Home assignment 2

Numerical Optimization and its Applications - Spring 2019
Gil Ben Shalom, 301908877
Tom Yaacov, 305578239

May 10, 2019

1 The efficiency of different iterative methods for solving a linear system

(a) Following are the implementation for the four methods:
Jacobi:

```
from numpy import diag, matmul, array
from numpy.linalg import inv, norm


def weighted_jacobi(A, b, x_0, maxIter, epsilon, w):
    D = diag(diag(A))
    x = x_0
    res = [norm(matmul(A, x) - b)]
    for i in range(maxIter):
        x = x + w * matmul(inv(D), b - matmul(A, x))
        res.append(norm(matmul(A, x) - b))
        if res[-1] / norm(b) < epsilon:
            break
    return x, array(res)
```

Gauss-Seidel:

```python
from numpy import matmul, tril, array
from numpy.linalg import inv, norm


def weighted_gauss_seidel(A, b, x_0, maxIter, epsilon, w):
    L_D = tril(A, k=0)
    x = x_0
    res = [norm(matmul(A, x) - b)]
    for i in range(maxIter):
        x = x + w * matmul(inv(L_D), b - matmul(A, x))
        res.append(norm(matmul(A, x) - b))
        if res[-1] / norm(b) < epsilon:
            break
    return x, array(res)
```

Steepest Descent:

```python
from numpy import matmul, array, dot, copy
from numpy.linalg import norm
from py_files.utils import is_pos_def


def steepest_decent(A, b, x_0, max_iter, epsilon):
    if not is_pos_def(A):
        print("matrix is not SPD, can't solve using steepest decent...")
        return None, None
    x = copy(x_0)
    r = b - matmul(A, x)
    all_r = [norm(r)]
    for k in range(max_iter):
        alph = dot(r, r) / dot(r, matmul(A, r))
        x = x + alph * r
        # res.append(norm(matmul(A, x) - b))
        r = b - matmul(A, x)
        all_r.append(norm(r))
        if norm(r) / norm(b) < epsilon:
            break
    return x, array(all_r)
```

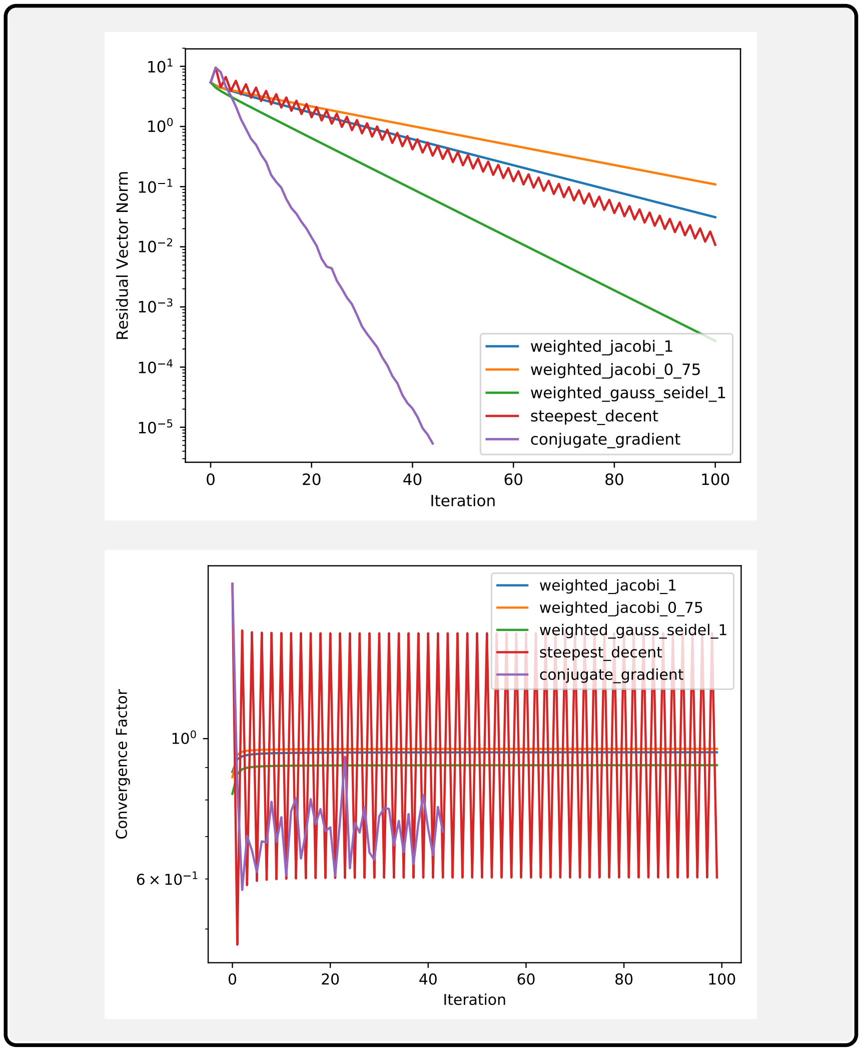
Conjugate Gradient

```python
from numpy import matmul, array, dot, copy
from numpy.linalg import norm
from py_files.utils import is_pos_def


def conjugate_gradient(A, b, x_0, max_iter, epsilon):
    if not is_pos_def(A):
        print("matrix is not SPD, can't solve using steepest decent...")
        return None, None
    x = copy(x_0)
    r = b - matmul(A, x)
    p = r
    all_r = [norm(r)]
    for k in range(max_iter):
        alph = dot(r, p) / dot(p, matmul(A, p))
        x = x + alph * p
        # res.append(norm(matmul(A, x) - b))
        r_prev = copy(r)
        r = b - matmul(A, x)
        all_r.append(norm(r))
        if norm(r) / norm(b) < epsilon:
            break
        beta = dot(r, r) / dot(r_prev, r_prev)
        p = r + beta * p
    return x, array(all_r)
```

