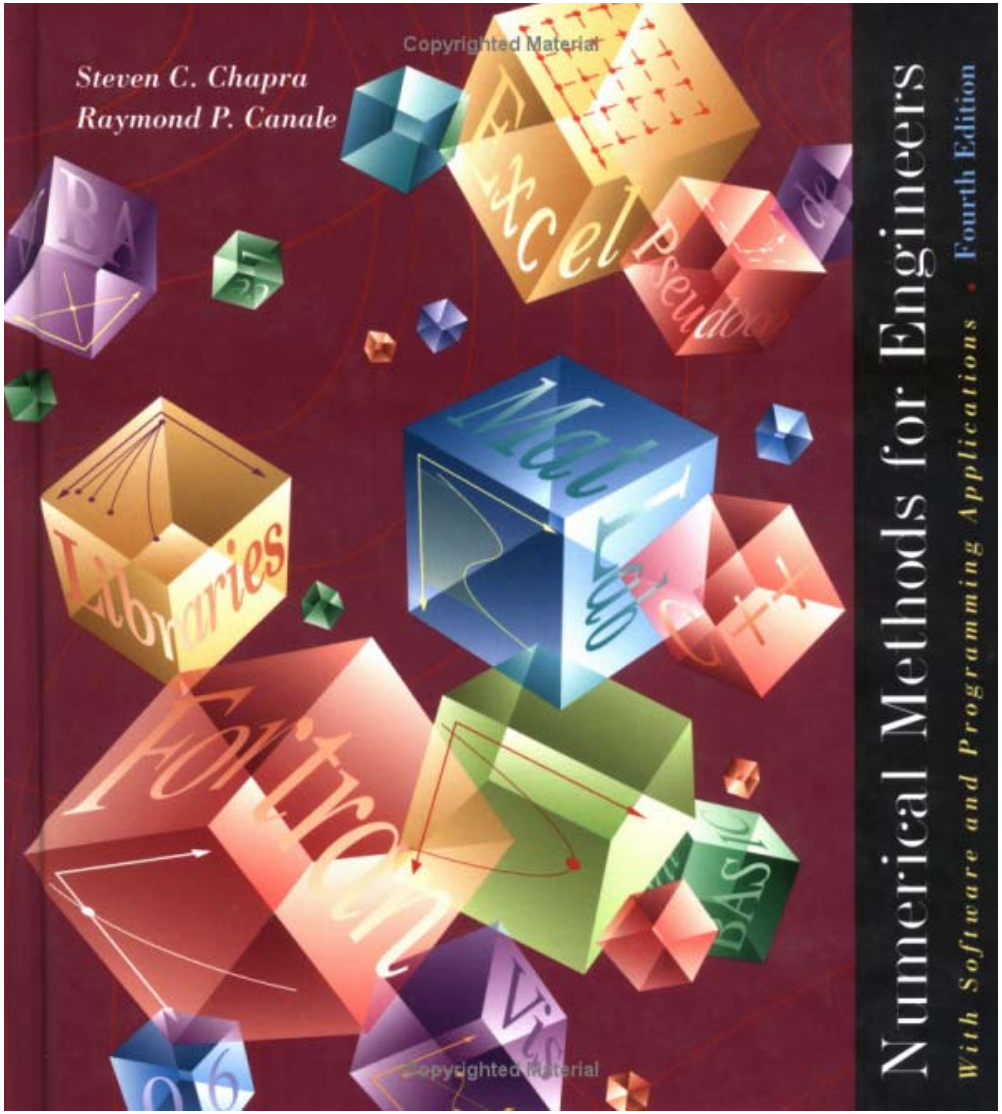

Steven C. Chapra
Raymond P. Canale
Copyrighted Material
Excel
Libraries
MATLAB
C++
FORTRAN
BASIC
Visual
Numerical Methods for Engineers
With Software and Programming Applications • Fourth Edition
Copyrighted Material

Parachutist Problem (1)

