Michael Cowan

Computational Physics

Problem Set 0

Chapter 1

1. This problem is a straightforward use of matrix operations. The result of each operation is as expected.

Code:

1. A=[1 2;3 4]; %Assign values to matrix A
2. disp('(a) A\*A=');
3. disp(A\*A); %Multiply A by itself
4. disp('(b) A.\*A=');
5. disp(A.\*A); %Multiply each element in A by itself
6. disp('(c) A^2=');
7. disp(A^2); %A^2 gives the same result as A\*A
8. disp('(d) A.^2=');
9. disp(A.^2); %A.^2 gives the same result as A.\*A
10. disp('(e) A/A=');
11. disp(A/A); %Dividing a matrix by itself gives an identity matrix
12. disp('(f) A./A=');
13. disp(A./A); %Dividing each matrix element by itself gives a matrix of 1's

Output:

(a) A\*A=

7 10

15 22

(b) A.\*A=

1 4

9 16

(c) A^2=

7 10

15 22

(d) A.^2=

1 4

9 16

(e) A/A=

1 0

0 1

(f) A./A=

1 1

1 1

2. Problem 2 has us practicing using various plotting functions in Matlab. This include plotting points instead of lines, and using logarithmic axes.

Part a

Code:

1. x=1:10;
2. y=x.^2;
3. plot(x,y)

Output:

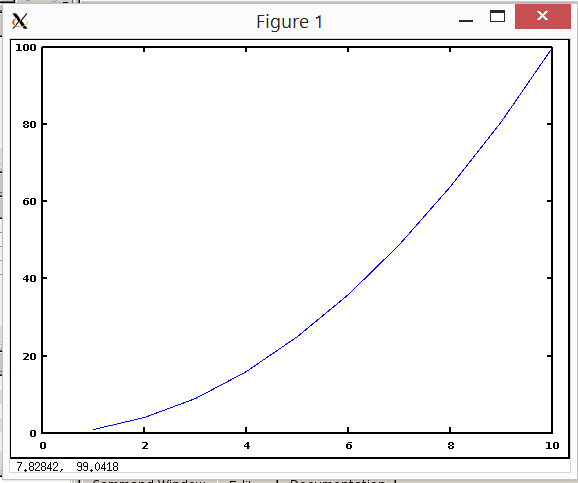


Figure 1: plot for 2a; the graph is a continuous line

Part b

Code:

1. x=1:10;
2. y=x.^2;
3. plot(x,y,'+')

Output:

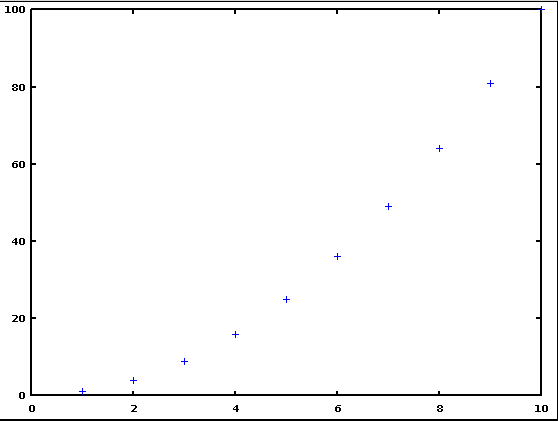


Figure 2: plot for 2b; the graph is the discrete points

Part c

Code:

1. x=1:10;
2. y=x.^2;
3. plot(x,y,'-',x,y,'+')

Output:

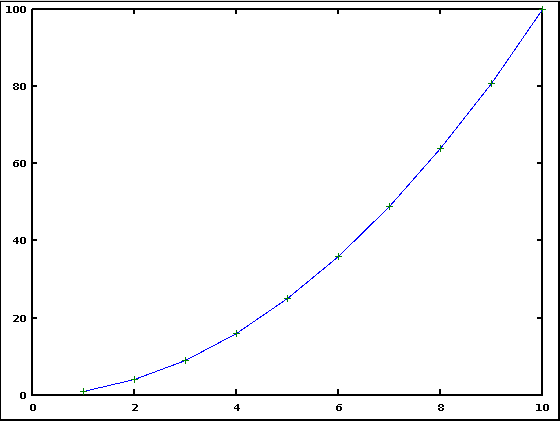


Figure 3: plot for 2c; the graph has both the discrete points and the continuous line

Part 2d:

Code:

1. x=1:10;
2. y=x.^2;
3. plot(x,y,'-',x(1:2:10),y(1:2:10),'+')

Output:

