Assignment #5 - Integrals

Hugo Hjertén, Sebastian Elm, Albin Johansson, Axel Lundholm, Adnan Mehmedagic

Task #2

The equation for the three point composite Simpson rule is:

$$\int_a^b f(x)\mathrm{dx} \approx \frac{h}{6}\left[f(a) + 4f(a+h/2) + \sum_{j=1}^{n-1} 2f(x_j) + 4f(x_j + h/2) + f(b)\right]$$

with error $\frac{(b-a)}{2880}h^4 f^{(4)}(\xi)$, where $h = (b-a)/n$ and $x_j = a + jh$ and

The equation for the three point composite Gauss rule is:

$$\int_a^b f(x)\mathrm{dx} \approx \frac{h}{18}\sum_{j=0}^{n-1}\left[5f\left(x_j + \frac{5-\sqrt{15}}{10}h\right) + 8f(x_j + \frac{5}{10}h) + 5f\left(x_j + \frac{5+\sqrt{15}}{10}h\right)\right].$$

where $h = h = (b-a)/n$ and $x_j = a + jh$.

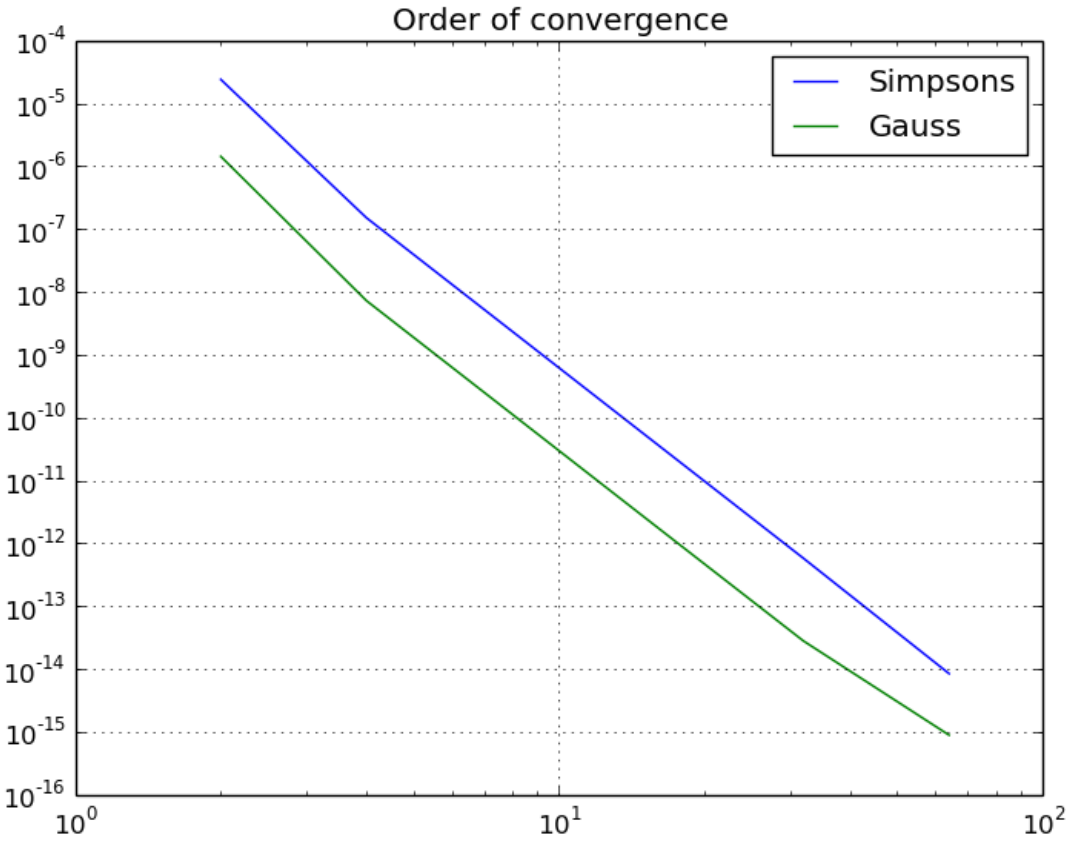
Below is the code for the three point composite Simpson rule and the three point composite Gauss rule.

```
def simpsons(f, a, b, n):

        h = abs(b-a)/n
        #print 'Stepsize = {}'.format(h)
        tmp = 0
        limit = int(n)
        for j in range(1,limit):
                xj = a + j*h
                #print xj
                tmp += 2.0*f(xj) + 4.0*f(xj + h/2.0)
                #print tmp
        integral = (h/6.0)*(f(a) + 4.0*f(a + h/2.0) + f(b) + tmp)
        return integral

def threePointGauss(f, a, b, n):
        h = abs(b-a)/n
        tmp = 0
        limit = int(n)
        for j in range(0, limit):
                xj = a + j*h
                tmp1 = f(xj + h*(5.0 - math.sqrt(15.0))/10.0)
                tmp2 = f(xj + 5.0*h/10.0)
                tmp3 = f(xj + h*(5.0 + math.sqrt(15.0))/10.0)
                tmp += 5.0*tmp1 + 8.0*tmp2 + 5.0*tmp3
        integral = h*tmp/18.0
        return integral
```

When using these two methods to find the integral of a function it becomes clear that the three point composite Gauss method always has a smaller error than the three point composite Simpsons method for the same number of steps (Y-axis is the error and the X-axis is the number of steps, n). In the figure below I used the function from task 3 and found the difference between the summated value of the integral using the two different methods and the real value, pi. The conclusion should be that the order of convergence for the composite Simpsons method is 4 and 6 for the composite Gauss. I do not see how I can come to that conclusion from this figure however.



Task #3

When examining the error for function listed below by using the error term function from Task 2, we could conclude that **14 steps** were needed to get an error smaller than 10^{-6}.

$$\int_0^1 \frac{4}{1+x^2}\, dx$$

The code can be found below:

```
def simpsonsError(a, b, n):
        h = abs(b-a)/n
        error = (abs(b-a)*96*h**4)/2880
        return error
def estimatedStep(a, b, error):
        n = 0
        est = 1
        while abs(est) > error:
                est = simpsonsError(0.0, 1.0, n+1)
                n += 1
        return n
```

To verify the estimation of n=14 and to find an equivalent value for the three-point composite Gauss rule we computed the integral within an error of 10^{-6} with the following code.

```
tol = 1e-6

while abs(solSimpsons - pi) > tol:
        stepsSimpsons += 1
        solSimpsons = simpsons(taskThree, 0.0, 1.0, stepsSimpsons)
while abs(solGauss - pi) > tol:
        stepsGauss += 1
        solGauss = threePointGauss(taskThree, 0.0, 1.0, stepsGauss)
```

For both methods a value of **n=3** was found. However, when the tolerance was changed to 10^{-8} instead, the Simpsons method required n=7 whilst Gauss' method only required n=4. None of these answers come even close to n=14 of course because that only corresponds to the maximum error possible. In other words, when calculating this integral the maximum error was not achieved when following the three point composite Simpsons rule.