nvkurbatova@sfedu.ru

Scientific Computer Software

(MatLab)

Presentation 4.

Topic 10. Procedures and programming elements

Курбатова Наталья Викторовна, к.ф.-м.н.,
доцент кафедры математического
моделирования, мехмат, ЮФУ

nvkurbatova@sfedu.ru

Contents:

- Procedures **Inline**
- **Anonymous.** Procedures without name
- Procedures with and without parameters
- Subprocedures
- Procedures with a variable number of I/O parameters
- Feval. Examples
- Programming operators: conditional, loop, switch, exception interception operator

INLINE - System Constructor

NameI=inline(funChar); % NameI – inline, funChar – expression of characteristics

Args=symvar(funChar); % Args – funChar arguments

g = inline(funChar, arg1, arg2,…)

isa(g, 'inline') % контроль типа

INLINE is n’t used in older versions!

Instead inline -- anonymous functions!

```
1     %% inline
2     z=inline('x.^2+y.^2-1');
3     z(2,3)
4     arg=symvar(z)
```

```
z: 1x1 inline =

     Inline function:
     val(x,y) = x.^2+y.^2-1
```

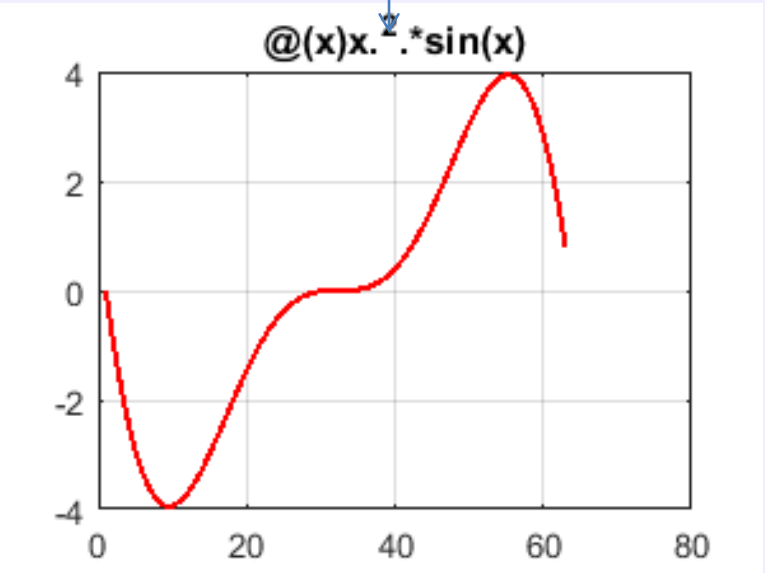
```
arg = 2×1 cell array
    'x'
    'y'
```

Anonymous function

FuncHandle = @FunctionName;

@ - конструктор function_handle

FuncHandle = @(ArgList)ExpressionChar



myAnonymous.m

```
function myAnonymous % without parameters
h=@(x)x.^2.*sin(x); % anonymous
val=-pi:0.1:pi; % h(val) –   ans vector
FuncPlot(h, val)
function FuncPlot(h, val) % subprocedure
  if isa(h, 'function_handle')   % Check  h
                                 as a function_handle!
   A = h(val);              % Construct A as h-function of val
     plot(A,'r-','linewidth',1.5); % Plot the resulting data
     title(func2str(h)), grid on   % Convert function to Char
    end
```

Procedures with and without parameters (PWOP)

Пример: файл myFunction.m

```
function myAnonymous % PWOP
h=@(x)x.^2.*sin(x); % anonymous
val=-pi:0.1:pi;
h(val) % vector ans - anonymous
```

Benefit

- **PWOP - an alternative to the script file**
- **All variables are local**
- **Execution is faster (no I/O operations)**
- **The name of the script file and the procedure are the same**

nvkurbatova@sfedu.ru

Subprocedures

- **Subprocedures –** are located in the body of the head file
- **Subprocedures** can have the name of the system procedure; the execution priority is given to the subprocedure
- **Subprocedures** is "not visible" to any other program (this allows f.e. to redefine standard functions – to please a sense of self-importance)
- **Procedure** is performed faster than the external procedure

For an **example** of the FuncPlot(h, val) subprocedure, see slide 4

Note that the subprocedure, for example, is FuncPlot(h, val) it may not have output parameters!

