Computer Science
FYS

MATLAB

Essential MATLAB for Scientists

Chap 3: Fundamentals (continued)

Dr. Delphine Syvilay
Dr. Madhurima Panja

5. Expressions and statements

- Statements

MATLAB statements are frequently of the form

- variable = expression e.g.
- s = u * t – g / 2 * t .^ 2;

The variable is always on the left side of the equal sign and the expression on its right side.

Any line that you enter in the Command Window or in a program, which MATLAB accepts, is a statement. A statement could be an assignment, a command or an expression:

- x = 29; % assignment
- clear % command
- pi/2 % expression

6. Output

- There are two straightforward ways of getting output from MATLAB:

  -by entering a variable name, assignment or expression on the command line, without a semicolon.

  -with the disp statement, e.g. disp(x).

6. Output

- disp()

  The general form of disp for a numeric variable is disp(variable)

  The variable name is not displayed. The content of the variable is displayed.

  You can display a message enclosed in a single quotes (called a string). Single quotes which are part of the message must be repeated, e.g.

- disp (‘ Pilate said, ‘’What is truth?” ’);

  To display a message and a numeric value on the same line, use the following:

- x = 2;
- disp([‘ The answer is ’ , num2str(x)]);

6. Output

- Format

  The word format refers to how something is laid out, which in MATLAB would be the output. MATLAB has the two basic output rules:

-It always attempts to display integers (whole numbers) exactly. Now, if the integer is too large, then it is displayed in scientific notation with five significant digits, e.g. 123456789 is displayed as is and 1234567890 is displayed 1.2346e9.

$$x$$

6. Output

- Format

  The word format refers to how something is laid out, which in MATLAB would be the output. MATLAB has the two basic output rules:

-It always attempts to display integers (whole numbers) exactly. Now, if the integer is too large, then it is displayed in scientific notation with five significant digits, e.g. 123456789 is displayed as is and 1234567890 is displayed 1.2346e9.

-Numbers with decimal parts are displayed with four significant digits. If the value x is in the range 0.001 < x < 1000, then it is displayed in fixed point form, otherwise scientific (floating point) notation is used. In the latter case, the mantissa is between 1 and 9,9999. For example, 1000.1 is displayed as $1.0001e^{+03}$

6. Output

- Format

-Try the following in the command window:

- 0.0011
- 0.0009
- 1/3
- 5/3
- 2999/3
- 2001/3

-That is what we call the default format.

6. Output

- Format

-Type the command:

- format short e
- 0.0123456

-Make the format into scientific notation with 15 significant digits

- format long e

-Now try 1/7

6. Output

- format

Make the format into fixed point notation with 4 significant figures

- format short
- 100/7

Make the format into fixed point notation with 15 significant figures

- format long
- 100/7
- pi

6. Output

- Format

-If you are note sure of the order of magnitude of your output, you can try

- format short g
- format long g

g stands for general. MATLAB decides in each case whether to use fixed or floating point.

-For financial calculations, use

- format bank
- 10000/7

6. Output

- Format

-Suppress somehow irritating line-feeds

- format compact

-Reverse the compact format

- format loose

-Get hexadecimal display

- format hex

-Display a number as a rational approximation

- format rat

-Display the default format

- Format

-To get help on format

- help format

6. Output

- Scale factors

Enter the following commands

- >> format compact
- >> x = [1e3 1 1e-4]

With format short and format long, a common scale factor is applied to the whole vector if its elements are very large, very small, of differ greatly in magnitude.

6. Output

- Scale factors

Now type the following

- >> format bank
- >> x

You should read

- x =

1000.00 1.00 0.00

6. Output

- Scale factors

Now type the following

- >> format short e
- >> x

You should read

- x =

1.0000e+003 1.0000e+000 1.0000e-004

7. Repeating with for

Repetition is implemented by the extremely powerful for construct. Let us first look at some examples of its use, followed by explanations.

Enter for following group of statements on the command line.

