20.05.2024

Yıldız Teknik Üniversitesi

Elektrik-Elektronik Fakültesi

Bilgisayar Mühendisliği Bölümü

BLM1022

Sayısal Analiz

Gr: 1

Prof. Dr. Banu Diri

Dönem Projesi

İsim: Ayşe Sude Cami

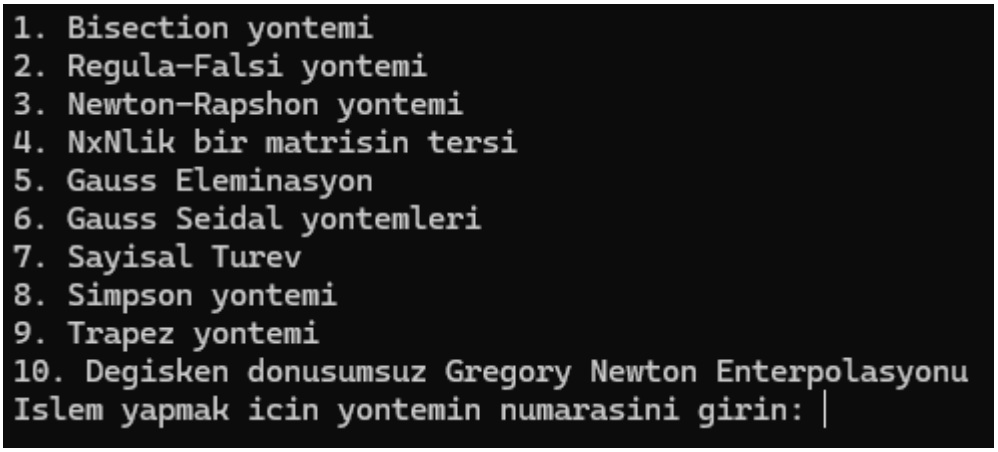
No: 22011064

E-posta: sude.cami@std.yildiz.edu.tr

İÇİNDEKİLER

ÖN BİLGİ

Programı çalıştırdığınızda hangi yöntemi kullanmak istediğiniz sorulur. Yöntemin numarasını girin.



Fonksiyon yazmanızı gerektiren yöntemlerde yazacağınız fonksiyonda sadece polinom ifadeler kullanabilirsiniz.

Paranteze alma “(“ ve üs alma “^” var. “e” sabitini fonksiyonda kullanabilirsiniz.

Önemli Kurallar

Tüm çarpma işlemleri için çarpma işareti koymalısınız.

Örneğin 2x değil 2*x veya 5(x^3-6) değil 5*(x^3-6) şeklinde yazmalısınız.

Fonksiyon girişinde virgüllü ifadeler bölüm olarak verilmeli.

Örneğin 0.2*x şeklinde değil (1/5)*x şeklinde yazılmalı.

Değer girerken nokta kullanmalısınız.

Örneğin x değerini girerken 0,2 değil 0.2 şeklinde yazılmalı.

Yaptığım Yöntemler									
1	2	3	4	5	6	7	8	9	10
1	1	1	1	1	1	1	1	1	1

MAIN FONKSİYONU

```c
int main() {
    char expression[MAX_SIZE];
    int result;
    int yontem;
    printf("1. Bisection yontemi\n"
           "2. Regula-Falsi yontemi\n"
           "3. Newton-Rapshon yontemi\n"
           "4. NxNlik bir matrisin tersi\n"
           "5. Gauss Eleminasyon\n"
           "6. Gauss Seidal yontemleri\n"
           "7. Sayisal Turev\n"
           "8. Simpson yontemi\n"
           "9. Trapez yontemi\n"
           "10. Degisken donusumsuz Gregory Newton Enterpolasyonu\n"
           "Islem yapmak icin yontemin numarasini girin: ");

    scanf("%d", &yontem);

    switch(yontem) {
        case 1:
        BisectionFonkAl();
            break;
        case 2:
        RegulaFalsiFonkAl();
            break;
        case 3:
        NewtonRaphsonFonkAl();
            break;
        case 4:
        MatrisTersiAl();
            break;
        case 5:
        GauusElemeYontemi();
            break;
        case 6:
        GaussSeidelFonkAl();
            break;
        case 7:
        SayisalTurevFonkAl();
            break;
        case 8:
        Simpson13FonkAl();
            break;
        case 9:
        TrapezFonkAl();
            break;
        case 10:
        GregoryNewtonEnt();
            break;
        default:
            printf("Gecersiz yontem girdiniz.");
            break;
    }

    return 0;
}
```

