Eigenvalues and Singular Values

Aleksandar Donev
Courant Institute, NYU[1]
donev@courant.nyu.edu

[1]Course MATH-UA.0252/MA-UY_4424, Spring 2021

Spring 2021

Outline

Outline

Eigenvalue Decomposition

- For a square matrix $\mathbf{A} \in \mathbb{C}^{n \times n}$, there exists at least one λ such that

$$\mathbf{A}\mathbf{x} = \lambda \mathbf{x} \quad \Rightarrow \quad (\mathbf{A} - \lambda \mathbf{I})\,\mathbf{x} = \mathbf{0}$$

- Putting the **eigenvectors** $\mathbf{x}_j$ as columns in a matrix $\mathbf{X}$, and the **eigenvalues** λ_j on the diagonal of a diagonal matrix $\mathbf{\Lambda}$, we get

$$\mathbf{A}\mathbf{X} = \mathbf{X}\mathbf{\Lambda}.$$

- A matrix is **non-defective** or **diagonalizable** if there exist n **linearly independent eigenvectors**, i.e., if the matrix $\mathbf{X}$ is invertible:

$$\mathbf{X}^{-1}\mathbf{A}\mathbf{X} = \mathbf{\Lambda}$$

leading to the **eigen-decomposition** of the matrix

$$\mathbf{A} = \mathbf{X}\mathbf{\Lambda}\mathbf{X}^{-1}.$$

Unitarily Diagonalizable Matrices

- A **unitary** (complex) or **orthogonal** (real) matrix $\mathbf{U}$ has orthogonal colums each of which has unit L_2 norm:

$$\mathbf{U}^{-1} = \mathbf{U}^{\star}.$$

  Recall that star denotes **adjoint** (conjugate transpose).
- A matrix is **unitarily diagonalizable** if there exist n linearly independent **orthogonal eigenvectors**, $\mathbf{X} \equiv \mathbf{U}$,

$$\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\star}.$$

  There is a **geometric interpretation** of this (sphere->ellipsoid).
- Theorem: **Hermitian matrices**, $\mathbf{A}^{\star} = \mathbf{A}$, are unitarily diagonalizable and have **real eigenvalues**.
  For real matrices we use the term **symmetric**.

Non-diagonalizable Matrices

- For matrices that are not diagonalizable, one can use **Jordan form factorizations**, or, more relevant to numerical mathematics, the **Schur factorization** (decomposition):

$$\mathbf{A} = \mathbf{U}\mathbf{T}\mathbf{U}^{\star},$$

  where $\mathbf{T}$ is **upper-triangular** (unlike **Jordan form** where only nonzeros are on super-diagonal).
- The eigenvalues are on the diagonal of $\mathbf{T}$, and in fact if $\mathbf{A}$ is unitarily diagonalizable then $\mathbf{T} \equiv \mathbf{\Lambda}$.
- The Schur decomposition is **not unique** but it is the best generalization of the eigenvalue (spectral) decomposition to general matrices.

Singular Value Decomposition (SVD)

Every matrix has a **singular value decomposition (SVD)**

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\star = \sum_{i=1}^{p} \sigma_i \mathbf{u}_i \mathbf{v}_i^\star$$

$$[m \times n] = [m \times m]\,[m \times n]\,[n \times n]\,,$$

where $\mathbf{U}$ and $\mathbf{V}$ are **unitary matrices** whose columns are the left, $\mathbf{u}_i$, and the right, $\mathbf{v}_i$, **singular vectors**, and

$$\mathbf{\Sigma} = \text{Diag}\left\{\sigma_1, \sigma_2, \ldots, \sigma_p\right\}$$

is a **diagonal matrix** with real positive diagonal entries called **singular values** of the matrix

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0,$$

and $p = \min(m, n)$ is the maximum possible rank of the matrix.

