Numerical Analysis 1 – Class 12

Friday, April 7th, 2023

Subjects covered

- Numerical integration and Newton-Cotes methods in 1D.
- Gaussian quadrature in 1D.
- Clenshaw-Curtis and other quadrature methods.
- Gaussian quadrature in 2D.
- Integrating odd 2D shapes using a triangular mesh.

Reading

- Kutz, Chapt 4.2 – 4.3.
- Chapter on quadrature from "Numerical Computing with MATLAB", C. Moler (linked on Canvas).
- “The surveyor's area formula”, B. Braden (on Canvas)

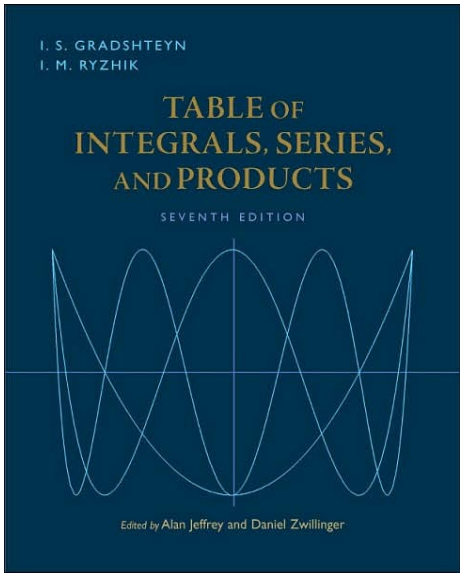
Problems

Most of the following problems require you to write a program. For each program you write, please make sure you also write a test which validates your program. Please use Canvas to upload your submissions under the “Assignments” link for this problem set.

Problem 1

Consider the definite integral

$$\int_0^1 \frac{\arctan(x)}{x\sqrt{1-x^2}}dx = \frac{\pi}{2}\ln(1+\sqrt{2}) \qquad (1)$$



This nasty-looking integral appears in the classic reference book, *Table of Integrals, Series, and Products*, by the Russian mathematicians I. S. Gradshteyn and I. M. Ryzhik. The book was first published in the 1940s and has been considered an essential reference by generations of working mathematicians and physicists. It is quite a large book; my copy is 1160 pages long.

In class I mentioned several quadrature methods beyond Gauss-Legendre. Equation (1) is of the form where the integral may be evaluated using Gauss-Chebyshev quadrature. That rule is

$$\int_{-1}^{1} \frac{f(t)}{\sqrt{1-t^2}}dt \approx \sum_{i=1}^{N} w_i f(t_i)$$

where the weights are given by the rule

$$w_i = \frac{\pi}{N}$$

and the sample points are

$$t_i = \cos\left(\frac{2i-1}{2N}\pi\right) \qquad i = 1 \ldots N$$

Note that these are the sample points we first encountered when using Chebyshev polynomials for interpolation – hence the name "Gauss-Chebyshev quadrature".

Please write a program to evaluate the integral in equation (1) using Gauss-Chebyshev quadrature for different values of *N*. To test, please compare your numerical result against the mathematically true result. What is the minimum *N* required to achieve accuracy better than 1e-4?

Problem 2

In class we studied the Simpson's rule approximation for 1D integrals. Two dimensional (and higher-dimensional) analogs of this 1D method exist. We will consider 2D Simpson's rule here.

In one dimension, Simpson's rule said that if you have a function sampled at an evenly-spaced set of points x_i , then the integral could be approximated by

$$I_1 = \int_a^b z(x)\,dx \approx h \sum_{i=0}^{N} w_i z(x_i)$$

with $h = \Delta x = x_2 - x_1$ and weight vector $w = \frac{1}{3}[1, 4, 2, 4, \cdots, 4, 2, 4, 1]$

