APPM 5600 - Homework 6

Eappen Nelluvelil

October, 8 2021

1. In class, we showed that the update to the basis for the residual space is given by

$$\mathbf{p}_{k+1} = \mathbf{r}_{k+1} - \frac{\langle \mathbf{p}_k, \mathbf{r}_{k+1} \rangle_{\mathbf{A}}}{||\mathbf{p}_k||^2_{\mathbf{A}}} \mathbf{p}_k. \tag{1}$$

In this exercise, we will remove the need to apply $\mathbf{A}$ to create this update (less computing $\mathbf{r}_{k+1}$).

(a) Using the fact that $\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k \mathbf{A}\mathbf{p}_k$ and $\mathbf{r}^T_{k+1}\mathbf{r}_k = 0$, show that $\langle \mathbf{p}_k, \mathbf{r}_{k+1} \rangle_{\mathbf{A}} = -\frac{||\mathbf{r}_{k+1}||^2_2}{\alpha_k}$.
Rearranging the formula for $\mathbf{r}_{k+1}$ to solve for $\mathbf{A}\mathbf{p}_k$, we have that

$$\mathbf{A}\mathbf{p}_k = \frac{\mathbf{r}_k - \mathbf{r}_{k+1}}{\alpha_k}.$$

Then, we have that

$$\begin{aligned}
\langle \mathbf{p}_k, \mathbf{r}_{k+1} \rangle_{\mathbf{A}} &= \mathbf{p}_k \mathbf{A} \mathbf{r}_{k+1} \\
&= \left(\frac{\mathbf{r}_k - \mathbf{r}_{k+1}}{\alpha_k} \right)^T \mathbf{r}_{k+1} \\
&= \frac{\mathbf{r}^T_k \mathbf{r}_{k+1} - \mathbf{r}^T_{k+1}\mathbf{r}_{k+1}}{\alpha_k} \\
&= -\frac{\mathbf{r}^T_{k+1}\mathbf{r}_{k+1}}{\alpha_k} \\
&= -\frac{||\mathbf{r}_{k+1}||^2_2}{\alpha_k},
\end{aligned}$$

as desired. Here, we used the fact that $\mathbf{r}_{k+1}$ must be orthogonal to the previous residuals.

(b) Rewrite $||\mathbf{p}_k||^2_{\mathbf{A}}$ in terms of $\mathbf{r}_k$.

We know that $\mathbf{p}_k = \mathbf{r}_k - \frac{\langle \mathbf{p}_{k-1}, \mathbf{r}_k \rangle_{\mathbf{A}}}{||\mathbf{p}_{k-1}||_{\mathbf{A}}^2} \mathbf{p}_{k-1}$. Then, we have that

$$\begin{aligned}
||\mathbf{p}_k||_{\mathbf{A}}^2 &= \langle \mathbf{p}_k, \mathbf{p}_k \rangle_{\mathbf{A}} \\
&= \mathbf{p}_k^T \mathbf{A} \mathbf{p}_k \\
&= \mathbf{p}_k^T \left(\frac{\mathbf{r}_k - \mathbf{r}_{k+1}}{\alpha_k} \right) \\
&= \frac{1}{\alpha_k} \left(\mathbf{p}_k^T \mathbf{r}_k - \mathbf{p}_k^T \mathbf{r}_{k+1} \right) \\
&= \frac{1}{\alpha_k} \left(\left(\mathbf{r}_k - \frac{\langle \mathbf{p}_{k-1}, \mathbf{r}_k \rangle_{\mathbf{A}}}{||\mathbf{p}_{k-1}||_{\mathbf{A}}^2} \mathbf{p}_{k-1} \right)^T \mathbf{r}_k \right) \\
&= \frac{1}{\alpha_k} \left(\mathbf{r}_k^T \mathbf{r}_k - \frac{\langle \mathbf{p}_{k-1}, \mathbf{r}_k \rangle_{\mathbf{A}}}{||\mathbf{p}_{k-1}||_{\mathbf{A}}^2} \mathbf{p}_{k-1}^T \mathbf{r}_k \right) \\
&= \frac{\mathbf{r}_k^T \mathbf{r}_k}{\alpha_k} \\
&= \frac{||\mathbf{r}_k||_2^2}{\alpha_k},
\end{aligned}$$

as desired. Here, we used the fact that $\mathbf{r}_{k+1}$ must be orthogonal to the previous conjugate directions.

