First of all, how to use programs: unpack everything in one folder. Open all the files simultaneously (because some of them are used functions, which are created in the separate files). Each file is named like number of task:

task_1 - forward kinematics

task_1_syms - forward kinematics in symbolic way

task_2 - inverse kinematics

task_3_classical - Jacobians by classical approach

task_3_cross_products - Jacobians by cross-products approach

task_5 - velocity of tool frame

task_6_inverse_differential_syms – joint trajectories deferential approach.

task_6_inverse_kinematics - joint trajectories by inverse kinematics

I hope, everything will be executable on your PC…

Description of each task

Task 1: solved the same way as previous homework. Axes chosen as: z – along translation or around which rotation is. Frame of third link is moved to second joint in case to coincide X and Z.

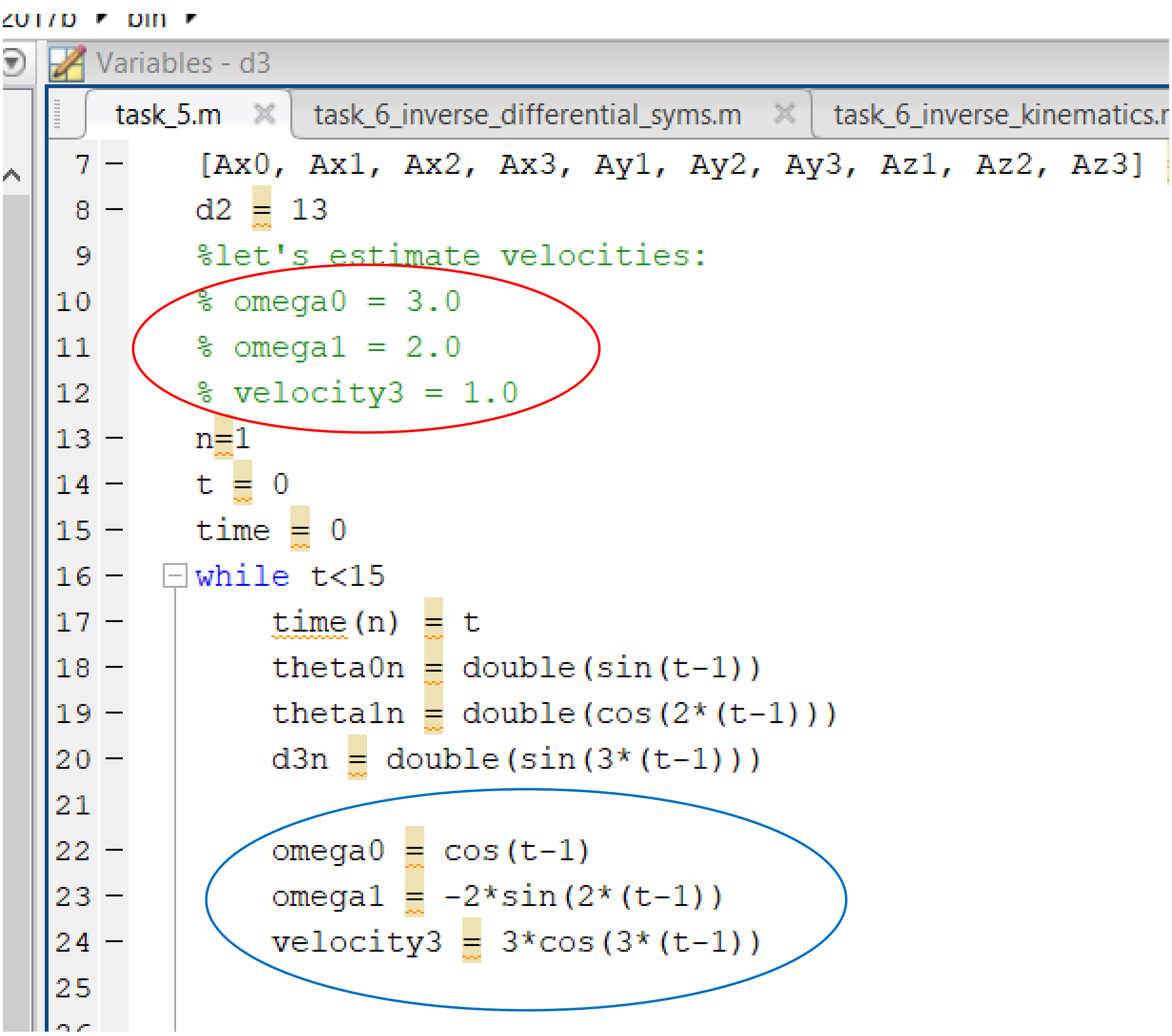
Task 2: solved the same way as previous homework (by geometrical approach).