- Mathematical model

$$F = ma \rightarrow a = \frac{F}{m} \rightarrow \frac{dv}{dt} = \frac{mg - cv}{m} \rightarrow \frac{dv}{dt} = g - \frac{c}{m}v$$

$$\frac{dv}{dt} + \frac{c}{m}v = g : \text{differential equation}$$

$$v_h = c_1 e^{\lambda t} \rightarrow \lambda = -\frac{c}{m}$$

$$v = c_1 e^{\left(-\frac{c}{m}\right)t} + c_2 \rightarrow \frac{c}{m}c_2 = g \rightarrow c_2 = \frac{gm}{c}$$

$$v = c_1 e^{\left(-\frac{c}{m}\right)t} + \frac{gm}{c} \leftarrow \left(t = 0, v = 0\right) \Rightarrow c_1 = -\frac{gm}{c}$$

$$v = \frac{gm}{c}\left[1 - e^{\left(-\frac{c}{m}\right)t}\right] : \text{analytical (or exact) solution}$$

```
>>syms v g c m
>>dsolve('Dv=g-c/m*v','v(0)=0')
```

Parachutist Problem (2)

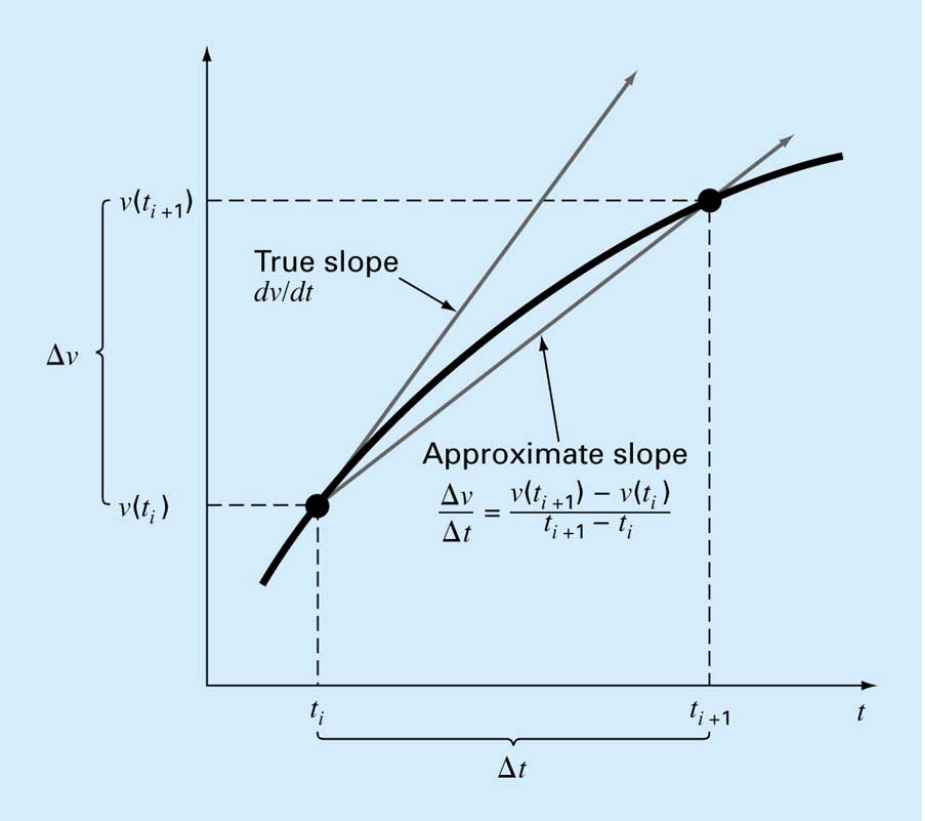
- Numerical methods

$$\frac{dv}{dt} = \lim_{\Delta t \to 0} \frac{\Delta v}{\Delta t} \cong \frac{\Delta v}{\Delta t} = \frac{v(t_{i+1}) - v(t_i)}{t_{i+1} - t_i}$$

$$\frac{dv}{dt} = g - \frac{c}{m} v \rightarrow \frac{v(t_{i+1}) - v(t_i)}{t_{i+1} - t_i} = g - \frac{c}{m} v(t_i)$$

$$\underbrace{v(t_{i+1})}_{new} = \underbrace{v(t_i)}_{old} + \underbrace{\left[g - \frac{c}{m} v(t_i) \right]}_{slope} \underbrace{(t_{i+1} - t_i)}_{step\ size}$$

