CMPSC/Math 451, Numerical Computation

Wen Shen

Department of Mathematics, Penn State University

Numerical integration: Introduction

Problem Description:
Given a function $f(x)$, defined on an interval $[a, b]$, we want to find an approximation to the integral

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

Main ideas:

- Cut up $[a, b]$ into smaller sub-intervals;
- In each sub-interval, find a polynomial $p_i(x) \approx f(x)$;
- Integrate $p_i(x)$ on each sub-interval, and sum them up.

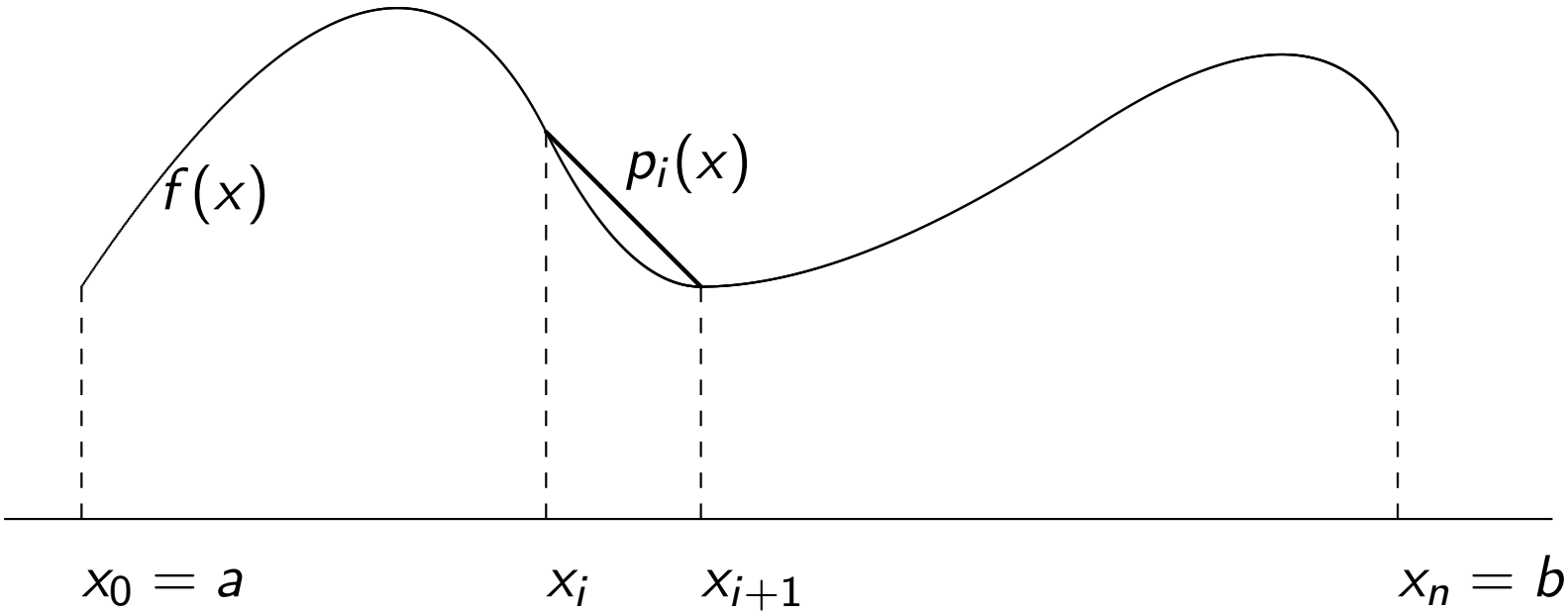
Trapezoid rule

The grid: We cut up $[a, b]$ into n sub-intervals:

$$x_0 = a, \quad x_i < x_{i+1}, \quad x_n = b$$

On $[x_i, x_{i+1}]$, approximate $f(x)$ by a linear polynomial p_i:

$$p_i(x_i) = f(x_i), \qquad p_i(x_{i+1}) = f(x_{i+1}).$$



On each sub-interval, the integral of p_i equals to the area of a trapezium:

$$\int_{x_i}^{x_{i+1}} p_i(x)\,dx = \frac{1}{2}(f(x_i) + f(x_{i+1}))(x_{i+1} - x_i).$$

Now, we use

$$\int_{x_i}^{x_{i+1}} f(x)\,dx \approx \int_{x_i}^{x_{i+1}} p_i(x)\,dx = \frac{1}{2}\left(f(x_{i+1}) + f(x_i)\right)(x_{i+1} - x_i)\ ,$$

and we sum up all the sub-intervals

$$\begin{aligned}\int_a^b f(x)\,dx &= \sum_{i=0}^{n-1}\int_{x_i}^{x_{i+1}} f(x)\,dx \approx \sum_{i=0}^{n-1}\int_{x_i}^{x_{i+1}} p_i(x)\,dx \\ &= \sum_{i=0}^{n-1}\frac{1}{2}\left(f(x_{i+1}) + f(x_i)\right)(x_{i+1} - x_i)\end{aligned}$$

Uniform Grid

$$h = \frac{b-a}{n}, \qquad x_{i+1} - x_i = h.$$

$$\begin{aligned}
\int_a^b f(x)\,dx &\approx \sum_{i=0}^{n-1} \frac{h}{2}\left(f(x_i) + f(x_{i+1})\right) \\
&= \frac{h}{2}\left[(f(x_0) + f(x_1)) + (f(x_1) + f(x_2)) + \cdots + (f(x_{n-1}) + f(x_n))\right] \\
&= \frac{h}{2}\left[f(x_0) + 2\sum_{i=1}^{n-1} f(x_i) + f(x_n)\right] \\
&= \underbrace{h\left[\frac{1}{2}(f(x_0) + f(x_n)) + \sum_{i=1}^{n-1} f(x_i)\right]}_{T(f;h)}
\end{aligned}$$

Trapezoid Rule : $$\int_a^b f(x)\,dx \approx h\left[\frac{1}{2}f(x_0) + \sum_{i=1}^{n-1} f(x_i) + \frac{1}{2}f(x_n)\right]$$

Example 1: Let $f(x) = \sqrt{x^2+1}$, compute $I = \int_{-1}^{1} f(x)\,dx$ by Trapezoid rule with $n = 10$.