- format compact
- for i = 1:5, disp(i), end

Now, change it slightly to

- for i = 1:3, disp(i), end

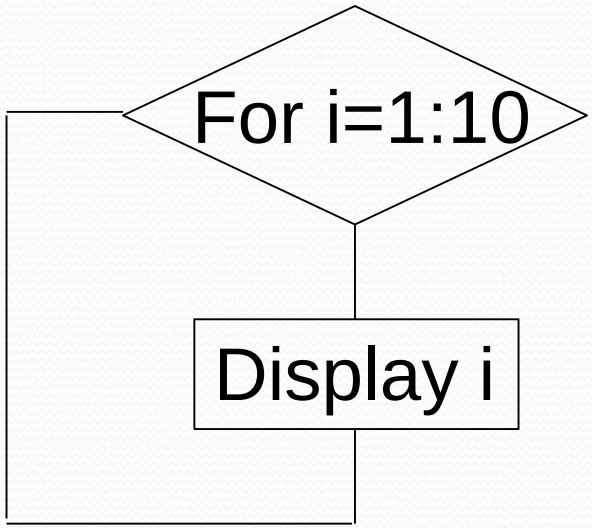
And now

- for i = 1:0, disp(i), end

The disp statement is repeated five times, then three times, and finally not at all.

7. Repeating with for

- Writing an algorithm



Matlab code

```
for i=1:10
        disp(i)
end
```

7. Repeating with for

- The basic for construct

In general, the most common form of the for loop is:

- for index = j:k
- statement 1
- statement 2
- ….
- statement p
- end

or

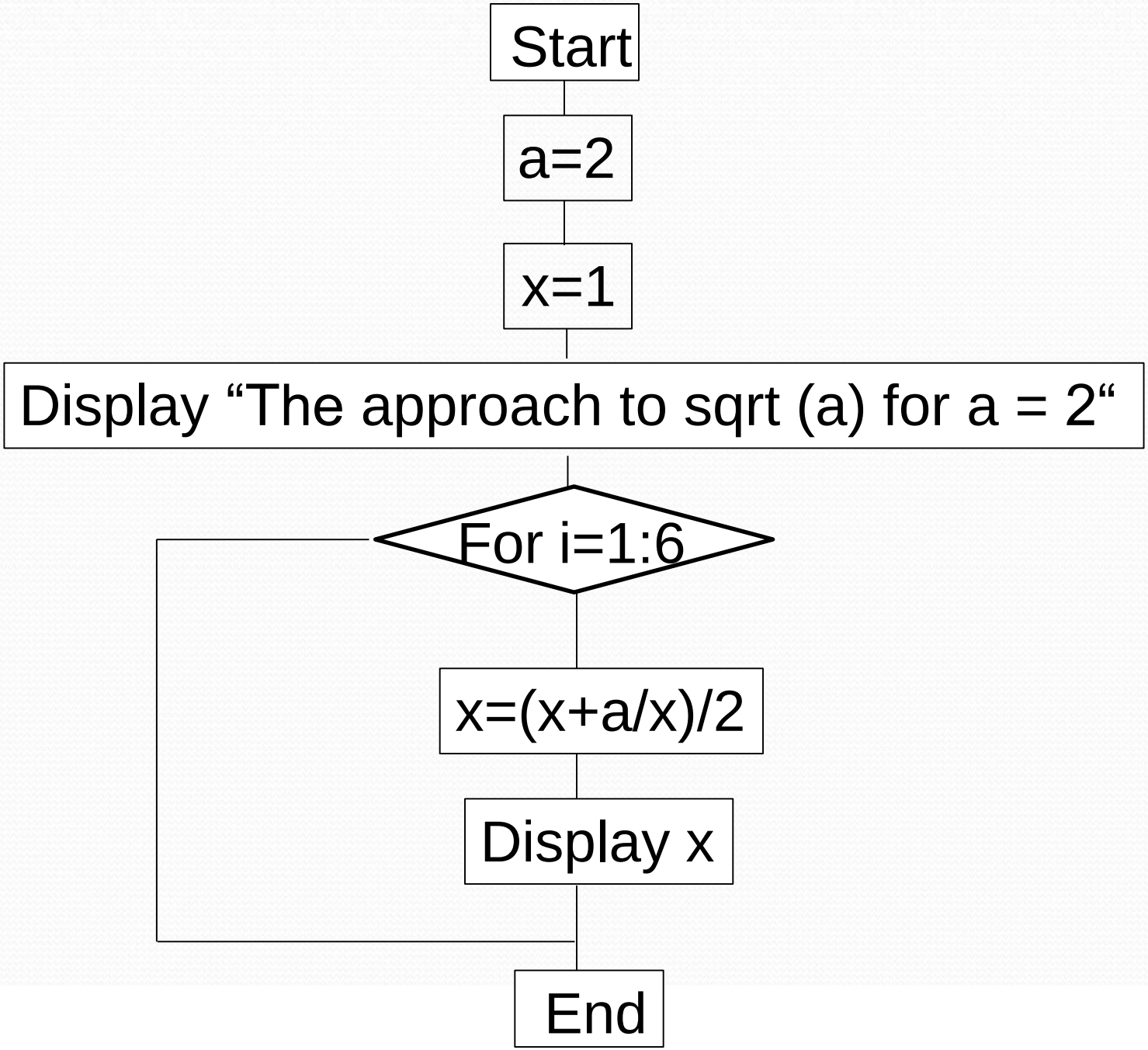
- for index = j:m:k
- statements
- end

7. Repeating with for

- Example: Square roots with Newton's method

  What are the algorithm to find the square root and a program with sample output for a = 2?

7. Repeating with for



7. Repeating with for

- Example: Square roots with Newton's method

```
a = 2;
x = 1;
disp( ['The approach to sqrt (a) for a = ', num2str(a)] );
for i = 1:6
x = (x + a / x ) / 2;
disp(x)
end
disp( 'Matlab"s value: ')
disp( sqrt(2) )
```

The value of x converges rather quickly towards the value calculated by MATLAB. Most computers and calculators use a similar method internally to compute square roots and other standard mathematical functions.

7. Repeating with for

- Example: Square roots with Newton's method:

```
a = 2;
x = 1;
disp( ['The approach to sqrt (a) for a = ', num2str(a)] );
for i = 1:6
x = (x + a / x ) / 2;
xi(i) = (x + a / x ) / 2;
disp(x)
end
disp( 'Matlab"s value: ')
disp( sqrt(2) )
```

7. Repeating with for

- Example: Factorials (!)

