CHAPTER 06 EXCERCISE SOLUTIONS (ALL IN ONE)

I've created a c file *integral.c* where I've written the functions of integrations methods (Trapezoidal,SImpson's 1/3 and Simpson's 3/8), root finding methods (Bisection and Secant) and a factorial function.

Putting `#include"integral.c"` in the program we wished to use these functions.

In [ ]:
```c
// integral.c
//Function to perform factorial
double factorial(int n)
{
  int i;
  double fact=1;
  for(i=n;i>=1;i--)
  {
    fact=fact*i;
  }
  return fact;
}
/************************************************/
/***************TRAPEZOIDAL METHOD***************/
/************ONE VARIABLE EXPRESSION************/
/************************************************/
double trapezoidal(double func(double x),double a,double b,int n)
{
  int i;
  double x,h,integral,sum=0;
  h=fabs(b-a)/n;
  // to calculate the summation term
  for(i=1;i<n;++i)
  {
    x=a+i*h;
    sum=sum+func(x);
  }
  integral=(h/2)*(func(a)+func(b)+2*sum);
  return integral;
}
/************************************************/
/***************TRAPEZOIDAL METHOD***************/
/************TWO VARIABLE EXPRESSION************/
/************************************************/
double trap(double f(double x,double var),double var,double a,double b,double n)
{
  int i;
  double x,h,integral,sum=0;
  h=fabs(b-a)/n;
  // to calculate the summation term
  for(i=1;i<n;++i)
  {
    x=a+i*h;
    sum=sum+f(x,var);
  }
  integral=(h/2)*(f(a,var)+f(x,var)+2*sum);
  return integral;
```

```
}
/************************************************/
/***************SIMPSON'S 1/3 METHOD**************/
/************ONE VARIABLE EXPRESSION************/
/************************************************/
double simpson13(double func(double x),double a,double b,double n)
{
  double x,h,integral,sum=0;
  int i;
  h=fabs(b-a)/n;
  // to calculate the summation term
  for(i=1;i<n;++i)
  {
    x=a+i*h;
    if(i%2==0){
      sum=sum+2*func(x);
    }
    else{
      sum=sum+4*func(x);
    }
  }
  integral=(h/3)*(func(a)+func(b)+sum);
  return integral;
}
/************************************************/
/***************SIMPSON'S 3/8 METHOD**************/
/************ONE VARIABLE EXPRESSION************/
/************************************************/
double simpson38(double func(double x),double a,double b,double n)
{
  double x,h,integral,sum=0;
  int i;
  h=fabs(b-a)/n;
  // to calculate the summation term
  for(i=1;i<n;++i)
  {
    x=a+i*h;
    if(i%3==0){
      sum=sum+2*func(x);
    }
    else{
      sum=sum+3*func(x);
    }
  }
  integral=(3*h/8)*(func(a)+func(b)+sum);
  return integral;
}

/************************************************/
/***************ROOT FINDING**************/
/*************BISECTION METHOD*************/
/************************************************/

double bisection(double f(double x),float a,float b)
{
  double x,xm,xl,xr,accuracy=0.00001,xinc=0.5,z;
  for (x=a;x<=b;x+=xinc)
    {
      if (f(x)*f(x+xinc)<=0)
      {
        xl=x;
```

```
        xr=x+xinc;
        do
        {
          xm=(xl+xr)/2.0;
          if (f(xm)*f(xl)>=0)
          {
            xl=xm;
          }
          if (f(xm)*f(xl)<=0)
          {
            xr=xm;
          }
          z=fabs((xl-xr)/(xl+xr));
          //printf("xm=%f\tf(xm)=%f\tz=%f\taccuracy=%f\n",xm,J0(xm),z,accuracy);
        }
        while(z>accuracy);
        printf("\nroot=%f\tf(xm)=%f\tz=%f\taccuracy=%f\n",xm,f(xm),z,accuracy);
      }
    }
}
/************************************************/
/***************ROOT FINDING*****************/
/***************SECANT METHOD****************/
/************************************************/
double sec(double f(double x), double x1, double x2)
{
    int i=1;
    double x3;
    //printf("iter\tx1\t\tx2\t\tx3\t\tf(x3)\n");
    do{
        x3=(x1*f(x2)-x2*f(x1))/(f(x2)-f(x1));
        //printf("%d\t%lf\t%lf\t%lf\t%lf\n",i,x1,x2,x3,f(x3));
        x1=x2;
        x2=x3;
        i++;
    }while(fabs(f(x3))>0.00001);
    return x3;
}
double secant(double f(double x), double a, double b)
{
    double x,xb; // x starting point a, xb tending from a to b
    for(x=a;x<=b;x=x+0.01)
    {
        xb=x+0.01;
        if(f(x)*f(xb)<=0){
            printf("\nIn the interval: %.3lf and %.3lf\n",x,xb);
            double root=sec(f,x,xb);
            printf("The root is: %.4lf\n",root);
        }
    }
}
```

