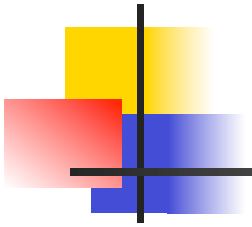
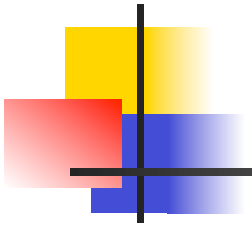


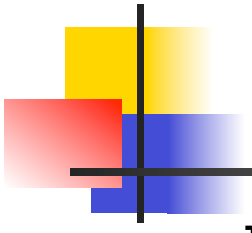
Computational Methods

Numeric Integration and Differentiation



Integration and Differentiation

- Integration and differentiation of functions can be performed in two ways
  - Symbolic integration and differentiation
    - If the function is given, the derivative and integral can often be derived as closed form functions
  - Numeric integration and differentiation
    - If a symbolic integration or differentiation is not possible, it can be solved numerically
    - If the function is unknown, numeric integration can be used to approximate the integral or derivative from a set of sample function values
- Numeric integration and differentiation techniques compute the integral or derivative value for a particular interval or point, not the functional form of the integral or derivative

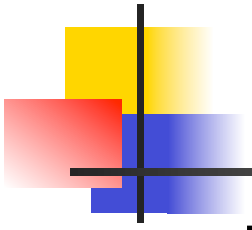


Numeric Integration and Riemann Sums

- Integration can be approximated by summation of normalized function values
  - Riemann sums approximate the integral by estimating it as the sum of the area of rectangles covering the interval with height equal to the function value at some point within the interval

$$\int_a^b f(x)dx \approx \sum_{i=1}^{n} (x_{i+1} - x_i) f(\xi_i) \quad , \quad x_i \le \xi_i \le x_{i+1}$$

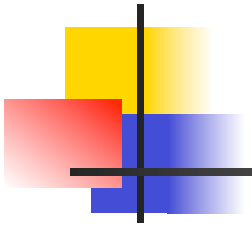
  - Riemann sums approach the true value of the integral as n approaches infinity for continuous bounded functions independent of the particular choice of ξ
- Absolute condition number of integration is equal to the width of the interval, b-a.
  - Integration is well-conditioned because it is effectively an averaging (or smoothing) operation which reduces the effect of errors.



Integration using Riemann Sums

- Riemann sums with even-spaced data points
  - The simplest form of numeric integration uses evenly spaced data points and the interval between them for the Riemann sum
    - Can use value of one of the interval's end points as value for Riemann sum
    - Can use the average of the end-point values of the interval
  - Evenly spaced points and use of sample point values for Riemann sums require a large number of samples to achieve good results
    - Does not take advantage of the characteristics of the function
    - Intractable very for high-dimensional functions
- Monte-Carlo Integration with randomly distributed points can lead to better performance in high-dimensional cases
  - Random sampling ensures convergence characteristics

$$\int_a^b f(x)dx \approx \frac{b-a}{n}\sum_{i=1}^{n} x_i$$

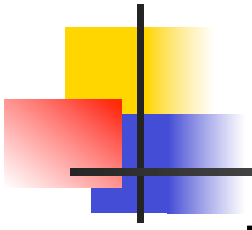


Numerical Quadrature

- Numerical quadrature is an extension of Riemann sums representing the integral as a weighted sum of a finite number of sample values

$$\int_a^b f(x)dx \approx Q_n(f) = \sum_{i=1}^{n} w_i f(x_i)$$

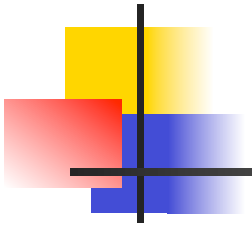
  - As opposed to Riemann sums the weight does not have to be related to the width of a sub-interval
  - Many quadrature rules are possible, determining how to choose the data points and how to compute the corresponding weights
- To achieve efficient results, numerical quadrature rules are based on polynomial interpolation
  - Takes advantage of the shape of the function by integrating the interpolating polynomial