Comparison to eigenvalue decomposition

- Recall the eigenvector decomposition for diagonalizable matrices

$$\mathbf{AX} = \mathbf{X\Lambda}.$$

- The singular value decomposition can be written similarly to the eigenvector one

$$\mathbf{AV} = \mathbf{U\Sigma}$$
$$\mathbf{A}^\star\mathbf{U} = \mathbf{V\Sigma}$$

and they both **diagonalize A**, but there are some important **differences**:

1. The SVD exists for any matrix, not just diagonalizable ones.
2. The SVD uses different vectors on the left and the right (different basis for the domain and image of the linear mapping represented by **A**).
3. The SVD always uses orthonormal basis (unitary matrices), not just for unitarily diagonalizable matrices.

Relation to Eigenvalues

- For **Hermitian (symmetric) matrices**, there is **no fundamental difference** between the SVD and eigenvalue decompositions (and also the Schur decomposition).
- The squared singular values are **eigenvalues of the normal matrix**:

$$\sigma_i(\mathbf{A}) = \sqrt{\lambda_i(\mathbf{A}\mathbf{A}^\star)} = \sqrt{\lambda_i(\mathbf{A}^\star\mathbf{A})}$$

since

$$\mathbf{A}^\star\mathbf{A} = (\mathbf{V}\mathbf{\Sigma}\mathbf{U}^\star)(\mathbf{U}\mathbf{\Sigma}\mathbf{V}^\star) = \mathbf{V}\mathbf{\Sigma}^2\mathbf{V}^\star$$

- Similarly, the singular vectors are eigenvectors of $\mathbf{A}^\star\mathbf{A}$ or $\mathbf{A}\mathbf{A}^\star$.

Matrix norms

- Recall: Matrix norm **induced** by a given vector norm:

$$\|\mathbf{A}\| = \sup_{\mathbf{x}\neq\mathbf{0}} \frac{\|\mathbf{A}\mathbf{x}\|}{\|\mathbf{x}\|} \quad \Rightarrow \|\mathbf{A}\mathbf{x}\| \leq \|\mathbf{A}\| \, \|\mathbf{x}\|$$

- Special cases of interest are:
  1. The 2-norm or **spectral norm**, $\|\mathbf{A}\|_2 = \sigma_1$ (largest singular value)
  2. The Euclidian or **Frobenius norm**, $\|\mathbf{A}\|_F = \sqrt{\sum_{i,j} |a_{ij}|^2}$ (note this is not an induced norm)
- Unitary matrices are important because they are **always well-conditioned**, $\kappa_2(\mathbf{U}) = 1$.

Outline

The need for iterative algorithms

- The eigenvalues are roots of the **characteristic polynomial** of **A**, which is generally of order n.
- According to Abel's theorem, there is no closed-form (rational) solution for $n \geq 5$.
  All eigenvalue algorithms must be iterative!
- There is an important distinction between iterative methods to:
  - Compute **all eigenvalues** (similarity transformations). These are based on dense-matrix factorizations such as the QR factorization, with total cost $O(n^3)$.
  - Compute **only one or a few eigenvalues**, typically the smallest or the largest one (e.g., power method). These are similar to iterative methods for solving linear systems.

Sparse Matrices

- Recall that for a diagonalizable matrix

$$\mathbf{A}^n = \mathbf{X}\mathbf{\Lambda}^n\mathbf{X}^{-1}$$

  and **assume well-separated eigenvalues** $|\lambda_1| > |\lambda_2| \geq |\lambda_3| \cdots |\lambda_n|$, and that the columns of $\mathbf{X}$ are normalized, $\|\mathbf{x}_j\| = 1$.
- For sparse matrices we sometimes only need to know a **few of the eigenvalues**/vectors, not all of them.
- Notably, knowing the eigenvector corresponding to the **smallest and largest (in magnitude) eigenvalues** is often most important (see Google Page Rank algorithm).

Iterative Method

- Any **initial guess** vector $\mathbf{q}_0$ can be represented in the linear basis formed by the eigenvectors

$$\mathbf{q}_0 = \mathbf{X}\mathbf{a}$$

- Recall iterative methods for linear systems: **Multiply a vector with the matrix A** many times:

$$\mathbf{q}_{k+1} = \mathbf{A}\mathbf{q}_k$$

$$\mathbf{q}_n = \mathbf{A}^n\mathbf{q}_0 = \left(\mathbf{X}\mathbf{\Lambda}^n\mathbf{X}^{-1}\right)\mathbf{X}\mathbf{a} = \mathbf{X}\left(\mathbf{\Lambda}^n\mathbf{a}\right)$$