Task 3: “task_3_classical”: from “task_1_syms” were taken x, y, z coordinates. In “task_1_syms” they are created from total transformation matrix – from 0-frame to end-effector. Yaw, pitch, roll coordinates had to be chosen from rotational matrixes as multiplication of rotation matrix and [0 0 1]’. But I just copied numbers.

“task_3_cross_product”: solved only in NUMERICAL way. From “task_1” are taken coordinates (x, y, z, r, p, y), multiplied by [0 0 1]’ in case to work with separate joint and cross-producted with translation from link (n-1) to link 0.

Task 4: the singularity is when angle of second joint is equal to 90 degrees. So, we can not find angle of the first joint (the end-effector is pointed straight up). Sorry, but I didn’t understood, what should I do in matlab.

Task 5: how I counted the velocity: took Jacobians and multiplied of the first derivatives of functions of each joint-coordinates. If the speed should be constant, just comment blue lines and discomment red:



Task 6: ”task_6_inverse_defferential_syms”: each trajectory was found only in symbolic way just because I need to do next deadline, sorry. Each trajectory was found as inverse Jacobian multiplied by velocity of each joint and integrated. Script is running for about 1.5 minutes with a lot of warnings, but don’t worry.

“task_6_inverse_kinematics”: implemented on the base of inverse kinematics. I just put into coordinates of EF equations from task-sheet and that’s all.

Below you can see results of executing programs.

Task 1

Primary data which I accept by myself (in blue) and result of executing script “task_1” (in red)

Variables - d3

task_1.m | task_1_syms.m | task_2.m | task_3_classical.m | task_3_cross_products.m

```
clear all
clc
%syms theta0 theta1 d1 d2 d3
theta0 = deg2rad(42), theta1 = deg2rad(24), d1 = 8, d2 = 13, d3 = 5

R0  = [cos(theta0) -sin(theta0) 0
       sin(theta0)  cos(theta0) 0
           0            0       1]

Rt0 = [1    0    0
       0    0   -1
       0    1    0]

```

Command Window

New to MATLAB? See resources for Getting Started.

```
py =

   11.0031


pz =

   15.3213


ans =

    0.7330
```

Task 2

Results of executing “task_2.m file”

```
%syms px py pz d1 a2
clear all
clc
[theta0, theta1, px, py, pz, d3, d2, d1,x0, y0, z0, A0,x1, y1, z1, A1, A2] = forward()
d3 = sqrt((pz-d1)^2+px^2)-d2
theta1 = rad2deg(atan2((pz-d1),sqrt(py^2+px^2)))
theta0 = rad2deg(atan2(py, px))
```

Command Window

New to MATLAB? See resources for Getting Started.

```
d3 =

    5.0000


theta1 =

   24.0000


theta0 =

    42

>>
```

Coincides with primary information from forward kinematics:

```
clear all
clc
%syms theta0 theta1 d1 d2 d3
theta0 = deg2rad(42), theta1 = deg2rad(24), d1 = 8, d2 = 13, d3 = 5

R0  = [cos(theta0) -sin(theta0) 0
       sin(theta0)  cos(theta0) 0
            0            0      1]

Rt0 = [1    0    0
       0    0   -1
       0    1    0]
```