Numerical Quadrature

- Numerical quadrature rules use samples at a finite number of points and computes the approximate integral as the integral of a polynomial interpolation function of these points
  - Rather than explicitly computing the interpolant, quadrature rules explicitly computes weights such that the the weighted sum of function values is equal to the integral of the interpolating polynomial
    - Since the integral of a weighted sum of basis polynomials is the sum of the integral of the weighted basis functions, the quadrature weights can be determined independently of the actual function values
      - Function values weigh the basis polynomial contributions in the quadrature sum
  - For Lagrange interpolation, the weights can be determined as

$$w_i = \int_a^b l_i(x)dx$$



Method of Undetermined Coefficients

- Method of undetermined coefficients uses a more general way to determine the quadrature weights
  - Consider weights as unknown parameters that have to be determined given the *n* sample locations (not their values)
  - To constrain coefficients, force them to correctly integrate the first *n* basis polynomials

$$\sum_{i=1}^{n} w_i p_k(x_i) = \int_a^b p_k(x)dx \quad , \quad 1 \le k \le n$$

    - If they correctly integrate base polynomials then they correctly integrate any sum of the base polynomials, i.e. any polynomial of degree *n-1*
    - Resulting system of equations are called moment equations

Undetermined Coefficients

- Common choice for quadrature rules use monomial basis and evenly spaced points with a sample point at each end of the integration interval
  - For n=3 results in

$$w_1 * 1 + w_2 * 1 + w_3 * 1 = \int_a^b 1 dx = b - a$$

$$w_1 * a + w_2 * (b+a)/2 + w_3 * b = \int_a^b x dx = \left(b^2 - a^2\right)/2$$

$$w_1 * a^2 + w_2 * \left((b+a)/2\right)^2 + w_3 * b^2 = \int_a^b x^2 dx = \left(b^3 - a^3\right)/3$$

  - Leading to linear system

$$\begin{pmatrix} 1 & 1 & 1 \\ a & (b+a)/2 & b \\ a^2 & \left((b+a)/2\right)^2 & b^2 \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} b-a \\ \left(b^2 - a^2\right)/2 \\ \left(b^3 - a^3\right)/3 \end{pmatrix}$$

Undetermined Coefficients

- Generally, monomial basis polynomials result in Vandermonde system for determining weights

$$\begin{pmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_n \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{n-1} & x_2^{n-1} & \cdots & x_n^{n-1} \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{pmatrix} = \begin{pmatrix} b-a \\ \left(b^2-a^2\right)/2 \\ \vdots \\ \left(b^n-a^n\right)/n \end{pmatrix}$$

  - For equally spaced points yields Newton-Cotes quadrature rules
    - Midpoint rule for one data point in the middle of the interval

$$M(f) = (b-a)\,f((a+b)/2)$$

    - Trapezoid rule for two data points at the ends of the interval

$$T(f) = (b-a)/2 * (f(a) + f(b))$$

    - Simpson's rule for three points (at the ends and in the middle)

$$S(f) = (b-a)/6 * \left(f(a) + 4f((a+b)/2) + f(b)\right)$$

Accuracy and Stability of Quadrature

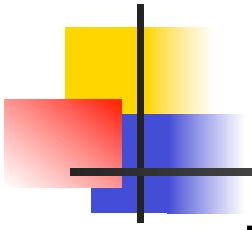
- Accuracy of the quadrature rule depends on the spacing between the data points and the degree of the polynomial that can be exactly solved for
  - From Taylor series:

$$\left|\int f - Q_n(f)\right| \le \frac{1}{4} h^{d+1} \left\| f^{(d)} \right\|_\infty \quad , \quad h = \max(x_{i+1} - x_i : i = 1,\ldots,n-1)$$

    - Can achieve higher accuracy either by reducing the spacing, h, between data points or by increasing the degree of the polynomial that can be correctly integrated
- Absolute condition number is sum of magnitude of weights

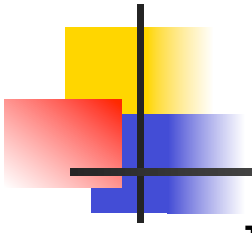
$$cond = \sum_{i=1}^{n} |w_i|$$

  - Since sum of weights is *b-a*, this is well conditioned if all weights are positive but can increase significantly if there are negative weights



Accuracy and Stability of Quadrature

- Equally spaced points for interpolation can lead to erratic behavior especially near the endpoints
  - For quadrature rules this means that weights can become negative, potentially leading to ill-conditioning
    - Newton-Cotes rules become arbitrarily ill-conditioned with increasing numbers of sample points
- Conditioning can be improved by selecting data points to form more accurate interpolations
  - Clenshaw-Curtis quadrature uses Chebyshev points
    - Weights in Clenshaw-Curtis quadrature are always positive and quadrature thus stays well-conditioned
- Accuracy can be further increased if higher degree polynomials could be integrated correctly using *n* data points