A similar formula holds in two dimensions. It is:

$$I_2 = \int_a^b \int_c^d z(x, y)\,dx\,dy \approx h_x h_y \sum_{j=0}^{M} \sum_{i=0}^{N} w_{ij} z(x_i, y_i)$$

with h_x and h_x defined in the expected way, and with weights

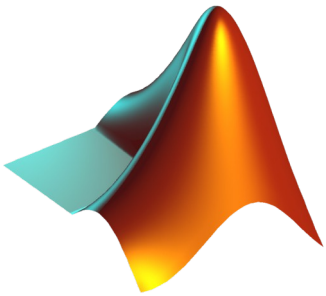
$$w_{ij} = \frac{1}{9}\begin{vmatrix} 1 & 4 & 2 & 4 & 2 & \cdots & 2 & 4 & 2 & 4 & 1 \\ 4 & 16 & 8 & 16 & 8 & \cdots & 8 & 16 & 8 & 16 & 4 \\ 2 & 8 & 4 & 8 & 4 & \cdots & 4 & 8 & 4 & 8 & 2 \\ 4 & 16 & 8 & 16 & 8 & \cdots & 8 & 16 & 8 & 16 & 4 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 4 & 16 & 8 & 16 & 8 & \cdots & 8 & 16 & 8 & 16 & 4 \\ 2 & 8 & 4 & 8 & 4 & \cdots & 4 & 8 & 4 & 8 & 2 \\ 4 & 16 & 8 & 16 & 8 & \cdots & 8 & 16 & 8 & 16 & 4 \\ 1 & 4 & 2 & 4 & 2 & \cdots & 2 & 4 & 2 & 4 & 1 \end{vmatrix}$$

(Note that the number of rows or cols must be odd.) Please write a program which implements 2D Simpson's rule. The inputs will be the matrix of sampled z values, along x and y sample point vectors.

To test your programs, check your results by integrating the functions $z(x, y) = (1 - x^2)(1 - y^2)$ and

$z(x,y)=\left(1-\cos^2\left(\frac{\pi x}{2}\right)\right)\left(1-\cos^2\left(\frac{\pi y}{2}\right)\right)$ analytically over the square bounded by $x=\pm 1$, $y=\pm 1$, then comparing the analytic results against your program's results. (Feel free to use Wolfram Alpha to get the analytic results.)

Now use your 2D Simpson's rule program to integrate the Matlab's logo function (right). Assume the function is defined on the domain $x,y\in[-1,1]$. You can get (and plot) the logo using these Matlab commands:



```
N = 101;  % Must be odd
s = linspace(-1, 1, N);
[x, y] = meshgrid(s, s);
z = membrane(1, (N-1)/2);
surf(x, y, z)
```

What value do you get for the integral? To 5 decimal places I get 1.04834....

Problem 3

In class we looked at using triangular meshes as a way to integrate objects of arbitrary shape in 2 dimensions. In this problem you will integrate over a mesh to find the surface area of some 3 dimensional objects: First, a unit cube, then the surface area of a unit sphere, and finally the surface area of a rabbit (the "Stanford bunny").

I have placed STL files on Canvas for all three objects. An STL file describes a 3D object as a collection of triangles and vertex points similar to the representation described in class. (As an aside, the STL file format was originally developed for CAD systems but is now finding widespread use describing geometry for 3D printing systems. If you have done any 3D printing it is likely you have dealt with an STL file.) Use the following Matlab functions to read in the files and create the triangle and vertex arrays:

```
S = stlread("sphere.stl")
pts = S.Points
tris = S.ConnectivityList
```

If you wish to view the STL files using a powerful 3D viewer, I recommend you download and install Paraview from this location https://www.paraview.org/.

Please write a program which does the following:

- Read in an STL file describing a 3-dimensional geometric object using the above Matlab functions.
- Plot the object using the Matlab function trisurf(). This step is important to verify you correctly read in the object.
- Loop over all the triangles in the object and compute their area. To compute each triangle's area, compute the vectors describing two of the triangle's edges, $\vec{u}=\vec{p}_2-\vec{p}_1$ and $\vec{v}=\vec{p}_3-\vec{p}_1$, then compute the area from the (norm of the) cross product, $A=(1/2)\|\vec{u}\times\vec{v}\|$.

- Sum the individual triangle areas to get the total surface area of the input object.
- Print out the sum.

Regarding testing:

- “cube.stl” is a unit cube. The total area of its six sides is 6. Your program should report the area is 6.
- “sphere.stl” is a unit sphere (radius = 1). Use standard formulas to determine the surface area of the unit sphere. If your program correctly computes the surface of the unit sphere (within 1% or less) then you can be confident your program is correct.
- “bunny.stl” is a rabbit of unknown surface area. Your program should just print out the area.

With this problem I hope to convince you that although the math is not deep (at least in this case), the technique of representing geometric objects using triangular meshes is extremely useful when dealing with objects having complicated shapes. Since most objects in the real world have complicated shapes, meshes are an important way to represent them in a computer.