EE5471: Numerical Integration

Harishankar Ramachandran
EE Dept, IIT Madras

August 15, 2011

1 Reading Portion

Chapter 4 of Numerical Recipes.

2 Carry over from Previous Assignments

You will need `polint` and `spline` routines. Please make sure you have got them working from Python.

3 Programming Portion

The total power in an Electromagnetic mode has to be computed in order to properly normalize the fields. The integral involved is

$$\frac{1}{2}\int\int \left(E_x H_y^* - E_y H_x^*\right) dxdy$$

For dielectric fibres, this calculation involves the following integral.

$$I = \frac{2}{a^2}\int_0^a J_\nu^2(\kappa r)\, rdr + \frac{2}{a^2}\left|\frac{J_\nu(\kappa a)}{K_\nu(\gamma a)}\right|^2 \int_a^\infty K_\nu^2(\gamma r)\, rdr \tag{1}$$

where J_ν is the Bessel function of the first kind and K_ν is the modified Bessel function of the second kind (standard special functions).

The exact solution to Eq. (1) is known to be

$$\left[J_\nu^2(\kappa a) - J_{\nu+1}(\kappa a) J_{\nu-1}(\kappa a)\right] + \left|\frac{J_\nu(\kappa a)}{K_\nu(\gamma a)}\right|^2 \left[K_{\nu+1}(\kappa a) K_{\nu-1}(\kappa a) - K_\nu^2(\kappa a)\right] \tag{2}$$

In this assignment, we are going to explore the numerical evaluation of this integral for the case of $\nu = 3$, $\kappa a = 2.7$ and $\gamma a = 1.2$.

1. Transform the integral to the dimensionless variable $u = r/a$. You should obtain:

$$I = 2\int_0^1 J_\nu^2(ku)udu + 2\left|\frac{J_\nu(k)}{K_\nu(g)}\right|^2 \int_1^\infty K_\nu^2(gu)udu$$

where $k = \kappa a = 2.7$ and $g = \gamma a = 1.2$.

2. Graph the integrand in Python from 0 to ∞ (use a semi-log plot, since the function amplitudes vary widely, but the function is positive everywhere). Study its characteristics, since those will be required to design your algorithm. Is the function continuous? Is it smooth everywhere?

3. Use Python's built-in integrator, `quad`, found in `scipy.integrate` to do the integration. Verify the given solution. How many calls were required?

   - You need to define the integrand as a Python function.
   - The function form depends on whether $u < 1$ or $u > 1$.
   - Ask for full output, and look at `scipy.integrate.quad_explain()` for details. One of the outputs is the number of calls made to the function.

4. Use the trapezoidal method and obtain the integral. How does the error scale with h? Does it matter if $r = a$ is one of the r_j values or if it lies between some r_j and r_{j+1}?Obtain the scaling of accuracy vs number of function calls both ways. (Accuracy is obtained by comparing to the theoretical answer).

   - You can use `scipy.integrate.trapz` or code your own code.
   - Note that number of function calls would be $N = 1 + (b-a)/h$

5. Use the built in `scipy.integrate.romberg` and study the scaling of real error and estimated error versus the number of function calls. Note that you should be able to explain the working of the NR program in the viva and discuss the same in your program output.

6. Split the romberg integrals into $(0,1)$ and $(1,\infty)$. Repeat the study. Explain the difference.

7. Use spline integration to improve over trapezoidal integration. What scaling do you get? Can you explain why? **Hint:** Look at the spline interpolated function near $r = a$.

8. Break up the spline integration into two separate parts, from 0 to 1 and from 1 to $u_{\max}$. How do the scalings change? What do you use as end conditions to terminate the splines? What does this tell us about spline fitting and spline integration?

9. Write a C code to implement Romberg where h is reduced to $h/3$ on each step and interface it to Python. Note that you should not evaluate the integrand at any point more than once. Numerically compare its performance with the standard Romberg.

10. $J_3(2.7r/a)$ is not uniform in magnitude over $r/a \in [0,1]$. According to the theory of Bessel functions,

$$J_3(x) \approx \frac{(x/2)^3}{3!} - \frac{(x/2)^5}{4!} + \ldots$$

    Use the first two terms as an approximation for $J_3(x)$ and simplify the integrand. Is your Romberg algorithm optimal? Can you transform the problem to make it more optimal (see below)? How do the real and estimated error scale with the number of function evaluations for the transformed problem?

11. Plot all the scalings on a single graph, and compare the various approaches used to understand which method works best.

Note: When is Romberg optimal?

Any scheme that samples a function to obtain its integral is optimal if it uses each and every sample equally well. So each sample must give the same amount of useful information. A function that is nearly zero everywhere except in a very small region is not usefully integrated by a uniformly sampled scheme - I would be much better off breaking the domain into multiple regions. A better approach would be to tranform the *function* to make it uniform:

Suppose I am given a function $f(x) = 0.01x^2$ that I want to integrate from 0 to 10. I define a transformation $u(x)$ such that $f(x)dx = g(u)du$, i.e.,

$$g(u) = f(x)\frac{dx}{du}$$

My goal is to make $g(u)$ as close to uniform as possible. So I guess $u = x^\alpha$ which means $x = u^{1/\alpha}$. Then,

$$\begin{aligned} g(u) &= u^{2/\alpha}\left(\frac{1}{\alpha}\right)\frac{u^{1/\alpha}}{u} \\ &= \frac{1}{\alpha}u^{\frac{3-\alpha}{\alpha}} \end{aligned}$$

That gives me the integral

$$I = 0.01\int_0^{10} x^2 dx = \frac{0.01}{3}\int_0^{1000} du = \frac{10}{3} = 0.01\frac{10^3}{3}$$

For a complex integral, obviously I cannot make $g(u)$ exactly a constant. But the more constant I make it, the better Romberg will perform.