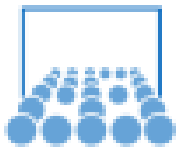
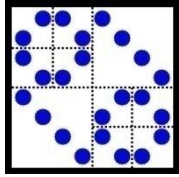


8.3 Divide & Conquer for tridiagonal $A=A^T$



A divide and conquer approach for computing eigenvalues of a symmetric tridiagonal matrix T.

$$T = \begin{pmatrix} a_1 & b_1 & & \\ b_1 & a_2 & & \\ & & \ddots & b_{n-1} \\ & & b_{n-1} & a_n \end{pmatrix}$$

Idea: Split T in two tridiagonal matrices T_1 and T_2.
Compute eigenvalues of T_1 and T_2.
Recover the original eigenvalues of T as perturbations.

Repeat recursively/parallel.

Splitting of T

Set $v := (0 \quad \cdots\cdots \quad 0 \quad 1 \mid \theta \quad 0 \quad \cdots\cdots \quad 0)^T$

$$\tilde{T} := T - \rho v v^T$$ Rank-1 perturbation of T

Aim: Generate zeros at the sub/superdiagonal entries in the middle of T

$$\tilde{T}(m:m+1, m:m+1) = \begin{pmatrix} a_m & b_m \\ b_m & a_{m+1} \end{pmatrix} - \rho \begin{pmatrix} 1 & \theta \\ \theta & \theta^2 \end{pmatrix} =$$

$$= \begin{pmatrix} a_m - \rho & b_m - \rho\theta \\ b_m - \rho\theta & a_{m+1} - \rho\theta^2 \end{pmatrix} \overset{!}{=} \begin{pmatrix} * & 0 \\ 0 & * \end{pmatrix}$$

$$\rho\theta = b_m \longrightarrow T = \begin{pmatrix} T_1 & 0 \\ 0 & T_2 \end{pmatrix} + \rho v v^T$$

Relation between T and T_1, T_2

Assume, that we know the eigenvalues and eigenvectors of T_1 and T_2 .

How can we get the eigenpairs of T?

$$T_1 = U_1 \Lambda_1 U_1^T, \qquad T_2 = U_2 \Lambda_2 U_2^T, \quad \overset{?}{\Rightarrow} \quad T = U \Lambda U^T$$

Note, that T is a rank-1 perturbation of diag(T_1 , T_2).

Recover the original eigenvalues as perturbations of eigenvalues of T_1 and T_2 .

$$T = \begin{pmatrix} T_1 & 0 \\ 0 & T_2 \end{pmatrix} + \rho v v^T = \begin{pmatrix} U_1 \Lambda_1 U_1^T & 0 \\ 0 & U_2 \Lambda_2 U_2^T \end{pmatrix} + \rho v v^T = \begin{pmatrix} U_1 & 0 \\ 0 & U_2 \end{pmatrix} \begin{pmatrix} \Lambda_1 & 0 \\ 0 & \Lambda_2 \end{pmatrix} \begin{pmatrix} U_1^T & 0 \\ 0 & U_2^T \end{pmatrix} + \rho v v^T$$

$$\Longrightarrow \quad \begin{pmatrix} U_1^T & 0 \\ 0 & U_2^T \end{pmatrix} T \begin{pmatrix} U_1 & 0 \\ 0 & U_2 \end{pmatrix} = \begin{pmatrix} \Lambda_1 & 0 \\ 0 & \Lambda_2 \end{pmatrix} + \rho \tilde{v} \tilde{v}^T$$

Computing the eigenvector

Hence, we need to compute the eigenvalues of a matrix

of the form "diagonal + rank-1": $D + \rho \tilde{v} \tilde{v}^T$

Let λ_i and u_i be an eigenpair of $D + \rho v v^T$. Then it holds

$$\left(D + \rho \tilde{v} \tilde{v}^T\right) u_i = \lambda_i u_i \Leftrightarrow \left(D - \lambda_i I\right) u_i + \rho \left(\tilde{v}^T u_i\right) \tilde{v} = 0$$

$$u_i = const \cdot \left(D - \lambda_i I\right)^{-1} \tilde{v}$$

Hence, if we know λ_i, then we easily get the eigenvector u_i .

Eigenvalues as Zeros

Furthermore, we get the equation

$$\tilde{v}^T \left(D - \lambda_i I\right)^{-1} \cdot \left[\left(D - \lambda_i I\right) u_i + \rho\left(\tilde{v}^T u_i\right) \tilde{v}\right] = 0$$

$$\tilde{v}^T u_i + \rho\left[\tilde{v}^T \left(D - \lambda_i I\right)^{-1} \tilde{v}\right] \cdot \left[\tilde{v}^T u_i\right] = 0$$

$$f(\lambda) = 1 + \rho\left[\tilde{v}^T \left(D - \lambda I\right)^{-1} \tilde{v}\right] = 1 + \rho\left(\frac{\tilde{v}_1^2}{d_1 - \lambda} + \cdots + \frac{\tilde{v}_n^2}{d_n - \lambda}\right) = 0$$

