SLCO4A - Aula Prática 1

Prof. Willian R. Bispo M. Nunes

Sumário

Introdução ao MATLAB

Operações aritméticas simples

```
clear all
clc
close all
%Operações básicas
2+5
```

```
ans = 7
```

```
5-1
```

```
ans = 4
```

```
12-2
```

```
ans = 10
```

```
2\4
```

```
ans = 2
```

```
2/4
```

```
ans = 0.5000
```

```
20*23
```

```
ans = 460
```

```
exp(2)
```

```
ans = 7.3891
```

```
10e1+1e1
```

```
ans = 110
```

```
%Operações com números decimais

%Números na base de potência 10
```

Comentários

```
% comentários
```

Constantes

```
pi
```

```
ans = 3.1416
```

```
% representação alternativa para números imaginários
inf
```

```
ans = Inf
```

```
-inf
```

```
ans = -Inf
```

```
NaN % número não identificado (not a number)
```

```
ans = NaN
```

```
realmax
```

```
ans = 1.7977e+308
```

```
realmin
```

```
ans = 2.2251e-308
```

Built-In Functions

```
%raiz quadrada
sqrt(2)
```

```
ans = 1.4142
```

```
%valor absoluto
abs(-10)
```

```
ans = 10
```

```
sqrt(abs(2^2+4^2))
```

```
ans = 4.4721
```

```
%função sinal
t=-10:1:10
```

```
t = 1×21
   -10    -9    -8    -7    -6    -5    -4    -3    -2    -1     0     1     2⋯
```

```
sign(20)
```

```
ans = 1
```

```


%exponencial
exp(10)
```

```
ans = 2.2026e+04
```

```
%logaritmo natural
log(exp(1))
```

```
ans = 1
```

```
log(10)
```

```
ans = 2.3026
```

```
%logaritmo na base 10
log10(1000)
```

```
ans = 3
```

```
%logaritmo na base 2
log2(2^8)
```

```
ans = 8
```

```
%seno em radianos
sin(pi)
```

```
ans = 1.2246e-16
```

```
%cosseno em radianos
cos(pi)
```

```
ans = -1
```

```
%tangente em radianos
tan(pi)
```

```
ans = -1.2246e-16
```

```
%seno em graus
sind(45)
```

```
ans = 0.7071
```

```
%cosseno em graus
```

```
cosd(45)
```

```
ans = 0.7071
```

```
%tangente em graus
tand(45)
```

```
ans = 1
```

```
%arco seno sen^-1 em rad
asin(1)
```

```
ans = 1.5708
```

```
%arco cosseno cos^-1 em rad
acos(1)
```

```
ans = 0
```

```
%arco cosseno em graus
acosd(sqrt(2)/2)
```

```
ans = 45
```

```
%arco tangente tan^-1 em rad
ceil (0.4) %arrendondamento para o maior inteiro mais próximo
```

```
ans = 1
```

```
floor(0.4) %arrendondamento para o menor inteiro mais próximo
```

```
ans = 0
```

```
round(1.6) % arrendondamento para o inteiro mais próximo
```

```
ans = 2
```

```
round(1.4)
```

```
ans = 1
```

```
factorial(4) %fatorial
```

```
ans = 24
```

```
factorial(10)
```

```
ans = 3628800
```