Power Method

- As $n \to \infty$, the **eigenvalue of largest modulus** λ_0 will dominate,

$$\mathbf{\Lambda}^n = \lambda_1^n \text{Diag}\left\{1, \left(\frac{\lambda_2}{\lambda_1}\right)^n, \dots\right\} \to \text{Diag}\left\{\lambda_1^n, 0, \dots, 0\right\}$$

$$\mathbf{q}_n = \mathbf{X}\left(\mathbf{\Lambda}^n \mathbf{a}\right) \to \lambda_1^n \mathbf{X} \begin{bmatrix} a_1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \lambda_1^n \mathbf{x}_1$$

- Therefore the **normalized iterates** converge to the eigenvector:

$$\tilde{\mathbf{q}}_n = \frac{\mathbf{q}_n}{\|\mathbf{q}_n\|} \to \mathbf{x}_1$$

- The **Rayleigh quotient** converges to the eigenvalue:

$$r_A\left(\mathbf{q}_n\right) = \frac{\mathbf{q}_n^\star \mathbf{A} \mathbf{q}_n}{\mathbf{q}_n \cdot \mathbf{q}_n} = \tilde{\mathbf{q}}_n^\star \mathbf{A} \tilde{\mathbf{q}}_n \to \lambda_1$$

An alternative derivation

$$q_0 = \sum_n a_i \cdot x_i$$

$$q_n = A^n q_0 = \sum_n a_i (A^n x_i)$$

$$= \sum a_i (\lambda_i^n x_i) =$$

$$= \sum a_i \lambda_1^n \left(\frac{\lambda_i}{\lambda_1}\right)^n \cdot x_i \xrightarrow[n\to\infty]{as} a_1 \lambda_1^n x_1 + O\left(\left|\frac{\lambda_2}{\lambda_1}\right|^n\right)$$

$$\Rightarrow \boxed{\tilde{q}_n \to x_1 + O(\varepsilon)\, x_2 + O(\varepsilon) x_3 + \cdots \quad \text{where} \quad \varepsilon = \left|\frac{\lambda_2}{\lambda_1}\right|^n}$$

Now $\tilde{q}_n^* A \tilde{q}_n = x_1^* A x_1 + O(\varepsilon^2) \sum_{i,j\neq 1} x_i^* A x_j$

$$\Rightarrow \boxed{\hat{\lambda}_n'' = \lambda_1 + O(\varepsilon^2)}$$

Power Iteration

Start with an initial guess $\mathbf{q}_0$, and then iterate:

1. Compute **matrix-vector product** and normalize it:

$$\mathbf{q}_k = \frac{\mathbf{A}\mathbf{q}_{k-1}}{\left\|\mathbf{A}\mathbf{q}_{k-1}\right\|}$$

2. Use Raleigh quotient to obtain **eigenvalue estimate**:

$$\hat{\lambda}_k = \mathbf{q}_k^\star \mathbf{A}\mathbf{q}_k$$

3. **Test for convergence**: Evaluate the residual

$$\mathbf{r}_k = \mathbf{A}\mathbf{q}_k - \hat{\lambda}_k \mathbf{q}_k$$

and terminate if the residual norm is smaller than some tolerance, e.g., for tolerance $\epsilon \ll 1$,

$$\|\mathbf{r}_k\| \approx \left|\lambda_1 - \hat{\lambda}_k\right| \leq \epsilon \hat{\lambda}_k.$$

Convergence Estimates

- The normalized iterates converge to the eigenvector **linearly**:

$$\|\mathbf{q}_k - (\pm\mathbf{x}_1)\| = O\left(\left|\frac{\lambda_2}{\lambda_1}\right|^k\right)$$

- Typically the eigenvalue estimate converges **quadratically**:

$$\left\|\hat{\lambda}_k - \lambda_1\right\| \sim O\left(\left|\frac{\lambda_2}{\lambda_1}\right|^{2k}\right)$$

- The power method is fast when the **dominant eigenvalue is well-separated from the rest** (even if it is degenerate).
- This conclusion is rather general for all iterative methods: Convergence is good for **well-separated** eigenvalues, bad otherwise.