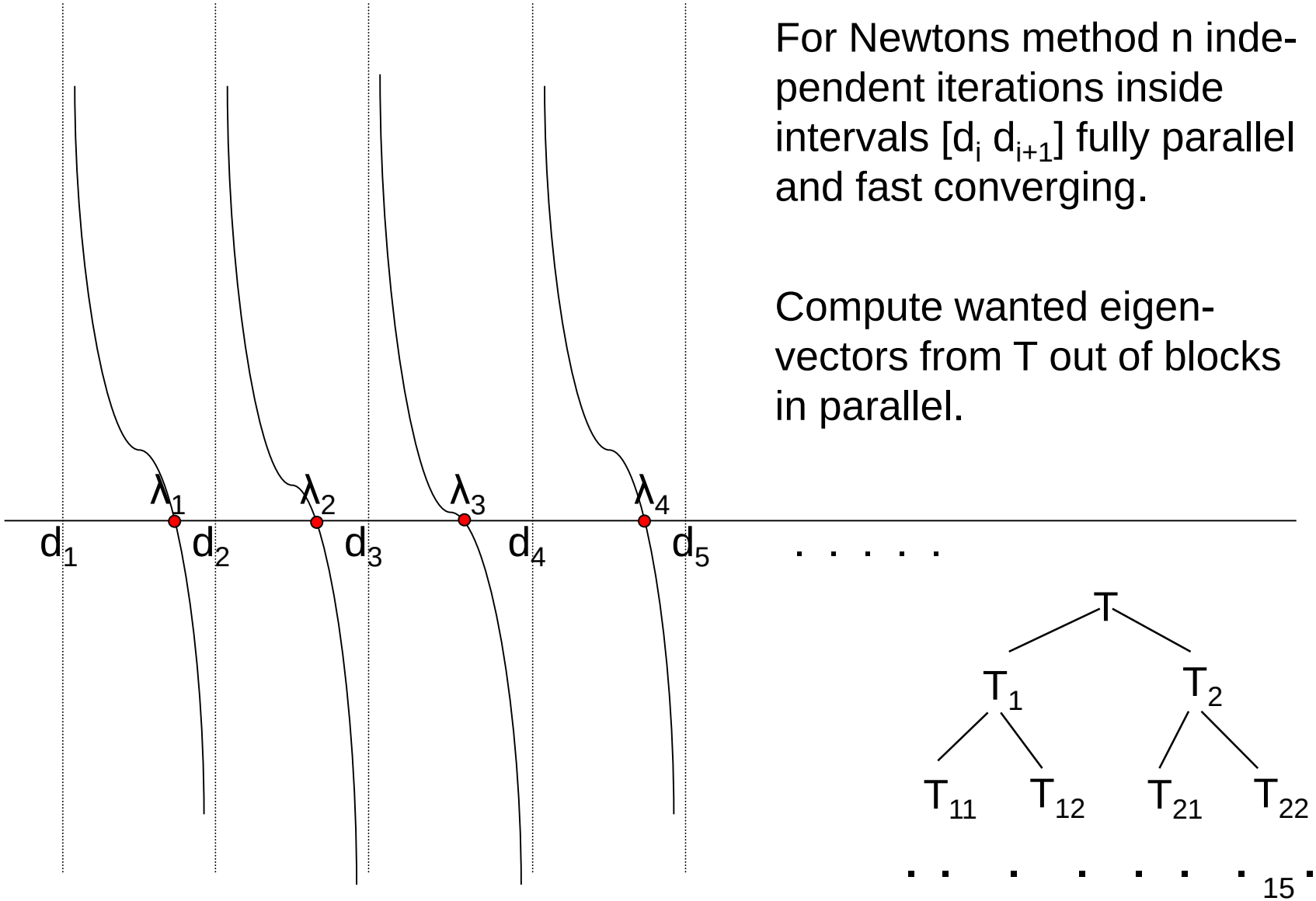
Use Newton's method, to determine the zeroes of function f(λ)

These zeroes are the eigenvalues of $D + \rho \tilde{v} \tilde{v}^T$

and therefore also of T.

Repeat recursively for T_1 and T_2 .

Zeros and poles of f(λ):



For Newtons method n independent iterations inside intervals $[d_i\ d_{i+1}]$ fully parallel and fast converging.

Compute wanted eigenvectors from T out of blocks in parallel.

8.4 Algorithms for computing a few eigenpairs:

Vector iteration:

$$x^{(k)} = A^k x^{(0)} \Big/ \left\| A^k x^{(0)} \right\| \to v$$

eigenvector to eigenvalue with maximum absolute value!

Easy to parallelize (only Ax), but slow convergence! Only λ_{max}!
Compare Krylov!

Subspace Iteration: Apply the same idea to set of vectors $U^{(0)}=(x^{(0)},\ldots,x^{(m)})$
For k=0,1,..: Replace $U^{(k)}$ by $AU^{(k)}$. Orthogonalize $A\,U^{(k)} \rightarrow U^{(k+1)}$.
Eigenvectors/values of $U^{(k)T}A\,U^{(k)}$.
Also possible: Increase U upto certain limit. Then restart with one vector.

Inverse iteration:

Apply vector iteration on shifted problem $(A - \sigma I)^{-1}$
for computing the eigenvector nearest to σ.

