Laporan - Praktikum Sistem Kontrol Optimal (SVIK214509)

Nama Kelompok	Kelompok 12	
Anggota	1) Alim Satria Fi'i Wijaya Kusuma	21/483503/SV/20304
	2) Dani Yudha Kusuma	21/483564/SV/20362
Hari/Tanggal Pelaksanaan	24 Agustus 2022 dan 31 Agustus 2022	
Modul ke -	1 dan 2	
Dosen	Jans Hendry, S.T., M.Eng.	
Asisten	1) Ervan Abi Surya 2) Fakhruddin Hanif N	

Paraf Asisten	Paraf Dosen

1.	Operasi sederhana pada 2 matrix (transpose, jumlah, kurangi, dan perkalian)
a.	Operasi transpose
	`` static void transpose_matrix(int row, int col, int matrix_result[row][col],int matrix_input[row][col]){ int matrix_tmp[row][col]; for (int x = 0; x < row; x++){ for (int y = 0; y < col; y++) matrix_tmp[y][x] = matrix_input[x][y]; } for (int x = 0; x < row; x++){ for (int y = 0; y < col; y++) matrix_result[x][y] = matrix_tmp[x][y]; } } ``` Operasi transpose pada matrix dilakukan dengan menukar baris dengan kolom sehingga hasil akhir transpose merupakan kebalikan baris dan kolom dari matriks yang sebelumnya.
b.	Operasi penjumlahan
	`` static void add_matrix(int row, int col, int matrix_result[row][col],int matrix_input_1[row][col],int matrix_input_2[row][col]){ for (int x = 0; x < row; x++){ for (int y = 0; y < col; y++) matrix_result[x][y] = matrix_input_1[x][y] + matrix_input_2[x][y]; } } ``` Operasi penjumlahan nilai matrix dilakukan dengan menjumlahkan isi matrix yang memiliki posisi baris dan kolom yang sama.
c.	Operasi Pengurangan

	`` static void subtract_matrix(int row, int col, int matrix_result[row][col],int matrix_input_1[row][col],int matrix_input_2[row][col]){ for (int x = 0; x < row; x++){ for (int y = 0; y < col; y++) matrix_result[x][y] = matrix_input_1[x][y] - matrix_input_2[x][y]; } } ``` Operasi pengurangan pada matrix dapat dilakukan dengan mengurangi isi matrix yang memiliki posisi baris dan kolom yang sama.
d.	Operasi Perkalian
	`` static void multiply_matrix(int row, int col, int matrix_result[row][col],int matrix_input_1[row][col],int matrix_input_2[row][col]){ for (int x = 0; x < row; x++){ for (int y = 0; y < col; y++){ matrix_result[x][y] = 0; for (int z = 0; z < col; z++) matrix_result[x][y] += matrix_input_1[x][z]*matrix_input_2[z][y]; } } } ``` Operasi perkalian matriks dilakukan dengan mengalikan isi matriks secara silang, isi matriks dengan posisi baris dikalikan dengan isi matriks di bagian kolom.
e.	Main Program
	Program bekerja dengan menggunakan fungsi-fungsi yang telah dideskripsikan diatas. terdapat beberapa fungsi penting yang digunakan pada main program. fungsi-fungsi tersebut antara lain, ● input_matrix : berfungsi untuk memberikan input pada matrix ● print_matrix : berfungsi untuk mencetak matrix ● add_matrix : berfungsi untuk menjumlahkan matrix ● subtract_matrix : berfungsi untuk melakukan pengurangan pada matrix ● multiply_matrix : berfungsi untuk melakukan operasi perkalian pada matrix

```c
#define ROW 2
#define COL 2

#include "MatrixLib.h"

struct matrix_t{
    int data[2][2];
};

struct matrix_t A,B,C;

int main(void){

    // INPUT MATRIX
    input_matrix(ROW,COL,A.data);
    printf("Print Matrix A : \r\n");
    print_matrix(ROW,COL,A.data);

    input_matrix(ROW,COL,B.data);
    printf("Print Matrix B : \r\n");
    print_matrix(ROW,COL,B.data);

    // ADD MATRIX
    add_matrix(ROW,COL,C.data,A.data,B.data);
    printf("Add Matrix : \r\n");
    print_matrix(ROW,COL,C.data);

    // SUBTRACT MATRIX
    subtract_matrix(ROW,COL,C.data,A.data,B.data);
    printf("Subtract Matrix : \r\n");
    print_matrix(ROW,COL,C.data);

    // MULTIPLY MATRIX
    multiply_matrix(ROW,COL,C.data,A.data,B.data);
    printf("Multiply Matrix : \r\n");
    print_matrix(ROW,COL,C.data);

    //TRANSPOSE MATRIX
    transpose_matrix(ROW,COL,A.data,A.data);
    printf("Transpose Matrix A : \r\n");
    print_matrix(ROW,COL,A.data);

    transpose_matrix(ROW,COL,B.data,B.data);
    printf("Transpose Matrix B : \r\n");
    print_matrix(ROW,COL,B.data);

    return 0;
}
```

