Chapter 13. Numerical Integration
Outline

• 13.1 Introduction
• 13.2 Rectangle Method
• 13.3 Trapezoidal Rule
• 13.4 Simpson’s ⅓ Rule
• 13.5 Simpson’s ⅜ Rule
• 13.6 Newton–Cotes Formulas
• 13.7 Multiple-Application Scheme
• 13.8 Gauss Quadrature
• 13.9 Using the MATLAB Functions
### MATLAB Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>trapz</code> (cumtrapz)</td>
<td>Integration using trapezoidal method</td>
</tr>
<tr>
<td><code>quad</code></td>
<td>Integration using recursive adaptive Simpson’s rule</td>
</tr>
<tr>
<td><code>quadl</code></td>
<td>Integration using adaptive Labatto quadrature</td>
</tr>
<tr>
<td><code>quadgk</code></td>
<td>Integration using adaptive Gauss-Kronrod quadrature</td>
</tr>
<tr>
<td><code>quadv</code></td>
<td>Vectorized version of <code>quad</code> for an array-valued input function</td>
</tr>
<tr>
<td><code>dblquad</code></td>
<td>Evaluation of a double integral over a rectangle</td>
</tr>
<tr>
<td><code>quad2d</code></td>
<td>Evaluation of a double integral using 2-D quadrature</td>
</tr>
<tr>
<td><code>triplequad</code></td>
<td>Evaluation of a triple integral over a rectangular cuboid</td>
</tr>
</tbody>
</table>

- `trapz` (cumtrapz) is the only function working on set of data as opposed to the analytical function.
- `quad` may be most efficient for low accuracies with nonsmooth integrands.
- `quadl` may be more efficient than `quad` at higher accuracies with smooth integrands.
- `quadgk` may be most efficient for oscillatory integrands and any smooth integrand at high accuracies. It supports infinite intervals and can handle moderate singularities at the endpoints. It also supports contour integration along piecewise linear paths.
- `dblquad` and `triplequad` allow using your own quadrature function (instead of `quad`, `quadl`, or `quadgk`).

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>integral</code></td>
<td>Vectorized adaptive quadrature (will replace <code>quad</code>, <code>quadv</code> and <code>quadl</code>)</td>
</tr>
<tr>
<td><code>integral2</code></td>
<td>Vectorized adaptive quadrature (will replace <code>dblquad</code> and <code>quad2d</code>)</td>
</tr>
<tr>
<td><code>integral3</code></td>
<td>Vectorized adaptive quadrature (will replace <code>triplequad</code>)</td>
</tr>
</tbody>
</table>

All rights reserved. No part of this publication may be reproduced, distributed, or transmitted, unless for course participation, in any form or by any means, or stored in a database or retrieval system, without the prior written permission of the Publisher and/or Author. Contact the American Institute of Aeronautics and Astronautics, Professional Development Programs, 12700 Sunrise Valley Drive, Suite 200, Reston, VA 20191-5807.
The integral Family
(introduced in R2012a)

**integral**
- evaluates a single integral
\[ \int_{a}^{b} f(x) \, dx \]
  - including: improper integrals \((a=-\infty \text{ and/or } b=\infty)\),
    - integrals with singularities at the boundaries,
    - integrals with complex contours \((a\text{ and } b\text{ are complex numbers})\),
    - improper integrals of the oscillatory functions

**integral2**
- evaluates a double integral
\[ \int_{a}^{b} \int_{c(x)}^{d(x)} f(x, y) \, dy \, dx \]
  - including: improper integrals \((a=-\infty \text{ and/or } b=\infty)\),
    - integrals with singularities at the boundaries,
    - integrals over generalized 2D regions (non-rectangular regions),
    - integrals in polar coordinates

**integral3**
- evaluates a triple integral
\[ \int_{a}^{b} \int_{c(x)}^{d(x)} \int_{g(x,y)}^{h(x,y)} f(x, y, z) \, dz \, dy \, dx \]
  - including: improper integrals \((a=-\infty \text{ and/or } b=\infty)\),
    - integrals with singularities at the boundaries,
    - integrals over generalized 3D regions,
    - integrals in spherical coordinates
Sample Points

\[ y = f(x) \]

\[ f(x_n) = f(x_{n-1}) \]

\[ x_i - x_{i-1} \neq x_{i+1} - x_i \]

\[ x_i - x_{i-1} = x_{i+1} - x_i = h \]

\[ y = f(x) \]

\[ f(x_1), f(x_2), \ldots, f(x_n) \]

\[ x_1 = a, x_2, x_3, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_{n-2}, x_n = b \]

All rights reserved. No part of this publication may be reproduced, distributed, or transmitted, unless for course participation, in any form or by any means, or stored in a database or retrieval system, without the prior written permission of the Publisher and/or Author. Contact the American Institute of Aeronautics and Astronautics, Professional Development Programs, 12700 Sunrise Valley Drive, Suite 200, Reston, VA 20191-5807.
Straightforward Approach

**Right rectangles formula**

\[ I_{RR} \approx \sum_{i=2}^{n} f(x_{i-1})(x_{i} - x_{i-1}) \]