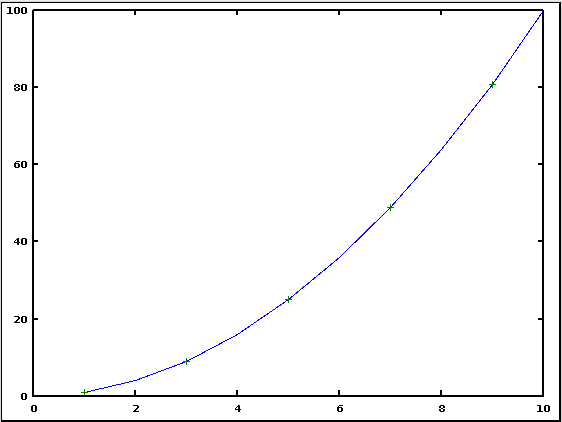


Figure 4: plot for 2d; similar to 2c, but with only every other discrete point marked

Part 2e

Code:

1. x=1:10;
2. y=x.^2;
3. semilogy(x,y)

Output:

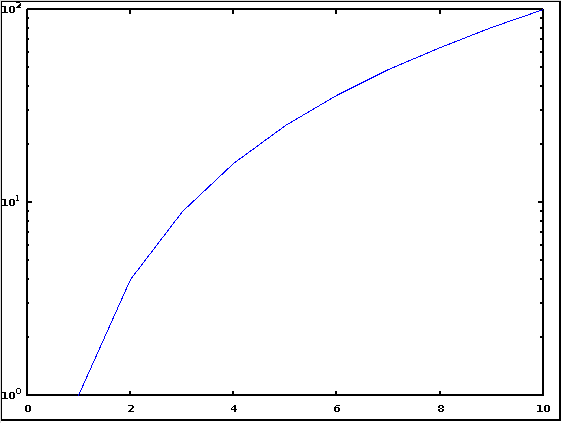


Figure 5: plot for 2e; changes the y-axis to a logarithmic scale

Part 2f

Code:

1. x=1:10;
2. y=x.^2;
3. loglog(x,y,'+')

Output:

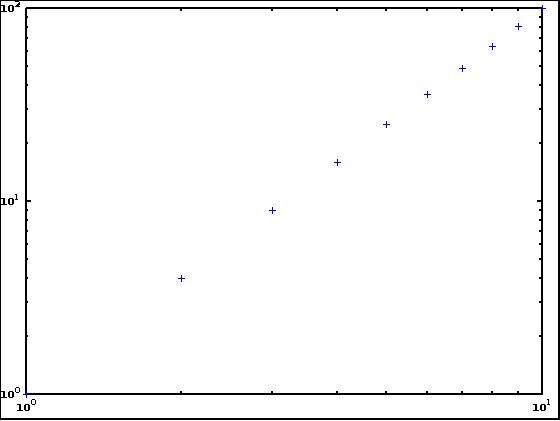


Figure 6: plot for 2f; both axes are now logarithmic and only the discrete points are shown

3. In this problem we attempt to reproduce four plots given in the book.

Part a

Code:

1. x=0:pi/32:6\*pi;
2. y=exp(-x/4).\*sin(x);
3. plot(x,y);
4. title('f(x)=exp(-x/4)\*sin(x)');
5. xlabel('x');
6. ylabel('f(x)')

Output:

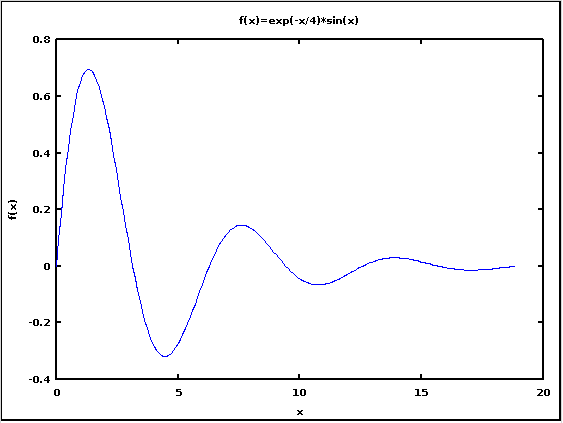


Figure7: plot for 3a

Part b

Code:

1. x=0:pi/32:6\*pi;
2. y=exp(-x/4).\*sin(x);
3. z=exp(-x/4);
4. plot(x,y,'-',x,z,'--',x(1:16:193),y(1:16:193),'o');
5. title('f(x)=exp(-x/4)\*sin(x)');
6. xlabel('x');
7. ylabel('f(x)')

Output:

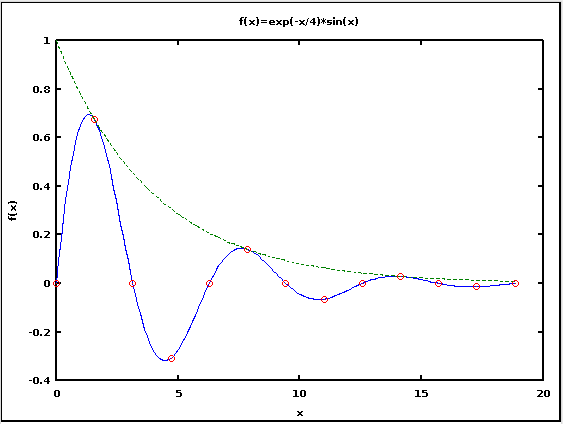


Figure 8: plot for 3b; both the graphs of f(x) and exp(-x/4) are shown

I was not able to figure out how to add the label for exp(-x/4).