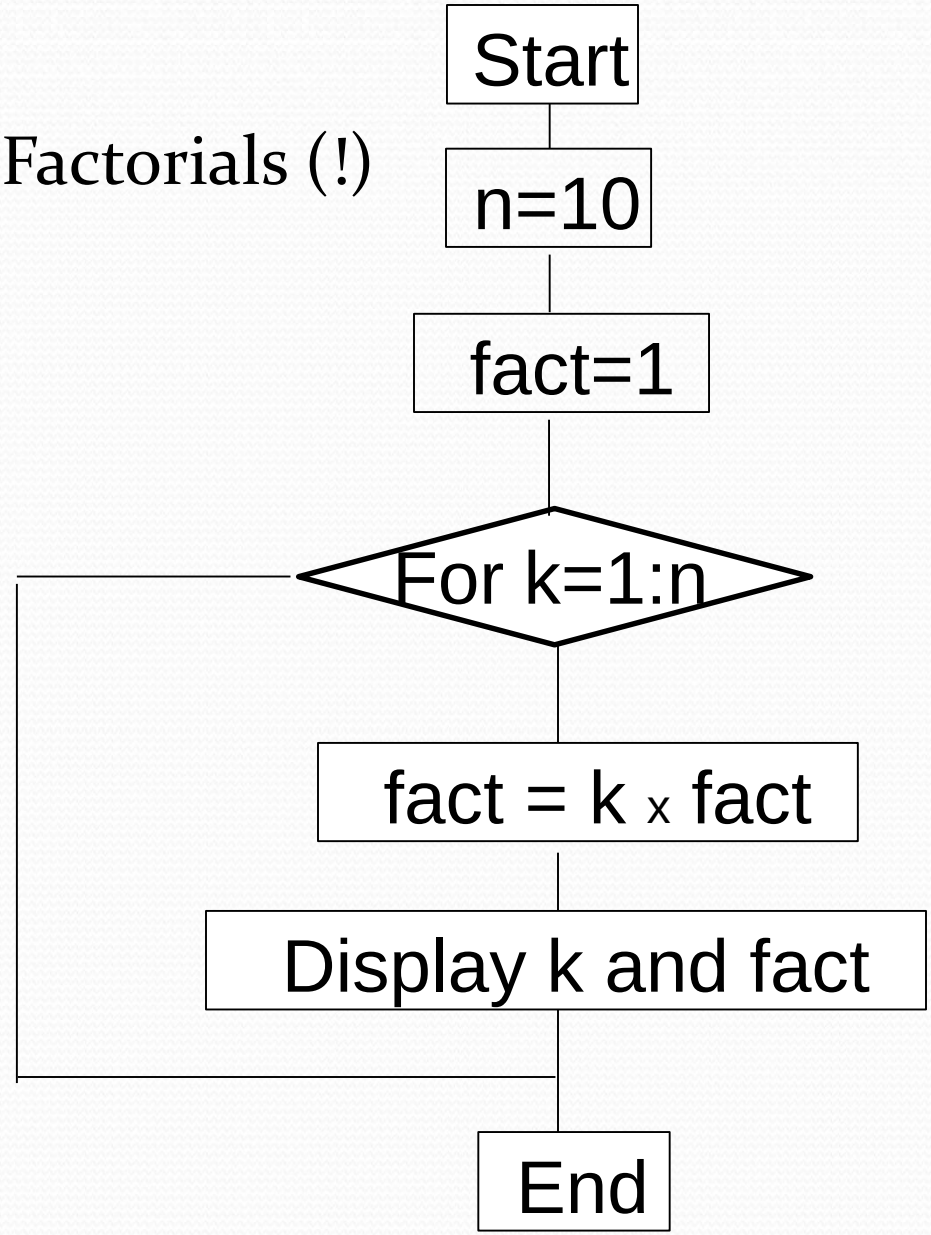
  Create an algorithm to generate a list of n! (n factorial or n shriek) knowing that:

  $n! = 1 \times 2 \times 3 \times \cdots \times (n-1) \times n$

  With 10 loops

7. Repeating with for

- Example: Factorials (!)



7. Repeating with for

- Example: Factorials (!)

  Matlab code:

```
n=10;
fact = 1;
for k = 1:n
fact = k * fact;  disp ( [k    fact]);
end
```

Try to figure out what is the largest value of n for which MATLAB can find n! Advise, move the disp statement out from the loop.

7. Repeating with for

- Example: Limit of a sequence

  The for loops are ideal for computing successive members of a sequence (as shown in Newton's method to calculate the square root of a number).

  Now, consider the following sequence

$$x^n = \frac{a^n}{n!} \quad \textit{for } n = 1, 2, 3, \ldots \textit{ and } a \textit{ is a constant}$$

  The question is: what is the limit of this sequence as n gets indefinitely large?

  Clearly, if we compute x^n directly as, we could get into trouble, and numerical overflow could occur.

7. Repeating with for

- Example: Limit of a sequence

  The situation is neatly transformed if we notice that is related tothe previous term n-1 as follows:

$$x^n = \frac{a^n}{n!}$$

$$x^n = \frac{a.a^{n-1}}{n.(n-1)!}$$

$$x^n = \frac{a}{n}.x_{n-1}$$

  There are not any numerical problem now.