**Left rectangles formula**

\[ I_{LR} \approx \sum_{i=2}^{n} f(x_{i})(x_{i} - x_{i-1}) \]
Trapezoidal Integration

For \( x_i - x_{i-1} = x_{i+1} - x_i = h \)

\[
I^T \approx \frac{h}{2} \left[ f_1 + 2 \sum_{i=2}^{n-1} f_i + f_n \right] + E
\]

where \( E = \sum_{i=1}^{n-1} e_i = \bar{e}(n-1) = \bar{e} \frac{b-a}{h} \)

\[ \bar{e} = f(h) \]
Formal Approach

\[ I(x_{i+1}) = I(x_i) + h I'(x_i) + \frac{h^2}{2!} I''(x_i) + \frac{h^3}{3!} I'''(x_i) + \ldots \]

\[ f(x_{i+1}) = f(x_i) + hf'(x_i) + \frac{h^2}{2!} f''(x_i) + \ldots \]

\[ I'(x_i) = f(x_i) \]
\[ I''(x_i) = f'(x_i) \]
\[ I'''(x_i) = f''(x_i) \]

\[ f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{h} - \frac{h}{2!} f''(x_i) + \ldots \]

\[ e_i \approx -\frac{h^3}{12} f''(x_i) \]

\[ E = \frac{b-a}{h} \bar{e} \approx -\frac{h^2}{12} (b-a) f''(\bar{x}) \]

All rights reserved. No part of this publication may be reproduced, distributed, or transmitted, unless for course participation, in any form or by any means, or stored in a database or retrieval system, without the prior written permission of the Publisher and/or Author. Contact the American Institute of Aeronautics and Astronautics, Professional Development Programs, 12700 Sunrise Valley Drive, Suite 200, Reston, VA 20191-5807.
Let's Try Symbolic Math Toolbox

%% Define symbolic variables
syms I_iplus1 I_i f_iplus1 f_i I_prime I_2prime I_3prime h f_prime f_2prime

%% Introduce Taylor series equation for the integral
Eq = -I_iplus1 + I_i + h*I_prime + h^2*I_2prime/factorial(2) + h^3*I_3prime/factorial(3);

%% Make the first set of substitutes
Eq = subs(Eq, {I_prime I_2prime I_3prime}, {f_i f_prime f_2prime});

%% Substitute function derivative with the forward difference approximation
Eq = subs(Eq, f_prime, (f_iplus1 - f_i)/h - h*f_2prime/2);

%% Compute a single area under the curve
ans = solve(Eq, I_iplus1) - I_i;

%% Collect coefficients and display the result
r = collect(ans, h); pretty(r)

\[
e_i \approx -\frac{h^3}{12} f''(x_i) \quad \Delta I(x_i) \approx h \frac{f(x_i) + f(x_{i+1})}{2}
\]
Another Look at Trapezoidal Integration

Why not to try higher-order interpolation formulas?
For each double interval we have the following interpolation

Without loss of generality, assume \( x_i - x_{i-1} = x_{i+1} - x_i = h \)

The coefficients \( K, L, \) and \( M \) are found from

Then the integral for each double interval is

\[
\Delta I_i = \int_{x_i}^{x_i+2h} f^*(x)dx + e_i = \frac{8}{3} h^3 K + 2h^2 L + 2hM + e_i
\]
Let's Rely on Symbolic Math Toolbox Again

%% Define symbolic variables
syms x_i f_i f_iplus1 f_iplus2 h x

%% Solve for coefficients of a parabolic interpolation
A = [ x_i^2 x_i 1;
      (x_i+h)^2 x_i+h 1;
      (x_i+2*h)^2 x_i+2*h 1];
b = [f_i; f_iplus1; f_iplus2];
coef = A\b;

%% Compute the integral
dl = int([x^2, x, 1]*coef, x_i, x_i+2*h);

%% Display the result
pretty(dl)

\[ \Delta I(x_i) \approx \frac{1}{3} h \left( f(x_i) + 4f(x_{i+1}) + f(x_{i+2}) \right) \]
Formal Approach

\[ I(x_{i-1}) = I(x_i) - h I'(x_i) + \frac{h^2}{2!} I''(x_i) - \frac{h^3}{3!} I'''(x_i) + \ldots \]

\[ I(x_{i+1}) = I(x_i) + h I'(x_i) + \frac{h^2}{2!} I''(x_i) + \frac{h^3}{3!} I'''(x_i) + \ldots \]

\[ \Delta I_{i-1} = I(x_{i+1}) - I(x_{i-1}) = 2h I'(x_i) + 2 \frac{h^3}{3!} I'''(x_i) + 2 \frac{h^5}{5!} I^{(v)}(x_i) + \ldots \]

\[ I'(x_i) = f(x_i) \]
\[ I''(x_i) = f''(x_i) \]
\[ I^{(v)}(x_i) = f^{(iv)}(x_i) \]
\[ f''''(x_i) = \frac{f(x_{i+1}) - 2f(x_i) + f(x_{i-1})}{h^2} - \frac{h^2}{12} f^{(iv)}(x_i) - \ldots \]

\[ \Delta I_{i-1} = I(x_{i+1}) - I(x_{i-1}) = \frac{1}{3} h \left( f(x_{i-1}) + 4f(x_i) + f(x_{i+1}) \right) + O(h^5) \]

\[ e_{i-1} \approx -\frac{h^5}{90} f^{(iv)}(x_i) \]
\[ E = \frac{b-a}{2h} \overline{e} \approx -\frac{h^4}{180} (b-a) f^{(iv)}(\overline{x}) \]
%% Define symbolic variables (assume middle point to be i-th point)
syms I_prime I_2prime I_3prime I_4prime I_5prime I_i h
syms f_iminus1 f_i f_iplus1 f_prime f_2prime f_3prime f_4prime

