Math 365 **Homework 6** **Due November 16, 2016**
Grady Wright

1. (**Degree of precision**, 10 pts) The degree of precision of a quadrature formula is defined as the largest positive integer m for which the formula gives the exact answer when applied to x^k, $k = 0, 1, \ldots, m$. That is, the quadrature formula $Q(f)$ has a degree of precision m if

$$Q(x^k) - \int_a^b x^k \, dx = 0, \quad k = 0, 1, \ldots, m. \tag{1}$$

The integration interval $[a, b]$ can be set to anything so long as $a < b$, so typically one chooses $a = -1$ & $b = 1$, or $a = 0$ & $b = 1$.

   (a) Determine the degree of precision of Trapezoidal rule by writing a **s**cript that determines the m such that (1) is satisfied. Note that the number of subintervals one uses is irrelevant (as long as it's at least 1). Also, you will not be able to check exact equality between $Q(f)$ and $\int_a^b x^k \, dx$ in your code since you are using floating point arithmetic. Instead, check that the absolute relative difference between $Q(f)$ and $\int_a^b x^k \, dx$ is smaller than 10^{-14}.

   (b) Repeat part (a) for Simpson's rule.

2. (**Gaussian Quadrature**, 20 pts) Midpoint, trapezoidal, and Simpson's rule linearly combine the evaluations of the integrand at equally spaced nodes in order to approximate

$$I(f) = \int_a^b f(x) \, dx.$$

These formulas are designed to have some fixed degree of precision (defined above) regardless of the number of intervals used in the approximation of the integral. Gaussian quadrature (GQ) rules also use a linear combination of values of the integrand to approximate the integral, but instead of using equally spaced nodes, they use non-equally spaced nodes so that the linear combination of function values at these nodes *maximizes* the degree of precision that can be achieved. For example, the nodes and weights for the standard Gaussian quadrature formula with $n = 9$ are given by

i	x_i	c_i
1	−0.96816023950763	0.081274388361574
2	−0.83603110732664	0.18064816069486
3	−0.61337143270059	0.26061069640294
4	−0.32425342340381	0.31234707704
5	0	0.33023935500126
6	0.32425342340381	0.31234707704
7	0.61337143270059	0.26061069640294
8	0.83603110732664	0.18064816069486
9	0.96816023950763	0.081274388361574

These nodes and weights are used as follows:

$$\int_{-1}^{1} f(x)dx \;=\; \sum_{i=1}^{9} c_i f(x_i)$$

(a) Using the above GQ formula, compute an approximation to

$$\begin{aligned} I(f_1) &= \int_{-1}^{1} (x-1)^2 e^{-x^2} dx\,, \\ I(f_2) &= 2\int_{-1}^{1} \frac{1}{1+x^2} dx\,, \end{aligned}$$

which are the same function from problem 5 of homework 5. Report the absolute error in each approximation.

(b) Compare the errors you obtained in part (a) with errors for Simpson's rule using $n = 8$ subintervals, which corresponds to $n = 9$ function evaluations, the same as the GQ method. What, if any, conclusions can be drawn?

(c) Determine the degree of precision of the 9-point GQ formula given above using the same technique as discussed in problem 1.

3. (**Integral equations**, 25 pts) The temperature problem from Homework #3 was a specific example of a numerical method for solving the second order differential equation

$$u''(x) = f(x)$$

for some prescribed function $f(x)$ and some prescribed boundary conditions. The discretization of the second derivative in this equation led to a linear system of equations $A\mathbf{u} = \mathbf{f}$, which we solved using the Thomas algorithm (`tridisolve`) or just using backslash on sparse matrices in MATLAB . An alternative method for solving (3) is to use an *integral equation* formulation of the above problem.

For (3) on the interval $[0, 1]$ and subject to $u(0) = a$ and $u(1) = b$, this formulation leads to the following formal solution:

$$u(x) = \left(a - \int_0^x \xi f(\xi)\, d\xi\right)(1-x) + \left(b + \int_x^1 (\xi - 1) f(\xi)\, d\xi\right) x, \tag{2}$$

This is referred to as an integral equation.

(a) Use the `quad` function in MATLAB to numerically evaluate the above integrals for solving the Heat conduction problem for a rod from homework 3. In this case, the function $f(x)$ in (3) is

$$f(x) = -\frac{S}{\rho c_p \beta} \exp\left(-\left(\frac{x - 1/2}{\varepsilon}\right)^2\right),$$

and the boundary conditions are $u(0) = a = 293.15$ and $u(1) = b = 293.15$. The parameters should be set to $\beta = 10^{-3}$, $S = 1 \times 10^4$, $c_p = 200$, $\rho = 10$, and $\varepsilon = 5 \times 10^{-2}$. Evaluate the solution for $x_j = jh$, $j = 0, 1, \ldots N$, and take $N = 1000$. Plot your solution along with the exact solution on the same graph. Add a title, axis labels and a legend. Your plot should closely resemble the plot shown in Figure 1.

(b) Using the same f and parameters from part (a), determine the value of x in the interval $[0, 1/2]$ where $u(x) = 273.15 + 70$. For this problem you need to use the `fzero` function on a function involving the integral equation formula (2). This will be the point at which skin will burn in less than a second if touching the rod.

