Elements of Mathematics

Exercise Sheet 11

Submission due date: **25.01.2022, 10:15h**

THEORY

1 Convergence Speed of Linear Iterations

Let $M \in \mathbb{R}^{n\times n}$ be *symmetric* with $\rho(M) < 1$, let $N \in \mathbb{R}^{n\times n}$ and $x_0, b \in \mathbb{R}^n$. Consider the fixed point iteration

$$x_{k+1} = Mx_k + Nb$$

and show the following convergence result

$$\|x_k - x^*\|_2 \leq \rho(M)^k \|x_0 - x^*\|_2,$$

where x^* is the associated fixed point. Thus, the smaller the spectral radius, the faster does the method converge.

Hint: For symmetric matrices $M \in \mathbb{R}^{n\times n}$ you can use $\|Mx\|_2 \leq \rho(M)\|x\|_2$ for all $x \in \mathbb{R}^n$.

(6 Points)

Solution:

Since $\rho(M) < 1$, the iteration converges to the fixed point $x^* = Mx^* + Nb$. We use this representation in the formulas. We find

$$\|x^k - x^*\|_2 = \|Mx^{k-1} + Nb - (Mx^* + Nb)\|_2 = \|M(x^{k-1} - x^*)\|_2 \overset{\text{[Hint]}}{\leq} \rho(M) \underbrace{\|x^{k-1} - x^*\|_2}_{\leq \rho(M)\|x^{k-2} - x^*\|}.$$

By inserting the iteration instruction repeatedly we ultimately arrive at

$$\|x^k - x^*\|_2 \leq \rho(M)^k \|x^0 - x^*\|_2.$$

2 Richardson Iteration

Let $A \in \mathbb{R}^{n\times n}$ be a symmetric and positive definite matrix. Consider the (relaxed) Richardson iteration

$$x_{k+1} = (I - \theta A)x_k + \theta b$$

with (symmetric) iteration matrix $M_\theta := I - \theta A$.

1. Let λ_{max} (λ_{min}) denote the largest (smallest) eigenvalue of A. Show that the spectral radius of M_θ is given by
$$\rho(M_\theta) = \max\{|1 - \theta\lambda_{max}|,\ |1 - \theta\lambda_{min}|\}.$$
Hint: Note that A has only *positive* eigenvalues. Determine the spectrum of M_θ by observing that it is a scaled and then shifted version of A.

2. Determine all $\theta \in \mathbb{R}$ for which the Richardson iteration converges.
   Hint: Use 1. and find all $\theta \in \mathbb{R}$ for which $\rho(M_\theta) < 1$.

3. Draw the function $\theta \mapsto \rho(M_\theta)$ to determine the optimal $\hat{\theta}$, which fulfills

$$\rho(M_{\hat{\theta}}) \le \rho(M_\theta)$$

   for all $\theta \in \mathbb{R}$.
   Hint: We find $\hat{\theta} = \frac{2}{\lambda_{max} + \lambda_{min}}$.

4. Show that for the optimal relaxation parameter $\hat{\theta}$ it holds that

$$\rho(M_{\hat{\theta}}) = \frac{\text{cond}(A) - 1}{\text{cond}(A) + 1},$$

   where $\text{cond}(A) := \frac{\lambda_{max}}{\lambda_{min}}$ denotes the condition number of the symmetric matrix A. Consider the previous exercise: What impact does a large condition number (i.e., $\text{cond}(A) \gg 1$) have on the convergence speed?

(8 Points)

Solution:

We have $x^{k+1} = \underbrace{(I - \theta A)}_{=:M_\theta} x^k + \theta b = M_\theta x^k + \theta b$.