%% Taylor series expansions for I_plus1 and I_minus1
Eq1 = I_i + h*I_prime + h^2*I_2prime/factorial(2) + h^3*I_3prime/factorial(3) +...
     + h^4*I_4prime/factorial(4) + h^5*I_5prime/factorial(5);
Eq2 = I_i - h*I_prime + h^2*I_2prime/factorial(2) - h^3*I_3prime/factorial(3) +...
     + h^4*I_4prime/factorial(4) - h^5*I_5prime/factorial(5);

Eq3 = Eq1 - Eq2;  % Subtract I_minus1 from I_plus1

%% Make the first set of substitutes
Eq4 = subs(Eq3, {I_prime, I_2prime, I_3prime, I_4prime, I_5prime}, {f_i, f_prime, f_2prime, f_3prime, ...
                    f_4prime});

%% Substitute central difference approximation for the second-order derivative
Eq5 = subs(Eq4, f_2prime, (f_iminus1 - 2*f_i + f_iplus1)/h^2 - 1/12*h^2*f_4prime);

%% Collect coefficients and display the result
r = collect(Eq5, h); pretty(r)

\[ e_{i-1} \approx -\frac{h^5}{90} f^{(iv)}(x_i) \]
\[ \Delta I(x_{i-1}) \approx \frac{1}{3} h \left( f(x_{i-1}) + 4f(x_i) + f(x_{i+1}) \right) \]
Cubic Interpolation

For each triple interval we have the following interpolation

\[ f^*(x) = Kx^3 + Lx^2 + Mx + N \]

Without loss of generality, assume \( x_i - x_{i-1} = x_{i+1} - x_i = h \)

The coefficients \( K, L, M, \) and \( N \) are found from

\[
\begin{bmatrix}
  x_i^3 & x_i^2 & x_i & 1 \\
  (x_i + h)^3 & (x_i + h)^2 & x_i + h & 1 \\
  (x_i + 2h)^3 & (x_i + 2h)^2 & x_i + 2h & 1 \\
  (x_i + 3h)^3 & (x_i + 3h)^2 & x_i + 3h & 1
\end{bmatrix} \begin{bmatrix} K \\ L \\ M \\ N \end{bmatrix} = \begin{bmatrix} f_i \\ f_{i+1} \\ f_{i+2} \\ f_{i+3} \end{bmatrix}
\]

Then the integral for each triple interval is

\[
\Delta I_i = \int_{x_i}^{x_{i+3}} f^*(x)\,dx + e_i = \frac{81}{4} h^4 K + 9 h^3 L + \frac{9}{2} h^2 M + 3hN + e_i
\]
%% Define symbolic variables
syms x_i f_i f_iplus1 f_iplus2 f_iplus3 h x

%% Solve for coefficients of a cubic interpolation
A = [ x_i^3  x_i^2  x_i  1;
     (x_i+h)^3  (x_i+h)^2  x_i+h  1;
     (x_i+2*h)^3  (x_i+2*h)^2  x_i+2*h  1;
     (x_i+3*h)^3  (x_i+3*h)^2  x_i+3*h  1];
b= [f_i; f_iplus1; f_iplus2; f_iplus3];

coef = A\b;

%% Compute the integral
dl = int([x^3, x^2, x, 1]*coef, x_i, x_i+3*h);

%% Display the result
pretty(dl)

\[ \Delta I(x_i) \approx \frac{3}{8} h \left( f(x_i) + 3 f(x_{i+1}) + 3 f(x_{i+2}) + f(x_{i+3}) \right) \]
\[ e_i \approx -\frac{3h^5}{80} f^{(iv)}(x_i) \]
Newton–Cotes Closed Integration Formulas

\[ \int_{x_i}^{x_i+mh} f(x) \, dx = \alpha h \left[ w_1 f(x_i) + w_2 f(x_{i+1}) + w_3 f(x_{i+2}) + \cdots + w_{m+1} f(x_{i+m}) \right] + e_i \]

\[ E(h) \approx \frac{\bar{e}}{h} \]