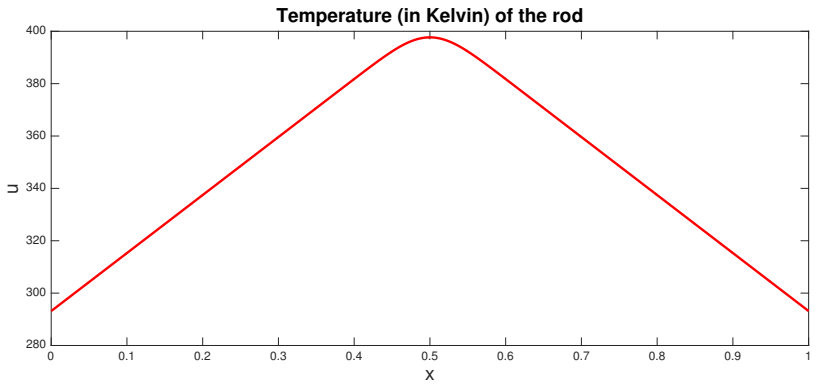

Figure 1: Plot of the solution to part

4. (**Quasi-Monte Carlo integration**, 20 pts) For numerically approximating high-dimensional integrals, the standard integration techniques (Trapezoidal, midpoint, Simpsons) are not useful because they suffer from the *curse of dimensionality*. As discussed in class, these types of integrals are best approximated using Quasi-Monte Carlo integration methods. In this problem you will compare two quasi-Monte Carlo integration techniques that we discussed in class and look at how they compare to Trapezoidal rule. For the comparisons you will compute an approximation to the 6D integral

$$I(f) = \int_0^1 \int_0^1 \int_0^1 \int_0^1 \int_0^1 \int_0^1 \sin\left(\frac{\pi}{2}\left(x_1 + x_2 + x_3 + x_4 + x_5 + x_6\right)\right) dx_1\, dx_2\, dx_3\, dx_4\, dx_5\, dx_6\,. \tag{3}$$

   (a) Determine the exact solution to (3).

   (b) Compute an approximation to the integral using quasi-Monte Carlo with $N = 10,000$ random points over the six dimensional hyper-cube $[0,1]^6$ generated from the `rand(N,6)` function. Set the seed for the random number generator using `rng(11032016)` before generating the points. Report the relative error in the approximation.

   (c) Repeat part (b) but use the Halton sequence points generated from `net(haltonseq(6),N)`.

   (d) Compute an approximation using Trapezoidal rule with $n = 10$ points in each direction. The code below computes the Trapezoidal rule approximation to a six dimensional integral over $[0,1]^6$. You just have to supply n and the integrand.

```
h = 1/n;
d = 6;
% Trapezoidal rule weights in one dimension
w = w1;
% Generate the trapezoidal rule weights in 6 dimensions
for j=1:d-1
    w = bsxfun(@times,w,reshape(w1,[ones(1,j) n+1]));
end
% Grid of points
[x1,x2,x3,x4,x5,x6] = ndgrid(linspace(0,1,n+1));
g = w.*f(x1,x2,x3,x4,x5,x6);
Qtrap = sum(g(:));
```

   (e) Compare the methods from (b), (c), and (d) in terms of accuracy vs the total number of points used.

5. (**Pursuit curves**, 25 pts) A fox is chasing a rabbit, who is happily following a path traced out by the parametric curve $\mathbf{R}(t) = (x_R(t), y_R(t))$. If the fox moves at the same speed as the

rabbit (let's say speed 1), and always heads towards the rabbit, we can write down a differential equation for the fox's trajectory $\mathbf{F}(t) = (x_F(t), y_F(t))$ as

$$\mathbf{F}'(t) = \| \mathbf{R}'(t) \| \frac{\mathbf{R} - \mathbf{F}}{\| \mathbf{R} - \mathbf{F} \|}$$

Suppose the rabbit runs in a circular path around a duck pond. This path is given by $\mathbf{R}(t) = (\cos(t), \sin(t))$. Solve for the pursuit path traced out by the fox, and plot your results. Use `ode45`. Try out different initial locations for the fox and the rabbit (assume the rabbit is always on the curve $\mathbf{R}(\mathbf{t})$). Turn in a plot of the pursuit curve for three different starting configurations.

To compute the norm of a vector, use `norm`. That is, $\| \mathbf{v} \| =$ `norm(v, 2)`. See Figure 2.

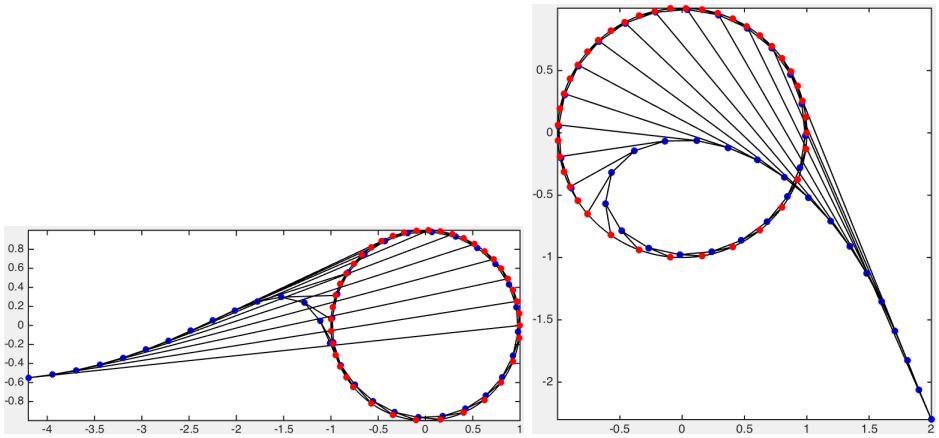

Figure 2: Pursuit curves from two different starting locations. The fox (blue) is chasing the rabbit (red). Both are moving at unit speed.