Numerical Quadrature

We now look at two related topics: quadrature and integration

quadrature: evaluation of integral of a known function over a specified domain.

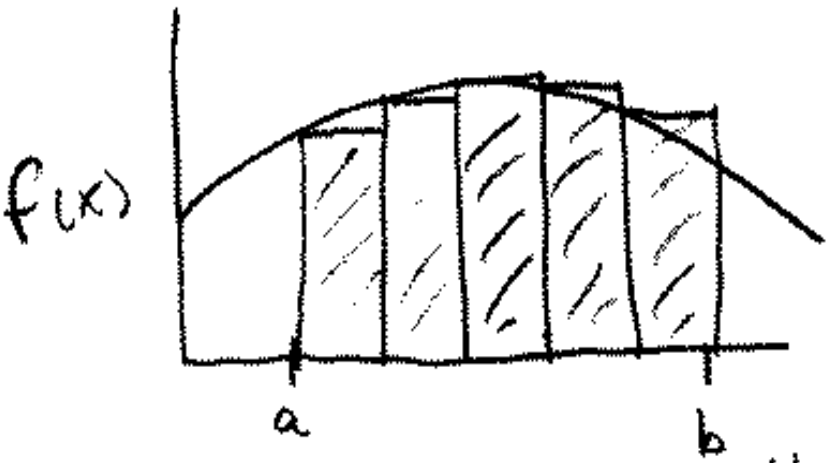
integration: integration of a set of differential equations

Let's look at quadrature first. We want to solve:

$$I = \int_a^b f(x)\,dx$$

How do we do this? We evaluate $f(x)$ at several points in the domain $[a, b]$ and combine them to estimate I

Thus:



we sum the area of the rectangles as an estimate of I

$$I = \sum_{i=1}^{n} w_i f(x_i) + R_n$$

$w_i \equiv$ weights

$x_i \equiv$ nodes

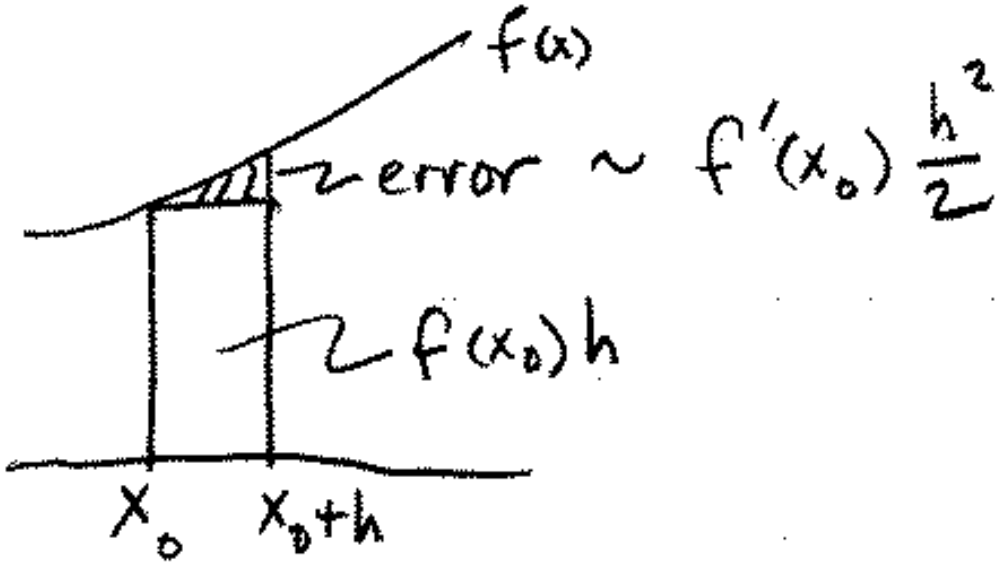
$R_n \equiv$ error in approximation

In this case, with n <u>panels</u>,

we have:

$$w_i = \frac{b-a}{n} \quad , \quad x_i = a + \frac{b-a}{n}(i-1)$$

What is R_n? We make some error in each interval:



Thus in <u>each panel</u> the error is

$$O\left(f'\frac{h^2}{2}\right)$$

The number of panels is $n = \frac{b-a}{h}$

↳ panel width

Thus the <u>total error</u> is:

$$O\left(f' h^2 \frac{b-a}{h}\right) \sim O(f' h (b-a))$$

So this rule is of order $h^{(1)}$

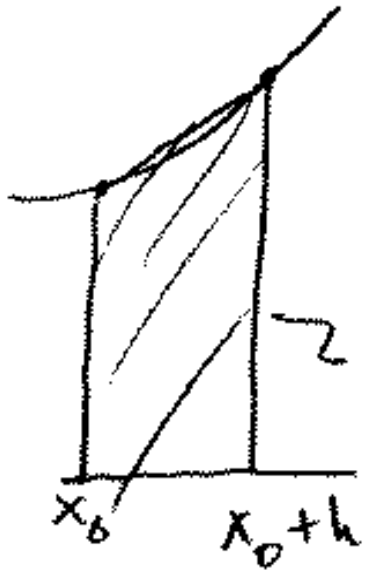
$$R_n = O\left(\frac{(b-a)^2}{n} f'\right)$$

