Numerical Methods I
Numerical Integration

Aleksandar Donev
Courant Institute, NYU[1]
donev@courant.nyu.edu

[1]MATH-GA 2011.003 / CSCI-GA 2945.003, Fall 2014

Dec 4th, 2014

Outline

1. Numerical Integration in 1D
   - Low-Order
   - Spectral
2. Adaptive / Refinement Methods
3. Higher Dimensions
4. Conclusions

Numerical Quadrature

- We want to numerically approximate a definite integral

$$J = \int_a^b f(x)dx.$$

- The function $f(x)$ may not have a closed-form integral, or it may itself not be in closed form.
- Recall that the integral gives the area under the curve $f(x)$, and also the **Riemann sum**:

$$\lim_{n\to\infty} \sum_{i=0}^{n} f(t_i)(x_{i+1} - x_i) = J, \text{ where } x_i \leq t_i \leq x_{i+1}$$

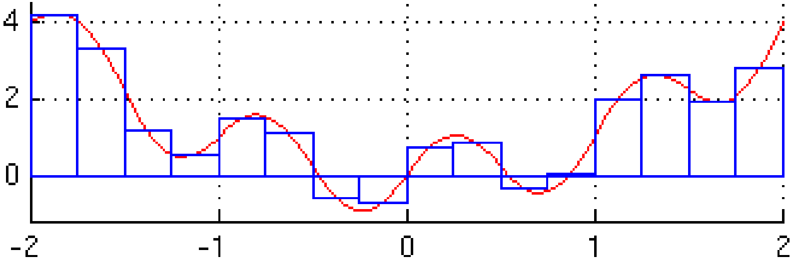
- A **quadrature formula** approximates the Riemann integral as a **discrete sum** over a set of n nodes:

$$J \approx J_n = \sum_{i=1}^{n} \alpha_i f(x_i)$$

Midpoint Quadrature

Split the interval into n intervals of width $h = (b-a)/n$ (**stepsize**), and then take as the nodes the midpoint of each interval:

$$x_k = a + (2k-1)h/2, \quad k = 1, \ldots, n$$



$$J_n = h \sum_{k=1}^{n} f(x_k), \text{ and clearly } \lim_{n\to\infty} J_n = J$$

Quadrature Error

- Focus on one of the sub intervals, and estimate the quadrature error using the midpoint rule assuming $f(x) \in C^{(2)}$:

$$\varepsilon^{(i)} = \left[\int_{x_i-h/2}^{x_i+h/2} f(x)dx\right] - hf(x_i)$$

- Expanding $f(x)$ into a Taylor series around x_i to first order,

$$f(x) = f(x_i) + f'(x_i)(x - x_i) + \frac{1}{2}f''\left[\eta(x)\right](x - x_i)^2,$$

$$\int_{x_i-h/2}^{x_i+h/2} f(x)dx = hf(x_i) + \frac{1}{2}\int_{x_i-h/2}^{x_i+h/2} f''\left[\eta(x)\right](x - x_i)^2 dx$$

- The **generalized mean value theorem for integrals** says: If $f(x) \in C^{(0)}$ and $g(x) \geq 0$,

$$\int_a^b g(x)f(x)dx = f(\eta)\int_a^b g(x)dx \text{ where } a < \eta < b$$

Composite Quadrature Error

- Taking now $g(x) = (x - x_i)^2 \geq 0$, we get the **local error** estimate

$$\varepsilon^{(i)} = \frac{1}{2} \int_{x_i - h/2}^{x_i + h/2} f'' [\eta(x)] (x - x_i)^2 dx = f'' [\xi] \frac{1}{2} \int_h (x - x_i)^2 dx = \frac{h^3}{24} f'' [\xi]$$

- Now, combining the errors from all of the intervals together gives the **global error**

$$\varepsilon = \int_a^b f(x) dx - h \sum_{k=1}^{n} f(x_k) = J - J_n = \frac{h^3}{24} \sum_{k=1}^{n} f'' [\xi_k]$$

- Use a discrete generalization of the mean value theorem to get:

$$\varepsilon = \frac{h^3}{24} n \left(f'' [\xi] \right) = \frac{b-a}{24} \cdot h^2 \cdot f'' [\xi],$$

where $a < \xi < b$.