(b) Following are the system and parameters definition, methods calls, residual vector norm and convergence factor plotting:

```python
import numpy as np
from scipy.sparse import spdiags
import matplotlib.pyplot as plt
from py_files.part_1_gauss_seidel import weighted_gauss_seidel
from py_files.part_1_jacobi import weighted_jacobi
from py_files.part_1_sd import steepest_decent
from py_files.part_1_cg import conjugate_gradient

n = 100
# TODO: not sure if .toarray() is the right approach
A = spdiags(np.array([-np.ones(n), 2.1 * np.ones(n), -np.ones(n)]),
            np.array([-1, 0, 1]), n, n).toarray()
x_0 = np.zeros(n)
b = np.random.rand(n)
```

```
maxIter = 100
epsilon = 1e-6

res = dict()
x, res['weighted_jacobi_1'] = weighted_jacobi(A, b, x_0, maxIter, epsilon, 1)
#print('weighted_jacobi_1 result:', x)
x, res['weighted_jacobi_0_75'] = weighted_jacobi(A, b, x_0, maxIter, epsilon, 0.75)
#print('weighted_jacobi_0_75 result:', x)
x, res['weighted_gauss_seidel_1'] = weighted_gauss_seidel(A, b, x_0, maxIter, epsilo
#print('weighted_gauss_seidel_1 result:', x)
x, res['steepest_decent'] = steepest_decent(A, b, x_0, maxIter, epsilon)
#print('steepest_decent result:', x)
x, res['conjugate_gradient'] = conjugate_gradient(A, b, x_0, maxIter, epsilon)
#print('conjugate_gradient result:', x)

convergence_factor = dict()
for alg_res in res:
    convergence_factor[alg_res] = res[alg_res][1:] / res[alg_res][:-1]

plt.figure()
for alg_res in res:
    plt.semilogy(res[alg_res], label=alg_res)
plt.legend()
plt.xlabel("Iteration")
plt.ylabel("Residual Vector Norm")

# Save the plot as .pdf and include it in the .tex document
plt.savefig("myplot1.pdf", bbox_inches="tight")
print(r"\saveandshowplot{myplot1.pdf}")

plt.figure()
for alg_con in convergence_factor:
    plt.semilogy(convergence_factor[alg_con], label=alg_con)
plt.legend()
plt.xlabel("Iteration")
plt.ylabel("Convergence Factor")

# Save the plot as .pdf and include it in the .tex document
plt.savefig("myplot2.pdf", bbox_inches="tight")
print(r"\saveandshowplot{myplot2.pdf}")
```



2 Convergence properties

(a)

Lemma 1.

$$0 < \alpha < \frac{2}{\lambda_{max}} \Rightarrow \rho(I - \alpha A) < 1$$

Proof. A is symmetric positive definite matrix, thus:

$$0 < \lambda_{min} \leq \cdots \leq \lambda_{max}$$

therefore, we get that:

$$\rho(I - \alpha A) = max(|1 - \alpha\lambda_{min}|, |1 - \alpha\lambda_{max}|)$$

