HW 3 Key

Table of Contents

Due to the nature of programming, your code may vary in style and function but the output should be the same.

Problem 4.2

```
clc; clear all; close all; format compact;

% a) 89
fprintf('Part a: %i\n',bin2dec('1011001'))

% b) 0.6875
fprintf('Part b: %.5f\n',2^-2 + 2^-4 + 2^-5)

% c) 6.28125
fprintf('Part c: %.5f\n',2^2+2^1+2^-2+2^-5)
```

```
Part a: 89
Part b: 0.34375
Part c: 6.28125
```

Problem 4.4

```
clc; clear all; format compact; close all;

%Initiating E
E = 1;

%Looping to find where 1 + E = 1
while 1
    if 1 + E/2 > 1
        E = E/2;
    else
        break
    end
end

%Printing results
fprintf('Epsilon = %E\n',E)
```

```
Epsilon = 2.220446E-16
```

Problem 4.11

```
clc;clear all;format compact; close all;

x = pi/3; %Number to take the cos of
ErS = 1; %Stopping error (percent)
i = 0; %Starting value for terms
CosEst = 0; %Initializing estimate
fprintf('Terms\t Estimate\t True Err\t Approx Err\n')
while 1
    OldEst = CosEst;

    CosEst = CosEst + (-1)^i * x^(2*i) / factorial(2*i);

    ErT = abs((cos(x)-CosEst)/cos(x))*100; %True error
    ErR = abs((CosEst-OldEst)/CosEst)*100; %Relative error

    i = i + 1; %Iterating for next terms

    %Displaying results
    fprintf(' %3i\t %6.6f    %6.2f%%\t %6.4f%%\n',i,CosEst,ErT,ErR)

    if ErR <= ErS %Break condition
        break
    end
end
```

```
Terms  Estimate  True Err  Approx Err
   1  1.000000    100.00%  100.0000%
   2  0.451689      9.66%  121.3914%
   3  0.501796      0.36%  9.9856%
   4  0.499965      0.01%  0.3664%
```

Problem 19.2

```
clc;clear all;format compact;close all;

func = @(x) 1-exp(-x); %Declaring function

% a) Analytical
Tru = 3 + exp(-4); %Analytical solution
Est = Tru;
fprintf('Part a:  %.7f (%.5f%% error)\n',Est,abs((Tru-Est)/Tru*100))

% b) Single trapezoid
Est = 4*(func(0)+func(4))/2;
fprintf('Part b:  %.7f (%.5f%% error)\n',Est,abs((Tru-Est)/Tru*100))

% c2) Two trapezoids
Est = 2*(func(0)+func(2))/2+2*(func(2)+func(4))/2;
fprintf('Part c1: %.7f (%.5f%% error)\n',Est,abs((Tru-Est)/Tru*100))
```

```
% c4) Four trapezoids
Est = 1*(func(0)+func(1))/2 + ...
      1*(func(1)+func(2))/2 + ...
      1*(func(2)+func(3))/2 + ...
      1*(func(3)+func(4))/2;
fprintf('Part c2: %.7f (%.5f%% error)\n',Est,abs((Tru-Est)/Tru*100))

% d) Simpson's 1/3 rule
Est = (2/3)*(func(0)+4*func(2)+func(4));
fprintf('Part d:  %.7f (%.5f%% error)\n',Est,abs((Tru-Est)/Tru*100))

% e) Composite Simpson's 1/3 rule

%Less efficient, easier to understand version
Est = 0;
Est = 1/3*(func(0)+4*func(1)+func(2))+1/3*(func(2)+4*func(3)+func(4));
fprintf('Part e:  %.7f (%.5f%% error)\n',Est,abs((Tru-Est)/Tru*100))

% f) Simpson's 3/8 rule
Est = 3*(4/3)/8 * (func(0)+3*func(4/3)+3*func(8/3)+func(4));
fprintf('Part f:  %.7f (%.5f%% error)\n',Est,abs((Tru-Est)/Tru*100))

% g) Composite Simpson's rules (1/3, 3/8)
h = 4/5;Est = h/3*(func(0)+4*func(h)+func(2*h)) ... %1/3 part

 +3*h/8*(func(2*h)+3*func(3*h)+3*func(4*h)+func(5*h)); %3/8 part
fprintf('Part g:  %.7f (%.5f%% error)\n',Est,abs((Tru-Est)/Tru*100))
```

```
Part a:  3.0183156 (0.00000% error)
Part b:  1.9633687 (34.95151% error)
Part c1: 2.7110138 (10.18124% error)
Part c2: 2.9378404 (2.66623% error)
Part d:  2.9602288 (1.92448% error)
Part e:  3.0134493 (0.16123% error)
Part f:  2.9912213 (0.89766% error)
Part g:  3.0158142 (0.08287% error)
```