PROGRAMDA KULLANDIĞIM FONKSİYONLAR

```c
// Fonksiyon alma fonksiyonları (Shunting Yard)
void initializeStack(struct Stack *stack, int size);
void destroyStack(struct Stack *stack);
int isEmpty(struct Stack *stack);
int isFull(struct Stack *stack);
char peek(struct Stack *stack);
char pop(struct Stack *stack);
void push(char element, struct Stack *stack);
int precedence(char ch);
int isOperand(char element);
char* infixToPostfix(char *expressionArray, struct Stack *stack);

// Yöntem fonksiyonları
void BisectionFonkAl();
double bisection(double (*f)(char*, double), double a, double b, double tol, char *postfix);

void RegulaFalsiFonkAl();
double regulaFalsi(char *infix, double a, double b, double tol, int maxIter);

void NewtonRaphsonFonkAl();
double newtonRaphson(char* infix, double x0, double tolerance);

float determinant(float **matrix, int n);
void adjoint(float **matrix, float **adj, int n);
int inverse(float **matrix, float **inverse, int n);
void MatrisTersiAl();

void gauss_eleme(float **matris, float *sonuclar, int n);
int GauusElemeYontemi();

void GaussSeidelFonkAl();
void gaussSeidel(double matris[MAX_SIZE][MAX_SIZE], double sonuclar[MAX_SIZE], int n, double x[MAX_SIZE], double hata);

double forwardDifference(double (*func)(char*, double), char* postfix, double x, double h);
double backwardDifference(double (*func)(char*, double), char* postfix, double x, double h);
double centralDifference(double (*func)(char*, double), char* postfix, double x, double h);
void SayisalTurevFonkAl();

double SimpsonKurali(char* postfix, double a, double b, int n);
void Simpson13FonkAl();

double trapezYontemi(double a, double b, int n, char* function);
void TrapezFonkAl();

void calculateCoefficients(double x[], double y[], int n, double coeff[]);
double interpolate(double x[], double y[], int n, double coeff[], double xi);
void GregoryNewtonEnt();
```

Kullanıcıdan fonksiyon almak ve üzerinde işlem yapabilmek için kullandığım fonksiyonlar:

```
void initializeStack(struct Stack *stack, int size) {
    stack->totalSize = size;
    stack->top = -1;
    stack->arrayStack = (char *)malloc(stack->totalSize * sizeof(char));
}

void destroyStack(struct Stack *stack) {
    free(stack->arrayStack);
}

int isEmpty(struct Stack *stack) {
    return stack->top == -1;
}

int isFull(struct Stack *stack) {
    return stack->top == stack->totalSize - 1;
}

char peek(struct Stack *stack) {
    if (isEmpty(stack))
        return -1;
    return stack->arrayStack[stack->top];
}

char pop(struct Stack *stack) {
    if (isEmpty(stack))
        return -1;
    return stack->arrayStack[stack->top--];
}

void push(char element, struct Stack *stack) {
    if (isFull(stack)) {
        printf("Stack is already Full.");
        return;
    }
    stack->arrayStack[++stack->top] = element;
}

int precedence(char op) {
    if (op == '+' || op == '-')
        return 1;
    if (op == '*' || op == '/')
        return 2;
    if (op == '^')
        return 3;
    return 0;
}

int isOperand(char element) {
    return (element >= 'A' && element <= 'Z') || (element >= 'a' && element <= 'z') || element
== 'e';
}

int isInteger(char element) {
    return (element >= '0' && element <= '9');
}

char* infixToPostfix(char *expressionArray, struct Stack *stack) {
    initializeStack(stack, strlen(expressionArray));
    int postfixSize = strlen(expressionArray) * 2;
    char *postfix = (char *)malloc(postfixSize * sizeof(char));
```

```
    int postfixIndex = 0;
    int currIndex = 0;

    while (expressionArray[currIndex] != '\0') {
        if (isOperand(expressionArray[currIndex]) || isInteger(expressionArray[currIndex])) {
            while (isOperand(expressionArray[currIndex]) ||
isInteger(expressionArray[currIndex])) {
                postfix[postfixIndex++] = expressionArray[currIndex++];
            }
            postfix[postfixIndex++] = ' ';
        } else if (expressionArray[currIndex] == '(') {
            push(expressionArray[currIndex], stack);
            currIndex++;
        } else if (expressionArray[currIndex] == ')') {
            while (peek(stack) != '(') {
                postfix[postfixIndex++] = pop(stack);
                postfix[postfixIndex++] = ' ';
            }
            pop(stack);
            currIndex++;
        } else {
            while (!isEmpty(stack) && precedence(peek(stack)) >=
precedence(expressionArray[currIndex])) {
                postfix[postfixIndex++] = pop(stack);
                postfix[postfixIndex++] = ' ';
            }
            push(expressionArray[currIndex], stack);
            currIndex++;
        }
    }

    while (!isEmpty(stack)) {
        postfix[postfixIndex++] = pop(stack);
        postfix[postfixIndex++] = ' ';
    }

    postfix[postfixIndex] = '\0';
    destroyStack(stack);
    return postfix;
}

typedef struct {
    int top;
    double stack[MAX_SIZE];
} stack2;

void displayStack(stack2* s) {
    int i = 0;
    while (s->top >= i) {
        printf("%.2lf ", s->stack[i]);
        i++;
    }
    printf("stack ended\n");
}

void initializeStack2(stack2 *s) {
    s->top = -1;
}

void push2(stack2 *s, double item) {
    if (s->top >= MAX_SIZE - 1) {
        printf("Stack Overflow\n");
        exit(EXIT_FAILURE);
```

```
    }
    s->top++;
    s->stack[s->top] = item;
}

double pop2(stack2 *s) {
    if (s->top < 0) {
        printf("Stack Underflow\n");
        exit(EXIT_FAILURE);
    }
    double item = s->stack[s->top];
    s->top--;
    return item;
}

int is_operator(char symbol) {
    return (symbol == '+' || symbol == '-' || symbol == '*' || symbol == '/' || symbol == '^');
}

double evaluate(char* expression, stack2 *s, double x) {
    double operand1, operand2, result;
    char *token = strtok(expression, " ");
    while (token != NULL) {
        if (isdigit(*token) || (*token == '-' && isdigit(*(token + 1)))) {
            push2(s, atof(token));
        } else if (*token == 'x') {
            push2(s, x);
        } else if (*token == 'e') {
            push2(s, 2.718);
        } else if (is_operator(*token)) {
            operand2 = pop2(s);
            operand1 = pop2(s);
            switch(*token) {
                case '+': result = operand1 + operand2; break;
                case '-': result = operand1 - operand2; break;
                case '*': result = operand1 * operand2; break;
                case '/':
                    if (operand2 == 0) {
                        printf("Division by zero error\n");
                        exit(1);
                    }
                    result = operand1 / operand2;
                    break;
                case '^': result = pow(operand1, operand2); break;
            }
            push2(s, result);
        }
        token = strtok(NULL, " ");
    }
    return pop2(s);
}

double evaluateExpression(char* postfix, double x) {
    stack2 s;
    initializeStack2(&s);
    char* postfix_copy = strdup(postfix);
    if (!postfix_copy) {
        printf("Memory allocation failed\n");
        exit(EXIT_FAILURE);
    }
    double result = evaluate(postfix_copy, &s, x);
    free(postfix_copy);
    return result; }
```