Interpolatory Quadrature

Instead of integrating $f(x)$, integrate a polynomial interpolant $\phi(x) \approx f(x)$:

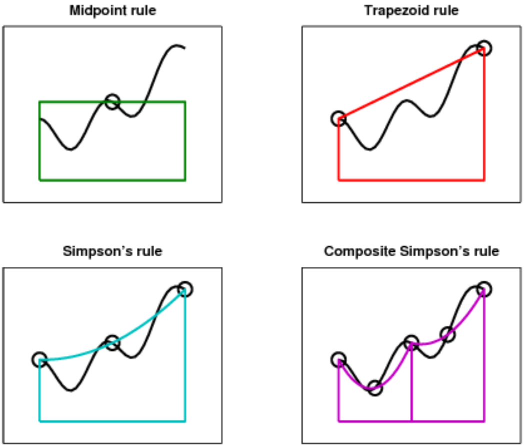


Figure 6.2. *Four quadrature rules.*

Trapezoidal Rule

- Consider integrating an **interpolating function** $\phi(x)$ which passes through $n+1$ **nodes** x_i:

$$\phi(x_i) = y_i = f(x_i) \text{ for } i = 0, 2, \dots, m.$$

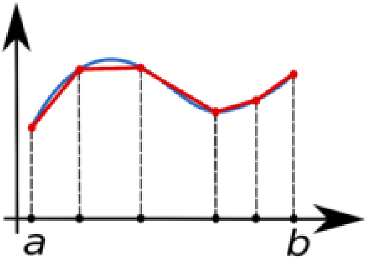
- First take the **piecewise linear interpolant** and integrate it over the sub-interval $I_i = [x_{i-1}, x_i]$:

$$\phi_i^{(1)}(x) = y_{i-1} + \frac{y_i - y_{i-1}}{x_i - x_{i-1}}(x - x_i)$$

to get the **trapezoidal formula** (the area of a trapezoid):

$$\int_{x \in I_i} \phi_i^{(1)}(x) dx = h \frac{f(x_{i-1}) + f(x_i)}{2}$$

Composite Trapezoidal Rule



- Now add the integrals over all of the sub-intervals we get the **composite trapezoidal quadrature rule**:

$$\int_a^b f(x)dx \approx \frac{h}{2}\sum_{i=1}^{n}\left[f(x_{i-1})+f(x_i)\right]$$

$$= \frac{h}{2}\left[f(x_0)+2f(x_1)+\cdots+2f(x_{n-1})+f(x_n)\right]$$

with similar error to the midpoint rule.

Simpson's Quadrature Formula

- As for the midpoint rule, split the interval into n intervals of width $h = (b - a)/n$, and then take as the nodes the endpoints and midpoint of each interval:

$$x_k = a + kh, \quad k = 0, \ldots, n$$
$$\bar{x}_k = a + (2k - 1)h/2, \quad k = 1, \ldots, n$$

- Then, take the **piecewise quadratic interpolant** $\phi_i(x)$ in the sub-interval $I_i = [x_{i-1}, x_i]$ to be the parabola passing through the nodes (x_{i-1}, y_{i-1}), (x_i, y_i), and $(\bar{x}_i, \bar{y}_i)$.
- Integrating this interpolant in each interval and summing gives the **Simpson quadrature rule**:

$$J_S = \frac{h}{6}\left[f(x_0) + 4f(\bar{x}_1) + 2f(x_1) + \cdots + 2f(x_{n-1}) + 4f(\bar{x}_n) + f(x_n)\right]$$