Problem 19.12

```
clc;clear all;format compact; close all;

func = @(x) 5 + 0.25*x.^2;

% a) Analytical solution
Tru = 5*11+(0.25/3)*11^3;
fprintf('Part a: %.4f\n',Tru)

% b) Trapezoids
Est = 0;
for i = 1:11
    Est = Est + (func(i-1)+func(i))/2;
end
fprintf('Part b: %.4f\n',Est)
```

```
% c) Simpson's rule
Est = 0;
for i = 1:4
    Est = Est + (1/3)*(func(2*i-2)+4*func(2*i-1)+func(2*i)); %1/3 rule
end
Est = Est + (3/8)*(func(8)+3*func(9)+3*func(10)+func(11)); %3/8 rule
fprintf('Part c: %.4f\n',Est)
```

```
Part a: 165.9167
Part b: 166.3750
Part c: 165.9167
```

Problem 19.16

```
clc;clear all;format compact;close all;

%Declaring arrays
t = [0 10 20 30 35 40 45 50];
Q = [4 4.8 5.2 5 4.6 4.3 4.3 5];
c = [10 35 55 52 40 37 32 34];

IntSum = 0; %Starting integral at 0
for i = 1:length(t)-1
    %Using trapezoidal rule (note the varying increments!)
    IntSum = IntSum + (t(i+1)-t(i)) * (Q(i)*c(i)+Q(i+1)*c(i+1))/2;
end

fprintf('The integral is approximately %.2f\n',IntSum)
```

```
The integral is approximately 9518.50
```