If $f' \equiv 0$ ($f = cst$) then the rule is exact ($R_n = 0$)

We call a quadrature rule of polynomial degree ℓ if it gets all polynomials of degree ℓ exactly, but makes errors in polynomials of degree $(\ell+1)$

This rule was of degree zero

What other rules are there? How about the trapezoidal rule:



$$I \approx \frac{f(x_0)+f(x_0+h)}{2} h$$

2 trapezoid

what is the error?

We are approximating $f(x)$ with a line from $f(x_0)$ to $f(x_0+h)$. The error in this is proportional to the curvature, or f''!

$$\text{error} \sim O(h^3 f'')$$

↳ require h^3 for dimensions to work out.

Since the number of intervals is again:

$$n = \frac{b-a}{h}$$

we get for the total error:

$$R_n = O\left((b-a)h^2 f''\right) = O\left(\frac{(b-a)^3}{n^2} f''\right)$$

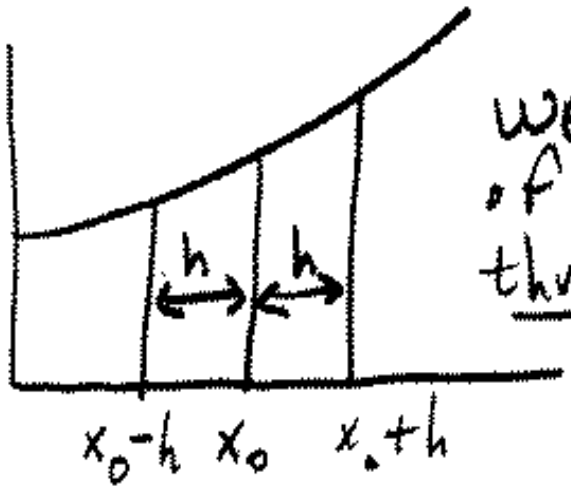
with:

$$w_i = \frac{b-a}{n} \begin{cases} \frac{1}{2} & i = 0, n \\ 1 & 1 \le i \le n-1 \end{cases}$$

$$x_i = a + \frac{(b-a)}{n} i$$

The trapezoidal rule is of polynomical degree 1 because it gets lines correct but makes errors on quadratic functions!

We can do better using Simpson's Rule:



we take a pair of panels with three function evaluations

we fit a parabola through these three points:

$$Q(x) = f(x_0-h)\,\frac{(x-x_0)(x-x_0-h)}{2h^2} + f(x_0)\,\frac{(x-x_0-h)(x-x_0+h)}{-h^2} + f(x_0+h)\,\frac{(x-x_0+h)(x-x_0)}{2h^2}$$

to get the quadrature rule

$$I \approx f(x_0 - h) w_1 + f(x_0) w_2 + f(x_0 + h) w_3$$

where $w_1 = w_3 = \frac{h}{3}$ $\quad w_2 = \frac{4h}{3}$

Thus in general:

$$I = \sum_{i=0}^{2n} w_i f\left(a + \frac{b-a}{2n} i\right) + R_{2n}$$

$$w_i = \frac{h}{3} \begin{cases} 1 & i = 0, 2n \\ 4 & i = \text{odd} \\ 2 & i = \text{even} \end{cases} \quad (\text{other than } i = 0, 2n)$$

The error is:

$$R_n = O\left(f^{IV} h^4 (b-a)\right)$$

So Simpson's rule is of polynomial degree 3

It gets cubics right even though we fit it with a parabola.

This occurs because of the symmetry of the rule.

Ok, let's try this rule out:

$$\int_0^2 x^3 \, dx = \frac{1}{4}(2^4 - 0) = \frac{16}{4} = 4$$

We use a 2 panel approximation:

$$I \approx \frac{2-0}{6}\left(f(0) + 4f(1) + f(2)\right)$$

$$= \frac{24}{6} = 4 \text{ - no error}$$

This is because Simpson's rule gets all cubics exactly! $f^{IV}(x) = 0$

Now let's try integrating x^4:

$$\int_0^2 x^4 \, dx = \frac{1}{5}(2^5 - 0) = \frac{32}{5} = 6.4$$

So:

$$I \approx \frac{2-0}{6}\left(f(0) + 4f(1) + f(2)\right) = \frac{40}{6} = 6.67$$

The error is thus $0.266\ldots = \frac{4}{15}$

Let's cut h in half:

$$\int_0^2 x^4\,dx \approx \frac{1}{6}\left(0 + 4\left(\tfrac{1}{2}\right)^4 + 2(1)^4 + 4\left(\tfrac{3}{2}\right)^4 + (2)^4\right)$$

$$= \frac{77}{12} = 6.417$$

The error is now only $0.0166\ldots = \frac{1}{60}$ which is much smaller.