Hasil Program

```
matrix[0][0] = 1
matrix[0][1] = 2
matrix[1][0] = 3
matrix[1][1] = 4

Print Matrix A :

1       2

3       4


matrix[0][0] = 5
matrix[0][1] = 6
matrix[1][0] = 7
matrix[1][1] = 8

Print Matrix B :

5       6

7       8
```

```
Transpose Matrix A :

1       3

2       4

Transpose Matrix B :

5       7

6       8
```

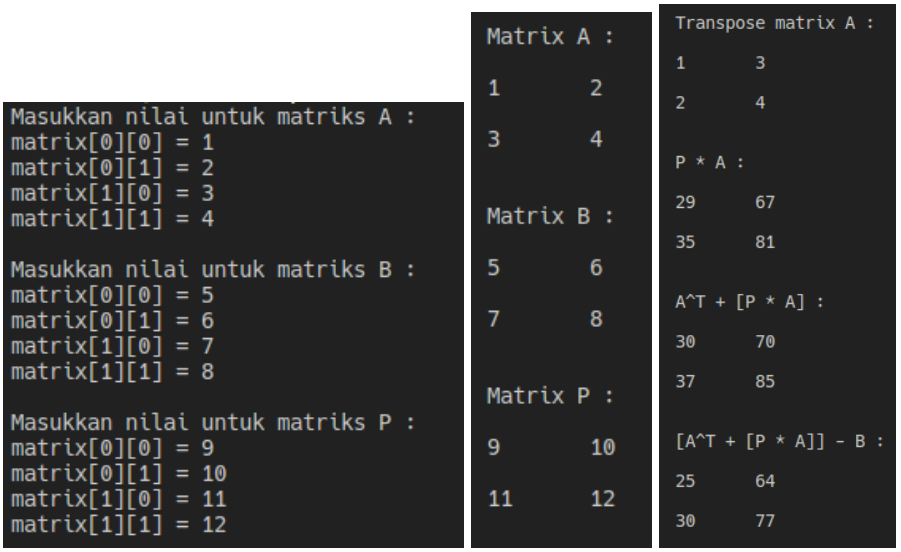
```
Add Matrix :

6       8

10      12

Subtract Matrix :

-4      -4

-4      -4

Multiply Matrix :

19      22

43      50
```