$$\varepsilon = J - J_s = -\frac{(b-a)}{2880} \cdot h^4 \cdot f^{(4)}(\xi).$$

Higher-Order Newton Cotes formula

- One can in principle use a higher-order polynomial interpolant on the nodes to get formally higher accuracy, e.g., using the **Lagrange** interpolant we talked about previously:

$$\phi(x) = \sum_{i=0}^{m} y_i \phi_i(x)$$

$$\phi_i(x_j) = \delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}.$$

$$\phi_i(x) = \frac{\prod_{j \neq i}(x - x_j)}{\prod_{j \neq i}(x_i - x_j)} = \frac{w_{m+1}(x)}{(x - x_i)w'_{m+1}(x_i)}$$

- Note that these only achieve higher-order accuracy for **smooth** functions.

Lagrange integration

- Integrating the interpolant gives the **Newton-Cotes quadrature**:

$$\int_a^b f(x)dx \approx \int_a^b \phi(x)dx = \int_a^b \sum_{i=0}^{m} y_i \phi_i(x)dx = h \sum_{i=0}^{m} w_i f(x_i)$$

  where it is easy to see that the **weights** w_i do not depend on $[a, b]$ and can be **pre-tabulated**:

$$w_i = \int_{-1}^{1} \phi_i(x)dx$$

- We can of course split the whole interval into sub-intervals and do the Lagrange interpolant piecewise, giving a **composite Newton-Cotes quadrature**.
- Just as for interpolation, increasing the number of equally-spaced nodes does not help much and is **not generally a good idea**.

Non-Equi Spaced Grids

- To reach higher accuracy, instead of using higher-degree polynomial interpolants (recall Runge's phenomenon), let's try using n **non-equispaced nodes**:

$$J \approx J_n = \sum_{i=1}^{n} w_i f(x_i)$$

- Question: For some choice of the weights and nodes, is it possible to compute

$$\int_a^b p_m(x)dx$$

exactly for any polynomial $p_m(x)$ of degree at most m?
- This **degree of exactness** m would guarantee that the accuracy will be $\varepsilon \sim f^{(m+1)}(\xi)$ since all the lower-order derivatives are done exactly.
- This so-called **spectral accuracy** (limited by smoothness only) cannot be achived by piecewise, i.e., local, approximations (limited by order of local approximation).

Review: Legendre Polynomials

- Recall the triangular family of **orthogonal Legendre polynomials**:

$$\begin{aligned}
\phi_0(x) =& 1 \\
\phi_1(x) =& x \\
\phi_2(x) =& \frac{1}{2}(3x^2 - 1) \\
\phi_3(x) =& \frac{1}{2}(5x^3 - 3x) \\
\phi_{k+1}(x) =& \frac{2k+1}{k+1} x \phi_k(x) - \frac{k}{k+1}\phi_{k-1}(x) = \frac{1}{2^n n!}\frac{d^n}{dx^n}\left[\left(x^2 - 1\right)^n\right]
\end{aligned}$$

- These are orthogonal on $I = [-1, 1]$:

$$\int_{-1}^{-1} \phi_i(x)\phi_j(x)dx = \delta_{ij} \cdot \frac{2}{2i+1}.$$

Orthogonal Polynomial Integration

- Recall the question we studied in Orthogonal Polynomials lecture: How to easily compute
$$\int_a^b p_{2m}(x)dx$$
exactly for a polynomial $p_{2m}(x)$ of degree at most $2m$?
- Let's first consider polynomials of degree at most m
$$\int_a^b p_m(x)dx = ?$$
- Any polynomial $p_m(x)$ of degree at most m can be expressed in the Lagrange basis:
$$p_m(x) = \sum_{i=0}^{m} p_m(x_i)\varphi_i(x)$$

