Sistem Kontrol Optimal Modul 1 & 2

Kelompok 12 :

Alim Satria Fi'i Wijaya Kusuma
Dani Yudha Kusuma

Operasi Matematika sederhana pada matriks

Penjumlahan

```c
1 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]){
2
3     for (int x = 0; x < row; x++){
4         for (int y = 0; y < col; y++)
5             matrix_result[x][y] = matrix_input_1[x][y] + matrix_input_2[x][y];
6     }
7 }
```

Pengurangan

```
1 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]){
2
3     for (int x = 0; x < row; x++){
4         for (int y = 0; y < col; y++)
5             matrix_result[x][y] = matrix_input_1[x][y] - matrix_input_2[x][y];
6     }
7 }
```

Perkalian

```c
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];
        }
    }
}
```

Transpose

```c
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];
    }
}
```

Program 1 Modul 1

```c
1 #define ROW 2
2 #define COL 2
3
4 #include "MatrixLib.h"
5
6 struct matrix_t{
7     int data[2][2];
8 };
9
10 struct matrix_t A,B,C;
11
12 int main(void){
13
14     // INPUT MATRIX
15     input_matrix(ROW,COL,A.data);
16     printf("Print Matrix A : \r\n");
17     print_matrix(ROW,COL,A.data);
18
19     input_matrix(ROW,COL,B.data);
20     printf("Print Matrix B : \r\n");
21     print_matrix(ROW,COL,B.data);
22
```

```c
23     // ADD MATRIX
24     add_matrix(ROW,COL,C.data,A.data,B.data);
25     printf("Add Matrix : \r\n");
26     print_matrix(ROW,COL,C.data);
27
28     // SUBTRACT MATRIX
29     subtract_matrix(ROW,COL,C.data,A.data,B.data);
30     printf("Subtract Matrix : \r\n");
31     print_matrix(ROW,COL,C.data);
32
33     // MULTIPLY MATRIX
34     multiply_matrix(ROW,COL,C.data,A.data,B.data);
35     printf("Multiply Matrix : \r\n");
36     print_matrix(ROW,COL,C.data);
37
38     //TRANSPOSE MATRIX
39     transpose_matrix(ROW,COL,A.data,A.data);
40     printf("Transpose Matrix A : \r\n");
41     print_matrix(ROW,COL,A.data);
42
43     transpose_matrix(ROW,COL,B.data,B.data);
44     printf("Transpose Matrix B : \r\n");
45     print_matrix(ROW,COL,B.data);
46
```

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
```

```
Add Matrix :

6          8

10         12

Subtract Matrix :

-4         -4

-4         -4

Multiply Matrix :

19         22

43         50
```

```
Transpose Matrix A :

1          3

2          4

Transpose Matrix B :

5          7

6          8
```

Program 2 Modul 1

```c
1 #define ROW 2
2 #define COL 2
3
4 #include "MatrixLib.h"
5
6 struct matrix_t{
7     int data[2][2];
8 };
9
10 struct matrix_t A,B,P, C,D,E, F;
11
12 int main (){
13
14     // Berapakah hasil dari persamaan berikut:
15     // A^T+P*A-B = C+P*A-B
16     // Dengan nilai matriks A, P dan B ditentukan oleh kalian sendiri
17
18     printf("Masukkan nilai untuk matriks A : \n");
19     input_matrix(ROW,COL,A.data);
20     printf("Masukkan nilai untuk matriks B : \n");
21     input_matrix(ROW,COL,B.data);
22     printf("Masukkan nilai untuk matriks P : \n");
23     input_matrix(ROW,COL,P.data);
24
```

```c
25     printf("Matrix A :\n");
26     print_matrix(ROW,COL,A.data);
27     printf("Matrix B :\n");
28     print_matrix(ROW,COL,B.data);
29     printf("Matrix P :\n");
30     print_matrix(ROW,COL,P.data);
31
32     multiply_matrix(ROW,COL,C.data,P.data,A.data);
33     printf("P * A : \n");
34     print_matrix(ROW,COL,C.data);
35
36     transpose_matrix(ROW,COL,A.data,A.data);
37     printf("Transpose matrix A : \n");
38     print_matrix(ROW,COL,A.data);
39
40     add_matrix(ROW,COL,D.data,A.data,C.data);
41     printf("A^T + [P * A] : \n");
42     print_matrix(ROW,COL,D.data);
43
44     subtract_matrix(ROW,COL,E.data,D.data,B.data);
45     printf("[A^T + [P * A]] - B : \n");
46     print_matrix(ROW,COL,E.data);
47
48     return 0;
49 }
```

Hasil Program

```
Masukkan nilai untuk matriks A :
matrix[0][0] = 1
matrix[0][1] = 2
matrix[1][0] = 3
matrix[1][1] = 4