Task 3

Classical approach (the results are in the command line blue – numerical, red – symbolic):

Variables - d3 | Editor - task_3_classical.m

task_1.m | task_1_syms.m | task_2.m | task_3_classical.m | task_3_cross_products.m | task_5.m | task_6_inverse_differential_

```
px1 = double(subs(px1s,{theta1,theta0,d2,d3},{deg2rad(24),deg2rad(42),13,5}))
px2s = diff(px, theta1, 1)
px2 = double(subs(px2s,{theta0,theta1,d2,d3},{deg2rad(42),deg2rad(24),13,5}))
px3s = diff(px, d3, 1)
px3 = double(subs(px3s,{theta0,theta1},{deg2rad(42),deg2rad(24)}))
```

Command Window

New to MATLAB? See resources for Getting Started.

```
J =

  -11.0031   -5.4408    0.6789
   12.2201   -4.8989    0.6113
         0   16.4438    0.4067
         0         0         0
         0         0         0
    1.0000    1.0000         0


ans =

[ -cos(theta1)*sin(theta0)*(d2 + d3), -cos(theta0)*sin(theta1)*(d2 + d3), cos(theta0)*cos(theta1)]
[  cos(theta0)*cos(theta1)*(d2 + d3), -sin(theta0)*sin(theta1)*(d2 + d3), cos(theta1)*sin(theta0)]
[                                  0,             cos(theta1)*(d2 + d3),             sin(theta1)]
[                                  0,                                  0,                       0]
[                                  0,                                  0,                       0]
[                                  1,                                  1,                       0]
```

Geometrical approach:

Variables - d3

task_1.m | task_1_syms.m | task_2.m | task_3_classical.m | task_3_cross_products.m

```
[theta0, theta1, px, py, pz, d3, d2, d1,x0, y0, z0, A0,x1, y1, z1,

ZC = [x0 y0 z0] %zero-link coordinates
OC = [x1 y1 z1] %one-link coorinates
EC = [px py pz] %end-effector coordinates

RomegaX = 0 %rotation of revolute joint around X
RomegaY = 0 %rotation of revolute joint around Y
RomegaZ = 1 %rotation of revolute joint around Z
RO = [RomegaX RomegaY RomegaZ] %Rotational-matrix for revolute joi

TomegaX = 0 %rotation of translational joint around X
TomegaY = 0 %rotation of translational joint around y
TomegaZ = 0 %rotation of translational joint around Z
TO = [TomegaX TomegaY TomegaZ] %Rotational-matrix for prosmatic jo
```

Command Window

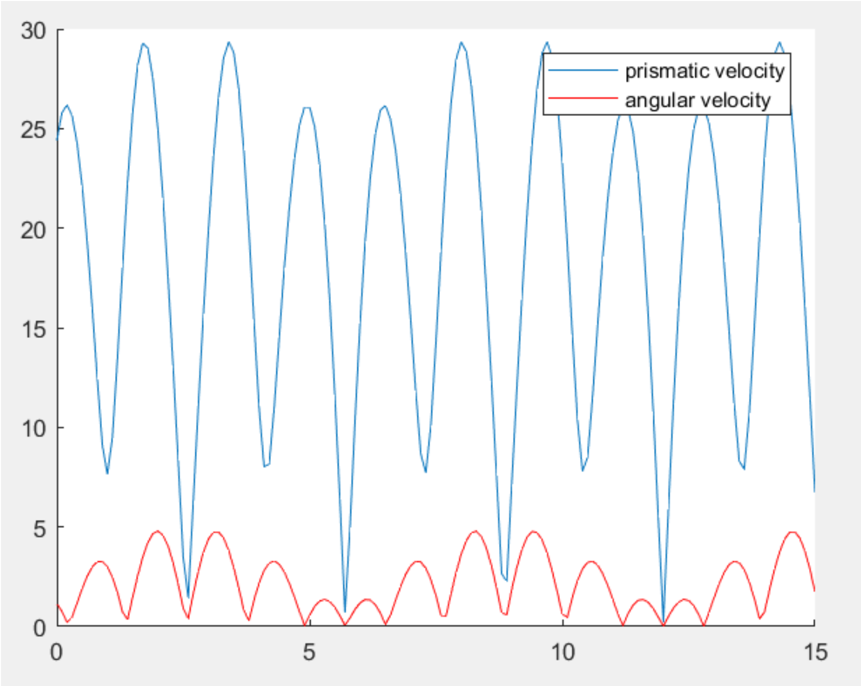
New to MATLAB? See resources for Getting Started.

```
J =

  -11.0031   -5.4408    0.6789
   12.2201   -4.8989    0.6113
         0   16.4438    0.4067
         0         0         0
         0         0         0
    1.0000    1.0000         0
```