<table>
<thead>
<tr>
<th>Number of intervals ( m )</th>
<th>Multiplier ( \alpha )</th>
<th>Weighting factors ( w_k )</th>
<th>Order of error ( E )</th>
<th>Also known as</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1/2</td>
<td>1 1</td>
<td>( h^2 )</td>
<td>Trapezoidal Rule</td>
</tr>
<tr>
<td>2</td>
<td>1/3</td>
<td>1 4 1</td>
<td>( h^4 )</td>
<td>Simpson’s ¼ Rule</td>
</tr>
<tr>
<td>3</td>
<td>3/8</td>
<td>1 3 3 1</td>
<td>( h^4 )</td>
<td>Simpson’s ⅜ Rule</td>
</tr>
<tr>
<td>4</td>
<td>2/45</td>
<td>7 32 12 32 7</td>
<td>( h^6 )</td>
<td>Boole’s Rule</td>
</tr>
<tr>
<td>5</td>
<td>5/288</td>
<td>19 75 50 50 75 19</td>
<td>( h^6 )</td>
<td></td>
</tr>
</tbody>
</table>
Consider Trapezoidal \((m=1)\) or Simpson’s \(\frac{1}{3} (m=2)\) Rule
\[
I \approx I(h) + E(h) \quad E \sim h^{2m} f^{(2m)}(\bar{x})
\]

Compute the integral using two different steps
\[
I(h_1) + E(h_1) = I(h_2) + E(h_2) \quad E(h_1) \approx E(h_2) \left( \frac{h_1}{h_2} \right)^{2m} \quad h_1 > h_2
\]

Estimate an error for the finer step
\[
E(h_2) \approx \frac{I(h_1) - I(h_2)}{1 - (h_1 / h_2)^{2m}}
\]

Correct the value of the integral
\[
I^{MA} = I(h_2) + E(h_2) = I(h_2) + \frac{I(h_2) - I(h_1)}{(h_1 / h_2)^{2m} - 1} = \frac{(h_1 / h_2)^{2m} I(h_2) - I(h_1)}{(h_1 / h_2)^{2m} - 1}
\]

If \(h_2=\frac{1}{2}h_1\) then
\[
|E(h_2)| \approx \frac{|I(h_2) - I(h_1)|}{4^m - 1} \quad I^{MA} = \frac{4^m I(\frac{1}{2}h) - I(h)}{4^m - 1}
\]
Quadrature is a numerical method used to find the area under the graph of a function, that is, to compute a definite integral:

\[ q = \int_{a}^{b} f(x) \, dx \]
Gauss Quadrature: The Idea

\[ I_{a-b} \approx c_a f(a) + c_b f(b) \]

To calculate \( c_a \) and \( c_b \) let us require to provide exact solutions for \( f(x) \) being

i) a constant
ii) a straight line

These two conditions yield two equations

\[ c_a + c_b = \int_a^b 1 \, dx = b - a \]

\[ c_a a + c_b b = \int_a^b x \, dx = \frac{b^2 - a^2}{2} \]

Being resolved they give

\[ c_a = c_b = \frac{b - a}{2} \]

that is,

\[ I_{a-b} \approx \frac{b-a}{2} \left( f(a) + f(b) \right) \]

This is a Trapezoidal formula!
Gauss Quadrature: The Approach

\[ \int_{a}^{b} f(x) \, dx \approx c_1 f(x_1) + c_2 f(x_2) \quad \text{for} \quad a \leq x_1 \leq x_2 \leq b \]

To calculate for unknowns, \( x_1, x_2, c_1, \) and \( c_2, \) we now require to provide exact solutions for \( f(x) \) being

i) a constant

ii) a straight line

iii) a parabolic function

iv) a cubic function

These four conditions yield four equations

\[ c_1 f(x_1) + c_2 f(x_2) = c_1 + c_2 = \int_{-1}^{1} 1 \, dx = 2 \]

\[ c_1 f(x_1) + c_2 f(x_2) = c_1 x_1 + c_2 x_2 = \int_{-1}^{1} x \, dx = 0 \]

\[ c_1 f(x_1) + c_2 f(x_2) = c_1 x_1^2 + c_2 x_2^2 = \int_{-1}^{1} x^2 \, dx = \frac{2}{3} \]

\[ c_1 f(x_1) + c_2 f(x_2) = c_1 x_1^3 + c_2 x_2^3 = \int_{-1}^{1} x^3 \, dx = 0 \]

(for simplicity we scaled the original domain \([a;b]\) to that of \([-1;1]\))
%% Define symbolic variables
syms c1 c2 x1 x2

%% Define four algebraic equations
Eq1 = c1+c2-2;
Eq2 = c1*x1+c2*x2;
Eq3 = c1*x1^2+c2*x2^2-2/3;
Eq4 = c1*x1^3+c2*x2^3;

%% Solve equations
coef = solve(Eq1, Eq2, Eq3, Eq4, c1, c2, x1, x2);

%% Display the results
Weights = [coef.c1(1) coef.c2(1)];
Location = [coef.x1(1) coef.x2(1)];
pretty(Weights)
pretty(Location)

WeightsD = eval(Weights)
LocationD = eval(Location)

WeightsD = 
1 1

LocationD =
0.5774 -0.5774
The Two-Point Gauss–Legendre Formula

\[ I \approx f\left(\frac{-1}{\sqrt{3}}\right) + f\left(\frac{1}{\sqrt{3}}\right) \]

Back to the original domain

\[ \int_a^b f(x)dx = \int_{-1}^1 f(\zeta(t))\zeta'(t)dt \]

\[ x \in [a; b] \quad \quad \quad \quad t \in [-1; 1] \]

Linear mapping* between [-1;1] and [a;b] domains

\[ x = \frac{b-a}{2} t + \frac{a+b}{2} \quad \quad \quad t = \pm \frac{1}{\sqrt{3}} \]

*To handle singularities at endpoints and infinite endpoints the \texttt{quadgk} and \texttt{integral} functions use more sophisticated mapping
Could We Use Three Points?