Expensive! Ill-conditioned linear system!

Rayleigh Quotient Iteration

Rayleigh Quotient iteration: Start with vector y and real $\rho = y^T Ay / y^T y$ and repeat:

$$v = y/\|y\|; \qquad y_{new} = (A - \rho I)^{-1} v; \qquad \rho = \rho + y_{new}^T v / \|y_{new}\|^2 ;= \frac{y_{new}^T A y_{new}}{y_{new}^T y_{new}}$$

Inverse Iteration with replacing the shift σ by the newest eigenvalue estimate.

y: new eigenvector estimate → leads to new eigenvalue estimate:

$$\rho_{new} = \frac{y_{new}^T A y_{new}}{y_{new}^T y_{new}} = \frac{y_{new}^T (A - \rho I + \rho I) y_{new}}{y_{new}^T y_{new}} = \frac{y_{new}^T (A - \rho I) y_{new}}{y_{new}^T y_{new}} + \rho = \frac{y_{new}^T v}{\|y_{new}\|^2} + \rho$$

Fast convergence, but uncertain to which eigenvalue we will converge.
Expensive! Ill-conditioned!

8.5 Arnoldi (Lanczos) for sparse A

Use the transformation on Hessenberg (tridiagonal) form described for GMRES.
Compute the eigenvalues of the small Hessenberg matrix and use them as approximations for the eigenvalues of the original matrix.

By Arnoldi Orthogonalization of the Krylov subspace (b,Ab,A^2b,…) we get the relation

$$Au_{j-1} = \sum_{k=1}^{j-1} h_{k,j-1} u_k + \tilde{u}_j = \sum_{k=1}^{j} h_{k,j-1} u_k$$

$$A \cdot U_m = A\begin{pmatrix} u_1 & \dots & u_m \end{pmatrix} = \begin{pmatrix} u_1 & \dots & u_{m+1} \end{pmatrix} \cdot \tilde{H}_{m+1,m} = U_m H_{m,m} + h_{m+1,m} u_{m+1}$$

Eigenvalues of $H_{m,m}$ as approximations for A.
(Small $h_{m+1,m}$ → good approximation).
Good approximation for extreme eigenvalues.
For symmetric A, H is tridiagonal.

The same approach can be applied on: f(A)b ~ Uf(H)U^Tb

8.6 Jacobi-Davidson for sparse A

Idea: - No Krylov subspace, more related to Rayleigh quotient and subspace it.
- choose subspace relative to eigenvalue we are looking for
- include preconditioning;

Starting point:
Consider eigenvalue approximations derived by $V_m^H A V_m$
for subspace relative to V_m.
The eigenpairs of $V_m^H A V_m$ are used as approximations to eigenvalues of A

How to choose new subspace V_{m+1} with additional vector u such that the new approximation for special eigenvalues is strongly improved?

For first eigenpair approximation u_m and $t_m=(u_m^H A u_m)/(u_m^H u_m)$, we try to improve these approximations by small corrections u and t to get better estimates u_m+u and t_m+t

$$A(u_m+u)=(t_m+t)(u_m+u), \qquad u \perp u_m,\ u_m^H u_m = 1$$

Jacobi-Davidson II

$$A(u_m + u) = (t_m + t)(u_m + u), \qquad u \perp u_m$$

$$\Downarrow$$

$$(A - t_m I)u = tu_m - (A - t_m I)u_m + \cancel{tu}$$

ignore correction t u of second order

Use orthogonal projection with $I - u_m u_m^H$ from the left.
This leads to

$$(I - u_m u_m^H)(A - t_m I)(I - u_m u_m^H)\, u = -(I - u_m u_m^H)(A - t_m I)u_m$$

$$(I - u_m u_m^H)(A - t_m I)(I - u_m u_m^H)\, u = -(A - t_m I)u_m + u_m \cancel{\left(u_m^H A u_m\right) - t_m u_m \left(u_m^H u_m\right)}$$

$= 0$

Jacobi-Davidson III

For new approximation we have to solve

$$\left(I - u_m u_m^H\right)\left(A - t_m I\right)\left(I - u_m u_m^H\right) u = -\left(A - t_m I\right) u_m$$

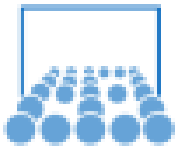
$$P\left(A - t_m I\right) P u = -r_m \qquad or \qquad \tilde{A} u = -r_m$$

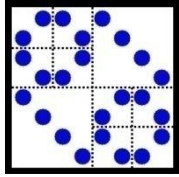
$A - t_m I$ gets ill-conditioned for t_m near eigenvalue, but P is a projection orthogonal to the near singular vector!

$\tilde{A}$ is singular, but linear system is still solvable.

Compared to Inverse Iteration/RQI:

Replace ill-conditioned by singular system.





Jacobi-Davidson IV

New eigenvector estimate $u_m+u \rightarrow v_{m+1}$ also leads to new eigenvalue estimate $t_{m+1} \rightarrow (v_{m+1}^H A v_{m+1})/(v_{m+1}^H v_{m+1})$.