Eigenvalues in MATLAB

- The **Schur decomposition** is provided by $[U, T] = schur(A)$.
- In MATLAB, sophisticated variants of the **QR algorithm** (LAPACK library) are implemented in the function *eig*:

$$\Lambda = eig(A)$$

$$[X, \Lambda] = eig(A)$$

- For large or sparse matrices, iterative methods based on the **Arnoldi iteration** (ARPACK library), can be used to obtain a **few** of the largest eigenvalues:

$$\Lambda = eigs(A, n_{eigs})$$

$$[X, \Lambda] = eigs(A, n_{eigs})$$

Outline

Sensitivity (conditioning) of the SVD

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\star$$

- Since unitary matrices have unit 2-norm,

$$\|\delta\Sigma\|_2 \approx \|\delta A\|_2 .$$

- The SVD computation is always **perfectly well-conditioned**!
- However, this refers to absolute errors: The **relative error** of small singular values will be large.
- The **power of the SVD** lies in the fact that it always exists and can be computed stably...but it is somewhat **expensive to compute**.

Computing the SVD

- The SVD can be computed by performing an eigenvalue computation for the **normal matrix $\mathbf{A}^*\mathbf{A}$** (a positive-semidefinite matrix).
- This squares the condition number for small singular values and is **not numerically-stable**.
- Instead, modern algorithms use an algorithm based on computing eigenvalues / eigenvectors using the QR factorization.
- The cost of the calculation is $\sim O(mn^2)$, of the same order as eigenvalue calculation if $m \sim n$.

Reduced SVD

The **full (standard) SVD**

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\star = \sum_{i=1}^{p} \sigma_i \mathbf{u}_i \mathbf{v}_i^\star$$
$$[m \times n] = [m \times m]\,[m \times n]\,[n \times n]\,,$$

is in practice often computed in **reduced (economy) SVD** form, where $\mathbf{\Sigma}$ is $[p \times p]$:

$$[m \times n] = [m \times n]\,[n \times n]\,[n \times n] \quad \text{for} \quad m > n$$
$$[m \times n] = [m \times m]\,[m \times m]\,[m \times n] \quad \text{for} \quad n > m$$

This contains all the information as the full SVD but can be **cheaper to compute** if $m \gg n$ or $m \ll n$.

In MATLAB

- $[U, \Sigma, V] = svd(A)$ for **full SVD**, computed using a QR-like method.
- $[U, \Sigma, V] = svd(A,' econ')$ for **economy SVD**.
- The **least-squares solution** for square, overdetermined, underdetermined, or even rank-defficient systems can be computed using *svd* or *pinv* (pseudo-inverse, see homework).
- The q largest singular values and corresponding approximation can be computed efficiently for **sparse matrices** using

$$[U, \Sigma, V] = svds(A, q).$$

Outline

Rank-Revealing Properties

- Assume the rank of the matrix is r, that is, the dimension of the range of $\mathbf{A}$ is r and the dimension of the null-space of $\mathbf{A}$ is $n - r$ (recall the fundamental theorem of linear algebra).
- The SVD is a **rank-revealing** matrix factorization because only r of the singular values are nonzero,

$$\sigma_{r+1} = \cdots = \sigma_p = 0.$$

- The left singular vectors $\{\mathbf{u}_1, \ldots, \mathbf{u}_r\}$ form an **orthonormal basis for the range** (column space, or image) of $\mathbf{A}$.
- The right singular vectors $\{\mathbf{v}_{r+1}, \ldots, \mathbf{v}_n\}$ form an **orthonormal basis for the null-space** (kernel) of $\mathbf{A}$.