(c) Substitute these expressions into (1) to get a technique for evaluating the next basis vector for the residual space without any applications of the matrix $\mathbf{A}$.

Substituting the results from parts (a) and (b) into (1), we have that the next $\mathbf{A}$-conjugate direction, $\mathbf{p}_{k+1}$, is given by

$$\mathbf{p}_{k+1} = \mathbf{r}_{k+1} + \frac{||\mathbf{r}_{k+1}||_2^2}{||\mathbf{r}_k||_2^2} \mathbf{p}_k.$$

2. Consider a sparse 500×500 matrix $\mathbf{A}$ constructed as follows.

(a) Put a 1 in each diagonal entry.

(b) In each off-diagonal entry, put a random number from the uniform distribution $[-1, 1]$, but make sure to maintain symmetry. Then, replace each off-diagonal entry with $|a_{ij}| > \tau$ by 0, where τ is a parameter ($\tau = 0.01, 0.05, 0.1$, and 0.2). (You should have four matrices $\mathbf{A}_j$, one for each τ.)

Take the right hand side to be a random vector $\mathbf{b}$, and set the tolerance to 10^{-10}.

(a) Write a steepest descent and conjugate gradient solver.

See attached code.

(b) Apply steepest descent to solve each of the linear systems, and plot the residual for each iteration ($||\mathbf{r}_n||$ versus the iteration n) on a `semilogy` scale. Put all of these plots on the same graph.

See attached code.

(c) Apply conjugate gradient to solve each of the linear systems, and plot the residual for each iteration ($||\mathbf{r}_n||$ versus the iteration n) on a `semilogy` scale. Put all of these plots on the same graph.

See attached code.

(d) What do you observe about the convergence of these methods? If the methods do not converge for any choices of τ, explain what is happening.

When we set $\tau = 0.01$, 0.05, or 0.1, both iterative methods converge to a solution, but the conjugate gradient method takes fewer iterations than the steepest descent method in each case. However, when

$\tau = 0.2$, the steepest descent method fails to converge, and the corresponding residuals grow larger. The conjugate gradient method converges to a solution very slowly, and the `semilogy` plot of the iterations vs the 2-norm of the residuals shows that the 2-norms of the residuals do not decay consistently, i.e., we do not get consistent improvement in the residual in each iteration.

This is because when $\tau = 0.2$, the corresponding matrix is symmetric but not positive-definite, which means that the steepest descent and conjugate gradient iterations are not guaranteed to converge. We see this in the code, as the steepest descent iteration fails to converge, and the conjugate gradient iteration converges very slowly.

(e) How does the residual compare with the error bounds provided in class? (This is strictly comparing the numbers that you get.)

The error bound on the steepest descent iteration is given by

$$||\mathbf{e}_{k+1}||_{\mathbf{A}} \leq \left(\frac{\lambda_{\max} - \lambda_{\min}}{\lambda_{\max} + \lambda_{\min}}\right) ||\mathbf{e}_k||_{\mathbf{A}},$$

where $k \geq 0$, and the error bound on the conjugate gradient iteration is given by

$$||\mathbf{e}_{k+1}||_{\mathbf{A}} \leq 2\left(\frac{1 - \sqrt{\kappa\left(\mathbf{A}^{-1}\right)}}{1 + \sqrt{\kappa\left(\mathbf{A}^{-1}\right)}}\right)^{k+1} ||\mathbf{x}^*||_{\mathbf{A}}.$$

For $\tau = 0.01, 0.05$, and 0.1 the $\mathbf{A}$-norm of the errors of the iterates from the steepest descent and conjugate gradient iterations satisfy these error bounds. In the code, the error bounds are displayed for the next iterate of steepest descent and the latest iterate of conjugate gradient.

However, when $\tau = 0.2$, these error bounds are no longer true because $\mathbf{A}$ is no longer positive-definite. This can also be seen in the code output.