Choose the new estimate v_{m+1} to enlarge the subspace V_m by the new vector u to V_{m+1}.

Compute eigenpairs of $V_{m+1}^H A V_{m+1}$ and choose next eigenvector approximation u_{m+1} appropriately, e.g. maximum, minimum, close to σ.

Repeat this step a few times.

Restart the whole process with last best approximation as starting vector u_1 , resp. 1-dim subspace V_1.

Advantages: Allows to compute also inner eigenvalues without solving more and more ill-conditioned problems like Rayleigh QI.

Jacobi-Davidson V

Main step: Solve linear system

$$P\left(A - t_m I\right)Pu = -r_m \qquad or \qquad P(A - t_m I)\tilde{u} = -r_m$$

approximately.

Therefore, we use a few steps of preconditioned cg or GMRES.

Preconditioner: M^{-1} preconditioner for A → $PM^{-1}P$ preconditioner for PAP

In each iteration step we have to multiply with A, with P, and solve in M.

Simple preconditioner: M = diag(A)
Better preconditioner: SPAI or MSPAI, ILU

8.7 Bisection for computing eigenvalues of a tridiagonal matrix

Observation: The characteristic polynomial of a tridiagonal matrix can be evaluated via the matrix entries in form of a sequence of polynomials with increasing degree:

$$p(\lambda) = \det(T - \lambda I) = \det \begin{pmatrix} \delta_1 - \lambda & \gamma_2 & 0 & \cdots & 0 \\ \gamma_2 & \delta_2 - \lambda & \gamma_3 & & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & \gamma_{n-1} & \delta_{n-1} - \lambda & \gamma_n \\ 0 & \cdots & 0 & \gamma_n & \delta_n - \lambda \end{pmatrix}$$

$$p_0(\lambda) = 1$$

$$p_1(\lambda) = \delta_1 - \lambda$$

$$p_2(\lambda) = (\delta_2 - \lambda)p_1(\lambda) - \gamma_2^2 p_0(\lambda)$$

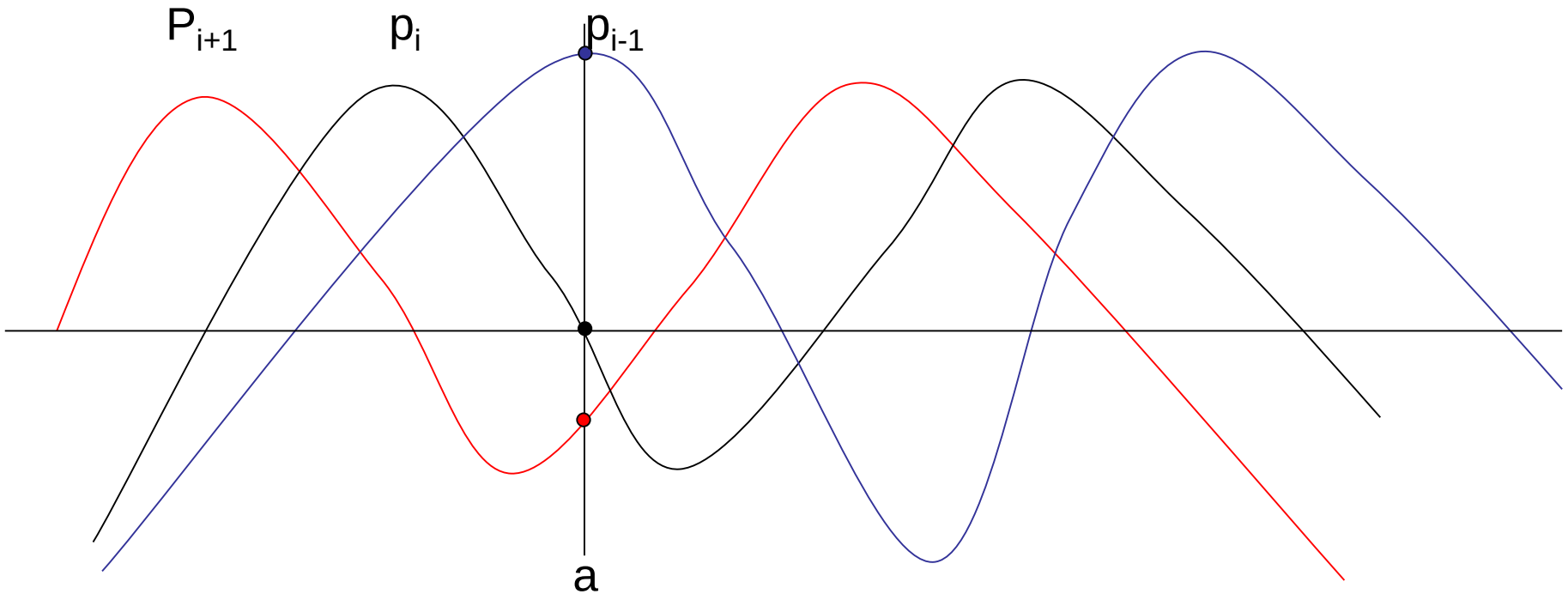
$$p_i(\lambda) = (\delta_i - \lambda)p_{i-1}(\lambda) - \gamma_i^2 p_{i-2}(\lambda), \qquad i = 3,4,\ldots,n$$

$$p(\lambda) = p_n(\lambda)$$

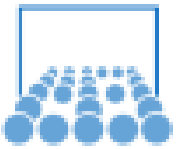
The sequence of polynomials is a Sturm chain:

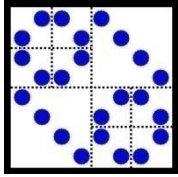
1. All p_i have only single zeros
2. $\text{sign}(p_{n-1}(a)) = -\text{sign}(p'_n(a))$ for all real zeros a of $p_n(x)$
3. For i=1,2,…,n-1: $p_{i+1}(a)p_{i-1}(a) < 0$ for all real zeros a of $p_i(x)$
4. The polynomial $p_0(x)$ does not change ist sign

Proof by induction.



At all zeros of p_i the neighbors p_{i-1} and p_{i+1} must have different sign.



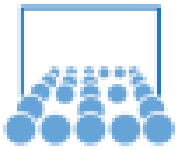


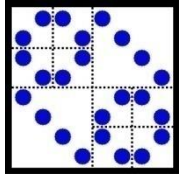
Define w(a):= # sign changes in $p_i(a)$, i=1,…,n .

It holds: w(a) = # zeros of $p_n(x)$ for x<a.

Consider eigenvalues ordered $\lambda_1 < \lambda_2 < \ldots < \lambda_{n-1} < \lambda_n$.
We want to find λ_i, the i-th zero of $p_n(x)$.

It holds: $\lambda_i < a \rightarrow$ w(a) = [# zeros left of a] >= i





Bisection Algorithm:

Choose an interval I=[a_0 ,b_0] which contains λ_i.
Therefore: w(b_0) >= i and w(a_0) < i.
Evaluate the polynomial sequence for a=(a_0+b_0)/2 and count the sign changes in the sequence $p_i(a)$ → w(a).

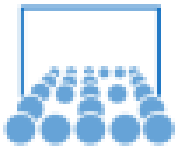
If w(a) >= i: Replace in interval I b_0 by a
Otherwise: Replace in interval I a_0 by a.

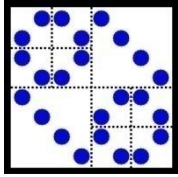
Generates converging sequence of smaller and smaller intervals that contain the eigenvalue λ_i certainly.

Advantages:

- can be easily parallelized on top level
- can be used with high or low accuracy
- gives only eigenvalues

Allows also Newton method for zeros of p_n





8.8 MRRR for eigenvectors

Multiple Relatively Robust Representations

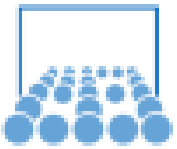
Idea: Given tridiagonal matrix.
Assume the eigenvalues to be computed fast in parallel e.g. via Bisection to high accuracy.

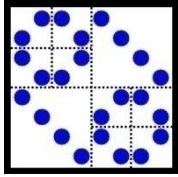
Use inverse iteration for computing the eigenvectors to high accuracy.

Observations:

Inverse iteration is cheap, because of tridiagonal form! (??)

But eigenvalue cluster lead to low accuracy in eigenvectors!





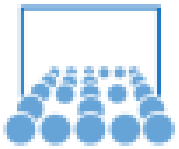
Outline

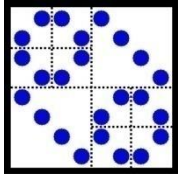
1. Choose μ such that $T + \mu I$ is positive definite.
2. Compute Cholesky factorization $T + \mu I = L D L^T$.
3. Compute eigenvalues of $L D L^T$ to high relative accuracy (Bisection)
4. Group eigenvalues according to their relative gap

   a) Isolated eigenvalue: Compute eigenvector via twisted factorization and Inverse Iteration

   b) Cluster:

   - Choose μ near cluster and compute $L D L^T - \mu I = L_1 D_1 L_1^T$.
   - Refine eigenvalues in cluster to high relative accuracy
   - Set $L \leftarrow L_1$, $D \leftarrow D_1$.
   - Repeat step 4 for eigenvalues in this cluster.





Twisted factorization: $T - \hat{\lambda} I = N_r D_r N_r^T$

$$N_r = \begin{pmatrix} x & & & & & & \\ x & x & & & & & \\ & . & . & & & & \\ & & x & \gamma_r & x & & \\ & & & & . & . & \\ & & & & & x & x \\ & & & & & & x \end{pmatrix}$$

r is chosen to minimize $|\gamma_r|$.
Guarantuees fast convergence of Inverse Iteration.

Advantages of twisted factorization?