: Euler's method



Packages and Programming

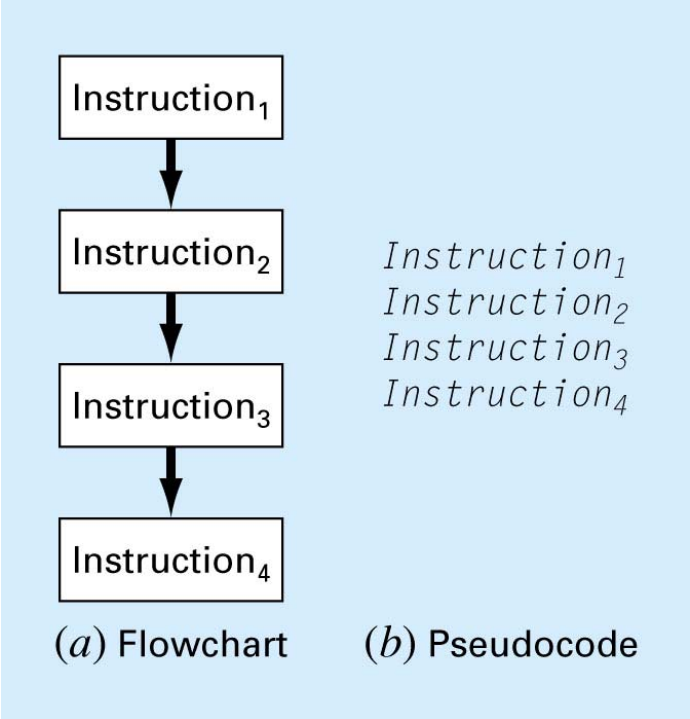
- Software users
  - Those who take what they are given
  - Power user: write Excel VBA, MATLAB M-files
- Computer programs
  - A set of instructions that direct the computer to perform a certain task
- Programming topics
  - Simple information representation: variables, type declaration
  - Advanced information representation: data structure, array
  - Mathematical formulas: assignment, intrinsic functions
  - Input/output
  - Logical representation: sequence, selection, repetition
  - Modular programming: functions, subroutines