Note that when we halved the interval we decreased the error by a factor of 16. This was because the error was $O(h^4)$

Now for Gaussian Quadrature

Suppose we want to integrate

$I = \int_a^b f(x)\,dx$ using only 2 pts

Where do we put them?

If we use the trapezoidal rule we put them at a & b. This is **not** the optimum choice!

Instead we let

$$I \approx w_1 f(x_1) + w_2 f(x_2)$$

and choose all four parameters such that we can integrate the highest degree polynomial possible!

We have 4 parameters, so we can integrate an arbitrary cubic polynomial with 4 constants!

Let $m = \frac{b+a}{2}$

we want to pick x_1, x_2, w_1, w_2 so that we integrate without error:

$$f(x) = 1 = (x-m)^0$$
$$f(x) = (x-m)^1$$
$$f(x) = (x-m)^2$$
$$f(x) = (x-m)^3$$

Any cubic is a linear combination of these four functions. If we get these right, we get them all right!

We get the four equations;

$$(1)\int_a^b (x-m)^0\,dx = (b-a) = w_1 + w_2$$

$$(2)\int_a^b (x-m)^1\,dx = 0 = w_1(x_1-m) + w_2(x_2-m)$$

$$(3)\int_a^b (x-m)^2\,dx = \frac{(b-a)^3}{12} = w_1(x_1-m)^2 + w_2(x_2-m)^2$$

and:

$$(4)\int_a^b (x-m)^3\,dx = 0 = w_1(x_1-m)^3 + w_2(x_2-m)^3$$

The second and fourth equations enforce symmetry:

$$w_1 = w_2\,,\quad (x_1-m) = -(x_2-m)$$

From the first equation $w_1 = w_2 = \frac{b-a}{2}$

and from the third we get:

$$x_1 = \frac{a+b}{2} - \frac{b-a}{2\sqrt{3}}, \quad x_2 = \frac{a+b}{2} + \frac{b-a}{2\sqrt{3}}$$

So we get the quadrature rule:

$$I \approx \frac{b-a}{2}\left[f\left(\frac{a+b}{2} - \frac{b-a}{2\sqrt{3}}\right) + f\left(\frac{a+b}{2} + \frac{b-a}{2\sqrt{3}}\right)\right]$$

Let's use this to integrate x^4:

$$I = \int_0^2 x^4 dx \approx \frac{2-0}{2}\left[\left(1 - \frac{1}{\sqrt{3}}\right)^4 + \left(1 + \frac{1}{\sqrt{3}}\right)^4\right]$$

$$= \frac{56}{9} = 6.222..$$

The error is just $\frac{8}{45}$ which is less than the 3pt. Simpson's rule!

In general:

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

where $R_n = \frac{(b-a)^{2n+1}(n!)^4}{(2n+1)[(2n)!]^3} f^{2n}(\xi)$

where $\xi \in [a, b]$

Provided $f(x)$ has $2n$ continuous derivatives on $[a, b]$

If $f(x)$ has singularities in itself or any of its derivatives the error can be much larger.

The Gaussian quadrature rules are of polynomial degree $2n-1$. This is because it has $2n$ adjustable parameters.

What are the properties of Gaussian Quadrature?

1) The weights & nodes are, in general, irrational numbers. The exceptions are:

$n = 1$: the midpoint rule

$w_1 = (b-a)$, $x_1 = m = \frac{b+a}{2}$

$n = 2$: $w_1 = w_2 = \frac{(b-a)}{2}$

n = odd : the midpoint m is a node

As a consequence, lists of weights and nodes are usually provided in subroutines that do the quadrature.

2) Gaussian rules are <u>open</u>: the function is never evaluated at the edges of the domain of integration. This is convenient for integrals like:

$$\int_0^1 \frac{\sin x}{x}\, dx$$

which is well behaved at $x=0$, but may lead to errors in closed rule quadrature.

3) Nodes for the n-point rule are <u>disjoint</u> from those of the m-point rule, with the exception that the midpoint is always a node for odd n, m.

Two sets are <u>disjoint</u> if they have no elements in common.

4) Among all rules using n-point evaluation the n-point Gaussian rule will produce the most accurate estimate for a smooth function.

5) The weights in gaussian quadrature are always positive. This would not be true if the nodes were evenly spaced and the weights determined optimally. Thus increasing the number of nodes always improves accuracy. For evenly spaced nodes at large n you can get large positive and negative weights leading to roundoff error.

6) The n Gauss nodes of the n-point rule are the roots of the n^{th} Legendre polynomial.

