0 Instructions

This problem set is **due on Tuesday, November 5 by midnight.**. Submit your solutions on Canvas. Include the names of everyone you worked with on this problem set. Include any code you used to solve the problems as part of your submission.

1 Practice

Problem 1.1. Derive the Simpson's rule approximation

$$\int_a^b f(x)dx \approx \frac{b-a}{6}\left(f(a) + 4f(\frac{a+b}{2}) + f(b)\right)$$

by first approximating $f(x) \approx P_2(x)$ using a degree 2 polynomial $P_2(x)$ and then computing the integral $\int_a^b P_2(x)dx$ to obtain the correct integration weights.

Problem 1.2. Explain what issues composite integration helps to avoid in numerical integration.

2 Gaussian Quadrature

The points for Gaussian quadrature on the interval $[-1, 1]$ come from roots of the *Legendre polynomials* $Q_n(x)$, which have the following recursive definition:

$$Q_{n+1}(x) = xQ_n(x) - \frac{n^2}{4n^2-1}Q_{n-1}(x) \qquad \text{and} \qquad Q_0(x) = 1, Q_1(x) = x.$$

Problem 2.1. Use the definition above to find $Q_5(x)$, and give (exact) formulas for the five roots $x_0, \dots, x_4$ of $Q_5(x)$.

The weights for Gaussian quadrature on the interval $[-1, 1]$ come from the Lagrange polynomials we have talked about so far. Specifically, if $x_0, \dots, x_n$ are the roots of Q_{n+1}, then the weights are computed by

$$w_i = \int_{-1}^1 L_{n,i}(x)^2 dx.$$

Problem 2.2. Find the weights $w_0, \dots, w_4$ so that the following approximation is exact

$$\int_{-1}^1 f(x)dx = \sum_{k=0}^4 w_k f(x_k)$$

whenever f is a polynomial of degree less than 10.

Problem 2.3. For any real numbers $a < b$, find exact formulas for points $y_0, \dots, y_4$ and weights $u_0, \dots, u_4$ so that

$$\int_a^b f(x)dx = \sum_{k=0}^4 u_k f(y_k)$$

whenever f is a polynomial of degree less than 10.

We have the following error bound for 5-point Gaussian quadrature

$$\int_a^b f(x)dx - \sum_{k=0}^{4} u_k f(y_k) = \frac{f^{(10)}(\xi)}{10!} \int_a^b (y-y_0)^2 \dots (y-y_4)^2 dx$$

and for general n-point Gaussian quadrature, we have the corresponding error bound

$$\int_a^b f(x)dx - \sum_{k=0}^{n-1} u_k f(y_k) = \frac{f^{(2n)}(\xi)}{(2n)!} \int_a^b (y-y_0)^2 \dots (y-y_{n-1})^2 dx.$$

Problem 2.4. Using your results in the previous problems, implement 5-point Gaussian quadrature as follows. Your function should take the following inputs

1. $a < b$ endpoints of integration interval
2. f function to integrate

and return an approximation to the integral $\int_a^b f(x)dx$ using 5-point Gaussian quadrature that you found in the previous problems. Test your code by integrating the monomials $1, x, \dots, x^9$ on the interval $[-1, 1]$ and seeing whether you get the exact result. (Remember that n-point Gaussian integration exactly integrates polynomials of degree at most $2n - 1$, so we should expect to integrate polynomials of degree up to 9 correctly.)

3 Adaptive integration

Adaptive integration is typically the most effective technique for performing numerical integration. While we discussed adaptive quadrature using Simpson's rule in class, it can easily be modified to use any integration scheme. This problem uses the five point Gaussian integration rule that you derived in the previous problems.

Problem 3.1. Implement an adaptive 5-point Gaussian quadrature rule as follows. Your function should take the following inputs

1. $a < b$ endpoints of the integration interval
2. f function to integrate
3. *tol* tolerance for integral accuracy

and return an approximation to the integral $\int_a^b f(x)dx$ using the following procedure.

1. Initialize an "in-progress" list with $[(a_1 = a, b_1 = b, t_1 \approx \int_a^b f(x)dx)]$, a numerical variable to keep track of the integral so far, and a "done" list.
2. While the "in-progress" list is not empty, remove the last triple (a_k, b_k, t_k) and use it to produce two new triples $(a_k, \frac{a_k+b_k}{2}, t_l)$ and $(\frac{a_k+b_k}{2}, b_k, t_r)$ where integrals t_l and t_r are approximated using 5-point Guassian quadrature.

3. If the difference is larger than the specified tolerance

$$|t_k - (t_l + t_r)| > tol \max(|t_j|, |t_l| + |t_r|)$$

then add the two new triples back to the "in-progress" list. Otherwise, increment the integral by $t_l + t_r$ and add the two new triples to the "done" list.

Problem 3.2. Use your code to approximate the integral

$$\int_0^1 x^{-x} dx$$

and tabulate the total number of function evaluations needed to obtain p-digit accuracy for $1 \le p \le 14$.

Problem 3.3. Implement composite Simpson's rule as follows. Your function should take the inputs

1. $a < b$ endpoints of the integration interval
2. f function to integrate
3. num number of points to use (should be an odd number!)

and return an approximation to the integral $\int_a^b f(x)dx$ by splitting up the interval $[a, b]$ into $(num - 1)/2$ subintervals and applying Simpson's rule to each subinterval. For example, if $num = 5$, then we'll use the five points $a, a + h, a + 2h, a + 3h, a + 4h = b$ and produce the $(5 - 1)/2 = 2$ of subintervals $(a, a + h, a + 2h)$ and $(a + 2h, a + 3h, a + 4h)$ where we apply Simpson's rule on each part:

$$\int_a^b f(x)dx \approx \frac{h}{3}(f(a) + 4f(a + h) + f(a + 2h)) + \frac{h}{3}(f(a + 2h) + 4f(a + 3h) + f(a + 4f)).$$

Use your code to approximate the same integral as before

$$\int_0^1 x^{-x} dx$$

with $num = 101, 201, \ldots, 1701$. Tabulate the total number of function evaluations along with the accuracy you obtained for each value of num. Compare your results in this problem with the previous problem.