PROBLEM 1:

In [ ]:

```
// problem 1:
#include<stdio.h>
#include<math.h>
#include"integral.c"

// defining the function to evaluate
```

```
double f(double x){
    return atan(x)/(x*x);
}

int main()
{
    int n=2;    // starting with two interval
    double a=5,b=10,integral,answer;

    //using simpsons 1/3 until they converge to the accuracy
    do{
        integral=answer;
        n=n+2;  //n must be even
        answer=simpson13(f,a,b,n);
    }while(fabs(answer-integral)>=0.00001);

    printf("\nThe integral using Simpson's Rule is: %lf\n",answer);
    // using trapezoidal until they converge to the accuracy
        n=2;
    do{
        integral=answer;
        n++;
        answer=trapezoidal(f,a,b,n);
    }while(fabs(answer-integral)>=0.00001);
    printf("The integral using Trapezoidal Rule is: %lf",answer);
}
```

OUTPUT:

The integral using Simpson's Rule is: 0.142205

The integral using Trapezoidal Rule is: 0.142294

PROBLEM 2:

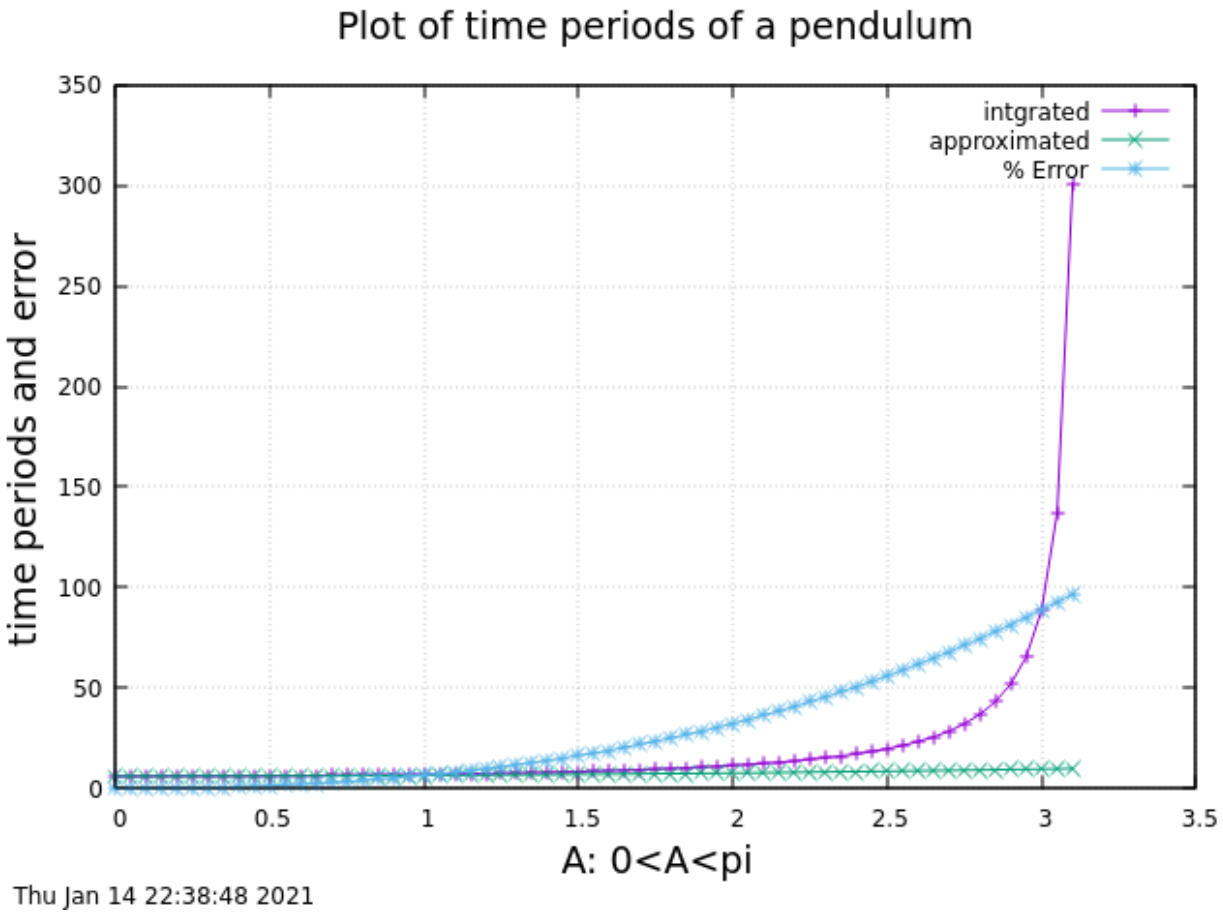
In [ ]:

```
// problem 2:
#include<stdio.h>
#include<math.h>
#include"integral.c"
#define pi 3.1415927

// function to be intgrated fxa(x,A)
double fxa(double x,double A){
    return 1/(1-sin(A/2)*sin(A/2)*sin(x)*sin(x));
}
// function for approx time-period T1(A)
double T1(double A){
    return 2*pi*(1+pow(A/4,2));
}
// function for whole time-period T(func,A,n)
double T(double f(double x,double A),double A,int n){
    return 4*trap(fxa,A,0,pi/2,n);
}
// main function to do our job
int main()
{
    FILE*fp=NULL;
    fp=fopen("prob2.txt","w");

    double A,t,t1,error;
    // getting values for A range
```

```
    for(A=0;A<=pi;A+=0.05)
    {
        t=T(fxa,A,300);   // time-period n=300
        t1=T1(A);   // approx time-period
        error=(t-t1)/t*100; // % error in both
        fprintf(fp,"%lf\t%lf\t%lf\t%lf\n",A,t,t1,error);
    }
}
```

