# Tutorial: Using MATLAB© for Mathematical Programming

## APS502 - Financial Engineering I

Dexter Wu

Mechanical and Industrial Engineering
University of Toronto

March 24, 2016

# Outline

- Matrix Construction.
- Optimization toolbox in MATLAB©.
    - LP and QP solver.
    - Numerical examples in MATLAB©.
- Q&A and Exercises

# Basic operators

## Definition

A **Matrix** is a table of numbers, similar to a spreadsheet.

$$A = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}, B = [2, 1], C = \begin{bmatrix} 0 \\ 3 \end{bmatrix},$$

$$D = \begin{bmatrix} 3 & 4 \\ 5 & 3 \\ 7 & 2 \end{bmatrix}, D^T = \begin{bmatrix} 3 & 5 & 7 \\ 4 & 3 & 2 \end{bmatrix}, E = \begin{bmatrix} 4 & 3 \\ 1 & 1 \end{bmatrix}, E^{-1} = \begin{bmatrix} 1 & -3 \\ -1 & 4 \end{bmatrix}$$

- Matlab code:
  1. >> A = [1 4 7; 2 5 8; 3 6 9;]
  2. >> B = [2 1]; C = [0; 3;];
  3. >> D = [3 4; 5 3; 7 2;]; D_trans = D';
  4. >> E = [4 3; 1 1;]; E_inv = inv(E);

# Basic Operators

| Operators | Examples |
|---|---|
| +, - | $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} + \begin{bmatrix} 5 & 7 \\ 6 & 8 \end{bmatrix} = \begin{bmatrix} 1+5 & 2+7 \\ 3+6 & 4+8 \end{bmatrix}$ |
| *, ^ | $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} * \begin{bmatrix} 5 & 7 \\ 6 & 8 \end{bmatrix} = \begin{bmatrix} 1*5+2*6 & 1*7+2*8 \\ 3*5+4*6 & 3*7+4*8 \end{bmatrix}$ (matrix multiplication) |
| .*, .^ | $\begin{bmatrix} 1 & 2 & 0 \\ 3 & 4 & -1 \end{bmatrix} .* \begin{bmatrix} 5 & 7 & 2 \\ 6 & 8 & 5 \end{bmatrix} = \begin{bmatrix} 1*5 & 2*7 & 0*2 \\ 3*6 & 4*8 & -1*5 \end{bmatrix}$ (array multiplication) |
| \, / | A/B = A*inv(B)<br>A\B = inv(A)*B ($B = I$) |
| :, ' | A(1:3, 4:6) sub-matrix of A that row from 1 to 3 and column from 4 to 6 |

# Building Matrix

- Manual Entry
- By some standard functions
  1. zeros(m,n) creates an (m,n) matrix of zeros;
  2. ones(m,n) creates an (m,n) matrix of ones;
  3. eye(n) creates the (n,n) identity matrix;
- By loops
  1. cat(dim, A, B)
  2. blkdiag() or diag(v)
  3. repmat(), reshape() can simplify the loop

# Sparse Matrix

- A sparse matrix is a matrix mostly populated by zeros
- There are many engineering and optimization problems that can be stated using sparse coefficient matrices (ex: graph/network optimization)
- MATLAB has the ability to create, store and manipulate sparse matrices efficiently

# Create sparse matrix

- A = sparse(m, n) creates an (m,n) sparse matrix of zeros
- B = sparse(A) converts matrix A to a sparse matrix B

```
>> A=zeros(3)

A =

     0     0     0
     0     0     0
     0     0     0
```

```
>> B=sparse(A)

B =

   All zero sparse: 3-by-3
```

```
>> whos
  Name      Size            Bytes  Class     Attributes

  A         3x3                72  double
  B         3x3                28  double    sparse
```

# Optimization toolbox

- Optimization Toolbox provides widely used algorithms for standard and large-scale optimization.
- The toolbox includes solvers for:
    1. Linear Programming (LP)
    2. Quadratic Programming (QP)
    3. Binary Integer Programming (BIP)
    4. Non-Linear Programming (NLP)

# Solve LPs using linprog (1)

Linprog solves a problem of this form:

$$\begin{aligned} \min_x \quad & f^T x \\ s.t.\ & A * x \leq b \\ & Aeq * x = beq \\ & lb \leq x \leq ub \end{aligned}$$

where $f$, $x$, $lb$, $ub = n \times 1$ vector;

$A = m_1 \times n$ matrix, $Aeq = m_2 \times n$ matrix;

$b = m_1 \times 1$ vector, $beq = m_2 \times 1$ vector.

