Numerical Linear Algebra for Computational Science and Information Engineering

Lecture 09
Basic Iterative Methods for Eigenvalue Problems

Nicolas Venkovic
nicolas.venkovic@tum.de

Group of Computational Mathematics
School of Computation, Information and Technology
Technical University of Munich

Winter 2025-26

Outline I

Computing eigenvalues exactly is impossible

- **Computing eigenvalues and eigenvectors is** a very **difficult** task.

  There is **no direct method** for computing eigenvalues of matrices of size five or higher in general.

  That is, there is **no algorithm** that **can compute eigenvalues exactly** assuming exact arithmetic.

- Moreover, it can be proved that **a method that computes eigenvalues exactly cannot exist** for general matrices of size five or higher.

  The reason for this is the Abel-Ruffini theorem, which states that **no direct method** exists **to find exact zeros of a polynomial** of degree five or higher.

  That is the case because **computing the roots of any polynomial is equivalent to finding the eigenvalues of a matrix**.

  Thus, since there is no method for finiding zeros of a polynomial, then there cannot exist an exact method for finding eigenvalues of a general matrix.

Computing eigenvalues exactly is impossible, cont'd

You saw one side of the equivalence between solving for eigenvalues of a general matrix and solving for the zeros of a polynomial in your Linear Algebra class.

To see the other direction, consider a generic polynomial given by

$$p(x) = x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0.$$

Then, there is a matrix

$$A = \begin{pmatrix} 0 & 1 & & & & \\ & 0 & 1 & & & \\ & & 0 & & & \\ & \vdots & & & & \\ & & & & 0 & 1 \\ -a_0 & -a_1 & -a_2 & \cdots & -a_{n-2} & -a_{n-1} \end{pmatrix}$$

such that, if we pick $u = [1\, z\, z^2\, \ldots\, z^{n-1}]^T$ where z is a root of $p(x)$, then we have $Au = zu$ so that (z, u) is an eigenpair of A.

Consequently, all roots of $p(x)$ are eigenvalues of A.

Convention

- Let us denote $A = X\Lambda X^{-1}$ an eigendecomposition of A.

  In this lecture, all the algorithms will normalize vectors, i.e., replace x by $x/\|x\|_2$ during the iterative process.

  Therefore, when discussing convergence, we will assume the columns of X have norm 1.

  This is done without loss of generality, since $A = X\Lambda X^{-1}$ remains valid irrespective of the magnitude of the columns of X.

  Moreover, in many places, results will be stated "up to a sign" or "up to a unit complex factor", because even with normed columns, the matrix X of an eigendecomposition is not unique.

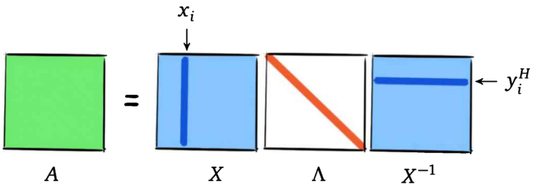
Methods for computing a single eigenvalue

Section 5.1 in Darve & Wootters (2021)

Taking powers of A

- Suppose that A is a square diagonalizable matrix.

  Then A has an eigenvalue decomposition $A = X\Lambda Y^H$ where the columns x_i of X are right eigenvectors of A, and the columns y_i of $Y := X^{-H}$ are left eigenvectors of A:



Darve, E., & Wootters, M. (2021). Numerical linear algebra with Julia. Society for Industrial and Applied Mathematics.

  One thing about the eigendecomposition is that powers of A are such that

$$A^k = X\Lambda^k Y^H = \sum_i \lambda_i^k x_i y_i^H.$$

  Notice that, even if A is real, it can have complex eigenvalues and vectors.

  Note also that left and right eigenvectors of A coincide if A is normal.

Taking powers of A, cont'd

- Let us assume the eigenvalues of A are ordered such that

$$|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n|$$

  where, in particular, the largest eigenvalue has magnitude strictly greater than the second one.

  Then, even for moderate values of the power k, we expect λ_1^k to dominate in A^k, i.e., $|\lambda_1^k| \gg |\lambda_2^k| \geq \cdots \geq |\lambda_n^k|$ so that

$$A^k = \lambda_1^k x_1 y_1^H + \cdots + \lambda_n^k x_n y_n^H \approx \lambda_1^k x_1 y_1^H.$$

- Let's multiply A^k by a random vector z, such that $y_1^H z$ is not too small, then

$$A^k z \approx \lambda_1^k x_1 y_1^H z = \lambda_1^k (y_1^H z) x_1$$

  so that $A^k z / \|A^k z\|_2$ gives a **good approximation of** x_1.