OUTPUT

Program generated a text file "prob2.txt" and the plot of this data file is below:



PROBLEM 3:

In [ ]:

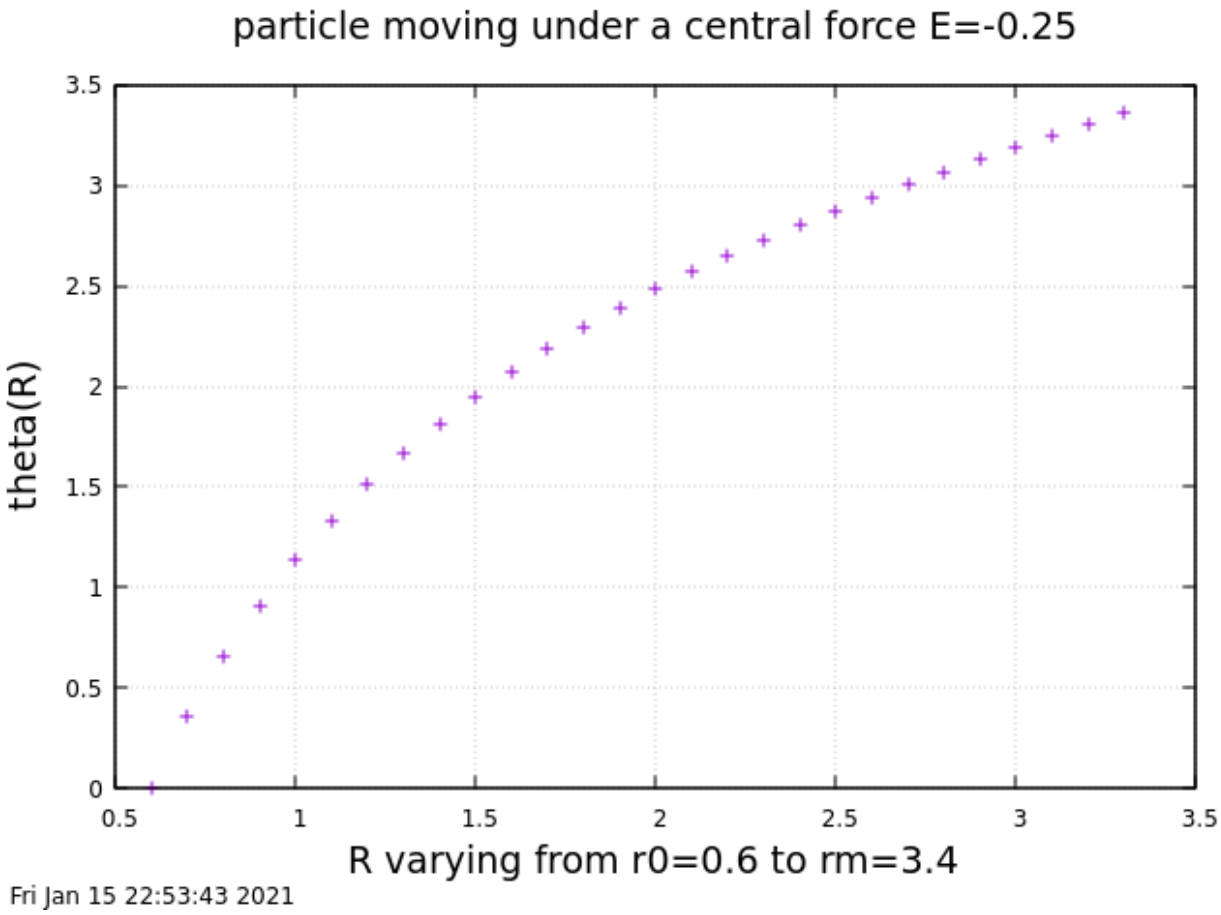
```
// problem 3
#include<stdio.h>
#include<math.h>
// function to be intgrated fre(r,E)
double fre(double r,double E)
{
    return (1/(r*r*sqrt(2*E+2/r+1/(r*r))));
}
// gauss quadrature function to evaluate integration
double gauss(double f(double r,double E),double r,double E,double a, double b)
{
    double x1,x2;
    x1=((b-a)/2.0)*(1/1.73)+((b+a)/2);
    x2=((b-a)/2.0)*(-1/1.73)+((b+a)/2);
    return (b-a)/a*(f(x1,E)+f(x2,E));
}
```

```c
int main()
{
    FILE*fp=NULL;
    fp=fopen("prob3.txt","w");

    double E,r,r0,rm;
    printf("Enter the value of E:");
    scanf("%lf",&E);
    printf("Enter the lower and upper limit r0 & rm:");
    scanf("%lf%lf",&r0,&rm);

    // varaying R from r0 to rm
    for (r=r0;r<=rm;r=r+0.1)
    {
        fprintf(fp,"%lf\t%lf\n",r,gauss(fre,r,E,r0,r));
    }
}
```