Part c

Code:

1. x=0:.2:6\*pi;
2. y=exp(-x).\*(sin(x)).^2;
3. semilogy(x,y,'+');
4. axis([0,6\*pi,10^-10,10^0]); %sets the ranges of the axes
5. title('f(x)=exp(-x)\*sin(x)');
6. xlabel('x');
7. ylabel('f(x)');

Output:

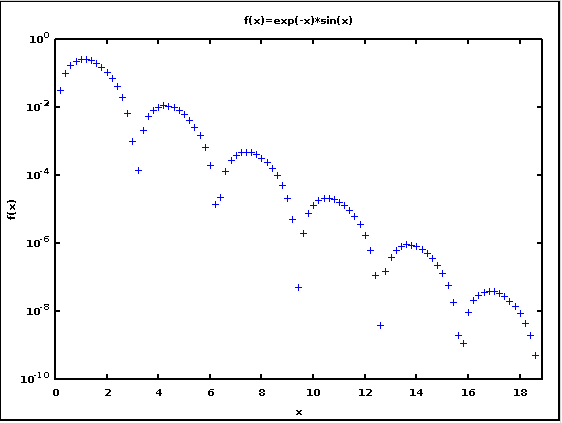


Figure 9: plot for 3c; y-axis is logarithmic

Part d

Code:

1. x=0:pi/64:6\*pi;
2. y=(sin(3\*x)).^2;
3. polar(x,y)

Output:

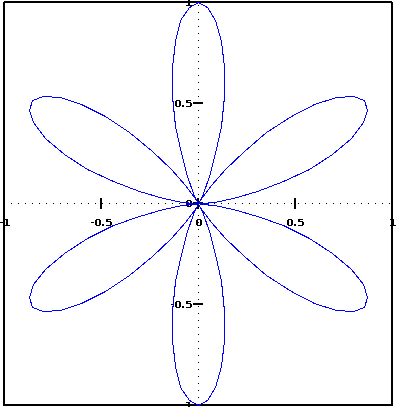


Figure 10: plot for 3d; used the polar plot function

4. This problem has us use some matrix operations other than the ones used in problem 1. In part a, we find the inverse of the matrix, and check the result by multiplying the original matrix by its inverse to produce an identity matrix. In part b we find the eigenvalues of each of the matrices.

Part a

Code:

1. A=[1 2 3;0 4 5;0 0 6]; %assign value to matrix A
2. disp('A=');
3. disp (A); %print A
4. disp('inv(A)=');
5. disp(inv(A)); %print inverse of A
6. disp('A\*inv(A)=');
7. disp(A\*inv(A)); %check that A multiplied by its inverse produces the identity matrix
8. B=[1 0 0;0 2 0;0 0 3]; %assign value to matrix B
9. disp('B=');
10. disp (B); %print B
11. disp('inv(B)=');
12. disp(inv(B)); %print inverse of B
13. disp('B\*inv(B)=');
14. disp(B\*inv(B)); %check that B multiplied by its inverse produces the identity matrix
15. C=[1 2 3;4 5 6;7 8 9]; %assign value to matrix C
16. disp('C=');
17. disp (C); %print C
18. disp('inv(C)=');
19. disp(inv(C)); %print inverse of C
20. disp('C\*inv(C)=');
21. disp(C\*inv(C)); %check that C multiplied by its inverse produces the identity matrix
22. D=[1 1/2 1/3;1/4 1/5 1/6;1/7 1/8 1/9]; %assign value to matrix D
23. disp('D=');
24. disp (D); %print D
25. disp('inv(D)=');
26. disp(inv(D)); %print inverse of D
27. disp('D\*inv(D)=');
28. disp(D\*inv(D)); %check that D multiplied by its inverse produces the identity matrix

Output:

A=

1 2 3

0 4 5

0 0 6

inv(A)=

1.00000 -0.50000 -0.08333

0.00000 0.25000 -0.20833

0.00000 0.00000 0.16667

A\*inv(A)=

1.00000 0.00000 0.00000

0.00000 1.00000 0.00000

0.00000 0.00000 1.00000

B=

1 0 0

0 2 0

0 0 3

inv(B)=

1.00000 -0.00000 -0.00000

0.00000 0.50000 -0.00000

0.00000 0.00000 0.33333

B\*inv(B)=

1 0 0

0 1 0

0 0 1

C=

1 2 3

4 5 6

7 8 9

inv(C)=

-4.5036e+015 9.0072e+015 -4.5036e+015

9.0072e+015 -1.8014e+016 9.0072e+015

-4.5036e+015 9.0072e+015 -4.5036e+015

C\*inv(C)=

2 0 2

8 0 0

16 0 8

D=

1.00000 0.50000 0.33333

0.25000 0.20000 0.16667

0.14286 0.12500 0.11111

inv(D)=

4.6667 -46.6667 56.0000

-13.3333 213.3333 -280.0000

9.0000 -180.0000 252.0000

D\*inv(D)=

1.00000 0.00000 0.00000

-0.00000 1.00000 0.00000

-0.00000 0.00000 1.00000

The result for the inverse of C is notable, as it seems to give matrix elements on the order of 1015. Multiplying C by the given inverse gives a matrix that is not the identity matrix. Both of these things happen because C is actually a singular matrix (det(C)=0), so it does not have an inverse. The other three matrices produce the expected results.

Part b

Code:

1. disp('eig(A)=');
2. disp(eig(A)); %print the eigenvalues of A
3. disp('eig(B)=');
4. disp(eig(B)); %print the eigenvalues of B
5. disp('eig(C)=');
6. disp(eig(C)); %print the eigenvalues of C
7. disp('eig(D)=');
8. disp(eig(D)); %print the eigenvalues of D

Output:

eig(A)=

warning: inverse: matrix singular to machine precision, rcond = 1.54198e-018

warning: inverse: matrix singular to machine precision, rcond = 1.54198e-018

1

4

6

eig(B)=

1

2

3

eig(C)=

1.6117e+001

-1.1168e+000

-1.3037e-015

eig(D)=

1.1941847

0.1147546

0.0021718

The error warning seems to be for matrix C again, but a matrix with determinant 0 does have eigenvalues. The third eigenvalue for C, -1.3037e-015, should be interpreted as 0.

5. The problem is to use an expansion to estimate the value of the exponential of a matrix. We perform an estimate of the value using the first six terms of the expansion, then compare it to the exact value given by the Matlab function expm().

Code:

1. A=[1 2 3;0 4 5;0 0 6];
2. B=[1 0 0;0 2 0;0 0 3];
3. C=[1 2 3;4 5 6;7 8 9];
4. D=[1 1/2 1/3;1/4 1/5 1/6;1/7 1/8 1/9];
5. w=0; %assign each output function an initial value of 0
6. x=0;
7. y=0;
8. z=0;
9. for n=0:5 %iterates the loop 6 times, with n incrementing by 1 each iteration
10. w=w+(A^n)/factorial(n); %each iteration is added to the total from the previous iteration
11. x=x+(B^n)/factorial(n);
12. y=y+(C^n)/factorial(n);
13. z=z+(D^n)/factorial(n);
14. end
15. disp('e^A=');
16. disp(w); %approximate value of e^A
17. disp('expm(A)=');
18. disp(expm(A)); %exact value of e^A
19. disp('e^B=');
20. disp(x); %approximate value of e^B
21. disp('expm(B)=');
22. disp(expm(B)); %exact value of e^B
23. disp('e^C=');
24. disp(y); %approximate value of e^C
25. disp('expm(C)=');
26. disp(expm(C)); %exact value of e^C
27. disp('e^D=');
28. disp(z); %approximate value of e^D
29. disp('expm(D)=');
30. disp(expm(D)); %exact value of e^D

Output:

e^A=

2.71667 26.76667 216.41667

0.00000 42.86667 342.33333

0.00000 0.00000 179.80000

expm(A)=

2.71828 34.58658 554.67037

0.00000 54.59815 872.07661

0.00000 0.00000 403.42879

e^B=

2.71667 0.00000 0.00000

0.00000 7.26667 0.00000

0.00000 0.00000 18.40000

expm(B)=

2.71828 0.00000 0.00000

0.00000 7.38906 0.00000

0.00000 0.00000 20.08554

e^C=

1425.2 1750.4 2076.6

3226.3 3965.2 4702.2

5028.4 6178.1 7328.8

expm(C)=

1.1189e+06 1.3748e+06 1.6307e+06

2.5339e+06 3.1134e+06 3.6929e+06

3.9489e+06 4.8520e+06 5.7552e+06

e^D=

2.90544 0.99697 0.68539

0.50087 1.34219 0.26936

0.29017 0.20916 1.17218

expm(D)=

2.90940 0.99916 0.68695

0.50197 1.34280 0.26979

0.29082 0.20952 1.17244

The estimate seems to be better the smaller the values of the individual elements of the matrix. Matrix D gives the best estimate, with all of the elements being ≤1. Matrix C, on the other hand, with elements ranging from 1 to 9, has the estimate being off by three orders of magnitude from the exact value of the expm function.