%% Define symbolic variables
syms c1 c2 c3 x1 x2 x3

%% Define six algebraic equations
Eq1 = c1+c2+c3-2;
Eq2 = c1*x1+c2*x2+c3*x3;
Eq3 = c1*x1^2+c2*x2^2+c3*x3^2-2/3;
Eq4 = c1*x1^3+c2*x2^3+c3*x3^3;
Eq5 = c1*x1^4+c2*x2^4+c3*x3^4-2/5;
Eq6 = c1*x1^5+c2*x2^5+c3*x3^5;

%% Solve equations
coef=solve(Eq1,Eq2,Eq3,Eq4,Eq5,Eq6,...
            c1,c2,c3,x1,x2,x3);

%% Display the results
Weights = [coef.c1(1) coef.c2(1) coef.c3(1)];
Location = [coef.x1(1) coef.x2(1) coef.x3(1)];
pretty(Weights), pretty(Location)

WeightsD = eval(Weights)
LocationD = eval(Location)
Gauss–Legendre Formulas

\[ I = c_1 f(x_1) + c_2 f(x_2) + c_3 f(x_3) + \cdots + c_k f(x_k) + E \]

<table>
<thead>
<tr>
<th>Number of points ( k )</th>
<th>Weighting factors ( c_i )</th>
<th>Function arguments ( x_i )</th>
<th>Truncation error ( E )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>( c_1 = 1 ) ( c_2 = 1 )</td>
<td>( -x_1 = x_2 = 0.577350269 )</td>
<td>( f^{(iv)}(x) )</td>
</tr>
<tr>
<td>3</td>
<td>( c_1 = c_3 = 0.5555556 ) ( c_2 = 0.8888889 )</td>
<td>( -x_1 = x_3 = 0.774598889 ) ( x_2 = 0 )</td>
<td>( f^{(vi)}(x) )</td>
</tr>
<tr>
<td>4</td>
<td>( c_1 = c_4 = 0.3478548 ) ( c_2 = c_3 = 0.6521452 )</td>
<td>( -x_1 = x_4 = 0.861136312 ) ( -x_2 = x_3 = 0.339981044 )</td>
<td>( f^{(viii)}(x) )</td>
</tr>
<tr>
<td>5</td>
<td>( c_1 = c_5 = 0.2369969 ) ( c_2 = c_4 = 0.4786287 ) ( c_3 = 0.5688889 )</td>
<td>( -x_1 = x_5 = 0.906179846 ) ( -x_2 = x_4 = 0.538469931 ) ( x_3 = 0 )</td>
<td>( f^{(v)}(x) )</td>
</tr>
<tr>
<td>6</td>
<td>( c_1 = c_6 = 0.1713245 ) ( c_2 = c_5 = 0.3607616 ) ( c_3 = c_4 = 0.4679139 )</td>
<td>( -x_1 = x_6 = 0.932469514 ) ( -x_2 = x_5 = 0.661209386 ) ( -x_3 = x_4 = 0.238619186 )</td>
<td>( f^{(xii)}(x) )</td>
</tr>
</tbody>
</table>
Gauss–Legendre Points Visualization

- 2 points
- 3 points
- 4 points
- 5 points
- 6 points
The function values for the lower-order estimates (4-point Lobatto and 7-point Gauss) can be re-used to compute the higher-order estimates (with the different weights).

The difference between the higher-order and lower-order estimates

$$\left| I_{\text{higher-order}} - I_{\text{lower-order}} \right|$$

is used to estimate the local error and adjust the step (integration interval).
Syntax for MATLAB Functions

(available in R2018a)

- **T=trapz(x,y)** uses trapezoidal integration to compute the integral of **y** with with spacing increment **x**, where **y** is an array that contains the values of the function at the points contained in **x** (**x** may be a scalar as well)

- **Q=quad(fun,a,b,tol)** uses a recursive adaptive Simpson’s rule to compute the integral of function **fun** with **a** as the lower limit and **b** as the upper limit. The parameter **tol** is optional, and specifies the error tolerance desired. The default tolerance is **10^-6**

- **Q=quadl(fun,a,b,tol)** uses a recursive adaptive Lobatto integration algorithm to compute the integral of function **fun** from **a** to **b** with an error tolerance of **tol**

- **Q=quadgk(fun,a,b,par,val)** approximates the integral of function **fun** from **a** to **b** using high-order global adaptive quadrature and default error tolerances (10^-6 and 10^-10 for the relative and absolute error, respectively). **par-val** pairs allow changing the default tolerances

