Numerical Methods I – Project 2 Report

Problem

Write a computer program to implement Simpson's and trapezoidal rules for computing the integral $\int_a^b f(x)dx$, where

$$f(x) = b_0 + \sum_{k=1}^{n} (a_k \sin(kx) + b_k \cos(kx)).$$

Compare the results.

Report Content:

- Programming Fourier Series Form
- Programming Simpson's Approach
- Programming Trapezoidal Rule's Approach
- Testing
- Conclusion

- **Programming Fourier Series Form:**

A helper function is programmed, which evaluates the following expression:

$$\mathrm{f(x)} = b_0 + \sum_{k=1}^{n} a_k \sin kx + b_k \cos kx$$

Given the sequences a_k, b_k for $k = 0$ to n (a_0 is arbitrary), n, and x, the function evaluates the expression, see below the used function (used in both Simpson's and Trap scripts).

```
function answer=our_function(aVector,bVector,n,x)
    answer=0;
    for k=0:n
        answer=answer+aVector(k+1)*sin(k*x)+bVector(k+1)*cos(k*x);
    end
end
```

- **Programming Simpson's Approach:**

The script is composed of the following steps:

1. Initializing the variable h which holds the partitions width.
2. Initializing the variable fx, holds our function's value on the points of the partitions $a + ih$.
3. Looping to implement Simpson's rule, the image shown below.

$$= \frac{h}{3}\left[f(x_0) + 2\sum_{j=1}^{n/2-1} f(x_{2j}) + 4\sum_{j=1}^{n/2} f(x_{2j-1}) + f(x_n)\right],$$

where $x_j = a + jh$ for $j = 0, 1, \ldots, n-1, n$ with $h = (b-a)/n$; in particula

The code can be found below:

```
function integral=Simpson(a,b,N,aVector,bVector,n)
    % a,b are limits of integration.
    % N is the number of partitions for Simposon(must be positive even)
    % aVector and bVector are ak and bk (fourier coefficients: a_0 to a_n
    % n is the fourier summation upper limit


    % initialize answer
    integral=0;

    % check if N is positive even, exit if not
    if (mod(N,2)~=0) || (N<=0)
        fprintf("N must be postive and even!");
        exit
    end

    % partitions width
    h=(b-a)/N;
    % initialize the vector that will hold our function values
    fx=zeros(1,N+1);
    % calculate the values of the function at each point, store in the
    % vector fx
    for i=0:N
        % evaluate at a+ih. The our_function is defined below
        fx(i+1)=our_function(aVector,bVector,n,a+i*h);
    end

    % apply Simpson's rule, each summation term is done alone for
    % readability.
    for i=1:(N/2-1)
        integral=integral+2*fx(2*i+1);
    end

    for i=1:(N/2)
        integral=integral+4*fx(2*i);
    end

    integral=integral+fx(1)+fx(N+1);
    integral=integral*h/3;
    % end of simpson
end
```

- **Programming Trapezoidal Rule's Approach:**

The script is composed of the following steps:

4. Initializing the variable h which holds the partitions width.
5. Initializing the variable fx, holds our function's value on the points of the partitions $a + ih$.
6. Looping to implement Trapezoidal rule, the image shown below.

$$= \Delta x \left(\sum_{k=1}^{N-1} f(x_k) + \frac{f(x_N) + f(x_0)}{2} \right)$$

The code can be found below:

```
function integral=Trap(a,b,N,aVector,bVector,n)
    % a,b are limits of integration.
    % N is the number of partitions for Trapezoidal rule
    % aVector and bVector are ak and bk (fourier coefficients: a_0 to a_n
    % n is the fourier summation upper limit


    % initialize answer
    integral=0;



    % partitions width
    h=(b-a)/N;
    % initialize the vector that will hold our function values
    fx=zeros(1,N+1);
    % calculate the values of the function at each point, store in the
    % vector fx
    for i=0:N
        % evaluate at a+ih. The our_function is defined below
        fx(i+1)=our_function(aVector,bVector,n,a+i*h);
    end

    % apply Trapezoidal rule
    for i=1:N-1
        integral=integral+fx(i+1);
    end

    integral=integral+(fx(N+1)+fx(1))/2;
    integral=integral*h;
    % end
end
```

- **Testing:**

A test for the function $f(x) = x$ is performed, integrating from 0 to 1, with number of partitions N=26, and Fourier Series of order n=30.

As can be concluded by the answers of the scripts, comparing them to the expected answer of 0.5, both approaches gave accurate reliable results.

The following is the test script:

```
% this is a test for the fourier series of f(x)=x
% an=(2*(-1)^n) /n for n>=1. bn is zero for all n

% we will take fourier series of order n=30
aVector=zeros(31,1);
bVector=zeros(31,1);
% calculate ak
for k=2:31
    % these are ak. remember, indices are shifted bcs vectors in matlab
    % start with index 1, not 0.
    aVector(k)=(2*(-1)^k)/(k-1);
end

% call simpson and trapezoidal rules, for N=26,
% limits of integration 0 to 1. answer is expected to be 0.5

Trap(0,1,26,aVector,bVector,30)
Simpson(0,1,26,aVector,bVector,30)
```

And the following is the result of running the script:

```
COMMAND WINDOW

>> Test

ans =

    0.4997


ans =

    0.4997

>>
```

- **Conclusion**

I would like to say that both the Simpson's and Trapezoidal approaches were efficient and gave approximately the same result for the same number of partitions.