Structured Programming

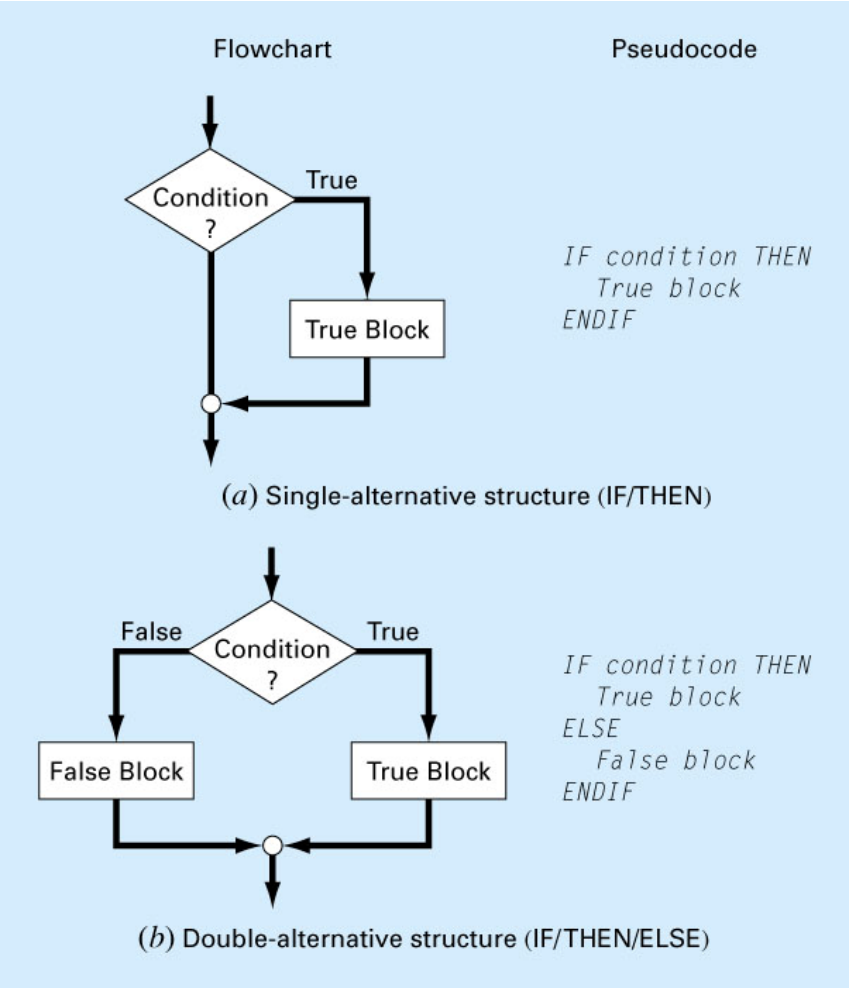
- Benefits to writing organized, well-structured code
  - Shorter time to develop, test, and update
- Three fundamental control structures
  - Sequence, selection and repetition
- Flowchart
  - Visual or graphical representation of an algorithm
- Pseudocode
  - Code-like statements in place of the graphical symbols of the flowchart
  - Bridges the gap between flowcharts and computer code

Logical Representation (1)