Gauss Weights

- Repeating what we did for Newton-Cotes quadrature:

$$\int_a^b p_m(x)dx = \sum_{i=0}^{m} p_m(x_i)\left[\int_a^b \varphi_i(x)dx\right] = \sum_{i=0}^{m} w_i p_m(x_i),$$

where the **Gauss weights w** are given by

$$w_i = \int_a^b \varphi_i(x)dx.$$

- Recall: If we choose the **nodes to be zeros of** $\phi_{m+1}(x)$, then

$$\int_a^b p_{2m}(x)dx = \sum_{i=0}^{m} w_i p_{2m}(x_i)$$

Gauss Integration

- This gives the **Gauss quadrature** based on the **Gauss nodes and weights**, usually pre-tabulated for the standard interval $[-1, 1]$:

$$\int_a^b f(x)dx \approx \frac{b-a}{2}\sum_{i=0}^{m} w_i f(x_i).$$

- Gauss quadrature has the highest possible **degree of exactness**, i.e., it is exact for polynomials of degree up to $2n+1$.
- The low-order Gauss formulae are:

$$n = 1 : \int_{-1}^{1} f(x)dx \approx f\left(-\frac{1}{\sqrt{3}}\right) + f\left(\frac{1}{\sqrt{3}}\right)$$

$$n = 2 : \int_{-1}^{1} f(x)dx \approx \frac{5}{9}f\left(-\frac{\sqrt{15}}{5}\right) + \frac{8}{9}f(0) + \frac{5}{9}f\left(\frac{\sqrt{15}}{5}\right)$$

Asymptotic Error Expansions

- The idea in Richardson extrapolation (recall homework 4) is to use an error estimate formula to **extrapolate a more accurate answer** from less-accurate answers.
- Assume that we have a quadrature formula for which we have a theoretical error estimate:

$$J_h = \sum_{i=1}^{n} \alpha_i f(x_i) = J + \alpha h^p + O\left(h^{p+1}\right)$$

- Recall the **big O notation**: $g(x) = O(h^p)$ if:

$$\exists \, (h_0, C) > 0 \text{ s.t. } \; |g(x)| < C \, |h|^p \; \text{ whenever } \; |h| < h_0$$

- For trapezoidal formula

$$\varepsilon = \frac{b-a}{24} \cdot h^2 \cdot f''[\xi] = O\left(h^2\right).$$

Richardson Extrapolation

- Now repeat the calculation but with step size $2h$ (for equi-spaced nodes just skip the odd nodes):

$$\tilde{J}(h) = J + \alpha h^p + O\left(h^{p+1}\right)$$
$$\tilde{J}(2h) = J + \alpha 2^p h^p + O\left(h^{p+1}\right)$$

- Solve for α and obtain

$$J = \frac{2^p \tilde{J}(h) - \tilde{J}(2h)}{2^p - 1} + O\left(h^{p+1}\right),$$

which now has order of accuracy $p+1$ instead of p.

- The composite trapezoidal quadrature gives $\tilde{J}(h)$ with order of accuracy $p = 2$, $\tilde{J}(h) = J + O\left(h^2\right)$.

Romberg Quadrature

- Assume that we have evaluated $f(x)$ at $n = 2^m + 1$ equi-spaced nodes, $h = 2^{-m}(b - a)$, giving approximation $\tilde{J}(h)$.
- We can also easily compute $\tilde{J}(2h)$ by simply skipping the odd nodes. And also $\tilde{J}(4h)$, and in general, $\tilde{J}(2^q h)$, $q = 0, \ldots, m$.
- We can keep applying **Richardson extrapolation recursively** to get **Romberg's quadrature**:
  Combine $\tilde{J}(2^q h)$ and $\tilde{J}(2^{q-1} h)$ to get a better estimate. Then combine the estimates to get an even better estimates, etc.

$$J_{r,0} = \tilde{J}\left(\frac{b-a}{2^r}\right), \quad r = 0, \ldots, m$$

$$J_{r,q+1} = \frac{4^{q+1} J_{r,q} - J_{r-1,q}}{4^{q+1} - 1}, \quad q = 0, \ldots, m-1, \quad r = q+1, \ldots, m$$

- The final answer, $J_{m,m} = J + O\left(h^{2(m+1)}\right)$ is much more accurate than the starting $J_{m,0} = J + O\left(h^2\right)$, for **smooth** functions.

