



# Пакеты научных вычислений

## Лекция 8. Специальные возможности MatLab

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



# Содержание:

- **Анимация. Конструктор animatedline**
- **3D примитивы**
- **Построение винтовой линии**
- **К оформлению в т.ч. GUI interpreter  
TEX и Latex**
- **Создание приложений**
- **Разное**



## Пример 1

### эволюционного пострениа кривой

```
figure; h = animatedline;  
axis([0 6*pi -3 2])  
h.LineWidth=1.5;  
h.Color=[1 0 0]  
x = linspace(0,6*pi,1000);  
y = cos(2*x).*exp(sin(x));  
    for k = 1:length(x),  
addpoints(h,x(k),y(k));  
        drawnow;  
        pause(0.025)  
    end
```

## Синтаксис и алгоритм:

- 1) `h = animatedline` декларирование анимационной линии
- 2) Построение всех точек (координат) линии
- 3) Последовательное добавление точек кривой анимационной линии – **addpoints**
- 4) Обновление процесса отображения точек – **drawnow** ( для восприятия **pause(sec)** )
  - a) **drawnow limitrate** -  
20 визуализаций / сек
  - a) **drawnow nocallbacks** (по нажатию кнопки в GUI)
  - b) **drawnow limitrate nocallback** –  
ограничивает количество обновлений, если средство визуализации занято



# Анимация

nvkurbatova@sfedu.ru

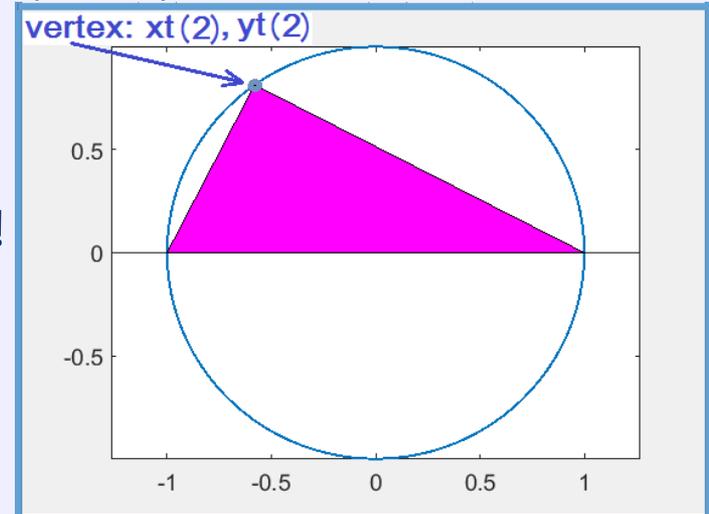
## Пример 2

```
clear  
theta = linspace(-pi,pi); xc = cos(theta); yc = -sin(theta);  
plot(xc,yc, 'linewidth',1.5); axis equal, hold on  
xt = [-1 0 1 -1]; yt = [0 0 0 0];  
t=area(xt,yt); % начальный треугольник  
hold off % не нужны прежние треугольники!
```

**Начинаем движение вершины (vertex)**

**По окружности:**

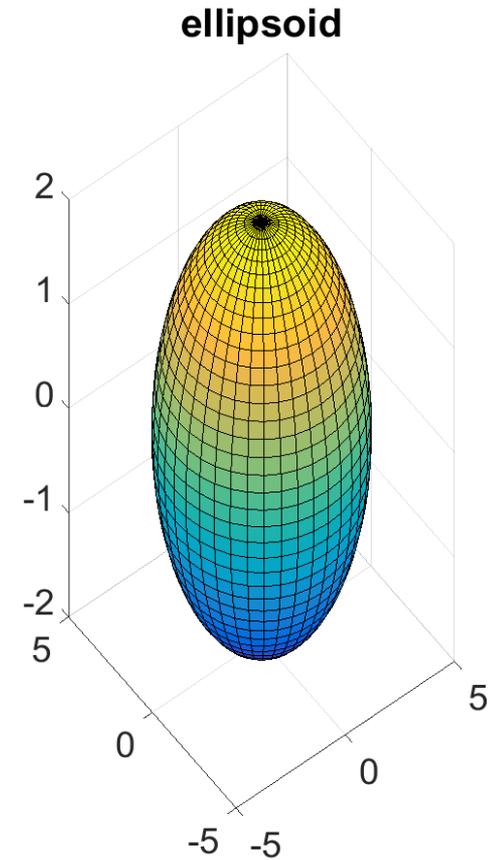
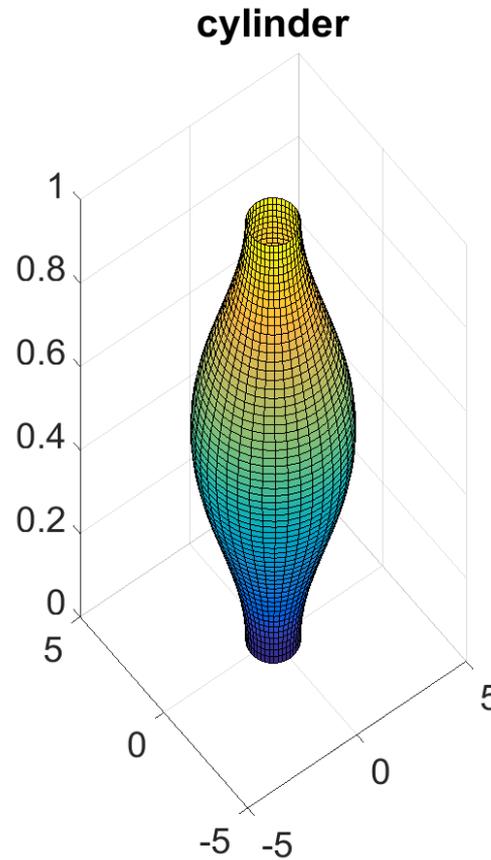
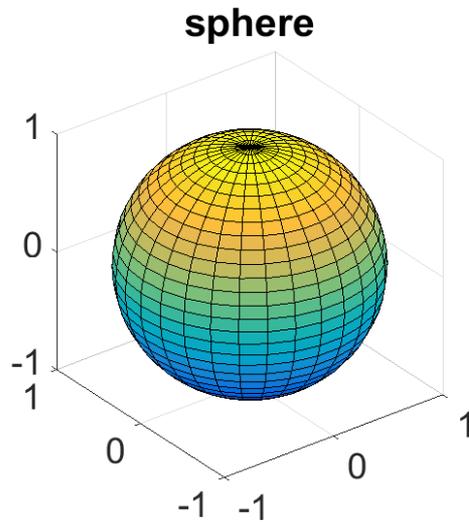
```
for j = 1:length(theta)%-10  
    xt(2) = xc(j); yt(2) = yc(j); % determine new vertex values  
    t.XData = xt; t.YData = yt; % update data properties  
    t.FaceColor='magenta'  
    drawnow limitrate % DISPLAY updated figure  
    % опция limitrate ограничивает количество обновлений  
    % до 20 кадров в секунду  
    pause(0.05)  
end
```





# 3D Поверхности: сфера, цилиндр, эллипсоид

Не поднимается язык сказать – примитивы!





## 3D Поверхности: синтаксис и код

### Constructor - sphere, N=20 - default

```
subplot(1,3,1); N=30; title('sphere')
```

```
[X,Y,Z] = sphere(N); surf(X,Y,Z); axis square
```

### Constructor - cylinder, N=20 - default

```
subplot(1,3,2); title('cylinder')
```

Радиус цилиндра переменной величины вдоль OZ:

```
eta=-pi:0.1:pi; Rc=2+cos(eta); %Rc - R current
```

```
[Xc,Yc,Zc] = cylinder(Rc,40); % R=1 or cylinder(R)
```

```
surf(Xc,Yc,Zc);
```

### Constructor - ellipsoid, N=20 - default

```
subplot(1,3,3)
```

```
[XC,YC,ZC,XR,YR,ZR,N]=deal(0,0,0,4,4,2,40);
```

```
[X,Y,Z]=ellipsoid(XC,YC,ZC,XR,YR,ZR,N);
```

```
surf(X,Y,Z); title('ellipsoid')
```

**Результат на предыдущем слайде !**



## Строим винтовую линию:

$H$  - cylinder height;  $\text{phi}=0:\text{step}:2*\text{pi}$ ;

$h$  is the distance between the point on the helix and the XOY plane after a complete rotation (**высота одного витка**)

```
H=10; h = 2; r=5; x=[]; y=[]; z=[]; step=0.01; subplot(1,2,2)
```

**the number of points on the spiral at full rotation:**

```
n=length(0:step:2*pi); Hc=0; % Hc - H current
```

```
for Hi=0:h:H-h
```

```
    for phi=0:step:2*pi
```

```
        x=[x(:);r*cos(phi)];
```

```
        y=[y(:);r*sin(phi)];
```

```
        Hc=Hc+h/n; z=[z(:);Hc]; % Explain!
```

```
    end
```

```
end
```

```
plot3(x,y,z,'r-')
```

```
title('\fontname{Courier}\fontsize{14}\color[rgb]{0.05,0.0,0.95} h=2,0<\phi<2\pi', ...
```

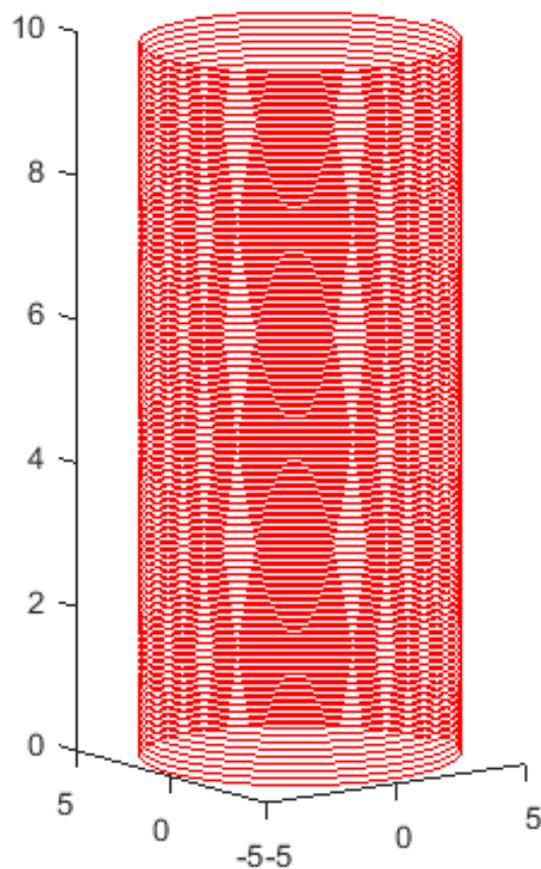
```
'interpreter','tex') % Explain!
```

**'interpreter','tex' - default ML  
v2014...!**

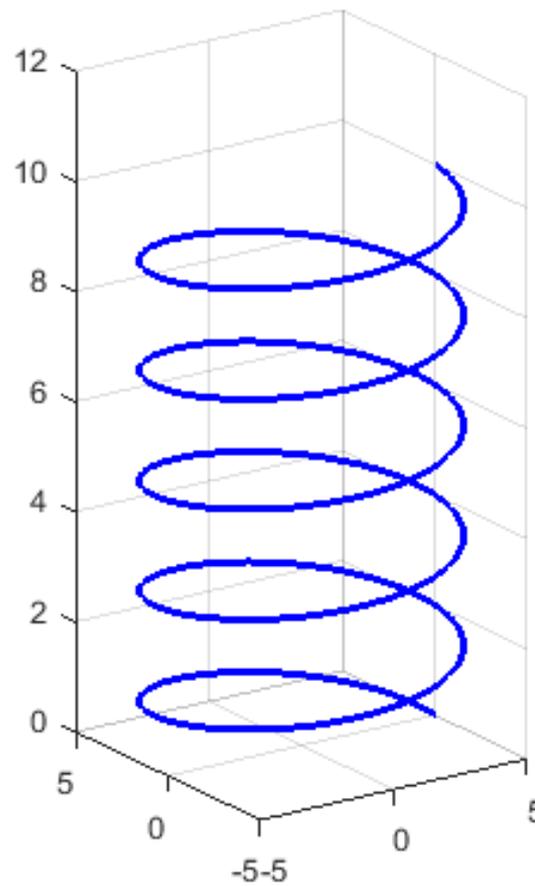


# Пример "цилиндра" и винтовой линии

$h=0.1, 0 < \phi < 2\pi$



$h=2, 0 < \phi < 2\pi$





# >>help interpreter

nvkurbatova@sfedu.ru

## TeX Markup

Modifier	Description	Example
<code>^{ }</code>	Superscript	<code>'text^{superscript}'</code>
<code>_{ }</code>	Subscript	<code>'text_{subscript}'</code>
<code>\bf</code>	Bold font	<code>'\bf text'</code>
<code>\it</code>	Italic font	<code>'\it text'</code>
<code>\sl</code>	Oblique font (usually the same as italic font)	<code>'\sl text'</code>
<code>\rm</code>	Normal font	<code>'\rm text'</code>
<code>\fontname{specifier}</code>	Font name — Set specifier as the name of a font family. You can use this in combination with other modifiers.	<code>'\fontname{Courier} text'</code>
<code>\fontsize{specifier}</code>	Font size — Set specifier as a numeric scalar value in point units to change the font size.	<code>'\fontsize{15} text'</code>