Power iteration

- The power iteration is based on this idea of taking powers of A to approximate the largest eigenpair. The algorithm is as follows:

```
1. Sample a random vector $q^{(0)} \in \mathbb{C}^n$
2. $q^{(0)} := q^{(0)}/\|q^{(0)}\|_2$
3. For $k = 0, 1, 2 \ldots$
4.     $z^{(k)} := Aq^{(k)}$
5.     $\lambda^{(k+1)} = z^{(k)H}q^{(k)}$
6.     $q^{(k+1)} := z^{(k)}/\|z^{(k)}\|_2$
```

  where $(\lambda^{(k)}, q^{(k)})$ is an iterate approximating the largest eigenpair of A. At the k-th step, the approximate eigenvector is

$$q^{(k)} = A^k q^{(0)}/\|A^k q^{(0)}\|_2,$$

  and the corresponding approximate eigenvalue is $\lambda^{(k)} = q^{(k)H}Aq^{(k)}$.
  Note that, even though $q^{(k)}$ is formed with A^k, the matrix power A^k is not explicitly computed.
  Instead, we just perform repeated matrix-vector products.

Convergence of power iteration

- Let us assume again that the eigenpairs $(\lambda_1, x_1), \dots, (\lambda_n, x_n)$ of A are ordered such that $|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n|$.
- The starting vector $q^{(0)}$ can be expressed in the basis formed by the eigenvectors of A, i.e.,

$$q^{(0)} = \alpha_1 x_1 + \cdots + \alpha_n x_n.$$

For the method to work, we need to assume $\alpha_1 \neq 0$, that is, $q^{(0)}$ is not orthogonal to x_1.
- Then, we have

$$A^k q^{(0)} = \sum_{i=1}^{n} \alpha_i A^k x_i = \sum_{i=1}^{n} \alpha_i \lambda_i^k x_i$$

which can be factorized as follows:

$$\begin{aligned} A^k q^{(0)} &= \alpha_1 \lambda_1^k x_1 + \alpha_2 \lambda_2^k x_2 + \cdots + \alpha_n \lambda_n^k x_n \\ & \alpha_1 \lambda_1^k \left(x_1 + \frac{\alpha_2}{\alpha_1} \left(\frac{\lambda_2}{\lambda_1} \right)^k x_2 + \cdots + \frac{\alpha_n}{\alpha_1} \left(\frac{\lambda_n}{\lambda_1} \right)^k x_n \right) \end{aligned}$$

Convergence of power iteration, cont'd

From that expression, we have

$$\|A^k q^{(0)}\|_2 = |\alpha_1 \lambda_1^k|(1 + \mathcal{O}(|\lambda_2/\lambda_1|)) \text{ and}$$
$$\|(\alpha_1\lambda_1^k)^{-1} A^k q^{(0)} - x_1\|_2 = \mathcal{O}\left(\left|\frac{\lambda_2}{\lambda_1}\right|^k\right)$$

which, along with the fact that $\|A^k q^{(0)}\|_2 \approx |\alpha_1 \lambda_1^k|$ implies that our estimate $q^{(k)} = A^k q^{(0)}/\|A^k q^{(0)}\|$ approaches x_1 with an error $\mathcal{O}(|\lambda_2/\lambda_1|^k)$.

In summary, we have

$$\|q^{(k)} - x_1\|_2 = \mathcal{O}\left(\left|\frac{\lambda_2}{\lambda_1}\right|^k\right) \quad \text{and} \quad |\lambda^{(k)} - \lambda_1| = \mathcal{O}\left(\left|\frac{\lambda_2}{\lambda_1}\right|^k\right).$$

- Although it is a good starting point, this version of power iteration is limited as it cannot find approximates of any eigenvalue except the largest one. It also cannot leverage given approximations of λ_i.

Inverse iteration

- Assume we are equipped with an approximation μ of the eigenvalue λ_i of A.
- An **inverse iteration** uses μ to form an abritrarily good approximation of λ_i.
- If μ is a good approximation of λ_i, then

  The shifted matrix $A - \mu I_n$ has a small eigenvalue $\lambda_i - \mu$.