- Sequence

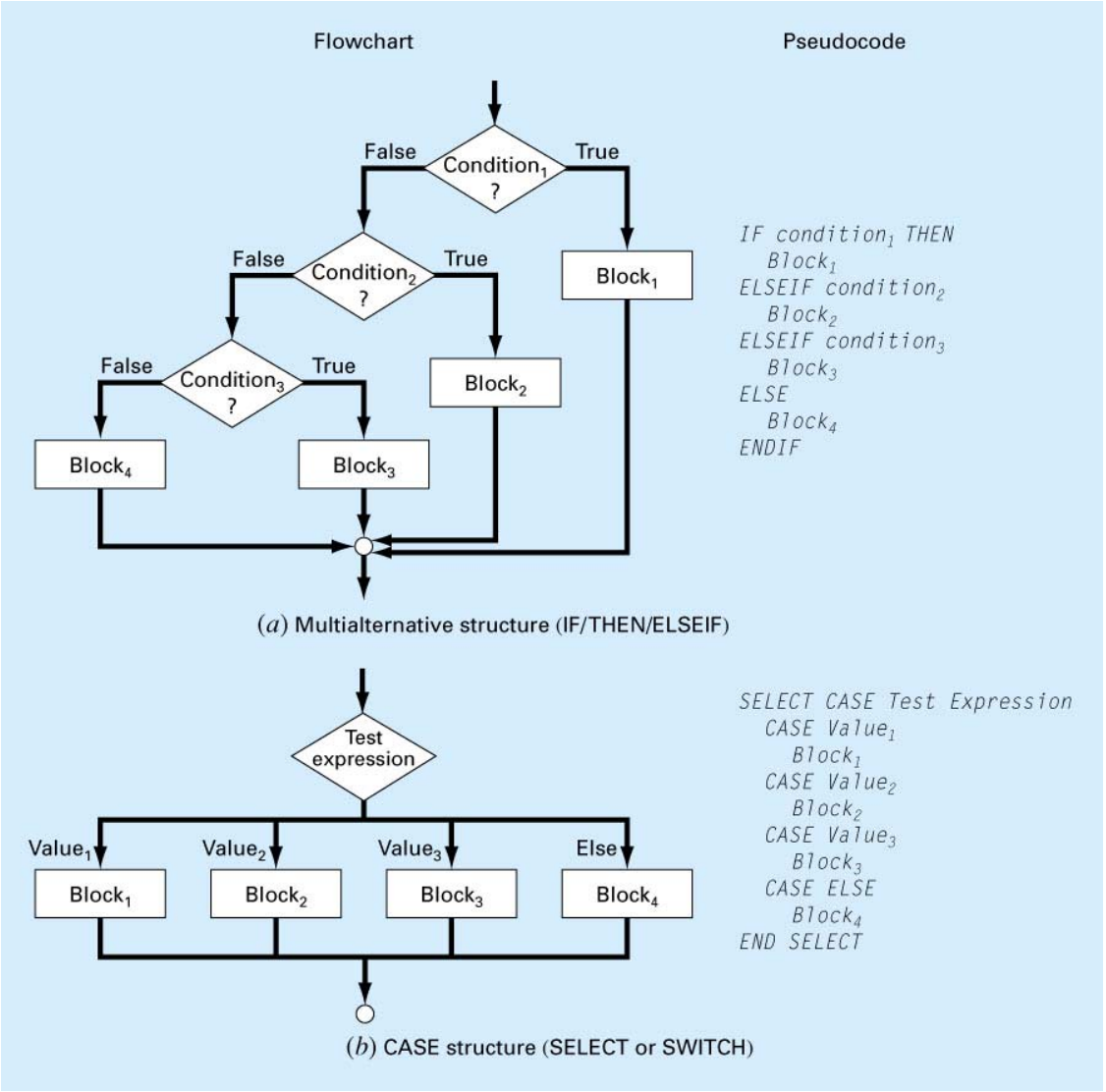


- Selection (1)



Logical Representation (2)

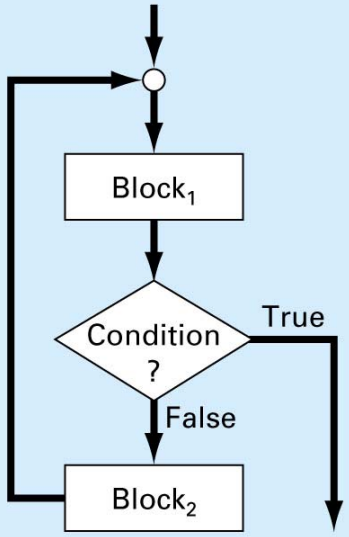
- Selection (2)



Logical Representation (3)

- Repetition

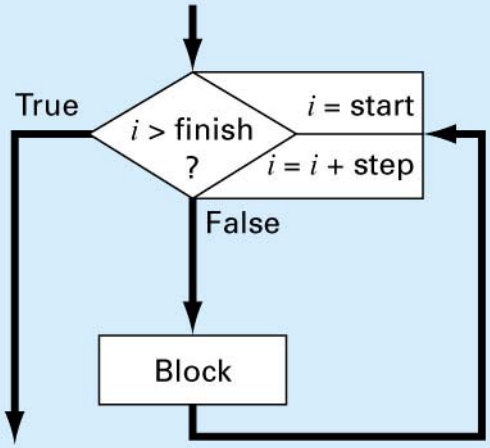
Flowchart



Pseudocode

```
DO
  Block1
  IF condition EXIT
  Block2
ENDDO
```

Flowchart



Pseudocode

```
DOFOR i = start, finish, step
  Block
ENDDO
```

Modular Programming

- Divide into small subprograms, or modules
  - Independent and self-contained as possible
  - Designed to perform a specific,well-defined function
  - Short and highly focused
- High-level languages such as Fortran 90 or C
  - Functions: single return
  - Subroutines: several returns
- Software packages like Excel and MATLAB
  - Subprograms