Answer. We can set up the data

i	x_i	f_i
0	-1	1.4142136
1	-0.8	1.2806248
2	-0.6	1.1661904
3	-0.4	1.077033
4	-0.2	1.0198039
5	0	1.0
6	0.2	1.0198039
7	0.4	1.077033
8	0.6	1.1661904
9	0.8	1.2806248
10	1	1.4142136

Here $h = 2/10 = 0.2$.

By the formula, we get

$$T = h\left[(f_0 + f_{10})/2 + \sum_{i=1}^{9} f_i\right] = 2.3003035.$$

Sample codes for Trapezoid Rule in Matlab.

Let $f(x) = x^2 + \sin(x)$. It cam be defined in file "`func.m`" as:

```
function v=func(x)
    v=x.^2 + sin(x);
end
```

In the following script, the integral value is stored in the variable '`T`'.

```
h=(b-a)/n;  x=[a:h:b];
T = (func(a)+func(b))/2;
for i=2:1:n,  T = T + func(x(i)); end
T = T*h;
```

Or, one may use directly the Matlab vector function '`sum`', and the code could be very short:

```
h=(b-a)/n;
x=[a+h:h:b-h]; % inner points
T = ((func(a)+func(b))/2 + sum(func(x)))*h;
```

Error estimates for Trapezoid rule.

We define the error:

$$E_T(f;h) \doteq I(f) - T(f;h) = \sum_{i=0}^{n-1} \int_{x_i}^{x_{i+1}} [f(x) - p_i(x)]\, dx = \sum_{i=0}^{n-1} E_{T,i}(f;h),$$

where $E_{T,i}(f;h)$ is the error on each sub-interval

$$E_{T,i}(f;h) = \int_{x_i}^{x_{i+1}} [f(x) - p_i(x)]\, dx, \qquad (i = 0, 1, \cdots, n-1)$$

Error bound with polynomial interpolation:

$$f(x) - p_i(x) = \frac{1}{2} f''(\xi_i)(x - x_i)(x - x_{i+1}), \qquad (x_i < \xi_i < x_{i+1})$$

Error estimate on each sub-interval:

$$E_{T,i}(f;h) \;=\; \frac{1}{2} f''(\xi_i) \int_{x_i}^{x_{i+1}} (x - x_i)(x - x_{i+1})\, dx \;=\; -\frac{1}{12} h^3 f''(\xi_i).$$

(You may work out the details of the integral!)

The total error is:

$$\begin{aligned} E_T(f;h) &= \sum_{i=0}^{n-1} E_{T,i}(f;h) = \sum_{i=0}^{n-1} -\frac{1}{12}h^3 f''(\xi_i) \\ &= -\frac{1}{12}h^3 \underbrace{\left[\sum_{i=0}^{n-1} f''(\xi_i)\right]}_{=f''(\xi)} \cdot \frac{1}{n} \cdot \underbrace{\frac{b-a}{h}}_{=n} \end{aligned}$$

which gives

$$E_T(f;h) = -\frac{b-a}{12}h^2 f''(\xi), \qquad \xi \in (a,b).$$

Error bound

$$|E_T(f;h)| \leq \frac{b-a}{12}h^2 \max_{x\in(a,b)} \left|f''(x)\right| .$$

Example 2. Consider function $f(x) = e^x$, and the integral $I(f) = \int_0^2 e^x \, dx$. What is the minimum number of points to be used in the trapezoid rule to ensure en error $\leq 0.5 \times 10^{-4}$?

Answer. We have

$$f'(x) = e^x, \quad f''(x) = e^x, \quad a = 0, \quad b = 2, \qquad \max_{x \in (a,b)} \left| f''(x) \right| = e^2.$$

By error bound, it is sufficient to require

$$\begin{aligned} & |E_T(f;h)| \leq \frac{1}{6} h^2 e^2 \leq 0.5 \times 10^{-4} \\ \Rightarrow \quad & h^2 \leq 0.5 \times 10^{-4} \times 6 \times e^{-2} \approx 4.06 \times 10^{-5} \\ \Rightarrow \quad & \frac{2}{n} = h \leq \sqrt{4.06 \times 10^{-5}} = 0.0064 \\ \Rightarrow \quad & n \geq \frac{2}{0.0064} \approx 313.8 \end{aligned}$$