  The shift-and-invert matrix $(A - \mu I_n)^{-1}$ has a large eigenvalue $1/(\lambda_i - \mu)$.

  So, a power iteration applied to $(A - \mu I_n)^{-1}$ should allow us to calculate x_i very quickly, since $1/(\lambda_i - \mu)$ is now the largest eigenvalue, with the corresponding eigenvector x_i.

  The algorithm of inverse iteration is as follows:

```
1. Sample a random vector $q^{(0)} \in \mathbb{C}^n$
2. $q^{(0)} := q^{(0)}/\|q^{(0)}\|_2$
3. For $k = 0, 1, 2 \ldots$
4.     Solve for $z^{(k)}$ s.t. $(A - \mu I_n)z^{(k)} = q^{(k)}$ // $z^{(k)} := (A - \mu I_n)^{-1}q^{(k)}$
5.     $q^{(k+1)} := z^{(k)}/\|z^{(k)}\|_2$
6.     $\lambda^{(k+1)} = q^{(k+1)H} A q^{(k+1)}$
```

Convergence of inverse iteration

- Similarly to power iteration, we can characterize the convergence of inverse interations by

$$|\lambda^{(k)} - \lambda_i| = \mathcal{O}\left(\left|\frac{\lambda_i - \mu}{\lambda_j - \mu}\right|^k\right)$$

where λ_i and λ_j are the closest and second closest eigenvalues of A to μ, respectively.

If $|\lambda_i - \mu| \ll |\lambda_j - \mu|$, then the convergence is fast.

Rayleigh quotient iteration

- As inverse iterations progess, the iterate $\lambda^{(k)}$ becomes a better approximation of the eigenvalue λ_i than μ.

  One could use this fact to redefine the shift μ and get faster convergence.
- Let us assume the matrix A is real and symmetric so that its eigenvalues and eigenvectors are real, and the eigenvectors are orthogonal.
- The idea to **update the shift μ during the iteration** is deployed in an algortihm called **Rayleigh quotient iteration**.

  Let us consider the **Rayleigh quotient** given by $r(x) = \frac{x^T Ax}{x^T x}$ for $x \neq 0$.

  The Rayleigh quotient is used to approximate an eigenvalue.

  Indeed, note that if x is an eigenvector of A, i.e., $Ax = \lambda x$, then $r(x) = \lambda$ is the corresponding eigenvalue.

Rayleigh quotient iteration, cont'd

- The algorithm for Rayleigh quotient iterations is as follows:

```
1. Sample a random vector $q^{(0)} \in \mathbb{C}^n$
2. $q^{(0)} := q^{(0)}/\|q^{(0)}\|_2$
3. $\lambda^{(0)} := \mu$
4. For $k = 0, 1, 2 \ldots$
5.     Solve for $z^{(k)}$ such that $(A - \lambda^{(k)} I_n) z^{(k)} = q^{(k)}$
6.     $q^{(k+1)} := z^{(k)}/\|z^{(k)}\|_2$
7.     $\lambda^{(k+1)} = q^{(k+1)H} A q^{(k+1)}$
```

- Rayleigh quotient iterations converge faster than inverse iterations.

Convergence of Rayleigh quotient iterations

- Note first that the gradient of the Rayleigh quotient r for a symmetric A is given by $\nabla r(x) = \frac{2}{x^Tx}(Ax - r(x)x)$ so that $r(x_i) = \lambda_i$ implies $\nabla r(x_i) = 0$.
  More often than not, the zeros of ∇r are saddle points, as the Rayleigh quotient is only minimized (resp. maximized) at the smallest eigenpair (resp. largest eigen-pair).
  In particular, we remember the Courant-Fischer theorem from lecture 1 which states
  $$\lambda_{min} = \min_{x\neq 0} \frac{x^TAx}{x^Tx} \text{ and } \lambda_{max} = \max_{x\neq 0} \frac{x^TAx}{x^Tx}.$$
- Then, suppose that y is close to an eigenvector x_i, by Taylor expansion around x_i, we have

$$r(y) \approx r(x_i) + \nabla r(x_i)^T(y - x_i) + (y - x_i)^T H(x_i)(y - x_i)$$

$r(x_i) = \lambda_i$

This is zero since $\nabla r(x_i) = 0$

Here, H is the Hessian matrix.
We have:
$$(y-x_i)^T H(x_i)(y-x_i) \leq \|H(x_i)\|\,\|y-x_i\|^2 = O\left(\|y-x_i\|^2\right)$$

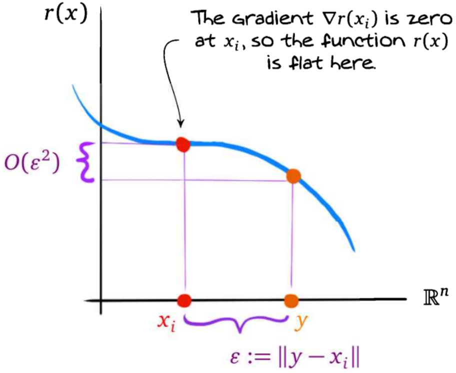
Darve, E., & Wootters, M. (2021). Numerical linear algebra with Julia. Society for Industrial and Applied Mathematics.

Convergence of Rayleigh quotient iterations, cont'd

Consequently, the first order term disapears, leaving us with

$$r(y) = \lambda_i + \mathcal{O}(\|y - x_i\|_2^2)$$

and the behavior of the Rayleigh quotient near an eigenvector x_i is as follows:


$r(x)$
The gradient $\nabla r(x_i)$ is zero at x_i, so the function $r(x)$ is flat here.
$O(\varepsilon^2)$
$\mathbb{R}^n$
x_i
y
$\varepsilon := \|y - x_i\|$

Darve, E., & Wootters, M. (2021). Numerical linear algebra with Julia. Society for Industrial and Applied Mathematics.

Basic QR iteration

Section 5.2 in Darve & Wootters (2021)

Basic QR iteration

- The PageRank algorithm is a variant of power iteration aimed at finding the largest eigenvector of a modified adjacency matrix of a web graph.
  However, in general, iterative methods for computing a single eigenpair have limited applicability.
- Unlike those previously covered iterative methods for eigenvalue solving, QR iterations aim at **finding all the eigenvalues** of a matrix.
- The QR iteration was elected **one of the 10 best algorithms of the 20th century** by Dongarra and Sullivan (2000).
- The QR iteration is the **state of the art eigensolver for small dense eigenvalue problems**. It is implemented **in LAPACK**, and it serves as a **building block of larger**, possibly sparse **iterative eigensolvers**.
- An important assumption of this Section is that A is **diagonalizable with separate eigenvalues**, i.e., such that $|\lambda_1| > |\lambda_2| > \cdots > |\lambda_n|$.
  Because A is real with separate eigenvalues, we have that eigen- and Schur decompositions of A are real.

Dongarra, J., & Sullivan, F. (2000). Guest editor's introduction: The top 10 algorithms. Computing in Science & Engineering 2, 22-23.

Orthogonal iteration for $r = 2$

- Orthogonal iterations allow us to recover more than one eigenvalue at once.
- For starter, consider that only $r = 2$ eigenvalues are needed.
  Then, the pseudocode of orthogonal iterations is as follows:

```
1. Sample two random vector $q_1, q_2 \in \mathbb{R}^n$
2. While not converged :
3.     $q_1 := Aq_1;\ q_2 := Aq_2$
4.     Project $q_2$ onto the space orthogonal to $q_1$ // $q_2 := \left(I_n - \frac{q_1 q_1^T}{q_1^T q_1}\right) q_2$
5.     $q_1 := q_1/\|q_1\|_2,\ q_2 := q_2/\|q_2\|_2$
6. Return $q_1^T A q_1$ and $q_2^T A q_2$
```

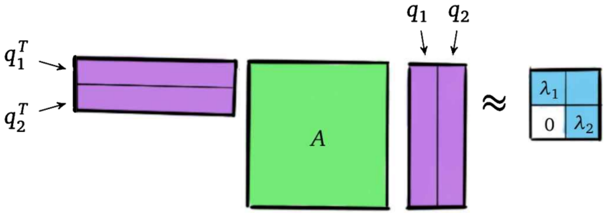
Disregarding the vector q_2, the vector q_1 undergoes a standard power iteration so that, at the k-th step, we have

$$q_1^{(k)} = \frac{A^k q_1^{(0)}}{\|A^k q_1^{(0)}\|_2}$$

which converges towards x_1.

Orthogonal iteration for $r = 2$, cont'd$_1$

- If we assume that $q_1 \approx x_1$ has already converged, then the update step for q_2 is of the form
$$q_2^{(k)} \approx (I_n - x_1 x_1^T) A q_2^{(k-1)}$$
where $I_n - x_1 x_1^T$ is the orthogonal projector onto $\text{span}\{x_1\}^\perp$.
Thus, q_2 is undergoing a power iteration with the matrix $(I_n - x_1 x_1^T)A$.
It can be shown that the largest eigenvalue of this matrix is λ_2 with an eigenvector along $(I_n - x_1 x_1^T)x_2$ towards which q_2 converges.
- Note that, if x_1 and x_2 are not orthogonal, then $(I_n - x_1 x_1^T)x_2$ is not aligned with x_2. However, we do have $\text{span}\{q_1, q_2\} = \text{span}\{x_1, x_2\}$.
Then, we claim that $Q^T A Q$ where $Q = [q_1\ q_2]$ converges to an upper-triangular matrix with λ_1 and λ_2 on the diagonal :



Darve, E., & Wootters, M. (2021). Numerical linear algebra with Julia. Society for Industrial and Applied Mathematics.

Orthogonal iteration for $r = 2$, cont'd$_2$

- First, the upper-triangularity is explained as follows:

$$q_2^T A q_1 \approx q_2^T A x_1 = \lambda_1 q_2^T x_1 \approx \lambda_1 q_2^T q_1 = 0$$

  so that the lower-left entry converges to zero.
- To see that λ_1 and λ_2 lie on the diagonal, it suffices to show that they are eigenvalues of $Q^T A Q$, as $Q^T A Q$ is triangular.

  For this, since $\text{span}\{q_1, q_2\} = \text{span}\{x_1, x_2\}$ after convergence, then there is $v_i \in \mathbb{R}^2$ such that $Q v_i \approx x_i$ and we have

$$Q^T A Q v_i \approx Q^T A x_i = \lambda_i Q^T x_i \approx \lambda_i Q^T Q v_i = \lambda_i v_i$$

  so that v_i is an eigenvector of $Q^T A Q$ with eigenvalue λ_i for $i = 1, 2$.
- Since Q is orthogonal and $Q^T A Q$ is upper triangular with the same eigenvalues as A, it seems that $Q(Q^T A Q)Q^T$ is a Schur decomposition of A.

Orthogonal iteration for general r

- When an arbitrary number r of eigenvalues is sought, the approximate eigenvectors are orthogonalized by performing a QR factorization, leading to the following pseudocode:

```
1. Sample a random matrix $Q_0 \in \mathbb{R}^{n \times r}$
2. $k := 0$
3. While not converged :
4.     $Y_{k+1} := AQ_k$
5.     Compute QR factorization $Q_{k+1}R_{k+1} = Y_{k+1}$
6.     $k := k + 1$
7. Return $\mathrm{diag}(Q_k^T AQ_k)$
```

  Similarly as with $r = 2$, this method converges to an upper-triangular matrix $Q_k^T AQ_k$ with eigenvalues $\lambda_1, \dots, \lambda_r$.

  Once the algorithm has converged, the approximate eigenvalues can be read from the diagonal of the Schur form $Q_k^T AQ_k$.

Convergence of orthogonal iteration for general r

- If A is symmetric, then the eigenvectors $x_1, \ldots, x_r$ are orthogonal, and the i-th column of Q_k, which we denote by $q_i^{(k)}$, converges to $\pm x_i$.

  For general matrices, things are different.
- Let us denote the matrices $Q^x \in \mathbb{R}^{n \times r}$ and $R^x \in \mathbb{R}^{r \times r}$ such that

$$[x_1 \, \ldots \, x_r] = Q^x R^x.$$

  We see that the iterate Q_k converges to Q^x:

  Since $q_1^{(k)}$ undergoes a normal power iteration, it converges to $x_1 = q_1^x$.

  For $q_2^{(k)}$, the QR decomposition ensures $\operatorname{span}\{x_1, x_2\} = \operatorname{span}\{q_1^x, q_2^x\}$ and we have

$$\operatorname{span}\{q_1^{(k)}, q_2^{(k)}\} \approx \operatorname{span}\{x_1, x_2\} = \operatorname{span}\{q_1^x, q_2^x\}$$

  Thus $q_2^{(k)}$ converges to something in the space $\operatorname{span}\{q_1^x, q_2^x\}$, and it also has to be orthogonal to $q_1^{(k)} \approx q_1^x$. Therefore $q_2^{(k)}$ has to converge to $\pm q_2^x$. Similarly, $q_i^{(k)}$ converges to $\pm q_i^x$. Overall, we have that Q_k converges to Q^x.