Control Structures (1)

Pseudocode	Excel VBA	MATLAB

IF/THEN:
IF *condition* THEN
  True block
ENDIF
	`If b<>0 then`
	`  r1 = -c / b`
	`End If`
		`if b ~= 0`
		`  r1 = -c / b;`
		`end`

IF/THEN/ELSE:
IF *condition* THEN
  True block
ELSE
  False block
ENDIF
	`If a < 0 Then`
	`  b = Sqr (Abs (a))`
	`Else`
	`  b = Sqr (a)`
	`End If`
		`if a < 0`
		`  b = sqrt (abs(a));`
		`else`
		`  b = sqrt (a);`
		`end`

IF/THEN/ELSEIF:
IF *condition1* THEN
  Block1
ELSEIF *condition2*
  Block2
ELSE
  Block3
ENDIF
	`If class = 1 Then`
	`  x = x + 8`
	`Else If class < 1 Then`
	`  x = x – 8`
	`Else`
	`  x = x – 64`
	`End If`
		`if case == 1`
		`  x = x + 8;`
		`elseif class < 1`
		`  x = x – 8;`
		`else`
		`  x = x – 64;`
		`end`

Control Structures (2)

Pseudocode	Excel VBA	MATLAB

CASE:
SELECT CASE *test expression*
  CASE *value1*
  block1
  CASE *value2*
  block2
  CASE ELSE
  block3
END SELECT
	`Select Case a + b`
	`  Case Is < -50`
	`  x = -5`
	`  Case Is < 0`
	`  x = -5 - (a + b) / 10`
	`  Case Else`
	`  x = 5`
	`End Select`
		`switch a + b`
		`  case 1`
		`  x = -5;`
		`  case 2`
		`  x= -5 – (a + b) / 10`
		`  otherwise`
		`  x = 5;`
		`end`

DOEXIT:
DO
  block1
  IF *condition* EXIT
  block2
ENDIF
	`Do`
	`  i = i + 1`
	`  IF i >= 10 Then Exit Do`
	`  j = i*x`
	`Loop`
		`while (1)`
		`  i = i + 1;`
		`  if i >= 10, break, end`
		`  j = i*x`
		`end`

COUNT-CONTROLLED LOOP:
DOFOR *i = start, finish, step*
  Block
ENDDO
	`For i = 1 To 10 Step 2`
	`  x = x + i`
	`Next i`
		`for i = 1:10:2`
		`  x = x + i`
		`end`

Parachutist Problem: Excel VBA

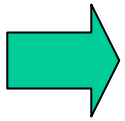
```
g = 9.8
INPUT cd, m
INPUT ti, vi, tf, dt
t = ti
v = vi
n = (tf - ti) / dt
DOFOR i = 1 TO n
  dvdt = g - (cd / m) * v
  v = v + dvdt * dt
  t = t + dt
ENDDO
DISPLAY v
```

```
h = dt
DO
  IF t + dt > tf THEN
    h = tf - t
  ENDIF
  dvdt = g - (cd / m) * v
  v = v + dvdt * h
  t = t + h
  IF t ≥ tf EXIT
ENDDO
DISPLAY v
```

```
FUNCTION Euler(dt, ti, tf, yi)
t = ti
y = yi
h = dt
DO
  IF t + dt > tf THEN
    h = tf - t
  ENDIF
  dydt = dy(t, y)
  y = y + dydt * h
  t = t + h
  IF t ≥ tf EXIT
ENDDO
Euler = y
END
```