Masukkan nilai untuk matriks B :
matrix[0][0] = 5
matrix[0][1] = 6
matrix[1][0] = 7
matrix[1][1] = 8

Masukkan nilai untuk matriks P :
matrix[0][0] = 9
matrix[0][1] = 10
matrix[1][0] = 11
matrix[1][1] = 12
```

```
Matrix A :

1        2

3        4

Matrix B :

5        6

7        8

Matrix P :

9        10

11       12
```

```
P * A :

39       58

47       70

Transpose matrix A :

1        3

2        4

A^T + [P * A] :

40       61

49       74

[A^T + [P * A]] - B :

35       55

42       66
```

determinant ,minor, cofactor, adjoint dan invers

Determinant

```c
static float determinant_matrix(int row, int col, int matrix[row][col]){

    float a = 0,b = 0,c = 0;

    if (row == 3 && col == 3) {

        a = matrix[0][0]*(matrix[1][1]*matrix[2][2] - matrix[1][2]*matrix[2][1]);
        b = matrix[0][1]*(matrix[1][0]*matrix[2][2] - matrix[1][2]*matrix[2][0]);
        c = matrix[0][2]*(matrix[1][0]*matrix[2][1] - matrix[1][1]*matrix[2][0]);

        return a-b+c;
    }

    else if (row == 2 && col == 2) {

        a = matrix[0][0]*matrix[1][1];
        b = matrix[0][1]*matrix[1][0];

        return a-b;

    }
}
```

Minor

```c
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
                    }
                }
            }
        }
    }
}
```

Cofactor

```
1  static void get_cofactor(int row, int col, int matrix_result[row][col],int matrix_input[row][col][row - 1][col- 1]){
2
3      for (int x = 0; x < row; x++){
4          for (int y = 0; y < col; y++){
5              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]);
6          }
7      }
8
```

Adjoint = Transpose cofactor

```c
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];
    }
}
```

Invers

```c
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);
        }
    }
}
```

Program 3 Modul 1

```
1  #define ROW 3
2  #define COL 3
3
4  #define MINOR_ROW ROW-1
5  #define MINOR_COL COL-1
6
7  #define DIMENSION 3
8
9  #include "MatrixLib.h"
10
11 int A[ROW][COL] = {1,2,3,
12                    0,1,4,
13                    5,6,0};
14
15 int B[ROW][COL][MINOR_ROW][MINOR_COL];
16 int C[ROW][COL];
17 float D[ROW][COL];
18
```

```
18
19 int main(){
20
21    // input_matrix(A);
22    printf("Elemen Matrix A : \n");
23    print_matrix(ROW,COL,A);
24    printf("determinan Matrix A : %0.2f\n\n",determinant_matrix(ROW,COL,A));
25    minor_matrix(ROW,COL,B,A,DIMENSION);
26    print_minor_matrix(ROW,COL,B);
27    get_cofactor(ROW,COL,C,B);
28    printf("cofactor matrix :\n");
29    print_matrix(ROW,COL,C);
30    printf("adjoint matrix :\n"); //adjoint is transpose of cofactor matrix
31    transpose_matrix(ROW,COL,C,C);
32    print_matrix(ROW,COL,C);
33    printf("Invers matrix : \n");
34    invers_matrix(ROW,COL,D,C,determinant_matrix(ROW,COL,A));
35    print_float_matrix(ROW,COL,D);
36
37    return 0;
38 }
```

Hasil Program

```
Elemen Matrix A :

1        2        3

0        1        4

5        6        0

determinan Matrix A : 1.00
```

```
Minor matrix [0][0] :
1 4
6 0

Minor matrix [0][1] :
0 4
5 0

Minor matrix [0][2] :
0 1
5 6

Minor matrix [1][0] :
2 3
6 0

Minor matrix [1][1] :
1 3
5 0
```

```
Minor matrix [1][2] :
1 2
5 6

Minor matrix [2][0] :
2 3
1 4

Minor matrix [2][1] :
1 3
0 4

Minor matrix [2][2] :
1 2
0 1
```

```
cofactor matrix :

-24      -20      -5

-18      -15      -4

5        4        1

adjoint matrix :

-24      -18      5

-20      -15      4

-5       -4       1

Invers matrix :

