geq & 1 \\ K_0(\vec{q}, A) & = & \{\vec{0}\} \\ dim(K_m) & = & m \end{array}$$

we demand that the Krylov subspace basis vectors

$$K_m = span[\vec{q}_1, \vec{q}_2, \ldots, \vec{q}_m]$$

are orthogonal and normalized

$$\vec{q}_i^{\,H} \vec{q}_j = \delta_{ij} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

Algorithm of Lanczos method

- method starts by chosing any non-zero vector **q₁**,

  → must include contributions from the eigenvectors we are looking for,
  it is recommended to fill its elements with random numbers of Gauss distribution

$$\vec{q}_1 \in \mathbb{C}^n, \quad \vec{q}_1 \neq \vec{0}$$

- next we use the three-term recurrence formula for finding subsequent vectors

$$\boxed{A\vec{q}_i = \vec{y} = \gamma_i \vec{q}_{i-1} + \delta_i \vec{q}_i + \gamma_{i+1}\vec{q}_{i+1} \quad i \geq 1}$$

at start we assume

$$\gamma_1 \vec{q}_0 = \vec{0}$$

at start have knowledge about **q_1** and **q_0**=0 hence we may calculate third vector **q_2** and so on in next iterations

$$\vec{q}_i^{\,H} \cdot / \ldots \quad \Longrightarrow \quad \vec{q}_i^{\,H} A\vec{q}_i = \gamma_i \underbrace{\vec{q}_i^{\,H} \vec{q}_{i-1}}_{=0} + \delta_i \underbrace{\vec{q}_i^{\,H} \vec{q}_i}_{=1} + \gamma_{i+1} \underbrace{\vec{q}_i^{\,H} \vec{q}_{i+1}}_{=0} \quad \Longrightarrow \quad \delta_i = \vec{q}_i^{\,H} A\vec{q}_i$$

$$\vec{r}_{i+1} = \gamma_{i+1}\vec{q}_{i+1} = A\vec{q}_i - \delta_i \vec{q}_i - \gamma_i \vec{q}_{i-1} \quad \Longrightarrow \quad \vec{q}_{i+1} = \frac{\vec{r}_{i+1}}{\|\vec{r}_{i+1}\|_2}$$

$$\gamma_{i+1} = \vec{q}_{i+1}^{\,H} \vec{r}_{i+1}$$

- iterative process would stop if we encounter $\gamma_{i+1} = 0$

  in such case algorithm stops and must be restarted with new random vector **q₁**

Generated vectors **q** are used to form orthogonal matrix **Q$_m$**

$$\mathbb{C}^{n\times m} \ni Q_m = [\vec{q}_1, \ldots, \vec{q}_m] \qquad Q_m^H Q_m = I$$

while factors δ and γ are elements of tridiagonal matrix

$$T_m = \begin{bmatrix} \delta_1 & \gamma_2^* & & 0 \\ \gamma_2 & \delta_2 & \ddots & \\ & \ddots & \ddots & \gamma_m^* \\ 0 & & \gamma_m & \delta_m \end{bmatrix}$$

Actually, matrix Q transforms the matrix A to „compact tridiagonal”

$$\mathrm{AQ}_m = Q_m T_m \qquad (\text{''} + \text{''} \quad \gamma_{m+1})$$

$$\mathrm{Q}_m^H A Q_m = T_m$$

Eigenvalues of T approximates the largest eigenvalues of matrix A.

Diagonalization with iterative methods

After diagonalization of T we get approximate eigenvalues of A but we must transform back the eigenvectors

$$T_m \vec{z} = \lambda \vec{z}$$

$$\vec{x} = Q_m \vec{z}$$

$$A\vec{x} = A Q_m \boldsymbol{z} = Q_m T_m \vec{z} = \lambda Q_m \vec{z} = \lambda \vec{x}$$

Remarks:

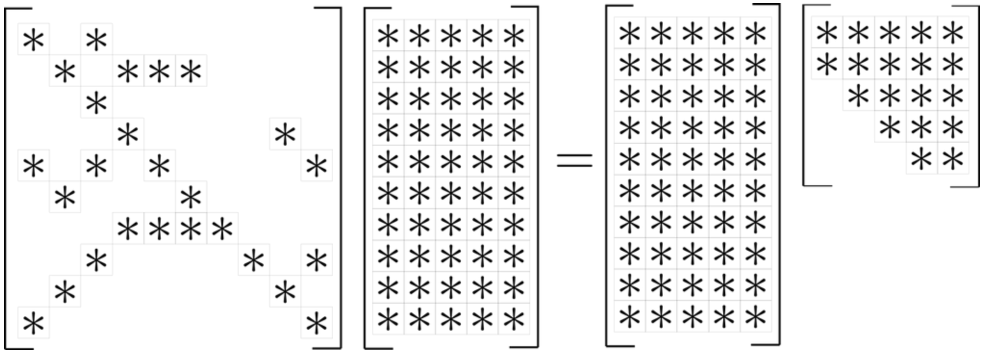
- only a fraction of m eigenvalues will satisfactorily approximate these of A, so when we are looking for N eigenvalues we must assume m>N
- if we are interested in an inner part of eigenvalues spectra, we may easily employ an shift and inverse modification, only the routine for solving SLE is needed
- and the worst information: because we operate on floating-point numbers the vectors **q** are quickly becomes perturbed due to numerical errors and becomes non-orthogonal, therefore expensive Gram-Schmidt orthogonalization is essential for Lanczos method to work

$$\tilde{\vec{q}}_{i+1} \quad \rightarrow \quad \vec{q}_{i+1} = \left(I - \sum_{j=1}^{i} \vec{q}_j \vec{q}_j^{\,H} \right) \tilde{\vec{q}}_{i+1}$$

Arnoldi method

Lanczos method is intended for diagonalization of Hermitian matrices, for diagonalization of sparse general matrix we may use Arnoldi method. It is also iterative method that generates set of orthogonal vectors q which form matrix Q but the second matrix containing transformation factors is Hessenberg matrix, which is then diagonalized.

$$AQ_m = Q_m \mathcal{H}_m$$



Remarks:

- Arnoldi method can be used also for hermitian matrices
- it requires Gram-Schmidt orthogonalization
- it is more stable than Lanczos method, and has the same efficency
- the numerical package **ARPACK** containing Arnoldi method seems to be a standard in diagonalization of large sparse matrices (written in Fortran, available in C/C++)

http://www.ime.unicamp.br/~chico/arpack++/