The matrix pseudo-inverse

- For square non-singular systems, $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$. Can we generalize the matrix inverse to non-square or rank-deficient matrices?
- Yes: **matrix pseudo-inverse** (Moore-Penrose inverse):

$$\mathbf{A}^{\dagger} = \mathbf{V}\mathbf{\Sigma}^{\dagger}\mathbf{U}^{\star},$$

where

$$\mathbf{\Sigma}^{\dagger} = \text{Diag}\left\{\sigma_1^{-1}, \sigma_2^{-1}, \ldots, \sigma_r^{-1}, 0, \ldots, 0\right\}.$$

- In numerical computations very small singular values should be considered to be zero (see homework).
- The least-squares solution to over- or under-determined linear systems $\mathbf{A}\mathbf{x} = \mathbf{b}$ can be obtained from:

$$\mathbf{x} = \mathbf{A}^{\dagger}\mathbf{b}.$$

Proof of Least-Squares (1)

$$\min_x \|Ax-b\|_2 \leftarrow \text{LEAST SQUARES}$$

SUCH THAT $\|x\|_2$ IS MINIMAL

$$(Ax-b)^*(Ax-b) = x^*(A^*A)x - 2x^*A^*b + \dots$$

USING SVD: $x^*A^*Ax = x^*V\Sigma^2V^*x$

AND $x^*A^*b = b^*(Ax) = (U^*b)\Sigma(V^*x)$

DENOTING
$$\begin{cases} V^*x = w & \leftarrow \text{NEW VARIABLE} \\ U^*b = c & \leftarrow \text{CONSTANT} \end{cases}$$

(1)

Proof of Least-Squares (2)

$$\|Ax-b\|_2^2 = w^*\Sigma^2 w - 2c^*\Sigma w + \dots$$

$$= \sum_{i=1}^{r} \sigma_i^2 |w_i|^2 - 2\sum_{i=1}^{r} (\sigma_i w_i) c_i^*$$

$$= \sum_{i=1}^{r} |\sigma_i w_i - c_i|^2 + \text{CONSTANTS}$$

WHICH IS MINIMIZED IF

$$\sigma_i w_i = c_i \Rightarrow w_i = \frac{c_i}{\sigma_i}$$

or

$$\boxed{(V^*x)_i = \frac{(U^*b)_i}{\sigma_i}, \quad i \le r} \quad (2)$$

Proof of Least-Squares (3)

How about $w_{r+1}, \ldots, w_m$?

$$\|x\|_2^2 = \|Vw\|_2^2 = w^*(V^*V)w^*$$

$$= \|w\|_2^2 = \sum |w_i|^2$$

So the norm of x is minimized if the norm of w is minimized.

$$\Rightarrow \boxed{w_{r+1} = \ldots = 0}$$

$$\Rightarrow x = Vw = V\Sigma^+ c =$$

$$= (V\Sigma^+ u^*)b = A^+ b$$

QED (3)

Low-rank approximations

- The SVD is a decomposition into **rank-1 outer product matrices**:

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\star = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^\star = \sum_{i=1}^{r} \mathbf{A}_i$$

- The rank-1 components $\mathbf{A}_i$ are called **principal components**, the most important ones corresponding to the larger σ_i.
- Ignoring all singular values/vectors except the first q, we get a **low-rank approximation**:

$$\mathbf{A} \approx \hat{\mathbf{A}}_q = \mathbf{U}_q \mathbf{\Sigma}_q \mathbf{V}_q^\star = \sum_{i=1}^{q} \sigma_i \mathbf{u}_i \mathbf{v}_i^\star.$$

- Theorem: This is the **best approximation** of rank-q in the Euclidian and Frobenius norm:

$$\left\| \mathbf{A} - \hat{\mathbf{A}}_q \right\|_2 = \sigma_{q+1}$$

Applications of SVD

- **Statistical analysis** (e.g., DNA microarray analysis, clustering), often called **Principal Component Analysis (PCA)**
- Data **compression** (e.g., image compression, explained next).
- **Feature extraction**, e.g., face or character recognition (see Eigenfaces on Wikipedia).
- **Latent semantic indexing** for context-sensitive searching (see Wikipedia).
- **Noise reduction** (e.g., weather prediction).