$|1 - \alpha\lambda_{max}|$, then we get that:

$$-1 < 1 - \alpha\lambda_{max} < 1 \Rightarrow |1 - \alpha\lambda_{max}| < 1$$

And,

$$-1 < 1 - 2\frac{\lambda_{min}}{\lambda_{max}} < 1 - \alpha\lambda_{min} < 1 \Rightarrow |1 - \alpha\lambda_{min}| < 1$$

thus,

$$max(|1 - \alpha\lambda_{min}|, |1 - \alpha\lambda_{max}|) < 1$$

so we get that:

$$\rho(I - \alpha A) < 1$$

□

In our case:

$$\alpha = \frac{1}{||A||}$$

we know that for any induced norm:

$$||A|| > \rho(A) = \lambda_{max}$$

thus,

$$\frac{1}{||A||} < \frac{1}{\lambda_{max}} < \frac{2}{\lambda_{max}}$$

therefore, by **Lemma 1** we get that

$$\rho(I - \alpha A) \leq 1$$

and the method converges.

(b) In the case A is indefinite, we a negative eigenvalue, therefore:

$$\rho(I - \alpha A) \geq |1 - \alpha\lambda_{min}|$$

$\lambda_{min} < 0$, by definition, therefore

$$\rho(I - \alpha A) \geq |1 - \alpha\lambda_{min}| > 1$$

(c) (i)

$$f(\mathbf{x}) = \frac{1}{2}||\mathbf{x}^* - \mathbf{x}||_A^2$$

$$\begin{aligned}
f(\mathbf{x}^{(k)}) &= \frac{1}{2}||\mathbf{x}^* - \mathbf{x^{(k)}}||_A^2 \\
&= \frac{1}{2}||\mathbf{e}^{(k)}||_A^2 \\
&= \frac{1}{2}((\mathbf{e}^{(k)})^T A\mathbf{e}^{(k)}) \\
&= \frac{1}{2}\langle \mathbf{e}^{(k)}, A\mathbf{e}^{(k)}\rangle
\end{aligned}$$

$$\begin{aligned}
f(\mathbf{x}^{(k+1)}) &= f(\mathbf{x}^{(k)}) + \alpha\mathbf{r}^{(k)} \\
&= \frac{1}{2}\langle \mathbf{e}^{(k)}, A\mathbf{e}^{(k)}\rangle - \alpha\langle \mathbf{r}^{(k)}, A\mathbf{e}^{(k)}\rangle + \frac{1}{2}\alpha^2\langle \mathbf{r}^{(k)}, A\mathbf{r}^{(k)}\rangle \\
&= \frac{1}{2}\langle \mathbf{e}^{(k)}, A\mathbf{e}^{(k)}\rangle - \frac{\langle \mathbf{r}^{(k)}, A\mathbf{e}^{(k)}\rangle^2}{\langle \mathbf{r}^{(k)}, A\mathbf{r}^{(k)}\rangle} + \frac{1}{2}\frac{\langle \mathbf{r}^{(k)}, A\mathbf{e}^{(k)}\rangle^2\langle \mathbf{r}^{(k)}, A\mathbf{r}^{(k)}\rangle}{\langle \mathbf{r}^{(k)}, A\mathbf{r}^{(k)}\rangle^2} \qquad * \, \alpha = \frac{\langle \mathbf{r}^{(k)}, A\mathbf{e}^{(k)}\rangle}{\langle \mathbf{r}^{(k)}, A\mathbf{r}^{(k)}\rangle} \\
&= \frac{1}{2}\langle \mathbf{e}^{(k)}, A\mathbf{e}^{(k)}\rangle - \frac{\langle \mathbf{r}^{(k)}, A\mathbf{e}^{(k)}\rangle^2}{\langle \mathbf{r}^{(k)}, A\mathbf{r}^{(k)}\rangle} + \frac{1}{2}\frac{\langle \mathbf{r}^{(k)}, A\mathbf{e}^{(k)}\rangle^2}{\langle \mathbf{r}^{(k)}, A\mathbf{r}^{(k)}\rangle} \\
&= \frac{1}{2}\langle \mathbf{e}^{(k)}, A\mathbf{e}^{(k)}\rangle - \frac{1}{2}\frac{\langle \mathbf{r}^{(k)}, A\mathbf{e}^{(k)}\rangle^2}{\langle \mathbf{r}^{(k)}, A\mathbf{r}^{(k)}\rangle} \\
&= f(\mathbf{x}^{(k)}) - \frac{1}{2}\frac{\langle \mathbf{r}^{(k)}, A\mathbf{e}^{(k)}\rangle^2}{\langle \mathbf{r}^{(k)}, A\mathbf{r}^{(k)}\rangle}
\end{aligned}$$