Váriaveis, vetores e matrizes

```
%Vetor linha
v=[1 2 3 4]
```

```
v = 1×4
     1     2     3     4
```

```


%vetor coluna
```

```
w=[1; 2; 3; 4]
```

```
w = 4×1
     1
     2
     3
     4
```

```
%tamanho do vetor ou matriz
A=[1 2 3; 4 5 6; 7 8 9]
```

```
A = 3×3
     1     2     3
     4     5     6
     7     8     9
```

```
size(A)
```

```
ans = 1×2
     3     3
```

```
%acesso a um elemento indexado
w(2)+v(1)
```

```
ans = 3
```

```
A(1:2, 1:3)
```

```
ans = 2×3
     1     2     3
     4     5     6
```

```
%Vetor com elementos espaçamento definidos inicio:passo:ultimo elemento
t=0:1e-3:10
```

```
t = 1×10001
         0    0.0010    0.0020    0.0030    0.0040    0.0050    0.0060    0.0070 ⋯
```

```
t=linspace(0,1,1000)
```

```
t = 1×1000
         0    0.0010    0.0020    0.0030    0.0040    0.0050    0.0060    0.0070 ⋯
```

```
%Matriz


% Operações
x= v+w'
```

```
x = 1×4
     2     4     6     8
```

```
%soma
v+w
```

```
ans = 4×4
     2     3     4     5
     3     4     5     6
     4     5     6     7
     5     6     7     8
```

```
%subtração

%operação entre elementos indexados
s=2*v
```

```
s = 1×4
     2     4     6     8
```

```
s=2*v(1,2)
```

```
s = 4
```

```
%multiplicação
%a*b
3*eye(3)*A
```

```
ans = 3×3
     3     6     9
    12    15    18
    21    24    27
```

```
B=3.*A
```

```
B = 3×3
     3     6     9
    12    15    18
    21    24    27
```

```



%divisão
A./3
```

```
ans = 3×3
    0.3333    0.6667    1.0000
    1.3333    1.6667    2.0000
    2.3333    2.6667    3.0000
```

```
B./3
```

```
ans = 3×3
     1     2     3
     4     5     6
     7     8     9
```

```

%potencição

%Matriz inversa

```

Formatos

```
format %default format com 4 digitos decimais
pi
```

```
ans = 3.1416
```

```
format long
pi
```

```
ans =
   3.141592653589793
```

```
format short e %base 10
pi
```

```
ans =
   3.1416e+00
```

```
format long e
pi
```

```
ans =
     3.141592653589793e+00
```

```
format bank %2 digitos decimais
pi
```

```
ans =
          3.14
```

```
format
pi
```

```
ans = 3.1416
```

Gerenciamento de memória

```
who %mostra as variáveis presentes no workspace
```

```
Your variables are:

A    B    ans  s    t    v    w    x
```

```
whos %mostra as variáveis, o seu tipo e tamanho
```

```
  Name      Size             Bytes  Class     Attributes

  A         3x3                 72  double
  B         3x3                 72  double
  ans       1x1                  8  double
  s         1x1                  8  double
  t         1x1000            8000  double
  v         1x4                 32  double
  w         4x1                 32  double
  x         1x4                 32  double
```

Comandos para salvar, limpar e carregar variáveis do workspace

```
save ap1 %save um arquivo .mat no diretório atual
clear all %limpa workspace
load ap1 %carrega variáveis no workspace
```

