Maths 1262 quizzes

Wednesday Quiz 5 26/3/14 answers **Your answers should show all work.**

(1: 12 marks) Suppose a program includes a test of whether two double-precision vectors, of dimension n (which can be large), are orthogonal. How would you implement the test, and can you be confident about it?

Answer. Calculate the dot product $\sum x_i y_i$, with an error of at most $\gamma_n \sum |x_i y_i|$. Consider them to be orthogonal if the dot product is below a certain tolerance, 10^{-9}, for example.

Since $\sum |x_i||y_i|$ can be large even when the dot product is zero, one cannot be confident of the test.

(2: 13 marks) Calculate the LU factorisation

$$\begin{bmatrix} 1 & 0 & 0 \\ \cdot & 1 & 0 \\ \cdot & \cdot & 1 \end{bmatrix} \begin{bmatrix} \cdot & \cdot & \cdot \\ 0 & \cdot & \cdot \\ 0 & 0 & \cdot \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \end{bmatrix}$$

Recall that you calculate the first row of U, then the first column of L, and so on.

$$\begin{bmatrix} 1 & 0 & 0 \\ \cdot & 1 & 0 \\ \cdot & \cdot & 1 \end{bmatrix} \begin{bmatrix} \cdot & \cdot & \cdot \\ 0 & \cdot & \cdot \\ 0 & 0 & \cdot \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ \cdot & 1 & 0 \\ \cdot & \cdot & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \\ 0 & \cdot & \cdot \\ 0 & 0 & \cdot \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & \cdot & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \\ 0 & \cdot & \cdot \\ 0 & 0 & \cdot \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & \cdot & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \\ 0 & -1 & -5 \\ 0 & 0 & \cdot \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 5 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \\ 0 & -1 & -5 \\ 0 & 0 & \cdot \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 5 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \\ 0 & -1 & -5 \\ 0 & 0 & 18 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \end{bmatrix}$$

(3: 12 marks) Using $|L||U|$ as an estimate of $|\tilde{L}||\tilde{U}|$, estimate the floating-point error when computing the above factorisation in double precision. Recall that $\epsilon_{mach} = 2^{-52}$.

$$\frac{3 \times 2^{-52}}{1 - 3 \times 2^{-52}} \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 5 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & 5 \\ 0 & 0 & 18 \end{bmatrix} = \frac{3 \times 2^{-52}}{1 - 3 \times 2^{-52}} \begin{bmatrix} 1 & 2 & 3 \\ 2 & 5 & 11 \\ 3 & 11 & 52 \end{bmatrix}$$

(4: 13 marks) (i) Simulate the following program.

The `to_string()` function converts, e.g., the integer `12` to the string `"12"`. It doesn't work with my old C++ compiler, but does on aturing so long as you compile using `g++ -std=c++11 ...`.
The `append` function appends another string to a given string.

You simulate the recursive routine carefully, showing the values of the variables, and show what gets printed.

(ii) What does `recu(n)` do in general, given $n \geq 0$?

```
#include <iostream>
#include <string>
using namespace std;

string recu ( int n )
{
  int q,r;

  if ( n < 10 )
    return to_string( n );
  else
  {
    q = n/10; r = n%10;
    return to_string(r).append(recu (q));
  }
}

int main()
{
  int n = 1250;
  cout << "n " << n << " recu(n) " << recu(n) << endl;

  return 0;
}
```

```
main calls recu(1250)
  recu:
    n 1250 q 125 r 0
    calls recu(125)
    recu:
      n 125 q 12 r 5
      calls recu (12)
      recu:
        n 12 q 1 r 2
        calls recu (1)
        recu:
        n 1 returns "1"
      returns "2" "1"
    returns "5""21"
  returns "521""0"
prints:
n 1250 recu(n) 0521
```

(ii) It returns the decimal form of n with its digits reversed.