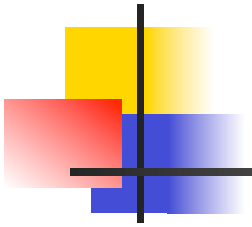


Gaussian Quadrature

- Increasing the degree of the polynomial that can be accurately integrated requires a larger number of variables in the system of equations
  - Can be achieved by increasing the number of data points
  - Can also be achieved by allowing the solution not only to pick weights but also to select the locations of the data points without increasing the number of data points
- Gaussian quadrature uses weights and data point locations as parameters, resulting in a system of non-linear equations

$$\sum_{i=1}^{n} w_i p_k(x_i) = \int_a^b p_k(x)dx \quad , \quad 1 \le k \le 2n$$

  - Accurately integrates the first *2n* base polynomials and therefore all polynomials of degree up to *2n-1*

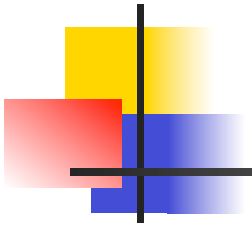


Gaussian Quadrature

- Gaussian quadrature can accurately integrate polynomials of degree *2n-1* using only *n* data points
  - Solution to system of non-linear equations is harder to derive but always contains a solution that has only positive weights
    - Data points locations might be irrational in solution
  - For a fixed integration interval the data point locations and the weights can be pre-computed as they do not depend on the function to be integrated
    - Data point locations can be scaled linearly to address a different integration interval
  - Non-linear system for 2 data points:

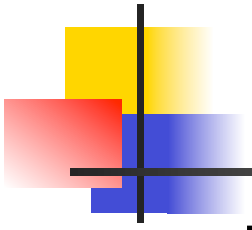
$$w_1 * 1 + w_2 * 1 = \int_{-1}^{1} 1 dx = 2 \quad , \quad w_1 * x_1 + w_2 * x_2 = \int_{-1}^{1} x dx = 0$$

$$w_1 * x_1^2 + w_2 * x_2^2 = \int_{-1}^{1} x^2 dx = 2/3 \quad , \quad w_1 * x_1^3 + w_2 * x_2^3 = \int_{-1}^{1} x^3 dx = 0$$



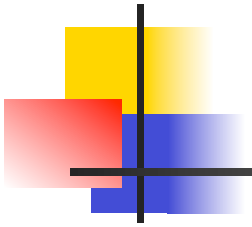
Progressive Quadrature Rules

- A sequence of quadrature rules is progressive if sample points for one *n* are a subset of the nodes for a larger *n*
  - Progressive quadrature rules allow to increase the number of data points without having to reevaluate all data points
    - Allows to increase the number sample points dynamically
    - Gaussian quadrature can accurately integrate polynomials of degree *2n-1* using only *n* data points
  - Newton-Cotes is progressive only when increasing the number of data points from *n* to *2n-1*, corresponding to subdividing each interval
  - Clenshaw-Curtis is not progressive
  - Gaussian Quadrature is not progressive
- Progressive extensions to quadrature rules can save significant amounts of work when increasing *n*



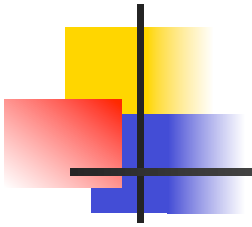
Kronrod Quadrature Rules

- Kronrad quadrature rules augment Gaussian quadrature rules by adding a 2n+1 point Kronrod rule to every n point Gaussian rule
  - Selection of Kronrod points as samples leads to a progressive quadrature rule with a higher degree than the number of data points
    - In practice these are computed by adding Kronrod rules to the Gaussian rules, imposing additional constraints
  - Gaussian/Kronrod pairs are progressive, i.e. the data points can be re-used as n is increased
  - Kronrod rules with 2n+1 data points are of degree 3n+1, thus lower than if using Gaussian rules (which would be of degree 4n+1)
    - Not as effective as Gaussian quadrature but progressive and thus more efficient if the number of data points is to be extended.



Composite Quadrature

- As in interpolation, two methods can be used to address larger numbers of data points
  - Quadrature with an increasing number of parameters, i.e. interpolation with a higher degree polynomial
  - Composite quadrature, i.e. piecewise polynomial interpolation and summing the integrals of all the polynomial pieces
- Composite quadrature has issues like piecewise interpolation
  - Composite quadrature is generally simpler to construct for large numbers of data points
  - Composite quadrature is stable if the quadrature rule for each segment is stable (equally spaced points can be used and be stable)
  - Accuracy increases as the width of each interval approaches 0
  - Not as accurate at approximating higher-order polynomials



Composite Quadrature

- Composite versions of low-degree Newton-Combs can be used effectively
  - Composite midpoint rule: 1 point per segment, k segments

