$\boxed{35}$ Let $A \in \mathbb{R}^{n\times n}$ be a symmetric matrix. Show that for $M_\tau = I - \tau A$ with $\tau \in \mathbb{R}$:

$$\|M_\tau\|_{\ell_2} \;=\; \max_{\lambda\in\sigma(A)} |1 - \tau\,\lambda| \;=\; q(\tau),$$

where $q(\tau) = \max\big(|1-\tau\,\lambda_{\max}(A)|,\ |1-\tau\,\lambda_{\min}(A)|\big)$.

$\boxed{36}$ Let $A \in \mathbb{R}^{n\times n}$ be a symmetric matrix with at least one negative and one positive eigenvalue (A is indefinite). Show that

$$\max_{\lambda\in\sigma(A)} |1-\tau\,\lambda| \;>\; 1 \qquad \forall \tau \neq 0.$$

$\boxed{37}$ Let $A \in \mathbb{R}^{n\times n}$ be a symmetric matrix and let $\lambda_- < 0$ and $\lambda_+ > 0$ be two eigenvalues with the corresponding eigenvectors e_- and e_+, respectively. Show that there is *no choice* for the parameter $\tau \in \mathbb{R}$ such that the following Richardson's method (with special initial value) converges.

$$\begin{aligned} x_0 &= x + e_- + e_+ \\ x_{k+1} &= x_k + \tau(b - Ax_k) \qquad \text{for } k = 0, 1, \ldots \end{aligned}$$

Here $x = A^{-1}b$ is the exact solution.

$\boxed{38}$ In the lecture, we showed for the CG method that

$$x_k \;\in\; x_0 + \mathcal{K}_k(A, r_0) \quad \text{and} \quad r_k \perp \mathcal{K}_k(A, r_0),$$

where $\perp$ means orthogonality in the corresponding inner product.

We consider now the GMRES method, which can also be applied to indefinite matrices. There, the iterates are constructed such that

$$x_k \;\in\; x_0 + \mathcal{K}_k(A, r_0) \quad \text{and} \quad \|b - Ax_k\|_{\ell_2} \;=\; \min_{y\in x_0+\mathcal{K}_k(A,r_0)} \|b - Ay\|_{\ell_2}\,.$$

Show that in this case

$$r_k \perp A\,\mathcal{K}_k(A, r_0),$$

where $\perp$ means ℓ_2-orthogonality.
Hint: Rewrite the above minimization problem as an equivalent variational problem.

Programming:

$\boxed{39}$ Write a function `CG(↓A, ↕x, ↓b, ↓C, ↕max_iter, ↕tol)` to solve the linear system

$$A\,\underline{x} \;=\; \underline{b}$$

with the preconditioned CG method. The parameter specification is the same as in Exercise $\boxed{33}$ (Richardson's method).
Hint: Use `cg.hh` (download or see next page) and rewrite it for your own purposes.

$\boxed{40}$ Solve the problem given in Exercise $\boxed{28*}$ (see Tutorial 5) with the Jacobi-preconditioned CG method. Try different equidistant meshes ($h = 1/10$, $h = 1/20$, $h = 1/100$, etc.) and report the number of iterations to reach the relative accuracy $\varepsilon = 10^{-6}$.

File cg.hh

```
#ifndef __CG_H
#define __CG_H

// Iterative template routine -- CG
//
// RICHARDSON solves the symmetric positive definite linear
// system Ax=b using the preconditioned conjugate gradient methd.
// The returned value indicates convergence within
// max_iter iterations (return value 0)
// or no convergence within max_iter iterations (return value 1)
// Upon successful return (0), the output arguments have the
// following values:
//        x: computed solution
// mat_iter: number of iterations to satisfy the stopping criterion
//      tol: residual after the final iteration

template <class MATRIX, class VECTOR, class PRECONDITIONER, class REAL>
int
CG (const MATRIX & A, VECTOR & x, const VECTOR & b,
    const PRECONDITIONER & M, int & max_iter, REAL & tol)
{
  REAL resid;
  VECTOR p(b.size ());
  VECTOR z(b.size ());
  VECTOR q(b.size ());
  REAL alpha, beta, rho, rho_1;
  REAL normb = norm (b);
  VECTOR r = b - A * x;

  if (normb == 0.0) normb = 1;
  resid = norm (r) / normb;

  if (resid <= tol)
    {
      tol = resid;
      max_iter = 0;
      return 0;
    }

  for (int i=1; i<=max_iter; i++)
    {
      z = M.solve (r);
      rho = inner_product (r, z);

      if (i==1)
        {
          p = z;
        }
      else
        {
          beta = rho / rho_1;
          p = z + beta * p;
```

```
        }

      q = A * p;
      alpha = rho / inner_product (p, q);

      x += alpha * p;
      r -= alpha * q;

      resid = norm(r) / normb;

      if (resid <= tol)
        {
          tol = resid;
          max_iter = i;
          return 0;
        }

      rho_1 = rho;
    }

  tol = resid;
  return 1;
}
```