Help MATLAB

```
help cos
```

```
  cos    Cosine of argument in radians.
     cos(X) is the cosine of the elements of X.

     See also acos, cosd, cospi.

     Documentation for cos
     Other functions named cos
```

```
lookfor cos
```

```
acos                           - Inverse cosine, result in radians.
acosd                          - Inverse cosine, result in degrees.
acosh                          - Inverse hyperbolic cosine.
acsc                           - Inverse cosecant, result in radian.
acscd                          - Inverse cosecant, result in degrees.
acsch                          - Inverse hyperbolic cosecant.
cos                            - Cosine of argument in radians.
cosd                           - Cosine of argument in degrees.
cosh                           - Hyperbolic cosine.
cospi                          - Compute cos(X*pi) accurately.
csc                            - Cosecant of argument in radians.
cscd                           - Cosecant of argument in degrees.
csch                           - Hyperbolic cosecant.
createSetCall                  - Cosntruct a codegen.codefunction to call set.
slsincos                       - This is a private mask helper file for sine and cosine blocks in
slsincoslut                    - This is a private mask helper file for sine and cosine blocks in
ee_battery_lse                 - Cost function used by ee_battery_opt_m.m
ee_solar_lse                   - Cost function used by ee_solar_opt_m.m
hdlsimmatlabsysobj             - Load instantiated HDL design for cosimulation with MATLAB
hdlsimulink                    - Load instantiated HDL design for cosimulation with Simulink
acos                           - Inverse cosine of gpuArray, result in radians
acosd                          - Inverse cosine of gpuArray, result in degrees
acosh                          - Inverse hyperbolic cosine of gpuArray
acsc                           - Inverse cosecant of gpuArray, result in radian
acscd                          - Inverse cosecant of gpuArray, result in degrees
acsch                          - Inverse hyperbolic cosecant of gpuArray
cos                            - Cosine of gpuArray in radians
cosd                           - Cosine of gpuArray in degrees
cosh                           - Hyperbolic cosine of gpuArray
cospi                          - Compute cos(X*pi) accurately.
csc                            - Cosecant of gpuArray in radians
cscd                           - Cosecant of gpuArray in degrees
csch                           - Hyperbolic cosecant of gpuArray
hedgeopt                       - Allocate an optimal hedge for a set of target costs or sensitivities.
vsimmatlabsysobj               - Load instantiated HDL model for cosimulation with MATLAB
vsimulink                      - Load instantiated HDL model for cosimulation with Simulink
krg                            - Kissell Research Group transaction cost analysis object.
costCurves                     - Market impact cost of order execution.
portfolioCostCurves            - Market impact cost of order execution.
timingRisk                     - Market impact cost estimate uncertainty.
KRGCostIndexExample            - KRG Cost Index Example.
calcjjdjj                      - Calculate the cost function JJ and the change in the cost function dJJ
AbstractPID                    - parent class for all PID MCOS objects
getF                           - returns the merit function value as a weighted sum of the following cost
minDeTrend                     - Finds minimum cost and its index form a table of values where
mpc_costcomputation            - Compute cumulative cost J as a function of I/O
```

```
validateFcns                   - tests prediction model and custom cost/constraint functions
validateFcns                   - tests prediction model and custom cost/constraint functions
mpcCustomCostFcnWrapper        - Wrapper to custom cost function to optimize by nonlinear programming
mpcCustomOptimization          - Gateway function to custom (nonlinear) cost function and constraints
zmsnlmpc_getCostJacobian       - Utility function to compute Jacobians for cost function using numerical
zmsnlmpc_objfunXMVDMVE         - Wrapper function for the cost function used by "fmincon" solver.
zmsnlmpc_objfunXMVE            - Wrapper function for the cost function used by "fmincon" solver.
znlmpc_computeJacobianCost     - Compute Jacobians for custom cost by perturbation.
znlmpc_objfun                  - Wrapper function for nonlinear MPC cost used by NLP solver.
fi_cordic_sincos_demo          - Compute Sine and Cosine Using CORDIC Rotation Kernel
fi_sin_cos_demo                - Calculate Fixed-Point Sine and Cosine
rotmat                         - Convert quaternion to a rotation matrix or direction cosine matrix
cosets                         - Produce cyclotomic cosets for a Galois field.
gfcosets                       - Produce cyclotomic cosets for a Galois field.
commblkagctransform            - Raised cosine filter blocks transform function
commblkrcfiltrx                - Raised cosine Receiver FIR filter blocks helper function.
commblkrcfilttransform         - Raised cosine filter blocks transform function
commblkrcfilttx                - Raised cosine Transmit filter blocks helper function.
commFixBrokenLinksRCos         - (unlnkBlk, newBlk)
rcfiltgaincompat               - Backwards compatible raised cosine filter gain
tocgaloisfield                 - GF math, filtering, transforms, cosets
rcosfir                        - Design a raised cosine FIR filter.
rcosflt                        - Filter the input signal using a raised cosine filter.
rcosiir                        - Design a raised cosine IIR filter.
rcosine                        - Design raised cosine filter.
TimeBuffer                     - SCOPESUTIL.TIMEBUFFER - An MCOS buffer object
cordicacos                     - CORDIC based approximation for the inverse cosine
cordiccos                      - CORDIC-based approximation of COS.
cordicsincos                   - CORDIC-based approximation of SIN and COS.
acos                           - Fixed-Point overload for acos
cos                            - Cosine of argument in radians.
EndogenousGlucoseProduction    - Helper function for insulindemo SimBiology demo
GlucoseAppearanceRate          - Helper function for insulindemo SimBiology demo
getSimDataForLineImpl          - Implementation of getSimDataForLine called from C++ through the COS
getSimDataInTimeRangeImpl      - Implementation of getSimDataInTimeRange called from C++ through the COS
useMCOSExtMgr                  - Returns true.
chirp                          - Swept-frequency cosine generator.
thiscost                       -
thiscost                       -
thiscost                       -
evalcost                       -
xcopt                          - Cost function for N-point complex frequency transformation.
dct2                           - 2-D discrete cosine transform.
dctmtx                         - Discrete cosine transform matrix.
idct2                          - 2-D inverse discrete cosine transform.
dct                            - