Introduction to Scientific and Engineering Computing, BIL108E

Karaman

INTRODUCTION TO SCIENTIFIC & ENGINEERING COMPUTING BIL 108E, CRN24023

Dr. S. Gökhan Karaman

Technical University of Istanbul

April 12, 2010

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

Tentative Course Schedule, CRN 24023

Week	Date	Topics
1	Feb. 08	Introduction to Scientific and Engineering Computing
2	Feb. 15	Introduction to Program Computing Environment
3	Feb. 22	Variables, Operations and Simple Plot
4	Mar. 01	Algorithms and Logic Operators
5	Mar. 08	Flow Control, Errors and Source of Errors
6	Mar. 15	Functions
6	Mar. 20	Exam 1
7	Mar. 22	Arrays
8	Mar. 29	Solving of Simple Equations
9	Apr. 05	Polynomials Examples
10	Apr. 12	Applications of Curve Fitting
11	Apr. 19	Applications of Interpolation
11	**Apr. 18**	**Exam 2**
12	Apr. 26	Applications of Numerical Integration
13	May 03	Symbolic Mathematics
14	May 10	Ordinary Differential Equation (ODE) Solutions with Built-in Functions

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

LECTURE # 10

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

NUMERICAL APPROXIMATION

NUMERICAL INTEGRATION AND DIFFERENTIATION

- To integrate a generic function, it is not possible to find a closed form of the primitive function.
- When a primitive is known, its use might not be easy.

$$f(x) = cos(4x)\,cos(3sin(x))$$

$$\int_0^{\pi} f(x)\,dx = \pi\left(\frac{3}{2}\right)\sum_{k=0}^{\infty}\frac{(-9/4)^k}{k!(k+4)!}$$

- Calculation on experimental measurements.
- **Use numerical methods to approximate the differentiation or integration.**

APPROXIMATION OF FUNCTION DERIVATIVES

- Consider a function $f : [a, b] \longrightarrow \mathbb{R}$
- Find an approximation of the first derivative(f') of f at a generic point $\bar{x}$ in interval (a, b).

$$\Delta f^{+}(\bar{x}) = \frac{f(\bar{x}+h) - f(\bar{x})}{h}$$

is an approximation of $f'(\bar{x})$, for h sufficiently small and positive h.

- The above approximation is defined as **FORWARD FINITE DIFFERENCE**.

APPROXIMATION OF FUNCTION DERIVATIVES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

APPROXIMATION OF FUNCTION DERIVATIVES

- To estimate the error, check the difference between the real value and approximation
- With using Taylor series

$$f(\bar{x}+h) = f(\bar{x}) + h f'(\bar{x}) + \frac{h^2}{2} f''(\xi)$$

Here ξ is in the interval $(\bar{x}, \bar{x}+h)$

- Then the forward finite difference is

$$\Delta f^{+}(\bar{x}) = f'(\bar{x}) + \frac{h}{2} f''(\xi)$$

- $\Delta f^{+}(\bar{x})$ is a **first order approximation** of $f'(\bar{x})$

APPROXIMATION OF FUNCTION DERIVATIVES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

APPROXIMATION OF FUNCTION DERIVATIVES

- With a similar procedure for a sufficiently small and negative h .

$$\Delta f^{-}(\bar{x}) = \frac{f(\bar{x}) - f(\bar{x}-h)}{h}$$

- This is called **BACKWARD FINITE DIFFERENCE**

APPROXIMATION OF FUNCTION DERIVATIVES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

APPROXIMATION OF FUNCTION DERIVATIVES

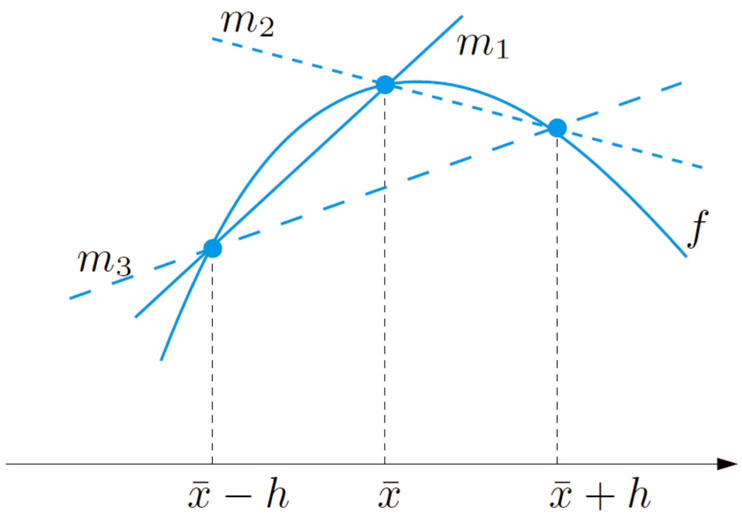
- CENTERED FINITE DIFFERENCE

$$\Delta f(\bar{x}) = \frac{f(\bar{x}+h) - f(\bar{x}-h)}{2h}$$

- This formula provides **second –order approximation**
- Error estimation

$$f'(\bar{x}) - \Delta f(\bar{x}) = \frac{h^2}{12}(f'''(\xi) + f'''(\eta))$$

APPROXIMATION OF FUNCTION DERIVATIVES



APPROXIMATION OF FUNCTION DERIVATIVES

- When $\bar{x} = x_i$ and $x_i = x_0 + i\,h$
  with $h > 0$, $f'(x_i)$ is approximated with
  - FORWARD FINITE DIFFERENCE
  - BACKWARD FINITE DIFFERENCE
  - CENTERED FINITE DIFFERENCE

APPROXIMATION OF FUNCTION DERIVATIVES

- **Note:** With the centered finite difference approximation, the centered formula cannot be used at beginning and ending points of interval. For this points use

  $\frac{1}{2h}[-3f(x_0) + 4f(x_1) - f(x_2)]$ at x_0

  $\frac{1}{2h}[3f(x_n) - 4f(x_{n-1}) + f(x_{n-2})]$ at x_n

EXAMPLE:

- The height $q(t)$ reached at time t by a fluid in a straight cylinder of radius $R = 1m$ with a circular hole of radius $r = 0.1m$ on the bottom, has been measured every 5 seconds yielding the following values

t	0.0	5.0	10.0	15.0	20.0
q(t)	0.6350	0.5336	0.4410	0.3572	0.2822

We want to compute an approximation of the emptying velocity $q(t)$ of the cylinder, then compare it with the one predicted by Torricelli's law: $q'(t) = -\gamma(r/R)^2\sqrt{g\,q(t)}$,

where g is the gravitational acceleration and $\gamma = 0.6$ is a correction factor.

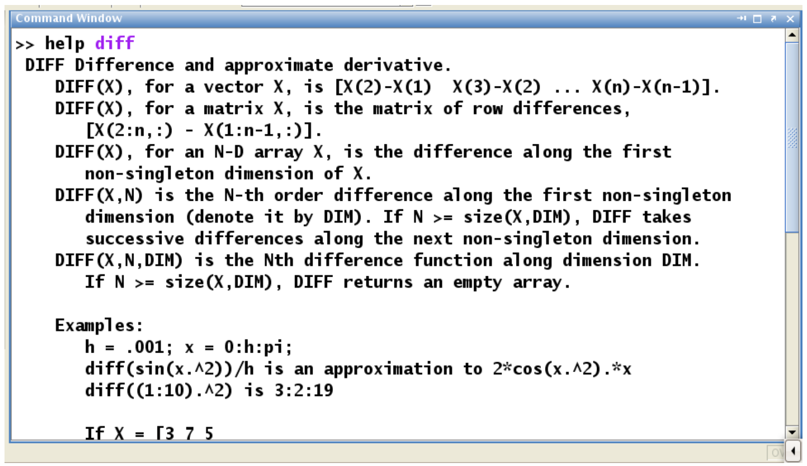
EXAMPLE:

t	0.0	5.0	10.0	15.0	20.0
q(t)	0.6350	0.5336	0.4410	0.3572	0.2822
$q'(t)$	-0.0212	-0.0194	-0.0176	-0.0159	-0.0141
Δq^{+}	-0.0203	-0.0185	-0.0168	-0.0150	
Δq^{-}		-0.0203	-0.0185	-0.0168	-0.0150
Δq		-0.0194	-0.0176	-0.0159	

MATLAB FUNCTIONS

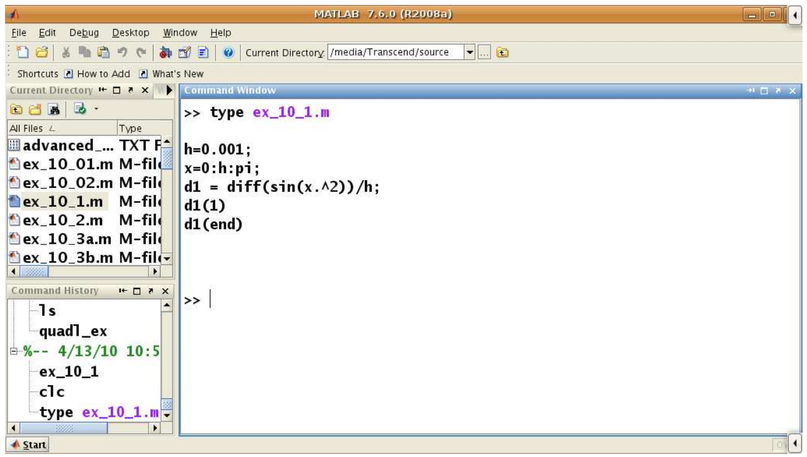
- `diff`
- `diff(y) ./ diff(x)`

MATLAB FUNCTIONS cont'd.



```
>> help diff
 DIFF Difference and approximate derivative.
    DIFF(X), for a vector X, is [X(2)-X(1)  X(3)-X(2) ... X(n)-X(n-1)].
    DIFF(X), for a matrix X, is the matrix of row differences,
       [X(2:n,:) - X(1:n-1,:)].
    DIFF(X), for an N-D array X, is the difference along the first
       non-singleton dimension of X.
    DIFF(X,N) is the N-th order difference along the first non-singleton
       dimension (denote it by DIM). If N >= size(X,DIM), DIFF takes
       successive differences along the next non-singleton dimension.
    DIFF(X,N,DIM) is the Nth difference function along dimension DIM.
       If N >= size(X,DIM), DIFF returns an empty array.

    Examples:
       h = .001; x = 0:h:pi;
       diff(sin(x.^2))/h is an approximation to 2*cos(x.^2).*x
       diff((1:10).^2) is 3:2:19

       If X = [3 7 5
```


EXAMPLE :

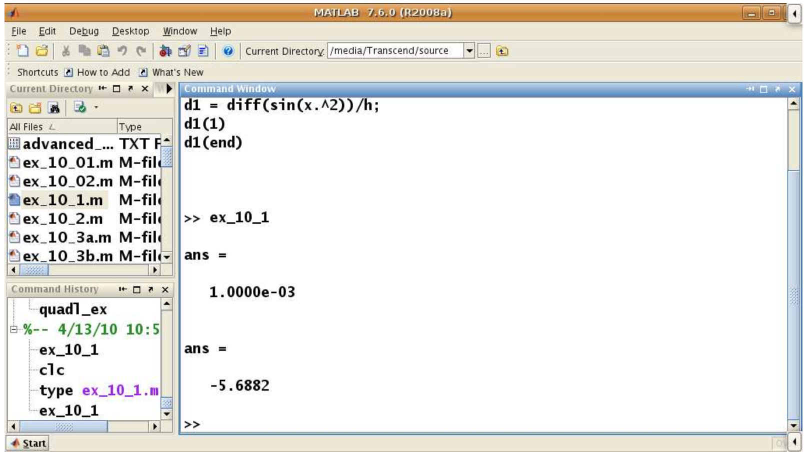


APPROXIMATION OF FUNCTION DERIVATIVES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

EXAMPLE :

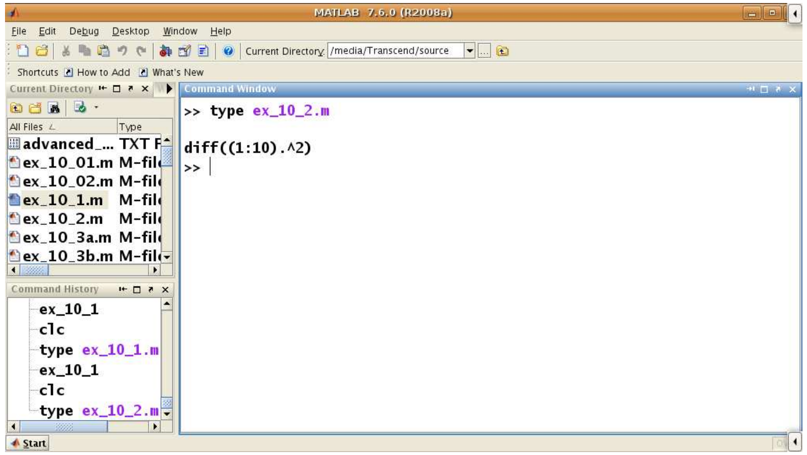


APPROXIMATION OF FUNCTION DERIVATIVES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

EXAMPLE :

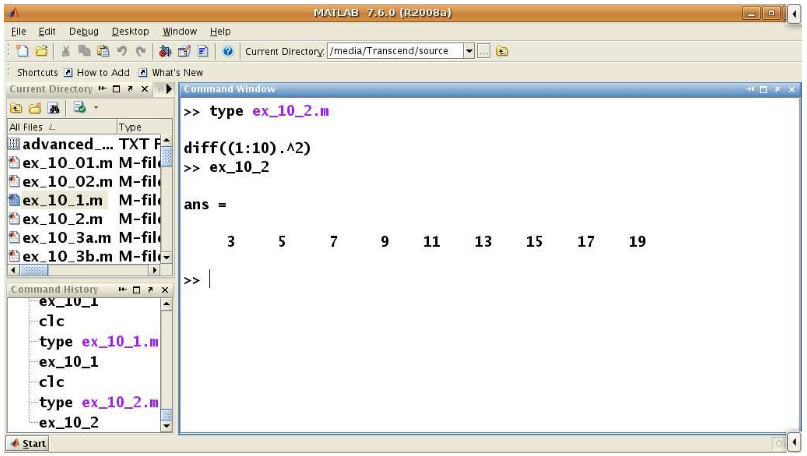


APPROXIMATION OF FUNCTION DERIVATIVES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

EXAMPLE :

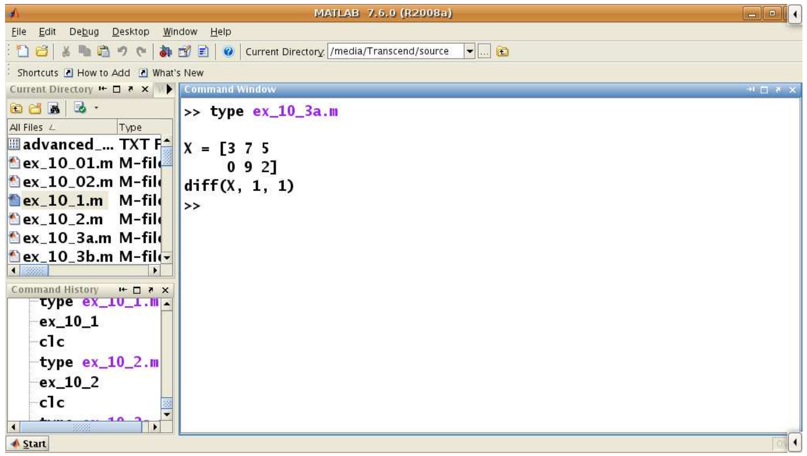


APPROXIMATION OF FUNCTION DERIVATIVES

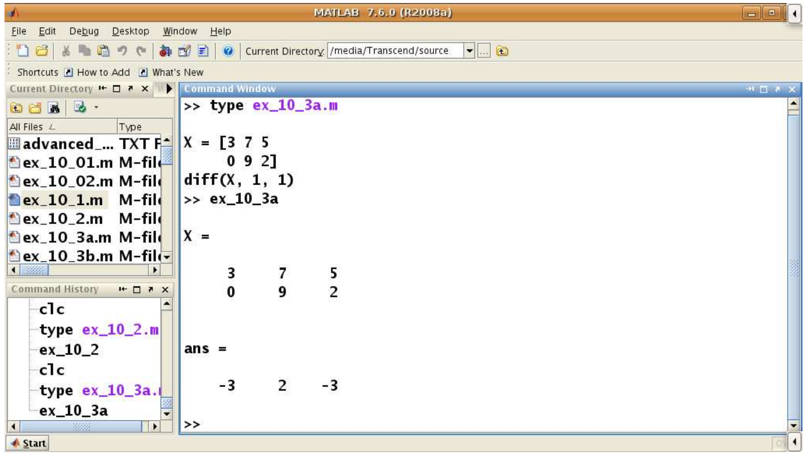
Introduction to Scientific and Engineering Computing, BIL108E

Karaman

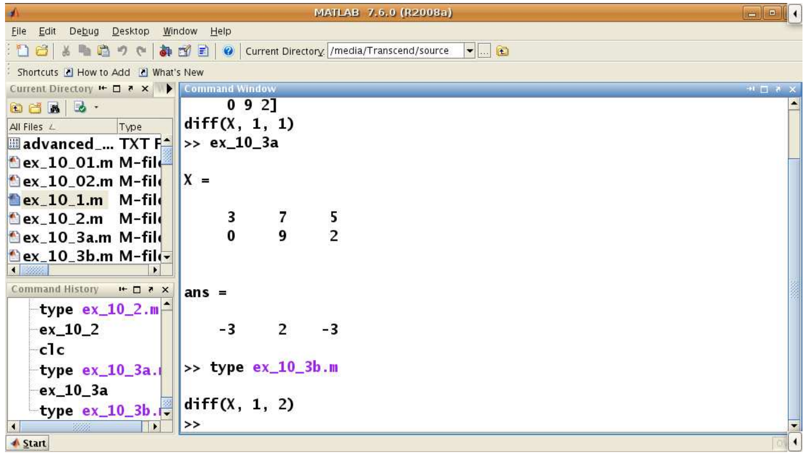
EXAMPLE :



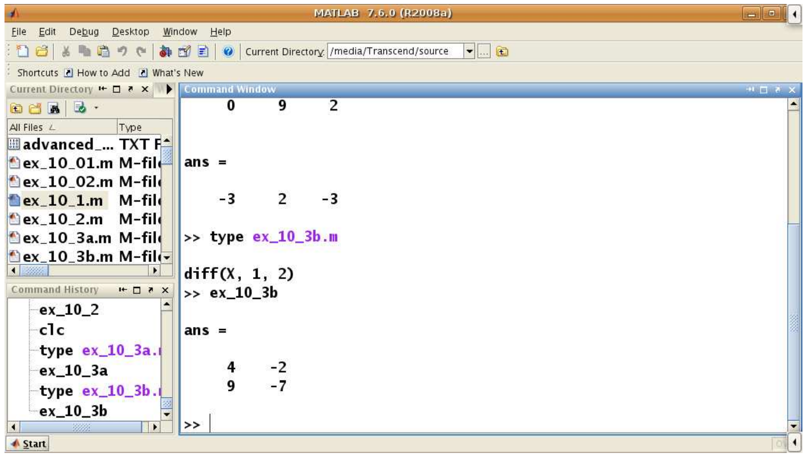
EXAMPLE :



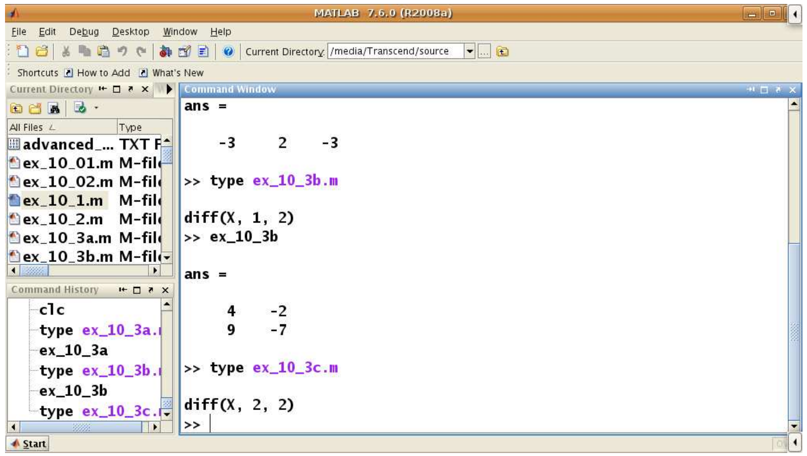
EXAMPLE :



EXAMPLE :



EXAMPLE :

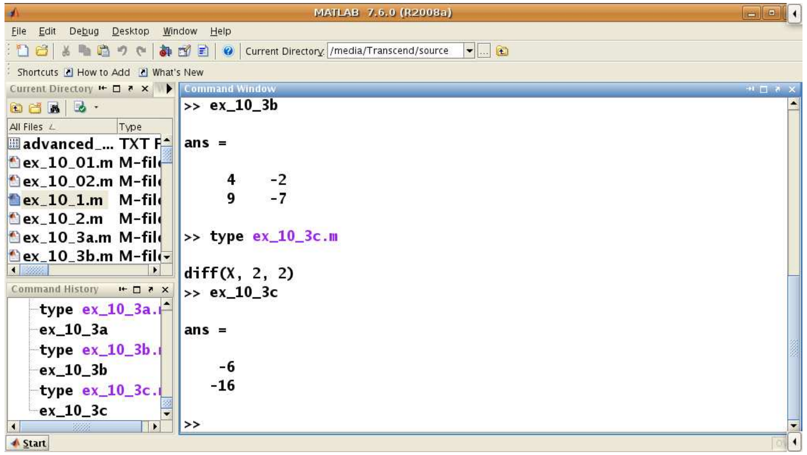


APPROXIMATION OF FUNCTION DERIVATIVES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

EXAMPLE :

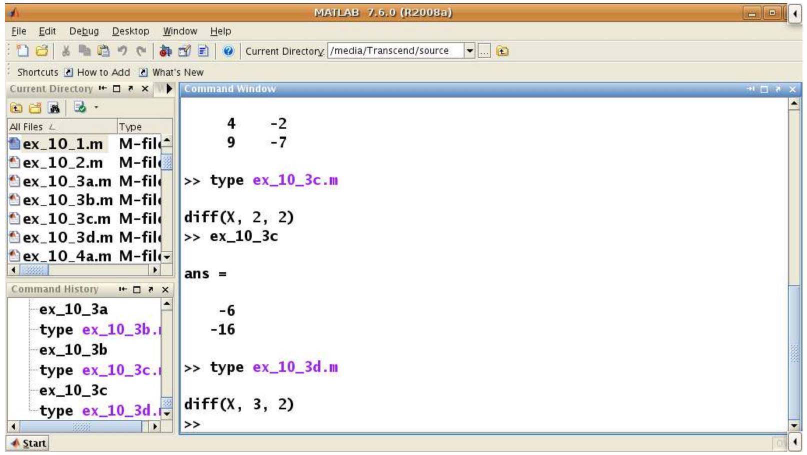


APPROXIMATION OF FUNCTION DERIVATIVES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

EXAMPLE :

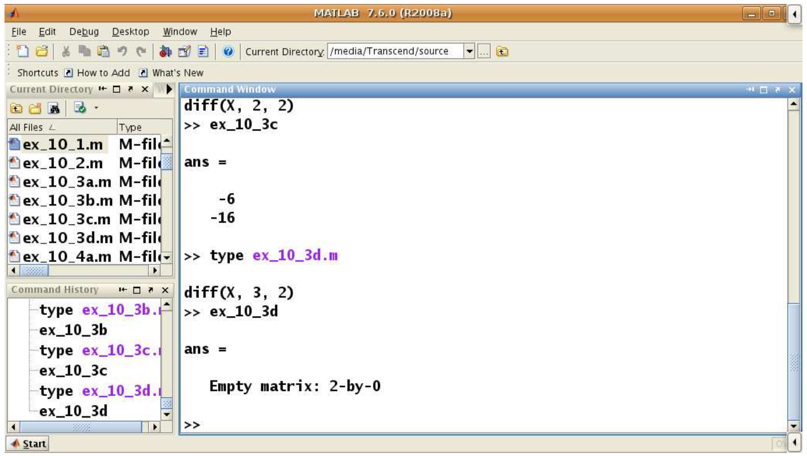


APPROXIMATION OF FUNCTION DERIVATIVES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

EXAMPLE :

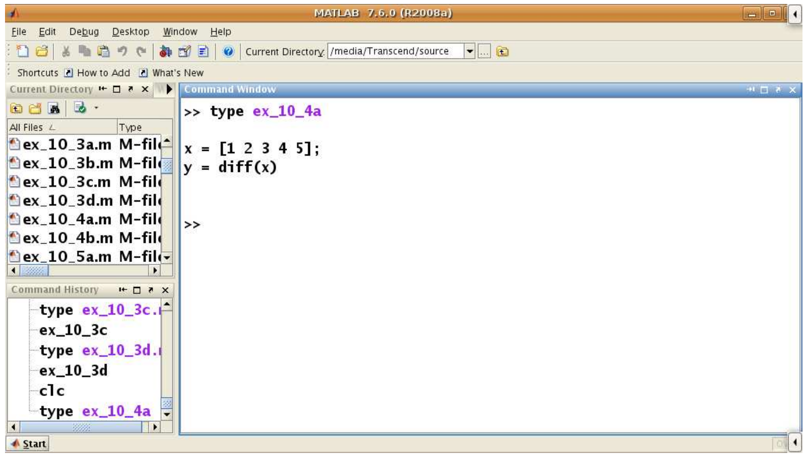


APPROXIMATION OF FUNCTION DERIVATIVES

Introduction to Scientific and Engineering Computing, BIL108E

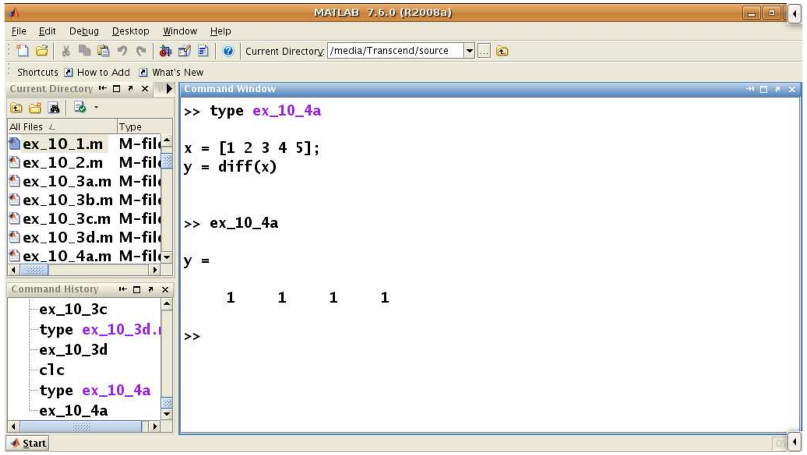
Karaman

EXAMPLE :