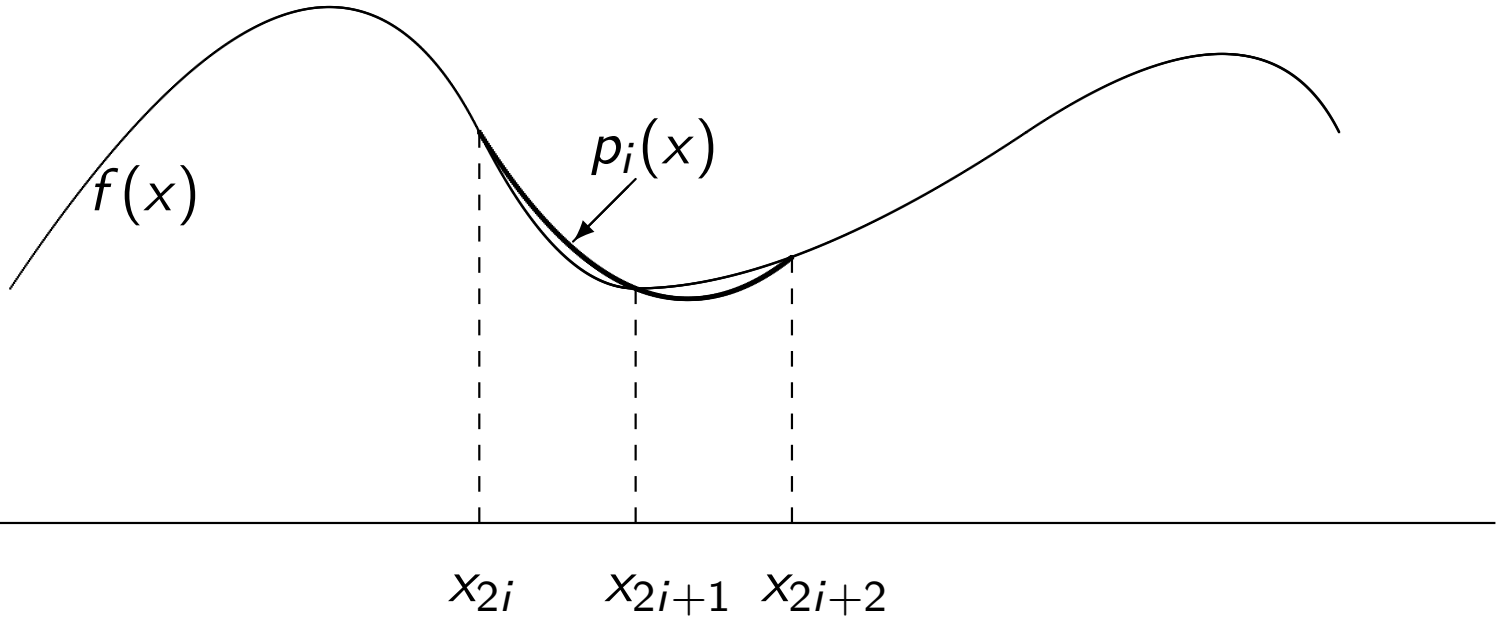
We need at least 314 points.

Simpson's rule

We now explorer possibility of using higher order polynomials.
We cut up $[a, b]$ into $2n$ equal sub-intervals

$$x_0 = a, \quad x_{2n} = b, \quad h = \frac{b-a}{2n}, \quad x_{i+1} - x_i = h$$

On $[x_{2i}, x_{2i+2}]$, interpolates $f(x)$ at the points $x_{2i}, x_{2i+1}, x_{2i+2}$ with a quadratic polynomial $p_i(x)$.



Note that in each sub-interval there is a point in the interior.

Lagrange form for $p_i(x)$:

$$\begin{aligned} p_i(x) &= f(x_{2i})\frac{(x-x_{2i+1})(x-x_{2i+2})}{(x_{2i}-x_{2i+1})(x_{2i}-x_{2i+2})} + f(x_{2i+1})\frac{(x-x_{2i})(x-x_{2i+2})}{(x_{2i+1}-x_{2i})(x_{2i+1}-x_{2i+2})} \\ &\quad + f(x_{2i+2})\frac{(x-x_{2i})(x-x_{2i+1})}{(x_{2i+2}-x_{2i})(x_{2i+2}-x_{2i+1})} \end{aligned}$$

With uniform nodes, this becomes

$$\begin{aligned} p_i(x) &= \frac{1}{2h^2}f(x_{2i})(x-x_{2i+1})(x-x_{2i+2}) - \frac{1}{h^2}f(x_{2i+1})(x-x_{2i})(x-x_{2i+2}) \\ &\quad + \frac{1}{2h^2}f(x_{2i+2})(x-x_{2i})(x-x_{2i+1}) \end{aligned}$$

We work out the integrals (try to fill in the details yourself!)

$$
\begin{aligned}
I_1 &= \int_{x_{2i}}^{x_{2i+2}} (x - x_{2i+1})(x - x_{2i+2})\, dx = \frac{2}{3}h^3, \\
I_2 &= \int_{x_{2i}}^{x_{2i+2}} -(x - x_{2i})(x - x_{2i+2})\, dx = \frac{4}{3}h^3, \\
I_3 &= \int_{x_{2i}}^{x_{2i+2}} (x - x_{2i})(x - x_{2i+1})\, dx = \frac{2}{3}h^3,
\end{aligned}
$$

Then

$$
\begin{aligned}
\int_{x_{2i}}^{x_{2i+2}} p_i(x)\, dx &= \frac{1}{2h^2} f(x_{2i}) \cdot I_1 + \frac{1}{h^2} f(x_{2i+1}) \cdot I_2 + \frac{1}{2h^2} f(x_{2i+2}) \cdot I_3 \\
&= \frac{1}{2h^2} f(x_{2i}) \frac{2}{3}h^3 + \frac{1}{h^2} f(x_{2i+1}) \frac{4}{3}h^3 + \frac{1}{2h^2} f(x_{2i+2}) \frac{2}{3}h^3 \\
&= \frac{h}{3} \left[f(x_{2i}) + 4f(x_{2i+1}) + f(x_{2i+2}) \right] .
\end{aligned}
$$

We now sum them up

$$\int_a^b f(x)\,dx \approx S(f;h) = \sum_{i=0}^{n-1} \int_{x_{2i}}^{x_{2i+2}} p_i(x)\,dx$$

$$= \frac{h}{3} \sum_{i=0}^{n-1} \left[f(x_{2i}) + 4f(x_{2i+1}) + f(x_{2i+2}) \right].$$

