MATH4610: HW04

Chandler Justice - A02313187

Question 1: Create files for the individual routeines we have discussed in class.

I created the following routines in the `src` directory of my `mathlib` directory:

```
dfapp_backward.c
dfapp_central.c
dfapp_forward.c
dmaceps.c
fx.c
inf_norm.c
l1_dist.c
l1_norm.c
l2_dist.c
l2_norm.c
linf_dist.c
linreg.c
reduce_matrix.c
smaceps.c
test-lib.c
```

which can be found on my github repositiory at this link.

Question 2: Create object modules for the routines

I created the following object modules in the `out` directory of my `mathlib` directory.

```
dfapp_backward.o
dmaceps.o
l1_dist.o
l2_norm.o
dfapp_central.o
fx.o
l1_norm.o
linf_dist.o
reduce_matrix.o
dfapp_forward.o
inf_norm.o
l2_dist.o
linreg.o
smaceps.o
```

Which can be found on my github repository at this link

Question 3: Use the `ar` command to create your library of routines

My `mathlib.a` library can be found on my github at this link

Question 4: Use the `ranlib` command to create an index of the library. Verify the results with `ar -t`.

Here is the output from my terminal when I run `ar -t mathlib.a`:

```
chandler@merci out % ar -t mathlib.a
__.SYMDEF
dfapp_backward.o
dfapp_central.o
dmaceps.o
fx.o
inf_norm.o
l1_dist.o
l1_norm.o
l2_dist.o
l2_norm.o
linf_dist.o
linreg.o
reduce_matrix.o
smaceps.o
```

Question 5: compile and test your codes with the test routine you created

Here is the output of my test routine. The code for this routeine can be found at this link

```
chandler@merci src % ./test
32 bit macepts value: 5.96046e-08
64 bit macepts value: 1.11022e-16
l2-norm: 5.96822
l1-norm: 11.54
inf-norm: 4
l2-dist: 7.42951
l1-dist: 12.56
linf-dist: 6.5
forward approx of derivative of x * x at x = 10, h = 0.01: 20.01
central approx of derivative of x * x at x = 10, h = 1: 20
backwards approx of derivative of x * x at x = 10, h = 0.01: 19.99
linear regression of (2,4), (3,5), (5,7), (7,10), (9,15): y = 1.518293x +
0.304878
```

Question 6: Create a software documentation page for each of the routeines you have created

My software documentation page can be found at **this link**.

Question 7: Create routines that reduce a square matrix into an upper triangular matrix and use back substitution to solve the resulting matrix

For my implementation, I placed the matrix reduction and back substitution into one routine. The link to the source code for this implementation is available at **this link**.

Question 8: Include a test problem in your test code and make sure matrix reduction results are accurate for a linear system contained in a matrix with at least `10` rows and `10` columns.

For my test code, I reduced the following matrix

```
3 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0
0 0 3 0 0 0 0 0 0 0
0 0 0 5 0 0 0 0 0 0
0 0 0 0 5 0 0 0 0 0
0 0 0 0 0 2 0 0 0 0
0 0 0 0 0 0 6 0 0 0
0 0 0 0 0 0 0 2 0 0
0 0 0 0 0 0 0 0 9 0
0 0 0 0 0 0 0 0 0 5
```

which reduces to the following solutions

```
x1=1.000000

x2=1.000000

x3=1.000000

x4=1.000000

x5=1.000000

x6=1.000000

x7=1.000000

x8=1.000000

x9=1.000000

x10=1.000000
```

Question 9: Validate the results of the test in question 8 using the tools developed in the shared library.

We can use the `l1-dist` function to determine the difference between the results obtained from `reduce-matrix` and the actual results. Creating a vector from the solutions obtained by my algorithm, and comparing them to the actual answer, I get a difference of `0`, meaning my algorithm produced accurate results.

Question 10: Create a table of contents for your software manual.

My table of contents is located at **this link**.