We get that:

$$f(\mathbf{x}^{(k+1)}) = f(\mathbf{x}^{(k)}) - \frac{1}{2}\frac{\langle \mathbf{r}^{(k)}, A\mathbf{e}^{(k)}\rangle^2}{\langle \mathbf{r}^{(k)}, A\mathbf{r}^{(k)}\rangle}$$

A is symmetric positive definite matrix, thus

$$\frac{\langle \mathbf{r}^{(k)}, A\mathbf{e}^{(k)}\rangle^2}{\langle \mathbf{r}^{(k)}, A\mathbf{r}^{(k)}\rangle} > 0$$

and therefore,

$$f(\mathbf{x}^{(k+1)}) = f(\mathbf{x}^{(k)}) - \frac{1}{2}\frac{\langle \mathbf{r}^{(k)}, A\mathbf{e}^{(k)}\rangle^2}{\langle \mathbf{r}^{(k)}, A\mathbf{r}^{(k)}\rangle} < f(\mathbf{x}^{(k)})$$

(ii) From previous section:

$$f(\mathbf{x}^{(k+1)}) = C^{(k)}f(\mathbf{x}^{(k)}) = f(\mathbf{x}^{(k)}) - \frac{1}{2}\frac{\langle \mathbf{r}^{(k)}, A\mathbf{e}^{(k)}\rangle^2}{\langle \mathbf{r}^{(k)}, A\mathbf{r}^{(k)}\rangle}$$

thus,

$$C^{(k)} = 1 - \frac{1}{2}\frac{\langle \mathbf{r}^{(k)}, A\mathbf{e}^{(k)}\rangle^2}{\langle \mathbf{r}^{(k)}, A\mathbf{r}^{(k)}\rangle f(\mathbf{x}^{(k)})}$$

finally,

$$C^{(k)} = 1 - \frac{\langle \mathbf{r}^{(k)}, A\mathbf{e}^{(k)}\rangle^2}{\langle \mathbf{r}^{(k)}, A\mathbf{r}^{(k)}\rangle\langle \mathbf{e}^{(k)}, A\mathbf{e}^{(k)}\rangle}$$

(iii) t

(iv) t

3 GMRES(1) method

(a)

$$||\mathbf{r}^{(k+1)}||_2 = ||\mathbf{b} - A\mathbf{x}^{(k+1)}||_2$$

we define the following scalar function $g(\alpha)$:

$$\begin{aligned} g(\alpha) &\triangleq f(\mathbf{x}^{(k)}) + \alpha\mathbf{r}^{(k)} \\ &= \frac{1}{2}||\mathbf{b} - A\mathbf{x}^{(k)} - \alpha A\mathbf{r}^{(k)}||_2 \\ &= \frac{1}{2}||\mathbf{r}^{(k)} - \alpha A\mathbf{r}^{(k)}||_2 \\ &= \frac{1}{2}(\mathbf{r}^{(k)})^T\mathbf{r}^{(k)} - \alpha(\mathbf{r}^{(k)})^T A\mathbf{r}^{(k)} + \frac{1}{2}\alpha^2(A\mathbf{r}^{(k)})^T A\mathbf{r}^{(k)} \end{aligned}$$

And the minimization of g with respect to α is done by:

$$g'(\alpha) = -(\mathbf{r}^{(k)})^T A\mathbf{r}^{(k)} + \alpha(A\mathbf{r}^{(k)})^T A\mathbf{r}^{(k)} = 0$$

$$\Rightarrow \alpha_{opt} = \frac{(\mathbf{r}^{(k)})^T A\mathbf{r}^{(k)}}{(A\mathbf{r}^{(k)})^T A\mathbf{r}^{(k)}} = \frac{(\mathbf{r}^{(k)})^T A\mathbf{r}^{(k)}}{(\mathbf{r}^{(k)})^T A^T A\mathbf{r}^{(k)}}$$

(b) (non-mandatory)

(c) t

(d) t

(e) t

4 Convexity

(a) i. e^{ax} is convex:

$$(e^{ax})'' = a^2 e^{ax} \geq 0 \quad \forall x$$

ii. $-log(x)$ is convex:

$$(-log(x))'' = \frac{1}{x^2} > 0 \quad \forall x > 0$$

iii. $log(x)$ is concave:

$$(log(x))'' = -\frac{1}{x^2} < 0 \quad \forall x > 0$$

iv. $|x|^a, \quad a \geq 1$ is convex:

$$\begin{aligned} f(\alpha x + (1-\alpha)y) &= |\alpha x + (1-\alpha)y|^a \\ &\leq \end{aligned}$$

v. t

(b) t

(c) t

5 Non Linear Optimization

(a) t

(b) t

(c) t