BİSECTİON YÖNTEMİ

Fonksiyonlarım:

```c
void BisectionFonkAl() {
    char expressionArray[40];
    struct Stack stack;
    double x;

    printf("Bir fonksiyon giriniz: ");
    scanf("%s", expressionArray);
    printf("Girilen fonksiyon: %s\n", expressionArray);

    char *postfix = infixToPostfix(expressionArray, &stack);
    printf("Postfix hali: %s\n", postfix);

    double a, b, tol;
    printf("Araligin basi: ");
    scanf("%lf", &a);
    printf("Araligin sonu: ");
    scanf("%lf", &b);
    printf("Durdurmak icin tolerans degeri: ");
    scanf("%lf", &tol);

    double result = bisection(evaluateExpression, a, b, tol, postfix);
    printf("Kok: %lf\n", result);

    free(postfix);
}
```

```c
double bisection(double (*f)(char*, double), double a, double b, double tol, char *postfix) {
    double c;
    int iter = 0;
    int itercarpim = 1;

    if (f(postfix, a) * f(postfix, b) >= 0) {
        printf("Belirtilen aralikta kok yok veya tek sayida kok var.\n");
        return -1;
    }

    while ((b - a)/itercarpim >= tol) {
        c = (a + b) / 2.0;
        double func_c = f(postfix, c);
        printf("%d. iterasyon sonucu: %lf\n",iter,func_c);

        if (func_c == 0.0) {
            printf("Kok bulundu: %lf\n", c);
            return c;
        } else if (func_c * f(postfix, a) < 0) {
            b = c;
        } else {
            a = c;
        }

        iter++;
        itercarpim *= 2;
    }

    printf("Iterasyon sayisi: %d\n", iter);
    printf("Kok tahmini: %lf\n", c);
    return c;
}
```

Terminalde çalıştırma örnekleri:

```
Islem yapmak icin yontemin numarasini girin: 1
Bir fonksiyon giriniz: x^3-7*x^2+14*x-6
Girilen fonksiyon: x^3-7*x^2+14*x-6
Postfix hali: x 3 ^ 7 x 2 ^ * - 14 x * + 6 -
Araligin basi: 0
Araligin sonu: 1
Durdurmak icin tolerans degeri: 0.01
0. iterasyon sonucu: -0.625000
1. iterasyon sonucu: 0.984375
2. iterasyon sonucu: 0.259766
3. iterasyon sonucu: -0.161865
Iterasyon sayisi: 4
Kok tahmini: 0.562500
Kok: 0.562500
```

```
Islem yapmak icin yontemin numarasini girin: 1
Bir fonksiyon giriniz: x^3-9
Girilen fonksiyon: x^3-9
Postfix hali: x 3 ^ 9 -
Araligin basi: 0
Araligin sonu: 8
Durdurmak icin tolerans degeri: 0.00001
0. iterasyon sonucu: 55.000000
1. iterasyon sonucu: -1.000000
2. iterasyon sonucu: 18.000000
3. iterasyon sonucu: 6.625000
4. iterasyon sonucu: 2.390625
5. iterasyon sonucu: 0.595703
6. iterasyon sonucu: -0.226318
7. iterasyon sonucu: 0.178558
8. iterasyon sonucu: -0.025402
9. iterasyon sonucu: 0.076196
Iterasyon sayisi: 10
Kok tahmini: 2.085938
Kok: 2.085938
```

REGULA-FALSİ YÖNTEMİ

Fonksiyonlarım:

```c
double regulaFalsi(char *infix, double a, double b, double tol, int maxIter) {
    int i;
    struct Stack stack;
    char* postfix = infixToPostfix(infix, &stack);

    double fa = evaluateExpression(postfix, a);
    double fb = evaluateExpression(postfix, b);
    if (fa * fb > 0) {
        printf("Fonksiyon a ve b noktalarinda ayni isaretlere sahip. Aralikta kok bulunamadi.\n");
        free(postfix);
        return NAN;
    }

    double c, fc;
    for (i = 0; i < maxIter; ++i) {
        c = b - (fb * (b - a)) / (fb - fa);
        fc = evaluateExpression(postfix, c);

        if (fabs(fc) < tol) {
            free(postfix);
            return c;
        }

        if (fc * fa < 0) {
            b = c;
            fb = fc;
        } else {
            a = c;
            fa = fc;
        }
    }

    free(postfix);
    return c;
}
```

```c
void RegulaFalsiFonkAl() {
    char expressionArray[40];
    struct Stack stack;
    double x;
    double a, b, tol;
    int maxIter;

    printf("Fonksiyonu giriniz: ");
    scanf("%s", expressionArray);
    printf("Girilen fonksiyon: %s\n", expressionArray);

    // Call the function
    char *postfix = infixToPostfix(expressionArray, &stack);
    printf("Postfix hali: %s\n", postfix);

    printf("Araligin uc noktalarini girin [a, b]: ");
    scanf("%lf %lf", &a, &b);

    printf("Hata payini girin: ");
    scanf("%lf", &tol);

    printf("Maximum iterasyon sayisini giriniz: ");
    scanf("%d", &maxIter);

    double root = regulaFalsi(expressionArray, a, b, tol, maxIter);
    if (!isnan(root)) {
        printf("Kok: %lf\n", root);
    } else {
        printf("Verilen aralikta kok bulunamadi.\n");
    }
}
```

