02635 Fall 2016 — Module 3

Homework

- Read chapter 5 and chapter 7 pp. 263-287 in "Beginning C"
- Read chapters 3 and 4 in "Writing Scientific Software"

Exercises — Part I

1. Do exercises 5-1, 5-2, and 5-4 in "Beginning C"
2. Take this quiz to test your understanding of *arrays*
3. Take this quiz to test your understanding of *pointers* (questions 5 and 6 are about dynamic memory allocation which is next week's topic, so feel free to skip these questions)
4. Analyze the following program. What does it compute? Rewrite the program to make the code more readable.

```c
#include <stdio.h>

int main(void) {

    int arr[10] = {19,74,13,67,44,80,7,36,9,77}, *p = arr;
    int i = -1;
    double val = 0.0;

    while (i++ < 10)
        val += *(p++);
    val /= 10;

    printf("Value: %.2f\n",val);

    return 0;
}
```

5. Write a program that creates two arrays `arr1` and `arr2` of length N and prompts the user to enter N double precision floating point numbers, storing these in the array `arr1`. Your program should then copy then contents of `arr1` to `arr2` using the `memcpy` routine, which is defined in `string.h`

and has the following prototype:

```
void *memcpy(void *dest, const void *src, size_t n)
```

Note that the last input n should be the number of bytes to copy (and not the number of elements in your array). Finally, write out the N numbers in `arr2` to check that these match the input.

Remark: You can define N using a preprocessor macro, say, `#define N 10`.

Exercises — Part II

Line integration in two dimensions

In last week's exercise, we looked at numerical methods for evaluating a definite integral of the form

$$\int_a^b f(x)\,dx.$$

Now suppose that we want to compute a so-called line integral of a function $f(x, y)$ of two variables. Given two points (x_1, y_1) and (x_2, y_2), we can express the line that passes through these two points parametrically as

$$\begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = (1-t)\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + t\begin{bmatrix} x_2 \\ y_2 \end{bmatrix}$$

where $t \in \mathbb{R}$ is a parameter. If we define

$$D(t) = \sqrt{\left(\frac{dx(t)}{dt}\right)^2 + \left(\frac{dy(t)}{dt}\right)^2},$$

the integral of f over the line segment from $(x(t_1), y(t_1))$ to $(x(t_2), y(t_2))$ can be expressed as

$$\int_{t_1}^{t_2} g(t)D(t)\,dt$$

where $g(t) = f(x(t), y(t))$.

Approximation on rectangular grid

Suppose we do not know the function f, but we know the function value at a finite number of grid points in a rectangular grid with lower left corner (x_l, y_l) and upper right corner (x_u, y_u). With $M \geq 2$ points in the vertical direction and $N \geq 2$ points in the horizontal direction, it is natural to store the function values in a matrix H of

size $M \times N$, i.e.,

$$H_{ij} = f\left(x_l + \frac{j-1}{N-1}(x_u - x_l), y_l + \frac{i-1}{M-1}(y_u - y_l)\right)$$

for $i = 1, \ldots, M$, and $j = 1, \ldots, N$.

The function value $f(x, y)$ at an arbitrary point (x, y) inside the grid (i.e., $x_l \le x \le x_u$ and $y_l \le y \le y_u$) can be approximated using *bilinear interpolation*. To understand how bilinear interpolation works, let us consider a simple grid with four grid points ($M = N = 2$)

$$(x_l, y_l),\ (x_l, y_u),\ (x_u, y_u),\ (x_u, y_l).$$

Using linear interpolation in the x direction, we obtain

$$f(x, y_l) \approx \frac{x_u - x}{x_u - x_l} f(x_l, y_l) + \frac{x - x_l}{x_u - x_l} f(x_u, y_l)$$

and

$$f(x, y_u) \approx \frac{x_u - x}{x_u - x_l} f(x_l, y_u) + \frac{x - x_l}{x_u - x_l} f(x_u, y_u),$$

and using linear interpolating in the y direction, we arrive at

$$\begin{aligned} f(x, y) &\approx \frac{y_u - y}{y_u - y_l} f(x, y_l) + \frac{y - y_l}{y_2 - y_l} f(x, y_u) \\ &= \frac{1}{(x_u - x_l)(y_u - y_l)} (f(x_l, y_l)c_{ll} + f(x_u, y_l)c_{ul} + f(x_l, y_u)c_{lu} + f(x_u, y_u)c_{uu}) \end{aligned}$$

where

$$c_{ll} = (x_u - x)(y_u - y),\ c_{ul} = (x - x_l)(y_u - y),\ c_{lu} = (x_u - x)(y - y_l),\ c_{uu} = (x - x_l)(y - y_l).$$

Thus, the approximation is a linear combination of the function values at the four grid points.

Using bilinear interpolation to approximate function values on a rectangular gird with a large number of grid points amounts to (i) finding four grid points that form a cell containing the point (x, y), and (ii) applying bilinear interpolation using the function values at the four grid points.

Exercises

1. Show that $D(t)$ is the distance between (x_1, y_1) and (x_2, y_2), and hence $D = D(t)$ is a constant and

$$\int_{t_1}^{t_2} g(t)D(t)\,dt = D\int_{t_1}^{t_2} g(t)\,dt.$$

2. Write a program to approximate the line integral

$$D \int_0^1 g(t)\, dt$$

using the trapezoidal rule with n subintervals. You may reuse some of the code that you wrote last week for numerical integration.

   - Use the function $f(x, y) = \cos(1 - xy)$ with the grid coordinates $x_l = y_l = 0$ and $x_u = y_u = 1$ to test your code.
   - Your program should prompt the user to enter n and the points (x_1, y_1) and (x_2, y_2) that define the line segment used for integration.
   - Define some test cases (i.e., points (x_1, y_1) and (x_2, y_2)) for which you can evaluate the integral analytically. Use these test cases to check that your code is correct.

3. Write down the expression for the elements of the matrix H using C-style indexing (i.e., starting at index 0 instead of index 1).
4. Given a rectangular grid with MN grid points and a point (x, y) where $x_l \leq x \leq x_u$ and $y_l \leq y \leq y_u$, find the four grid points needed for bilinear interpolation. Which elements of the matrix H contain the function values?
5. Extend you program to approximate the line integral using bilinear interpolation to approximate $g(t)$.

   - Your program should prompt the user to enter the grid dimensions M and N.
   - Store the function values at the grid points in a two-dimensional array (i.e., precompute the matrix H in the beginning of your program).

6. Review the priorities listed in the beginning of chapter 3 of "Writing Scientific Software". How would you characterize your code with respect to these priorities? What improvements, if any, can be made?