Ass5

May 31, 2019

1 Q1

Q1.1

Below is the formula for **a composite trapezoidal rule** $T_n(u)$ for $I = \int_b^a u(x)dx$ which requires n function evaluations at equidistant quadrature points and where the first and the last quadrature points coincide with the integration bounds a and b, respectively.

$$T(u) = \sum_{k=1}^{n} q(f, x_{k-1}, q_k) = h\left(\frac{u(x_0)}{2} + \sum_{k=1}^{n-1} u(x_k) + \frac{u(x_n)}{2}\right)$$

where $x_k = a + kh$

Q1.2

From the problem For v(x) with $x \in [0, 1]$, we do the transformation of variables $x = \alpha\xi + \beta$ such that

$\alpha + \beta = 1$

$-\alpha + \beta = 0$

Therefore we can solve that $\alpha = 1/2, \beta = 1/2$ giving the transformation

$$x = \frac{1}{2}\xi + \frac{1}{2}$$

Then we make transformation for integral: $dx = \alpha d\xi$

$$I = \int_0^1 v(x)dx = \int \alpha v(\alpha\xi + \beta)d\xi = \int_{-1}^1 u(\xi)d\xi$$

where

$$u(\xi) = \alpha v(\alpha\xi + \beta) = \frac{1}{2}v(\frac{1}{2}\xi + \frac{1}{2})$$

Q1.3

The transformed function introduced from previous question is

$$u(\xi) = \alpha v(\alpha\xi + \beta) = 1/2v(1/2\xi + 1/2) = \frac{1}{2}\frac{\xi+1}{2}\frac{\xi}{2}\frac{\xi-1}{2} = \frac{\xi(\xi^2-1)}{16}$$

We can see that u(ξ) is an odd function of ξ, therefore

$$\int_{-1}^1 u(\xi)d\xi = 0$$

With 2 quadrature points, we calculate the integral values of

midpoint rule: $Q(u) = (1-(-1))u(\frac{1-1}{2}) = 2u(0) = 0$
trapezoidal rule: $T(u) = (1-(-1))\frac{u(1)+u(-1)}{2} = 0$
Gaussian rule: As n=1, from lecture note: $Q(u) = u(-\frac{1}{\sqrt{3}}) + u(\frac{1}{\sqrt{3}}) = 0$
From observation, for this calculation these three methods are all accurate.
Q1.4
For $u = (\xi^2-1)^2$, we Compute the values of the (composite) trapezoidal rule for equidistant points, and 2,3 and 5 points:
n=1 h=2:

$$T(f) = 2\frac{u(-1)+u(1)}{2} = 0$$

n=2 h=1:

$$T(f) = \frac{u(-1)}{2} + u(0) + \frac{u(1)}{2} = 1$$

n=4 h=1/2:

$$T(f) = \frac{1}{2}[\frac{u(-1)}{2} + u(-\frac{1}{2}) + u(0) + u(\frac{1}{2}) + \frac{u(1)}{2}] = \frac{1}{2}(\frac{9}{16}\times 2+1) = \frac{17}{16}$$

The exact calculation is:

$$\int_{-1}^{1}(\xi^2-1)^2 d\xi = \frac{16}{15}$$

We construct the Romberg tableau $R_{i,j}$ using the equation

$$R[j,0] = T(2^{-j}(b-a))$$

$$R[j,k] = \frac{4^k R[j,k-1] - R[j-1,k-1]}{4^k-1}$$

and the following code similar to one in lecture.

```
In [1]: import sympy as sy
        import numpy as np
        import math

In [2]: import numpy as np
        f   = lambda x : (x**2-1)**2
        intf = 16/15

        # trapezoidal rule
        T = lambda f,n,a=-1.0,b=1.0 : (b-a)/n*((f(a)+f(b))/2.0 \
            + np.sum(f(np.linspace(a+(b-a)/n,b-(b-a)/n,n-1))))


        mr = 4; h = 0.1
        R = np.zeros((mr,mr))
        for n in range(mr):
            R[n,0] = T(f,2**n)
        for k in range(1,mr):
            for j in range(k,mr):
                R[j,k] = (4**k*R[j,k-1] - R[j-1,k-1])/(4**k-1)
        print(R)
        print("exact value: {}".format(intf))
```

```
[[0.         0.         0.         0.        ]
 [1.         1.33333333 0.         0.        ]
 [1.0625     1.08333333 1.06666667 0.        ]
 [1.06640625 1.06770833 1.06666667 1.06666667]]
exact value: 1.0666666666666667
```