Terminalde çalışma örnekleri:

```
Islem yapmak icin yontemin numarasini girin: 2
Fonksiyonu giriniz: x^3-2*x^2-5
Girilen fonksiyon: x^3-2*x^2-5
Postfix hali: x 3 ^ 2 x 2 ^ * - 5 -
Araligin uc noktalarini girin [a, b]: 2
3
Hata payini girin: 0.01
Maximum iterasyon sayisini giriniz: 100
Kok: 2.690140
```

```
Islem yapmak icin yontemin numarasini girin: 2
Fonksiyonu giriniz: x^3-2*x^2+3*x-5
Girilen fonksiyon: x^3-2*x^2+3*x-5
Postfix hali: x 3 ^ 2 x 2 ^ * - 3 x * + 5 -
Araligin uc noktalarini girin [a, b]: 1
2
Hata payini girin: 0.000001
Maximum iterasyon sayisini giriniz: 1000
Kok: 1.843734
```

NEWTON-RAPSHON YÖNTEMİ

Fonksiyonlarım:

```
double newtonRaphson(char* infix, double x0, double tolerance) {
    struct Stack stack;
    char* postfix = infixToPostfix(infix, &stack);
    printf("Fonksiyonu giriniz:: %s\n", postfix);
    double x = x0;
    double fx, dfx;
    double x1;

    while (1) {
        fx = evaluateExpression(postfix, x);

        double h = 1e-7;
        double fxh1 = evaluateExpression(postfix, x + h);
        double fxh2 = evaluateExpression(postfix, x - h);
        dfx = (fxh1 - fxh2) / (2 * h);

        x1 = x - fx / dfx;

        if (fabs(x1 - x) < tolerance) {
            free(postfix);
            return x1;
        }

        x = x1;
    }
}

void NewtonRaphsonFonkAl() {
    char expressionArray[40];
    double x0, tolerance;
    int maxIter;

    printf("Fonksiyonu girin: ");
    scanf("%s", expressionArray);
    printf("Girilen fonksiyon: %s\n", expressionArray);

    printf("Baslangic degerini girin: ");
    scanf("%lf", &x0);

    printf("Hata payini girin: ");
    scanf("%lf", &tolerance);

    double root = newtonRaphson(expressionArray, x0, tolerance);
    printf("Bulunan kok: %.7lf\n", root);
}
```

Terminalde çalışma örnekleri:

```
Islem yapmak icin yontemin numarasini girin: 3
Fonksiyonu girin: x^3-7*x^2+14*x-6
Girilen fonksiyon: x^3-7*x^2+14*x-6
Baslangic degerini girin: 0
Hata payini girin: 0.000001
Fonksiyonu giriniz:: x 3 ^ 7 x 2 ^ * - 14 x * + 6 -
Bulunan kok: 0.5857864
```

```
Islem yapmak icin yontemin numarasini girin: 3
Fonksiyonu girin: x^2-4*x+3
Girilen fonksiyon: x^2-4*x+3
Baslangic degerini girin: 0
Hata payini girin: 0.001
Fonksiyonu giriniz:: x 2 ^ 4 x * - 3 +
Bulunan kok: 1.0000000
```