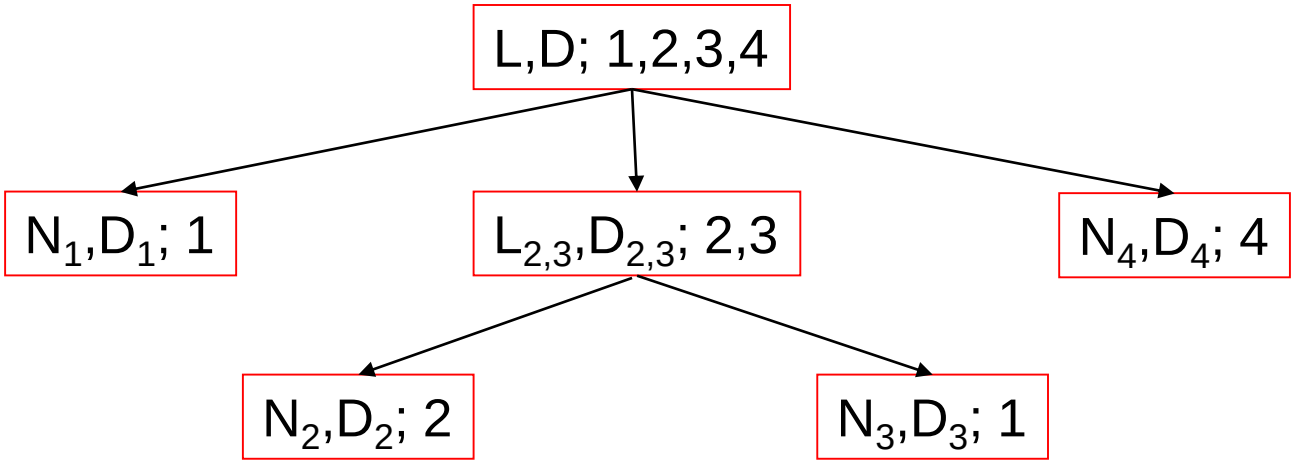
Conclusion

MR^3 allows the computation of eigenvectors with high accuracy

(also for clustered eigenvalues)

using only small number of inverse iteration steps.

Results in tree relative to clusters:



Use fixed point iteration (ILU) for Cholesky and twisted systems.

8.9 Sequential QR Algorithm for computing all Eigenvalues:

Standard algorithm for computing eigenpairs: QR-algorithm

Prestep: Transform A by Givens or Householder matrices to tridiagonal form.

$$G_{2,3} * \left(\begin{array}{ccc|cc} a_{11} & a_{12} & \boxed{a_{13}} & * & * \\ \hline a_{21} & a_{22} & a_{23} & * & * \\ \boxed{a_{31}} & a_{32} & a_{33} & * & * \\ \hline * & * & * & * & * \\ * & * & * & * & * \end{array}\right) * G_{2,3}^{H}$$

to eliminate a_{31} and a_{13}

Main difference to QR-factorization:

- Use subdiagonal entry for eliminating elements
- Apply Q from both sides
- Gives tridiagonal matrix (or upper Hessenberg for nonsymymetric A) .

For better parallelism use block Householder like in the QR-decomposition.

QR-Algorithm

First step:

By Householder matrices transform A by equivalence transformations on tridiagonal (upper Hessenberg) form: $A \rightarrow H \cdot A \cdot H^T = T$

For the following we assume A already tridiagonal (upper Hessenberg)

Second step: Compute QR-decomposition of A, $A = QR$ and

replace $A = A_{old}$ by $A_{new} = RQ$

$$A_{new} = RQ = (Q^T A)Q = Q^T AQ$$

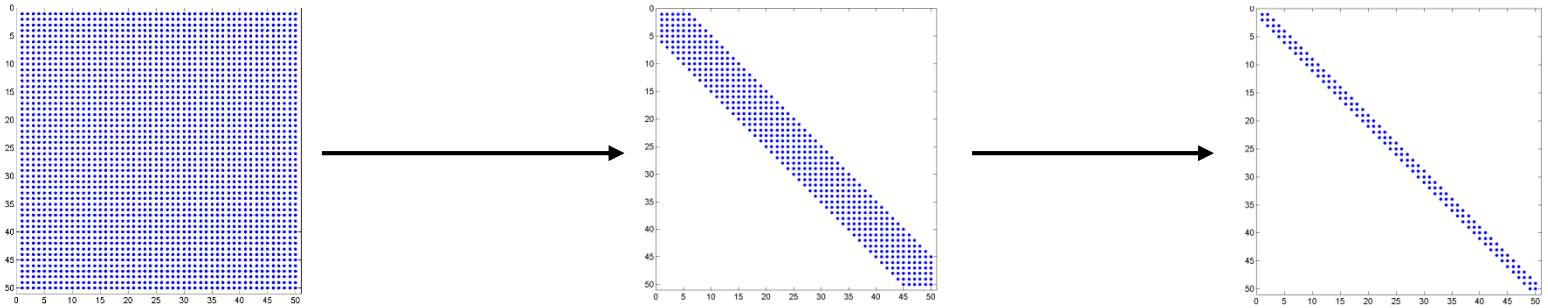
Therefore A and A_{new} have the same eigenvalues

Repeat these QR-steps until convergence against diagonal (upper triangular) matrix.
Use last diagonal entry r as shift A-rI, apply QR step on shifted matrix.

8.10 Twostep Tridiagonalization

Reduce full matrix to tridiagonal (upper Hessenberg)
Sequential BLAS1!

For allowing better parallelism reduce matrix A to
block-banded form, and then in a second step to tridiagonal form.



