COMPUTATION 2– Handout 2 **AUTUMN 2008**

LECTURE 2. MATRIX OPERATIONS
7 October 2008, Sackville/C53

Teaching materials are at
http://personalpages.manchester.ac.uk/staff/henry.tan/teaching/computation2/computation2.htm

I. Matrix manipulations: matrices and arrays

1. Matrix addition, subtraction and scalar multiplication

- Cutting and pasting
- Ranges + ***CSE***
- Named ranges + ***CSE***

2. Array functions generally returns more than one value. Complete the formula with *control-shift enter (CSE)*, not with just *enter*.

- Multiplying two matrices: MMULT
- Inverting a matrix: MINVERSE
- Determinant of a matrix: MDETERM

II. Solving simultaneous linear equations

Solve the simultaneous linear equations

$$\begin{cases} 2x+3y=7 \\ x-3y=-1 \end{cases}. \tag{1}$$

1. Using Excel Solver

Define

$$\Phi(x,y)=f_1^2+f_2^2,$$

where

$$\begin{aligned} f_1(x,y)&=2x+3y-7 \\ f_2(x,y)&=x-3y+1 \end{aligned}.$$

The solution of equations (1) can be achieved through minimizing the function $\Phi(x,y)=f_1^2+f_2^2$.

```
Function f1(x As Double, y As Double)
    f1 = 2 * x + 3 * y - 7
End Function

Function f2(x As Double, y As Double)
    f2 = x - 3 * y + 1
End Function

Function Phi(x As Double, y As Double)
    Phi = f1(x, y) ^ 2 + f2(x, y) ^ 2
End Function
```

2. Using matrix algebra

Equations (1) can be rewritten as

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$$

where

$$\mathbf{A} = \begin{bmatrix} 2 & 3 \\ 1 & -3 \end{bmatrix}, \ \mathbf{b} = \begin{bmatrix} 7 \\ -1 \end{bmatrix}.$$

From

$$\mathbf{A}^{-1} \cdot \mathbf{A} \cdot \mathbf{x} = \mathbf{A}^{-1} \cdot \mathbf{b}$$

The equations can solved through

$$\rightarrow \qquad \mathbf{x} = \mathbf{A}^{-1} \cdot \mathbf{b}$$

III. Writing VBA matrix functions

Transition between Excel range and VBA array

An Excel range is a rectangular group of one or more cells. To reference a range we use the top left cell separated from the bottom right reference by a colon. A VBA array refers to data organized in row and column fashion.

Sample 1: dot product of two vectors

```
Function dot_product(v1 As Range, v2 As Range) As Double
    dot_product = v1(1) * v2(1) + v1(2) * v2(2) + v1(3) * v2(3)
End Function
```

1st	cell of Excel range v1	→	VBA array v1(1)
2nd	cell of Excel range v1	→	VBA array v1(2)
3rd	cell of Excel range v1	→	VBA array v1(3)
1st	cell of Excel range v2	→	VBA array v2(1)
2nd	cell of Excel range v2	→	VBA array v2(2)
3rd	cell of Excel range v2	→	VBA array v2(3)

Sample 2: creating an identity matrix

```
Function identity_matrix(n As Integer)
    Dim a() As Double
    Dim i As Integer
    ReDim a(1 To n, 1 To n)
    For i = 1 To n
        a(i, i) = 1
    Next i
    identity_matrix = a
End Function
```

When declare a variable, it is a good habit to give it a specific data type, such as Range, Integer or Double.

```
Dim i As Integer
```

An array `a` of double-precision data is declared through

```
Dim a() As Double
```

It is sized with the ReDim statement

```
ReDim a(1 To n, 1 To n)
```

which defined array `a` as an $n \times n$ square matrix indexed from 1 to n.

Counter variable `i` is used in the For ... Next loop.