1	4	$\boxed{\begin{matrix}1\\1\end{matrix}} = 2$	4	1
x_{2i-2}	x_{2i-1}	x_{2i}	x_{2i+1}	x_{2i+2}

Simpson's Rule:

$$S(f;h) = \frac{h}{3} \left[f(x_0) + 4 \sum_{i=1}^{n} f(x_{2i-1}) + 2 \sum_{i=1}^{n-1} f(x_{2i}) + f(x_{2n}) \right]$$

Simpson's Rule: $$S(f;h) = \frac{h}{3}\left[f(x_0) + 4\sum_{i=1}^{n} f(x_{2i-1}) + 2\sum_{i=1}^{n-1} f(x_{2i}) + f(x_{2n})\right]$$

Example 3: Let $f(x) = \sqrt{x^2+1}$, and compute $I = \int_{-1}^{1} f(x)\,dx$ by Simpson's rule with $n = 5$, i.e. with $2n+1 = 11$ points.

Answer. We can set up the data (same as in Example 1):

i	x_i	f_i
0	-1	1.4142136
1	-0.8	1.2806248
2	-0.6	1.1661904
3	-0.4	1.077033
4	-0.2	1.0198039
5	0	1.0
6	0.2	1.0198039
7	0.4	1.077033
8	0.6	1.1661904
9	0.8	1.2806248
10	1	1.4142136

Here $h = 2/10 = 0.2$.

$$\begin{aligned} S(f;0.2) &= \frac{h}{3}\left[f_0 + 4(f_1+f_3+f_5+f_7+f_9)\right. \\ &\quad \left. +2(f_2+f_4+f_6+f_8) + f_{10}\right] \\ &= 2.2955778. \end{aligned}$$

Question to ponder:
This value is somewhat smaller than the number we get with trapezoid rule, and it is actually more accurate. Could you intuitively explain that for this particular example?

Sample codes: Let a, b, n be given, and let the function 'func' be defined.
To find the integral with Simpson's rule, one could possibly follow the following algorithm:

```
h=(b-a)/2/n;
xodd=[a+h:2*h:b-h]; % x_i with odd indices
xeven=[a+2*h:2*h:b-2*h];  % x_i with even indices
S=(h/3)*(func(a)+4*sum(func(xodd))+2*sum(func(xeven))+func(b));
```

Error Estimate for Simpson's Rule.

The basic error on each sub-interval is

$$E_{S,i}(f;h) = -\frac{1}{90}h^5 f^{(4)}(\xi_i), \qquad \xi_i \in (x_{2i}, x_{2i+2})\,. \tag{1}$$

(See lecture notes or textbook for the proof.)
Then, the total error is

$$E_S(f;h) = I(f) - S(f;h) = -\frac{1}{90}h^5 \sum_{i=0}^{n-1} f^{(4)}(\xi_i)\frac{1}{n}\cdot\frac{b-a}{2h} = -\frac{b-a}{180}h^4 f^{(4)}(\xi),$$

This gives us the error bound

$$|E_S(f;h)| \le \frac{b-a}{180}h^4 \max_{x\in(a,b)} \left|f^{(4)}(x)\right|.$$

Example 4. With $f(x) = e^x$ defined on $[0, 2]$, use Simpson's rule to compute $\int_0^2 f(x)\,dx$. In order to achieve an error $\leq 0.5 \times 10^{-4}$, how many points must we take?

Answer. We have

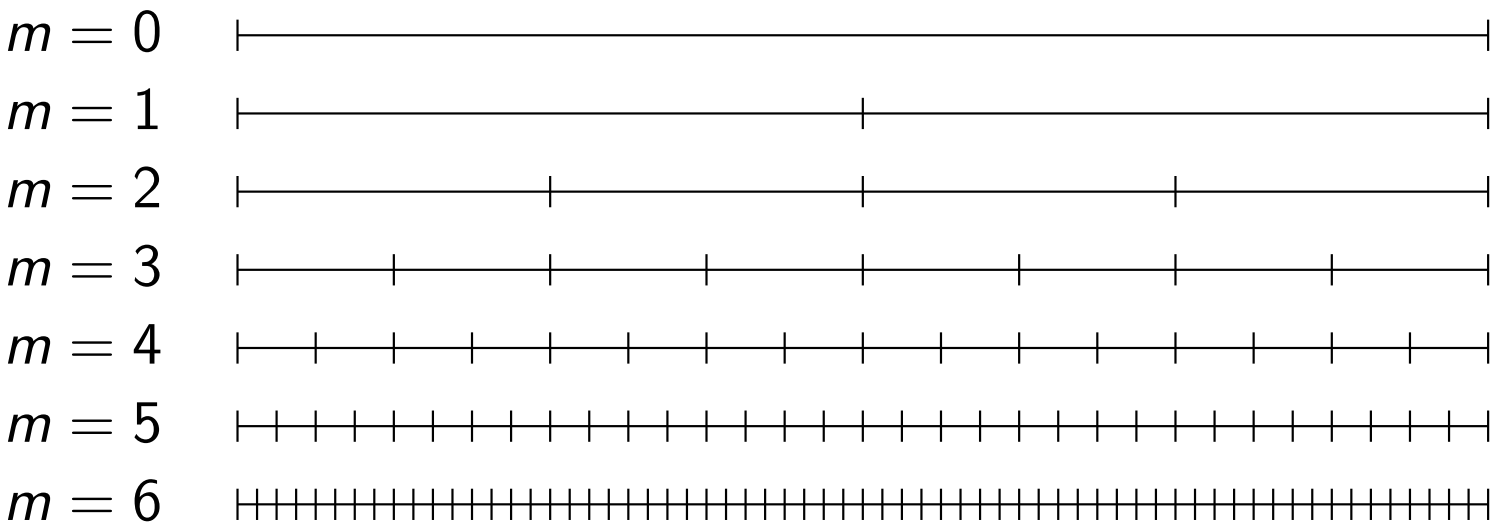
$$\begin{aligned} & |E_S(f; h)| \ \leq \ \frac{2}{180} h^4 e^2 \leq 0.5 \times 10^{-4} \\ \Rightarrow \quad & h^4 \leq 45/e^2 \times 10^{-4} \times = 6.09 \times 10^{-4} \\ \Rightarrow \quad & h \leq 0.1571 \\ \Rightarrow \quad & n = \frac{b-a}{2h} = 6.36 \approx 7 \end{aligned}$$

We need at least $2n + 1 = 15$ points.