Function

- Function has **one** output parameter **(«conditionally» one)**!
- A function can be part of an expression, unlike ***?
- **The name** of the function must match the name of the file in which it is defined!
- The last executed statement must be assigned an identifier with the name of the function!
- The function is defined by a syntactic construction:

NameF.m %

Function header:

```
function outputmy=NameF(input1,…,inputm) % the types of arguments are not explicitly specified!
% description of the parameter assignment
Function body: { function content - codes}
% the last executable statement :
outputmy=expression
end % not necessary, sometimes
```

Procedure

- **The procedure cannot be part of an expression!**
- **The name of the procedure must match the name of the file in which it is defined!**
- **The name of the external procedure cannot match the name of the built-in procedure (function)!**
- **There are no restrictions on the number of I/O parameters!**
- **The procedure is defined by the same syntactic construction as the function:**

Procedure header **:**

function [out1,…,outk]=NameF(input1,…,inputm) % the **Types are not explicitly specified!**

% description of the types and purpose of the parameters

Тело процедуры: {The purpose is to determine out1,…,outk }

end %

Procedures with a variable number of I/O parameters

Procedure header :

function [out1, …, outk, varargout]=NameF(inp1, …, inpn, varargin)
% setting parameters:
% inp1, …, inpn – required input parameters
% varargin – is a system array of cells that contains input parameters that the user passed to the procedure during the request, in addition to the required ones;
% out1, …, outk – required output parameters
% varargout – is a system array of cells that contains output parameters that the user needs, in addition to the required ones;
The programmer, in assigning parameters (help inside the procedure), specifies which input/output (I/O) parameters are provided in this procedure!

Procedure body (definition I/O parameters):

p=nargin % determines the number of all input parameters, then
p-n % is the number of input parameters in varargin during the current call
m=nargout % the number of all output parameters is determined, then
m-k % – the number of input parameters in varargout for the current call

Procedures with a variable number of I/O parameters (continuation)

Procedure body :

1) Determining the size of arrays of variable-length I/O parameters

p=nargin; m=nargout

pn=p-n, mk=m-k % size of varargin and varargout, respectively

2) Assigning values from varargin to local variables:

```
if (p-n>0), Par_1=varargin{1}
elseif (p-n>1)
Par_2=varargin{2}
    ooo
elsif(p-n>p-n-1) % or else
Par_pn=varargin{p-n}
end
```

3) The main ALGORITHM

4) Varargout formation:

```
if (m-k>0)
varargout{1}=expession_1;
elseif (m-k>1)
varargout{2}=expession_2;
    ooo
elsif(m-k>m-k-1) % or else
varargout{m-k}=expession_mk;
end
```

Instead of a conditional operator, you can use a switch:
switch, case, otherwise, end;

Perform function. Feval

syntax:

[y1,...,yn] = feval(fun,x1,...,xk) – execute the function; (x1,...,xk) – its arguments (one or more); fun – function_handle **or** имя built in function **or** external procedure.

Usage example:

Head procedure without parameters:

```
n=3, for i=1:n
      namef={'cos',@sin,'myextFun'};
      result(i)=feval(namef{i},pi/6);
      end
[result(1:n)]
```

External file: myextFun.m

```
function r=myextFun(arg)
r=(sin(arg))^2
end
```

The file name and the name of the external function are the same!

Explain what are the elements of namef?

Check it out!

>>which('cos') - finds the path to the built-in function

Conditional operator

Short form: if expression**, end**;

Full form : if expression1

statements1

elseif expression2

statements2

else

statements3

end

expressions: checktype(see help is* – isempty, isfinite, isnumeric, isvarname . . .),

relational operators **(help relop),**

logical (all, any), exist(string)

statements: the sequence of commands - the main code

Syntactic analogues of cycles in MatLab

Simplifying the syntax in ML (cycle replacement):

- **Element-by-element assignment:** A(i,j)= B(i,j) ↔ **A=B**
- **Assigning a constant to a matrix or part of it: A(1:k,p:p+k-1)=pi or A(:,:)=pi (Is that so? A=pi)**
- **Optimization techniques:** f.e. summation of vector elements **b – vector, b*(ones(size(b)))'**
- **Assignment by condition (cycle + condition): ops: ==, >, <, ~=** A(A ops value)=expr; f.e.(A(A>0)=rand(1))

A cycle with a known and unknown number of repetitions

Syntax

Case of known number of repetitions :

Short form: **for** variable = expr**,** statements**, end**; % в строку

Full form 1-st: **for** variable = initial_value:step:end_value
statements
end

Full form 2-nd: **for** variable = [vector] % or variable = [**matrix** or cell] **?**
statements
end

initial_value – the initial value of the **i**ndex **v**ariable(i.v.);

step – index step; in this case **for variable = initval:endval** (step=1 – default);

end_value– the final value of the index variable.

the case of an unknown number of repetitions:

короткая форма: **while**, expression, statements, **end**;

полная форма: while expression
statements
end;

Examples of cycles :