3. Suppose conjugate gradient is applied to a symmetric positive definite matrix $\mathbf{A}$ with the result $||\mathbf{e}_0||_{\mathbf{A}} = 1$ and $||\mathbf{e}_{10}||_{\mathbf{A}} = 2 \times 2^{-10}$, where $||\mathbf{e}_k||_{\mathbf{A}} = ||\mathbf{x}_k - \mathbf{x}^*||_{\mathbf{A}}$ and $\mathbf{x}^*$ is the true solution.

(a) What bound can you given on $\kappa(\mathbf{A})$?

Taking $\mathbf{x}_0 = \mathbf{0}$ in the conjugate gradient method, we have that $\mathbf{e}_0 = \mathbf{x}^*$, which implies that $||\mathbf{x}^*||_{\mathbf{A}} = 1$. We also know that

$$||\mathbf{e}_n||_{\mathbf{A}} \leq 2\left(\frac{1 - \sqrt{\kappa\left(\mathbf{A}^{-1}\right)}}{1 + \sqrt{\kappa\left(\mathbf{A}^{-1}\right)}}\right)^{n} ||\mathbf{x}^*||_{\mathbf{A}},$$

which implies that

$$||\mathbf{e}_{10}||_{\mathbf{A}} = 2 \cdot 2^{-10} \leq 2\left(\frac{1 - \sqrt{\kappa\left(\mathbf{A}^{-1}\right)}}{1 + \sqrt{\kappa\left(\mathbf{A}^{-1}\right)}}\right)^{10}.$$

Rearranging the above inequality, we have that

$$\left(\frac{1+\sqrt{\kappa\left(\mathbf{A}^{-1}\right)}}{2}\right)^{10} \leq \left(1-\sqrt{\kappa\left(\mathbf{A}^{-1}\right)}\right)^{10} \implies \frac{1+\sqrt{\kappa\left(\mathbf{A}^{-1}\right)}}{2} \leq 1-\sqrt{\kappa\left(\mathbf{A}^{-1}\right)}$$
$$\implies 1+\sqrt{\kappa\left(\mathbf{A}^{-1}\right)} \leq 2-2\sqrt{\kappa\left(\mathbf{A}^{-1}\right)}$$
$$\implies 3\sqrt{\kappa\left(\mathbf{A}^{-1}\right)} \leq 1$$
$$\implies \frac{1}{\kappa\left(\mathbf{A}\right)} \leq \frac{1}{9}$$
$$\implies \kappa\left(\mathbf{A}\right) \geq 9.$$

(b) What bound can you give on $||\mathbf{e}_{20}||_{\mathbf{A}}$?
From theorem 38.2 in Bau and Trefethen, we have the following error bound on the $\mathbf{A}$-norm of the errors resulting from the conjugate gradient iteration applied to the SPD system $\mathbf{Ax} = \mathbf{b}$,

$$||\mathbf{e}_n||_{\mathbf{A}} \leq ||\mathbf{e}_{n-1}||_{\mathbf{A}},$$

where $n \geq 1$. It follows that $||\mathbf{e}_{20}||_{\mathbf{A}} \leq ||\mathbf{e}_{10}||_{\mathbf{A}} = 2^{-9}$.

4. Consider the task of solving the following system of non-linear equations,

$$f_1(x,y) = 3x^2 + 4y^2 - 1 = 0,$$
$$f_2(x,y) = y^3 - 8x^3 - 1 = 0,$$

for the solution $\boldsymbol{\alpha}$ near $(x,y) = (-0.5, 0.25)$.

(a) Apply the fixed point iteration with

$$\mathbf{g}(\mathbf{x}) = \mathbf{x} - \begin{bmatrix} 0.016 & -0.17 \\ 0.52 & -0.26 \end{bmatrix} \begin{bmatrix} 3x^2 + 4y^2 - 1 \\ y^3 - 8x^3 - 1 \end{bmatrix}.$$

You can use $(-0.5, 0.25)$ as the initial guess. How many steps are needed to get an approximation to 7 digits of accuracy?
We perform the fixed point iteration and compute the relative error in the 2-norm. After 5 iterations, we get a solution, $\begin{bmatrix} -0.497251208023980 \\ 0.254078591468912 \end{bmatrix}$, that is within 7 digits of accuracy.

(b) Why is this a good choice for $\mathbf{g}(\mathbf{x})$?
The Jacobian of $\mathbf{f}$ is given by

$$\begin{bmatrix} 6x & 8y \\ -24x^2 & 3y^2 \end{bmatrix},$$

which, when evaluated at the initial guess, gives us $\begin{bmatrix} -3 & 2 \\ -6 & \frac{3}{16} \end{bmatrix}$. The inverse of the Jacobian evaluated at the initial guess is approximately $\begin{bmatrix} 0.016393442622951 & -0.174863387978142 \\ 0.524590163934426 & -0.262295081967213 \end{bmatrix}$. The matrix in the fixed point iteration is close to this inverse Jacobian. Thus, the fixed point iteration is almost a Newton iteration applied to $\mathbf{f}$, but the inverse Jacobian is not updated in each iteration, i.e., this is a lazy Newton iteration. This iteration converges to a fixed point of $\mathbf{g}$ near the initial guess because the infinity norm of the approximate inverse Jacobian at the initial guess is less than 1.

Homework 6

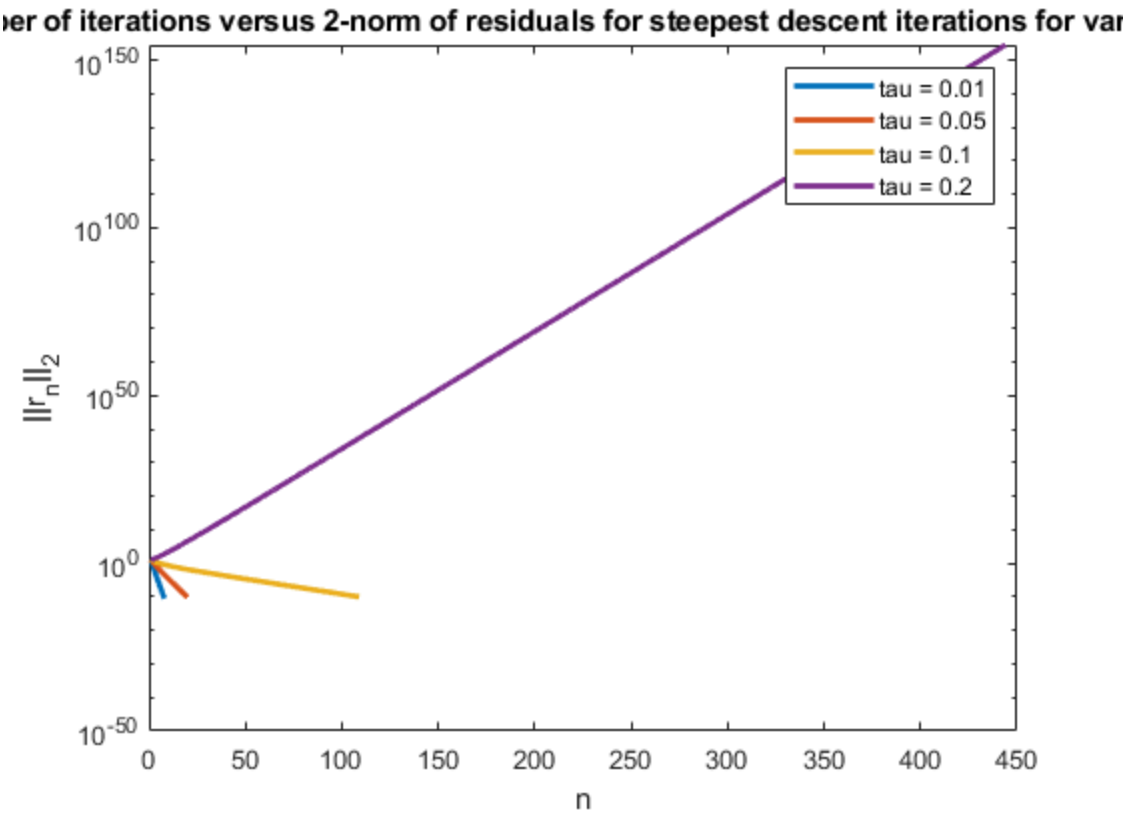
Table of Contents

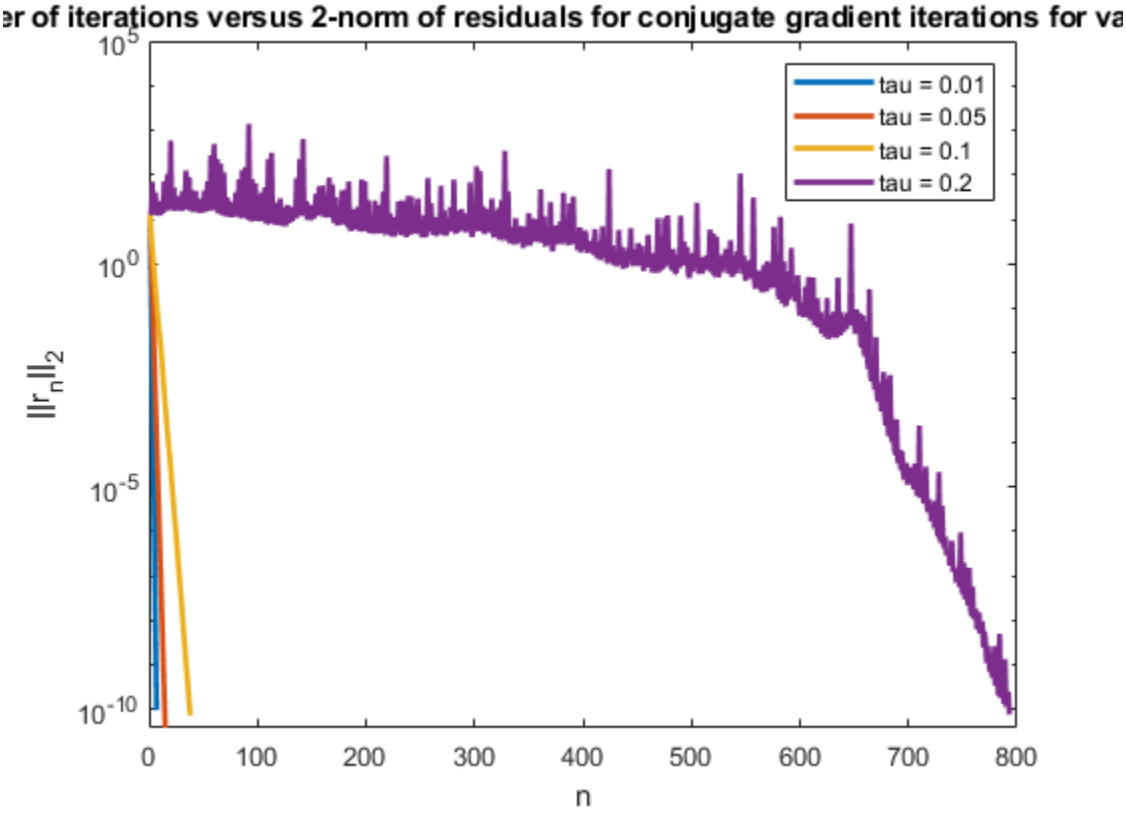
Problem 2

```
clear;
clc;
close all;

fprintf("Problem 2\n");
fprintf("---------\n\n");

rng(0);

max_iters = 1000;
tol = 1e-10;
n = 500;
taus = [0.01, 0.05, 0.1, 0.2];
x_0 = zeros(n, 1);
b = rand(n, 1);

% Create a matrix with entries from the uniform distribution on [-1, 1]
% and extract the strictly upper triangular portion of it
U_strict = triu(-1 + (1-(-1))*rand(n), 1);

diag_A = diag(ones(n, 1));

f1 = figure;
f2 = figure;

for i=1:length(taus)
    U_tau = U_strict;
    U_tau(abs(U_tau) > taus(i)) = 0;
    A_tau = diag_A + U_tau + U_tau';

    % Compute the true solution using backslash
    x_star = A_tau\b;

    fprintf("Smallest eigenvalue of A_tau: %e\n\n", min(eig(A_tau)));