In Gaussian quadrature routines the weights and nodes are given based on the interval $[-1, 1]$. The nodes are thus symmetric:

Thus for the four point rule:

x_i^*	w_i^*
$\pm 0.8611363\ldots$	$0.3478548\ldots$
$\pm 0.3399810\ldots$	$0.6521452\ldots$

Suppose we have some general interval $[a, b]$. We can use the stored values of x_i^* and w_i^* to get the values of x_i and w_i on the general interval via a mapping:

$$x_i = \frac{a+b}{2} + \frac{b-a}{2} x_i^*$$

$$w_i = \frac{b-a}{2} w_i^*$$

and:

$$\int_a^b f(x)\,dx = \sum_{i=1}^{n} w_i f(x_i) + R_n$$

yields the desired result.

Error Estimation

It's not enough just to compute some estimate for an integral. We must also estimate how accurate the estimate is!

One approach: for the trapezoidal rule, just cut the interval in half!

Let's look at the example

$$\int_0^2 x^3\,dx = \frac{1}{4}(2)^4 = 4$$

Suppose we use n panels:

$n=2$: $I \approx \frac{1}{2}(0)^3 + 1(1)^3 + \frac{1}{2}(2)^3 = 5$

$n=4$: $I \approx \frac{1}{2}\left[\frac{1}{2}(0)^3 + \left(\frac{1}{2}\right)^3 + (1)^3 + \left(\frac{3}{2}\right)^3 + \frac{1}{2}(2)^3\right] = 4\frac{1}{4}$

$n=8$: $I \approx \frac{1}{4}[\quad\quad] = 4\frac{1}{16}$

Each time the error was decreased by a factor of 4! This is because the error was $O(\frac{1}{n^2})$.

We can actually use this to improve our result via repeated Richardson Extrapolation:

$$I = I_n + \frac{\lambda}{n^2} + O\left(\frac{1}{n^4}\right)$$

↑ next order in error term.

Note that odd powers in n are killed off due to symmetry

Suppose we took the $n=2$ and $n=4$ results:

$$I = I_2 + \frac{\lambda}{4} + O\left(\frac{1}{2^4}\right)$$

$$I = I_4 + \frac{\lambda}{16} + O\left(\frac{1}{4^4}\right)$$

If we multipy the second equation by 4 and subtract from the first we can eliminate λ:

$$3I = 4I_4 - I_2 + O\left(\frac{1}{2^4}\right)$$

$$\therefore I = \frac{1}{3}\left(4I_4 - I_2\right) + O\left(h^4\right)$$

This actually just gives us Simpson's rule back again!

We have:

$$I_2 = \frac{b-a}{2}\left(\frac{1}{2}f(a) + f\left(a + \frac{b-a}{2}\right) + \frac{1}{2}f(b)\right)$$

$$I_4 = \frac{b-a}{4}\left(\frac{1}{2}f(a) + f\left(a + \frac{b-a}{4}\right) + f\left(a + \frac{b-a}{2}\right) + f\left(a + \frac{3}{4}(b-a)\right) + \frac{1}{2}f(b)\right)$$

Let $h = \frac{b-a}{4}$

Thus:

$$\frac{1}{3}\left(4I_4 - I_2\right) = \frac{1}{3}h\left[f(a) + 4f(a+h) + 2f(a+2h) + 4f(a+3h) + f(b)\right]$$

Note that by doubling the number

of panels we only have to evaluate the function at n new points: we can use all of the points from the first evaluation over again! This can make a big difference if it takes alot of time to make a function evaluation.

If we had just increased the number of panels by one we wouldn't have gotten a significant increase in accuracy and <u>all</u> of the interior nodes would change!

We can take this combination process to a higher level. Suppose we have an n panel and $2n$ panel pair of Simpson's rule estimates. The error in each is $1/n^4$. Thus:

$$I = S_n + \frac{\lambda}{n^4} + O\left(\frac{1}{n^6}\right)$$

$$I = S_{2n} + \frac{\lambda}{(2n)^4} + O\left(\frac{1}{n^6}\right)$$

We can thus get an $O(\frac{1}{n^6})$ rule from the combination:

$$I = \frac{1}{2^4 - 1}\left(2^4 S_{2n} - S_n\right) + O\left(\frac{1}{n^6}\right)$$

This process can be repeated again and again in what is known as repeated Richardson extrapolation, or Romberg Integration, and the ultimat result is known as the Newton-Cotes formula.

Let's see how this works. Suppose we calculate an integral of $f(x)$ over $[a, b]$ using $n = 1, 2, 4, 8, 16, \ldots$ panel Trapezoidal rules. This gives us the series of estimates:

$$T_1, T_2, T_4, T_8, T_{16}, \ldots$$

Note that if we quit at $n = 16$ panels, we would only need 17 function evaluations - the nodes for the $n/2$ panel rule are included in those for the n-panel rule.

