2S03 Assignment 2

Ned Nedialkov

3 October 2018

Due: 17 October in class

Problem 1 (10 points) Consider solving the linear system

$$Ax = b,$$

where A is an $n \times n$ matrix and b is an n vector.

- (7 points) Write a function

  ```
  int linsolve(int n, double A[], double b[], double x[])
  ```

  where n is the size of the problem, `A` is an array of size $\geq n^2$, and b and x are arrays of size $\geq n$. The matrix A is stored row-wise as a one-dimensional array.

  If this function computes a solution x, it should return 1. If it fails, for example when A is singular, it should return 0. You can implement simple Gauss elimination and without pivoting.

  Store this function in file `linsolve.c`.

- (2 points) Write a function

  ```
  void genmatrix(int n, double A[])
  ```

  that fills an $n \times n$ matrix with random numbers and function

  ```
  void genvector(int n, double b[])
  ```

  that fills an n vector with random numbers.

  Store these functions in file `gendata.c`.

- Create a file `checkresult.c` with content

```
void checkresult(int n, int flag, double A[], double b[],
   double x[])
{

}
```

This function does not do anything, but when we check the correctness of your implementation, we will call it with our implementation of `checkresult`.

- Then use the following main program

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define N 8000
void genmatrix(int n, double A[]);
void genvector(int n, double b[]);
void checkresult(int n, int flag, double A[], double b[],
   double x[]);
int linsolve(int n, double A[], double b[], double x[]);
double A[N * N], B[N*N], b[N], c[N], x[N];
int main(int argc, char **argv) {
  if (argc != 2) {
    printf("Usage:␣main␣n\n");
    return 1;
  }
  int n = atoi(argv[1]);
  if (n > N) {
    printf("n␣=␣%d␣must␣be␣<=␣%d\n", n, N);
    return 2;
  }
  genmatrix(n, A);
  genvector(n, b);
  // Save A and b so we use it to check the result.
  memmove(B,A,n*n*sizeof(double));
  memmove(c,b,n*sizeof(double));
  // You can overwrite A and b.
  int success = linsolve(n, A, b, x);
  checkresult(n, success, B, c, x);
  return 0;
}
```

- (2 points) Create a makefile such that when make is typed, an executable with name linsolve is created.

Submit hard code of your programs and the files linsolve.c, gendata.c, main.c, and makefile to SVN under directory A2/P1

Problem 2 (5 points) You are given the function

```
#include<sys/resource.h>
#include<unistd.h>
double gettime() {
  struct rusage usage;
  getrusage(RUSAGE_SELF, &usage);
  struct timeval time;
  time = usage.ru_utime;
  return time.tv_sec+time.tv_usec/1e6;
}
```

Time the execution of your linsolve using the following main program

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#define N 8000
void genmatrix(int n, double A[]);
void genvector(int n, double b[]);
void checkresult(int n, int flag, double A[], double b[],
   double x[]);
int linsolve(int n, double A[], double b[], double x[]);
double gettime();
double A[N * N], b[N], x[N];
int main() {
  int n;
  genmatrix(N, A);
  genvector(N, b);
  for (n = 1000; n <= N; n *= 2) {
    double t       = gettime();
    int    success = linsolve(n, A, b, x);
    printf("␣%␣8d␣␣␣%.1e␣\n", n, gettime() - t);
  }
  return 0;
}
```

Produce executables without optimization, and then with flags -O1, -O2, -O3. Complete the following table

n	time (sec)			
	no optim	`-O1`	`-O2`	`-O3`
1000				
2000				
4000				
8000				

Submit all your C files and a makefile to directory `A2/P2` and a hardcopy of this table. When `make` is typed, an executable `linsolve` should be created in this directory. (Do not submit a text file of this table to SVN)

Problem 3 (5 points) The sinc function is defined by

$$\operatorname{sinc}(x) = \begin{cases} \frac{\sin(\pi x)}{\pi x}, & x \neq 0, \\ 1, & x = 0. \end{cases} \tag{1}$$

You are given a positive integer N, interval (a, b), and $x \in (a, b)$. Let

$$h = \frac{\pi}{\sqrt{2N}} \tag{2}$$

$$\rho(x, a, b) = \frac{x - a}{b - x} \tag{3}$$

$$\lambda(x, h, a, b, j) = \operatorname{sinc}\left(\frac{\log \rho(x, a, b) - jh}{h}\right), \qquad j = -N, ..., N. \tag{4}$$

Write the C functions

```
double sinc(double x);
double comph(int N);
double comprho(double x, double a, double b);
double complambda(double x, double h, double a, double b, int j);
```

that implement (1–4), respectively. Use the π constant from `math.h`, defined as `M_PI`.

Write the function

```
void wj(double x, double a, double b, int N, double *w)
```

that returns in the array `w` the values

$$w_{-N} = \left(1 + e^{-Nh}\right)\left(\frac{1}{1 + \rho(x, a, b)} - \sum_{j=-N+1}^{N} \frac{\lambda(x, h, a, b, j)}{1 + e^{jh}}\right)$$

$$w_j = \lambda(x, h, a, b, j), \qquad j = -N + 1, ..., N - 1,$$

$$w_N = \left(1 + e^{-Nh}\right)\left(\frac{\rho(x, a, b)}{1 + \rho(x, a, b)} - \sum_{j=-N}^{N-1} \frac{e^{jh}\lambda(x, h, a, b, j)}{1 + e^{jh}}\right)$$

w_{-N} must be stored in `w`[0], w_{-N+1} must be stored in `w`[1], and so on.

Store the above functions in file `sinc.c`.

Use the following main program

```
#include <stdio.h>
#include <stdlib.h>
extern void wj(double x, double a, double b, int N, double *w);
int main(int argc, char **argv) {
  if (argc != 5) {
    printf("\n␣Usage␣sinc␣N␣␣a␣b␣x␣\n");
    return 1;
  }
  int    N = atoi(argv[1]);
  double a = atof(argv[2]);
  double b = atof(argv[3]);
  double x = atof(argv[4]);
  double w[2 * N + 1];
  wj(x, a, b, N, w);
  for (int i = 0; i < 2*N+1; i++)
    printf("%␣3i␣␣%␣e\n", i, w[i]);
  return 0;
}
```

Submit to SVN under `A2/P3` the files `sinc.c` and `main.c`, and a makefile such that when `make` is typed, an executable with name `sinc` is created. Submit a hardcopy of `sinc.c`.