Adaptive (Automatic) Integration

- We would like a way to control the error of the integration, that is, specify a **target error** ε_{max} and let the algorithm figure out the correct step size h to satisfy

$$|\varepsilon| \lesssim \varepsilon_{max},$$

  where ε is an **error estimate**.
- Importantly, h may **vary adaptively** in different parts of the integration interval:
  Smaller step size when the function has larger derivatives.
- The crucial step is obtaining an error estimate: Use the same idea as in Richardson extrapolation.

Error Estimate for Simpson's Quadrature

- Assume we are using Simpson's quadrature and compute the integral $J(h)$ with step size h.
- Then also compute integrals for the left half and for the right half with step size $h/2$, $J(h/2) = J_L(h/2) + J_R(h/2)$.

$$J = J(h) - \frac{1}{2880} \cdot h^5 \cdot f^{(4)}(\xi)$$

$$J = J(h/2) - \frac{1}{2880} \cdot \frac{h^4}{32} \cdot \left[f^{(4)}(\xi_L) + f^{(4)}(\xi_R) \right].$$

- Now assume that the fourth derivative varies little over the interval, $f^{(4)}(\xi_L) \approx f^{(4)}(\xi_L) \approx f^{(4)}(\xi)$, to estimate:

$$\frac{1}{2880} \cdot h^5 \cdot f^{(4)}(\xi) \approx \frac{16}{15} \left[J(h) - J(h/2) \right]$$

$$J(h/2) - J \approx \varepsilon = \frac{1}{16} \left[J(h) - J(h/2) \right].$$

Adaptive Integration

- Now assume that we have split the integration interval $[a, b]$ into sub-intervals, and we are considering computing the integral over the sub-interval $[\alpha, \beta]$, with stepsize

$$h = \beta - \alpha.$$

- We need to compute this sub-integral with accuracy

$$|\varepsilon(\alpha, \beta)| = \frac{1}{16} \left|[J(h) - J(h/2)]\right| \leq \varepsilon \frac{h}{b-a}.$$

- An adaptive integration algorithm is $J \approx J(a, b, \epsilon)$ where the **recursive function** is:

$$J(\alpha, \beta, \epsilon) = \begin{cases} J(h/2) & \text{if } |J(h) - J(h/2)| \leq 16\varepsilon \\ J(\alpha, \frac{\alpha+\beta}{2}, \frac{\epsilon}{2}) + J(\frac{\alpha+\beta}{2}, \beta, \frac{\epsilon}{2}) & \text{otherwise} \end{cases}$$

- In practice one also stops the refinement if $h < h_{min}$ and is more conservative e.g., use 10 instead of 16.

Piecewise constant / linear basis functions

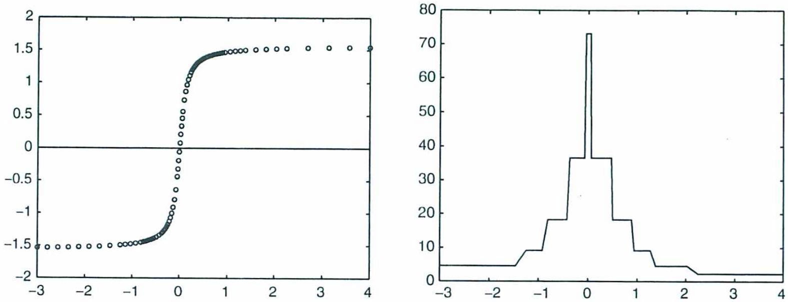


Fig. 9.4. Distribution of quadrature nodes (*left*); density of the integration stepsize in the approximation of the integral of Example 9.9 (*right*)