1. By definition we have $\rho(M_\theta) = \max_{\lambda \in \sigma(M_\theta)} |\lambda|$.

   (a) Thus we first determine the spectrum of $M_\theta = I - \theta A$:

$$\begin{aligned} \lambda \in \sigma(A) &\overset{[\text{scaling:}]\cdot(-\theta)}{\Rightarrow} -\theta\lambda \in \sigma(-\theta A) \\ &\overset{[\text{shift:}]+1}{\Rightarrow} 1 - \theta\lambda \in \sigma(I - \theta A) \end{aligned}$$

$$\Rightarrow \quad \sigma(M_\theta) = \{1 - \theta\lambda : \lambda \in \sigma(A)\}$$

   (b) Determine the maximum of this set:
   By definition of λ_{max}, λ_{min} and $\sigma(A) \subset (0, +\infty)$, so that we obtain

$$\begin{aligned} & 1 - \theta\lambda_{\text{max}} \le 1 - \theta\lambda \le 1 - \theta\lambda_{\text{min}} && \forall \lambda \in \sigma(A) \\ \Rightarrow\ & |1 - \theta\lambda| \le \max\{|1 - \theta\lambda_{\text{max}}|,\ |1 - \theta\lambda_{\text{min}}|\} && \forall \lambda \in \sigma(A) \\ \Rightarrow\ & \rho(M_\theta) = \max\{|1 - \theta\lambda_{\text{max}}|,\ |1 - \theta\lambda_{\text{min}}|\}. \end{aligned}$$

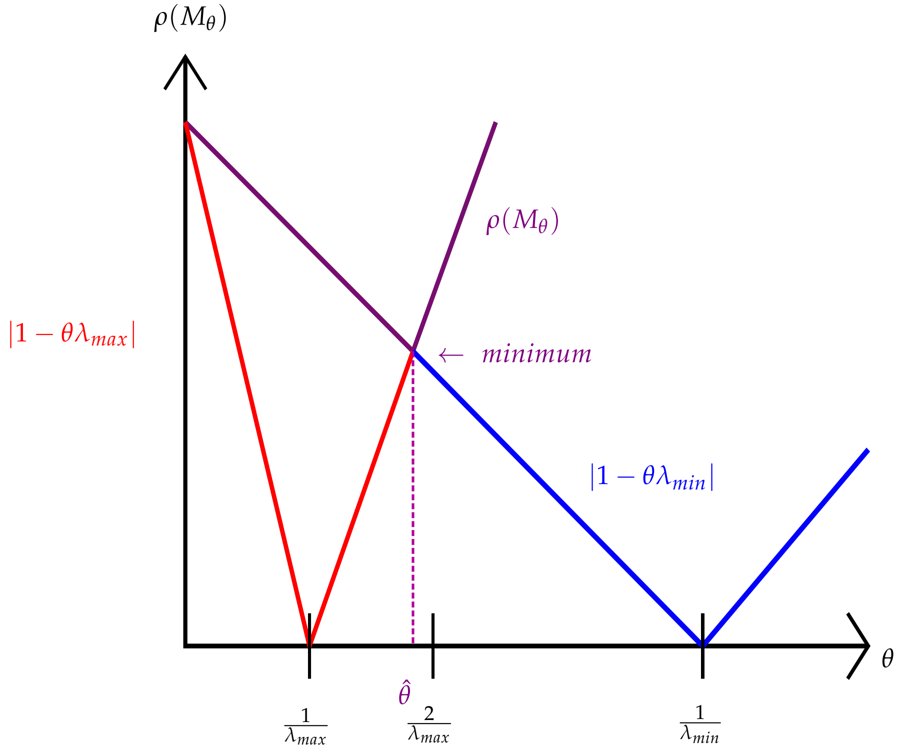
2.

$$\begin{aligned} & \rho(M_\theta) = \max\{|1 - \theta\lambda_{\text{max}}|,\ |1 - \theta\lambda_{\text{min}}|\} < 1 \\ \Leftrightarrow \quad & -1 \overset{a)}{<} 1 - \theta\lambda_{\text{max}} \le 1 - \theta\lambda_{\text{min}} \overset{b)}{<} 1 \\ a) \quad \Leftrightarrow \quad & -2 < -\theta\lambda_{\text{max}} \quad \Leftrightarrow \quad \theta < \frac{2}{\lambda_{\text{max}}} \\ b) \quad \Leftrightarrow \quad & -\theta\lambda_{\text{min}} < 0 \quad \Leftrightarrow \quad \theta > 0 \end{aligned}$$