-24.00   -18.00   5.00

-20.00   -15.00   4.00

-5.00    -4.00    1.00
```

Eigen Value

Eigen value

```c
1 static void eigen_val(int row, int col, int matrix[row][col]){
2
3     float lamda[2];
4     int matrix_temp[2][2];
5
6     // solve using quadratic formula
7
8     int a = 1;
9     int b = -1 * (matrix[0][0] + matrix[1][1]);
10    int c = (-1 * matrix[0][0] * -1 * matrix[1][1]) - (-1 * matrix[0][1] * -1 * matrix[1][0]);
11
12    printf("%d\n",b);
13    printf("%d\n",c);
14
15    print_matrix(2,2,matrix);
16
17    for(int x = 1; x <= 2; x++){
18        lamda[x] = (-b + pow(-1,x) * sqrt(pow(b,2) - (4 * a *c)))/ (2 * a);
19        printf("lamda [%d] : %0.2f\n",x,lamda[x]);
20    }
21 }
```

Program 4 Modul 1

```c
1  #include "MatrixLib.h"
2
3  static int A[2][2] = {3,0,
4                        8,-1};
5
6  int main(){
7
8      eigen_val(2,2,A);
9
10     return 0;
11 }
```
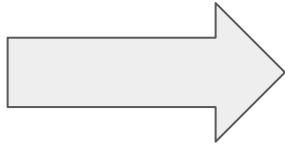
Hasil Program

```
3        0

8       -1

lamda [1] : -1.00
lamda [2] : 3.00
```

Mencari matriks hessian

Representasi Fungsi ke matriks

$$f(x, y) = x^3y^3 + 3x^2 + y^2$$

	x^0	x^1	x^2	x^3
y^0	0	0	3	0
y^1	0	0	0	0
y^2	1	0	0	0
y^3	0	0	0	1

Fungsi turunan terhadap x y

```c
1 void get_differential_multivar(int row,int col, int diff_x, int diff_y, int function_input[row][col], int
  function_output[row][col]){
2
3     int function_temp[row][col];
4
5     if(diff_x != 0){
6         for (int i = 0; i < row; i++){
7             for (int j = 0; j < col; j++){
8                 int x = (diff_x != diff_y) ? (get_factorial(j)) : (j * diff_x);
9                 function_temp[i][j - diff_x] = function_input[i][j] * x;
10            }
11        }
12    }
13    else
14        transfer_matrix(row,col,function_temp,function_input);
15
16    if(diff_y != 0){
17        for (int i = 0; i < row; i++){
18            for (int j = 0; j < col; j++){
19                int x = (diff_x != diff_y) ? (get_factorial(i)) : (i * diff_y);
20                function_output[i - diff_y][j] = function_temp[i][j] * x;
21            }
22        }
23    }
24    else
25        transfer_matrix(row,col,function_output,function_temp);
26
27    for(int i = 0; i < row; i++){ // clean error value
28        for(int j = 0; j < col; j++)
29            if(i >= row - diff_y || j >= col - diff_x)
30                function_output[i][j] = 0;
31    }
32 }
```

Menghitung matriks hessian

```c
1 void hessian_matrix(int row, int col,int function_input[row][col]){
2
3     int x = 0;
4
5     print("\n");
6
7     for (int i = 0; i < 2; i++){
8         for (int j = 0; j < 2; j++){
9             printf("Matrix hessian [%d][%d] : \n",i,j);
10            get_differential_multivar(row,col,step[x][0],step[x][1],function_input,hessian_output[i][j].data);
11            print_function_matrix(row,col,hessian_output[i][j].data);
12            x++;
13        }
14    }
15 }
```

Program 1 Modul 2

```c
1 #define poly_x 3
2 #define poly_y 3
3
4 #define x_index poly_x + 1 //COL
5 #define y_index poly_y + 1 //ROW
6
7 #include "CalculusLib.h"
8
9 static int function_input[4][4] = {0,   0,  3,  0, // f(x,y) = x^3*y^3 + 3x^2 + y^2
10                                    0,   0,  0,  0,
11                                    1,   0,  0,  0,
12                                    0,   0,  0,  1};
13
14 int function_output[4][4];
15
16 int main(){
17
18     printf("Representasi fungsi polynomial dengan matrix :\n");
19     print_function_matrix(y_index,x_index,function_input);
20     printf("Hessian matrix : \n");
21     hessian_matrix(y_index,x_index,function_input,2,-1);
22
23     return 0;
24
25 }
```

Hasil program


```
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
```
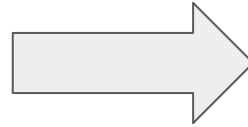
$$\frac{\partial^2}{\partial x^2} \Rightarrow 6xy^3 + 6$$

$$\frac{\partial^2}{\partial x \partial y} \Rightarrow 9x^2 y^2$$

Hasil program


```
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
```
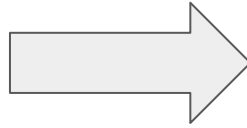
$$\frac{\partial^2}{\partial y \partial x} \Rightarrow 9x^2 y^2$$

$$\frac{\partial^2}{\partial y^2} \Rightarrow 6x^3 y + 2$$

Terimakasih