1) for e=eye(n), e, end; % at the *i*-th step, the *i*-th column vector is calculated e(:, *i*)
2) while (A&B);
3) while (A|B);
4) while (b~=0) &(a/b>3*pi)
   if exist('myfun.m')&(myfun(x)>=y)
   if iscell(A) & all(cellfun('isreal',A)) % **explain!** See details: >>help cellfun(fun,A)
5) eps=1; while(abs(1+eps)>1), eps=eps/2, end; eps=2*eps

Explantion (2-5):

(2): if A (logical) is zero, then B is not calculated, the result **2)** for any B is false;

(3): if the elements of A are logical units, the matrix B is not calculated, the result **3)** for any B is true;

(4): the loop is executed until a/b > 3*pi and division by zero is excluded **4)**;the existence and magnitude of the function, the materialityof the elements of the array of cells are controlled

(5): at first glance, it looks like an endless cycle, but it will stop when it **reaches machine zero;**

It would be nice to find out the number of steps (divisions)!

The first three students who send me the answer and the code will receive 3 points!!!

Example 1 of the infinite loop

```
1 -    clear all;
2 -    file1 = fopen('output_code.txt','r'); % открываем output_code.txt
3      %                                      для чтения
4 -    file2 = fopen('line.txt', 'w');        % открываем  line.txt
5      %                                      для записи
6 -    while 1    % выполняется всегда
7 -    tline=fgetl(file1);
8 -    if ~ischar(tline), break, end          % выход из цикла по break
9 -    fprintf(file2, '%s;\n', tline);       % подавляем строку-вывода ";"
10 -   disp(tline)
11 -   end
12 -   fclose(file1);
13 -   fclose(file2);
```

it is always executed

edit_string.m | input_code

```
1   clear
2    x=0:0.1:pi
3    y=sin(x)
4   plot(x,y)
5    hold on
6    grid on
7   g=cos(x)
8    plot(x,g)
9    title('two graphs')
10  disp('thats all!')
```

edit_string.m | input_code

```
1   clear;
2    x=0:0.1:pi;
3    y=sin(x);
4   plot(x,y);
5    hold on;
6    grid on;
7   g=cos(x);
8    plot(x,g);
9    title('two graphs');
10  disp('thats all!');
```

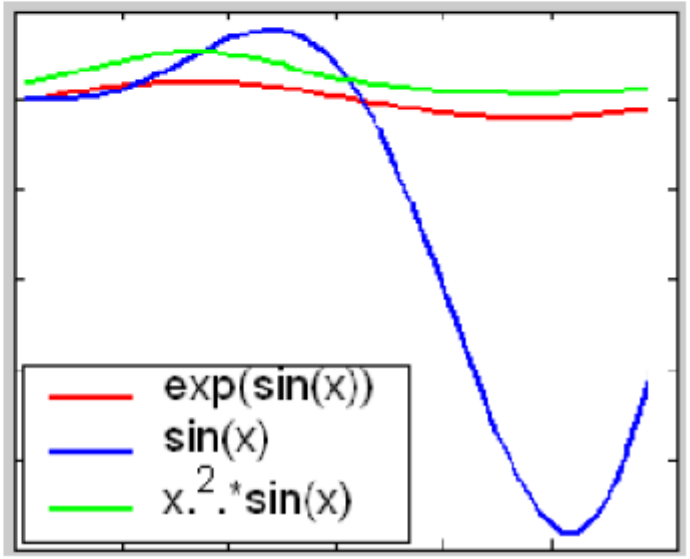
output_code.txt

fgetl (file) - % reading lines from a text file

status = feof (fileID) returns one if the previous read operation sets the end of the file

Example 2nd of the infinite loop

```
clear
% example if
a=randi(10)
if nnz(a)>prod(size(a))/2
    s=sparse(a)
end
% example for
   x=[0:0.1:2*pi]';
   mycolor={'red','blue','green'}
   leg={}
A={'sin(x)','x.^2.*sin(x)','exp(sin(x))'}
for f=A
    k=1
    while ~strcmp(A{k},f{:})
        k=k+1
    end
    r=plot(eval(f{:})),
    set(r,'color',mycolor{k})
    set(r,'linewidth',1.5)
    leg=union(leg,f)
        hold on
end
    t=legend(leg{:})
    set(t,'fontsize',14)
```



Switch

```
switch calculated_expression
case possible_expression_1
statements,
...
case possible_expression_end
statements,
end
```

$calculated\_expression$ – вычисляемое выражение, в т.ч. строка; сравнивается со значениями $possible\_expression\_1, \ldots possible\_expression\_end$; при совпадении выполняются инструкции "помеченной" траектории

Пример переключателя:

```
method = 'Bilinear';
switch method
case {'linear','bilinear'}
disp('Method is linear')
case 'cubic'
disp('Method is cubic')
otherwise
disp('Unknown method')
end
```

Interception Operator

```
try
statements
catch
statements
end
```

```
>> a=1:10; a*a
Error using  *  Inner matrix
dimensions must agree.

>> lasterr
ans =
Error using * Inner matrix
dimensions must agree.
```

- The last error system message stores in the system variable lasterr;
- an erroneous situation is possible at the try-catch stage;
- the error is processed at the catch-end stage;
- a logically determined list of errors should be provided (prepared);
- using the findstr operator, you should identify the essence of the error;
- use the switch to select a branch of the algorithm.

nvkurbatova@sfedu.ru

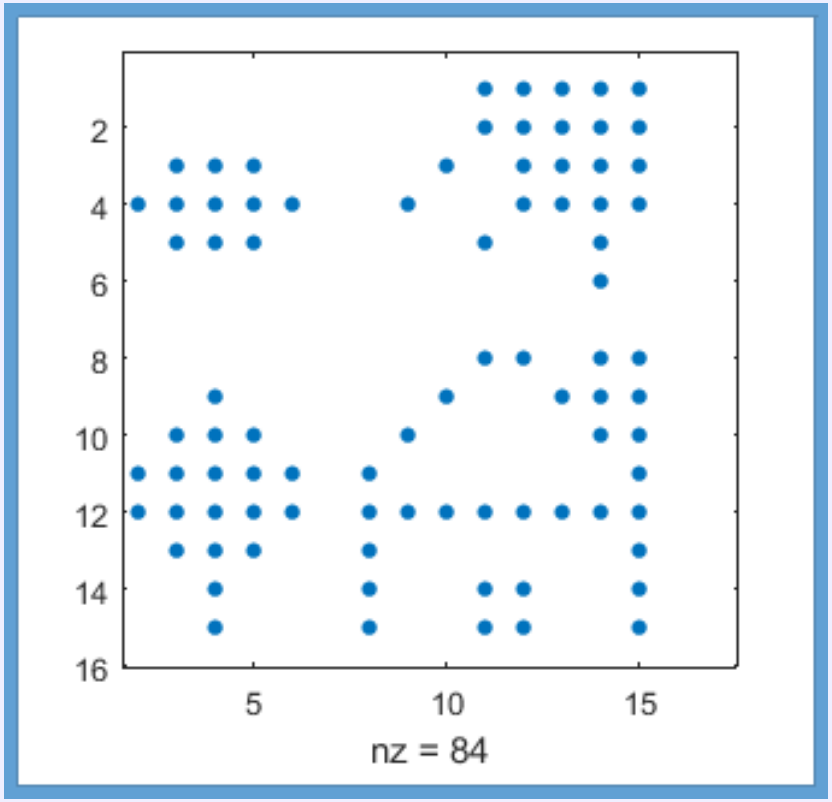
Bonus

In the latest versions of MatLab, functions for determining local minima and maxima are "working":