We use error approximation $e_h \approx R[j,j] - R[j+1,j+1]$ for calculation,
n=1 j=0 $e_h \approx 1.33$
n=2 j=1 $e_h \approx 0.26$
n=4 j=2 $e_h \approx 0$
Hence we conclude that the calculation with 5 points is exact.
Also, from the Euler-Maclaurin formula,

$$T(f,h) - \int_a^b f(x)\,dx = \sum_{k=1}^{m-1} h^{2k} \frac{B_{2k}}{(2k)!} \left(f^{(2k-1)}(b) - f^{(2k-1)}(a) \right) + h^{2m} \frac{B_{2m}}{(2m)!}(b-a) f^{(2m)}(\xi)$$

(where B_{2k} are the Bernoulli numbers)
we know that the columns converge and become more precise as j increases. Hence the calculation with 5 points is exact.
Q1.5
Gauss quadrature with $n+1$ points is exact for all polynomials $p(x)$ of degree up to $2n+1$, i.e.,

$$Q(p) = \int_{-1}^{1} p(x)\,dx$$

the degree of polynomial $u = (\xi^2 - 1)^2$ is 4, so n=2 and 3 quadrature points is enough.
We compute the legendre polynomials for degree up to 3 recursively, starting with $q_0(x) = 1$ and

$$q_{k+1}(x) = x q_k(x) - c_{k-1} q_{k-1}(x)$$

where

$$c_{k-1} = \frac{\int_{-1}^{1} x q_k(x) q_{k-1}(x)\,dx}{\int_{-1}^{1} q_{k-1}^2\,dx}$$

In [3]:
```
# computing Legendre polynomials qk(x)
n = 3

x = sy.Symbol('x')

qkm1  = 1
qk    = x

for k in range(n):
    qkp1 = sy.simplify(x*qk - sy.integrate(x*qk*qkm1,(x,-1,1))/sy.integrate(qkm1**2,(x,-
    qkm1 = sy.expand(qk)
```

```
            qk    = qkp1

            print("q{:1d}(x) = {}".format(k+1,qkm1))
```

```
q1(x) = x
q2(x) = x**2 - 1/3
q3(x) = x**3 - 3*x/5
```

3 quadrature points z_k are the zeros of the Legendre polynomial q_3

```
In [4]: # compute the Gauss quadrature points
        c = sy.Poly(qkm1).all_coeffs() # Legendre coefficients
        z = np.roots(c)  # zeros Legendre fct = quad. pts
        z.sort() # sort by size
        z
```

```
Out[4]: array([-0.77459667,  0.        ,  0.77459667])
```

We compute the weights using the Lagrange interpolation formula, which is

$$w_k = \int_{-1}^{1} l_k(x)\,dx$$

```
In [5]: # compute the weights
        n = z.shape[0]-1
        w = np.zeros(n+1)
        x = sy.Symbol('x')
        print("\n n = {}:".format(n))
        for j in range(n+1):
            lj = 1
            for k in range(n+1):
                if (k!=j): lj *= (x-z[k])/(z[j]-z[k])
            w[j] = float(sy.integrate(lj,(x,-1,1)))
            print("w{} = {:4.4f}".format(j,w[j]),end='    ')
```

```
 n = 2:
w0 = 0.5556    w1 = 0.8889    w2 = 0.5556
```

2 Q2

Q2.1

For $n = 2$ $x_k = 0,1,2$ We use the method of undertermined coefficient

$$D_h(u,\xi) = \sum_{k=0}^{n} c_k u(\xi_k)$$

consider monomials $p(x) = 1, x, x^2, \dots$ system of equations with Vandermonde matrix

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 2 \\ 0 & 1 & 4 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 2x \end{bmatrix}$$

by Gaussian elimination:

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 2 \\ 0 & 0 & 2 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 2\xi - 1 \end{bmatrix}$$

we get the solution for coefficient

$$c_2 = \xi - 1/2,\ c_1 = 2(1-\xi),\ c_0 = \xi - 3/2$$

By transformation $z = x + \xi h$, $u(\xi) = v(x + \xi h)$ and $u'(\xi) = hv'(x + \xi h)$
We get the approximation formula for $v'(z)$

$$v'(z) = \frac{u'(\xi)}{h} \approx \frac{c_0 u(0) + c_1 u(1) + c_2 u(2)}{h} = \frac{(\xi - 3/2)v(x) + 2(1-\xi)v(x+h) + (\xi - 1/2)v(x+2h)}{h}$$

The coefficients is computed such that formula is exact for polynomials p of degree up to $n = 2$

$$D_h(p, x) = p'(x)$$

so this approxiamtion is equal to v'(z) at any $z \in R$ for $v(z) = 1, z, z^2$
Coefficients for $\xi = 3/4$ are therefore:

$$c_0 = \xi - 3/2 = -3/4,\ c_1 = 2(1-\xi) = 1/2,\ c_2 = \xi - 1/2 = 1/4$$

And the approximation formula reads:

$$v'(z) = v'(x + \frac{3}{4}h) \approx \frac{-\frac{3}{4}v(x) + \frac{1}{2}v(x+h) + \frac{1}{4}v(x+2h)}{h}$$

Q2.2
Using the approximation formula above for $v(x) = e^x$, $\xi = 3/4$ and transformation $z = \xi h = \frac{3}{4}h$, we get:

$$v'(z) = v'(\frac{3}{4}h) \approx \frac{-\frac{3}{4}v(0) + \frac{1}{2}v(h) + \frac{1}{4}v(2h)}{h} = \frac{-\frac{3}{4} + \frac{1}{2}e^h + \frac{1}{4}e^{2h}}{h}$$

We calculate its value for $h = 1, 0.5, 0.25$
$h = 1$ $v'(\xi h) \approx 2.46$
$h = 0.5$ $v'(\xi h) \approx 1.51$
$h = 0.25$ $v'(\xi h) \approx 1.22$
The real value for this problem is

$$v'(\xi h) = e^{\xi h} = e^{\frac{3}{4}h}$$

$h = 1$ $v'(\xi h) = 2.12$,
$h = 0.5$ $v'(\xi h) = 1.45$,
$h = 0.25$ $v'(\xi h) = 1.21$
The error is

$h = 1$ $e(\xi h) = 0.339$,
$h = 0.5$ $e(\xi h) = 0.053$,
$h = 0.25$ $e(\xi h) = 0.011$

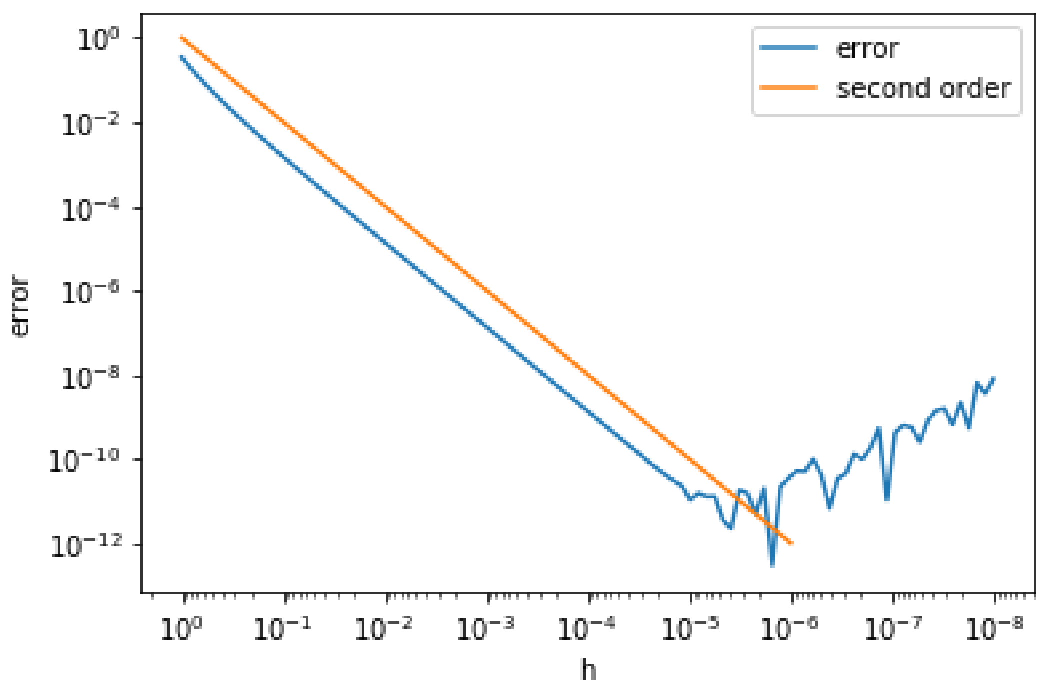
From the value We can guess that the dependence of the error on h is quadratic since as the h decrease to half of its previous value, the corresponding error decrease to approximate a quarter of its previous value.