Recall: With trapezoid rule, we need at least 314 points. The Simpson's rule uses much fewer points.

Recursive trapezoid rule; composite schemes

Divide $[a, b]$ into 2^m equal sub-intervals.



$$h_m = \frac{b-a}{2^m}, \qquad h_{m+1} = \frac{1}{2} h_m$$

Trapezoid rule:

$$T(f;h_m) = h_m \cdot \left[\frac{1}{2}f(a) + \frac{1}{2}f(b) + \sum_{i=1}^{2^m-1} f(a+ih_m)\right]$$

$$T(f;h_{m+1}) = h_{m+1} \cdot \left[\frac{1}{2}f(a) + \frac{1}{2}f(b) + \sum_{i=1}^{2^{m+1}-1} f(a+ih_{m+1})\right]$$

We can re-arrange the terms in $T(f;h_{m+1})$:

$$\begin{aligned} T(f;h_{m+1}) &= \frac{h_m}{2}\left[\frac{1}{2}f(a) + \frac{1}{2}f(b) + \sum_{i=1}^{2^m-1} f(a+ih_m) \right. \\ &\quad \left. + \sum_{i=0}^{2^m-1} f(a+(2i+1)h_{m+1})\right] \\ &= \frac{1}{2}T(f;h_m) + h_{m+1}\sum_{i=0}^{2^m-1} f(a+(2i+1)h_{m+1}) \end{aligned}$$

Advantages:

1. One can keep the computation for a level m. If this turns out to be not accurate enough, then add one more level to get better approximation. $\Rightarrow$ flexibility.
2. This formula allows us to compute a sequence of approximations to a define integral using the trapezoid rule without re-evaluating the integrand at points where it has already been evaluated. $\Rightarrow$ efficiency.

Question: A similar recursive algorithm could be defined using Simpson's rule. Would you like to try it?

Useful for the next topic: Romberg Algorithm.

Richardson Extrapolation

Given $T(f;h), T(f;h/2), T(f;h/4), \cdots$, a sequence of approximations by Trapezoid rule with different values of h.
Idea: One could combine these numbers in particular ways to get much higher order approximations.
The particular form of the eventual algorithm depends on the detailed error formula.
One can show: If $f^{(n)}$ exists and is bounded, the error for trapezoid rule satisfies the Euler MacLaurin's formula

$$E(f;h) = I(f) - T(f;h) = a_2 h^2 + a_4 h^4 + a_6 h^6 + \cdots + a_n h^n$$

Here a_n depends on the derivatives $f^{(n)}$.
Proof can be achieved by Talor series.

Error formula: $E(f;h) = I(f) - T(f;h) = a_2h^2 + a_4h^4 + a_6h^6 + \cdots + a_nh^n$

When we half the grid size h, the error formula becomes

$$E(f;\frac{h}{2}) = I(f) - T(f;\frac{h}{2}) = a_2(\frac{h}{2})^2 + a_4(\frac{h}{2})^4 + a_6(\frac{h}{2})^6 + \cdots + a_n(\frac{h}{2})^n$$

$$\begin{aligned} (1) \quad I(f) &= T(f;h) + a_2h^2 + a_4h^4 + a_6h^6 + \cdots \\ (2) \quad I(f) &= T(f;\frac{h}{2}) + a_2(\frac{h}{2})^2 + a_4(\frac{h}{2})^4 + a_6(\frac{h}{2})^6 + \cdots + a_n(\frac{h}{2})^n \end{aligned}$$

The goal: cancel the leading error term to get a higher order approximation. Multiplying (2) by 2^2 and subtract (1), we get

$$\begin{aligned} (2^2-1)\cdot I(f) &= 2^2 \cdot T(f;h/2) - T(f;h) + a_4'h^4 + a_6'h^6 + \cdots \\ \Rightarrow I(f) &= \underbrace{\frac{4}{3}T(f;h/2) - \frac{1}{3}T(f;h)}_{U(h)} + \tilde{a}_4h^4 + \tilde{a}_6h^6 + \cdots \end{aligned}$$

A 4th order approximation:

$$U(h) = \frac{4}{3} T(f; h/2) - \frac{1}{3} T(f; h) = \frac{2^2 T(f; h/2) - T(f; h)}{2^2 - 1}$$

This idea is called the *Richardson extrapolation*.