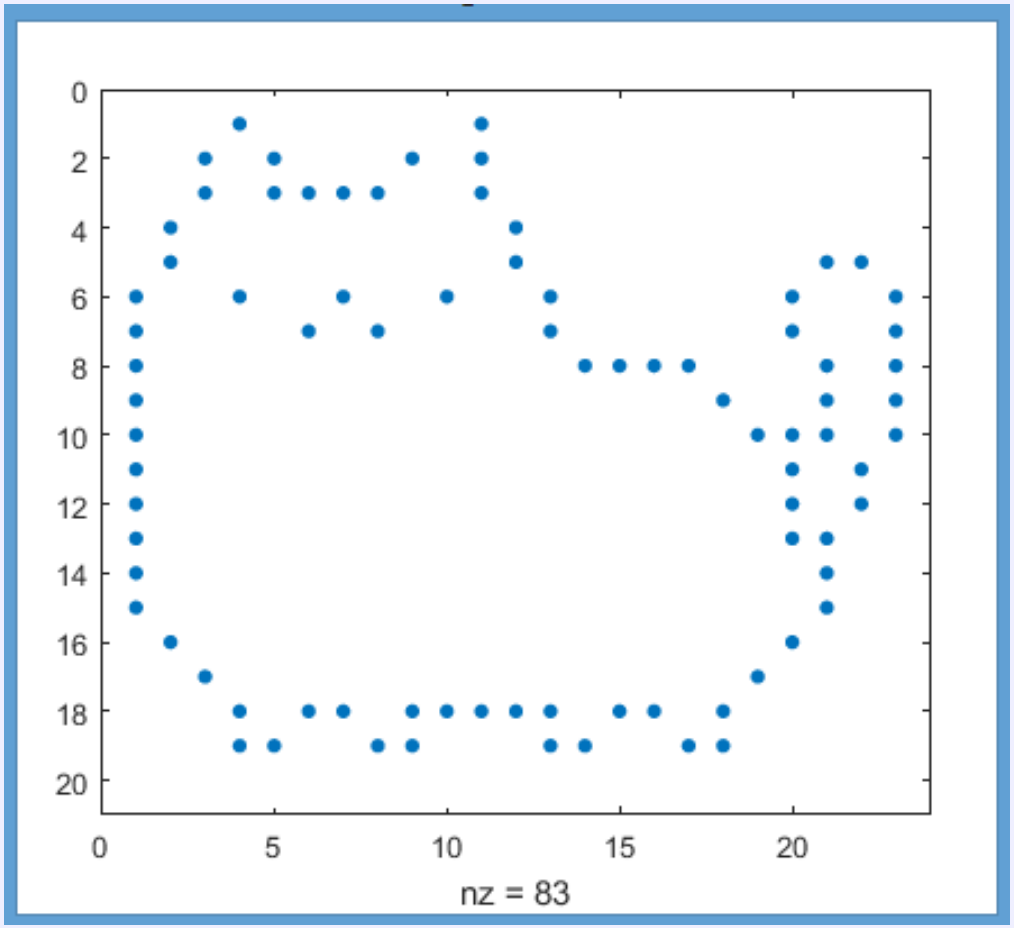
islocalmin u islocalmax !

It is interesting (?) to get into the code of these system functions, compare it with the algorithm in L.3 p.16!!!

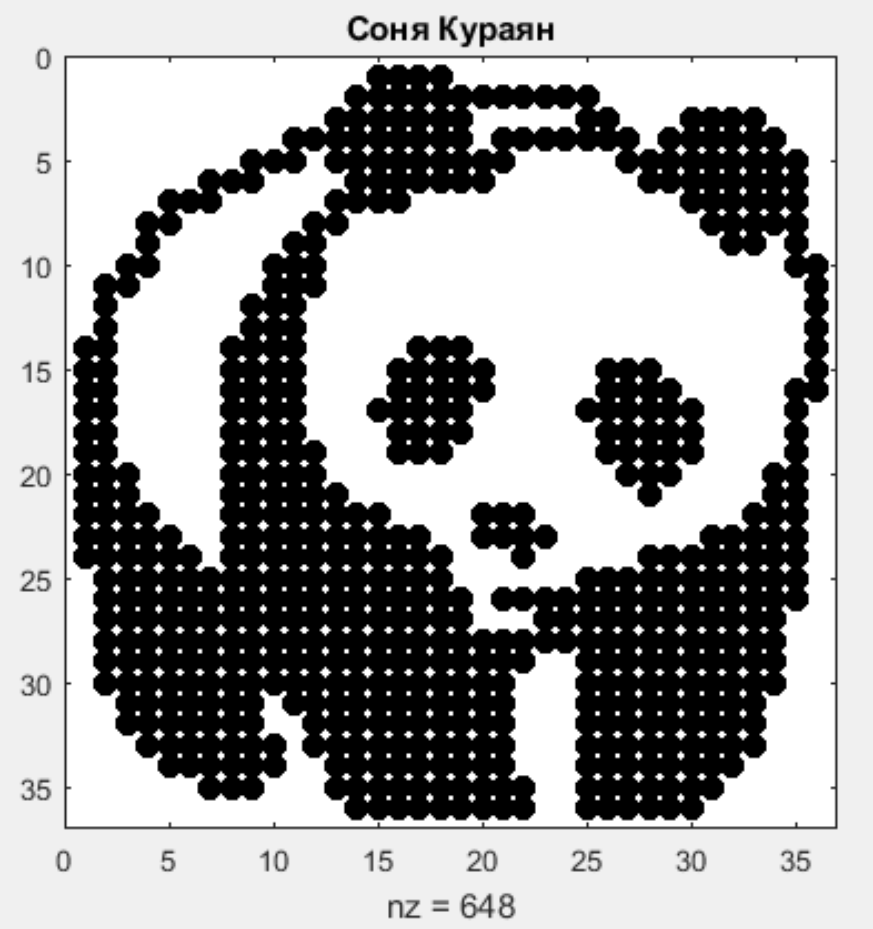
Ilya Vyaznikov: he drew the picture first in the class!