## LaTeX Markup

To use LaTeX markup, set the Interpreter property to 'latex'. Use dollar symbols around the text, for example, use `' $\int_1^{20} x^2 dx$ '` for inline mode or `'
$$\int_1^{20} x^2 dx$$
'` for display mode.

For more information about the LaTeX system, see The LaTeX Project website at <http://www.latex-project.org/>.



# Математические формулы:

title('\$\$\sqrt{x+y}\$\$', 'Interpreter', 'latex', 'FontSize', 14)

Math	LaTeX	Example	Output
Fraction	<code>\frac{}</code>	$P(A)=\frac{n(A)}{n(U)}$	$P(A) = \frac{n(A)}{n(U)}$
Square root	<code>\sqrt{}</code>	$\sqrt{x+y}$	$\sqrt{x+y}$
Superscript	<code>u^n</code>	$u_n=u_1r^{n-1}$	$u_n = u_1r^{n-1}$
Subscript	<code>u_n</code>	$u_n=u_1+(n-1)d$	$u_n = u_1 + (n - 1)d$
Greek letters	<code>\alpha</code>	<code>\alpha \beta \pi \Gamma</code>	$\alpha\beta\pi\Gamma$
Text	<code>\text{}</code>	$A=\pi r^2\text{ , where r is the radius}$	$A = \pi r^2$ ,where r is the radius
Sigma	<code>\sum</code>	$n=\sum_{i=1}^{\infty} f_i$	$n = \sum_{i=1}^{\infty} f_i$
Limit	<code>\lim</code>	$\lim_{x \rightarrow \infty}$	$\lim_{x \rightarrow \infty}$
Integration	<code>\int</code>	$\int_a^b x^2 dx$	$\int_a^b x^2 dx$
Trig	<code>\sin</code>	$\sin()$	$\sin(x+y)$
Comparison	<code>\leq</code>	<code>\leq \geq \approx \neq</code>	$\leq \geq \approx \neq$
Overline	<code>\overline{}</code>	$\overline{x}$	$\bar{x}$
Tilde ~	<code>\sim</code>	$X \sim N(\mu, \sigma^2)$	$X \sim N(\mu, \sigma^2)$
Number sets	<code>\mathbb{}</code>	$\mathbb{I, R, Q, N, Z}$	$\mathbb{I, R, Q, N, Z}$

# Построение замкнутой области

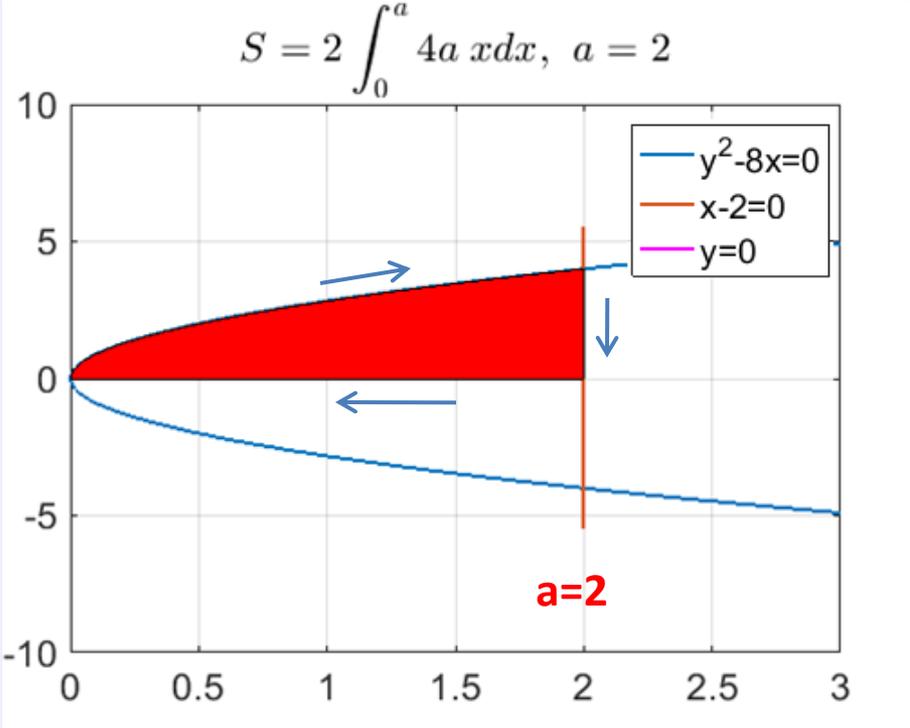


## заклученной между кривыми:

```

syms x y; a=2; f=@(x,y) (y.^2-4*a.*x); p=@(x,y)x-2
fimplicit(f,[0 a+1 -10 10]), hold on,
r=fimplicit(p, [1.9 2.1 -5.5 5.5]), step=0.01; grid on
plot(0:step:2,zeros(size(0:step:2)),'m-')
set(get(gca,'children'),'linewidth',1.5)
legend('y^2=8x','x=2','y=0')% Tex
title('$S=2\int_0^a 4a x dx, \ a=2 $',...
'interpreter','latex')

```



## **Contour for operation fill:**

```

step=0.01;
x1=0:step:2; y1=(4*a.*x1).^(1/2)
y2=4:-step:0; x2=2*ones(size(y2))
x3=2:-step:0; y3=zeros(size(x3))
X=[x1'; x2'; x3']; Y=[y1'; y2'; y3']
t=fill(X,Y,'r'),
xlim(gca,[0,3]),ylim(gca,[-10,10])

```

**xline(x0)** - конструктор прямой, параллельной OY

**yline(y0)** - конструктор прямой, параллельной OX



# Пример эволюции синтаксиса! Recall!

it's interesting to compare I and II!

1. clear, n=10; `vec: 1x10 double =`  
0 2 3 0 0 0 3 0 1 0

2. `vec=round(rand(1,n)).*randi(6,[1,n]); V=vec;`

I): `LogicalVectorfind: 1x10 logical =`  
0 0 1 0 0 0 1 0 0 0

3. `LogicalVectorfind=V>2; V(LogicalVectorfind)=10`

II): `V: 1x10 double =`  
0 2 10 0 0 0 10 0 1 0

4. `ind=find(vec>2) % ind - positions in the vector`

5. `vec(ind)=10`  
`ind: 1x2 double =`  
3 7

6. **I & II - the same!**

```
vec: 1x10 double =  
0 2 10 0 0 0 10 0 1 0
```



[nvkurbatova@sfedu.ru](mailto:nvkurbatova@sfedu.ru)

# Этапы создания приложений, в т.ч. GUI

# Файловое взаимодействие приложения

The screenshot shows the MATLAB Package App interface. The top ribbon includes tabs for PLOTS, APPS, EDITOR, PUBLISH, and VIEW. The APPS tab is active, showing various tool icons like Curve Fitting, Optimization, PID Tuner, System Identification, Signal Analyzer, Image Acquisition, Instrument Control, SimBiology, MATLAB Coder, Application Compiler, Classification Learner, and Distribution Fitting. The main window title is "Package App" and the address bar shows the file path: "F:\Nata\2017\Matlab 2017\2023\Лекции Work Folder\MyPresentation\Lecture 8\ML\ExeMy\myexe.prj".

The interface is divided into three main sections:

- Pick main file:** Shows the main file "myexe.m" with a "Remove main file" button.
- Describe your app:** Contains fields for app name ("myexeproba", version "1.0"), "NVK" (circled in red), "Email", "Company", and "Summary". A "Set as default contact" button is also present.
- Package into installation file:** Shows the "Output folder" as "F:\Nata\2017\Matlab 2017\2023\Л" and a "Package" button (circled in red).

On the left side, there are sections for "Files included through analysis" (with a "Rerun analysis" button) and "Shared resources and helper files" (with an "Add files/folders" button).

At the bottom, a "Products" dropdown menu is open, showing "MATLAB" selected (circled in red) and other options like Simulink, Aerospace Blockset, etc. An "Apply" button is at the bottom right of the dropdown.

# Приложение. Шаг 1



**APPS**

1

MATLAB Coder

Select Review Define Check Generate Finish

Open

Имя файла: myexe.m

3

Открыть

Отмена

Generate code for function: Enter a function name

2

The MATLAB Coder workflow generates standalone C and C++ code from MATLAB code. **To begin, select your entry-point function(s).**



# Приложение. Шаг 2

nvkurbatova@sfedu.ru

MATLAB Coder - myexe.prj

Select Define Check Generate Finish

Выбрать **Next**, чтобы перейти на этап **Define!**

## MATLAB Coder

Numeric Conversion **None** (4 (согласно логике))

the same

Entry-Point Functions:

myexe

+ Add Entry-Point Function (5 для многомодульного)

Project location: \\Лекции Work Folder\MyPresentation\Lecture 8\ML\ExeMy\myexe.prj

Next

# Приложение. Шаг 3

nvkurbatova@sfedu.ru



MATLAB Coder - myexe.prj

## Define Input Types

To convert MATLAB to C, you must define the type of each input for every entry point function.  
[Learn more](#)

To **automatically define input types**, call myexe or enter a script that calls myexe in the MATLAB prompt below:

```
>> myexe
```

Autodefine Input Types

**Определение типов входных параметров:**

myexe.m	step	double(1 x 1)
---------	------	---------------

Add global

Продлжить

Back Next



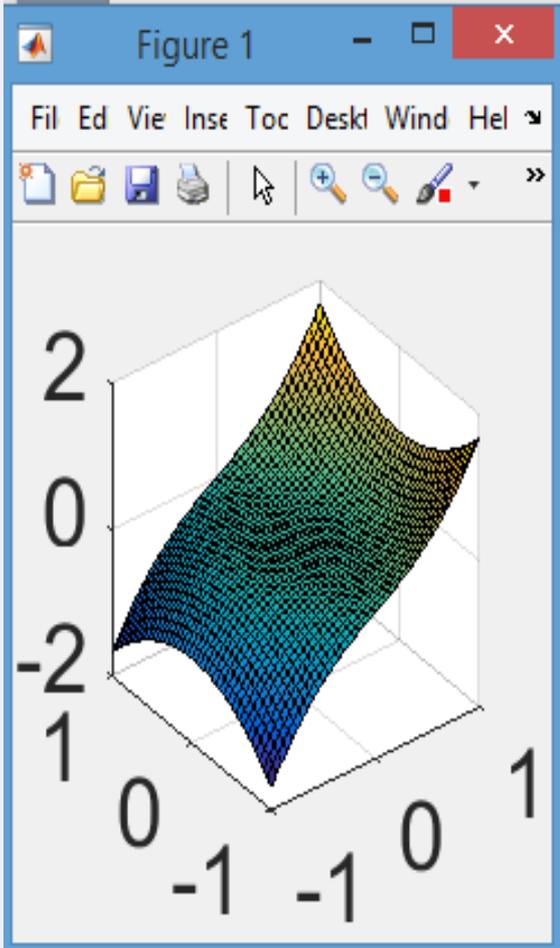
# Тестирование и запуск

nvkurbatova@sfedu.ru



## Check for Run-Time Issues

SETTINGS CHECK



This step creates a MEX function from your MATLAB function(s), invokes the MEX function, and reports issues that may be hard to diagnose in the generated C code.

[Learn more](#)

Enter code or select a script that exercises **myexe**:

```
>> myexe (0.05)
```

Collect MATLAB line execution counts

Check for Issues

✔ No issues detected. [View MATLAB line execution counts](#)