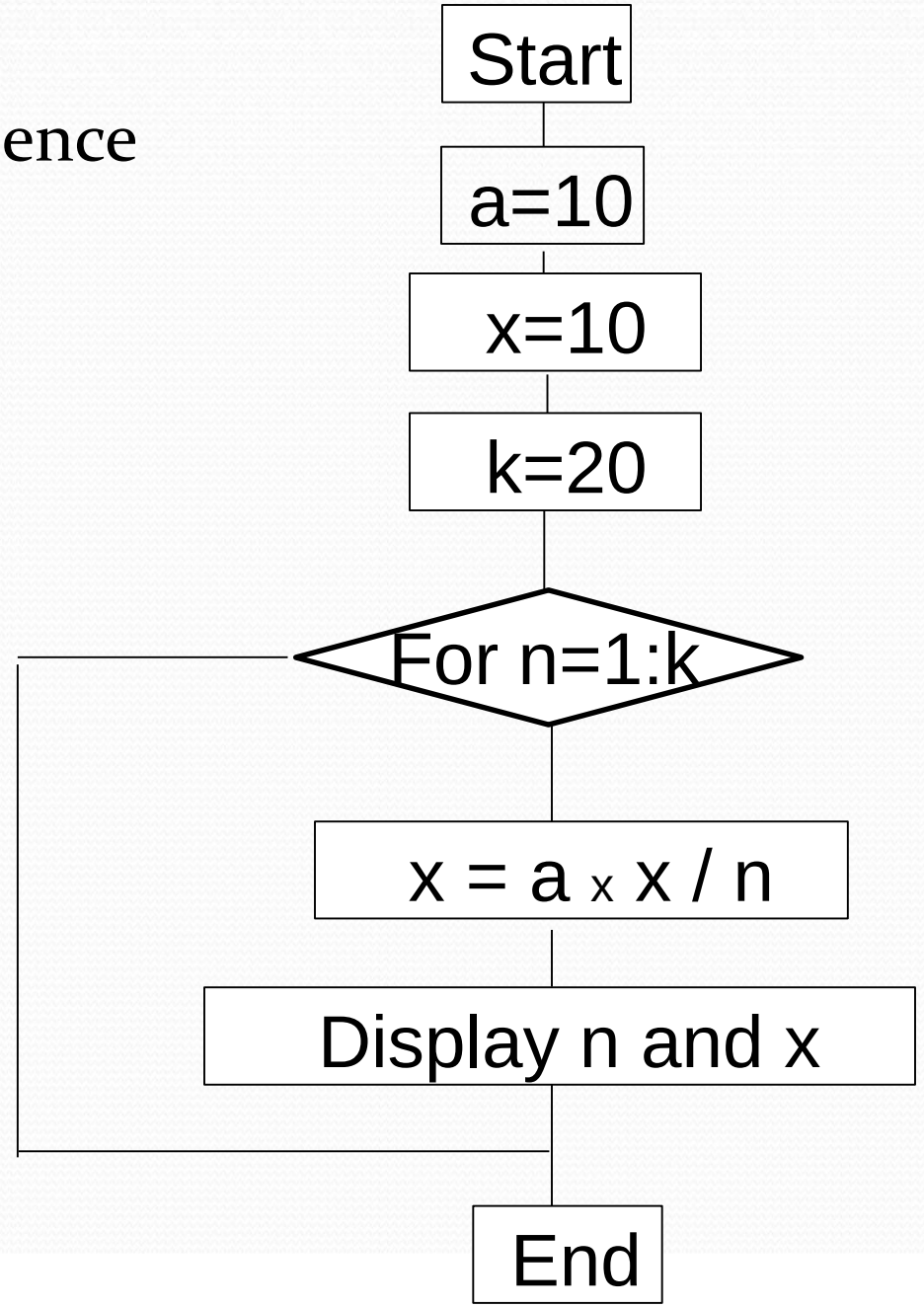
7. Repeating with for

- Example: Limit of a sequence

Create the algorithm to compute x_n for a = 10 , start with x = 10, and increasing values of n until k=20. Then display the result

7. Repeating with for

- Example: Limit of a sequence



7. Repeating with for

- Example: Limit of a sequence

The following program computes x_n for a = 10 and increasing values of n.