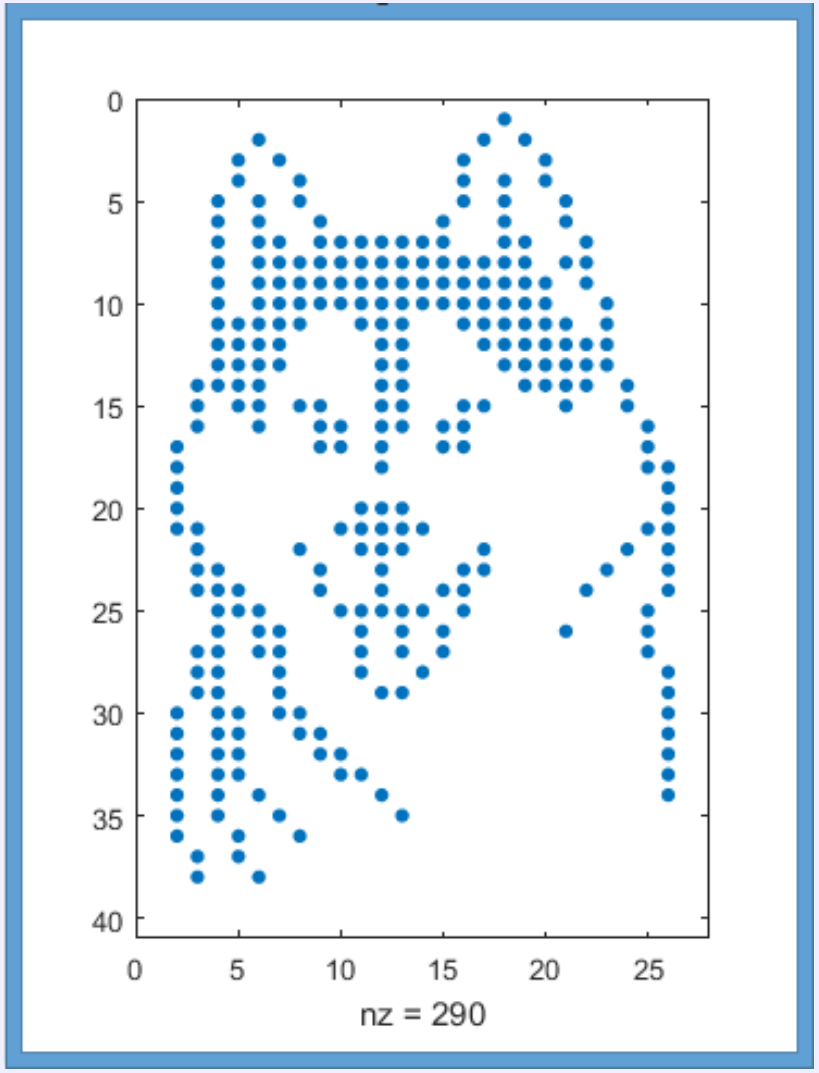
Tinchurin Ruslan



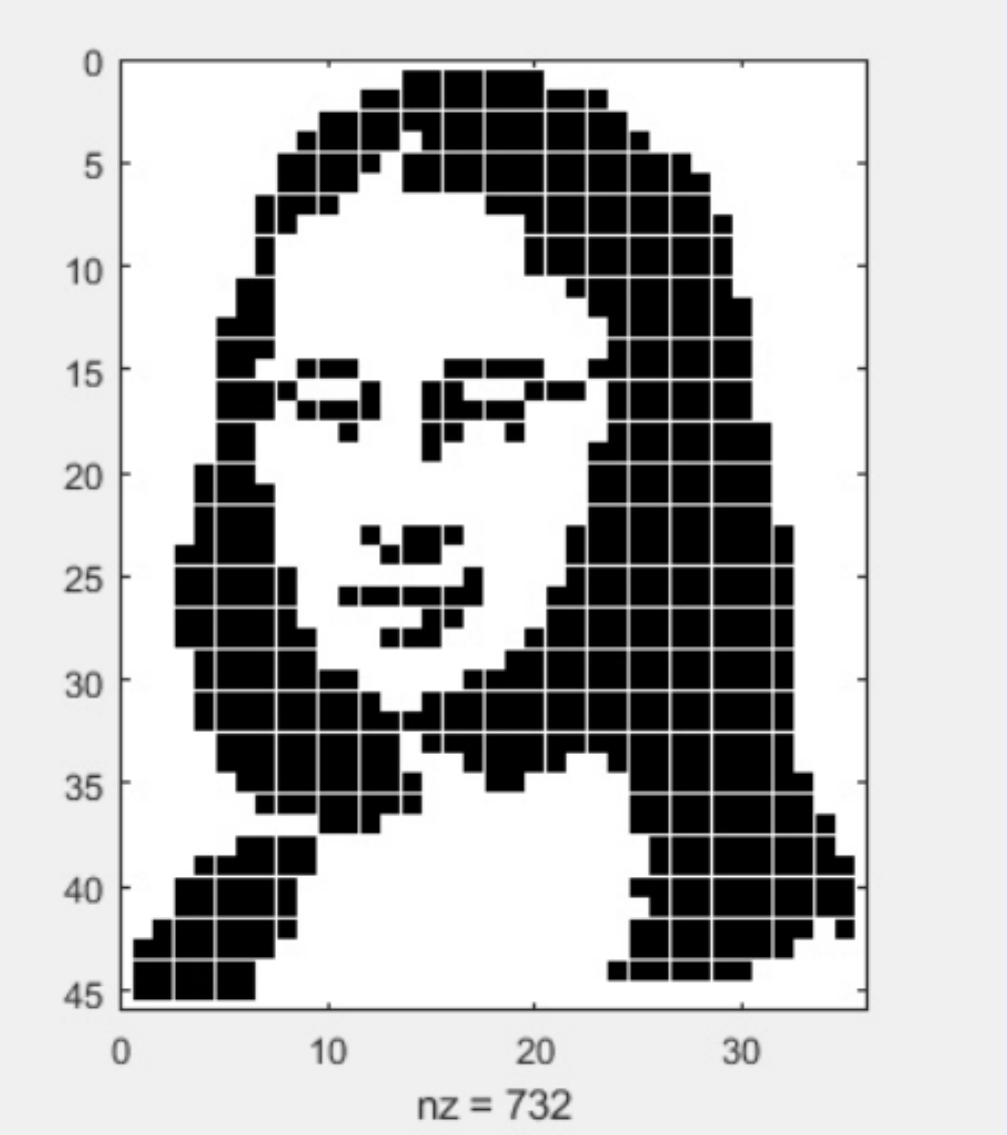
Sofia Kurayan



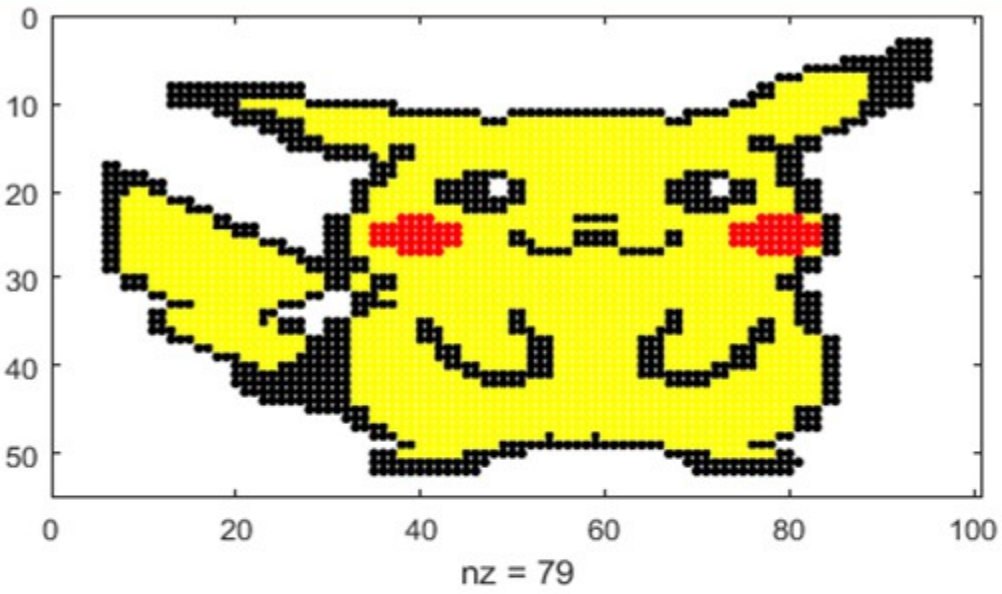
Melekhova Daria