   All in all:

$$\rho(M_\theta) < 1 \quad \Leftrightarrow \quad \theta \in \left(0, \frac{2}{\lambda_{\text{max}}}\right)$$

3. Let us plot $\theta \mapsto \rho(M_\theta) = \max\{|1-\theta\lambda_{\max}|,\ |1-\theta\lambda_{\min}|\}$



Thus: $\hat{\theta}$ is determined by the intersection

$$
\begin{aligned}
&{}''/'' \qquad -(1-\theta\lambda_{\max}) \overset{!}{=} 1-\theta\lambda_{\min} \qquad ''\backslash'' \\
&\Leftrightarrow \quad \theta\lambda_{\max} - 1 = 1 - \theta\lambda_{\min} \\
&\Leftrightarrow \quad \theta = \frac{2}{\lambda_{\min}+\lambda_{\max}}
\end{aligned}
$$

4. Inserting 3. into 2. gives

$$
\begin{aligned}
\rho(M_\theta) &= |1-\hat{\theta}\lambda_{\min}| \quad (= |1-\hat{\theta}\lambda_{\max}|) \\
&= \left|1 - \frac{2}{\lambda_{\min}+\lambda_{\max}}\lambda_{\min}\right| \\
&= \left|\frac{\lambda_{\max}-\lambda_{\min}}{\lambda_{\min}+\lambda_{\max}}\right| \\
&= \frac{\lambda_{\max}-\lambda_{\min}}{\lambda_{\min}+\lambda_{\max}} \\
&= \frac{\lambda_{\min}\left(\frac{\lambda_{\max}}{\lambda_{\min}}-1\right)}{\lambda_{\min}\left(1+\frac{\lambda_{\max}}{\lambda_{\min}}\right)} \\
&= \frac{\mathrm{cond}(A)-1}{\mathrm{cond}(A)+1}.
\end{aligned}
$$

Thus:

$$
\begin{aligned}
&(1)\ \mathrm{cond}(A) \gg 1 \quad \Rightarrow \quad \rho(M_\theta) \approx 1 \quad \rightarrow \quad \text{slow convergence} \\
&(2)\ \mathrm{cond}(A) \approx 1 \quad \Rightarrow \quad \rho(M_\theta) \approx 0 \quad \rightarrow \quad \text{fast convergence}
\end{aligned}
$$

3 Weighted Jacobi, Gauß–Seidel and Sucessive Over–Relaxation

9 Bonus points

Let $A \in \mathbb{R}^{n\times n}$ be a matrix with nonzero diagonal entries $a_{ii} \neq 0$ and consider the splitting $A = L + D + U$ into lower triangular, diagonal and upper triangular part of A. Also recall that splitting methods are of the form

$$x^{k+1} = (I - NA)x^k + Nb,$$

where the significant matrix $M := I - NA$ is called iteration matrix.

Show the following:

1. **Weighted Jacobi:** $N = \theta D^{-1}$
   - For the iteration matrix we find
   $$M_{Jac} := I - \theta D^{-1}A = (1-\theta)I - \theta D^{-1}(L+U)$$
   - The i-the component of $x^{k+1} = (I - NA)x^k + Nb$ is given by
   $$x_i^{k+1} = (1-\theta)x_i^k + \frac{\theta}{a_{ii}}\left(b_i - \sum_{j\neq i} a_{ij}\, x_j^{k+1}\right).$$
2. **Gauß–Seidel:** $N = (L+D)^{-1}$
   - For the iteration matrix we find
   $$M_{GS} := I - (L+D)^{-1}A = -(L+D)^{-1}U$$
   - The i-the component of $x^{k+1} = (I - NA)x^k + Nb$ is given by
   $$x_i^{k+1} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}\, x_j^{k+1} - \sum_{j=i+1}^{n} a_{ij}\, x_j^{k}\right).$$
3. **Sucessive Over–Relaxation (variant of Gauß-Seidel):** $N = \theta \cdot (\theta L + D)^{-1}$
   - For the iteration matrix we find
   $$M_{SOR} := I - \theta(\theta L + D)^{-1}A = (\theta L + D)^{-1}((1-\theta)D - \theta U).$$
   - The i-the component of $x^{k+1} = (I - NA)x^k + Nb$ is given by
   $$x_i^{k+1} = (1-\theta)x_i^k + \frac{\theta}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}\, x_j^{k+1} - \sum_{j=i+1}^{n} a_{ij}\, x_j^{k}\right).$$