OUTPUT

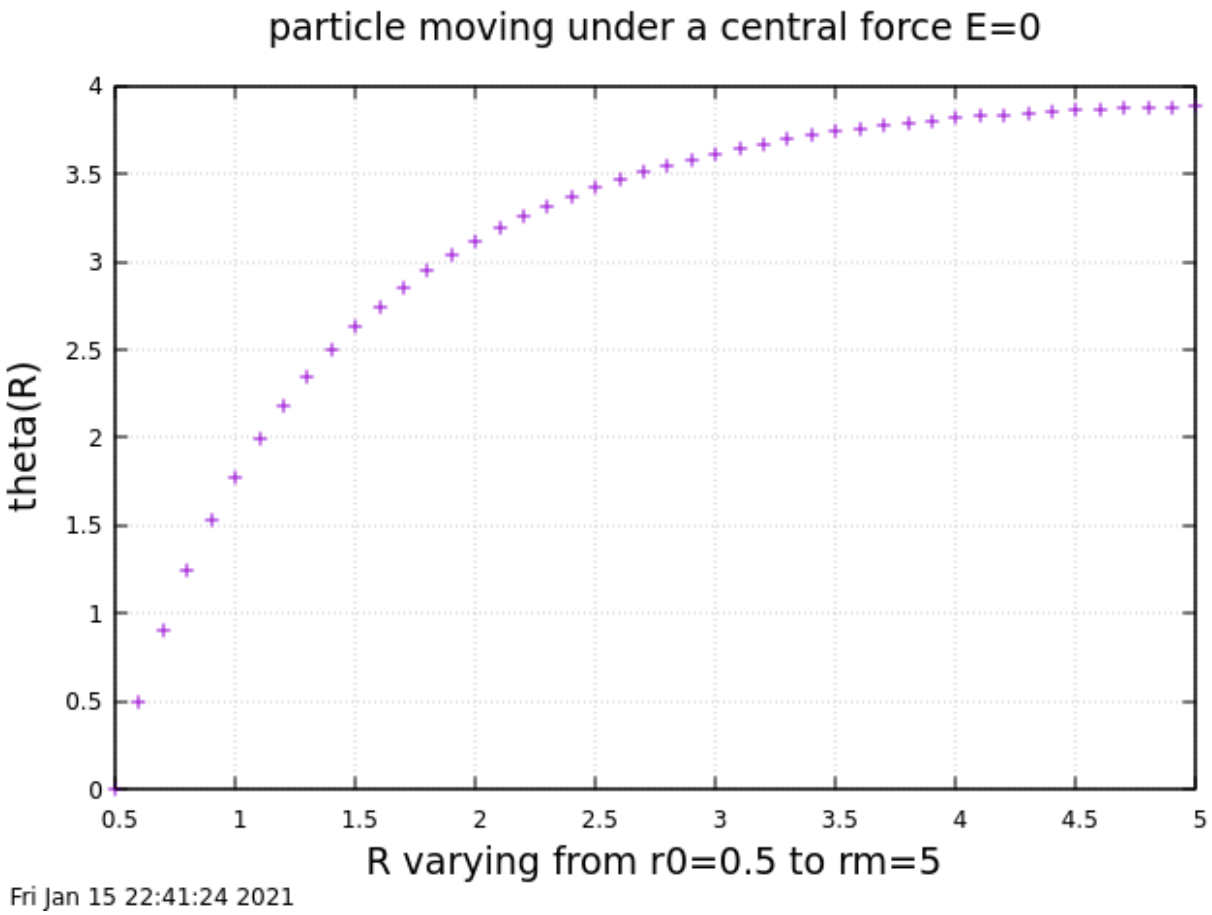
Enter the value of E:-0.25

Enter the lower and upper limit r0 & rm:0.6 3.4



Enter the value of E:0

Enter the lower and upper limit r0 & rm:0.5 5



PROBLEM 4:

In [ ]:
```c
// problem 4:
#include<stdio.h>
#include<math.h>
#include"integral.c"
#define pi 3.1415927

// function to be intgrated ftx(t,x)
double ftx(double t,double x){
    return cos(pow(x,1.5)*cos(t))*pow(sin(t),3);
}
// function which root is needed Fx(x)
double Fx(double x){
    return trap(ftx,x,0,pi,500);        // n=500
}
// main program to do neccessary job
int main()
{
        float a=0,b=5; // range of roots
        printf("USING BISECTION METHOD\n");
        bisection(Fx,a,b);
        printf("\nUSING SECANT METHOD\n");
        secant(Fx,a,b);
}
```

OUTPUT

USING BISECTION METHOD

root=2.722992 f(xm)=0.000013 z=0.000006 accuracy=0.000010
root=3.907898 f(xm)=0.000003 z=0.000008 accuracy=0.000010
root=4.917297 f(xm)=0.000001 z=0.000006 accuracy=0.000010

USING SECANT METHOD

In the interval: 2.720 and 2.730
The root is: 2.7230

In the interval: 3.900 and 3.910
The root is: 3.9079

In the interval: 4.910 and 4.920
The root is: 4.9173

HENCE THE SMALLEST ROOT IS:2.722992 (bisection)

PROBLEM 5:

In [ ]:
```c
// problem 5:
#include<stdio.h>
#include<math.h>
#include"integral.c"
#define pi 3.1415927

// function to be intgrated fxz(x,z)