Convergence to the Schur decomposition

- Now that we know that Q_k converges to Q^x, we can analyze the matrix $Q_k^T A Q_k$, which converges to $Q^{xT} A Q^x$ up to some columnwise sign changes.
- Since $AX = X\Lambda$ where $X = [x_1, \dots, x_r]$ and $\Lambda = \mathrm{diag}(\lambda_1, \dots, \lambda_r)$, the definitions of Q^x and R^x imply that

$$\begin{aligned} AX &= X\Lambda \\ AQ^x R^x &= Q^x R^x \Lambda \\ Q^{xT} A Q^x R^x (R^x)^{-1} &= Q^{xT} Q^x R^x \Lambda (R^x)^{-1} \\ Q^{xT} A Q^x &= R^x \Lambda (R^x)^{-1}. \end{aligned}$$

  Since R^x is upper triangular and Λ is diagonal, we have that $R^x \Lambda (R^x)^{-1}$ is upper triangular.

  More particularly, we also have that $Q^{xT} A Q^x$ is upper triangular with the eigenvalues $\lambda_1, \dots, \lambda_r$ on the diagonal.

  Then, the matrix $Q_k^T A Q_k$ converges to $Q^{xT} A Q^x$, which is upper triangular and has the top r eigenvalues of A on the diagonal.

Convergence of orthogonal iteration

- The convergence analysis being sequential, i.e., we assumed $q_1^{(k)} \approx q_1^x$, then showed that $q_2^{(k)}$ converges to q_2^x, and so on; may lead to think that the convergence of orthogonal iteration is slow. I.e., we first have to wait that $q_1^{(k)}$ converges, then $q_2^{(k)}$, and so on.

  But, in fact, what actually happens is that all of the $q_i^{(k)}$ converge simultaneously.
- It can be shown that the convergence of the iterate Q_k to Q^x depends, similarly as before, on the separation between λ_r and λ_{r+1}. In particular, we have

$$\|Q_k Q_k^T - Q^x Q^{xT}\|_2 = \mathcal{O}\left(\left|\frac{\lambda_{r+1}}{\lambda_r}\right|^k\right).$$

  That is, the smaller $|\lambda_{r+1}/\lambda_r|$, the faster the convergence of Q_k to Q^x.

QR iteration

- QR iterations are a re-framing of orthogonal iterations with $r = n$.

  QR iterations yield the full Schur decomposition $T = Q^T AQ$ of A where T is an n-by-n upper triangular matrix with the eigenvalues of A on the diagonal, and Q is a n-by-n orthogonal matrix of a QR decomposition of the eigenvectors X of A.
- The iterate of QR iteration is denoted by Q_k with a corresponding matrix $T_k := Q_k^T AQ_k$.
- The formulation of QR iterations is more commonly expressed as a recurrence from $T_k = Q_k^T AQ_k$ to $T_{k+1} = Q_{k+1}^T AQ_{k+1}$.

  From the definition of orthogonal iterations, we have

  $$Q_{k+1}R_{k+1} = AQ_k \;\text{ so that }\; T_k = Q_k^T AQ_k = Q_k^T Q_{k+1}R_{k+1}$$

  and, since $r = n$, we have $Q_kQ_k^T = I_n$ and

  $$R_{k+1}Q_k^T = Q_{k+1}^T A \;\text{ so that }\; T_{k+1} = Q_{k+1}^T AQ_{k+1} = R_{k+1}Q_k^T Q_{k+1}$$

QR iteration, cont'd$_1$

Then, as we let $U_{k+1} := Q_k^T Q_{k+1}$, we have

$$T_k = U_{k+1} R_{k+1}$$
$$T_{k+1} = R_{k+1} U_{k+1}$$

where R_{k+1} is upper triangular, and U_{k+1} is orthogonal.
Note that $U_{k+1}R_{k+1}$ is a QR decomposition of $T_k = Q_k^T A Q_k$.
This yields the following pseudocode to compute the eigenvalues of A :

```
1. $T_0 := A$
2. $k := 0$
3. While not converged :
4.     Compute QR factorization $U_{k+1}R_{k+1} = T_k$
5.     $T_{k+1} := R_{k+1}U_{k+1}$
6.     $k := k + 1$
7. Return $\mathrm{diag}(T_k)$
```

Notice that A is only needed at the start of the algorithm, after what we only repeatedly compute QR decompositions and switch the factors.