- a = 10;
- x = 10;
- k = 20;
- for n =1:k
- x = a * x / n; % you can also write: x = (a/n) * x ; disp([n x]);
- end

7. Repeating with for

Note the following points carefully:

1. j:k is a vector with elements j, j+1, j+2, …, k

7. Repeating with for

Note the following points carefully:

1. j:k is a vector with elements j, j+1, j+2, …, k
2. j:m:k is a vector with elements j, j+m, j+2m,… such that the last element does not exceed k if m > 0, or is not less than k if m < 0

7. Repeating with for

Note the following points carefully:

1. j:k is a vector with elements j, j+1, j+2, …, k
2. j:m:k is a vector with elements j, j+m, j+2m,… such that the last element does not exceed k if m > 0, or is not less than k if m < 0
3. On completion of the for loop, the index contains the last value used.
4. If the vector j:k or j:m:k is empty, statements are not executed, and control passes to the statement following the end statement.

7. Repeating with for

- for in a single line

If you insist on using for in a single line, although it is not recommended, here is the general form

- for index = j:k, statements, end

or

- for index = j:m:k, statements, end

7. Repeating with for

- for in a single line

If you insist on using for in a single line, although it is not recommended, here is the general form

- for index = j:k, statements, end

or

- for index = j:m:k, statements, end

Attention

-Do not forget the commas. Semicolons will of course do if appropriate.

-Statements can be one or more statements, separated by commas or semicolons.

-If you leave out end, MATLAB will wait for you to enter it. Nothing will happen until you do so.

7. Repeating with for

- More general for

A more general form of for is:

- for index = v

where v is a vector. The index moves through each element of the vector in turn.

Exercise

Create variables at the command window:

- `a = 2`
- `b = 3`
- `a + b`
- `first_string = 'My name is '`
- `second_string = 'yournamehere,andpleasedontcopyandpastei t,justwriteyourname,yourownname,thatonet hatyourparentsgaveyoumanyyearsago'`
- `first_string + second_string`

Exercise

Create variables based on other variables:

- `c = a * 2`
- `d = cos(b)`
- `e = c + d`
- `r = 5`
- `A = 2 * pi * r`
- `C = 2 * pi * r`
- `x = 0`
- `curve_f = sin(x) + cos(x/3+1)`

Exercise

- Create a matrix A of order 4*3 with elements 15 to 3 in a decreasing order arranged row-wise.
- Create a matrix B of order 3 * 4 with elements 1 to 12 arranged column wise.
- Find a suitable way to perform element wise sum, product, and power of A and B.

7. Repeating with for

- Avoid for loops by vectorising

  There are situations where a for loop is essential.

  However, the way MATLAB has been designed, for loops tend to be inefficient in terms of computing time.