We may thus combine T_1 & T_2 to get S_2; T_2 & T_4 to get S_4, and so on.

$$S_2 = \frac{1}{2^2-1}\left(2^2 T_2 - T_1\right)$$

$$S_4 = \frac{1}{2^2-1}\left(2^2 T_2 - T_1\right)$$

The simpson's rule estimates can be combined as well. If we look at this in matrix form we get:

$$\begin{array}{llll} T_1 & & & \\ T_2 & S_2 & & \\ T_4 & S_4 & P_4 & \\ T_8 & S_8 & P_8 & Q_8 \\ T_{16} & S_{16} & P_{16} & Q_{16} \quad R_{16} \\ \vdots & \vdots & \vdots & \vdots \end{array}$$

Thus, the iteration process produces a lower triangular matrix. The most accurate integral estimate is given by the bottom-right element, and the estimate of the error is given by the difference between the last two elements of the main diagonal!

If we start with 2^K panels (a total of 2^{K+1} evaluations) then our extrapolation matrix is of size $(K+1)\times(K+1)$, and the error in the last estimate is:

$$\text{error} \sim O\left(h^{2(K+1)} f^{(2(K+1))}(\xi)\,(b-a)\right)$$

where $h = \frac{b-a}{2^K}$

If $\underset{\approx}{M}$ is our matrix, then an estimate for this error is:

$$\text{error} < \left| M_{(K+1),(K+1)} - M_{(K+1),K} \right|$$

How about error estimation in Gaussian quadrature? Because the nodes are <u>disjoint</u>, doubling the number of points does not allow you to reuse <u>any</u>! A way around this was suggested by Kronod.

Use the <u>same</u> n points over again, <u>but</u> add $n+1$ new ones and $2n+1$ new weights, chosen optimally!

Even if $f(x)$ is non-singular, it's derivatives may be! To integrate these functions we require adaptive quadrature.

Basically, adaptive quadrature is just a way of making the discretization finer where a polynomial approximation is poor.

An adaptive quadrature algorithm has two parts

1) A local quadrature module which computes the integral over the domain $[\alpha, \beta]$ and also provides an error estimate.

2) A control strategy which decides what portion of the interval to subdivide.

This algorithm works as follows:

1) start by feeding $[a, b]$ to the LQM. If the error $< \varepsilon$ (desired tolerance) then quit. This will probably happen for smooth functions!

2) If the total error $> \varepsilon$, divide the interval with the largest contribution to the error in 2 pieces, and send both to the LQM.

3) Add up the error. If it's less than ε then quit, if it's greater go to (2).

Usually such an algorithm will have a stop built in to avoid too fine a level of discretization.

Matlab offers two adaptive quadrature functions, 'quad' and 'quad8'. The first uses Simpson's rule for the LQM, the second uses an 8-panel Newton-Cotes rule. Both are <u>closed</u> rules, so all singularities <u>must</u> be removed analytically.

In general, quadrature algorithms have trouble with steeply varying functions. For example, suppose we want to calculate the integral:

$$I = \frac{1}{\sqrt{2\pi}} \int_{-N}^{N} t^2 e^{-t^2/2} dt$$

which is (as $N \to \infty$) the variance of the normal (Gaussian) distribution.

If we plug in $N = 10$ we can run into trouble! The function looks like:

Using 'quad' on this function gives you about 10 pages of error messages! (It does get the right answer $I = 1$ eventually, though)

There is no substitute for knowing the answer before you start!

Often we want to integrate over infinite & semi-infinite domains. How do we deal with this?

Example:
$$I = \int_0^\infty e^{-x} \cos^2 x^2 \, dx$$

One way is to simply truncate at some large number A:

$$\text{Say } I \approx Q = \int_0^A e^{-x} \cos^2 x^2 \, dx$$

This produces an error:
$$I - Q = \int_A^\infty e^{-x} \cos^2 x^2 \, dx < \int_A^\infty e^{-x} \, dx = e^{-A}$$

which is exponentially small!

Unfortunately, this is not usually true.

In general $I - Q = \int_A^\infty f(x) \, dx$

if $f(x) \cong x^{-3/2}$ at large x then

$$\int_A^\infty x^{-3/2} dx = -\frac{1}{2} A^{-1/2}$$

Thus for $A = 10^4$, error is still $\mathcal{O}(10^{-2})$!

A better way is to transform the integral by mapping the domain.