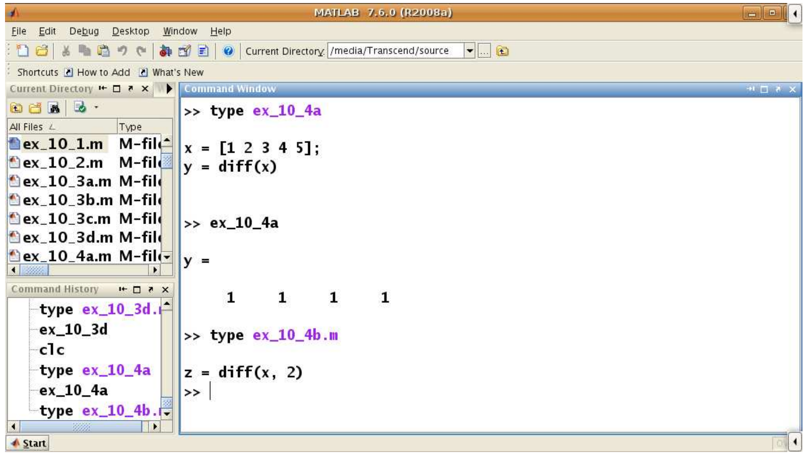
APPROXIMATION OF FUNCTION DERIVATIVES

EXAMPLE :



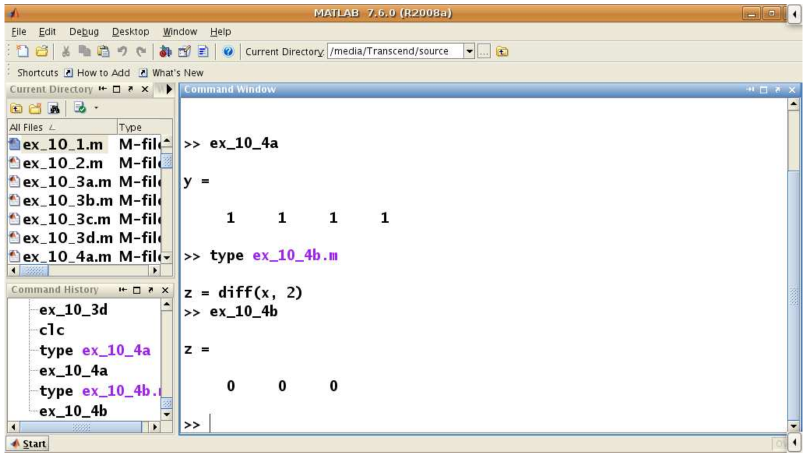
APPROXIMATION OF FUNCTION DERIVATIVES

EXAMPLE :



APPROXIMATION OF FUNCTION DERIVATIVES

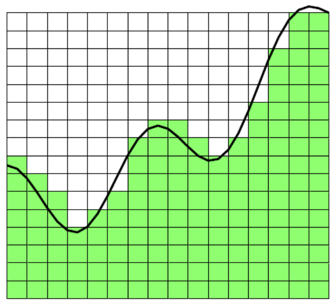
EXAMPLE :



APPROXIMATION OF FUNCTION INTEGRALS

QUADRATURE

- The word "quadrature" reminds us an elementary technique for finding the area under the curve.
- Plot the function on graph paper and **count the number of little squares** that lie underneath the curve.



Area under the curve is counted / calculated.

APPROXIMATION OF INTEGRALS

- Numerical Methods for approximating the integral

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

- Here f is an arbitrary continuous function

APPROXIMATION OF INTEGRALS

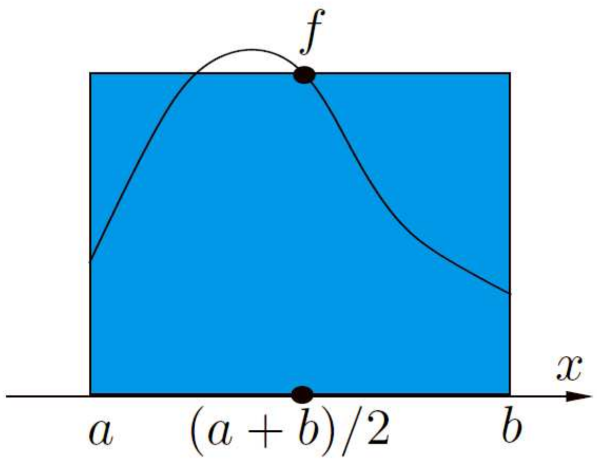
- Midpoint Quadrature
- Trapezoidal Quadrature
- Simpson Quadrature
- Gauß–Legendre Formula
- Adaptive Simpson Formula

APPROXIMATION OF INTEGRALS

Newton–Cotes equation

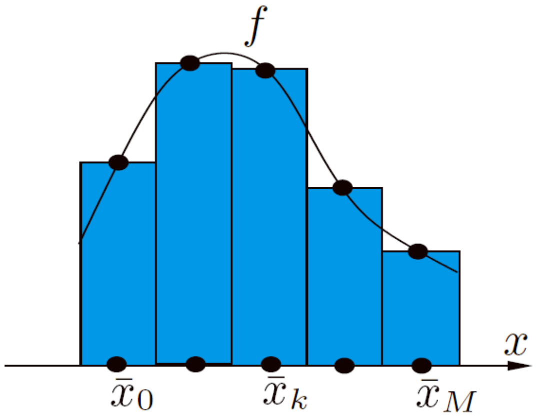
- Define the function $f(x)$ as an approximation with polynom $P(x)$, and use it on an equally partitioned interval (a, b).
- Calculation with this method is also named as **composite** quadrature.

MIDPOINT QUADRATURE



APPROXIMATION OF INTEGRALS

MIDPOINT QUADRATURE



APPROXIMATION OF INTEGRALS

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

MIDPOINT QUADRATURE

- Approximate the integral $I(f)$ for the interval $[a, b]$
- Divide the interval $I_k = [x_{k-1}, x_k]$ for $k = 1, \ldots, M$ into subintervals.
- $x_k = a + k\,H$, $k = 0, \ldots, M$ and $H = (b-a)/M$

$$I(f) = \sum_{k=1}^{M} \int_{I_k} f(x)\, dx$$

APPROXIMATION OF INTEGRALS

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

MIDPOINT QUADRATURE

- Approximate the function f with a polynomial $\bar{f}$ on I_k
- $\bar{x}_k = \frac{x_{k-1} - x_k}{2}$
- $I^c_{mp}(f) = H \sum_{k=1}^{M} f(\bar{x}_k)$
  This is called
  COMPOSITE MIDPOINT QUADRATURE
- Second –order approximate with respect to H

APPROXIMATION OF INTEGRALS

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

CLASSIC MIDPOINT FORMULA

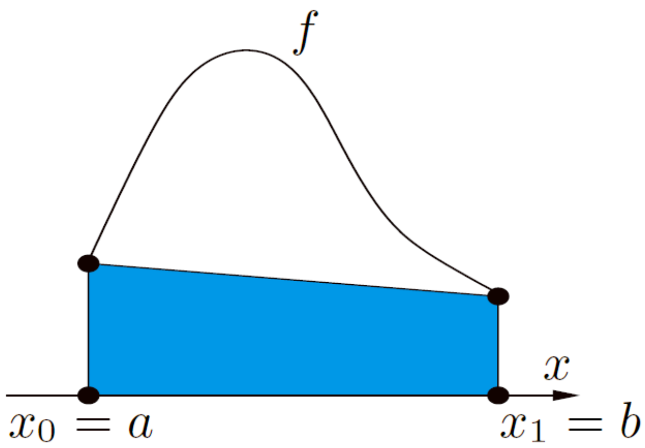
- Here the number of partitions M=1.

$$I_{mp}(f) = (b-a) f((a+b)/2)$$

- Estimated error,

$$I(f) - I_{mp}(f) = \frac{(b-a)^3}{24} f''(\xi)$$

TRAPEZOIDAL QUADRATURE

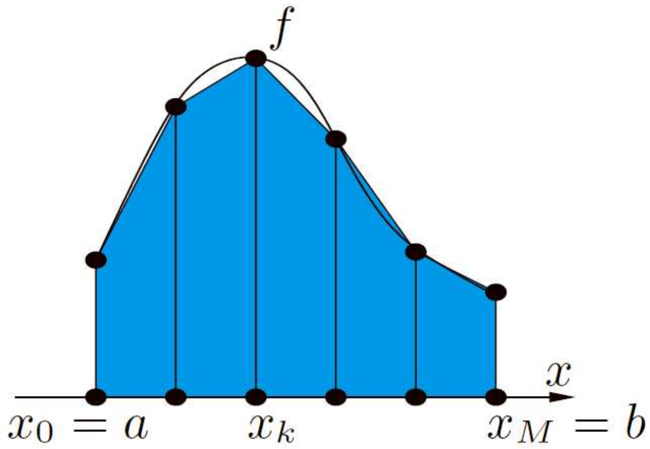


APPROXIMATION OF INTEGRALS

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

TRAPEZOIDAL QUADRATURE



APPROXIMATION OF INTEGRALS

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

TRAPEZOIDAL QUADRATURE

Calculation is done with the area of a trapezoidal.

$$I_t^c(f) = \frac{H}{2}\sum_{k=1}^{M}(f(x_k)+f(x_{k-1})) = \frac{H}{2}(f(a)+f(b))+H\sum_{k=1}^{M-1} f(x_k)$$

$$I_t(f) = \frac{b-a}{2}(f(a)+f(b))$$

APPROXIMATION OF INTEGRALS

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

SIMPSON QUADRATURE

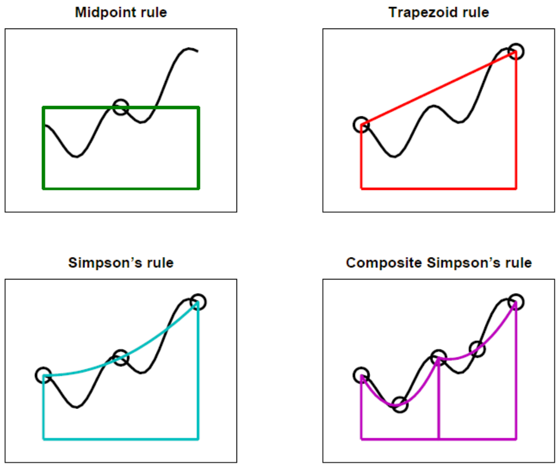
Approximate the function by a parabola. This rule can be applied to the even number of segments (odd number of points).

$$I_s^c(f) = \frac{H}{6}\sum_{k=1}^{M}(f(x_{k-1}) + 4f(\bar{x}_k) + f(x_k))$$

$$I_s(f) = \frac{b-a}{6}(f(a) + 4f((a+b)/2) + f(b))$$

APPROXIMATION OF INTEGRALS

APPROXIMATION OF INTEGRALS



APPROXIMATION OF INTEGRALS

INTERPOLATORY QUADRATURES

GAUß–LEGENDRE FORMULA

$$I_{appr}(f) = \sum_{j=0}^{n} \alpha_j f(y_j)$$

- α_j: quadrature weights
- y_j: quadrature nodes

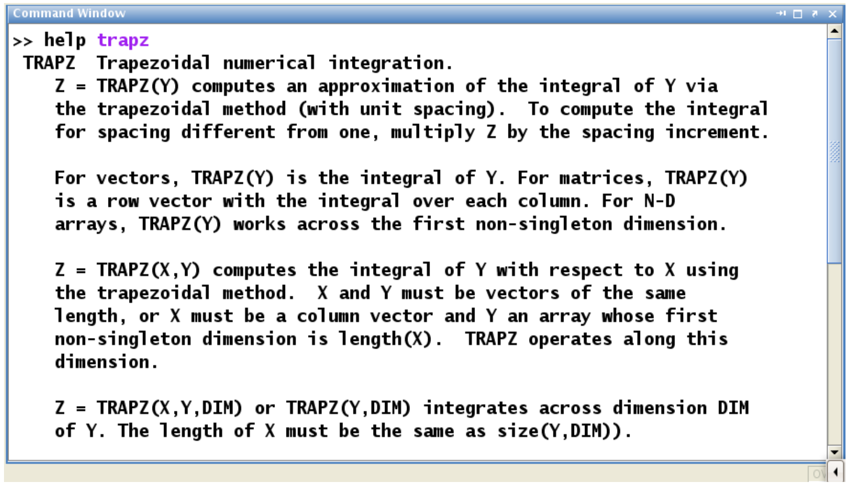
APPROXIMATION OF INTEGRALS

MATLAB FUNCTIONS

- `trapz` : Uses areas of trapezoidals.
- `cumtrapz` : Uses composite trapezoidal quadrature
- `quad` : Uses the adaptive Simpson quadrature algorithm.
- `quadl` : Uses Gauß–Legendre Formula

APPROXIMATION OF INTEGRALS

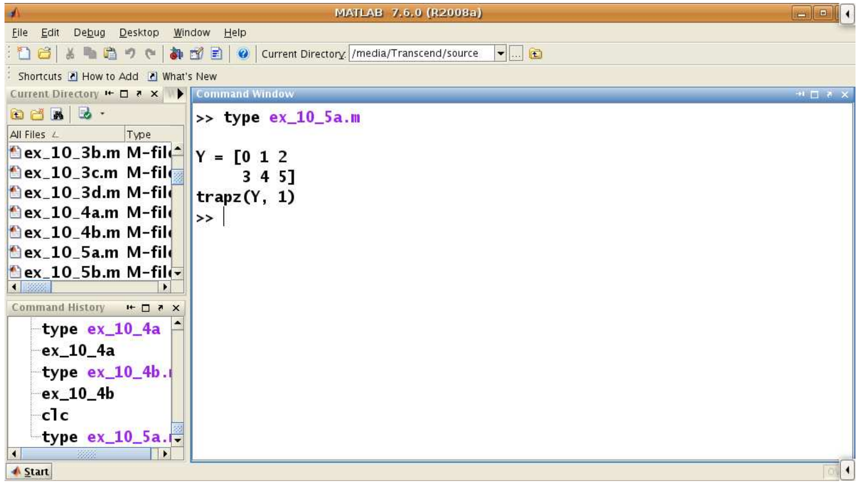
`trapz`


Command Window

```
>> help trapz
 TRAPZ  Trapezoidal numerical integration.
    Z = TRAPZ(Y) computes an approximation of the integral of Y via
    the trapezoidal method (with unit spacing).  To compute the integral
    for spacing different from one, multiply Z by the spacing increment.

    For vectors, TRAPZ(Y) is the integral of Y. For matrices, TRAPZ(Y)
    is a row vector with the integral over each column. For N-D
    arrays, TRAPZ(Y) works across the first non-singleton dimension.

    Z = TRAPZ(X,Y) computes the integral of Y with respect to X using
    the trapezoidal method.  X and Y must be vectors of the same
    length, or X must be a column vector and Y an array whose first
    non-singleton dimension is length(X).  TRAPZ operates along this
    dimension.

    Z = TRAPZ(X,Y,DIM) or TRAPZ(Y,DIM) integrates across dimension DIM
    of Y. The length of X must be the same as size(Y,DIM)).
```

