Report

Eigen open source library

The reason I used Eigen library is it integrated different matrix manipulations inside and all these manipulations are optimized.

Linear Equation Algorithm

For a long period, I have been working on comparing different algorithms to solve our linear equations. These algorithms include LU decomposition, LLT, LDLT, QR, Conjugate Gradient, BiCGSTAB.

After comparing the performances of these algorithms with different input matrices, we decided to use LU decomposition(considering accuracy and efficiency). I tried 3 different versions of LU decomposition, one is called LU which is from Eigen library. One is superLU sequential from SuperLU open source library, and one is superLU distribution version.

These are the benchmarks that I tested for these three algorithms:

	LU	SuperLU	SuperLU_DIST
1000 x 1000 7,000	183ms	110ms 486,022 nnz in L+U	100ms 379,091 nonzeros in L+U
5,000 x 5,000 35,000	15s	9.4s 11,774,945 nnz in L+U	3.31s 9,233,428 nonzeros in L+U
10,000 x 10,000 70,000	134s	80s 46,196,731 nnz in L+U	27s 37,050,042 nonzeros in L+U

15,000 x 15,000 105,000	464s	288s 104,775,128 nnz in L+U	109s 82,640,287 nonzeros in L+U
20,000 x 20,000 140,000	1205s	758s 188,204,860 nnz in L+U	291s 145,082,055 nonzeros in L+U
10,000 x 10,000 50,000	64s	42s 29,741,070 nnz in L+U	18s 4,617,951 nonzeros in L+U
10,000 x 10,000 70,000 (37,050,042 nonzeros in L+U)	134s	80s 46,196,731 nnz in L+U	27s 37,050,042 nonzeros in L+U
10,000 x 10,000 90,000	204s	116s 57,771,962 nnz in L+U	36s 46,084,154 nonzeros in L+U
10,000 x 10,000 110,000	247s	135s 65,415,173 nnz in L+U	42s 53,191,193 nonzeros in L+U
10,000 x 10,000 130,000	272	154s 70,719,216 nnz in L+U	42s 58,585,536 nonzeros in L+U
Jacobi Matrix 1 18,482 x 18,482 (149,892 NNZ)	502s	40ms 512,136 nnz in L+U	150ms (381,662 nonzeros in L+U)
Jacobi Matrix 2 5,492 x 5,492 (37,156 NNZ)	8s	10ms 132,139 nnz in L+U	60ms 108,408 nonzeros in L+U)

Jacobi Matrix 3 6240 x 6240 (41,896 NNZ)	12s	20ms 151,206 nnz in L+U	60ms 122,830 nonzeros in L+U
Dense 200 x 200 40,000 nonzeors in L+U	Paritial Pivoting 2ms	10 ms	10ms
Dense 400 x 400 160,000 nonzeros in L+U	Paritial Pivoting 20ms	20ms	40ms
Dense 600 x 600 360,000 NNZ in L+U	Paritial Pivoting 47ms	40ms	90ms
Dense 800 x 800 640,000 nnz in L+U	Paritial Pivoting 96ms	90ms	160ms
Dense 1000 x 1000 1,000,000 nnz in L+U	Paritial Pivoting 180ms	160ms	X
Jacobi Matrix(case5) 10 x 10 (68 NNZ) 100 nnz in L+U		less than 1ms	less than 1ms
Jacobi Matrix(case2746) 5492 x 5492 (37,100 NNZ)		20 ms 141,255 nnz in L+U	60ms 109,360 nnz in L+U
Jacobi Matrix(case 3120) 6240 x 6240 (41,819 NNZ)		20ms 161,142 nnz in L+U	75ms 125,397 nnz in L+U
Jacobi Matrix(case 9241) 18482 x 18482 (149,874 NNZ)		40ms 574,767 nnz in L+U	140ms 390,769 nnz in L+U

These records are just for solving linear equation time, which means actual execution time would definitely larger than these.

Conclusion: we choose SuperLU sequential

Gaussian Elimination

Implemented this algorithm to decrease the size of original matrices. The codes of this algorithm are included inside superlu.cpp file. But we didn’t use this method in our project even for once.

Bus reform & Y bus generate

use Eigen to implement these two functions, one is to reform bus the other one is to generate y bus. These two functions work perfectly. The codes are included in superlu.cpp file.

ODE45 from ODEint open source library

Used ODEint library to use the ode45(Matlab) in C++, I have googled so many different ode implement in C++ and found out this one is the most accurate as ODE45 in Matlab. The ode works but it runs couple times in one step(which should be exactly one time). I am still working on this to see if there any ways to control this issue.

Compiler Optimization

Modify the compile parameters to -O3 in order to make programs running faster.

Solving equations for multiple times but with same sparsity of A

A x = B, when we tried to solve this equation for multiple times and A1, A2,A3….are the same sparsity with A, we can reuse some datas from the first time. There are four options to do this. The datas below are my result:

Matrix	DOFACT	SamePattern	SampPattern_SameRowPerm	FACTORED
case2746wp	1st: 22ms 2nd: 22ms	1st: 23ms 2nd: 17ms	1st:23ms 2nd:17ms	1st:23ms 2nd:0.658ms
case3120sp	1st: 26ms 2nd: 25ms	1st: 24ms 2nd: 18ms	1st:25ms 2nd:18ms	1st:24ms 2nd:0.605ms

case9241pegase	1st: 69ms 2nd: 65ms	1st: 58ms 2nd: 47ms	1st:56ms 2nd:45ms	1st:56ms 2nd: 1.5 ms