Advantage:
First step allows block/BLAS3 operations and is good in parallel.
second step is cheap; can be implemented e.g. by MR^3 or D&C.

Disadvantage: For eigenvectors 2 transformations necessary

Bothsided Householder for Tridiagonalization

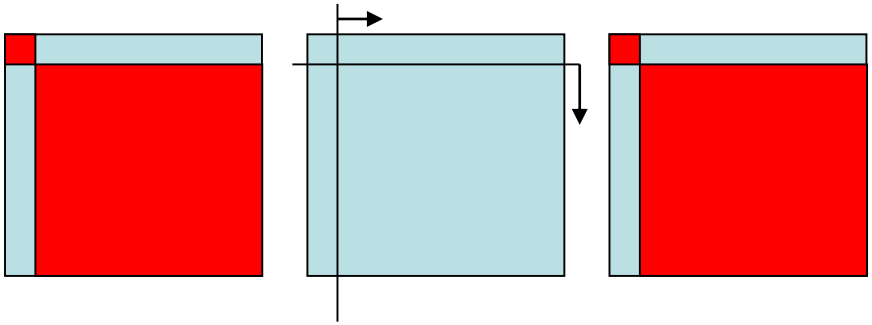
Compute Householder vector u in order to eliminate subtridiagonal entries in the first column/row.

Apply

$$A \rightarrow (I-2uu^H)A(I-2uu^H) = A - 2u(u^HA) - 2(Au)u^H + 4\,uu^H(u^HAu) =$$

$$= A - 2u(u^HA+ru^H) - 2(Au+ru)u^H =$$

$$= A - uy^H - yu^H$$



Two matrix update steps

To reduce BLAS2 operations work blockwise,

$A \rightarrow A - UY^H - YU^H$ (BLAS3)

but still first Au is needed (BLAS2).

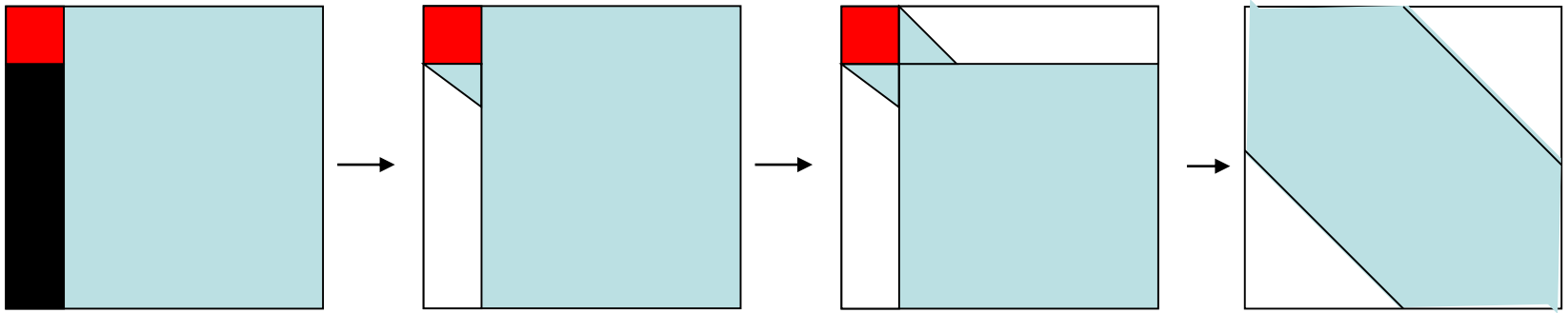
Block-Band reduction

In the first step find QR decomposition of subblock $A(1 + b : n, 1 : n_b) = A_1$ where b is the bandwidth and n_b is a block size.

Compute QR decomposition of black part A_1:

Applying (I , Q^H) from the left leads to triangular form of black part.

Applying from both sides: Band structure.

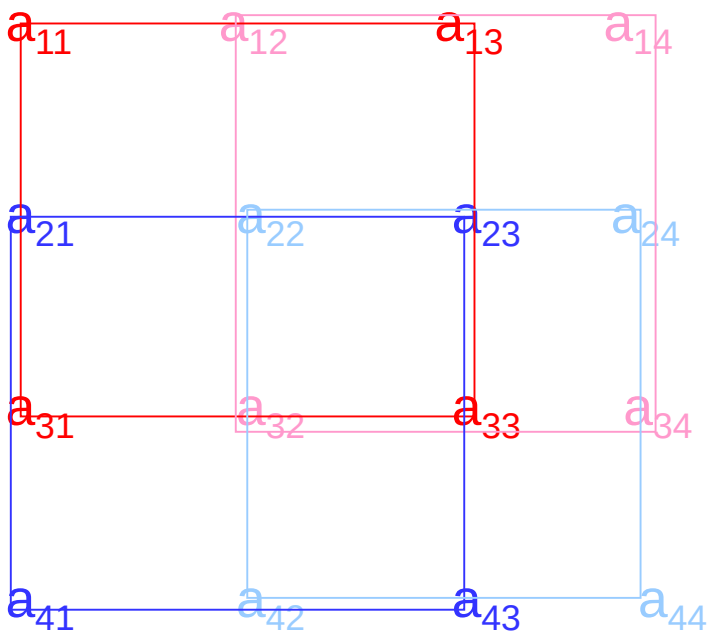


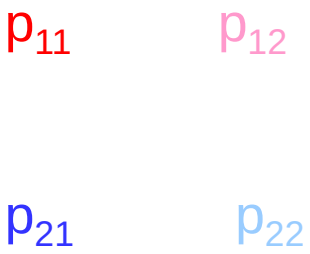
Store Householder vectors on positions of new generated zeros.