Regular Grids in Two Dimensions

- A **separable integral** can be done by doing integration along one axes first, then another:

$$J = \int_{x=0}^{1} \int_{y=0}^{1} f(x,y)dxdy = \int_{x=0}^{1} dx \left[\int_{y=0}^{1} f(x,y)dy \right]$$

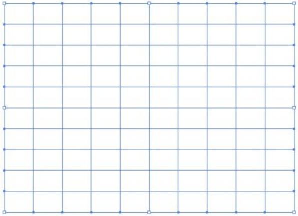
- Consider evaluating the function at nodes on a **regular grid**

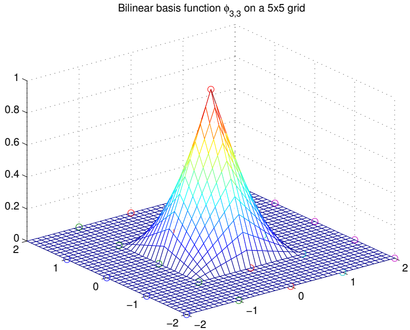
$$\mathbf{x}_{i,j} = \{x_i, y_j\}, \quad f_{i,j} = f(\mathbf{x}_{i,j}).$$

- We can use **separable basis** functions:

$$\phi_{i,j}(\mathbf{x}) = \phi_i(x)\phi_j(y).$$

Bilinear basis functions





Piecewise-Polynomial Integration

- Use a different interpolation function $\phi_{(i,j)} : \Omega_{i,j} \to \mathbb{R}$ in each rectange of the grid

$$\Omega_{i,j} = [x_i, x_{i+1}] \times [y_j, y_{j+1}],$$

and it is sufficient to look at a **unit reference rectangle** $\hat{\Omega} = [0, 1] \times [0, 1]$.

- Recall: The equivalent of piecewise linear interpolation in 1D is the **piecewise bilinear interpolation**

$$\phi_{(i,j)}(x, y) = \phi_{(i)}^{(x)}(x) \cdot \phi_{(j)}^{(y)}(y),$$

where $\phi_{(i)}^{(x)}$ and $\phi_{(j)}^{(y)}$ are linear function.

- The global interpolant can be written in the **tent-function basis**

$$\phi(x, y) = \sum_{i,j} f_{i,j}\phi_{i,j}(x, y).$$

Bilinear Integration

- The composite **two-dimensional trapezoidal quadrature** is then:

$$J \approx \int_{x=0}^{1} \int_{y=0}^{1} \phi(x,y)dxdy = \sum_{i,j} f_{i,j} \int \int \phi_{i,j}(x,y)dxdy = \sum_{i,j} w_{i,j} f_{i,j}$$

- Consider one of the corners $(0,0)$ of the reference rectangle and the corresponding basis $\hat{\phi}_{0,0}$ restricted to $\hat{\Omega}$:

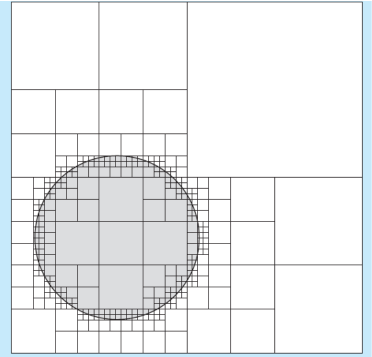
$$\hat{\phi}_{0,0}(\hat{x},\hat{y}) = (1-\hat{x})(1-\hat{y})$$