2. Program untuk menyelesaikan persamaan $A^T + P * A - B$ (Menggunakan library yang sama dengan nomor 1)

```c
#define ROW 2
#define COL 2

#include "MatrixLib.h"

struct matrix_t{
    int data[2][2];
};

struct matrix_t A,B,P, C,D,E, F;

int main (){

    // Berapakah hasil dari persamaan berikut:
    // A^T+P*A-B = C+P*A-B
    // Dengan nilai matriks A, P dan B ditentukan oleh kalian sendiri

    printf("Masukkan nilai untuk matriks A : \n");
    input_matrix(ROW,COL,A.data);
    printf("Masukkan nilai untuk matriks B : \n");
    input_matrix(ROW,COL,B.data);
    printf("Masukkan nilai untuk matriks P : \n");
    input_matrix(ROW,COL,P.data);

    printf("Matrix A :\n");
    print_matrix(ROW,COL,A.data);
    printf("Matrix B :\n");
    print_matrix(ROW,COL,B.data);
    printf("Matrix P :\n");
    print_matrix(ROW,COL,P.data);

    transpose_matrix(ROW,COL,A.data,A.data);
    printf("Transpose matrix A : \n");
    print_matrix(ROW,COL,A.data);

    multiply_matrix(ROW,COL,C.data,P.data,A.data);
    printf("P * A : \n");
    print_matrix(ROW,COL,C.data);

    add_matrix(ROW,COL,D.data,A.data,C.data);
    printf("A^T + [P * A] : \n");
    print_matrix(ROW,COL,D.data);

    subtract_matrix(ROW,COL,E.data,D.data,B.data);
    printf("[A^T + [P * A]] - B : \n");
    print_matrix(ROW,COL,E.data);

    return 0;
}
```

Operasi persamaan pada soal diatas menggunakan beberapa fungsi pada library yang digunakan pada library satu. fungsi-fungsi tersebut antara lain,

- input_matrix
- print_matrix

	• transpose_matrix • add_matrix • subtract_matrix • multiply_matrix
	Hasil Program
	
3.	Operasi Matriks Mencari Determinan, Minor, Cofactor, Adjoint (Transpose Cofactor), dan Invers
a.	Mencari Minor
	`` static void minor_matrix(int row, int col, int matrix_result[row][col][row - 1][col- 1],int matrix_input[row][col],int dimension){ for(int target_row = 0; target_row < row; target_row++){ for(int target_col = 0; target_col < col; target_col++){ int minor_row = 0,minor_col = 0; for (int x = 0; x < row; x++){ // row matrix for (int y = 0; y < col; y++){ // col matrix if(x != target_row && y != target_col) matrix_result[target_row][target_col][minor_row][minor_col++] = matrix_input[x][y]; if(minor_col == dimension - 1){ // dimension of minor matrix minor_col = 0; // reset minor col minor_row++; // add new minor row } } } } } } ``` matriks minor dapat dicari dengan menampung isi matrix selain isi matrix yang memiliki baris atau kolom yang sama dengan matriks minor yang dicari.
b.	Mencari matriks cofactor

```
static void get_cofactor(int row, int col, int matrix_result[row][col],int matrix_input[row][col][row - 1][col- 1]){

    for (int x = 0; x < row; x++){
        for (int y = 0; y < col; y++){
            matrix_result[x][y] = (matrix_input[x][y][0][0]*matrix_input[x][y][1][1]) - (matrix_input[x][y][0][1]*matrix_input[x][y][1][0]);
        }
    }

```

Matriks cofactor dapat dicari dengan mencari determinan matriks minor. determinan tersebut lalu ditampung di sebuah matriks. matriks tersebut merupakan matriks cofactor.

c. Mencari Adjoint (Transpose Matriks cofactor)

```
static void transpose_matrix(int row, int col, int matrix_result[row][col],int matrix_input[row][col]){

    int matrix_tmp[row][col];

    for (int x = 0; x < row; x++){
        for (int y = 0; y < col; y++)
            matrix_tmp[y][x] = matrix_input[x][y];
    }

    for (int x = 0; x < row; x++){
        for (int y = 0; y < col; y++)
            matrix_result[x][y] = matrix_tmp[x][y];
    }
}
```

Mencari adjoint dapat dilakukan dengan mencari transpose dari cofactor matriks.