In the following code, we study the dependence of the error on h in a wider range of h (10^{-8}~1).

In [30]:
```
import matplotlib.pyplot as plt
h=np.logspace(0,-8,100)
approx = (-0.75+0.5*np.exp(h)+0.25*np.exp(2.*h))/h
real = np.exp(0.75*h)
errvec= approx - real

h1=np.logspace(0,-6,100)

plt.loglog(h, abs(errvec),label="error")
plt.loglog(h1,h1**2,label="second order")
plt.xlabel("h")
plt.ylabel("error")
plt.legend()
plt.gca().invert_xaxis()
plt.show()
```



From the plot in loglog space we can clearly see that the dependence of the error on h is quadratic before rounding error dominates.

Q2.3
By taylor expansion including third order error term:

$$v(x) = v(x+\xi h) - v'(x+\xi h)\xi h + \frac{v''(x+\xi h)}{2}\xi^2 h^2 + \frac{v'''(x+\xi h)}{6}\xi^3 h^3 + O(h^4)$$

$$v(x+h) = v(x+\xi h) + v'(x+\xi h)(1-\xi)h + \frac{v''(x+\xi h)}{2}(1-\xi)^2 h^2 + \frac{v'''(x+\xi h)}{6}(1-\xi)^3 h^3 + O(h^4)$$

$$v(x+2h) = v(x+\xi h) + v'(x+\xi h)(2-\xi)h + \frac{v''(x+\xi h)}{2}(2-\xi)^2 h^2 + \frac{v'''(x+\xi h)}{6}(2-\xi)^3 h^3 + O(h^4)$$

We plug the above expansion into the approximation formula for derivative

$$\frac{(\xi-3/2)v(x) + 2(1-\xi)v(x+h) + (\xi-1/2)v(x+2h)}{h} = v'(x+\xi h) + \frac{(2\xi^4 - 3\xi^3 - 3\xi^2 + 6\xi - 2)v'''(x+\xi h)}{6}h^2 -$$

The truncation error formula

$$e = v'(x+\xi h) - \frac{(\xi-3/2)v(x) + 2(1-\xi)v(x+h) + (\xi-1/2)v(x+2h)}{h} = \frac{(2\xi^4 - 3\xi^3 - 3\xi^2 + 6\xi - 2)v'''(x+\xi h}{6}$$

We apply this formula to previous problem x=0, $\xi = 3/4$ and $v(z) = exp(z)$

$$e = \frac{23v'''(\frac{3}{4}h)}{768}h^2 + O(h^3) = O(h^2)$$

We also including rounding error

$$\frac{\frac{3}{4}|v(0)| + \frac{1}{2}|v(h)| + \frac{1}{4}|v(2h)|}{h}\epsilon \approx \frac{3v(\frac{3}{4}h)\epsilon}{2h}$$