double fxz(double x,double z){
    return cos(z*cos(x));
}
// function which root is needed J0(z)
double J0(double z){
    return 1/(2*pi)*trap(fxz,z,0,2*pi,500);     // n=500
}
// main program to do neccessary job
int main()
{
  float a,b;
  a=0,b=12;     // range of roots
  secant(J0,a,b);
}
```

OUTPUT

In the interval: 2.400 and 2.410
The root is: 2.4048

In the interval: 5.520 and 5.530
The root is: 5.5201

In the interval: 8.650 and 8.660
The root is: 8.6537

In the interval: 11.790 and 11.800
The root is: 11.7915

PROBLEM 6:

In [ ]:

```c
// problem 6:
#include<stdio.h>
#include<math.h>
#include"integral.c"
#define pi 3.1415927

double f(double theta,double z){
    return cos(z*cos(theta))*pow(sin(theta),5);
}
double J2(double z){
    return pow(z,2)/(pow(2,3)*factorial(2))*trap(f,z,0,pi,1000);
}

int main()
{
  float a=0,b=10;       // range of roots
  printf("USING BISECTION METHOD\n");
  bisection(J2,a,b);
  printf("\nUSING SECANT METHOD\n");
  secant(J2,a,b);
}
```

OUTPUT

USING BISECTION METHOD

root=0.499992 f(xm)=0.016371 z=0.000008 accuracy=0.000010
root=5.763489 f(xm)=-0.000005 z=0.000005 accuracy=0.000010
root=9.095093 f(xm)=0.000009 z=0.000007 accuracy=0.000010

USING SECANT METHOD

In the interval: 0.000 and 0.010
The root is: 0.0000

In the interval: 5.760 and 5.770
The root is: 5.7635

In the interval: 9.090 and 9.100
The root is: 9.0950