Analyzing Matrix inversion algorithm

* Algorithm:

For calculate matrix inversion we use LUP decomposition method which describe as bellow:

For given matrix A:

LU factorization with partial pivoting as:

***PA = LU***

***L*** and ***U*** are lower and upper triangular matrices. unique factorization for matrix ***A***

require the lower triangular matrix *L* to be a unit triangular matrix.

***P*** is a permutation matrix which reorders the rows of ***A***.

Then for calculating matrix invers we solve bellow expression in defined manner as bellow:

***PA = LU => AA-1 = LU A-1 = PI:***

We Iteratively move over columns of ***I*** as ***b*** and solve equations:

1. First, we solve the equation ***Ly = Pb*** for y.
2. Second, we solve the equation ***Ux = y*** for x.

* Implementation:

Our code has two main methods:

*static int LUPdecompose(int size, Type A[MAX][MAX], int P[MAX]);*

which return LU matrix in A and permutation matrix in P.

*static int LUPinverse(int size, int P[MAX], Type LU[MAX][MAX],*

*Type B[MAX][MAX], Type X[MAX], Type Y[MAX]);*

which return invers of matrix in A in LU.

* Compilation:

Space complexity of algorithm is ***O(n2)*** which for large size of ***n*** may cause problem due to default stack size per application as ***2MB*** in my OS and compiler base config.

(500 \* 500) Matrix

Because of that I preserve ***4MB*** for stack size to prevent segment fault of code that cause sudden execution termination at the start of running.

For double data type code needed more stack size:

With ***6MB*** stack size I got result.

|  |
| --- |
| gcc -Wl,--stack,4194304 -Wall -pg lup\_matrix\_inverse.c -o2 -o int\_500\_out.exe |

-Wl,option:

Pass option as an option to the linker.

--stack, <size>:

where <size> is in bytes to set the stack size.

-o:

specify output exe file name

(1000 \* 1000) Matrix

For

int data type -> ***12MB*** stack size

float data type -> ***6MB*** stack size

double data type -> ***6MB*** stack size

* Performance profiling with Gprah:

|  |
| --- |
| gprof int\_500\_out.exe > int\_500\_profile-data.txt |

Computing an LU decomposition using either of these algorithms requires  {\textstyle {\frac {2}{3}}n^{3}}{\textstyle {\frac {2}{3}}n^{3}}2/3 \* n^3  floating-point operations

**Theoretical complexity**

 LU decomposition can be computed in time O(*M*(*n*)). *M*(*n*) ≥ *na where a > 2.*

*It means* O(*n*2.376)

Lu decomposition use partial pivoting.

Partial Pivoting

place the largest entry of the first

column of the matrix at the top of that first column. Find largest need O(n2) comparisons.