Zakharenko Anastasia, Semenchenko Artem Turygin Sasha, Tsulaya Lena

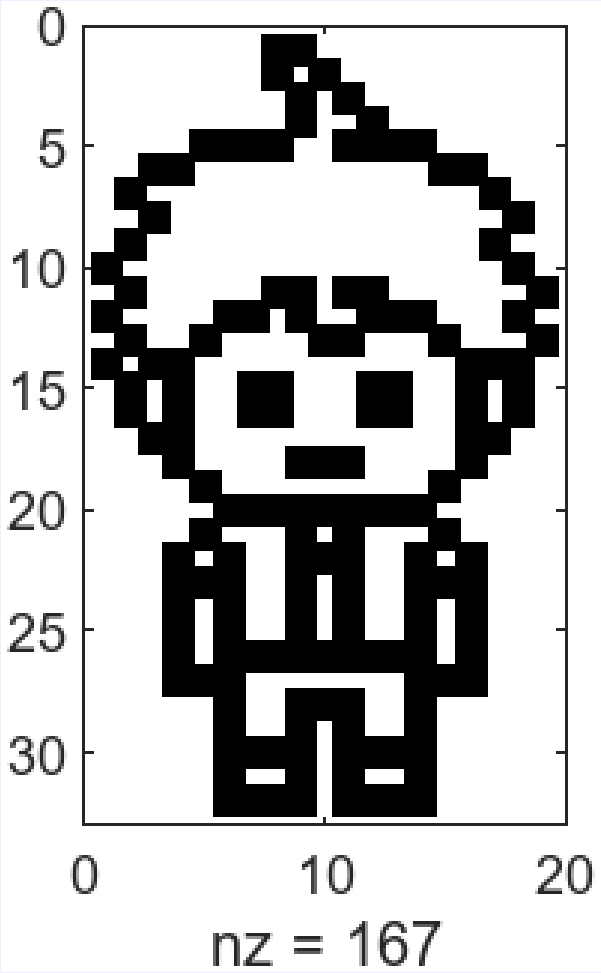


Maxim Movchan, 4 гр.

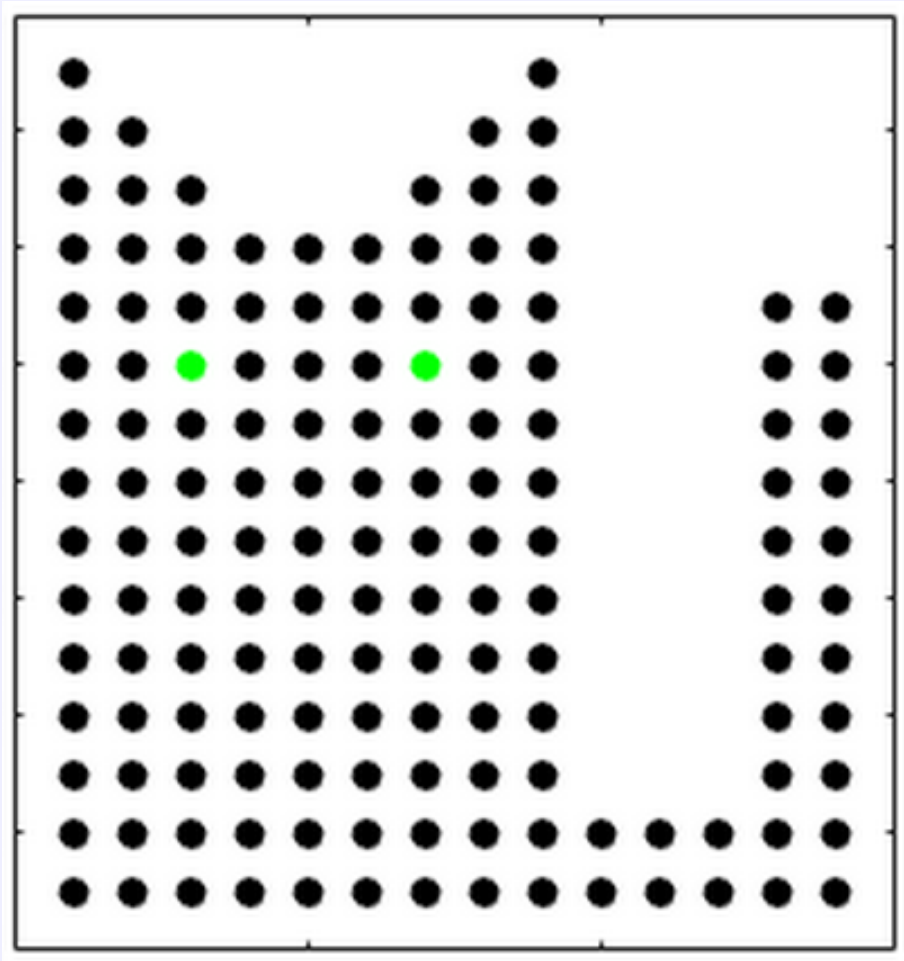



```
spy(A == 1, 'black', 10)
hold on
spy(A == 2, 'yellow',10)
spy(A == 3, 'red', 10)
hold off
```

2024: Timur Kaplunov



Veronika Shpakova



nvkurbatova@sfedu.ru

Thanks for your attention!

“The eyes are afraid, but the hands do”

A Russian folk proverb!