Homework 3: Numerical integration and differentiation

Due Friday Feb. 8 on Canvas. Answers to the extra credit will be considered as a factor in participation portion of you final grade (but you can get 100% there without it).

Special notes: A) For all the problems, be careful with how you set up your grid in x: make sure the h you use is the same as the spacing between points and that there are N and not $N \pm 1$ points. If you find that your error is not scaling as expected, mistakes in the setup of the evaluation grid may be the culprit.

B) To `print` more digits in the decimal `mynumber` than the default of ten, try in Python 2

```
print "printing 15 digits after decimal: ",  '{0:.15f}'.format(mynumber)
```

or in Python 3

```
print(format(mynumber, '.15f')
```

Questions:

1. Perform the integral
$$\int_0^2 dx(x^4 - 2x + 1),$$
which evaluates to 4.4, using
   (a) the rectangular integration method (midpoint centering) with $N = 10, 100,\ 1000$ midpoint evalulations;
   (b) trapezoidal integration method with $N = 10,\ 100\ 1000$ (following Newman's convention of evaluating at $N+1$ points);
   (c) Simpsons rule integration method with $N = 10,\ 100\ 1000$. (Note that Simpsons requires even N to work, following Newman's convention of evaluating function at $N+1$ points.)
   (d) Comment on the size of the error for each method. The Euler-McClaurin error formula for these methods is
$$\epsilon_{\text{trap}} = \frac{1}{12}h^2\left[f'(a) - f'(b)\right]; \quad (1)$$
$$\epsilon_{\text{trap}} = \frac{1}{180}h^4\left[f'''(a) - f'''(b)\right], \quad (2)$$

One can add these back to the solutions to make them more accurate. Try this for both Trapezoidal and Simpsons results and comment on the new solutions' accuracy. (Note the Euler-McClaurin error formula in Newman is off by a factor of 2 for the error on Simpsons method.)

(e) Now perform Romberg integration so that it has the precision of trapezoidal integration with $N = 9,\ 129,\ 1025$ (to roughly match the above and be consistent with $2^n + 1$ evaluates required by Romberg). It is easiest to do this with the scipy package, using `scipy.integrate.romb`.[1] You will have to pass `scipy.integrate.romb` an array as shown in the following example

```
import scipy.integrate as integrate

 #easy way to convert N=10 to N=9, N=1000 to N=1025 etc :
N = 2**int(np.log(N)/np.log(2)+.5)+1

xarr = np.linspace(0, 2, N) #make an array of size N
                            #with endpoints 0 and 2
h = xarr[1]-xarr[0]  #this is our stepsize
integrate.romb(myfunc(xarr), h)}
```

Alternatively, rather than install scipy, use someone's code on the internet (such as http://www.math-cs.gordon.edu/courses/ma342/python/romberg.py). Extra credit for writing your own Romberg integration routine!

(f) The gaussian quadrature method is even more accurate! Comment on how many evaluations of the above integral we would have to make with Gaussian quadrature for it to be exact (i.e. no error even if your computer could calculate with infinite precision). No calculation needed!

Perhaps the most popular integration routine in scientific computing, `scipy.intergrate.quad`, is a variant of Gaussian quadratures that uses a different set of orthogonal polynomial basis functions than Legendre Polynomials for which the weights are more easily computed. It also adaptively refines in regions where the quadrature rules are less accurate using an extension of the quadrature method.

(g) Repeat all the previous steps on the integral on $\int_0^2 dx \sin(11x)$,

[1] `scipy.integrate.romberg` is adaptive version we don't want to use. The adaptive version decides how to terminate based on supplied accuracy, following the discussion of Romberg in the textbook.

which evaluates to $(1 - \cos[22])/11$. Why do the methods fair rather poorly on this integral for the case $N \approx 10$?

2. Calculate the derivative of $f(x) = x^4 - 2x + 1$ analytically and numerically on a grid of N= 10 and 100 points spanning $0 \leq x \leq 2$ plus boundary padding to allow you to take the derivative. Do this for a derivative accurate to $\mathcal{O}(h)$ – the one sided derivative $[f(x+h) - f(x)]/h$ –, one accurate to $\mathcal{O}(h^2)$, and one accurate to $\mathcal{O}(h^4)$. All derivatives should evaluate $f(x)$ at grid points and not midpoints. Plot the absolute values of the fractional residuals as a function of x, using matplotlib.pyplot.semilogy where the fractional residuals are defined as

$$R(x) = \frac{\left(f'_{\text{numerical\_estimate}}(x) - f'_{\text{analytic}}(x)\right)}{f'_{\text{analytic}}(x)} \tag{3}$$

and of course $f'_{\text{analytic}} = 4x^3 - 2$. Comment on the size of the errors by comparing results of the the two N and why the $\mathcal{O}(h^4)$ does so well. For the latter it might be helpful to compare the results with a non-quartic polynomial function like $\sin(x)$ or x^5.

3. Exercise 5.14 in Newman: Gravitational pull of a uniform sheet with application to the Galaxy (2D integration).

4. For extra credit and your enjoyment (i.e. NOT REQUIRED): Exercise 5.17 in Newman on the Γ function