Nama / NPM Mahasiswa 1: _____ / _____ Nama / NPM Mahasiswa 2: _____ / _____ Kelas : ____

PR 2 Analisis Numerik

Cholesky Factorization [3 poin]

Cari matriks G sehingga $A = GG^{\top}$ dengan faktorisasi Cholesky di mana

$$A = \begin{pmatrix} 4 & -2 & 0 \\ -2 & 2 & -3 \\ 0 & -3 & 10 \end{pmatrix}$$

Tunjukkan langkah-langkah yang Anda lakukan.

Apabila sebuah matriks (misalkan A) memiliki faktorisasi Cholesky, dikatakan bahwa matriks tersebut definit [ ]. Namun, apabila ketika setelah dikalikan dengan -1 (misalkan menjadi $-A$) baru dapat memiliki faktorisasi Cholesky, matriks tersebut definit [ ].

Least Square Problem [4 poin]

Di Scele telah tersedia tiga buah *source code* bernama `normaleqnsolve.m`, `householdersolver.m`, dan `givensrotationsolver.m` yang berguna untuk menyelesaikan *least square problem*. Lengkapi baris yang ditandai *% TODO* pada *source code* dan tuliskan di kolom yang disediakan di bawah.

`householdersolver.m` baris 32

`givensrotationsolver.m` baris 22 dan baris 23

Dengan memanfaatkan program tersebut, selesaikan SPL $Ax = b$ di bawah

$$\begin{pmatrix} 3 & -1 & 2 \\ 4 & 1 & 0 \\ -3 & 2 & 1 \\ 1 & 1 & 5 \\ -2 & 0 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 4 \\ 5 \\ 0 \\ 7 \\ 1 \end{pmatrix}$$

yang memiliki solusi eksak

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

Perhatikan bahwa karena ada nilai eksak yang memenuhi SPL yang *over-determined* ini, kita peroleh nilai minimal $\|b - Ax\|_2$ adalah 0. Jalankan program yang telah Anda lengkapi dengan parameter A dan b di atas dan tuliskan keluaran apa yang diberikan oleh masing-masing program. Hitung pula *absolute error* $e(\hat{x}) = \|\hat{x} - x\|_2$ dari masing-masing hasil komputasi $\hat{x}$ terhadap solusi eksak x.

	Normal Equation	Householder	Givens Rotation
$\hat{\boldsymbol{x}}$			
$\boldsymbol{R}$			
$\boldsymbol{qb}$			
$\boldsymbol{e}(\hat{\boldsymbol{x}})$			

Note : gunakan perintah `format long` pada MATLAB/Octave agar hasil komputasi yang ditampilkan lebih akurat.

Floating Point Operations [3 poin]

Dalam komputasi numerik, operasi yang paling memakan waktu adalah operasi-operasi *floating-point* (FLOPs := *floating point operations*) yang dilakukan saat menjalankan algorima. FLOPs yang sering muncul adalah operasi aritmatika. Perhatikan bahwa satu FLOP adalah perhitungan untuk satu buah operasi dua buah bilangan *floating-point*.

Sebagai contoh, pada potongan kode

```
>> a = 1.3;
>> b = 2.1;
>> c = a + b;
```

Banyak FLOP yang dijalankan adalah 1, yaitu penjumlahan dua bilangan *floating-point* pada baris ketiga. Perhatikan bahwa baris pertama dan kedua tidak ada FLOP yang dilakukan.

Sementara itu, pada potongan kode

```
>> a = [0.3; 0.1; 0.4; 0.2];
>> b = [1.2; 1.3; 0.3; -0.2]
>> c = a + b;
```

Banyak FLOP yang dilakukan adalah 4 buah dari menghitung `a + b`, di mana terjadi 4 buah penjumlahan *floating-point*; satu untuk masing-masing komponen yang bersesuaian.

Untuk menghitung FLOPs dari suatu implementasi algoritma, Anda selalu dapat menghitungnya langsung dari implementasi algoritmanya pada MATLAB/Octave. Sebagai contoh, berikut adalah contoh implementasi *forward elimination* :

```
 1 function [x] = fe(L, b)
 2     % :param L: lower triangle matrix
 3     % :param b: vector
 4     % :return x: vector such that L * x = b
 5     [n n] = size(L);
 6     x = zeros(n, 1);
 7     for i = 1 : n
 8         x(i) = b(i);
 9         for j = 1 : i-1
10             x(i) = x(i) - L(i, j) * x(j);
11         endfor
12         x(i) = x(i) / L(i, i);
13     endfor
14 endfunction
```

Mari kita analisis algoritma ini. Anda melakukan *for loop* dengan variabel `i` dari 1 hingga n. Dalam *loop* ini dilakukan *for loop* lagi dengan variabel `j` dari 1 hingga i-1. Di dalam *loop* terakhir, pada setiap iterasi terdapat dua FLOP : satu pengurangan dan satu perkalian. Di luar *loop* tersebut, ada satu buah FLOP : satu pembagian. Anda dapat menghitung banyak FLOP pada program ini dalam n (ukuran matriks) dengan formula berikut :

$$\sum_{i=1}^{n}\left(1+\sum_{j=1}^{i-1}2\right)$$

Jika ekspresi sigma kita evaluasi dan disederhanakan, hasilnya adalah

$$\sum_{i=1}^{n}\big(1+2(i-1)\big)=\sum_{i=1}^{n}(2i-1)=2\sum_{i=1}^{n}i-\sum_{i=1}^{n}1=2\frac{n(n+1)}{2}-n=n^2$$

Sehingga diperoleh banyak FLOPs yang dilakukan program di atas tepat n^2 FLOPs.

Misalkan α, β adalah skalar, $\vec{u}, \vec{v}$ adalah vektor $N \times 1$, A matriks $M \times N$, B matriks $N \times L$, dan R matriks triangular berukuran $N \times N$. Anda dapat menggunakan tabel di bawah untuk membantu perhitungan jumlah FLOPs.

Operasi	**#FLOPs**
$sign(\alpha), \alpha^2, \sqrt{\alpha}$	1
$\alpha+\beta, \alpha-\beta, \alpha\cdot\beta, \alpha/\beta$	1
$\alpha * \vec{u}$	N
$\vec{u}+\vec{v}, \vec{u}-\vec{v}$	N
$\vec{u}^T * \vec{v}$	$2N-1$
$\|\vec{v}\|_2$	$2N$
$A * \vec{v}$	$M(2N-1)$
$A * B$	$ML(2N-1)$
$A\backslash\vec{v}$	$\frac{2}{3}N^3$
$R\backslash\vec{v}$	N^2

Hitunglah jumlah FLOPs pada program `householdersolver.m` **atau** `givensrotationsolver.m` untuk menyelesaikan SPL over-determined $Ax = b$ pada soal sebelumnya, untuk $m = 5$ dan $n = 3$.

Berikan penjelasan hitungan Anda dalam m dan n terlebih dahulu, kemudian substitusi nilai $m = 5$ dan $n = 3$ pada ekspresi yang Anda peroleh.

Tuliskan secara eksplisit program apa yang akan Anda hitung FLOPs-nya.