Extra Credit

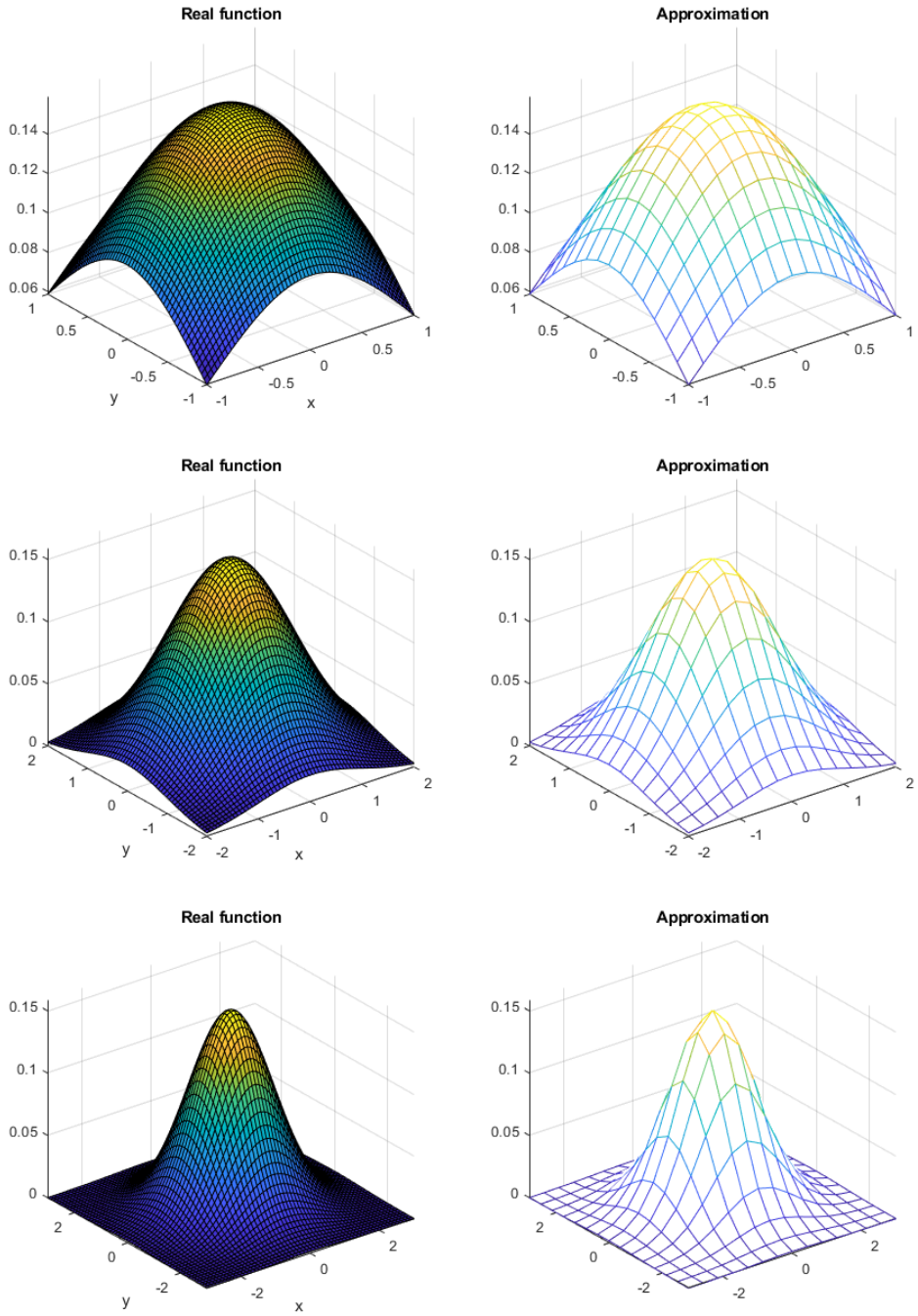
```
clc; clear all;format compact;close all;

%Declaring function, number of points in the X and Y directions
func = @(x,y) (1/(2*pi))*exp(-x.^2/2-y.^2/2);
Xpts = 15;
Ypts = 15;


fprintf('I\t Exact\t Approx\t Error\n')
%I(1)
IApprox = trap3D(func,-1,1,-1,1,Xpts,Ypts);
IExact = erf(1/sqrt(2))^2;
IErr = abs((IExact-IApprox)/IExact)*100;
fprintf('1\t %6.4f\t %6.4f\t %.2f%%\n',IExact,IApprox,IErr)
%I(2)
IApprox = trap3D(func,-2,2,-2,2,Xpts,Ypts);
IExact = erf(2/sqrt(2))^2;
IErr = abs((IExact-IApprox)/IExact)*100;
fprintf('2\t %6.4f\t %6.4f\t %.2f%%\n',IExact,IApprox,IErr)
```

```
%I(3)
IApprox = trap3D(func,-3,3,-3,3,Xpts,Ypts);
IExact = erf(3/sqrt(2))^2;
IErr = abs((IExact-IApprox)/IExact)*100;
fprintf('3\t %6.4f\t %6.4f\t %.2f%%\n',IExact,IApprox,IErr)
```

```
I  Exact  Approx  Error
1  0.4661  0.4649  0.24%
2  0.9111  0.9083  0.31%
3  0.9946  0.9938  0.08%
```



Published with MATLAB® R2018b