Tugas Kelompok Analisis Numerik : Topik 2

Kelompok B09

Maret 2019

1 Introduction

LU Factorization adalah metode mencari penyelesaian dari persamaan linear yang memiliki banyak variabel. Metode ini diimplementasikan dengan memecah persamaan $Ax = b$ menjadi:

1. $Ax = b$, di mana $A = LU$
2. $PAx = Pb$, sehingga $LUx = Pb$
3. $PAQ = LU$ diubah menjadi $A = P^T LUQ^T$ sehingga menghasilkan $P^T LUQ^T x = b$

Keterangan:

- L = matriks segitiga bawah
- U = matriks segitiga atas
- P = sembarang matriks permutasi (peubah baris)
- Q = sembarang matriks permutasi (peubah kolom)

LU Factorization dapat diimplementasikan secara (1) tanpa pivoting, (2) dengan partial pivoting, dan (3) dengan complete pivoting. Masing-masing implementasi dilakukan terhadap matriks sembarang berukuran $n \times n$ dengan hasil yang sudah ditentukan sebelumnya. Dari proses penghitungan tersebut, dari semua operasi LU Factorization akan dicari:

1. Akurasi
2. Residu
3. Performa

Selain itu, akan dilakukan juga perancangan matriks yang membuat hasil dari operasi *complete pivoting* lebih akurat dibandingkan dengan hasil operasi *partial pivoting*.

2 Why?

Selain karena ini merupakan tugas kelompok mata kuliah analisis numerik yang harus dikerjakan, kami juga tertarik dengan dampak pivoting dan akibatnya terhadap ketiga nilai diatas (akurasi, residu dan performa). Dapatkah pivoting mengatasi ketidakstabilan algoritma yang terjadi dengan sendirinya akibat melakukan *Gaussian Elimination*?

3 Content

Berikut langkah-langkah yang kami lakukan untuk mengimplementasikan algoritma LU Factorization :

1. Menyusun beberapa matriks dengan elemen acak sebagai *sample test case*.
2. Menentukan solusi bagi *sample test case*.
3. Menghitung hasil persamaan dengan menggunakan LU Factorization.
4. Membandingkan solusi *computed* dengan solusi eksak untuk mendapatkan elemen yang dicari

3.1 Menyusun Matriks

Langkah pertama yang kami lakukan adalah menyusun beberapa matriks berelemen acak sebagai *sampel testcase*, yang dengan algoritma yang dilampirkan, kami ulangi beberapa kali untuk mendapatkan beberapa sampel. Kami menggunakan *random number generator* dari Octave yang memiliki jarak antara [0,1]. Jika ditemukan terdapat angka yang lebih besar sama dengan 1, pengacakan matriks kami ulangi agar memiliki *computational error* yang lebih kecil. Berikut adalah matriks - matriks yang kami hasilkan (A_n) beserta solusi eksaknya (x_n) :

$$A_1 = \begin{pmatrix} 0.335394 & 0.242280 & 0.724621 \\ 0.942398 & 0.322238 & 0.843656 \\ 0.453402 & 0.682878 & 0.029942 \end{pmatrix}, x_1 = \begin{pmatrix} 0.018359 \\ 0.857143 \\ 0.908972 \end{pmatrix}$$

$$A_2 = \begin{pmatrix} 0.513457 & 0.698920 & 0.104252 \\ 0.028104 & 0.954041 & 0.707279 \\ 0.858575 & 0.268288 & 0.965124 \end{pmatrix}, x_2 = \begin{pmatrix} 0.382498 \\ 0.837098 \\ 0.095663 \end{pmatrix}$$

$$\vdots$$

$$A_{10} = \begin{pmatrix} 0.620267 & 0.225583 & 0.039757 \\ 0.024812 & 0.012100 & 0.951708 \\ 0.754752 & 0.902093 & 0.301750 \end{pmatrix}, x_{10} = \begin{pmatrix} 0.355533 \\ 0.930304 \\ 0.094666 \end{pmatrix}$$

3.2 Menghitung Solusi Eksak

Langkah kedua, kami menghitung solusi dari matriks-matriks diatas dengan cara mengalikan A_n dengan x_n. Di tahap ini, kami berhipotesis bahwa kami akan mendapat *computational error* yang disebabkan keterbatasan mesin dalam menghitung dan *information loss* akibat pembulatan.

