Author: Yiling Liu

Student ID: 22214014

Three Cases of the Solutions to systems of linear equations

For matrix A and vector b (len(A) == len(b))

There will be 3 cases if we want to solve the problem $Ax = b$

let n = len(b)

case1. One Solution

$r(A) = r(A|b) = n$

case2. No Solution

$r(A) < r(A|b)$

case3. Infinite Number of Solution

$r(A) = r(A) < n$

This program will handle all three cases: export result with file name answer.csv in case 1, Print out error "Infinite number of answers or no answer" and exit the code in case 2 and 3.

Gaussian Elimination

General Idea

There are two methods to do eliminate elements in the first step of Gaussian Elimination(transform the raw matrix into an upper triangular matrix):

To explain how elimination works, define an operator $\oplus$;

For $a_{22} - \frac{a_{21}}{a_{11}} * a_{12}$, it could be expressed as $a_{22} \oplus (a_{21}, a_{11}, a_{12})$

Naive Method

Do the elimination in order as the following example shows,no matter what a_{ij} is for elimination is, if it is not zero.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a_{22} \oplus (a_{21}, a_{11}, a_{12}) & a_{23} \oplus (a_{21}, a_{11}, a_{13}) \\ 0 & a_{32} \oplus (a_{31}, a_{11}, a_{12}) & a_{33} \oplus (a_{31}, a_{11}, a_{13}) \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a_{22} \oplus (a_{21}, a_{11}, a_{12}) & a_{23} \oplus (a_{21}, a_{11}, a_{13}) \\ 0 & 0 & a_{33} \oplus (a_{32}, a_{22} \oplus (a_{12}, a_{11}, a_{12}), a_{23} \oplus (a_{21}, a_{11}, a_{13})) \end{bmatrix}$$

Pivot Element Method

Find the number with maximum absolute value in that row, and use it to eliminate. I will use this method in this project because it is closer to the real-world numerical calculation schema.

Pivot method will be used in this project because if the absolute value of a_{ij} in naive method is extremely small, it will cause the explosive increasement of other element, and the increasement of rounding error.

It could also handle the bonus problem in this assignment. If the maximum absolute value in column k is 0, that means: $a_1 * x_1 + a_2 * x_2 + ... + 0 * x_k + ... + a_n * x_n = c$

After elimination it will be

$0 * x_k = c$

If c == 0, the number of solution is infinite because for any x, $0 * x \equiv 0$.

If c != 0, $0 * x \not\equiv 0$, the determinant has no answer.

Parallel in this Project

There are several places for us to apply parallel method in this project

- Generate matrix with random number

  There is no relationship between cells

- Select row contains largest absolute value in a specific column

  It is a cumulative operation, but cannot be done by `reduction`. But we can still distribute works to threads by using tricks.

- Use a row to eliminate other rows in both elimination and calculation

  If you use row a to eliminate row b and row c, there is no relationship between b and c)

Code Structure

heads.h

include all common reference. I use a static size instead of a dynamic one.If you want to test a large matrix, you can change `SIZE`, `SCALE` and `RANGE`. But make sure your input is valid otherwise the program cannot handle it.

generator.c

`matrix_generator(int nthreads)` generates a random squared matrix(do change with global 2d array `float matrix[SIZE][SIZE]`)

`vector_generator(int nthreads)` generates a random vector(do change with global 2d array `vec[SIZE][1]`).

swap.c

`int find_maxrow(int nthreads, int col)` return the row index which contains maximum number in a specific column. If the largest element in column `col` is 0, print the error message and exit the program.

`void swap(int a, int b)` swap row a and row b in augmented matrix.

Extra time spent in this step is $O(n^2)$, but it will makes the code much robust. The reason already stated in Gaussian Elimination-Pivot Element Method

eliminate.c

`eliminate(int base, int target, int col)` use `base` to eliminate `target`. col is just for fast locate.

`void eliminate_all(int nthreads)` eliminate all rows in the matrix to form a upper triangular matrix.

main.c

`void calculate(int nthreads)` calculates the matrix and put the answer in global 2d array `float answers[SIZE][1]`

`void exporting(void* arr_2d, int rownum, int colnum, char* fname)` export a 2d array to a file.

`int main(int argc, char* argv[])` runner

Use the Program through Terminal

Two kinds of input are accepted

Under windows, if you want to run serial code, please type

`ge.exe` in terminal after compiling(Assume the app name after compiling is ge.exe)

To run parallel code, input

`ge.exe -p num_of_thread`. `num_of_thread` should be a positive integer.

If you use any invalid argument, there will be an error shown in the terminal: "Invalid argument!" and the code will be stopped and exit.

Testing Result

When `SIZE == 3`, `RANGE == 3`, `SCALE == 1`

After Generation

Matrix(matrix.csv):

```
2.000000,2.000000,1.000000,
1.000000,2.000000,1.000000,
-3.000000,-3.000000,1.000000,
```

Vector(vector.csv):

```
-1.000000,
2.000000,
2.000000,
```

After Swapping

Matrix(converted_matrix.csv):

```
-3.000000,-3.000000,1.000000,
1.000000,2.000000,1.000000,
2.000000,2.000000,1.000000,
```

Vector(converted_vector.csv):

```
2.000000,
2.000000,
-1.000000,
```

After Elimination:

Matrix(eliminated_matrix.csv):

```
-3.000000,-3.000000,1.000000,
0.000000,1.000000,1.333333,
0.000000,0.000000,1.666667,
```

Vector(eliminated_vector.csv):

```
2.000000,
2.666667,
0.333333,
```

After Calculation

Result(answer.csv):

```
-3.000000,
2.400000,
0.200000,
```

Test Parallel

To test the efficiency of parallel, comment all exporting functions(I/O cannot be parallel in this project, and it takes too much time) and change the size of matrix into 4096 * 4096, and let `RANGE` = 1000, `SCALE` = 10000

Serial Code:

```
Start Processing......
Matrix size: 4096 * 4096
Infinite number of answers
Spent time:0.273000
```

Parallel Code(12 threads):

```
Start Processing......
Matrix size: 4096 * 4096
You are using 12 threads
Infinite number of answers
Spent time:0.051000
```

It is clear that we can save 80% time through parallel.

Environment

Operation System: Windows 10, 64bit

IDE: VS but finally changed into VSCode

Compiler: VS default but finally GCC(MinGW)

RAM: 16GB

CPU: Intel® Core™ i7-8750H, 6 cores

What to Improve while applying it into Industrial Schema

It is a project at school so I simplified it, just to show how I do gaussian elimination in both serial and parallel ways.

To make it as a library or real calculating app, what we have to do is:

a. `SIZE`, `RANGE`, `SCALE` should not be pre-defined as a static method. The size of matrix and vector to be a dynamic variable defined by user(we need to use `malloc` to create that matrix.)

b. Function `exporting()` should be open or close by user's command, like allow user to add an argument `-l` to export csv files.

c. Allow users to import a file contains matrix

Update Log

2020-08-13: First Version

2020-08-17: update `REAEME.md` and optimize `eliminate.c`

If `multi` == 0, will not do elimination anymore in the current row. Optimization of time: `O(n)` (if `multi` == 0)

2020-09-13: update instruction. The matrix we generated is a squared matrix (length = height)