Numerical Integration

Mech 510

Fall, 2018

This exercise comes in several parts, with the intention that you can choose to find your own level of difficulty by where you choose to start.

1 Before Class

On Canvas, there are several functions that we'll be integrating in this exercise. Downloading these before class is a good idea, so that you don't get hung up with network issues in class.

Also, write a short subroutine that takes as arguments three values of a definite integrals, on three consecutive meshes whose sizes have a ratio of 2, and computes the order of accuracy and extrapolated ($\Delta x = 0$) value of the integral. That is, you should write this function:

```
C version:
void richardson(const double fine, const double medium,
                const double coarse, double* order,
                double* extrapolated);

FORTRAN90 version:
subroutine richardson(fine, medium, coarse,
                      order, extrapolated)
  REAL*8, intent(in) :: fine, medium, coarse
  REAL*8, intent(out) :: order, extrapolated
```

To test the correctness of your subroutine, pass it the data 0.56, 0.44, 0.41 for fine, medium, and coarse. You should get order = 2 and extrapolated = 0.5. This is an example of a *unit test*, a test of a single subroutine with known input and output, to ensure correct functioning of that subroutine in isolation.

2 Trapezoidal Rule Integration

Write a subroutine that does trapezoidal rule integration. Write this as a function that takes four arguments: the function to be integrated, the limts of integration, and the number of intervals; the function should return the value of the integral:

```
C version:
double trapezoidal(double(*func)(const double), const double lowerLimit,
                   const double upperLimit, const int numIntervals)

FORTRAN90 version:
function trapezoidal(func, lower, upper, numIntervals)
  REAL*8 :: trapezoidal
  REAL*8, external :: func
  REAL*8, intent(in) :: lower, upper
  INTEGER, intent(in) :: numIntervals
```

That first argument means "a pointer to a function that takes a `const double` argument and returns a `double`". You'll also notice that I'm a big fan of declaring arguments as `const` when appropriate. This tells readers of your code "I don't intend to ever change this variable in this function"; the compiler also knows, and it will warn you if you violate that intention.

Use `trapezoidal` to integrate the function $\frac{\pi}{2}\sin(\pi x)$, (coded as a function that takes x as an argument and returns the value[1]), starting with a coarse mesh with 10 intervals on (0,1). Once that appears to be working satisfactorily (a test with two intervals should be easy to compare to a hand-calculated value), repeat for 20 and 40 intervals. Use your `richardson` routine to confirm that trapezoidal rule integration is really second order and that you get the correct analytic answer, which is 1.

3 Gauss Quadrature

Gauss quadrature has the advantage of being $2k$ order accurate for k integration points per interval. Essentially, the quadrature point locations and weights are chosen so that monomials up to degree $2k-1$ are integrated exactly. Write a Gauss quadrature routine that takes the same arguments as trapezoidal, above, plus a final argument for the number of quadrature points per interval. Here's a table of Gauss point locations and weights, for integration on a unit interval; I've included the trapezoidal rule also, for comparison:

Type	Order	Point	x_i	w_i
Trapezoidal	2	1	0	$\frac{1}{2}$
		2	1	$\frac{1}{2}$
Gauss	2	1	$\frac{1}{2}$	1
	4	1	$\frac{3+\sqrt{3}}{6}$	$\frac{1}{2}$
		2	$\frac{3-\sqrt{3}}{6}$	$\frac{1}{2}$
	6	1	$\frac{5+\sqrt{15}}{10}$	$\frac{5}{18}$
		2	$\frac{1}{2}$	$\frac{4}{9}$
		3	$\frac{5-\sqrt{15}}{10}$	$\frac{5}{18}$

Do the same tests as for `trapezoidal`, to show that your Gauss quadrature implementation has the advertised order of accuracy.

[1] You'll want to use the true value for π, to machine precision. In C/C++, use `M_PI`. In Fortran, I've always used `4*atan(1.d0)`, but there may be a pre-defined symbol for it in modern Fortran.

4 Stress Test

There are still a number of things that may have gone wrong with your code

Now compile and link your code with the functions provided on Canvas. Use `trapezoidal` and/or `gauss` to find the value of the integrals of `exponential` (limits: 1 to 3) and `bessel` (same limits).

5 Two Dimensional Numerical Integration

Also sometimes called cubature. For a rectangular grid, you can simply use a tensor product of one-dimensional quadrature rules. For instance,

Type	Order	Point	x_i	y_i	w_i
Trap	2	1	0	0	$\frac{1}{4}$
		2	1	0	$\frac{1}{4}$
		3	0	1	$\frac{1}{4}$
		4	1	1	$\frac{1}{4}$
Gauss	2	1	$\frac{1}{2}$	$\frac{1}{2}$	1
	4	1	$\frac{3+\sqrt{3}}{6}$	$\frac{3+\sqrt{3}}{6}$	$\frac{1}{4}$
		2	$\frac{3-\sqrt{3}}{6}$	$\frac{3+\sqrt{3}}{6}$	$\frac{1}{4}$
		3	$\frac{3+\sqrt{3}}{6}$	$\frac{3-\sqrt{3}}{6}$	$\frac{1}{4}$
		4	$\frac{3-\sqrt{3}}{6}$	$\frac{3-\sqrt{3}}{6}$	$\frac{1}{4}$

Type	Order	Point	x_i	y_i	w_i
Gauss	6	1	$\frac{5+\sqrt{15}}{10}$	$\frac{5+\sqrt{15}}{10}$	$\frac{25}{324}$
		2	$\frac{1}{2}$	$\frac{5+\sqrt{15}}{10}$	$\frac{10}{81}$
		3	$\frac{5-\sqrt{15}}{10}$	$\frac{5+\sqrt{15}}{10}$	$\frac{25}{324}$
		4	$\frac{5+\sqrt{15}}{10}$	$\frac{1}{2}$	$\frac{10}{81}$
		5	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{16}{81}$
		6	$\frac{5-\sqrt{15}}{10}$	$\frac{1}{2}$	$\frac{10}{81}$
		7	$\frac{5+\sqrt{15}}{10}$	$\frac{5-\sqrt{15}}{10}$	$\frac{25}{324}$
		8	$\frac{1}{2}$	$\frac{5-\sqrt{15}}{10}$	$\frac{10}{81}$
		9	$\frac{5-\sqrt{15}}{10}$	$\frac{5-\sqrt{15}}{10}$	$\frac{25}{324}$

Even though I've listed these as separate points, it may make more sense to code this as a nested loop for the two directions of integration.

Write a function gauss2d that does 2D integration over a region between (xmin,ymin) and (xmax, ymax) with the same number of intervals in each direction. Verify that your routine gives the correct integral (1) and correct order of accuracy for the `sinsin` function provided on the square (0,0) to (1,1). Then evaluate the integral of the `sinsinh` function on that same domain.