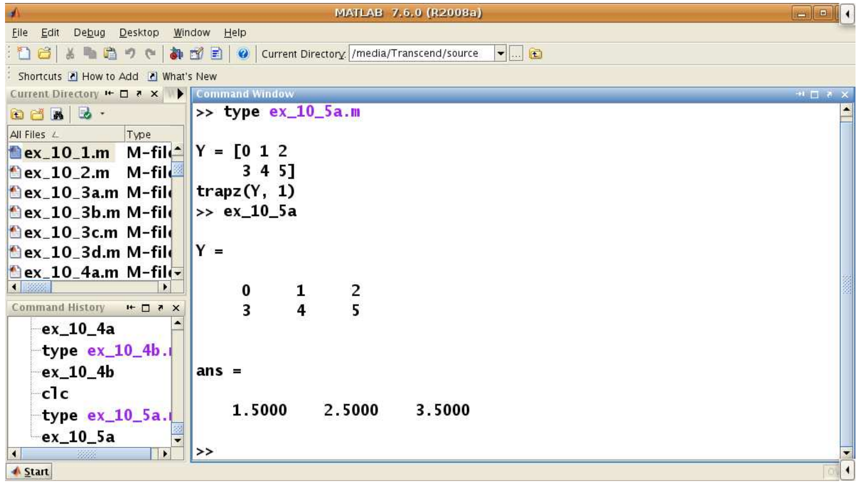
APPROXIMATION OF INTEGRALS

trapz



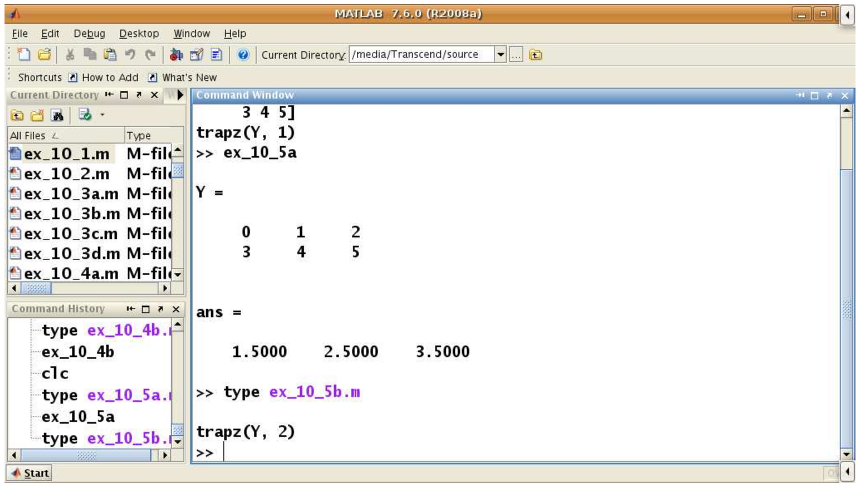
APPROXIMATION OF INTEGRALS

trapz



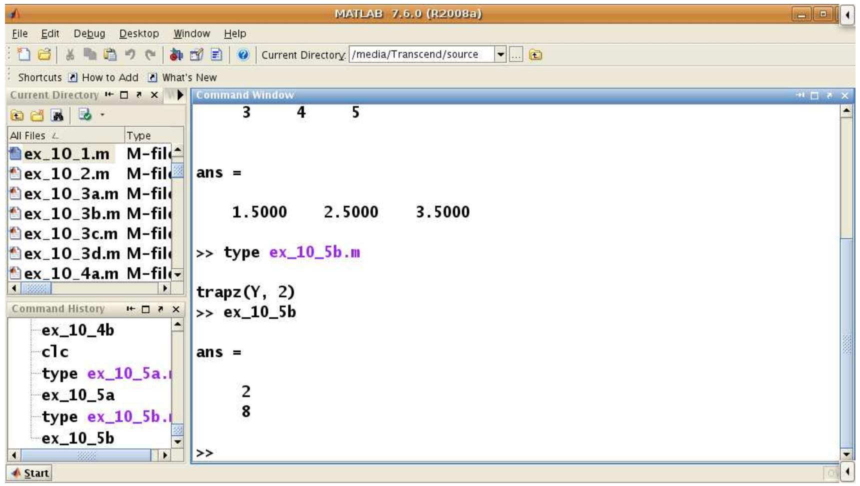
APPROXIMATION OF INTEGRALS

trapz



APPROXIMATION OF INTEGRALS

trapz



cumtrapz

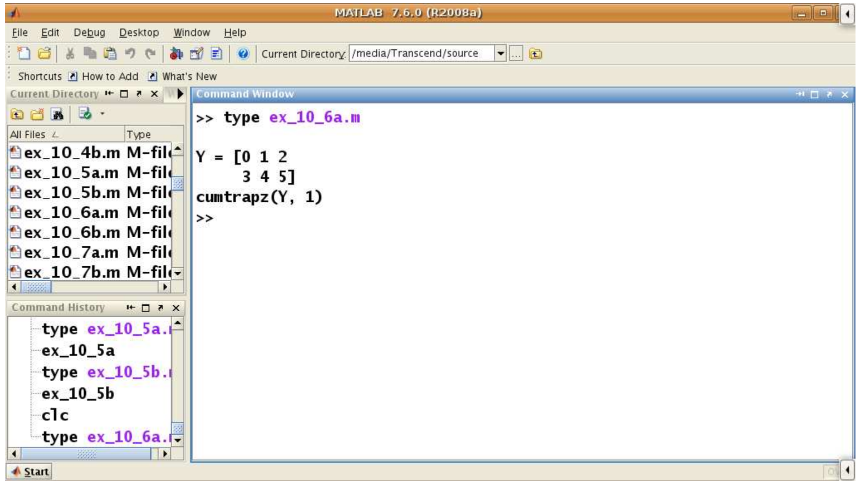
```
>> help cumtrapz
 CUMTRAPZ Cumulative trapezoidal numerical integration.
    Z = CUMTRAPZ(Y) computes an approximation of the cumulative
    integral of Y via the trapezoidal method (with unit spacing).  To
    compute the integral for spacing different from one, multiply Z by
    the spacing increment.

    For vectors, CUMTRAPZ(Y) is a vector containing the cumulative
    integral of Y. For matrices, CUMTRAPZ(Y) is a matrix the same size as
    X with the cumulative integral over each column. For N-D arrays,
    CUMTRAPZ(Y) works along the first non-singleton dimension.

    Z = CUMTRAPZ(X,Y) computes the cumulative integral of Y with respect
    to X using trapezoidal integration.  X and Y must be vectors of the
    same length, or X must be a column vector and Y an array whose first
    non-singleton dimension is length(X).  CUMTRAPZ operates across this
    dimension.
```