Use Tall&Skinny or Cholesky QR.

2D-Cyclic Data Distribution

4 x 4 – Matrix on 2 x 2 processor array

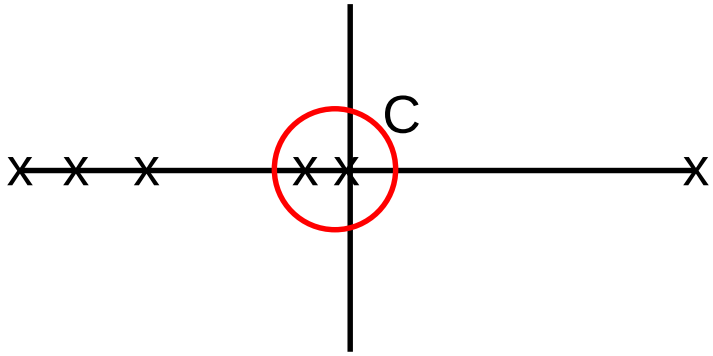




Advantage: better load balancing because matrices and Householder vectors are getting smaller.

8.11 FEAST

Use integration over closed curve C in complex plane in order to derive an approximation to the subspace built by the eigenvectors related to the eigenvalues inclosed by the curve.



Closed curve contains 2 eigenvalues with 2 (orthogonal) eigenvectors → 2-dim subspace

FEAST c't

$$U := \frac{1}{2\pi i}\int_C (zI - A)^{-1} Y \quad dz$$

For rank 2 matrix Y, the computed matrix U contains the span of the 2 eigenvectors in C.

With U computed build the small matrices

$$A_U = U^H A U, \qquad B_U = U^H U$$

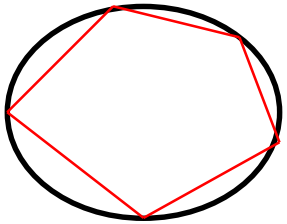
and solve the small eigenvalue problem

$$A_U W = B_U W \cdot \Lambda$$

Repeat with Y = X = U*W until convergence.

FEAST c't

Main work: Use quadrature rule with discretization points z_j , j=1,…,p, in C to compute the integral.



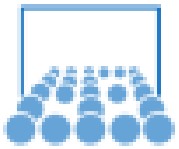
Therefore, we need to solve

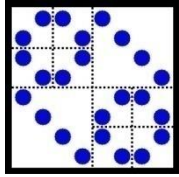
$$\left(z_j I - A\right) U_j = Y$$

for different z_j and blocks $Y=(y_1,\ldots,y_m)$

$$U = \frac{1}{2\pi i}\sum_{j=1}^{p} \omega_j \varphi(t_j)\left(\varphi(t_j) I - A\right)^{-1} Y =$$

$$= \frac{1}{2\pi i}\sum_{j=1}^{p} \omega_j z_j \left(z_j I - A\right)^{-1} Y$$





Advantages:

First level of parallelism:

Choose different curves containing all wanted eigenvalues.

Problem: Gap

Second level of parallelism:

Solve linear equations for different z_j and

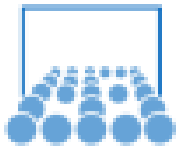
for one z_j for different columns of Y

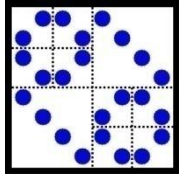
Problem: Might be near singular for z_j close to eigenvalue

Third level of parallelism:

Parallelize iterative solver.

Problem: Convergence for sparse iterative solver





Problem

Linear equations are extremely ill-conditioned if eigenvalues are close to curve and therefore $z_j I - A$ very ill-conditioned.

Slow convergence of iterative solver!

Preconditioning?

Large eigenvalue clusters. How to separate?

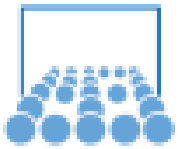
Generalized Eigenvalue Problem:

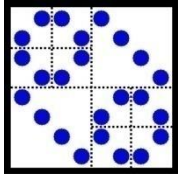
$$Ax = \lambda Bx, A = A^H, B = B^H > 0, spd$$

Compute Cholesky decomposition $B=L^HL$ and solve

$$Ax = \lambda L^H Lx \Leftrightarrow \left(L^{-H} AL^{-1}\right)Lx = Lx \Leftrightarrow$$

$$\Leftrightarrow \left(L^{-H} AL^{-1}\right)y = y, y = Lx$$





Different Questions:

Sparse or dense?

Eigenvectors or not?

All eigenvalues or only a few?

Interior eigenvalues?

Clustered eigenvalues?

Real or complex?

Symmetric or not?

Generalized or not?