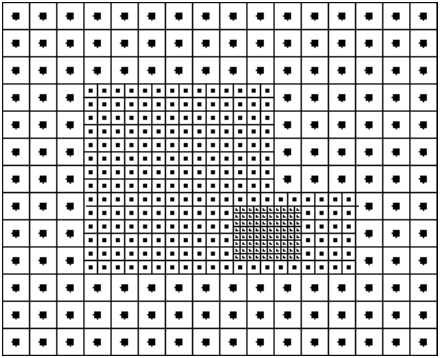
- Now integrate $\hat{\phi}_{0,0}$ over $\hat{\Omega}$:

$$\int_{\hat{\Omega}} \hat{\phi}_{0,0}(\hat{x},\hat{y})d\hat{x}d\hat{y} = \frac{1}{4}.$$

- Since each **interior node** contributes to 4 rectangles, its weight is 1. **Edge nodes** contribute to 2 rectangles, so their weight is $1/2$. **Corners** contribute to only one rectangle, so their weight is $1/4$.

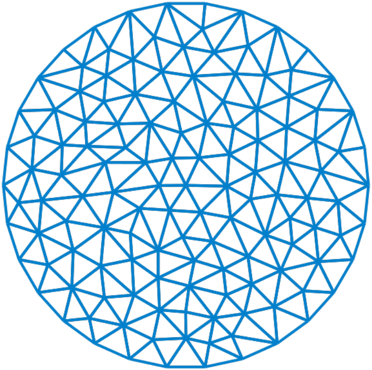
Adaptive Meshes: Quadtrees and Block-Structured

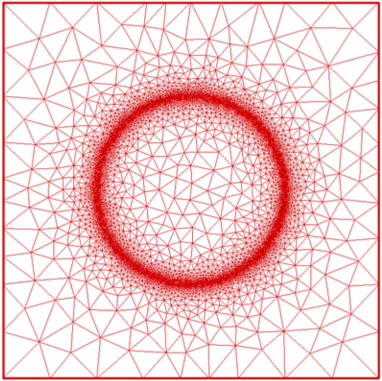




Irregular (Simplicial) Meshes

Any polygon can be triangulated into arbitrarily many **disjoint triangles**. Similarly **tetrahedral meshes** in 3D.

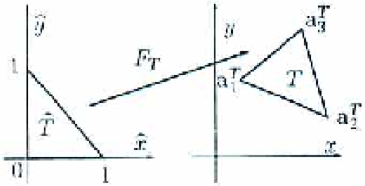




Basis functions on triangles

- For irregular grids the x and y directions are no longer separable.
- But the idea of using piecewise polynomial basis functions on a **reference triangle** $\hat{T}$ still applies.
- For a linear function we need 3 coefficients (x, y, const), for quadratic 6 $(x, y, x^2, y^2, xy, \text{const})$.
- For example, for piecewise linear we have the basis functions

$$\hat{\phi}_1(\hat{x}, \hat{y}) = 1 - (x + y) \text{ for node } (0,0)$$
$$\hat{\phi}_2(\hat{x}, \hat{y}) = x \text{ for node } (1,0).$$



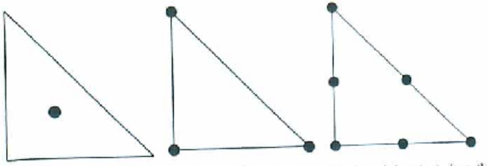


Fig. 8.8. Local interpolation nodes on $\hat{T}$ for $k = 0$ (*left*), $k = 1$ (*center*), $k = 2$ (*right*)

Piecewise constant / linear basis functions

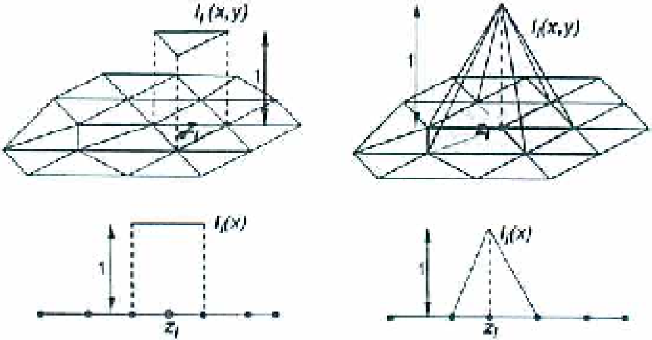


Fig. 8.7. Characteristic piecewise Lagrange polynomial, in two and one space dimensions. Left, $k = 0$; right, $k = 1$