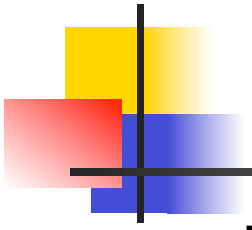
$$M_k(f) = \sum_{i=1}^{k} (x_{k+1} - x_k) f((x_k + x_{k+1})/2) = h \sum_{i=1}^{k} f((x_k + x_{k+1})/2)$$

  - Composite trapezoid rule: 2 endpoints of each segment, k segments

$$T_k(f) = \sum_{i=1}^{k} (x_{k+1} - x_k)/2 * (f(x_k) + f(x_{k+1})) =$$

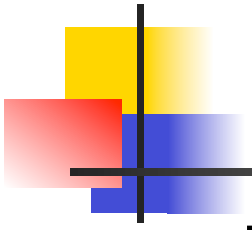
$$h * \left(1/2 (f(a) + f(b)) + \sum_{i=2}^{k} f(x_i) \right)$$

- Difference in estimate between two different levels of subdivision (e.g. k, 2k-1 for equally spaced points) can be used to estimate the accuracy of the estiamte
  - Error estimate can make composite quadrature more efficient



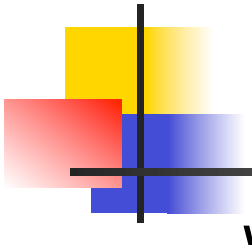
Adaptive Quadrature

- Uniform segments are very inefficient for many functions and leads to the need for a very large number of data points
  - Adaptive quadrature uses an error estimate on each segment to decide which segments to further subdivide
    - To estimate the error on a segment, two different quadrature rules are applied to each segment and their difference is used to estimate the remaining error.
    - If error estimate on segment is higher than threshold, the segment is subdivided and the procedure is repeated
  - Adaptive quadrature samples more densely in areas that are harder to integrate and more sparsely in areas that are easy to integrate
  - Quality of adaptive quadrature solution depends on the quality of the error estimate produced by the difference of the two quadrature rules
    - There are functions for which the error estimate will be low despite the actual error being high due to point selection. Thus the result can be potentially very inaccurate
    - Works very well in most situations in practice



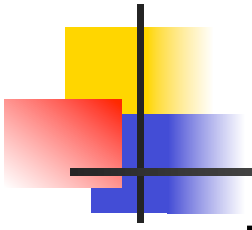
Numerical Integration

- Multiple methods exist for numeric integration
  - Simple Riemann sums are very simple for computing integrals but not very efficient in terms of the number of function evaluations needed
  - Monte Carlo integration using randomly sampled points can be efficient for high dimensional functions where using equally spaced data points in all dimensions would be too numerous to compute
  - Quadrature rules increase the efficiency of using data points by effectively computing the precise integral of the polynomial interpolation of the sample points
    - Gives better estimate of the integral by considering higher order terms
      - Newton-Cotes: degree n-1 polynomial for n data points
      - Clenshaw-Curtis: degree n-1 polynomial for n data points
      - Gaussian quadrature: degree 2n-1 polynomial for n data points
    - Composite quadrature uses piecewise inetrpolation



Numerical Differentiation

- While integration is a well-conditioned problem, differentiation is inherently sensitive
  - Functions with very similar function values will have similar integrals
  - Functions with very similar function values can have very different derivatives
- Most common numerical differentiation rules are derived from the Taylor series expansion of the function
  - Finite difference methods
  - Higher order derivatives often computed as derivative of interpolation
- Derivative of interpolation function might not be a good approximation of the derivative of the function
  - Quality of approximation depends heavily on the choice of interpolant



Finite Difference Approximations

- For smooth and differentiable functions, the Taylor series expansion provides a means of estimating the first and second derivatives from the function values at sample points
  - Forward difference approximation of the derivative:

$$f(x+h) = f(x) + hf'(x) + h^2/2\, f''(x) + \ldots$$

$$\Rightarrow \quad f'(x) = \frac{f(x+h) - f(x)}{h} - \frac{h}{2} f''(x) - \ldots \approx \frac{f(x+h) - f(x)}{h}$$

    - Provides a first order approximation with an error that is O(h) using two data points
  - Backward difference approximation of the derivative:

$$f(x-h) = f(x) - hf'(x) + h^2/2\, f''(x) - \ldots$$

$$\Rightarrow \quad f'(x) = \frac{f(x) - f(x-h)}{h} + \frac{h}{2} f''(x) - \ldots \approx \frac{f(x) - f(x-h)}{h}$$

    - Provides a first order approximation with an error that is O(h) using two data points