Generating trial code



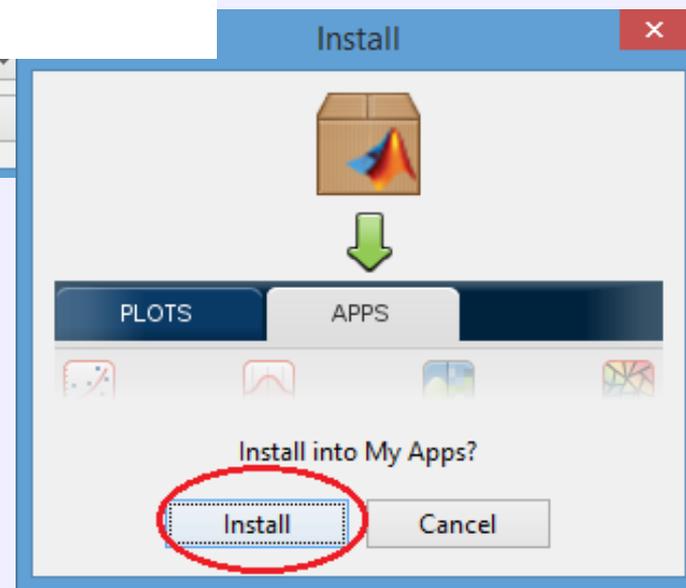
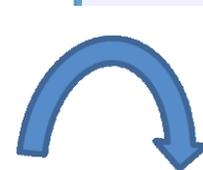
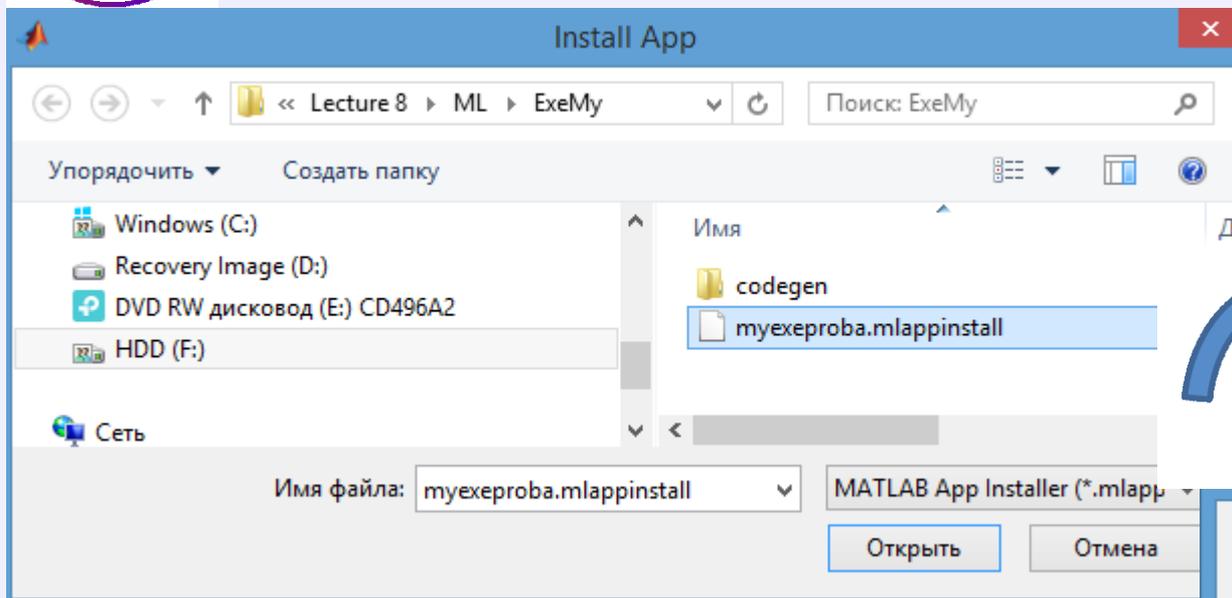
Building MEX



Running test file with MEX



# Приложение пользователя на панели





Интересно, самостоятельно,  
поощряется баллами (>10)

- **MatLab Report Generator**
- **Stand alone \*.exe: Application Compiler**



**Спасибо за внимание!**

**“Разве ты не заметил, что способный к математике  
изоцрен во всех науках в природе?!”**

**Платон**

**(423 – 347гг) до н. э.**



**Золотых Н.Ю.**

**Краткая сводка по языку MatLab**

**ссылка на \*.pdf:**

<chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/http://www.uic.unn.ru:8103/~zny/matlab/Book/matlabref.pdf>



# Interruptible on

Кроме функции `pause` есть и другие функций MATLAB, наличие которых в подфункции обработки одного события (свойство `Interruptible` которого установлено в `'on'`) приведет к прерыванию ее работы при возникновении другого события. Вот список этих функций:

- `drawnow` - обновление графического окна;
- `figure` - создание графического окна;
- `getframe` - создание массива с кадрами (для получения анимации из меняющегося содержимого осей);
- `waitfor` - приостановка выполнения команд до тех пор, пока некоторый объект не будет удален или заданное его свойство не изменится.