Let $x = p(t)$

If we pick $p(t) = -\ln t$ or $\frac{t}{1-t}$

we map the domain $[0, \infty]$ onto $[0, 1]$

e.g. if $x = -\ln t$

then: $$\int_0^\infty f(x)\,dx = -\int_1^0 f(-\ln t)\,\frac{dt}{t}$$

$$= \int_0^1 f(-\ln t)\,\frac{dt}{t}$$

This can work very well:

if $f(x) = e^{-x}$ then:

$$\int_0^\infty e^{-x}\,dx = \int_0^1 e^{\ln t}\,\frac{dt}{t} = \int_0^1 dt = 1!$$

or rather poorly:

if $f(x) = x^{-2}$

$$\int_0^\infty x^{-2}\,dx = \int_0^1 \frac{1}{(\ln t)^2}\,\frac{1}{t}\,dt$$

which has an integrable singularity at the origin! In general, the choice of mapping depends on the function - you want to choose a mapping that captures the asymptotic behavior of the function!

You can also combine mapping with truncation: map the integrand so that it is exponentially small in t, and then truncate it.

So far we have just looked at one-dimensional quadrature. Often we need to integrate over 2 or more dimensions, e.g.

$$I = \iint_D f(x,y)\,dA$$

Say, calculate the lift on a wing by integrating the pressure over the wing surface!

We can write the n-point 2-D quadrature rule as:

$$I = \sum_{i=1}^{n} w_i f(x_i, y_i) + R_n$$

This formula is of polynomial degree d if $R_n = 0$ for any bivariate polynomial of degree d but non-zero for some polynomial of degree $d+1$

What is a bivariate polynomial?

It is a linear combination of terms

$$x^p y^q \quad \text{where} \quad p+q \leq 2$$

For example, the most general polynomial of bivariate degree 2 is:

$$a x^2 + b y^2 + c x y + d x + e y + f = 0$$

There are 6 parameters, thus to integrate this exactly we require at least 3 nodes:

$$I = w_1 f(x_1, y_1) + w_2 f(x_2, y_2)$$

For each node we have 3 adjustable parameters: x_i, y_i, and w_i

The problem is that the location and weights are dependent on the domain shape!

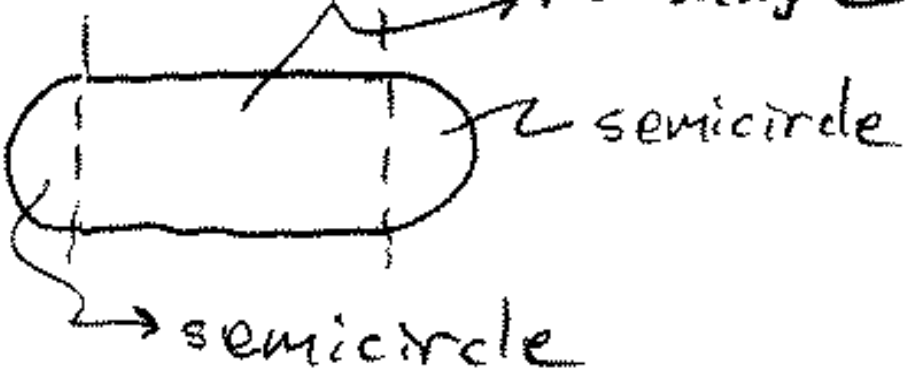
Optimum node locations and weights have been derived for rectangles, circles, and triangles.

What happens for more complex shapes?

There are a number of choices:

1) Map onto a simple domain via a change of variables. For example, a simple stretching maps an ellipse onto a circle, and polar coords. is a map of a circle onto a rectangle! Complex variables, through a technique called <u>conformal mapping</u> provides many useful mappings.

2) Break the domain of integration into a union of simpler shapes if possible. Example:



Solve for the integral on each of these bits separately

3) Approximate the domain with a union of triangles or other shapes:

error due to approximation.

The error gets smaller as the triangles are smaller (finer discretization)

4) Imbed the domain in a rectangle or other shape and let $f(x,y)=0$ outside the domain. This is very simple to do, but leads to a discontinuity in $f(x,y)$ - and hence to quadrature errors

---

In general, mapping is the best approach - if you can do it! As always, there is a tradeoff between the difficulty in setting the problem up (e.g. a complex mapping) and the amount of computer work required (e.g., dealing with discontinuities).

Suppose the domain is a rectangle (or a rectangular panel in a more complex domain)

we wish to solve:

$$\int_a^b \int_\alpha^\beta f(x,y)\,dx\,dy$$

Let $\int_\alpha^\beta f(x,y)\,dx = \sum_{i=1}^{n} w_i f(x_i,y) + R_1(y)$

and $\int_a^b g(y)\,dy = \sum_{j=1}^{n} P_j\, g(y_j) + R_2$

Thus:

$$\int_a^b \int_\alpha^\beta f(x,y)\,dx\,dy = \int_a^b \left[\sum_{i=1}^{n} w_i f(x_i,y) + R_1(y)\right] dx$$

$$= \sum_{i=1}^{n} w_i \int_a^b f(x_i,y)\,dy + \int_a^b R_1(y)\,dy$$

Let $f(x_i,y) = g_i(y)$

Thus:

$$I = \sum_{i=1}^{n} w_i \left[\sum_{j=1}^{m} P_j f(x_i, y_j) + R_2 \right] + \int_a^b R_1 \, dy$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{m} w_i P_j f(x_i, y_j) + \underbrace{\sum_{i=1}^{n} w_i R_2 + \int_a^b R_1 \, dy}_{R \text{ (total error)}}$$

This is a product formula: essentially we evaluate the integral over one variable first and then over the other. n & m are, in general, different.

