TDT4127 Programming and Numerics
Week 37

Numerical integration
Algorithms: midpoint, trapezoidal and Simpson’s rules
Learning goals

- Goals
  - Numerical integration
    - Algorithm statements
      - Midpoint rule, Trapezoidal rule, Simpson’s rule
    - Error analysis
    - Implementation tips

- Curriculum
  - Exercise 3
  - Auditorium exercise 1
Numerical integration

• Everyone loves to integrate! But it can be hard.

\[ \int_{0}^{1} \tan(\cos(\sin(e^{x^5}))) \, dx = ? \]

• Integrating in 1D = Finding area under the graph

• The idea: Approximate \( f(x) \) by something easier to integrate
  – In particular, polynomials are really easy and approximate well!
Midpoint rule

- Approximate the function by a constant and integrate
  - Best constant is the value at the midpoint, $f((a+b)/2)$.

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

- Approximate the function $f$ by a **linear** function $g$
  - Choose $g$ to interpolate $f$ at the endpoints; $g(a) = f(a)$, $g(b) = f(b)$
    
    $$g(x) = f(a)(x-b)/(a-b) + f(b)(x-a)/(b-a)$$

$$\int_{a}^{b} f(x)\,dx \approx (f(a) + f(b)) \frac{b-a}{2}$$
Simpson’s rule

• Approximate the function \( f \) by a quadratic function \( g \)
  
  - Interpolate at \( c = (a+b)/2; \) \( g(a) = f(a), \) \( g(b) = f(b), \) \( g(c) = f(c) \)

\[
f(x) \approx g(x) = f(a) \frac{(x-b)(x-c)}{(a-b)(a-c)} + f(b) \frac{(x-a)(x-c)}{(b-a)(b-c)} + f(c) \frac{(x-a)(x-b)}{(c-b)(c-b)}.
\]

\[
\int_a^b f(x) \, dx \approx \frac{b-a}{6} \left( f(a) + 4f(c) + f(b) \right).
\]
Composite rules

• All three rules give *alright* estimates
  – But we can see with the naked eye that they make mistakes!
• To improve, we split the interval \([a,b]\) into smaller ones

\[
\int_a^b f(x) \, dx = \int_a^c f(x) \, dx + \int_c^b f(x) \, dx \approx \int_a^c g(x) \, dx + \int_c^b g(x) \, dx
\]
  – This is called a *composite* method
    • We often drop «composite» from the name
  – Typically, we call the number of intervals \(N\)
  – We will consider intervals of fixed width \(h\)
    • Non-fixed widths; is something we’ll get back to in November
  – Splitting an interval of width \((b-a)\) into \(N\) parts gives a width of \(h=(b-a)/N\).
Composite midpoint rule

- Use a **constant** approximation on each subinterval
  - Subintervals: \([x_k, x_{k+1}], k = 0, \ldots, N-1\). \(x_k = a + kh. \quad h=(b-a)/N.\)

\[
\int_a^b f(x) \, dx \approx h \sum_{k=0}^{N-1} f(c_k), \quad c_k = \frac{x_{k+1} + x_k}{2}
\]
Composite midpoint rule

• Use a constant approximation on each subinterval
  – Subintervals: \([x_k, x_{k+1}], k = 0, \ldots, N-1\). \(x_k = a + kh\). \(h = (b-a)/N\).

\[
\int_{a}^{b} f(x)\,dx \approx h \sum_{k=0}^{N-1} f(c_k), \quad c_k = \frac{x_{k+1} + x_k}{2}
\]
Composite midpoint rule

- Use a **constant** approximation on each subinterval
  - Subintervals: \([x_k, x_{k+1}], k = 0, \ldots, N-1. \quad x_k = a + kh. \quad h=(b-a)/N.

\[
\int_{a}^{b} f(x) \, dx \approx h \sum_{k=0}^{N-1} f(c_k), \quad c_k = \frac{x_{k+1} + x_k}{2}
\]
Composite trapezoidal rule

- Use a **linear** approximation on each subinterval
  - Subintervals: \([x_k, x_{k+1}], k = 0, \ldots, N-1\). \(x_k = a + kh.\) \(h=(b-a)/N.\)

\[
\int_a^b f(x)\,dx \approx \frac{h}{2} \left( f(x_0) + 2 \sum_{k=1}^{N-1} f(x_k) + f(x_N) \right)
\]

\(N = 2, \ h = 0.5\)
Composite trapezoidal rule

- Use a **linear** approximation on each subinterval
  - Subintervals: \([x_k, x_{k+1}], k = 0, \ldots, N-1.\)  
    \[x_k = a + kh.\]  
    \[h = (b-a)/N.\]

\[
\int_a^b f(x) \, dx \approx \frac{h}{2} \left( f(x_0) + 2 \sum_{k=1}^{N-1} f(x_k) + f(x_N) \right)
\]

\(N = 5, \ h = 0.2\)
Composite trapezoidal rule

- Use a **linear** approximation on each subinterval
  - Subintervals: \([x_k, x_{k+1}], k = 0, \ldots, N-1.\) \(x_k = a + kh.\) \(h=(b-a)/N.\)

\[
\int_a^b f(x)\,dx \approx \frac{h}{2} \left( f(x_0) + 2 \sum_{k=1}^{N-1} f(x_k) + f(x_N) \right)
\]
Composite Simpson’s rule