We do not have to stop here. One can go into many levels of this manipulation! $\Rightarrow$ Romberg Algorithm

Romberg Algorithm

Given $T(f;h), T(f;h/2)$, compute

$$U(h) = T(f;h/2) + \frac{T(f;h/2) - T(f;h)}{2^2 - 1}$$

then

$$I(f) = U(h) + \tilde{a}_4 h^4 + \tilde{a}_6 h^6 + \cdots$$

We can iterate this idea. Assume we have computed $U(h), U(h/2)$,

$$\begin{aligned} (3) \quad I(f) &= U(h) + \tilde{a}_4 h^4 + \tilde{a}_6 h^6 + \cdots \\ (4) \quad I(f) &= U(h/2) + \tilde{a}_4 (h/2)^4 + \tilde{a}_6 (h/2)^6 + \cdots \end{aligned}$$

Cancel the leading error term: $(4) \times 2^4 - (3)$

$$(2^4 - 1) I(f) = 2^4 U(h/2) - U(h) + \tilde{a}'_6 h^6 + \cdots$$

Let

$$V(h) = \frac{2^4 U(h/2) - U(h)}{2^4 - 1} = U(h/2) + \frac{U(h/2) - U(h)}{2^4 - 1}.$$

Then

$$I(f) = V(h) + \tilde{a}_6' h^6 + \cdots \qquad \text{6th order approximation}$$

So $V(h)$ is even better than $U(h)$.

One can keep doing this several layers, until desired accuracy is reached.

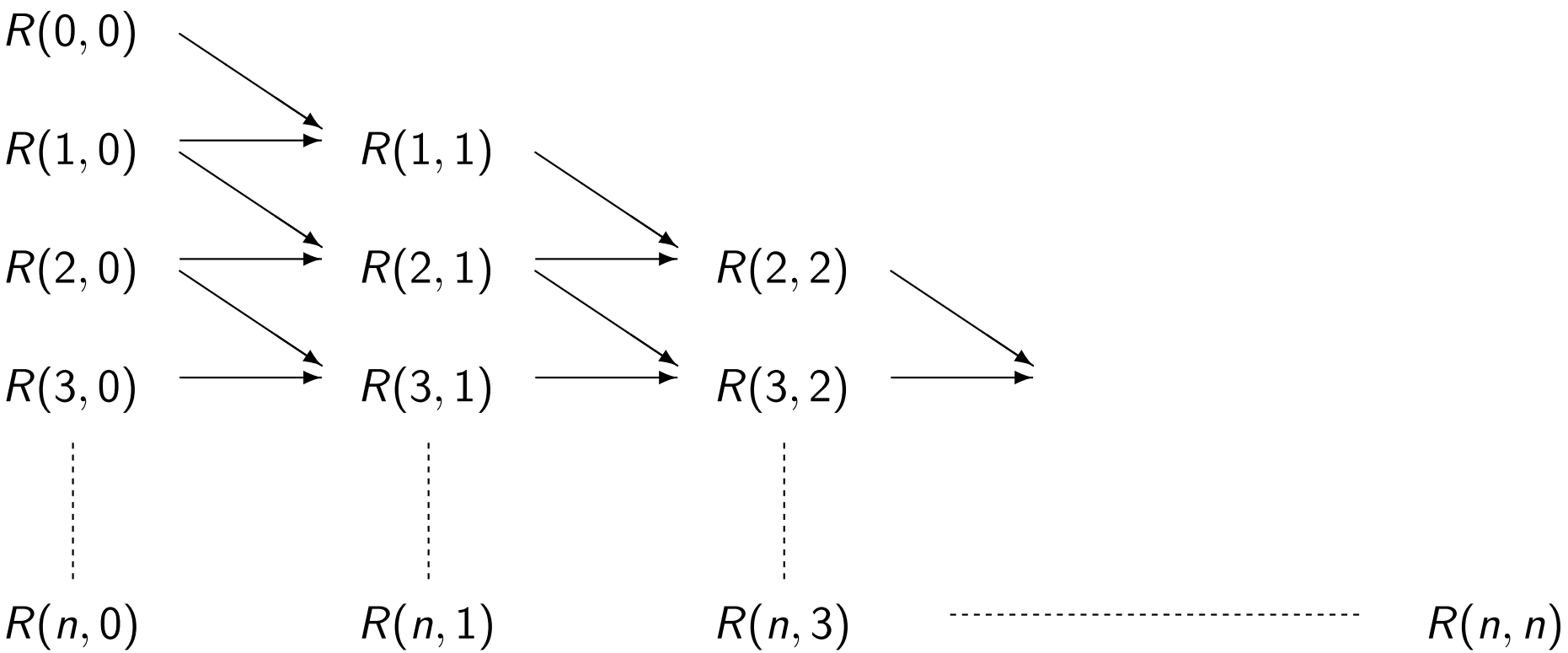
This gives the **Romberg Algorithm**

Romberg Algorithm

Set $H = b - a$, define:

$$\begin{aligned} R(0,0) &= T(f;H) = \frac{H}{2}(f(a)+f(b)) \\ R(1,0) &= T(f;H/2) \\ R(2,0) &= T(f;H/(2^2)) \\ &\vdots \\ R(n,0) &= T(f;H/(2^n)) \end{aligned}$$

Romberg triangle



The entry $R(n, m)$ is computed as

$$R(n, m) = R(n, m-1) + \frac{R(n, m-1) - R(n-1, m-1)}{2^{2m} - 1}$$

Accuracy:

$$I(f) = R(n, m) + \mathcal{O}(h^{2(m+1)}), \qquad h = \frac{H}{2^m}.$$

Algorithm can be done either column-by-column or row-by-row.

Romberg algorithm pseudo-code, using column-by-column

R =romberg(f, a, b, n)

$R = n \times n$ matrix

$h = b - a$; $R(1,1) = [f(a) + f(b)] * h/2$;

for $i = 1$ to $n - 1$ do % 1st column recursive trapezoid

$R(i+1,1) = R(i,1)/2$; $h = h/2$;

for $k = 1$ to 2^{i-1} do

$R(i+1,1) = R(i+1,1) + h * f(a + (2k-1)h)$

end

end

for $j = 2$ to n do % 2 to n column

for $i = j$ to n do

$R(i,j) = R(i,j-1) + \frac{1}{4^{j-1}-1}[R(i,j-1) - R(i-1,j-1)]$

end

end

CMPSC/Math 451, Numerical Computation

Wen Shen

Department of Mathematics, Penn State University

Adaptive Simpson's quadrature scheme

Intuition: Local error is large where function changes a lot, i.e, where $f^{(k)}$'s are big.