MATRİS TERSİ ALMA YÖNTEMİ

```
float determinant(float **matrix, int n) {
    int i,x,j;
    float det = 0.0;
    if (n == 1) {
        return matrix[0][0];
    }

    float **submatrix = (float **)malloc((n-1) * sizeof(float *));
    for (i = 0; i < n-1; i++) {
        submatrix[i] = (float *)malloc((n-1) * sizeof(float));
    }

    for (x = 0; x < n; x++) {
        int subi = 0;
        for (i = 1; i < n; i++) {
            int subj = 0;
            for (j = 0; j < n; j++) {
                if (j != x) {
                    submatrix[subi][subj] = matrix[i][j];
                    subj++;
                }
            }
            subi++;
        }
        det += (x % 2 == 0 ? 1 : -1) * matrix[0][x] * determinant(submatrix, n-1);
    }

    for (i = 0; i < n-1; i++) {
        free(submatrix[i]);
    }
    free(submatrix);

    return det;
}

// Matrisin adjoint'ini (eşlenik matris) hesaplayan fonksiyon
void adjoint(float **matrix, float **adj, int n) {
    int i,j,x,y;
    if (n == 1) {
        adj[0][0] = 1;
        return;
    }

    int sign = 1;
    float **submatrix = (float **)malloc((n-1) * sizeof(float *));
    for (i = 0; i < n-1; i++) {
        submatrix[i] = (float *)malloc((n-1) * sizeof(float));
    }

    for (i = 0; i < n; i++) {
```

```
        for (j = 0; j < n; j++) {
            int subi = 0;
            for (x = 0; x < n; x++) {
                if (x != i) {
                    int subj = 0;
                    for (y = 0; y < n; y++) {
                        if (y != j) {
                            submatrix[subi][subj] = matrix[x][y];
                            subj++;
                        }
                    }
                    subi++;
                }
            }
            sign = ((i+j) % 2 == 0) ? 1 : -1;
            adj[j][i] = sign * determinant(submatrix, n-1);
        }
    }

    for (i = 0; i < n-1; i++) {
        free(submatrix[i]);
    }
    free(submatrix);
}

// Matrisin tersini hesaplayan fonksiyon
int inverse(float **matrix, float **inverse, int n) {
    int i,j;
    float det = determinant(matrix, n);
    if (det == 0) {
        printf("Matrisin tersi yok.\n");
        return 0;
    }

    float **adj = (float **)malloc(n * sizeof(float *));
    for (i = 0; i < n; i++) {
        adj[i] = (float *)malloc(n * sizeof(float));
    }

    adjoint(matrix, adj, n);

    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            inverse[i][j] = adj[i][j] / det;
        }
    }

    for (i = 0; i < n; i++) {
        free(adj[i]);
    }
    free(adj);
```

```
    return 1;
}

void MatrisTersiAl() {
    int n,i,j;
    printf("Matris boyutunu girin: ");
    scanf("%d", &n);

    float **matrix = (float **)malloc(n * sizeof(float *));
    float **inverseMatrix = (float **)malloc(n * sizeof(float *));
    for (i = 0; i < n; i++) {
        matrix[i] = (float *)malloc(n * sizeof(float));
        inverseMatrix[i] = (float *)malloc(n * sizeof(float));
    }

    printf("Matris elemanlarini girin:\n");
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            scanf("%f", &matrix[i][j]);
        }
    }

    if (inverse(matrix, inverseMatrix, n)) {
        printf("Matrisin tersi:\n");
        for (i = 0; i < n; i++) {
            for (j = 0; j < n; j++) {
                printf("%0.4f ", inverseMatrix[i][j]);
            }
            printf("\n");
        }
    }

    for (i = 0; i < n; i++) {
        free(matrix[i]);
        free(inverseMatrix[i]);
    }
    free(matrix);
    free(inverseMatrix);
}
```

Terminalde çalıştırma örnekleri:

```
Islem yapmak icin yontemin numarasini girin: 4
Matrisin boyutunu girin: 3
Matrisin elemanlarini girin:
2
0
-1
0
1
3
-2
0
4
Matrisin Tersi:
0.67 0.00 0.17
-1.00 1.00 -1.00
0.33 0.00 0.33
```

```
Islem yapmak icin yontemin numarasini girin: 4
Matrisin boyutunu girin: 2
Matrisin elemanlarini girin:
1
1
0
1
Matrisin Tersi:
1.00 -1.00
0.00 1.00
```

GAUSS ELEMİNASYON YÖNTEMİ

Fonksiyonlarım:

```c
void gauss_eleme(float **matris, float *sonuclar, int n) {
    int i, j, k;
    float oran, temp;

    // İleriye doğru eleme adımı
    for (i = 0; i < n - 1; i++) {
        for (j = i + 1; j < n; j++) {
            oran = matris[j][i] / matris[i][i];
            for (k = i; k < n; k++) {
                matris[j][k] -= oran * matris[i][k];
            }
            sonuclar[j] -= oran * sonuclar[i];
        }
    }

    // Geriye doğru eleme adımı
    for (i = n - 1; i >= 0; i--) {
        temp = 0;
        for (j = i + 1; j < n; j++) {
            temp += matris[i][j] * sonuclar[j];
        }
        sonuclar[i] = (sonuclar[i] - temp) / matris[i][i];
    }
}
```

```c
int GauusElemeYontemi() {
    int n, i, j;

    printf("Denklem sisteminin boyutunu girin: ");
    scanf("%d", &n);

    float **matris = (float **)malloc(n * sizeof(float *));
    float *sonuclar = (float *)malloc(n * sizeof(float));

    printf("Katsayi matrisinin elemanlarini girin:\n");
    for (i = 0; i < n; i++) {
        matris[i] = (float *)malloc(n * sizeof(float));
        for (j = 0; j < n; j++) {
            scanf("%f", &matris[i][j]);
        }
    }

    printf("Sonuclar vektorunu girin:\n");
    for (i = 0; i < n; i++) {
        scanf("%f", &sonuclar[i]);
    }

    gauss_eleme(matris, sonuclar, n);
    printf("Denklem sisteminin cozumleri:\n");
    for (i = 0; i < n; i++) {
        printf("x%d = %.2f\n", i + 1, sonuclar[i]);
    }

    for (i = 0; i < n; i++) {
        free(matris[i]);
    }
    free(matris);
    free(sonuclar);

    return 0;
}
```