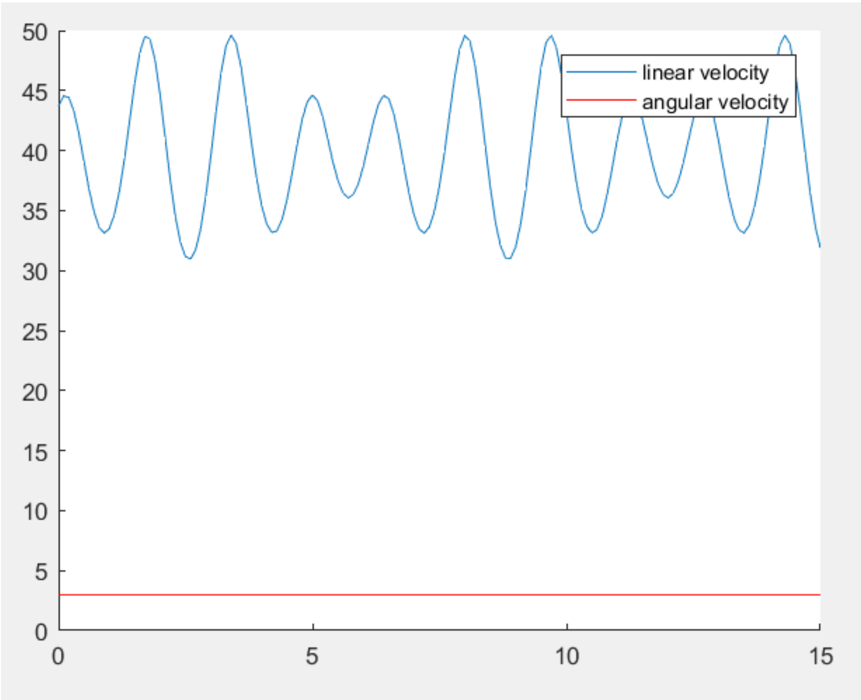
The results are coincide, so, everything is correct.

Task 5

Resultative graph with changing speed:



And with permanent speed:

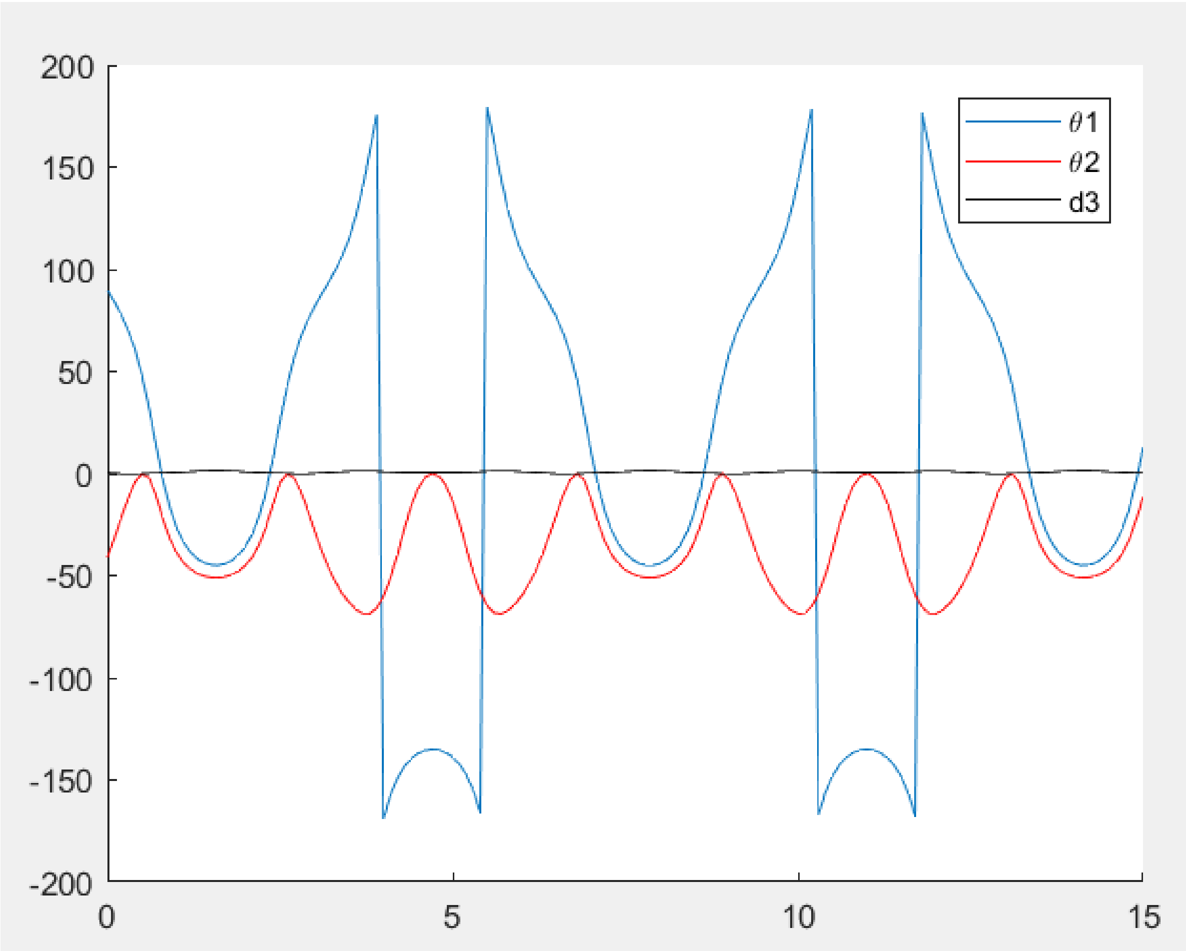


Task 6

Inverse differential approach (as far as was solved by symbolic way, no proper output in following way):

```
Command Window
New to MATLAB? See resources for Getting Started.

Warning: Unable to display symbolic object because 'symengine' was reset. Repeat commands to regenerate result.
> In sym/disp (line 57)
  In sym/display>displayVariable (line 85)
  In sym/display (line 47)
  In task_6_inverse_differential_syms (line 56)

theta1_sys =

Warning: Unable to display symbolic object because 'symengine' was reset. Repeat commands to regenerate result.
> In sym/disp (line 57)
  In sym/display>displayVariable (line 85)
  In sym/display (line 47)
  In task_6_inverse_differential_syms (line 57)

theta0_sys =

Warning: Unable to display symbolic object because 'symengine' was reset. Repeat commands to regenerate result.
> In sym/disp (line 57)
  In sym/display>displayVariable (line 85)
  In sym/display (line 47)
  In task_6_inverse_differential_syms (line 58)

t =

    2.0000
```

Inverse approach:



Very bad mode for the first-joint: reverse everywhere