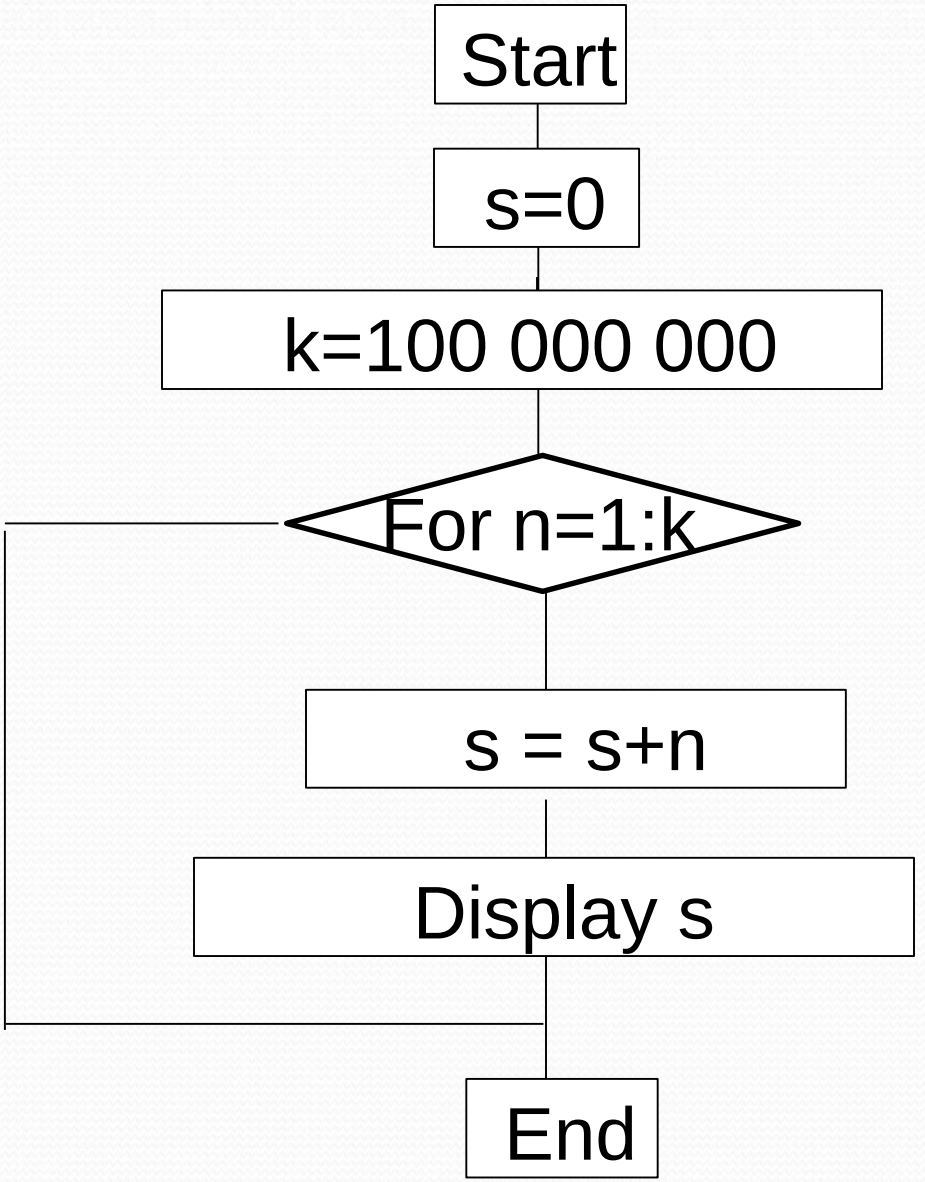
  On many occasions, it is possible to vectorize the expression, making use of array operations where necessary.

  That is shown in the following example.

  Suppose we need to evaluate $\sum_{n=1}^{100\,000\,000} n$

  Create the algorithm

7. Repeating with for



7. Repeating with for

- to = clock;
- s = 0;
- for n = 1:100 000 000
- s = s + n;
- end
- disp(s)
- etime(clock, to)

The MATLAB function *clock* returns a six-element vector with the current date and time in the format year, month, day, hour, minute, seconds. So *to* records when the calculation starts.

The function *etime* returns the time in seconds elapsed between its two arguments.