- **Q=quadv(fun,a,b,tol)** vectorized version of **quad** for an array-valued input function **fun**

- **Q=dblquad(fun,xmin,xmax,ymin,ymax,tol,method)** employs either the **quad** (**method** is omitted) or **quadl** (**method** is specified as **@quadl** or the function handle of a user-defined quadrature method) to evaluate a double integral of **fun** with the limits of integration of **xmin** to **xmax** and **ymin** to **ymax**

- **Q=quad2d(fun,a,b,par,val)** evaluates a double integral of **fun** using 2-D quadrature. **par-val** pairs allow changing the default tolerances, limit the maximum number of function evaluations and produce a failure plot

- **Q=triplequad(fun,xmin,xmax,ymin,ymax,zmin,zmax,tol,method)** employs either the **quad** (**method** is omitted) or **quadl** (**method** is specified as **@quadl** or 'quadl') to evaluate a triple integral of **fun** over a rectangular cuboid
Syntax for the \textit{integral} Family Functions

- $Q=\text{integral}(\text{fun},x_{\text{min}},x_{\text{max}},\text{param},\text{val})$ numerically integrates function \text{fun} from $x_{\text{min}}$ to $x_{\text{max}}$ using global adaptive quadrature and default tolerances of $10^{-10}$ and $10^{-6}$ for the absolute and relative errors, respectively. \text{fun} must be a function handle. Optional comma-separated \text{param-val} pairs allow changing error tolerances, specifying waypoints to integrate in the complex plane, and deal with array-valued or vector-valued function \text{fun}.

- $Q=\text{integral2}(\text{fun},x_{\text{min}},x_{\text{max}},y_{\text{min}},y_{\text{max}},\text{param},\text{val})$ approximates the integral of the function $f(x,y)$ over the planar region $x_{\text{min}} \leq x \leq x_{\text{max}}$ and $y_{\text{min}}(x) \leq y \leq y_{\text{max}}(x)$. Optional comma-separated \text{param-val} pairs allow changing error tolerances and choose between the usage of 1-D and 2-D quadrature (‘iterated’ or ‘tiled’ method, respectively).

- $Q=\text{integral3}(\text{fun},x_{\text{min}},x_{\text{max}},y_{\text{min}},y_{\text{max}},z_{\text{min}},z_{\text{max}},\text{param},\text{val})$ approximates the integral of the function $f(x,y,z)$ over the region $x_{\text{min}} \leq x \leq x_{\text{max}}$, $y_{\text{min}}(x) \leq y \leq y_{\text{max}}(x)$ and $z_{\text{min}}(x,y) \leq z \leq z_{\text{max}}(x,y)$. Optional comma-separated \text{param-val} pairs allow changing error tolerances and choose between the usage of 1-D and 2-D quadrature (‘iterated’ or ‘tiled’ method, respectively).
Utilizing the `cumtrapz` Function

```
subplot(211)
plot(t,accg,'.-.'), hold
acc=accg-9.8*sind(theta));  % remove the G component
plot(t,acc,'.--'), grid,
legend('G-inclusive','G-excluded','location','n')
xlabel('Time, s'), ylabel('Acceleration, m/s')

subplot(212)
plot(t,cumtrapz(t,acc)/1000,'.--','color',[0.85 0.325 0.098])
grid
xlabel('Time, s'), ylabel('Speed, km/s')
```

All rights reserved. No part of this publication may be reproduced, distributed, or transmitted, unless for course participation, in any form or by any means, or stored in a database or retrieval system, without the prior written permission of the Publisher and/or Author. Contact the American Institute of Aeronautics and Astronautics, Professional Development Programs, 12700 Sunrise Valley Drive, Suite 200, Reston, VA 20191-5807.
Evaluating Single Integrals with `trapz` and `quad`

```matlab
>> syms x
>> f=int(sin(x));
>> fun=inline(char(f));
>> I=fun(pi)-fun(0)
I =
    2

>> x=linspace(0,pi,9); y=sin(x);
>> ITR=trapz(x,y)
>> fprintf('Trapezoidal rule error is %3.2f \n',abs(ITR-2)/2*100)
>> Eest=abs((pi/9)^2*pi/12)
ITR =
    1.9797
Trapezoidal rule error is 1.02 %
Eest =
    0.0319

numerical solution using `trapz`

```matlab
>> [qG,nG]=quad(@sin,0,pi,1e-10);
>> [qL,nL]=quadl(@sin,0,pi,1e-10);
>> disp([qG,nG; qL,nL])
2.0000   225.0000
2.0000    48.0000

numerical solution using `quad`, `quadl`, `quadgk`

```matlab
>> [qK,errK]=quadgk(@sin,0,pi)
qK =
    2.0000
errK =
    1.4244e-16
```

```
exact solution
\int_0^\pi \sin(x) \, dx
```

All rights reserved. No part of this publication may be reproduced, distributed, or transmitted, unless for course participation, in any form or by any means, or stored in a database or retrieval system, without the prior written permission of the Publisher and/or Author. Contact the American Institute of Aeronautics and Astronautics, Professional Development Programs, 12700 Sunrise Valley Drive, Suite 200, Reston, VA 20191-5807.
>> I=quadl('sin(x).*cos(x)',0,pi/2)
I =
    0.5000

