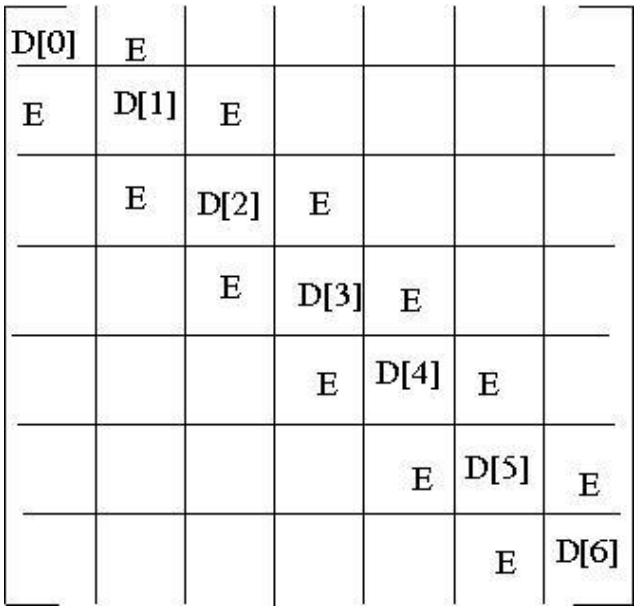
P4

Matrices and Vectors

In this lab we are going to learn how to do matrix algebra in C++. We start with simple vectors then introduce a vector of vectors to serve as a matrix structure. We then need to write functions that can multiply a matrix by a vector a matrix by a matrix and solve a tridiagonal set of linear equations.



Schedule

Preparation time : 3 hours

Lab time : 3 hours

Items provided

Tools : None

Components : None

Equipment : None

Software : MinGW and Eclipse

Items to bring

Essentials. A full list is available on the Laboratory website at https://secure.ecs.soton.ac.uk/notes/ellabs/databook/essentials/

Before you come to the lab, it is essential that you read through this document and complete ***all*** of the preparation work in section 2. If possible, prepare for the lab with your usual lab partner. Only preparation which is recorded in your laboratory logbook will contribute towards your mark for this exercise. There is no objection to several students working together on preparation, as long as all understand the results of that work. Before starting your preparation, read through all sections of these notes so that you are fully aware of what you will have to do in the lab.

> **Academic Integrity** – *If you undertake the preparation jointly with other students, it is important that you acknowledge this fact in your logbook. Similarly, you may want to use sources from the internet or books to help answer some of the questions. Again, record any sources in your logbook.*

Revision History

February 16, 2013	Jeff Reeve (jsr)	First version of this lab created
January 14, 2014	Jeff Reeve (jsr)	Typos corrected

1 Aims, Learning Outcomes and Outline

This laboratory exercise aims to:

- Set you up to be able to do matrix manipulations in C++
- Become familiar with random number support in C++
- Develop a Tridiagonal matrix solver in C++

Having successfully completed the lab, you will be able to:

- Learn to use vectors in place of dynamic arrays
- Learn to construct non trivial structures

You will develop structures that mean you can "safely" manipulate matrices and vectors. You are given a Tridiagonal solver that you need to convert to C++. To test your solver you will generate a Tridiagonal matrix (T), generate a vector of random numbers (L) and multiply L by T to get another vector (R). You can then test your Tridiagonal solver by solving the set of equations TL = R for the L values which you can check against your original test vector.

---

2 Preparation

Read through the course handbook statement on safety and safe working practices, and your copy of the standard operating procedure. Make sure that you understand how to work safely. Read through this document so you are aware of what you will be expected to do in the lab.

2.1 Preparation Section 1

Write a short program to get random numbers into a vector. You are required to use the STL <random> facilities. You need to include <iostream> <algorithm> and <vector>.

Eg

Std::default_random_engine dre;

std::uniform_real_distribution<double> dr(10,20);

dr(dre) results in a double between 10 and 20 from a uniform distribution.

2.2 Preparation Section 2

Write a short program that implements a matrix as a vector<vector<double>> and fills it with numbers.

2.3 Preparation Section 3

The following is C code that solves a system of linear equations AL = R, for the vector L

The matrix T has the elements D[i] along the diagonal and either side of the diagonal the elements are the same – the number E. The front cover gives you the idea.

```
void TridiagonalSolve(double E, double *D, double *R,  double *L, int n){
int i,;
double *c = malloc(n, sizeof(double));
Complex id;
/* Set the off diagonal elements */
     for (i = 0; i < n; i++)
          c[i] = E;
/*forward bit */
     c[0] /= D[0]; /*if /0 rearrange equations */
     R[0] /= D[0];
     for (i = 1; i < n; i++){
          id = D[i] - c[i - 1] * E;
          c[i] /= id; /* Last value calculated is redundant. */
          R[i] = (R[i] - R[i - 1] * E) / id;
     }
/* Now back substitute. */
     L[n - 1] = R[n - 1];
     for (i = n - 2; i >= 0; i--)
          L[i] = R[i] - c[i] * L[i + 1];
     }
}
```

Re-write this in C++ so that it returns a vector for L and uses only vectors and not dynamic arrays.

---

3 Laboratory Work

3.1 Subsection 1

Implement your random number code and your matrix code from the preparation.

Implement and test your tridiagonal solver.

Almost any values will do for the D[i] and E but don't have any of the D[i] = 0.

To test your solver you need to generate a Tridiagonal matrix (T), generate a vector of random numbers (L) and multiply L by T to get another vector (R). You can then test your Tridiagonal solver by solving the set of equations TL = R for the L values which you can check against your original test vector.

---

4 Optional Additional Work

Marks will only be awarded for this section if you have already completed all of Section 3 to an excellent standard and with excellent understanding.

Re-do the matrix solver to work when D, and R (and of course L) are complex types from your lab last week.

Appendices

References

HTTLAP

C++ web reference