MANE HW4

Jianfeng Yan

November 17, 2017

Problem 1

Table 1: results of Gauss integral

number of Gauss Quadrature in each direction	integral
3	0.8119
4	0.8119
5	0.8287

```
function [ f ] = gaussIntegrate(x1, x2, y1, y2, n_qp, func)
if (x1 > x2)
    error('x1 > x2');
end
if (y1 > y2)
    error ('y1 > y2');
end

xi = zeros(n_qp, 1);
w = zeros(n_qp, 1);

if n_qp == 1
    xi(1) = 0.0;
    w(1) = 2.0;
elseif n_qp == 2
    xi(1) = -0.577350269189625764509148 8;
    xi(2) = 0.577350269189625764509148 8;
    w(1) = 1.0;
    w(2) = 1.0;
elseif n_qp == 3
    xi(1) = -0.774596692414833770359;
    xi(2) = 0.0;
    xi(3) =  0.774596692414833770359;
    w(1) = 0.555555555555555555556;
    w(2) = 0.888888888888888888889;
    w(3) = 0.555555555555555555556;
elseif n_qp == 4
    xi(1) = -0.861136311594052575223 9;
    xi(2) = -0.339981043584856264802 7;
    xi(3) =  0.339981043584856264802 7;
    xi(4) =  0.861136311594052575223 9;
    w(1) = 0.347854845137453857373 1;
    w(2) = 0.652145154862546142626 9;
    w(3) = 0.652145154862546142626 9;
```

```
    w(4) = 0.34785484513745385731;
elseif n_qp == 5
    xi(1) = -sqrt(5 - 2*sqrt(10/7)) / 3;
    xi(2) = -sqrt(5 + 2*sqrt(10/7)) / 3;
    xi(3) = 0.0;
    xi(4) = -xi(2);
    xi(5) = -xi(1);
    w(1) = (322 + 13*sqrt(70)) / 900;
    w(2) = (322 - 12*sqrt(70)) / 900;
    w(3) = 128 / 225;
    w(4) = w(2);
    w(5) = w(1);
end

x = x1 + (xi + 1) * (x2 - x1) / 2;
y = y1 + (xi + 1) * (y2 - y1) / 2;
f = 0.0;
for i = 1 : n_qp
sub_sum = 0.0;
for j = 1 : n_qp
sub_sum = sub_sum + w(j) * func(x(i), y(j));
end
f  = f + w(i) * sub_sum;
end
f = f * (x2 - x1) * (y2 - y1) / 4;
end

integrand = @(x, y) sin(exp(-y))* exp(-x*y);
x1 = 0; x2 = 2; y1 = 0; y2 = 1;
n_qp = 3;
f = gaussIntegrate(x1, x2, y1, y2, n_qp, integrand);
```

Problem 2

1. There are two objectives in this problem. One possible way to deal with it is to transform the problem into a minimization problem with a single objective. For example, we can choose the new objective as

$$f = C/E$$

where C and E are the cost and efficient, respectively. Then we can solve the optimization problem as usual.

One way is to freeze either the cost or the efficiency based on existing database. Let's say we freeze the efficiency, then we can minimize the cost for multiple values of efficiency. In this way the rest work is to choose an appropriate solution from the results.

2. For the first approach, the implementation reads

```
objFunc = @(x) Cost(x) / Efficiency(x);
x0 = [1;1];
optns = optimoptions(@fmincon, 'Display', 'iter');
[x, fval] = fmincon(objFunc, x0, [], [], [], [], [], [], [], optns);
```

This gives us the optimizer $x^* = [-0.0000;\ 1.7513]$ with the optimal value 16.3471, the cost 13.0671, and the efficiency 0.7994.

3. In practice, this new objective function may be more complex. If we know the relationship between the commercial profit and (cost, efficient), this relationship can be used as the objective to be maximize. Otherwise, some algorithms targeting multiple objectives may be necessary.