7. Repeating with for

- to = clock;
- n = 1:100 000 000;
- s = sum(n)
- etime(clock, to)

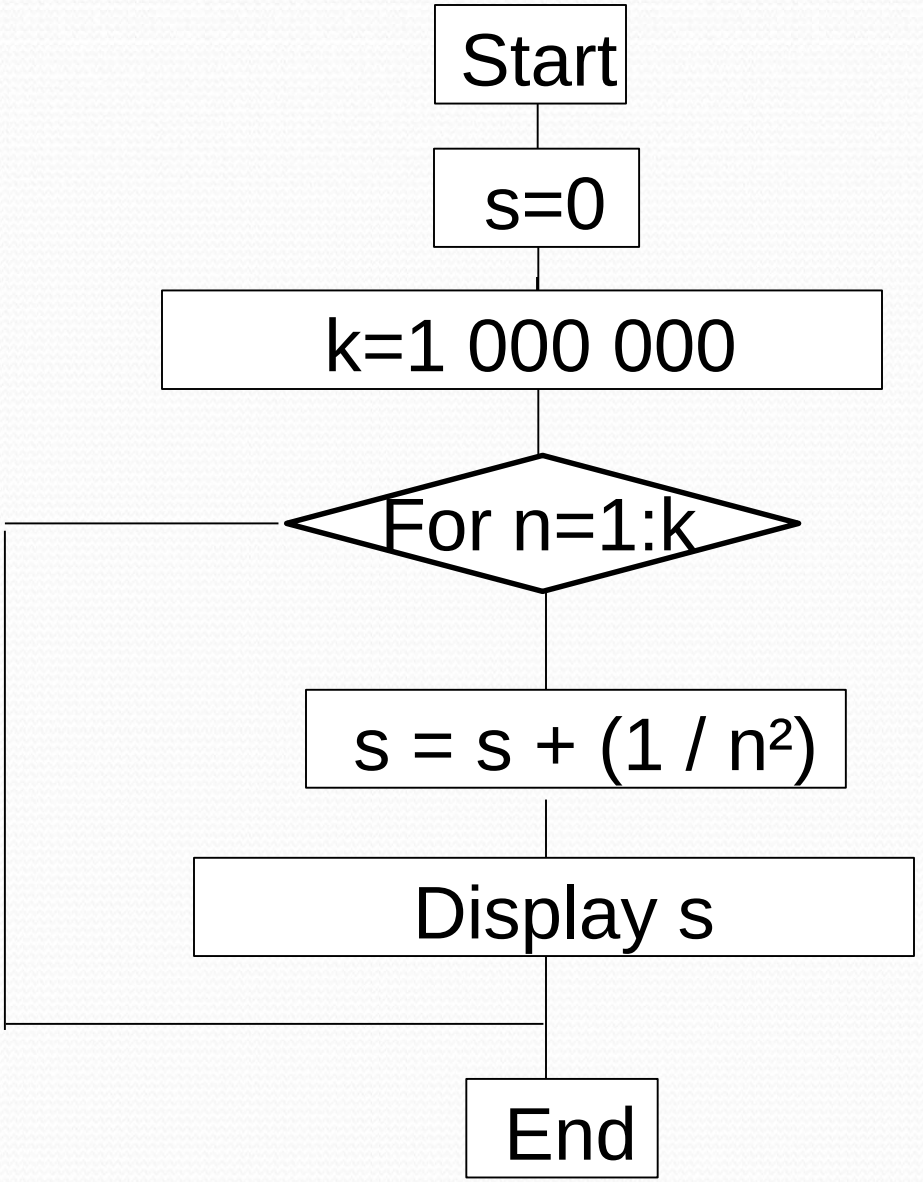
7. Repeating with for

The following shows a neater way of monitoring time using the tic and toc functions. Let us see how while evaluating

$$\sum_{n=1}^{1\,000\,000} \frac{1}{n^2}$$

Create the algorithm

7. Repeating with for



7. Repeating with for

The following shows a neater way of monitoring time using the tic and toc functions. Let us see how while evaluating

$$\sum_{n=1}^{1\,000\,000} \frac{1}{n^2}$$

- tic
- s = 0;
- for n = 1:1 000 000;
- s = s + 1 / n^2;
- end
- disp(s);
- toc

7. Repeating with for

- tic
- n = 1:1 000 000;
- s = sum(1 ./ n.^2)
- toc