Parachutist Problem: MATLAB

```
g=9.8;
m=input('mass (kg):');
cd=12.5;
ti=0;
tf=2;
vi=0;
dt=0.1;
t = ti;
v = vi;
h = dt;
while (1)
  if t + dt > tf
    h = tf - t;
  end
  dvdt = g - (cd / m) * v;
  v = v + dvdt * h;
  t = t + h;
  if t >= tf, break, end
end
disp('velocity (m/s):')
disp(v)
```



```
function euler = f(dt,ti,tf,yi,m,cd)
t = ti;
y = yi;
h = dt;
while (1)
  if t + dt > tf
    h = tf - t;
  end
  dydt = dy(t, y, m, cd);
  y = y + dydt * h;
  t = t + h;
  if t >= tf, break, end
end
euler = y;
```

```
function dy = f(t, v, m, cd)
g = 9.8;
dy = g - (cd / m) * v;
```

```
>> file_name
mass (kg): 100
velocity (m/s):
   17.4381
```

```
>> m=68.1;
>> cd=12.5;
>> ti=0;
>> tf=2.;
>> vi =0;
>> vt=0.1;
>> euler (dt,ti,tf,vi,m,cd)
```

MATLAB

- Data Analysis
  - 다양한 연산기능을 이용하여 각종 데이터 해석이 가능
- Visualization and Image Processing
  - 강력한 가시화 기능을 활용한 직감적인 해석결과 파악
- Programming & Algorithm Development
  - 구조화된 고급 언어로 알고리즘 개발을 지원
- Application Deployment
  - 작업 효율화를 지원하는 독립 실행 애플리케이션 배포
- Modeling & Simulation
  - 시뮬레이션 및 모델링을 위한 다양한 함수
- Test & Measurement
  - 계측기기로부터 데이터 수집 및 임의 신호 생성

Summary

Item	Excel	MATLAB
Root location	`Goal Seek` `Solve`	`fzero(f,x0,options)` `Roots(a)`
Linear Algebraic Equations	`minverse` `mmult`	
Curve Fitting	`Forecast, growth, intercept, linest, logest, slope, trend`	`Polyfit, interp1, interp2, spline, fft`

Integration and Differentiation		`quad`
		`trapz`
		`diff`

Ordinary Differential Equations	`macro`	`ode23, ode45`
		`ode15s, ode23s`
		`eig`

Partial Differential Equations	`cell ↔ rectangular grids`	

Root Location / Polynomial Manipulation

- Excel
  - Goal Seek, Solver
- MATLAB

Function	Description
fzero	Root of single function
roots	Find polynomial roots
poly	Construct polynomial with specified roots
polyval	Evaluate polynomial
polyvalm	Evaluate polynomial with matrix argument
residue	Partial-fraction expansion
polyder	Differentiate polynomial
conv	Multiply polynomials
deconv	Divide polynomials

Linear Algebra Equations (1)

- Excel
  - Functions: minverse, mmult, mdeterm, transpose
- MATLAB
  - Matrix analysis

Function	Description
cond	Matrix condition number
norm	Matrix or vector norm
rcond	LINPACK reciprocal condition number
rank	Number of linearly independent rows or columns
det	Determinant
trace	Sum of diagonal elements
null	Null space
orth	Orthogonalization
rref	Reduced row echelon form

Linear Algebra Equations (2)

– Linear Equations

Function	Description
\ and /	Linea equation solution; >>help slash
chol	Cholesky factorization
lu	Factors from Gauss elimination
inv	Matrix inverse
qr	Orthogonal-triangular decomposition
qrdelete	Delete a column from the QR factorization
qrinsert	Insert a column from the QR factorization
nnls	Nonnegative least squares
pinv	Pseudoinverse
lscov	Least squares in the presence of known covariance

Curve Fitting

- Excel

Function	Description
FORCAST	Returns a value along a linear trend
GROWTH	Returns values along an exponential trend
INTERCEPT	Returns the intercept of the linear regression line
LINEST	Returns the parameters of a linear trend
LOGEST	Returns the parameters of an exponential trend
SLOPE	Returns the slope of the linear regression line
TREND	Returns values along a linear trend

- MATLAB

Function	Description
polyfit	Fir polynomial to data
interp1	1-D interpolation (1-D table lookup)
interp2	2-D interpolation (2-D table lookup)
spline	Cubic spine data interpolation
fft	Discrete Fourier transform