    % [sd_residual_norms, sd_iters] = sd_solver(A_tau, b, x_0, max_iters,
 tol);
    [sd_residual_norms, sd_iters, x_old_sd, x_new_sd] = sd_solver_2(A_tau, b,
 x_0, max_iters, tol);
```

```
    fprintf("tau = %0.2f, norm of last residual: %e, iterations of SD: %d
\n", ...
        taus(i), sd_residual_norms(end), sd_iters);

    % Compute error bound for steepest descent iteration
    e_old_sd = norm((x_old_sd - x_star)'*A_tau*(x_old_sd - x_star));
    e_new_sd = norm((x_new_sd - x_star)'*A_tau*(x_new_sd - x_star));
    A_tau_eigs = eig(A_tau);
    sd_error_const = (max(A_tau_eigs)-min(A_tau_eigs))/
(max(A_tau_eigs)+min(A_tau_eigs));
    fprintf("Error bound for steepest descent at next iterate: %e <= %e\n
\n", ...
            e_new_sd, sd_error_const*e_old_sd);

    figure(f1);
    semilogy(1:sd_iters, sd_residual_norms, "LineWidth", 2, ...
             "DisplayName", strcat("tau = ", string(taus(i))));
    hold on;

    [cg_residual_norms, cg_iters, x_old_cg] = cg_solver(A_tau, b, x_0,
 max_iters, tol);

    fprintf("tau = %0.2f, norm of last residual: %e, iterations of CG: %d
\n", ...
        taus(i), cg_residual_norms(end), cg_iters);

    % Compute error bound for conjugate gradient iteration
    cond_A_tau = cond(A_tau);
    cg_error_const = 2*((1-sqrt(1/cond_A_tau))/(1+sqrt(1/
cond_A_tau)))^(cg_iters);
    e_new_cg = norm((x_old_cg - x_star)'*A_tau*(x_old_cg - x_star));
    x_star_A_norm = norm(x_star'*A_tau*x_star);
    fprintf("Error bound for conjugate gradient at latest iterate: %e <=
 %0.16e\n\n", ...
            e_new_cg, cg_error_const*x_star_A_norm);

    figure(f2);
    semilogy(1:cg_iters, cg_residual_norms, "LineWidth", 2, ...
             "DisplayName", strcat("tau = ", string(taus(i))));
    hold on;
end

figure(f1);