Sebagai contoh, kami melakukan perkalian antara A_{10} dengan x_{10} untuk mendapatkan solusi sebagai berikut :

$$\begin{pmatrix} 0.434149 \\ 0.110172 \\ 1.136125 \end{pmatrix}$$

3.3 Melakukan LU Factorization

Berikutnya, kami mencoba melakukan LU Factorization terhadap matriks-matriks di atas, sebagai contoh mencari LU factorization dari A_{10}.

Hasil LU Factorization tanpa pivoting (dalam format long):

$$L_{10} = \begin{pmatrix} 1.000000000000000 & 0.000000000000000 & 0.000000000000000 \\ 0.040001764258728 & 1.000000000000000 & 0.000000000000000 \\ 1.216819497673394 & 203.987485662375889 & 1.000000000000000 \end{pmatrix}$$

$$U_{10} = \begin{pmatrix} 0.620266552272608 & 0.225583244351472 & 0.039756722171906 \\ 0.000000000000000 & 0.003076653236833 & 0.950117589716836 \\ 0.000000000000000 & 0.000000000000000 & -193.558725412067815 \end{pmatrix}$$

Hasil LU Factorization dengan partial pivoting (dalam format long):

$$L_{10} = \begin{pmatrix} 1.000000000000000 & 0.000000000000000 & 0.000000000000000 \\ 0.821814576370644 & 1.000000000000000 & 0.000000000000000 \\ 0.032874032948365 & 0.034036596867590 & 1.000000000000000 \end{pmatrix}$$

$$U_{10} = \begin{pmatrix} 0.754752434559963 & 0.902092848111913 & 0.301749552568854 \\ 0.000000000000000 & -0.515769807466607 & -0.208225458542498 \\ 0.000000000000000 & 0.000000000000000 & 0.948875490001534 \end{pmatrix}$$

$$P_{10} = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

Hasil LU Factorization dengan complete pivoting (dalam format long):

$$L_{10} = \begin{pmatrix} 1.000000000000000 & 0.000000000000000 & 0.000000000000000 \\ 0.317061089284830 & 1.000000000000000 & 0.000000000000000 \\ 0.041774078970150 & 0.250571874702022 & 1.000000000000000 \end{pmatrix}$$

$$U_{10} = \begin{pmatrix} 1.000000000000000 & 0.000000000000000 & 0.000000000000000 \\ 0.317061089284830 & 1.000000000000000 & 0.000000000000000 \\ 0.041774078970150 & 0.250571874702022 & 1.000000000000000 \end{pmatrix}$$

$$P_{10} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}, Q_{10} = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

3.4 Menghitung Akurasi, Rasidual, dan Performa

Langkah keempat adalah menghitung akurasi, residual, dan performa dari masing-masing matriks yang didapatkan dari strategi pivoting yang berbeda. Penghitungan akurasi dan residual dilakukan dengan membuat kodenya terlebih dahulu, yang bisa dilihat di file "Kode Pendukung".

Untuk strategi 'No Pivoting' menggunakan *norm infinity*:

- Akurasi = $2.98349369395789e - 15$

- Residual = $1.99840144432528e-15$
- Performa = 19 flops. Jika A = matriks $n \times n$, cost= $\frac{2}{3}n^3$

Untuk strategi 'Partial Pivoting' menggunakan *norm infinity*:

- Akurasi = $2.98349369395789e-16$
- Residual = $2.22044604925031e-16$
- Performa = 19 flops + pivoting $O(n^2)$. Jika A = matriks $n \times n$, cost= $\frac{2}{3}n^3$

Untuk strategi 'Complete Pivoting' menggunakan *norm infinity*:

- Akurasi = $2.38679495516631e-16$
- Residual = $2.22044604925031e-16$
- Performa = 19 flops + pivoting $O(n^3)$. Jika A = matriks $n \times n$, cost= $\frac{4}{3}n^3$

4 Conclusion

Berdasarkan analisis dan perhitungan yang kami lakukan, kami menyimpulkan bahwa LU Factorization dapat digunakan untuk menentukan solusi dari sistem persamaan linear yang sulit ditentukan dengan metode eliminasi Gauss.