   Remark: We observe that SOR for $\theta = 1$ is Gauß–Seidel and otherwise is a combination of the previous step x^k and the Gauß-Seidel update. For spd matrices it allows for $\theta > 1$ which is why it is called over–relaxation.

Hint: For 2. and 3. cast the formulas into the form $x^{k+1} = T^{-1}w$ for some lower triangular matrix T and some vector w and then use forward substitution.

Solution:

We first recall the forward substitution formula for inverting lower triangular matrices: Let $T = (\ell_{ij}) \in \mathbb{R}^{n\times n}$ be triangular and $w \in \mathbb{R}^n$, then the i-th component of $z = T^{-1}w$ is given by

$$z_i = \frac{1}{\ell_{ii}}\left(w_i - \sum_{j=1}^{i-1} \ell_{ij} z_j\right), \quad i \in \{1, \dots, n\}.$$

1. **Jacobi:**
   - Using the splitting $A = L + D + U$ we find

     $$M_{Jac} := I - \theta D^{-1}A = I - \theta D^{-1}(L + D + U) = I - \theta(I + D^{-1}(L + U)) = (1-\theta)I - \theta D^{-1}(L+U)$$

   - The inverse of D is $D^{-1} = \operatorname{diag}(\frac{1}{a_{11}}, \dots, \frac{1}{a_{nn}})$. Thus the i-th component of $\theta D^{-1}b$ is given by $\theta\frac{b_i}{a_{ii}}$. Now applying the definition of the matrix vector product we find for the i-th component of $\theta D^{-1}Ax^k$ that $\theta\frac{1}{a_{ii}}\sum_{j=1}^n a_{ij}x_j^k$. Combining this we obtain for the i-th of the Jacobi iterate the searched formula

     $$\begin{aligned} x_i^{k+1} = x_i^k - \theta\frac{1}{a_{ii}}\sum_{j=1}^n a_{ij}x_j^k + \theta\frac{b_i}{a_{ii}} &= x_i^k - \theta x_i^k - \theta\frac{1}{a_{ii}}\sum_{j\neq i} a_{ij}x_j^k + \theta\frac{b_i}{a_{ii}} \\ &= (1-\theta)x_i^k + \frac{\theta}{a_{ii}}\left(b_i - \sum_{j\neq i} a_{ij}\, x_j^{k+1}\right). \end{aligned}$$

2. **Gauß–Seidel**
   - Using the splitting $A = L + D + U$ we find

     $$M_{GS} := I - (L+D)^{-1}A = I - (L+D)^{-1}(L+D+U) = I - (I + (L+D)^{-1}U) = -(L+D)^{-1}U$$

   - We cast the formula into a lower triangular system:

     $$\begin{aligned} x^{k+1} &= (I - NA)x^k + Nb = M_{GS}x^k + Nb \\ &= -(L+D)^{-1}Ux^k + (L+D)^{-1}b \\ &= (L+D)^{-1}(b - Ux^k) \end{aligned}$$

     Now we consider $z = x^{k+1}$, $T = (L+D)$ and $w = b - Ux^k$ and apply forward substitution to obtain

     $$\begin{aligned} x_i^{k+1} = z_i &= \frac{1}{\ell_{ii}}\left(w_i - \sum_{j=1}^{i-1}\ell_{ij}z_j\right), \quad i \in \{1, \dots, n\} \\ &= \frac{1}{a_{ii}}\left(b_i - \sum_{j=i+1}^{n} a_{ij}x_j^k - \sum_{j=1}^{i-1} a_{ij}x_j^{k+1}\right), \quad i \in \{1, \dots, n\}. \end{aligned}$$