If the two one-dim. rules are of degree $d_1 = 2n-1$, $d_2 = 2m-1$ (e.g., we use Gaussian quadrature) then the product rule will be exact for polynomials of the form:

$$x^p y^q : \quad p \le d_1, \; q \le d_2$$

The product rule is exact for a bivariate polynomial of degree $\min(d_1, d_2)$

but it will be exact for some polynomials of degree $\ell_1 + \ell_2$.

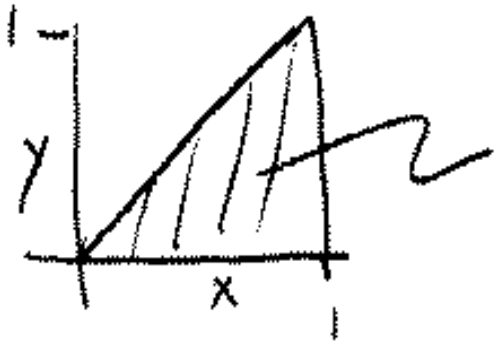
While this type of product rule is simple to code, it may not be the best choice. A 3x3 product rule using Gaussian quadrature is of bivariate degree 5 with 9 nodes.

There exists some rule of degree 5 with only 7 nodes! It's probably not worth trying to obtain it, however!

Mappings:

The most effective technique for doing multi-dim quadrature is mapping. Usually we want to map a function onto a rectangle. This may do strange things.

Example: map a triangle onto a rectangle:



$0 \le x \le 1$

$0 \le y \le x$

We can map this onto a rectangle:

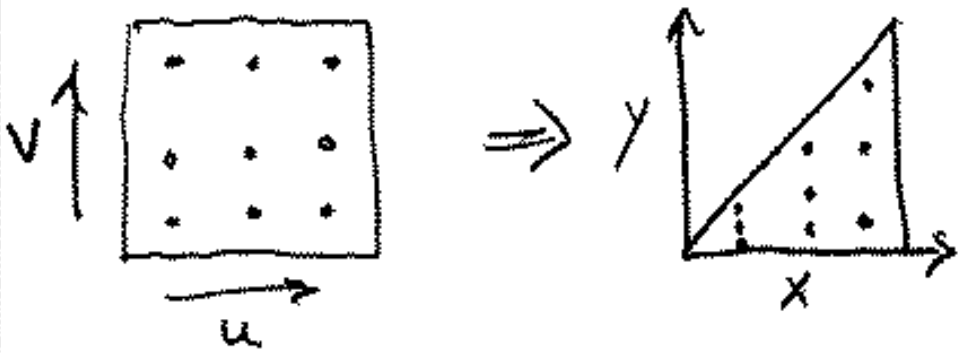
$u = x \quad , \quad v = y/x$

So that:

$$I \equiv \int_0^1 \int_0^x f(x,y)\, dy\, dx = \int_0^1 \int_0^1 f(u, uv)\, u\, du\, dv$$

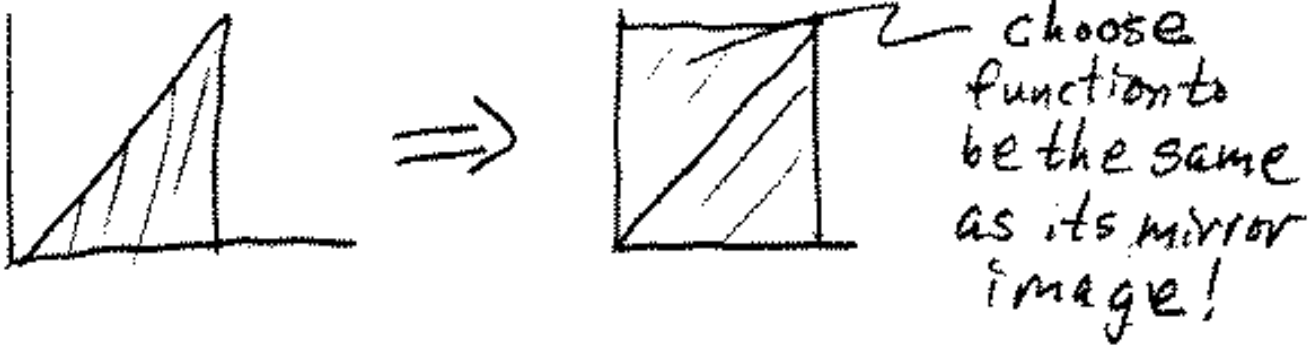
Note that $dy\,dx$ is transformed to $u\,du\,dv$!