Image Compression

```
>> A=rgb2gray(imread('basket.jpg'));
>> imshow(A);
>> [U,S,V]=svd(double(A));
>> r=25; % Rank-r approximation
>> Acomp=U(:,1:r)*S(1:r,1:r)*(V(:,1:r))';
>> imshow(uint8(Acomp));
```

Compressing an image of a basket

We used only 25 out of the ~ 400 singular values to construct a rank 25 approximation:

Outline

Estimating all eigenvalues / eigenvectors

- Iterative methods akin the power method are not suitable for estimating all eigenvalues.
- Basic idea: Build a sequence of matrices $\mathbf{A}_k$ that all share eigenvalues with $\mathbf{A}$ via **similarity transformations**:

$$\mathbf{A}_{k+1} = \mathbf{P}^{-1}\mathbf{A}_k\mathbf{P}, \text{ starting from } \mathbf{A}_1 = \mathbf{A}.$$

- The goal is to reduce the matrix $\mathbf{A}_{k\to\infty}$ to as close to diagonal as possible.
- A numerically stable and good way to do this is to use the QR factorization:

$$\mathbf{A}_k = \mathbf{Q}_{k+1}\mathbf{R}_{k+1}$$

$$\mathbf{A}_{k+1} = \mathbf{Q}_{k+1}^{-1}\mathbf{A}_k\mathbf{Q}_{k+1} = \left(\mathbf{Q}_{k+1}^{-1}\mathbf{Q}_{k+1}\right)\mathbf{R}_{k+1}\mathbf{Q}_{k+1} = \mathbf{R}_{k+1}\mathbf{Q}_{k+1}.$$

The basic QR method

- The behavior of the QR iteration can be understood most transparently as follows [following Trefethen and Bau]:
- Observation: The range of the matrix $\mathbf{A}^k$ converges to the space spanned by the eigenvectors of $\mathbf{A}$,
  with the eigenvectors corresponding to the largest eigenvalues dominating as $k \to \infty$ (so this is ill-conditioned).
- Recall: The columns of $\mathbf{Q}$ in $\mathbf{A} = \mathbf{QR}$ form an **orthonormal basis** for the range of $\mathbf{A}$.
- Idea: Form a **well-conditioned basis for the eigenspace** of $\mathbf{A}$ by factorizing:

$$\mathbf{A}^k = \tilde{\mathbf{Q}}_k \tilde{\mathbf{R}}_k$$

  and then calculate

$$\mathbf{A}_k = \tilde{\mathbf{Q}}_k^{-1} \mathbf{A} \tilde{\mathbf{Q}}_k = \tilde{\mathbf{Q}}_k^{\star} \mathbf{A} \tilde{\mathbf{Q}}_k.$$

- It is not too hard to show that this produces the **same sequence** of matrices $\mathbf{A}_k$ as the QR algorithm.

Why the QR algorithm works

- Summary: The columns of $\tilde{\mathbf{Q}}_k$ converge to the eigenvectors, and

$$\mathbf{A}_k = \tilde{\mathbf{Q}}_k^\star \mathbf{A} \tilde{\mathbf{Q}}_k.$$

- We can recognize the above as a matrix of Rayleigh quotients, which for diagonalizable matrices

$$(\mathbf{A}_k)_{ij} = \tilde{\mathbf{q}}_i^\star \mathbf{A} \tilde{\mathbf{q}}_j \to \lambda_i \delta_{ij} = \begin{cases} \lambda_i & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$$

  showing that (under suitable assumptions):

$$\mathbf{A}_k \to \mathbf{\Lambda}$$

- It can also be shown that

$$\tilde{\mathbf{Q}}_k = \mathbf{Q}_1 \mathbf{Q}_2 \cdots \mathbf{Q}_k \to \mathbf{X}$$

More on QR algorithm

- The convergence of the basic QR algorithm is closely related to that of the power method: It is only fast if all eigenvalues are **well-separated**.
- For more general (non-diagonalizable) matrices in complex arithmetic, the algorithm converges to the **Schur decomposition** $\mathbf{A} = \mathbf{UTU}^\star$,

$$\mathbf{A}_k \to \mathbf{T} \text{ and } \tilde{\mathbf{Q}}_k \to \mathbf{U}.$$

- It is possible to implement the algorithm entirely using real arithmetic (no complex numbers).
- There are several key improvements to the basic method that make this work in practice: **Hessenberg matrices** for faster QR factorization, **shifts** and **deflation** for acceleration.
- There are other sophisticated algorithms as well, such as the **divide-and-conquer algorithm**, and the best are implemented in the library LAPACK (MATLAB).