Terminalde çalışma örnekleri:

```
Islem yapmak icin yontemin numarasini girin: 5
Denklem sisteminin boyutunu girin: 4
Katsayi matrisinin elemanlarini girin:
4
-2
-1
3
3
1
-2
1
2
3
5
-1
1
-1
3
4
Sonuclar vektorunu girin:
15
0
26
27
Denklem sisteminin cozumleri:
x1 = 3.00
x2 = -1.00
x3 = 5.00
x4 = 2.00
```

$$\begin{bmatrix} 4 & -2 & -1 & 3 \\ 3 & 1 & -2 & 1 \\ 2 & 3 & 5 & -1 \\ 1 & -1 & 3 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 15 \\ 0 \\ 26 \\ 27 \end{bmatrix}$$

```
Islem yapmak icin yontemin numarasini girin: 5
Denklem sisteminin boyutunu girin: 3
Katsayi matrisinin elemanlarini girin:
3.6
2.4
-1.8
4.2
-5.8
2.1
0.8
3.5
6.5
Sonuclar vektorunu girin:
6.3
7.5
3.7
Denklem sisteminin cozumleri:
x1 = 1.81
x2 = 0.12
x3 = 0.28
```

$$3{,}6\,x + 2{,}4\,y - 1{,}8\,z = 6{,}3$$
$$4{,}2\,x - 5{,}8\,y + 2{,}1\,z = 7{,}5$$
$$0{,}8\,x + 3{,}5\,y + 6{,}5\,z = 3{,}7$$

GAUSS SEİDEL YÖNTEMİ

Fonksiyonlarım:

```c
void gaussSeidel(double matris[MAX_SIZE][MAX_SIZE], double sonuclar[MAX_SIZE], int n, double x[MAX_SIZE], double hata) {
    int i,j;
    double yeni_x[MAX_SIZE];
    double max_hata;
    int iterasyon = 0;

    do {
        max_hata = 0.0;

        for (i = 0; i < n; ++i) {
            double sum = 0.0;

            for (j = 0; j < n; ++j) {
                if (j != i) {
                    sum += matris[i][j] * x[j];
                }
            }

            yeni_x[i] = (sonuclar[i] - sum) / matris[i][i];

            double current_hata = fabs((yeni_x[i] - x[i]));
            if (current_hata > max_hata) {
                max_hata = current_hata;
            }

            x[i] = yeni_x[i];
        }

        ++iterasyon;
    } while (max_hata > hata && iterasyon < 1000);

    printf("\n%d iterasyon sonra hata: %.6lf\n", iterasyon, max_hata);
}
```

```c
void GaussSeidelFonkAl() {
    int n,i,j,k;
    printf("Denklem sayisini girin: ");
    scanf("%d", &n);

    double matris[MAX_SIZE][MAX_SIZE];
    double sonuclar[MAX_SIZE];
    double x[MAX_SIZE];
    double hata;

    printf("Denklemleri girin (matris formunda):\n");
    for (i = 0; i < n; ++i) {
        for (j = 0; j < n; ++j) {
            printf("Matris[%d][%d]: ", i+1, j+1);
            scanf("%lf", &matris[i][j]);
        }
    }

    printf("Sonuclari girin:\n");
    for (i = 0; i < n; ++i) {
        printf("Sonuc[%d]: ", i+1);
        scanf("%lf", &sonuclar[i]);
    }

    for (i = 0; i < n; ++i) {
        double max = fabs(matris[i][i]);
        int max_index = i;
        for (j = i + 1; j < n; ++j) {
            if (fabs(matris[j][i]) > max) {
                max = fabs(matris[j][i]);
                max_index = j;
            }
        }

        if (max_index != i) {
            for (k = 0; k < n; ++k) {
                double temp = matris[i][k];
                matris[i][k] = matris[max_index][k];
                matris[max_index][k] = temp;
            }
            double temp = sonuclar[i];
            sonuclar[i] = sonuclar[max_index];
            sonuclar[max_index] = temp;
        }
    }

    printf("Diyagonal elemanlari maksimum yapildiktan sonra matris:\n");
    for (i = 0; i < n; ++i) {
        for (j = 0; j < n; ++j) {
            printf("%.2lf ", matris[i][j]);
        }
        printf(" | %.2lf\n", sonuclar[i]);
    }

    printf("Baslangic degerlerini girin:\n");
    for (i = 0; i < n; ++i) {
        printf("x[%d]: ", i+1);
        scanf("%lf", &x[i]);
    }

    printf("Hata payini girin: ");
    scanf("%lf", &hata);

    gaussSeidel(matris, sonuclar, n, x, hata);

    printf("\nGauss-Seidel ile bulunan kokler:\n");
    for (i = 0; i < n; ++i) {
        printf("x[%d] = %.6lf\n", i+1, x[i]);
    }
}
```

Terminalde çalışma örnekleri:

```
Islem yapmak icin yontemin numarasini girin: 6
Denklem sayisini girin: 3
Denklemleri girin (matris formunda):
Matris[1][1]: -1
Matris[1][2]: 4
Matris[1][3]: -3
Matris[2][1]: 3
Matris[2][2]: 1
Matris[2][3]: -2
Matris[3][1]: 1
Matris[3][2]: -1
Matris[3][3]: 4
Sonuclari girin:
Sonuc[1]: -8
Sonuc[2]: 9
Sonuc[3]: 1
Diyagonal elemanlari maksimum yapildiktan sonra matris:
3.00 1.00 -2.00  | 9.00
-1.00 4.00 -3.00  | -8.00
1.00 -1.00 4.00  | 1.00
Baslangic degerlerini girin:
x[1]: 0
x[2]: 0
x[3]: 0
Hata payini girin: 0.001

7 iterasyon sonra hata: 0.000529

Gauss-Seidel ile bulunan kokler:
x[1] = 3.000031
x[2] = -1.999840
x[3] = -0.999968
```

SAYISAL TÜREV (MERKEZİ, İLERİ VE GERİ FARKLAR OPSİYONLU)