d. Mencari invers matriks

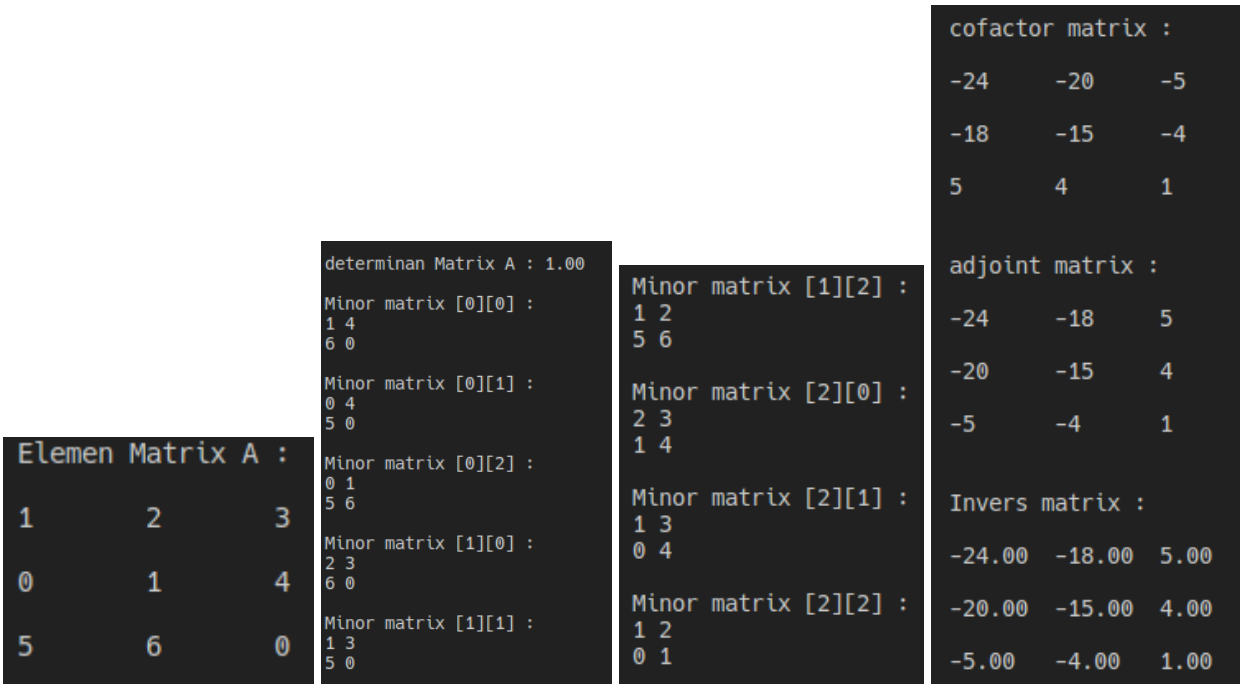
```
static void invers_matrix(int row, int col, float matrix_result[row][col],int matrix_input[row][col],float determinant){

    for (int x = 0; x < row; x++){
        for (int y = 0; y < col; y++){
            matrix_result[x][y] = matrix_input[x][y] / fabs(determinant);
        }
    }
}
```

Invers matriks dapat dicari dengan membagi matriks adjoint dengan determinant yang diabsolutkan

e. Main Program

```c
#define ROW 3
#define COL 3

#define MINOR_ROW ROW-1
#define MINOR_COL COL-1

#define DIMENSION 3

#include "MatrixLib.h"

int A[ROW][COL] = {1,2,3,0,1,4,5,6,0};
int B[ROW][COL][MINOR_ROW][MINOR_COL];
int C[ROW][COL];
float D[ROW][COL];

int main(){

    // input_matrix(A);
    printf("Elemen Matrix A : \n");
    print_matrix(ROW,COL,A);
    printf("determinan Matrix A : %0.2f\n\n",determinant_matrix(ROW,COL,A));
    minor_matrix(ROW,COL,B,A,DIMENSION);
    print_minor_matrix(ROW,COL,B);
    get_cofactor(ROW,COL,C,B);
    printf("cofactor matrix :\n");
    print_matrix(ROW,COL,C);
    printf("adjoint matrix :\n"); //adjoint is transpose of cofactor matrix
    transpose_matrix(ROW,COL,C,C);
    print_matrix(ROW,COL,C);
    printf("Invers matrix : \n");
    invers_matrix(ROW,COL,D,C,determinant_matrix(ROW,COL,A));
    print_float_matrix(ROW,COL,D);

    return 0;
}
```