LU Factorization dapat dilakukan dengan 3 metode, yaitu tanpa pivot, dengan pivot secara parsial, dan dengan pivot secara utuh. Dari ketiga metode tersebut, kami menyimpulkan bahwa secara general, tidak ada strategi yang lebih baik dari yang lainnya, tergantung kebutuhan (time, space, accuracy, kemudahan implementasi, etc). Tentu strategi yang memiliki *time and space complexity* paling rendah adalah no pivoting, walaupun keakuratan dan keberhasilan program dalam mencari solusi tidak baik. Strategi complete pivoting memiliki *time and space complexity* yang sedikit buruk, namun keakuratan dan keberhasilan program dalam mencari solusi yang terbaik. Partial pivoting memiliki *time and space complexity* yang cukup baik, dan secara umum memiliki akurasi yang hampir sama dengan *complete pivoting* (kecuali pada kasus terburuk tertentu). Menurut kami, jika diberi pilihan untuk menentukan strategi pivoting, kami akan memilih untuk menggunakan complete pivoting, karena walaupun sulit diimplementasikan dan kompleksitas pencarian kandidatnya $O(n^3)$ hal ini tidak terlalu signifikan karena *gaussian elimination*-nya sendiri memiliki kompleksitas $O(n^3)$.

5 Additional

Pada bagian ini, kami akan menjawab pertanyaan.
"Apakah anda dapat merancang matriks (yang tidak random) agar hasil complete pivoting lebih akurat secara signifikan dibandingkan partial pivoting?"

Kelompok kami setuju bahwa dapat dirancang matriks tidak random yang menyebabkan hasil *complete pivoting* lebih akurat secara signifikan dibandingkan dengan partial pivoting. Berikut contoh matriksnya:

$$A = \begin{pmatrix} 1 & 0 & 0 & 1 \\ -1 & 1 & 0 & 1 \\ -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & 1 \end{pmatrix}$$

pada partial pivoting, matriks LU yang dihasilkan adalah :

$$L = \begin{pmatrix} 1 & 0 & 0 & 1 \\ -1 & 1 & 0 & 1 \\ -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & 1 \end{pmatrix}, U = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 4 \\ 0 & 0 & 0 & 8 \end{pmatrix}, P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

pada *complete pivoting*, matriks LU yang dihasilkan adalah:

$$L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ -1 & 1 & 1 & 0 \\ -1 & 1 & 1 & 1 \end{pmatrix}, U = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 2 & 1 & 0 \\ 0 & 0 & -2 & 1 \\ 0 & 0 & 0 & -2 \end{pmatrix}, P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, Q = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

Dapat dilihat, ketika ukuran matriks bertambah besar *partial pivoting* menghasilkan matriks LU yang sangat "buruk", elemen terbesarnya bertumbuh sebanyak 2^{p-1}. Sehingga kemungkinan *floating point numbers* tidak lagi dapat merepresentasikannya secara akurat (semakin besar ukurannya akan semakin *sparse* bilangan *floating*-nya), atau bahkan *overflow*. Sedangkan algoritma *complete pivoting* cukup stabil jika dilihat.

6 Reference

1. Chiarandini, Marco (2015, August 19th). *LU Factorization*. Retrieved from http://imada.sdu.dk/ marco/DM554/Slides/dm554-lu.pdf

2. Chen, Oliver Y. K. (2007, February 11th). *A Comparison of Pivoting Strategies for the Direct LU Factorization*. Retrieved from http://archives.math.utk.edu/ICTCM/VOL08/C006/paper.pdf

3. Code With C Team (2015, May 24th). *LU Factorization Method in MATLAB*. Retrieved from www.codewithc.com/lu-factorization-method-in-matlab/

4. Heath, Michael T. (2002). *Scientific Computing An Introductory Survey*. New York. NY: McGraw-Hill

5. Henderson, Nick (2010, April 24th). *LU factorization with complete pivoting.*. Retrieved from www.mathworks.com/matlabcentral/fileexchange/27249-lu-factorization-with-complete-pivoting

6. Leung, K. Ming (2003, February 12th). *Matlab program for LU Factorization using Gaussian elimination without pivoting*. Retrieved from http://cis.poly.edu/ mleung/CS3734/s03/ch02/LU_factor.htm

7. Sauer, Timothy (2012). *Numerical Analysis*. Boston, MA: Pearson Education