Fonksiyonlarım:

```
double forwardDifference(double (*func)(char*, double), char* postfix, double x, double h) {
    double xh = x + h;
    double fxh = func(postfix, xh);
    double fx = func(postfix, x);
    return (fxh - fx) / h;
}

double backwardDifference(double (*func)(char*, double), char* postfix, double x, double h) {
    double xh = x - h;
    double fxh = func(postfix, xh);
    double fx = func(postfix, x);
    return (fx - fxh) / h;
}

double centralDifference(double (*func)(char*, double), char* postfix, double x, double h) {
    double xph = x + h;
    double xmh = x - h;
    double fxph = func(postfix, xph);
    double fxmh = func(postfix, xmh);
    return (fxph - fxmh) / (2 * h);
}

void SayisalTurevFonkAl() {
 char expressionArray[40];
    struct Stack stack;
    stack2 stack2;
    double x,h;

    printf("Fonksiyonu giriniz: ");
    scanf("%s", expressionArray);
    printf("Girilen fonksiyon: %s\n", expressionArray);

    char *postfix = infixToPostfix(expressionArray, &stack);
    printf("Postfix hali: %s\n", postfix);

    printf("x degerini girin: ");
    scanf("%lf", &x);
    printf("h degerini girin: ");
    scanf("%lf", &h);

    double forwardDiff = forwardDifference(evaluateExpression, postfix, x, h);
    double backwardDiff = backwardDifference(evaluateExpression, postfix, x, h);
    double centralDiff = centralDifference(evaluateExpression, postfix, x, h);

    printf("Ileri fark ile: %.10lf\n", forwardDiff);
    printf("Geri fark ile: %.10lf\n", backwardDiff);
    printf("Merkezi fark ile: %.10lf\n", centralDiff);

    free(postfix);
}
```

Terminalde çalışma örneği:

```
Islem yapmak icin yontemin numarasini girin: 7
Fonksiyonu giriniz: e^x
Girilen fonksiyon: e^x
Postfix hali: e x ^
x degerini girin: 1
h degerini girin: 0.1
Ileri fark ile: 2.8582341024
Geri fark ile: 2.5862639807
Merkezi fark ile: 2.7222490416
```

```
Islem yapmak icin yontemin numarasini girin: 7
Fonksiyonu giriniz: x^2
Girilen fonksiyon: x^2
Postfix hali: x 2 ^
x degerini girin: 2
h degerini girin: 0.1
Ileri fark ile: 4.1000000000
Geri fark ile: 3.9000000000
Merkezi fark ile: 4.0000000000
```

```
Islem yapmak icin yontemin numarasini girin: 7
Fonksiyonu giriniz: 2*x^2-3*x+4
Girilen fonksiyon: 2*x^2-3*x+4
Postfix hali: 2 x 2 ^ * 3 x * - 4 +
x degerini girin: 4
h degerini girin: 0.01
Ileri fark ile: 13.0200000000
Geri fark ile: 12.9800000000
Merkezi fark ile: 13.0000000000
```

SİMPSON 1/3 YÖNTEMİ

Fonksiyonlarım:

```c
double SimpsonKurali(char* postfix, double a, double b, int n) {
    int i;
    if (n % 2 != 0) {
        printf("n cift olmali\n");
        exit(EXIT_FAILURE);
    }

    double h = (b - a) / n;
    double sum = 0.0;
    double x;

    sum += evaluateExpression(postfix, a);
    sum += evaluateExpression(postfix, b);

    for (i = 1; i < n; i++) {
        x = a + i * h;
        if (i % 2 == 0) {
            sum += 2 * evaluateExpression(postfix, x);
        } else {
            sum += 4 * evaluateExpression(postfix, x);
        }
    }

    return (h / 3) * sum;
}

void Simpson13FonkAl() {
    char expressionArray[40];
    struct Stack stack;
    double a, b;
    int n;

    printf("Fonksiyonu giriniz: ");
    scanf("%s", expressionArray);
    printf("Girilen fonksiyon: %s\n", expressionArray);

    char *postfix = infixToPostfix(expressionArray, &stack);
    printf("Postfix hali: %s\n", postfix);

    printf("Alt siniri girin (a): ");
    scanf("%lf", &a);
    printf("Ust siniri girin (b): ");
    scanf("%lf", &b);
    printf("Aralik sayisini girin (n) (cift olmali): ");
    scanf("%d", &n);

    double result = SimpsonKurali(postfix, a, b, n);
    printf("Integrasyonun sonucu: %.4lf\n", result);

    free(postfix);
}
```

Terminalde çalıştırma örneği:

```
Islem yapmak icin yontemin numarasini girin: 8
Fonksiyonu giriniz: x^3
Girilen fonksiyon: x^3
Postfix hali: x 3 ^
Alt siniri girin (a): -1
Ust siniri girin (b): 2
Aralik sayisini girin (n) (cift olmali): 4
Integrasyonun sonucu: 3.7500
```

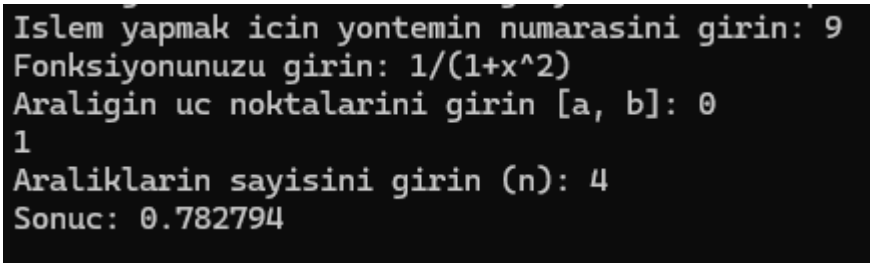
```
Islem yapmak icin yontemin numarasini girin: 8
Fonksiyonu giriniz: (1/5)+25*x-200*x^2+675*x^3-900*x^4+400*x^5
Girilen fonksiyon: (1/5)+25*x-200*x^2+675*x^3-900*x^4+400*x^5
Postfix hali: 1 5 / 25 x * + 200 x 2 ^ * - 675 x 3 ^ * + 900 x 4 ^ * - 400 x 5 ^ * +
Alt siniri girin (a): 0
Ust siniri girin (b): 0.8
Aralik sayisini girin (n) (cift olmali): 2
Integrasyonun sonucu: 1.3675
```