title("Number of iterations versus 2-norm of residuals for steepest descent
 iterations for various tau");
xlabel("n");
ylabel("||r_{n}||_{2}");
legend;

figure(f2);
title("Number of iterations versus 2-norm of residuals for conjugate gradient
 iterations for various tau");
xlabel("n");
```

```
ylabel("||r_{n}||_{2}");
legend;

Problem 2
---------

Smallest eigenvalue of A_tau: 9.711416e-01

tau = 0.01, norm of last residual: 2.692745e-11, iterations of SD: 8
Error bound for steepest descent at next iterate: 3.391908e-28 <= 1.351165e-26

tau = 0.01, norm of last residual: 9.814560e-11, iterations of CG: 7
Error bound for conjugate gradient at latest iterate: 9.647152e-21 <=
 3.8516978213207186e-11

Smallest eigenvalue of A_tau: 7.051883e-01

tau = 0.05, norm of last residual: 6.959479e-11, iterations of SD: 20
Error bound for steepest descent at next iterate: 3.421366e-23 <= 1.252923e-22

tau = 0.05, norm of last residual: 3.915573e-11, iterations of CG: 15
Error bound for conjugate gradient at latest iterate: 1.606588e-21 <=
 1.6465427187500402e-10

Smallest eigenvalue of A_tau: 1.870409e-01

tau = 0.10, norm of last residual: 9.475522e-11, iterations of SD: 109
Error bound for steepest descent at next iterate: 1.161239e-20 <= 1.429167e-20

tau = 0.10, norm of last residual: 7.411670e-11, iterations of CG: 38
Error bound for conjugate gradient at latest iterate: 9.032560e-21 <=
 4.0565272895214168e-09

Smallest eigenvalue of A_tau: -1.270429e+00

tau = 0.20, norm of last residual: NaN, iterations of SD: 445
Error bound for steepest descent at next iterate: NaN <= NaN

tau = 0.20, norm of last residual: 7.821014e-11, iterations of CG: 793
Error bound for conjugate gradient at latest iterate: 1.076976e-20 <=
 5.30745230586949O1e-33
```