3. **SOR**
   - We use

     $$N = \theta \cdot (\theta L + D)^{-1} = \left(\tfrac{1}{\theta}\right)^{-1}(\theta L + D)^{-1} = \left(L + \tfrac{1}{\theta}D\right)^{-1}$$

     and the splitting

     $$A = L + D + U = L + D + U \pm \tfrac{1}{\theta}D = (L + \tfrac{1}{\theta}D) + U + (1 - \tfrac{1}{\theta})D.$$

Then we find

$$
\begin{aligned}
M_{SOR} := I - NA = I - \left(L + \tfrac{1}{\theta}D\right)^{-1}\left((L + \tfrac{1}{\theta}D) + U + (1 - \tfrac{1}{\theta})D\right) \\
= -\left(L + \tfrac{1}{\theta}D\right)^{-1}\left(U + (1 - \tfrac{1}{\theta})D\right) \\
= \left(L + \tfrac{1}{\theta}D\right)^{-1}\left(\tfrac{1-\theta}{\theta}D - U\right) \\
= \theta \cdot (\theta L + D)^{-1}\left(\tfrac{1-\theta}{\theta}D - U\right) \\
= (\theta L + D)^{-1}\left((1-\theta)D - \theta U\right).
\end{aligned}
$$

- We cast the formula into a lower triangular system:

$$
\begin{aligned}
x^{k+1} &= (I - NA)x^k + Nb = M_{SOR}x^k + Nb \\
&= (\theta L + D)^{-1}\left((1-\theta)D - \theta U\right)x^k + \theta \cdot (\theta L + D)^{-1}b \\
&= (\theta L + D)^{-1}(\theta b - ((1-\theta)D - \theta U)x^k)
\end{aligned}
$$

Now we consider $z = x^{k+1}$, $T = (\theta L + D)$ and $w = \theta b - (1-\theta)Dx^k - \theta U x^k$ and apply forward substitution to obtain

$$
\begin{aligned}
x_i^{k+1} = z_i &= \frac{1}{\ell_{ii}}\left(w_i - \sum_{j=1}^{i-1}\ell_{ij}z_j\right), \quad i \in \{1,\dots,n\} \\
&= \frac{1}{a_{ii}}\left(\theta b_i - (1-\theta)a_{ii}x_i^k - \theta\sum_{j=i+1}^{n}a_{ij}x_j^k - \sum_{j=1}^{i-1}\theta a_{ij}x_j^{k+1}\right), \quad i \in \{1,\dots,n\} \\
&= (1-\theta)x_i^k + \frac{\theta}{a_{ii}}\left(b_i - \sum_{j=i+1}^{n}a_{ij}x_j^k - \sum_{j=1}^{i-1}a_{ij}\,x_j^{k+1}\right), \quad i \in \{1,\dots,n\}.
\end{aligned}
$$

PROGRAMMING

---

4 Splitting Methods: relax. Richardson, relax. Jacobi, Gauß-Seidel and SOR

1. Implement a function

```
x, error, numiter = iter_solve(A, b, x0, method="Jacobi", theta=.1, tol=1e-08, maxiter=50)
```

which takes as arguments

- A : a matrix $A \in \mathbb{R}^{n\times n}$
- b : a vector $b \in \mathbb{R}^n$
- x0 : an initial guess $x^0 \in \mathbb{R}^n$
- `method` : optional parameter to choose between relax. Richardson, weighted Jacobi, Gauß-Seidel and SOR and which is set to `"Jacobi"` by default
- theta : relaxation parameter θ which is set to 0.1 by default
(note: Gauß–Seidel is SOR with `theta=1.0`)
- tol : error tolerance as float, which is set to 10^{-8} by default
- maxiter : maximum number of iterations, which is set to 50 by default

and then solves the system $Ax = b$ by applying the specified iterative scheme. It shall then return

- x : list of all iterates x^k

- error : list containing all residuals $\|Ax^k - b\|_2$
- numiter : number of iterations that have been performed

The iteration shall break if the residual is tolerably small, i.e.,

$$\|Ax^k - b\|_2 < \texttt{tol}$$

or the maximum number of iterations maxiter has been reached.