Composite Quadrature on a Triangular Grid

- The integral over the whole grid is simply the **sum over all of the triangles**.
- So we focus on a triangle T, with d nodes, $d = 1$ for piecewise constant, $d = 3$ for piecewise linear, $d = 6$ for piecewise quadratic interpolants.

$$\int_T f(x,y)dxdy \approx \sum_{i=1}^{d} f_i \left(\int_T \phi_i^{(T)}(x,y)dxdy \right) = \sum_i w_i f_i$$

- By transforming from the **right angle reference triangle**:

$$w_i = \int_T \phi_i^{(T)}(x,y)dxdy = 2\left|T\right| \int_{\hat{T}} \hat{\phi}_i(\hat{x},\hat{y})d\hat{x}d\hat{y},$$

  where $\left|T\right|$ is the area of the triangle.
- For piecewise linear interpolant, we get $w_1 = w_2 = w_3 = \left|T\right|/3$, i.e., weight is $1/3$ for each **vertex node**.

Composite Quadrature on a Triangular Grid

- In fact, for symmetry, it may be better to think of an **equilateral reference triangle**.
- For piecewise quadratic interpolants, one obtains a quadrature that is exact for all polynomials of degree $p \leq 3$, since the integrals of cubic (odd) terms vanish by symmetry.
- The weights are: $w_v = \frac{27}{60}$ for the 1 **centroid**, $w_v = \frac{1}{20}$ for the 3 **vertices**, $w_v = \frac{2}{15}$ for the 3 **edge midpoints**.
- One can use Gauss integration over the reference triangles to get higher accuracy.

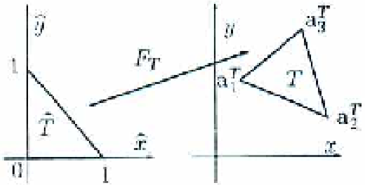


Fig. 8.8. Local interpolation nodes on $\hat{T}$ for $k = 0$ (*left*), $k = 1$ (*center*), $k = 2$ (*right*)

In MATLAB

- The MATLAB function $quad(f, a, b, \varepsilon)$ uses adaptive Simpson quadrature to compute the integral.
- The MATLAB function $quadl(f, a, b, \varepsilon)$ uses adaptive Gauss-Lobatto quadrature.
- MATLAB says: "The function *quad* may be more efficient with low accuracies or nonsmooth integrands."
- In two dimensions, for separable integrals over rectangles, use

$$J = dblquad(f, x_{min}, x_{max}, y_{min}, y_{max}, \varepsilon)$$

$$J = dblquad(f, x_{min}, x_{max}, y_{min}, y_{max}, \varepsilon, @quadl)$$

- There is also *triplequad*.

Conclusions/Summary

- Numerical integration or quadrature approximates an integral via a discrete **weighted sum** of function values over a set of **nodes**.
- Integration is based on interpolation: Integrate the interpolant to get a good approximation.
- Piecewise polynomial interpolation over **equi-spaced nodes** gives the **trapezoidal and Simpson quadratures** for lower order, and higher order are generally not recommended.
- Instead, it is better to use **Gauss integration** based on a special set of nodes and weights (orthogonal polynomials).
- In higher dimensions we split the domain into **rectangles for regular grids** (separable integration), or **triangles/tetrahedra for simplicial meshes**.
- Integration in high dimensions d becomes harder and harder because the number of nodes grows as N^d: **Curse of dimensionality**. Monte Carlo is one possible cure...