er of iterations versus 2-norm of residuals for steepest descent iterations for var
tau = 0.01
tau = 0.05
tau = 0.1
tau = 0.2
10^{150}
10^{100}
10^{50}
10^{0}
10^{-50}
$\|r_n\|_2$
0
50
100
150
200
250
300
350
400
450
n


er of iterations versus 2-norm of residuals for conjugate gradient iterations for va
tau = 0.01
tau = 0.05
tau = 0.1
tau = 0.2
10^{5}
10^{0}
10^{-5}
10^{-10}
$\|r_n\|_2$
0
100
200
300
400
500
600
700
800
n

Problem 4

```
clear;
close all;

fprintf("\n\nProblem 4\n");
fprintf("---------\n\n")

tol = 1e-7;

g = @(x, y) [x; y] - [0.016 -0.17; 0.52 -0.26]*[3*x.^(2) + 4*y.^(2) - 1;
 y.^(3) - 8*x.^(3) - 1];

x_old = [-0.5, 0.25];
x_new = g(x_old(1), x_old(2));
fixed_pt_iters = 1;

while norm(x_new-x_old, 2)/norm(x_old, 2) > tol
    x_old = x_new;
    x_new = g(x_old(1), x_old(2));
    fixed_pt_iters = fixed_pt_iters + 1;
end

fprintf("Converged fixed point iterate: (%f, %f)\n", x_new(1), x_new(2));
fprintf("Number of fixed point iterations to get relative error below %e: %d
\n", ...
        tol, fixed_pt_iters);
```

```
Problem 4
---------

Converged fixed point iterate: (-0.497251, 0.254079)
Number of fixed point iterations to get relative error below 1.000000e-07: 5
```

Published with MATLAB® R2021b

```
function [cg_residual_norms, iters, x_old] = cg_solver(A, b, x_0, max_iters,
 tol)
iters = 1;
x_old = x_0;
r_old = b-A*x_old;
cg_residual_norms = [norm(r_old, 2)];
p_old = r_old;

while (norm(r_old, 2) > tol) && (iters < max_iters)
    alpha_new = (r_old'*p_old)/(p_old'*A*p_old);
    x_new = x_old + alpha_new*p_old;
    r_new = r_old - alpha_new*A*p_old;
    cg_residual_norms = [cg_residual_norms; norm(r_new, 2)];
    beta_new = (r_new'*r_new)/(r_old'*r_old);
    p_new = r_new + beta_new*p_old;
    iters = iters + 1;
    r_old = r_new;
    p_old = p_new;
    x_old = x_new;
end
end
```

```
Not enough input arguments.

Error in cg_solver (line 3)
x_old = x_0;
```

Published with MATLAB® R2021b

```
function [sd_residual_norms, iters, x_old, x_new] = sd_solver_2(A, b, x_0,
 max_iters, tol)
    iters = 0;
    x_old = x_0;
    sd_residual_norms = [];

    r_k = Inf;
    while (norm(r_k, 2) > tol) && (iters < max_iters)
        r_k = b-A*x_old;
        sd_residual_norms = [sd_residual_norms; norm(r_k, 2)];
        alpha_k = (r_k'*r_k)/(r_k'*A*r_k);
        x_new = x_old + alpha_k*r_k;
        x_old = x_new;
        iters = iters + 1;
    end

    % Compute the next iterate (for problem 2e)
    r_k = b-A*x_old;
    alpha_k = (r_k'*r_k)/(r_k'*A*r_k);
    x_new = x_old + alpha_k*r_k;
end
```

```
Not enough input arguments.

Error in sd_solver_2 (line 3)
    x_old = x_0;
```

Published with MATLAB® R2021b