APPROXIMATION OF INTEGRALS

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

cumtrapz

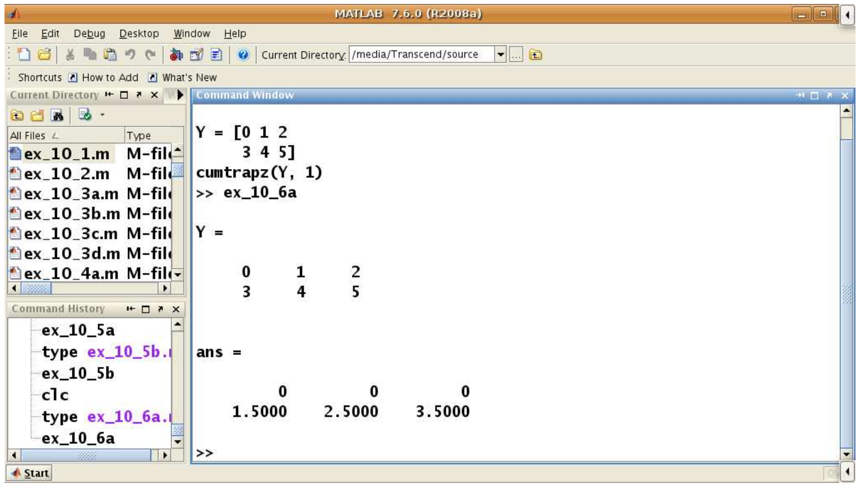


APPROXIMATION OF INTEGRALS

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

cumtrapz

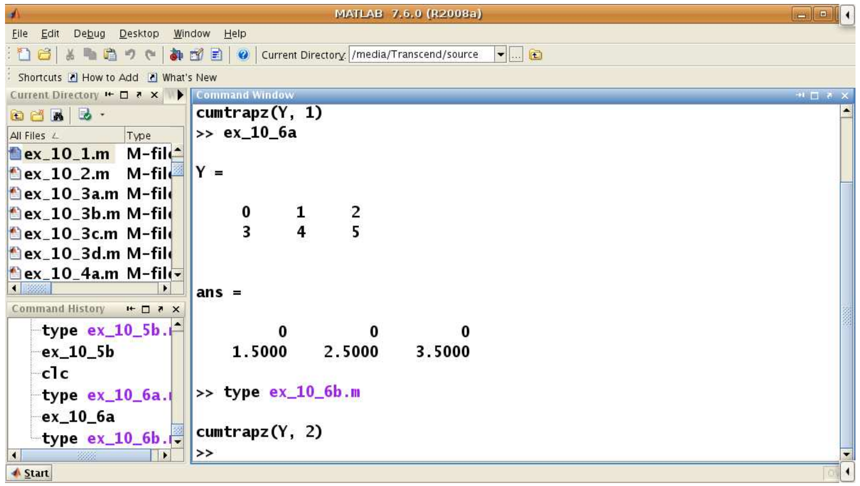


APPROXIMATION OF INTEGRALS

Introduction to Scientific and Engineering Computing, BIL108E

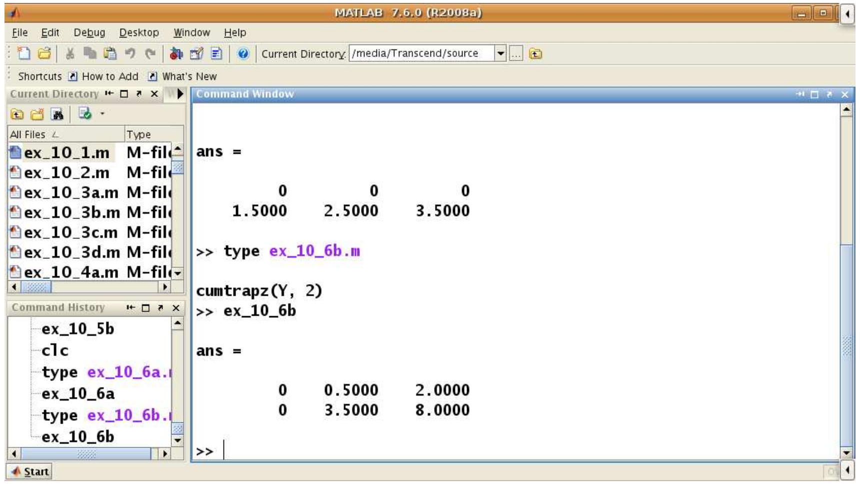
Karaman

cumtrapz



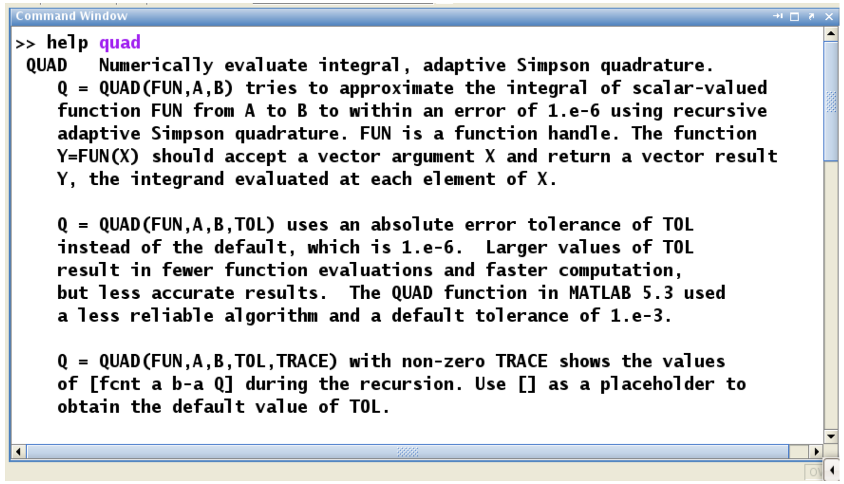
APPROXIMATION OF INTEGRALS

cumtrapz



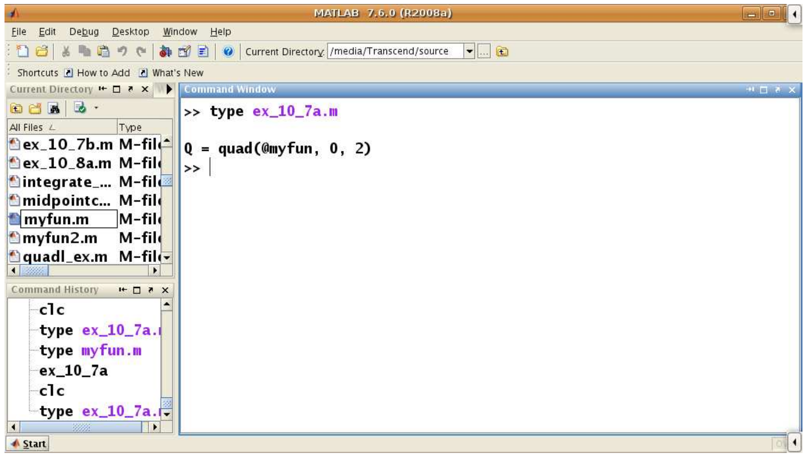
APPROXIMATION OF INTEGRALS

quad



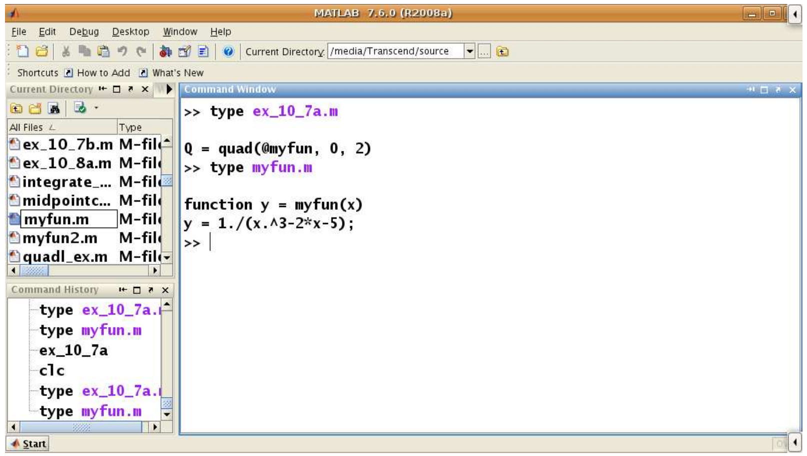
APPROXIMATION OF INTEGRALS

quad

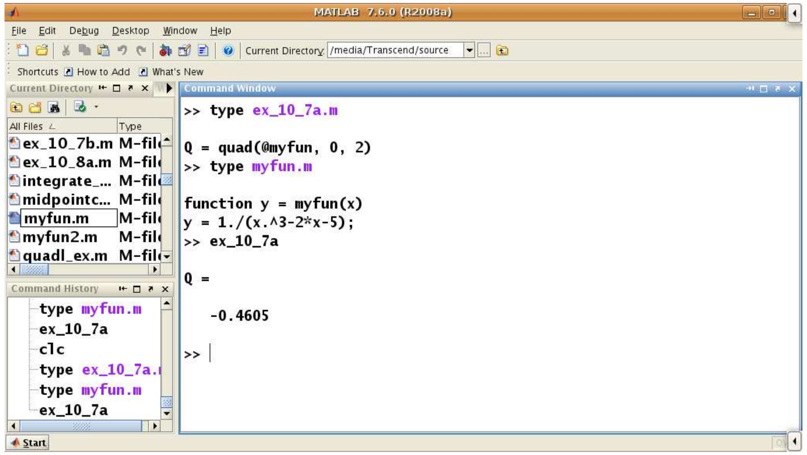


APPROXIMATION OF INTEGRALS

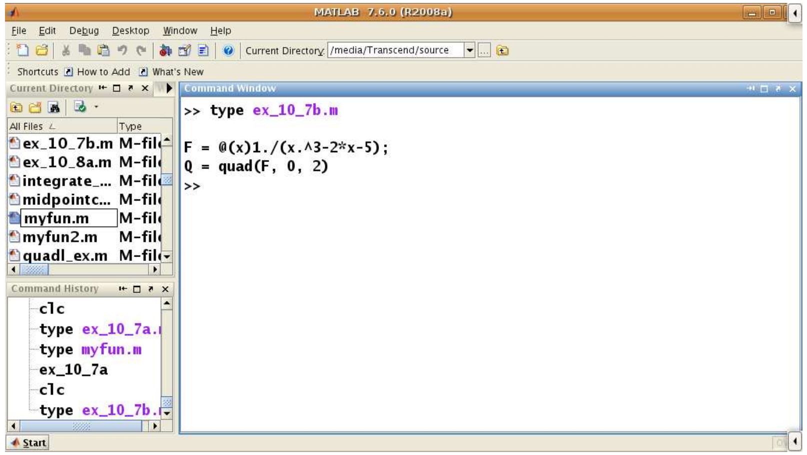
quad



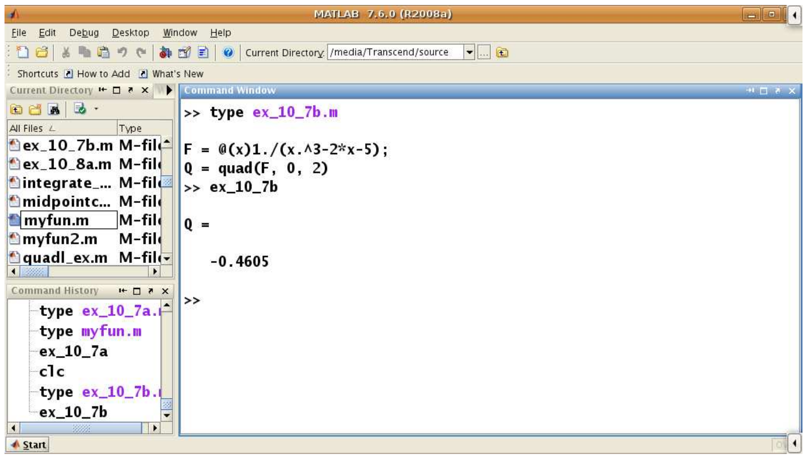
quad



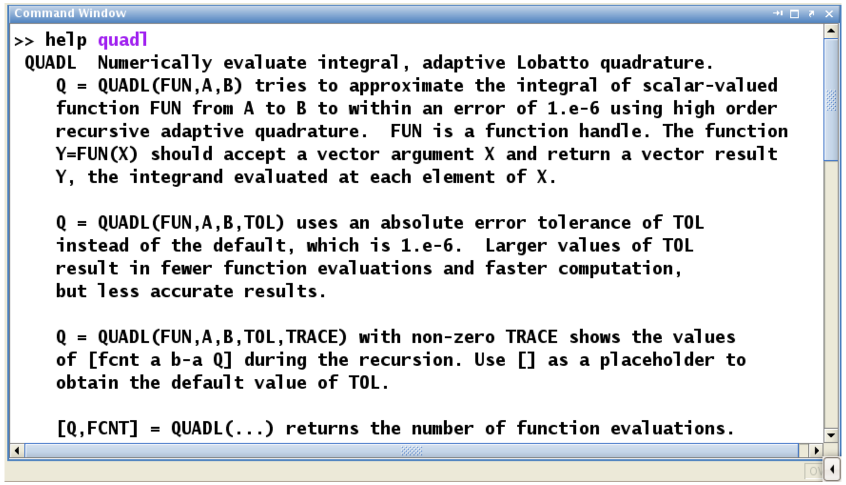
quad



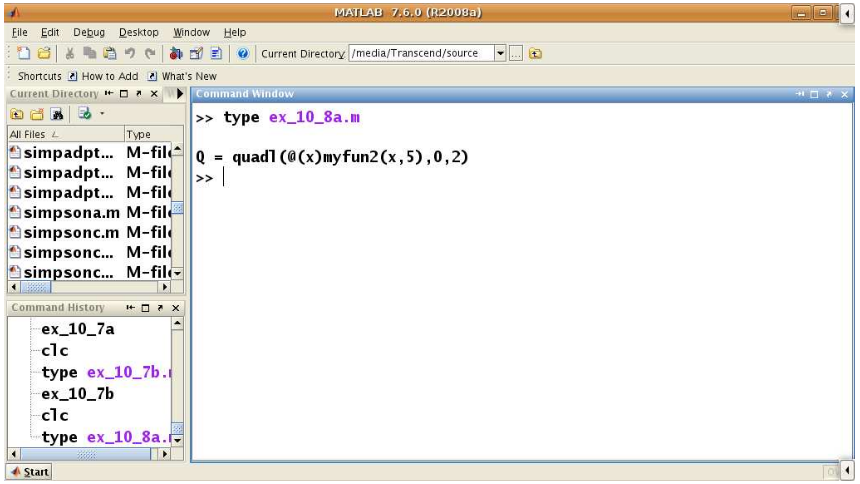
quad



quadl



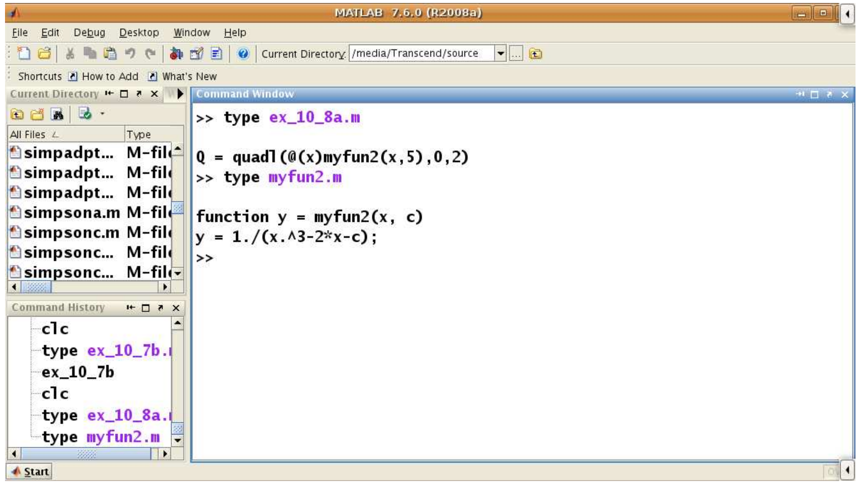
quadl



APPROXIMATION OF INTEGRALS

Introduction to Scientific and Engineering Computing, BIL108E
Karaman

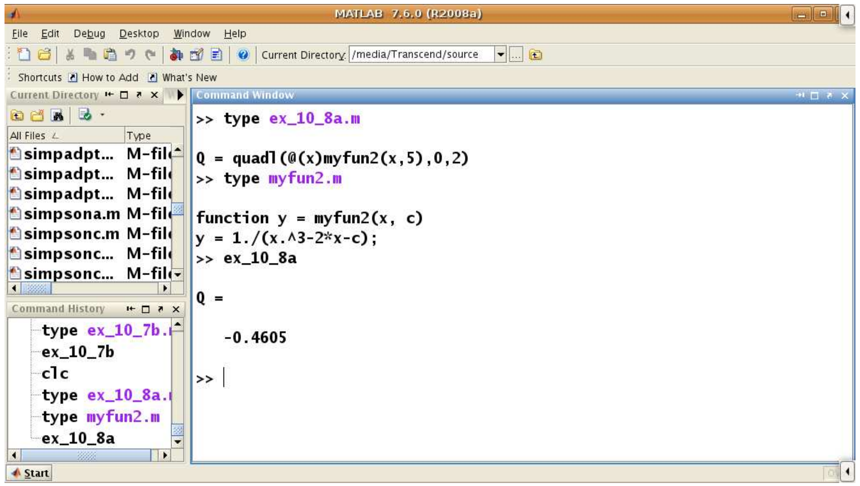
quadl



APPROXIMATION OF INTEGRALS

Introduction to Scientific and Engineering Computing, BIL108E
Karaman

quadl



APPROXIMATION OF INTEGRALS

Introduction to Scientific and Engineering Computing, BIL108E
Karaman

EXAMPLES:

Evaluate the following integral with different methods.

$$\int_0^{3\pi/2} cos(x)\, dx$$

Cosine is a built-in function in Matlab.

```
y=quad('cos',0,3*pi/2)
y=quadl('cos',0,3*pi/2)
```

EXAMPLES:

Evaluate the following integral with different methods.

$$\int_0^8 (x\, e^{-x^0.8} + 0.2)\, dx$$

```
quad('x.*exp(-x.^0.8)+0.2', 0,8)
quadl('x.*exp(-x.^0.8)+0.2', 0,8)
```


SOURCE:

```
function I=trapezoid(fun,a,b,npanel)
n=npanel+1; %total number of nodes
h=(b-a)/(n-1); %stepsize
x=a:h:b; %divide the interval
f=feval(fun,x); %evaluate the integral
I=h*(0.5*f(1)+sum(f(2:n-1))+0.5*f(n));
```


SOURCE:

```
function Imp=midpointc(a,b,M,f)
%MIDPOINTC Composite midpoint numerical integration.
%   IMP = MIDPOINTC(A,B,M,FUN) computes
%   an approximation of the integral
%   of the function FUN via the midpoint
%   method (with M equispaced intervals).
%   FUN accepts real scalar input x and
%   returns a real scalar
%   value. FUN can also be an inline object.
H=(b-a)/M;
x = linspace(a+H/2,b-H/2,M);
fmp=feval(f,x);
Imp=H*sum(fmp);
return
```