In uniform grid, the error is not evenly distributed, we waste effort in region where f changes little.

Adaptive grid: smaller grid size h in region where f varies more.

We illustrate this idea using Simpson's rule.

For interval $[a, b]$, $h = \frac{b-a}{2}$,

$$\text{Simpson's rule:} \quad S_1[a, b] = \frac{b-a}{6}\left[f(a) + 4f(\frac{a+b}{2}) + f(b)\right]$$

$$\text{Error:} \quad E_1[a, b] = -\frac{1}{90}h^5 f^{(4)}(\xi), \qquad \xi \in (a, b)$$

Then

$$I(f)[a, b] = S_1[a, b] + E_1[a, b]$$

Divide $[a, b]$ up in the middle, let $c = \frac{a+b}{2}$.

$$\begin{aligned} I(f)[a,b] &= I(f)[a,c] + I(f)[c,b] = S_1[a,c] + E_1[a,c] + S_1[c,b] + E_1[c,b] \\ &= S_2[a,b] + E_2[a,b] \end{aligned}$$

where

$$\begin{aligned} S_2[a,b] &= S_1[a,c] + S_1[c,b] \\ E_2[a,b] &= E_1[a,c] + E_1[c,b] = -\frac{1}{90}(h/2)^5 \left[f^{(4)}(\xi_1) + f^{(4)}(\xi_2) \right] \end{aligned}$$

Assume $f^{(4)}$ does NOT change much, then $E_1[a,c] \approx E_1[c,b]$,

$$E_2[a,b] \approx 2E_1[a,c] = 2\frac{1}{2^5} E_1[a,b] = \frac{1}{2^4} E_1[a,b]$$

$$I(f) = S_1 + E_1, \qquad I(f) = S_2 + E_2 = S_2 + \frac{1}{2^4} E_1$$

$$S_2[a,b] - S_1[a,b] = E_1 - E_2 = 2^4 E_2 - E_2 = 15 E_2$$

This means, we can compute a good approximation to the error E_2:

$$E_2 = \frac{1}{15}(S_2 - S_1)$$

If error tolerance is $|E_2| \leq \varepsilon$, we only need to require

$$\frac{S_2 - S_1}{2^4 - 1} \leq \varepsilon \qquad \text{A-priori error estimate}$$

Improve the result:

$$\Rightarrow \quad I = S_2 + E_2 = S_2 + \frac{S_2 - S_1}{15}$$

Note that this gives the best approximation when $f^{(4)} \approx const$.

An adaptive recursive algorithm:

Pseudocode: f: function, $[a, b]$ interval, ε: tolerance for error

answer=Simpson(f, a, b, ε)

compute S_1 and S_2

If $|S_2 - S_1| < 15\varepsilon$

answer= $S_2 + (S_2 - S_1)/15$;

else

$c = (a + b)/2$;

Lans=Simpson($f, a, c, \varepsilon/2$);

Rans=Simpson($f, c, b, \varepsilon/2$);

answer=Lans+Rans;

end

In Matlab: 'quad', 'quad8'

Question:
One could also design an adaptive trapezoid method. Would you like to try it and work out the algorithm?

Of course, due to the higher accuracy of Simpson's rule, it is the mostly commonly used one, even in adaptive form.

Gaussian quadrature

All the numerical integration rules follow the form

$$\int_a^b f(x)\,dx \approx A_0 f(x_0) + A_1 f(x_1) + \cdots + A_n f(x_n)\,.$$

Here $x_i \in [a, b]$ are called the *nodes*, and A_i's are the weights.
For example, the trapezoid rule is:

$$x_0 = a,\ x_1 = b,\ A_0 = A_1 = \frac{b-a}{2}$$

and the Simpson's rule corresponds to

$$x_0 = a,\ x_1 = \frac{a+b}{2},\ x_2 = b,\ A_0 = A_2 = (b-a)\frac{1}{6},\ A_1 = (b-a)\frac{2}{3}.$$

In these rules, we fix the points x_i, then we adjust the weights A_i.

Gaussian quadrature

Idea: allow both the nodes x_i and the weights A_i to be adjusted, to achieve high accuracy.

One chooses the nodes x_i and weight A_i $(i = 0, 1, 2, \cdots, N)$ such that the algorithm gives the exact value for polynomial functions $f(x)$ of highest possible degree m:

$$\int_a^b f(x)\,dx \;=\; A_0 f(x_0) + A_1 f(x_1) + \cdots + A_n f(x_m)\,, \qquad \text{if } f \in P^m. \quad (1)$$

Here, m is called **the degree of precision**.

To fix the idea, we consider now the interval $[-1, 1]$.
Start with $N = 1$, i.e., we have 2 nodes x_0, x_1 and 2 weights A_0, A_1.
The rule satisfies

$$\int_{-1}^{1} f(x)\,dx = A_0 f(x_0) + A_1 f(x_1), \qquad f \in \mathcal{P}_m$$

$\Rightarrow$ the rule must be exact for functions $1, x, x^2, x^3, \cdots, x^m$.
We will need 4 equations for 4 unknowns, therefore $m = 3$.
Recall that

$$\int_{-1}^{1} x^k\,dx = \begin{cases} \frac{2}{k+1}, & k \text{ even} \\ 0, & k \text{ odd.} \end{cases}$$

$$\begin{aligned} f(x) = 1: &\qquad A_0 + A_1 = 2 \\ f(x) = x: &\qquad A_0 x_0 + A_1 x_1 = 0 \\ f(x) = x^2: &\qquad A_0 x_0^2 + A_1 x_1^2 = \frac{2}{3} \\ f(x) = x^3: &\qquad A_0 x_0^3 + A_1 x_1^3 = 0 \end{aligned}$$

One could solve this system and find

$$x_0 = -\frac{1}{\sqrt{3}}, \quad x_1 = \frac{1}{\sqrt{3}}, \quad A_0 = 1, \quad A_1 = 1.$$

Note the symmetry: $x_0 = -x_1$ and $A_0 = A_1$, which should be expected and could be used to simply the computation!
Then, we have the Gaussian quadrature rule for $n = 1$,

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

This will have degree of precision 3.