Finite Difference Approximations

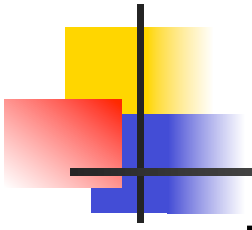
- Combining forward and backward equations improves performance
  - Centered difference approximation for derivative:

$$f(x+h) - f(x-h) = f(x) + hf'(x) + h^2/2\, f''(x) + \ldots$$
$$- f(x) + hf'(x) - h^2/2\, f''(x) + \ldots$$
$$= 2hf'(x) + 2h^3/3 f'''(x) + \ldots$$

$$\Rightarrow \quad f'(x) = \frac{f(x+h) - f(x-h)}{2h} - \frac{h^2}{6} f'''(x) - \ldots \approx \frac{f(x+h) - f(x-h)}{2h}$$

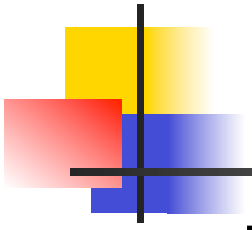
    - Provides a first order approximation with an error that is $O(h^2)$ using two data points
    - Computes lower error derivative for the point between the two data points
  - Centered difference equation for second derivative:

$$f(x+h) + f(x-h) \quad \Rightarrow \quad f''(x) \approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}$$



Richardson Extrapolation

- Difference equations increase in precision with smaller values of h
  - Small values of h can lead to large rounding errors due to the division by h and calculation of difference approximation is thus not stable for h➔0
- Richardson extrapolation computes the value of a function F(h) h➔0 by interpolating values for difference values of h given known results for two finite values of h
  - Richardson extrapolation uses information about the behavior of F(h) to obtain better estimates
    - For differentiation and integration the Taylor series expansions provides information regarding the behavior of the derivative or integral as h is changed



Richardson Extrapolation

- Richardson extrapolation for differentiation
  - Using forward approximation:

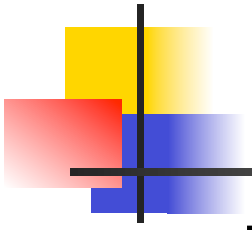
$$F_x(h) = \frac{f(x+h) - f(x)}{h} - \frac{h}{2} f''(x) - \ldots \approx a_0 + a_1 h + O(h^2)$$

    - In this polynomial, a_0 is the estimate of the derivative for h=0 and $O(h^2)$ is an estimate of the error in the extrapolation
  - Using centered approximation:

$$F_x(h) = \frac{f(x+h) - f(x-h)}{2h} - \frac{h^2}{6} f'''(x) - \ldots \approx a_0 + a_1 h^2 + O(h^4)$$

    - In this polynomial, again, a_0 is the estimate of the derivative for h=0 and $O(h^4)$ is an estimate of the error in the extrapolation
- Often Richardson extrapolation estimates F(h) using a two-term polynomial and two values for h that are multiples

$$F(h) = a_0 + a_1 h^p + O(h^r) \quad \Rightarrow \quad a_0 = F(h) + \frac{F(h) - F(h/k)}{k^{-p} - 1}$$

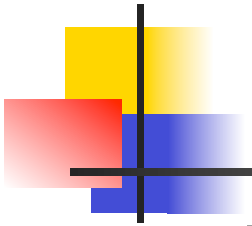


Romberg Integration

- Richardson extrapolation for composite quadrature integration with the trapezoid rule is called Romberg integration:
  - Composite quadrature is accurate as the segment width, h, approaches 0
  - Extrapolation for composite trapezoid is derived from observation that the dominant error term in trapezoid quadrature is $O(h^2)$:

$$F(h) = a_0 + a_1 h^2 + O(h^4) \quad \Rightarrow \quad a_0 = F(h) + \frac{F(h) - F(h/k)}{k^{-p} - 1}$$

    - Romberg integration estimates the behavior of the error near 0 from two subdivisions of the segments into intervals of width h and h/k
- Romberg integration can further increase the accuracy of composite quadrature without increasing the number of data points required



Numerical Integration and Differentiation

- Numerical integration is inherently well-conditioned
  - Quadrature provides an efficient means of approximating integrals
    - Quadrature weights are pre-computed reducing the cost of integration to the one of computing *n* function values for chosen sample points
  - Composite quadrature reduces the need for higher order weights
    - Less efficient in terms of data points needed but simpler in terms of weights and thus potentially more stable for high number of data points
- Numerical differentiation is inherently sensitive
  - Finite difference approximations provide estimates of derivatives
- Richardson extrapolation can further improve integration and differentiation by interpolating over different finite difference or segment lengths, *h*, and estimating the value for *h=0*