Gaussian Double Integral

May 16, 2018

```
In [2]: from sympy import*
        %matplotlib inline
        import matplotlib.pyplot as plt
        from __future__ import division
        x, y, z, t = symbols('x y z t')
        k, m, n = symbols('k m n', integer = True)
        f, g, h = symbols('f g h', cls = Function)
        import math

In [3]: r = [[0]]
        r.append([-0.5773502691896257,0.5773502691896257])
        r.append([0.0000000000000000,-0.7745966692414834,0.7745966692414834])
        r.append([-0.3399810435848563,0.3399810435848563,-0.8611363115940526,0.861136311594052
        r.append([0.0000000000000000,-0.5384693101056831,0.5384693101056831,-0.906179845938664
        r.append([0.6612093864662645,-0.6612093864662645,-0.2386191860831969,0.238619186083196
        ,0.9324695142031521])
        r.append([0.0000000000000000,0.4058451513773972,-0.4058451513773972,-0.741531185599394
        ,0.9491079123427585])
        r.append([-0.1834346424956498,0.1834346424956498,-0.5255324099163290,0.525532409916329
        ,0.9602898564975363])
        r.append([0.0000000000000000,-0.8360311073266358,0.8360311073266358,-0.968160239507626
        ,-0.3242534234038089,0.3242534234038089,-0.6133714327005904,0.6133714327005904])
        r.append([-0.1488743389816312,0.1488743389816312,-0.4333953941292472,0.433395394129247
        ,0.6794095682990244,-0.8650633666889845,0.8650633666889845,-0.9739065285171717,0.97390

In [4]: c=[[2]]
        c.append([1.0000000000000000,1.0000000000000000])
        c.append([0.8888888888888888,0.5555555555555556,0.5555555555555556])
        c.append([0.6521451548625461,0.6521451548625461,0.3478548451374538,0.3478548451374538]
        c.append([0.5688888888888889,0.4786286704993665,0.4786286704993665,0.2369268850561891,
        c.append([0.3607615730481386,0.3607615730481386,0.4679139345726910,0.4679139345726910,
                  0.1713244923791704,0.1713244923791704])
        c.append([0.4179591836734694,0.3818300505051189,0.3818300505051189,0.2797053914892766,
                 0.1294849661688697,0.1294849661688697])
        c.append([0.3626837833783620,0.3626837833783620,0.3137066458778873,0.3137066458778873,
                 0.1012285362903763,0.1012285362903763])
        c.append([0.3302393550012598,0.1806481606948574,0.1806481606948574,0.0812743883615744,
```

```
            0.3123470770400029,0.3123470770400029,0.2606106964029354,0.2606106964029354])
        c.append([0.2955242247147529,0.2955242247147529,0.2692667193099963,0.2692667193099963,0
            0.2190863625159820,0.1494513491505806,0.1494513491505806,0.0666713443086881,0
```

```
In [5]: def Gauss(a, b, m, n, cfn, dfn, f):
            h1 = (b - a)/2
            h2 = (b + a)/2
            J = 0
            for i in range(1, m + 1):
                JX = 0
                x = h1*r[m-1][i-1] + h2
                d1 = dfn(x)
                c1 = cfn(x)
                k1 = (d1 - c1)/2
                k2 = (d1 + c1)/2
                for j in range(1, n + 1):
                    y = k1*r[n-1][j-1] + k2
                    Q = f(x,y)
                    JX = JX + c[n-1][j-1]*Q
                J = J + c[m-1][i-1]*k1*JX
            J = h1*J
            return J
```

```
In [6]: def SimpsonDouble(a, b, m, n, c, d, f):
            h = (b - a)/n
            J1 = 0
            J2 = 0
            J3 = 0
            for i in range (0, n+1):
                x = a + i*h
                HX = (d(x) - c(x))/m
                K1 = f(x, c(x)) + f(x, d(x))
                K2 = 0
                K3 = 0

                for j in range (1, m):
                    y = c(x) + j*HX
                    Q = f(x, y)
                    if j%2 == 0:
                        K2 = K2 + Q
                    else:
                        K3 = K3 + Q
                L = ((K1 + 2*K2 + 4*K3)*HX)/3
                if i ==0 or i == n:
                    J1 = J1 + L
                elif i % 2 == 0:
                    J2 = J2 + L
                else:
```

```
                J3 = J3 + L
        J = h*(J1 + 2*J2 + 4*J3)/3
        return J
```

```
In [7]: def c1(x):
            return 0
        def d1(x):
            return 1
        def f1(x, y):
            return 1 / (x + y)**2
        error_1 = []
        error_2 = []
        for i in range(1, 11):
            n = i
            gauss = Gauss(1, 2, n, n, c1, d1, f1)
            simpson = SimpsonDouble(1, 2, n, n, c1, d1, f1)
            error1 = abs(gauss - ln(4/3))
            error2 = abs(simpson - ln(4/3))
            error_1 = error_1 + [error1]
            error_2 = error_2 + [error2]
```