# Solve LPs using linprog (2)

- Syntax:
  - [x, fval, exitflag, output, lambda] = linprog(f, A, b, Aeq, beq, lb, ub, x0, options)
- Linprog takes 6 input variables:
  - f: Objective coefficient
  - A, b: Inequality constraints ($\leq$)
  - Aeq, beq: Equality constraints
  - lb, ub: Variable bounds
  - x0: Initial guess
  - options: set parameters (e.g. Algorithm) for the solver
- Linprog returns 5 output variables:
  - x: Optimal Solution
  - fval: Objective Function Value
  - exitflag: optimal, infeasible, unbounded, etc.
  - output: information about the algorithm
  - lambda: dual variables

# LP example (1)

- Solve below LP:

$$\begin{aligned} \min_{x} \quad & 2x_1 - x_2 + x_3 \\ s.t. \; & 2x_1 - 3x_2 - x_3 \leq 9 \\ & 2x_1 - x_2 \leq -4 \\ & x_1 + 3x_3 = 6 \\ & x_1, x_3 \geq 0, x_2 \leq 0 \end{aligned}$$

- Build the input arguments for linprog:
  - c = [2, -1, 1]'
  - A = [2, -3, -1; 2, -1, 0]  b = [9; -4]
  - Aeq = [1, 0, 3]  beq = [6]
  - ub = [inf; 0; inf;]
  - lb = [0; -inf; 0;]
  - Ignore for now: x0, options
- Call linprog from Matlab
  - [x, fval] = linprog(c, A, b, Aeq, beq, lb, ub)

# LP example (2)

- Solve below LP:

$$
\begin{aligned}
\max_{x}\ & 2x_1 + x_2 + 3x_3 + x_4 \\
s.t.\ & x_1 + x_2 + x_3 + x_4 \le 5 \\
& 2x_1 - x_2 + 3x_3 \qquad = -4 \\
& x_1 \qquad - x_3 + x_4 \ge 1 \\
& x_1, x_3 \ge 0, x_2, x_4 \text{ unrestricted}
\end{aligned}
$$

- Build the input arguments for linprog:
  - c = -[2, 1, 3, 1]'
  - A = [1 1 1 1; -1 0 1 -1;]  b = [5; -1]
  - Aeq = [2, -1, 3, 0]  beq = [-4]
  - lb = [0; -inf; 0; -inf;]  ub = [inf; inf; inf; inf;]
  - Ignore for now: x0, options
- Call linprog from Matlab
  - [x, fval] = linprog(c, A, b, Aeq, beq, lb, ub)

# LP example (3) - Bond Portfolio Optimization

- Suppose that a bank has the following liability schedule:

| Year 1 | Year 2 | Year 3 |
|---|---|---|
| \$12,000 | \$18,000 | \$20,000 |

- The bank wishes to use three bonds to form a portfolio (a collection of bonds) today to hold until all bonds have matured and that will generate the required cash to meet the liabilities.

| Bond | 1 | 2 | 3 |
|---|---|---|---|
| Price | 102 | 99 | 98 |
| Coupon | 5 | 3.5 | 3.5 |
| Maturity year | 1 | 2 | 3 |

# LP example (3) - Bond Portfolio Optimization

- Let $x_i$ = amount of bond $i$ purchased, the problem can be formulated as below LP:

$$\begin{aligned} \min_x & 102x_1 + 99x_2 + 98x_3 \\ s.t. & 105x_1 + 3.5x_2 + 3.5x_3 \geq 12000 \\ & 103.5x_2 + 3.5x_3 \geq 18000 \\ & 103.5x_3 \geq 20000 \\ & x_1, x_2, x_3 \geq 0 \end{aligned}$$

- Build the input arguments for linprog:
  - c = [102 99 98]'
  - A = -[105 3.5 3.5; 0 103.5 3.5; 0 0 103.5] b = -[12000; 18000; 20000]
  - Aeq = [] beq = []
  - lb = [0; 0; 0;] ub = [inf; inf; inf;]
  - Ignore for now: x0, options
- Call linprog from Matlab
  - [x, fval] = linprog(c, A, b, Aeq, beq, lb, ub)

# Solve QPs using quadprog (1)

Quadprog solves a problem of this form:

$$\begin{aligned} \min_x \quad & \tfrac{1}{2}x^T Hx + f^T x \\ s.t. \quad & A * x \leq b \\ & Aeq * x = beq \\ & lb \leq x \leq ub \end{aligned}$$

where $f$, $x$, $lb$, $ub = n \times 1$ vector;

$A = m_1 \times n$ matrix, $Aeq = m_2 \times n$ matrix;

$b = m_1 \times 1$ vector, $beq = m_2 \times 1$ vector;

$H = n \times n$ symmetric matrix, usually require PSD.

# Solve QPs using quadprog (2)

- Syntax:
  - [x, fval, exitflag, output, lambda] = quadprog(H, f, A, b, Aeq, beq, lb, ub, x0, options)
- Linprog takes 7 input variables:
  - H: Symmetric matrix represents the quadratic term in objective
  - f: Coefficient vector for the linear term in objective
  - A, b: Inequality constraints ($\leq$)
  - Aeq, beq: Equality constraints
  - lb, ub: Variable bounds
  - x0: Initial guess
  - options: set parameters (e.g. Algorithm) for the solver
- Linprog returns 5 output variables:
  - x: Optimal Solution
  - fval: Objective Function Value
  - exitflag: optimal, infeasible, unbounded, etc.
  - output: information about the algorithm
  - lambda: dual variables

# QP example (1)

- Solve below QP:

$$\begin{aligned} \min_{x} \quad & .5x_1^2 - x_1x_2 + x_2^2 - 2x_1 - 6x_2 \\ s.t. \quad & x_1 + x_2 \leq 2 \\ & -x_1 + 2x_2 \leq 2 \\ & 2x_1 + x_2 \leq 3 \\ & x_1, x_2 \geq 0 \end{aligned}$$

- Build the input arguments for quadprog:
  - H = [1 -1; -1 2]
  - c = [-2 -6]'
  - A = [1 1; -1 2; 2 1;]  b = [2; 2; 3;]
  - Aeq = []  beq = []
  - lb = [0; 0;]  ub = [inf; inf;]
  - Ignore for now: x0, options
- Call quadprog from Matlab
  - [x, fval] = quadprog(H, c, A, b, Aeq, beq, lb, ub)

# QP example (2) - MVO

- The Mean-Variance Optimization (MVO) model is given as follows:

$$\begin{aligned} \min_x \textstyle\sum_{i=1}^n \sum_{j=1}^n \sigma_{ij} x_i x_j \\ s.t. \textstyle\sum_{i=1}^n \mu_i x_i \geq R \\ \textstyle\sum_{i=1}^n x_i = 1 \\ (x \geq 0) \end{aligned}$$

- In matrix form the MVO model is

$$\begin{aligned} \min_x \tfrac{1}{2} x^T Q x \\ s.t. \mu^T x \geq R \\ e^T x = 1 \\ (x \geq 0) \end{aligned}$$

- MVO can be solved by quadprog.

# QP example (2) - MVO

- Consider 3 securities with expected return given in Table 1 and with covariance given in Table 2:

| expected return | security 1 ($i = 1$) | security 2 ($i = 2$) | security 3 ($i = 3$) |
|---|---|---|---|
| $\mu_i$ | 9.73% | 6.57% | 5.37% |

| covariance $\sigma_{ij}$ | $i = 1$ | $i = 2$ | $i = 3$ |
|---|---|---|---|
| $i = 1$ | 0.02553 | -0.00327 | 0.00019 |
| $i = 2$ | -0.00327 | 0.013400 | -0.00027 |
| $i = 3$ | 0.00019 | -0.00027 | 0.00125 |

- If the goal return of portfolio is 5.5%, solve MVO with and without short selling.

# QP example (2) - MVO

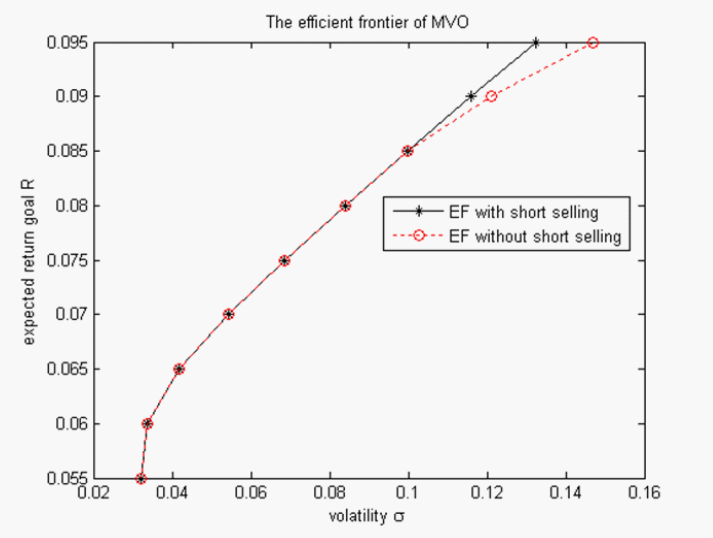
- MVO can be formulated as follows:

$$\min_x (0.02553)x_1^2 + (0.01340)x_2^2 + (0.00125)x_3^2$$
$$+2(-0.00327)x_1x_2 + 2(0.00019)x_1x_3 + 2(-0.00027)x_2x_3$$
$$s.t. (0.0972)x_1 + (0.0657)x_2 + (0.0537)x_3 \geq 0.055$$
$$x_1 + x_2 + x_3 = 1$$
$$(x_1, x_2, x_3 \geq 0) \text{ short selling is prohibited}$$

- Build the input arguments for quadprog:
  - Q = [0.02553, -0.00327, 0.00019; -0.00327, 0.013400, -0.00027; 0.00019, -0.00027, 0.00125;]
  - c = [0 0 0]'
  - A = -[0.0972, 0.0657, 0.0537]  b = -[0.055]
  - Aeq = [1 1 1]  beq = [1]
  - ub = [inf; inf; inf;]
  - lb = [-inf; -inf; -inf;] % with short selling
  - lb_without = [0; 0; 0;] % without short selling

# QP example (2) - MVO

- Vary R and we can get efficient frontiers of MVO:



# Numerical examples

- Numerical examples in MATLAB