Gaussian quadrature, $N = 2$

Nodes: x_0, x_1, x_2 , and weights: A_0, A_1, A_2.

$$\int_{-1}^{1} f(x)\,dx \approx A_0 f(x_0) + A_1 f(A_1) + A_2 f(x_2)$$

The rule should give exact solution for polynomials of degree $m = 5$.

$$\text{symmetry:} \quad x_0 = -x_2, \quad x_1 = 0, \quad A_0 = A_2,$$

Only need to check with $f(x) = 1, x^2, x^4$:

$$\begin{aligned} f(x) = 1: &\qquad 2A_0 + A_1 = 2 \\ f(x) = x^2: &\qquad 2A_0 x_0^2 = 2/3 \\ f(x) = x^4: &\qquad 2A_0 x_0^4 = 2/5 \end{aligned}$$

$$x_0 = \sqrt{3/5}, \quad x_1 = 0, \quad x_2 = -\sqrt{3/5}, \quad A_0 = A_2 = 5/9, \quad A_1 = 8/9.$$

$$\text{GQ:} \quad \int_{-1}^{1} f(x)\,dx \approx \frac{1}{9}\left[5f\left(-\sqrt{3/5}\right) + 8f(0) + 5f\left(\sqrt{3/5}\right)\right].$$

Table: Gaussian Quadrature Nodes and Weights

#points	Nodes x_i	Weights A_i
1	0	2
2	$\pm\sqrt{1/3}$	1
3	$\pm\sqrt{0.6}$	$5/9$
	0	$8/9$
4	$\pm\sqrt{(3-\sqrt{4.8})/7}$	$(18+\sqrt{30})/36$
	$\pm\sqrt{(3+\sqrt{4.8})/7}$	$(18-\sqrt{30})/36$
5	$\pm\sqrt{(5-\sqrt{40/7})/9}$	$(322+13\sqrt{70})/900$
	0	$128/225$
	$\pm\sqrt{(5+\sqrt{40/7})/9}$	$(322-13\sqrt{70})/900$

For general interval $[a, b]$, use the transformation:

$$t = \frac{2x - (a+b)}{b-a}, \qquad x = \frac{1}{2}(b-a)t + \frac{1}{2}(a+b)$$

so for $-1 \leq t \leq 1$ we have $a \leq x \leq b$.

Advantage:
(1). Higher order accuracy.
(2). Since all nodes are in the interior of the interval, these formulas can handle integrals of function that tends to infinite value at one end of the interval (provided that the integral is defined).

CMPSC/Math 451, Numerical Computation

Wen Shen

Department of Mathematics, Penn State University

Matlab: Effective programming

Three different ways of computing a vector product of two vectors.

$$z_i = x_i \cdot y_i, \qquad i = 1, 2, \ldots, n$$

Method 1:
Do not allocate memory space for z in advance.
Compute the elements in the new vector by a for-loop.

```
x = rand(n,1);     % random vector
y = rand(n,1);     % of length n
clear z;
t = cputime;
for i=1:n
   z(i) = x(i)*y(i);
end
cputime-t
```

Method 2:
Allocate space for z in advance, but still use a for loop.

```
z = zeros(n,1);
for i=1:n
   z(i) = x(i)*y(i);
end
```

Method 3:
Use vector operation in Matlab directly.

```
z = x.*y;
```

Result (The CPU-time measured in seconds):

n	Method 1	Method 2	Method 3
5000	2.86	0.24	0.00
10000	14.22	0.49	0.00
20000	59.65	0.97	0.01
100000	−−	4.87	0.03
1000000	−−	48.84	0.30

Moral: use pre-defined Matlab functions!

Romberg integration

$$I(f) = \int_0^{\pi/2} \cos(2x)e^{-x}dx, \qquad (= 0.2415759)$$

Result:

```
0.6221
0.3111    0.2074
0.2575    0.2397    0.2419
0.2455    0.2415    0.2416    0.2416
```

Error:

```
3.80e-01
6.94e-02   3.41e-02
1.59e-02   1.87e-03   2.81e-04
3.90e-03   1.11e-04   5.85e-06   1.47e-06
```

Expand the integration interval to 2π. Exact value$= 0.1996265$.
Result:

```
3.1475
1.7095    1.2302
0.5141    0.1156    0.0413
0.2570    0.1714    0.1751    0.1772
```

Error:

```
2.94e+00
1.50e+00   1.03e+00
3.14e-01   8.39e-02   1.58e-01
5.74e-02   2.82e-02   2.45e-02   2.24e-02
```

Matlab's adaptive Simpsons methode: quad

Syntax:

```
q = quad('f',a,b,toleranse,trace)
```

If the option *trace*$\neq 0$, Matlab will show the development of the integration.

```
quad('f3',0,2*pi,1.e-6,1)
```

Matlab's recursive numerical integration: quadl and quad8

Syntax and usage would be the same as for quad.
The functions use a high order recursive algorithm.