SOURCE:

```
function [Isic]=simpsonc(a,b,M,f,varargin)
%SIMPSONC Composite Simpson numerical integration.
%   ISIC = SIMPSONC(A,B,M,FUN) computes
%          an approximation of the integral
%          of the function FUN via the Simpson method
%          (with M equispaced intervals).
%          FUN accepts real scalar input
%          x and returns a real scalar
%          value. FUN can also be an inline object.
```

SOURCE cont'd.:

```
H=(b-a)/M;
x=linspace(a,b,M+1);
fpm=feval(f,x,varargin{:});
fpm(2:end-1) = 2*fpm(2:end-1);
Isic=H*sum(fpm)/6;
x=linspace(a+H/2,b-H/2,M);
fpm=feval(f,x,varargin{:});
Isic = Isic+2*H*sum(fpm)/3;
return
```


SOURCE:

```
function [JSf,nodes]=simpadpt(f,a,b,tol,hmin)
%SIMPADPT   Numerically evaluate integral,
%   adaptive Simpson quadrature.
%   JSF = SIMPADPT(FUN,A,B,TOL,HMIN)
%   tries to approximate the integral of function
%   FUN from A to B to within an error
%   of TOL using recursive
%   adaptive Simpson quadrature.
%   The inline function Y = FUN(V) should
%   accept a vector argument V and
%   return a vector result Y, the
%   integrand evaluated at each element of X.
%
%    [JSF,NODES] = SIMPADPT(...) returns the
%    distribution of nodes.
```


SOURCE cont'd.:

```
A=[a,b]; N=[]; S=[]; JSf = 0; ba = b - a; nodes=[];
while ~isempty(A),
  [deltaI,ISc]=caldeltai(A,f);
  if abs(deltaI) <= 15*tol*(A(2)-A(1))/ba;
     JSf = JSf + ISc;
     S = union(S,A);
     nodes = [nodes, A(1) (A(1)+A(2))*0.5 A(2)];
     S = [S(1), S(end)]; A = N; N = [];
```


SOURCE cont'd.:

```
  elseif A(2)-A(1) < hmin
     JSf=JSf+ISc;
     S = union(S,A);
     S = [S(1), S(end)]; A=N; N=[];
     warning('Too small step-length');
  else
     Am = (A(1)+A(2))*0.5;
     A = [A(1) Am];
     N = [Am, b];
  end
end
```

APPROXIMATION OF INTEGRALS

SOURCE cont'd.:

```
nodes=unique(nodes);
return

function [deltaI,ISc]=caldeltai(A,f)
L=A(2)-A(1);
t=[0; 0.25; 0.5; 0.5; 0.75; 1];
x=L*t+A(1);
L=L/6;
w=[1; 4; 1];
fx=feval(f,x);
IS=L*sum(fx([1 3 6]).*w);
ISc=0.5*L*sum(fx.*[w;w]);
deltaI=IS-ISc;
return
```

References

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

References for Week 10

1. Alfio Quarteroni, Fausto Saleri, Scientific Computing with Matlab and Octave, Springer, 2006.
2. Moler C, NumericalComputing with Matlab, Mathworks Inc., 2004 (http://www.mathworks.com/moler).
3. Thomas Huckle, Stefan Schneider, Numerische Methoden, Springer, 2006.