This sort of transformation may not be desirable, because it will place the quadrature nodes unevenly in the original space:



Thus the function may be undersampled in some regions and oversampled in others. You should take this (and the spatial behavior of the function) into account when deciding on a mapping!

As an alternative, we could have <u>continued</u> the function by <u>reflection</u>



Thus we pick $f(x,y)\Big|_{y>x} = f(y,x)$

and evaluate the integral over $0<x<1$ and $0<y<1$. The original integral will be half of this value.

While this leaves the points evenly distributed, it introduces a discontinuity in the derivative on the line $y=x$.

How can we apply adaptive quadrature to a 2-D integral?

If we have mapped to a rectangle, this is very simple:

We first evaluate over one variable, and then (in an outer loop or program) evaluate over the other!

Thus:

$$I = \int_a^b \left[\int_\alpha^\beta f(x,y)\,dy \right] dx = \int_a^b g(x)\,dx$$

where $$g(x) = \int_\alpha^\beta f(x,y)\,dy$$

Suppose the quadrature routine over y returns:

$$g(x) = g^*(x) + E_y(x)$$

where $E_y(x)$ is the error and g^* is the numerical result.

In the second integral (over x) we are not integrating $g(x)$, but rather the numerical result $g^*(x)$:

$$\int_a^b g^*(x)\,dx = \int_a^b [\, g(x) - E_y(x) \,]\,dx$$

and this will have some error as well!

Thus the combined result returns:

$$I = \int_a^b g(x)\,dx = I^* + \int_a^b E_y(x)\,dx + E_x$$

Thus both the integral over y <u>and</u> x contribute to the error.

We require (in adaptive quadrature) that the total error be less than some threshold ε:

$$\left| \int_a^b E_y(x)\,dx + E_x \right| < \varepsilon$$

This is accomplished by setting individual tolerances for both the x and y quadratures. Thus:

$$|E_y| < \varepsilon_y \,, \quad |E_x| < \varepsilon_x$$

where:

$$|(b-a)\varepsilon_y + \varepsilon_x| < \varepsilon$$

In general we require the inner tolerance to be smaller than the outer tolerance, say:

$$(b-a)\varepsilon_y \approx 0.1\,\varepsilon, \quad \varepsilon_x \approx 0.9\,\varepsilon$$

Feeding these values to the adaptive quadrature routines will result in the desired accuracy.

Monte Carlo Integration

So far all quadrature methods we have looked at are polynomial based - we are approximating a function locally with a high degree polynomial. Monte Carlo integration is completely different.

Suppose we have:

$$I = \int_a^b f(x)\,dx \equiv (b-a)\langle f(x)\rangle$$

where $\langle f(x)\rangle$ is the average of f over $[a, b]$.

The integral is thus just the average of the function times the width of the interval.

In Monte Carlo integration we want to compute this average!

We just pick N points chosen at random over the domain and then average the corresponding function values.

$$I \approx \theta_N \equiv (b-a)\frac{1}{N}\sum_{i=1}^{N} f(x_i)$$

As N goes to infinity, the error in θ_N will go to zero.

What is the error? For large N, θ_N approximates a normal distribution.

Thus:

$$E(\theta_N) = I$$

$$\sigma^2_{\theta_N} = E\left\{ \frac{(b-a)^2}{N} \frac{1}{N}\sum_{i=1}^{N} (f(x_i) - \bar{f})^2 \right\}$$

$$= \frac{(b-a)^2}{N}\sigma_f^2$$

where σ_f^2 is the variance of $f(x)$ over the interval.

The problem with this method is that it converges <u>very</u> slowly with N.

Even the Trapezoidal rule has an error which goes as $\frac{1}{N^2}$, while Monte Carlo integration error goes as $\sim \frac{1}{N^{1/2}}$!

To get 3 significant digits, you need around 10^6 points.

The primary advantage of this procedure is in multi-dimensional integrals:

$$I = \int_D f(\underset{\sim}{x})\, dV$$

where $\underset{\sim}{x}$ is an m-dimensional vector. If m is large, this is difficult to code using polynomial based quadrature. Also, if (say) $m = 12$ and we use just 3 point quadrature in each dimension, we require over 500,000 points! With this number of points Monte Carlo integration may well be more accurate!

The most important use is in simulation of random media and related topics.

You choose a number of random configurations of particles distributed in a matrix (say) and calculate effective properties such as thermal conductivity or dielectric constant for each configuration. You then just average the values together!

Monte Carlo integration was originally developed by von Neumann to solve the neutron diffusion problem important in designing the A-bomb. We use similar approaches for measuring shear-induced diffusion in suspensions.

Prof. Maginn uses these techniques to examine shape/size selectivity of zeolite catalysts.