QR iteration, cont'd$_2$

- In this algorithm, the matrix $U_{k+1} = Q_k^T Q_{k+1}$ represents an orthogonal correction.

  Since upon convergence $U_k \to I_n$, the determinant of U_k is 1 for large k, and we can interpret U_k as a small rotation on the orthogonal vectors in Q_k. In particular, we have:

  $$U_1 \dots U_{k+1} = Q_0^T Q_1 Q_1^T Q_2 \dots Q_k^T Q_{k+1} = Q_0 Q_{k+1} = Q_{k+1}$$

  because we chose $Q_0 = I_n$.

  As the algorithm converges, Q_k and Q_{k+1} become very close.
- In the symmetric case, $T_k = Q_k^T A Q_k$ is symmetric, but since it also converges to an upper symmetric matrix, it actually converges to a diagonal form, in which case the Schur decomposition is actually an eigendecomposition.

QR iteration, cont'd$_3$

- The QR iteration presented so far has drawbacks:
  - A QR factorization at cost $\mathcal{O}(n^3)$ is computed at each iteration.
  - The convergence depends heavily on the distribution of the eigenvalues, and it may never converge if two eigenvalues have the same magnitude.
- Improvements of the QR iteration method can be introduced to improve the robustness and efficiency:
  - The transformation of A into an upper Hessenberg form allows to decrease the cost of the QR factorizations.
  - A shifted version of the QR iteration can improve convergence, even when the eigenvalues are not well-separated, making the method robust to cases of eigenvalues with equal magnitudes.

Other methods and implementations

Section 5.2 in Darve & Wootters (2021)

Divide-and-conquer method

- A symmetric matrix can efficiently be transformed into a tridiagonal form using an orthogonal transformation

$$Q^T A Q = T.$$

  Then, the eigendecomposition of A can be obtained from that of T.

- The divide-and-conquer method splits the tridiagonal matrix into two tridiagonal blocks plus a rank-1 perturbation:

$$T = \begin{bmatrix} T_1 & \\ & T_2 \end{bmatrix} + \rho u u^T.$$

- The method proceeds as follows:
  1. Calculate the eigendecompositions of T_1 and T_2.
  2. The rank-1 perturbation allows to compute the eigenvalues of T given the eigendecompositions of T_1 and T_2.

Method of bissection

- The method of bissection also considers a tridiagonal form $Q^T A Q = T$.
- The eigenvalues of T are the roots of $p_n(\lambda) = \det(T - \lambda I_n)$.

  Finding these roots is generally a complex problem, but it can be simplified if we consider only the leading r-by-r block T_r of T and the corresponding characteristic polynomial

$$p_r(\lambda) = \det(T_r - \lambda I_r).$$

- As T is tridiagonal, it is possible to find a simple relation between p_r, p_{r-1} and p_{r-2}.

  Using this sequence of polynomials, the method of bissection is able to efficiently calculate the roots of p_n.

Existing implementations

- QR iteration:
  - Available for general matrices.
  - Implementation sometimes requires tridiagonalization.
  - Fastest to compute the eigendecomposition of small matrices ($n \leq 25$).
  - Algorithm behind the Matlab, NumPy and Julia functions.
  - Available in LAPACK as `ssyev` for dense symmetric matrices.
  - Available in LAPACK as `sstev` for symmetric tridiagonal matrices.
- Divide-and-conquer method:
  - Available for symmetric matrices.
  - Implementation requires tridiagonalization.
  - Fastest to compute the eigendecomposition of medium size tridiagonal matrices, i.e., for $n > 25$.
  - Available in LAPACK as `sstevd` for symmetric tridiagonal matrices, `sstevd` defaults to QR iteration for smaller matrices.
- Method of bissection:
  - Available in LAPACK as `ssyevx` for dense symmetric matrices.

Homework problems

Homework problem

Turn in **your own** solution to **Pb. 20**:

Pb. 19 Show that the gradient of the Rayleigh quotient of a symmetric matrix A given by

$$r(x) = \frac{x^T Ax}{x^T x} \text{ for } x \neq 0$$

is $\nabla r(x) = \frac{2}{x^T x}(Ax - r(x)x)$.

Pb. 20 Let (λ, x) be a right eigen-pair of A, (μ, y) be a left eigen-pair of A and μ be distinct from λ. Show that x and y are orthogonal.