Hasil Program



4.	Eigen Value & Eigen Vector
	Main Program
5.	Mencari Hessian Matrix
a.	Mencari turunan F(x,y)

```
void get_differential_multivar(int row,int col, int diff_x, int diff_y, int function_input[row][col], int
function_output[row][col]){

    int function_temp[row][col];

    if(diff_x != 0){
        for (int i = 0; i < row; i++){
            for (int j = 0; j < col; j++){
                int x = (diff_x != diff_y) ? (get_factorial(j)) : (j * diff_x);
                function_temp[i][j - diff_x] = function_input[i][j] * x;
            }
        }
    }
    else
        transfer_matrix(row,col,function_temp,function_input);

    if(diff_y != 0){
        for (int i = 0; i < row; i++){
            for (int j = 0; j < col; j++){
                int x = (diff_x != diff_y) ? (get_factorial(i)) : (i * diff_y);
                function_output[i - diff_y][j] = function_temp[i][j] * x;
            }
        }
    }
    else
        transfer_matrix(row,col,function_output,function_temp);

    for(int i = 0; i < row; i++){ // clean error value
        for(int j = 0; j < col; j++)
            if(i >= row - diff_y || j >= col - diff_x)
                function_output[i][j] = 0;
    }
}
```

Fungsi ini digunakan untuk menghitung nilai turunan dari fungsi input yang diberikan. fungsi memberikan turunan terhadap x dan y, nilai turunan dapat diatur (dapat diturunkan satu kali atau dua kali).

b.	Mencari hessian matrix

```
void hessian_matrix(int row, int col,int function_input[row][col]){

    int x = 0;

    print("\n");

    for (int i = 0; i < 2; i++){
        for (int j = 0; j < 2; j++){
            printf("Matrix hessian [%d][%d] : \n",i,j);
            get_differential_multivar(row,col,step[x][0],step[x][1],function_input,hessian_output[i][j].data);
            print_function_matrix(row,col,hessian_output[i][j].data);
            x++;
        }
    }
}
```

Fungsi ini bekerja dengan memasukkan fungsi input kedalam fungsi turunan dan mencetak matrix hessian.

c.	Main program

```c
#define poly_x 3
#define poly_y 3

#define x_index poly_x + 1 //COL
#define y_index poly_y + 1 //ROW

#include "CalculusLib.h"

static int function_input[4][4] = {0,   0,  3,  0, // f(x,y) = x^3*y^3 + 3x^2 + y^2
                                   0,   0,  0,  0,
                                   1,   0,  0,  0,
                                   0,   0,  0,  1};

int function_output[4][4];

int main(){

    printf("Representasi fungsi polynomial dengan matrix :\n");
    print_function_matrix(y_index,x_index,function_input);
    printf("Hessian matrix : \n");
    hessian_matrix(y_index,x_index,function_input,2,-1);

    return 0;

}
```

d. Hasil Program

```
Representasi fungsi polynomial dengan matrix :
0       0       3       0
0       0       0       0
1       0       0       0
0       0       0       1
```

```
Hessian matrix :
Matrix hessian [0][0] :
6       0       0       0
0       0       0       0
0       0       0       0
0       6       0       0

Matrix hessian [0][1] :
0       0       0       0
0       0       0       0
0       0       9       0
0       0       0       0
```

```
Matrix hessian [1][0] :
0       0       0       0
0       0       0       0
0       0       9       0
0       0       0       0

Matrix hessian [1][1] :
2       0       0       0
0       0       0       6
0       0       0       0
0       0       0       0
```