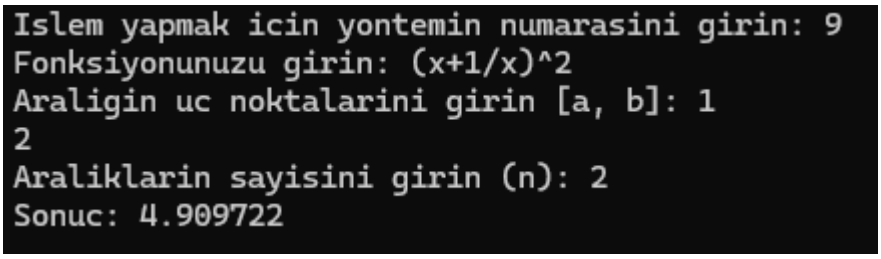
TRAPEZ YÖNTEMİ

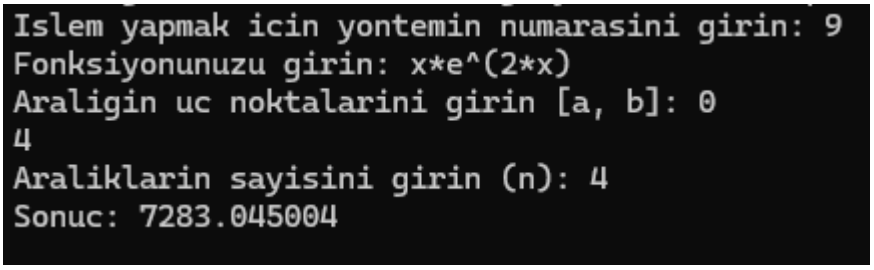
Fonksiyonlarım:

```c
double trapezYontemi(double a, double b, int n, char* function) {
    int i;
    double h = (b - a) / n;

    double sum = 0.0;

    for (i = 0; i <= n; i++) {
        double x = a + i * h;
        if (i == 0 || i == n) {
            sum += evaluateExpression(function, x);
        } else {
            sum += 2 * evaluateExpression(function, x);
        }
    }

    sum *= h / 2.0;

    return sum;
}

void TrapezFonkAl() {
    char function[MAX_SIZE];
    int n;
    double a, b;

    printf("Fonksiyonunuzu girin: ");
    scanf("%s", function);

    printf("Araligin uc noktalarini girin [a, b]: ");
    scanf("%lf %lf", &a, &b);

    printf("Araliklarin sayisini girin (n): ");
    scanf("%d", &n);

    struct Stack stack;
    char* postfix = infixToPostfix(function, &stack);

    double result = trapezYontemi(a, b, n, postfix);

    printf("Sonuc: %.6f\n", result);

    free(postfix);
}
```

Terminalde çalıştırma örneği:







DEĞİŞKEN DÖNÜŞÜMSÜZ GREGORY NEWTON ENTERPOLASYONU

Fonksiyonlarım:

```c
void calculateCoefficients(double x[], double y[], int n, double coeff[]) {
    int i, j;

    for (i = 0; i < n; i++) {
        coeff[i] = y[i];
    }

    for (i = 1; i < n; i++) {
        for (j = n - 1; j >= i; j--) {
            coeff[j] = (coeff[j] - coeff[j - 1]) / (x[j] - x[j - i]);
        }
    }
}

double interpolate(double x[], double y[], int n, double coeff[], double xi) {
    double result = coeff[n - 1];
    int i;

    for (i = n - 2; i >= 0; i--) {
        result = result * (xi - x[i]) + coeff[i];
    }
    return result;
}

void GregoryNewtonEnt() {
    int n,i;
    printf("Kac adet x ve y degeri gireceksiniz: ");
    scanf("%d", &n);

    double x[n], y[n];
    printf("x ve y degerlerini girin:\n");
    for (i = 0; i < n; i++) {
        printf("x[%d] = ", i);
        scanf("%lf", &x[i]);
        printf("y[%d] = ", i);
        scanf("%lf", &y[i]);
    }

    double coeff[n];
    calculateCoefficients(x, y, n, coeff);

    double xi;
    printf("Enterpolasyon yapilacak x degerini girin: ");
    scanf("%lf", &xi);

    double interpolatedValue = interpolate(x, y, n, coeff, xi);

    printf("Enterpolasyon sonucu: %.2f\n", interpolatedValue);
}
```

Terminalde çalışma örneği:

```
Islem yapmak icin yontemin numarasini girin: 10
Kac adet x ve y degeri gireceksiniz: 4
x ve y degerlerini girin:
x[0] = 1
y[0] = 1
x[1] = 2
y[1] = 8
x[2] = 3
y[2] = 27
x[3] = 4
y[3] = 64
Enterpolasyon yapilacak x degerini girin: 2.2
Enterpolasyon sonucu: 10.65
```