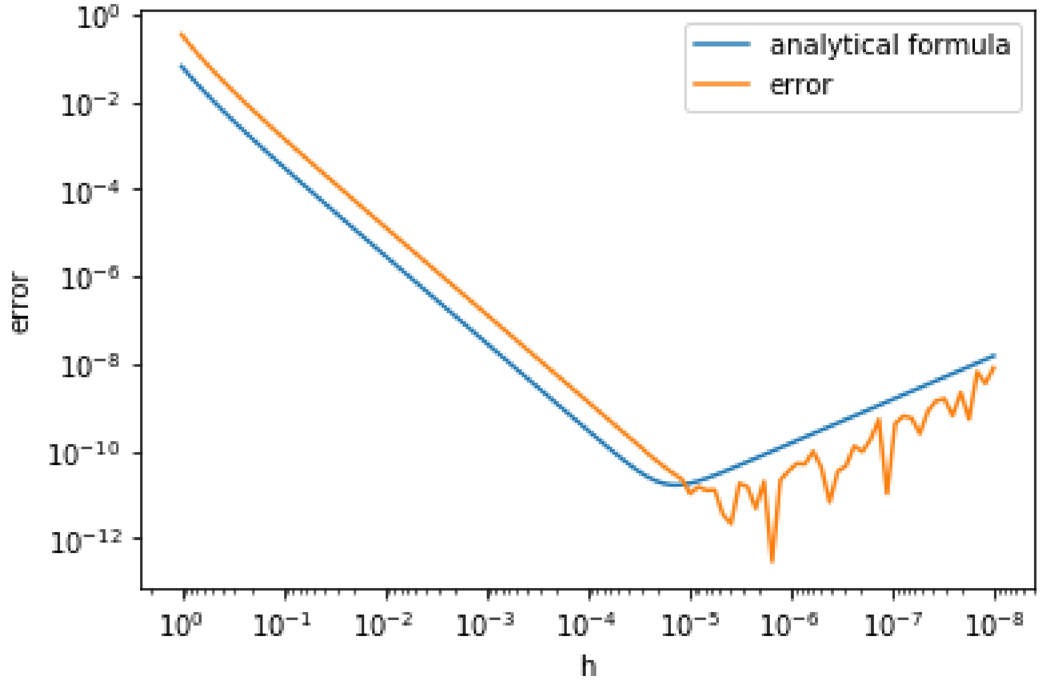
The total error are hence

$$|e| = \frac{|23v'''(\frac{3}{4}h)|}{768}h^2 + \frac{3v(\frac{3}{4}h)\epsilon}{2h} + O(h^3) = (\frac{23}{768}h^2 + \frac{3\epsilon}{2h})exp(\frac{3}{4}h) + O(h^3)$$

We compare our analytical formula with the real error and choose $\epsilon = 10^{-16}$ in the following code.

```
In [29]: ana = (23*h**2/768+3e-16/(2*h))*np.exp(0.75*h)
         plt.loglog(h,ana,label="analytical formula")
         plt.loglog(h, abs(errvec), label="error")
         plt.gca().invert_xaxis()
         plt.xlabel("h")
         plt.ylabel("error")
         plt.legend()

Out[29]: <matplotlib.legend.Legend at 0x15b13a26908>
```



Get the lower error bound by

```
In [28]: np.min(ana)
```

```
Out[28]: 1.6599614189045085e-11
```

The above error formula agrees well with the real case. As h can be arbitarily small, there is no upper bound from the error formula.

From the code, the lower bound is 1.66×10^{-11}, which is smaller than the value in Q2.2

3 Q3

Q3.1

The ODE reads

$$\frac{du}{dt} = u$$

Therefore

$$\frac{du}{u} = dt$$

By integration we get

$$u = Ce^{t}$$

Initial value u(0)=1 tells us that C=1 So the exact solution is:

$$u = e^{t}$$

Q3.2
From ODE we know $f(t,u) = u$
Euler's method is

$$u_{k+1} = u_k + (t_{k+1} - t_k) f(t_k, u_k)$$

for equidistant grid($t_k = hk$)

$$u_{k+1} = u_k + hf(t_k, u_k) = u_k + hu_k = (1+h)u_k$$

As $u_0 = u(t_0) = u(0) = 1$

$$u_k = (1+h)^k u_0 = (1+h)^k$$

as the approximation for $u(t_k)$, with $t_k = hk$
Q3.3
We use the formula $ln(1+h) = h - 0.5h^2 + O(h^3)$ and $t_k = hk$
Therefore

$$ln[(1+h)^k] = klog(1+h) = \frac{t_k}{h}(h - 0.5h^2 + O(h^3)) = t_k(1-0.5h) + O(h^2)$$

We calculate the exponential of them and use taylor expansion in the last step:

$$(1+h)^k = e^{t_k(1-0.5h)+O(h^2)} = e^{-0.5ht_k+O(h^2)}e^{t_k} = (1 - 0.5ht_k + O(h^2))e^{t_k}$$

We finally get

$$u(t_k) - u_k = e^{t_k} - (1+h)^k = e^{t_k} - (1 - 0.5ht_k + O(h^2))e^{t_k} = 0.5ht_ke^{t_k} + O(h^2)$$