>> I=quadl(vectorize('sin(x)*cos(x)'),0,pi/2)
I =
    0.5000

>> fun = @(x) [tan(x) sin(x); sin(x)^2 cos(x)];
>> I=quadv(fun,0,pi/2)
I =
    39.1458    1.0000
    0.7854    1.0000

>> [q,err]=quadgk(@(x)exp(-x/4).*sin(x),0,inf,'RelTol',1e-8,'AbsTol',1e-12)
q =
    0.9412
err =
    1.8575e-09

\[
|I_i - I_{i-1}| \leq \max(\epsilon^{\text{abs}}, \epsilon^{\text{rel}} |I_i|)
\]
Using the `integral` Function

```matlab
>> which integral
C:\...\R2019b\toolbox\matlab\funfun\integral.m

>> which('integralCalc','-all')
C:\...\R2019b\toolbox\matlab\funfun\private\integralCalc.m % Private to funfun

>> I=integral(@(x) 1./x.^2,1,inf)
I =
1

>> I=integral(@(x) exp(-x.^2).*log(x).^2,0,inf,'RelTol',1e-8,'AbsTol',1e-12)
I =
1.9475

>> I=integral(@(z) 1./z,1,1,'Waypoints',[i,-1,-i])
I =
0.0000 + 6.2832i

>> I=integral(@(x) cos((1:4)*x).*sin((4:-1:1)*x),0,1,'ArrayValued',true)
I =
0.4033 0.3015 -0.1582 -0.2600

All rights reserved. No part of this publication may be reproduced, distributed, or transmitted, unless for course participation, in any form or by any means, or stored in a database or retrieval system, without the prior written permission of the Publisher and/or Author. Contact the American Institute of Aeronautics and Astronautics, Professional Development Programs, 12700 Sunrise Valley Drive, Suite 200, Reston, VA 20191-5807.
Computing Double Integrals

\[ \int_0^2 \int_1^2 (6 - x^3 + xy^3) \, dy \, dx \]

\[
\begin{split}
\text{>> I=double(@(x,y)6-x.^3+x.*y.^3,0,2,1,2)} \\
I &= 15.5
\end{split}
\]

\[
\begin{split}
\text{>> I=quad2d(@(x,y)6-x.^3+x.*y.^3,0,2,1,2)} \\
I &= 15.5
\end{split}
\]

\[
\begin{split}
\text{>> I=integral2(@(x,y)6-x.^3+x.*y.^3,0,2,1,2,'Method','iterated');} \\
\text{>> I=integral2(@(x,y)6-x.^3+x.*y.^3,0,2,1,2,'Method','tiled');} \\
\text{>> I=integral2(@(x,y)6-x.^3+x.*y.^3,0,2,1,2,'Method','iterated','RelTol',0,'AbsTol',1e-12);}
\end{split}
\]

\[ |I_i - I_{i-1}| \leq \max \left( \varepsilon^{abs}, \varepsilon^{rel} |I_i| \right) \]
Experimenting with quad2d

\[ \int_{-\pi}^{\pi} \int_{-\pi}^{\pi} \left( e^{-0.05(x^2+y^2)} \sin(x) \cos(2y) + 0.5 \right) dy dx \]

```matlab
>> F=@(x,y)exp(-0.05*(x.^2+y.^2)).*sin(x).*cos(2*y)+0.5;
>> I=quad2d(F,-2*pi,2*pi,-pi,pi,...
    'MaxFunEvals',11,'FailurePlot',true);
Warning: Reached the maximum number of function evaluations (11). The result fails the global error test.
I =
    15.5
```