Hint: Implement the element-based formulas for the Jacobi, Gauß-Seidel and SOR method (see previous exercise).

2. **2d:** Test <u>all</u> methods on the following two-dimensional setting:

$$A = 4\begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}, \quad b = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

What is the exact solution x^*? Play around with the parameters x0, theta, tol and maxiter. Also create the following two plots for one fixed setting:

- Plot the error $\|Ax^k - b\|_2$ for each iterate $x^k \in \mathbb{R}^2$, $k = 1, \dots, m$, for <u>all</u> methods into one plot (use different colors).
- Plot the iterates $x^k \in \mathbb{R}^2$, $k = 1, \dots, m$, themselves for all methods into one plot (use different colors).

3. **nd:** Next, test all methods on the higher–dimensional analogue

$$A = n^2 \begin{pmatrix} 2 & -1 & 0 & \dots & 0 \\ -1 & 2 & -1 & & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & -1 & 2 & -1 \\ 0 & \dots & 0 & -1 & 2 \end{pmatrix} \in \mathbb{R}^{n \times n},$$

for different dimensions $n \in \mathbb{N}$ and data $b, x^0 \in \mathbb{R}^n$ of your choice. Play around with the parameters.

Hint: Of course, it can happen that the iterations do not converge. Use small values for θ when you use the Richardson iteration. This will assure that $\rho(I - NA) < 1$. *(9 Points)*