- Use a **quadratic** approximation on each subinterval
  - Subintervals: \([x_{2k}, x_{2k+2}], k = 0,\ldots,N-1\).
  - \(x_k = a + kh\). \(h=(b-a)/2N\).

\[
\int_a^b f(x)\,dx \approx \frac{h}{3} \left( f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + \cdots + 2f(x_{2N-2}) + 4f(x_{2N-1}) + f(x_{2N}) \right),
\]

- Note the odd/even coefficients of 4 and 2

\(N = 2, \ h = 0.5\)
Composite Simpson’s rule

• Use a **quadratic** approximation on each subinterval
  
  – Subintervals: \([x_{2k}, x_{2k+2}], \ k = 0,\ldots,N-1.\)  
  
  \[x_k = a + kh. \quad h = (b-a)/2N.\]

\[
\int_{a}^{b} f(x) \, dx \approx \frac{h}{3} \left( f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + \ldots + 2f(x_{2N-2}) + 4f(x_{2N-1}) + f(x_{2N}) \right),
\]

\[N = 5, \ h = 0.2\]

– Note the odd/even coefficients of 4 and 2
Composite Simpson’s rule

- Use a **quadratic** approximation on each subinterval
  - Subintervals: \([x_{2k}, x_{2k+2}], \ k = 0, \ldots, N-1. \quad x_k = a + kh. \quad h=(b-a)/2N.\]

\[
\int_a^b f(x)\,dx \approx \frac{h}{3} (f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + \ldots + 2f(x_{2N-2}) + 4f(x_{2N-1}) + f(x_{2N})) , \quad h = \frac{b-a}{2N}
\]

- Note the odd/even coefficients of 4 and 2
Implementation

• Sums and for loops go hand in hand
• Example:

\[ S = \sum_{k=0}^{N} a_k \]

Translation into code:

```python
S = 0
for k in range(0,N+1):
    a_k = ...
    S = S + a_k
```
Error analysis

• We can get an estimate for the error of the midpoint method assuming $f$ is \textit{continuously differentiable} (also written $C^1$)
  – A function $f$ is continuously diff’ble if $f'$ is continuous.
  – Examples: $f(x) = x^2$ and $f(x) = e^x$
  – Non-example: $f(x) = |x|

• The midpoint rule has an error estimate:
  $$E_{MP} = \left| \int_a^b f(x)dx - (b - a)f \left( \frac{a + b}{2} \right) \right| \leq \frac{(b - a)^3}{24}M$$

• Where $M$ is the maximum value of $|f''(x)|$ on $[a,b]$.
  – \textit{Note that this estimate requires continuous differentiability of} $f$
Error analysis

• For the composite midpoint method, we simply use the error estimate on each subinterval \([x_k, x_{k+1}]\)

\[
E_{\text{MP},k} = \left| \int_{x_k}^{x_{k+1}} f(x) \, dx - h f(c_k) \right| \leq \frac{h^3}{24} M
\]

• Summing up all of these, we find the total error

\[
E_{\text{CMP}} \leq \sum_{k=0}^{N-1} E_{\text{MP},k} \leq \sum_{k=0}^{N-1} \frac{h^3}{24} M = N \frac{h^3}{24} M = \frac{(b - a)^3}{24N^2} M
\]

• As \(N\) increases, the error decreases and so the approximation converges to the true integral as \(N \to \infty\)
Error analysis

• Similar estimates can be made for (composite) trapezoidal (TR) and (composite) Simpson’s (SI) rules

\[ E_{\text{TR}} = \left| \int_a^b f(x)dx - \frac{b-a}{2}(f(a) + f(b)) \right| \leq \frac{(b-a)^3}{12} M \]

\[ E_{\text{CTR}} \leq \frac{(b-a)^3}{12N^2} M \]

\[ E_{\text{SI}} = \left| \int_a^b f(x)dx - \frac{b-a}{6} \left( f(a) + 4f \left( \frac{a+b}{2} \right) + f(b) \right) \right| \leq \frac{(b-a)^5}{2880} M_4 \]

\[ E_{\text{CSI}} \leq \frac{(b-a)^5}{2880N^4} M_4, \quad M_4 = \max_{a \leq x \leq b} f^{(4)}(x) \]

• Note that composite Simpson goes as \( 1/N^4 \)
  – And requires a continuous 4th derivative of \( f \), (Notation: \( f \) is \( C^4 \)).
Guaranteed error estimates

• The error analysis is useful since it gives us the worst-case behaviour of the algorithm
• If we want, we can guarantee a level of precision in the numerical approximation
  – For example, to make sure the integral of a $C^4$ function has error at most $\epsilon$, use the Simpson’s rule and choose $N$ such that
    \[ E_{\text{CSI}} \leq \frac{(b - a)^5}{2880N^4} M_4 = \epsilon, \quad M_4 = \max_{a \leq x \leq b} |f''''(x)| \]
  – Note: the estimates may be too conservative, suggesting more iterations than necessary, but they are safe
Summary

• Numerical integration is used to evaluate integrals
• We have seen three methods
  – Midpoint rule, trapezoidal rule and Simpson’s rule
• Also seen the composite rules based on these
  – With error analysis, useful for guaranteeing errors!
Questions?