```matlab
>> I=quad2d(F,-2*pi,2*pi,@(x)-abs(x),@(x)x.^2,...
    'MaxFunEvals',250,'FailurePlot',true);
Warning: Reached the maximum number of function evaluations (250). The result fails the global error test.
I =
    15.5
```
Examples of Computing Triple Integrals

\[
\iiint_{V} (y \sin(x) + z \cos(x)) \, dz \, dy \, dx
\]

\[
\begin{align*}
\mathbf{F} &= @(x,y,z) \ y \cdot \sin(x) + z \cdot \cos(x); \\
\mathbf{I} &= \text{triplequad} (\mathbf{F},0,\pi,0,1,-1,1) \\
\mathbf{I} &= 2.0000
\end{align*}
\]

\[
\begin{align*}
\mathbf{F} &= @(x,y,z) \ y \cdot \sin(x) + z \cdot \cos(x); \\
\mathbf{I} &= \text{integral3} (\mathbf{F},0,\pi,0,1,-1,1,'Method','iterated'); \\
\mathbf{I} &= \text{integral3} (\mathbf{F},0,\pi,0,1,-1,1,'Method','tiled'); \\
\mathbf{I} &= \text{integral3} (\mathbf{F},0,\pi,0,1,-1,1,'RelTol',1e-4,'AbsTol',1e-6); \\
\end{align*}
\]

specifying a method of integration and tolerances

\[
\begin{align*}
\mathbf{I} &= \text{integral3} (\mathbf{F},-1,1,(@(x)x, @(x)2*x, @(x,y)x+y, @(x,y)x.^2+y.^2); \\
\end{align*}
\]
Volume of a Solid of Revolution

\[ \int_0^5 \pi r^2 \, dz = \int_0^5 \pi \left( 2 + \cos \left( \frac{\pi}{2} + \frac{2\pi - 0.5\pi}{5} \right) \right)^2 \, dz \]

\[ z = \text{linspace}(\pi/2, 2\times\pi); \quad [X, Y, Z] = \text{cylinder}(2 + \cos(z)); \]
\[ \text{surf}(X, Y, 5\times Z, X) \]
\[ \text{axis equal}, \quad \text{colormap hsv}, \quad \text{shading interp} \]
\[ \text{xlabel('x')}, \quad \text{ylabel('y')}, \quad \text{zlabel('5*z')} \]

\[ f = @(x) \pi \times (2 + \cos(\pi/2 + x \times (2\pi - \pi/2)/5)).^2; \]
\[ z = \text{linspace}(0, 5, 7); \quad s = f(z); \quad I = \text{quad}(f, 0, 5); \quad \text{ITR} = \text{trapz}(z, s); \quad h = z(2) - z(1); \]
\[ \text{IS13} = h \times (s(1) + 4 \times s(2) + 2 \times s(3) + 4 \times s(4) + 2 \times s(5) + 4 \times s(6) + s(7))/3; \]
\[ \text{IS38} = 3 \times h \times (s(1) + 3 \times s(2) + 3 \times s(3) + 3 \times s(4) + 3 \times s(5) + 3 \times s(6) + s(7))/8; \]

\[
\begin{array}{l}
\text{fprintf(' Method | Value | Rel. Error \n')}
\text{fprintf('Trapezoidal | %4.2f | %3.2f \%%', ITR, abs(I - ITR)/I*100)}
\text{fprintf('Simpson 1/3 | %4.2f | %3.2f \%%', IS13, abs(IS13 - I)/I*100)}
\text{fprintf('Simpson 3/8 | %4.2f | %3.2f \%%', IS38, abs(IS38 - I)/I*100)}
\text{fprintf('Exact value | %4.2f | \n', I)}
\end{array}
\]

<table>
<thead>
<tr>
<th>Method</th>
<th>Value</th>
<th>Rel. Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trapezoidal</td>
<td>58.05</td>
<td>1.21 %</td>
</tr>
<tr>
<td>Simpson 1/3</td>
<td>57.32</td>
<td>0.05 %</td>
</tr>
<tr>
<td>Simpson 3/8</td>
<td>58.92</td>
<td>2.73 %</td>
</tr>
<tr>
<td>Exact value</td>
<td>57.35</td>
<td></td>
</tr>
</tbody>
</table>
Using the `polyint` Function

r=[1,2,5];
a=poly(r);
b1=polyint(a);
C=10; b2=polyint(a,C);
x=linspace(min(r),max(r),30);
plot(x,polyval(a,x),'bo-.'), hold on
plot(r,polyval(a,r),'Marker','p','MarkerFaceColor','r','MarkerSize',15)
plot(x,polyval(b1,x),'+m'), plot(x,polyval(b2,x),'cv'), hold off, grid
legend('Polynomial','Roots','Integral with C=0', ['Integral with C=', int2str(C)],...
'Location','Best')

### Function Description

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>conv</code></td>
<td>computes a product of two polynomials</td>
</tr>
<tr>
<td><code>deconv</code></td>
<td>performs a division of two polynomials (returns the quotient and remainder)</td>
</tr>
<tr>
<td><code>poly</code></td>
<td>creates a polynomial with specified roots</td>
</tr>
<tr>
<td><code>polyder</code></td>
<td>calculates the derivative of polynomials analytically</td>
</tr>
<tr>
<td><code>polyeig</code></td>
<td>solves polynomial eigenvalue problem</td>
</tr>
<tr>
<td><code>polyfit</code></td>
<td>produces polynomial curve fitting</td>
</tr>
<tr>
<td><code>polyint</code></td>
<td>integrates polynomial analytically</td>
</tr>
<tr>
<td><code>polyval</code></td>
<td>evaluates polynomial at certain points</td>
</tr>
<tr>
<td><code>polyvalm</code></td>
<td>evaluates a polynomial in a matrix sense</td>
</tr>
<tr>
<td><code>residue</code></td>
<td>converts between partial fraction expansion and polynomial coefficients</td>
</tr>
<tr>
<td><code>roots</code></td>
<td>finds polynomial roots</td>
</tr>
<tr>
<td><code>poly2sym</code></td>
<td>converts a vector of polynomial coefficients to a symbolic polynomial</td>
</tr>
<tr>
<td><code>sym2poly</code></td>
<td>converts polynomial coefficients to a vector of polynomial coefficients</td>
</tr>
</tbody>
</table>
The End of Chapter 13

Questions?