Solution:

```
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")


# ------------------------------------------------------------#
#                      ITERATIVE SOLVER
# ------------------------------------------------------------#
def Richardson_step(A, b, theta, x):
    return x - theta * (A @ x - b)


def Jacobi_step(A, b, theta, x):
    n = len(x)
    xnew = np.zeros(n)
    for i in range(n):
        s1 = np.dot(A[i, :i], x[:i])
        s2 = np.dot(A[i, i + 1:], x[i + 1:])
```

```
        xnew[i] = (1. - theta) * x[i] + theta / A[i, i] * (b[i] - s1 - s2)
    return xnew


def SOR_step(A, b, theta, x):
    n = len(x)
    xnew = np.zeros(n)
    for i in range(n):
        s1 = np.dot(A[i, :i], xnew[:i])  # <-- here we use already the new info
        s2 = np.dot(A[i, i + 1:], x[i + 1:])
        xnew[i] = (1. - theta) * x[i] + theta / A[i, i] * (b[i] - s1 - s2)
    return xnew


def steepestDescent_step(A, b, theta, x):
    r = A@x - b
    theta = np.dot(r, r) / np.dot(A@r, r)
    return x - theta * (A @ x - b)


def conjugateGradient(A, b, x0, maxiter=50, tol=1e-8):
    X = [x0]
    n = len(x0)
    error = []
    r = b - A @ X[0]
    p = r
    alpha_alt = np.dot(r, r)
    for numiter in range(max(maxiter, n)):
        error += [np.linalg.norm(A.dot(X[-1]) - b)]
        if error[-1] < tol:
            return X, error, numiter
        v = A @ p
        lambd = alpha_alt / np.dot(v, p)
        X += [X[-1] + lambd * p]
        r = r - lambd * v
        alpha_neu = np.dot(r, r)
        p = r + alpha_neu/alpha_alt * p
        alpha_alt = alpha_neu
    return X, error, numiter


def iter_solve(A, b, x0, method="Jacobi", theta=.1, maxiter=50, tol=1e-8):
    """
    solves a system Ax = b, where A is assumed to be invertible,
    with relaxed splitting methods: Jacobi, Richardson

    Parameters
    ----------
    A : (n, n) numpy.ndarray
         system matrix
    b : (n,) numpy.ndarray
         right-hand side
    x0: (n,) numpy.ndarray
         initial guess
    method : string
              indicates method: "Jacobi" (=default), "Richardson", "GS", "SOR"
    theta : number (int or float)
            relaxation parameter (step length) default theta = 0.1
    tol : number (float)
            error tolerance, iteration stops if ||Ax-b|| < tol
    maxiter : int
        number of iterations that are performed , default m=50
```

```python
    Returns
    -------
    X : list of length N (=m or less), containing iterates
        columns represent iterates from x_0 to x_(N-1)
    error : list of length numiter containing norm of all residuals
    numiter : integer indicating how many iterations have been performed
    """
    X = [x0]
    error = []
    if method in ["GS", "SOR", "Jacobi", "steepestDescent"] and \
       np.prod(A.diagonal()) == 0:
        print(f"WARNING: Method was chosen to be {method} \
              but A has zero diagonal entries!")
        return None
    elif method == "GS":
        theta = 1.0
        method = "SOR"
    elif method == "CG":
        return conjugateGradient(A, b, x0, maxiter=maxiter, tol=tol)
    # choose the function to compute the iteration instruction
    # according to method
    stepInstructionDictionary = {"Jacobi": Jacobi_step,
                                 "Richardson": Richardson_step,
                                 "SOR": SOR_step,
                                 "steepestDescent": steepestDescent_step}
    stepInstruction = stepInstructionDictionary[method]

    # ITERATION
    for numiter in range(maxiter):
        error += [np.linalg.norm(A.dot(X[-1]) - b)]
        if error[-1] < tol:
            return X, error, numiter
        X += [stepInstruction(A, b, theta, X[-1])]

    return X, error, numiter


def main(A, b, x0, maxiter, methThet, plot=False, verbose=False):

    X = np.zeros((maxiter, 2, len(methThet)))
    colors = ['r', 'g', 'b', "y", "c", "m"]
    # ----------------------------------------------------------------#
    #                     PLOT error and iterates
    # ----------------------------------------------------------------#
    if plot:
        plt.figure()

    for i, method in enumerate(methThet):
        X, error, numiter = iter_solve(A, b, x0, method=method,
                                       theta=methThet[method],
                                       maxiter=maxiter)
        if verbose:
            print(method, "\n \t\t  >", f"NumIter = {numiter}",
                  f"\t residual = {error[-1]:0.2e}")
        if plot:
            plt.subplot(1, 2, 1)
            plt.plot(error, colors[i]+"-x")
            plt.title("Residual $||Ax_k - b||_2$")
            plt.legend(method)
            plt.subplot(1, 2, 2)
            X = np.array(X)
```

```python
            plt.plot(X[:, 0], X[:, 1], colors[i] + "o-")
            plt.legend(list(methThet.keys()))
            plt.title("Iterates $x_k$")
            plt.axis("equal")
    if plot:
        plt.show()
        plt.axis("equal")
    return X, error, numiter


if __name__ == "__main__":
    # ---------------------------------------------------------------#
    #       2d EXAMPLE
    # ---------------------------------------------------------------#
    A = 4. * np.array([[2, -1],
                       [-1, 2]])
    b = np.zeros(2)
    x0 = np.array([5, 8])
    maxiter = 100
    methThet = {"Richardson": 0.1, "Jacobi": 1, "GS": 1, "SOR": 1.2,
                "steepestDescent": -1, "CG": -1}
    print("-"*40+"\n 2d EXAMPLE (numiter, error) \n"+"-"*40)
    X, error, numiter = main(A, b, x0, maxiter,
                             methThet, plot=True, verbose=True)

    # ---------------------------------------------------------------#
    #               HIGHER DIMENSIONAL EXAMPLE
    # ---------------------------------------------------------------#
    n = 100  # 10000 # 100000
    A = n ** 2 * (2 * np.eye(n, k=0) - np.eye(n, k=1) - np.eye(n, k=-1))
    b = np.random.rand(n)
    x0 = np.random.rand(n)  # b#*100#
    firstMmethods = 4
    maxiter = 50
    methThet = {"Richardson": 0.00001, "Jacobi": 0.5, "GS": 1, "SOR": 1.9,
                "steepestDescent": 1}
    print("-"*40 + "\n nd EXAMPLE with n = {}\n".format(n) + "-"*40)
    X, error, numiter = main(A, b, x0, maxiter,
                             methThet, plot=0, verbose=True)
```

Total Number of Points = 23 (T:14, P:9)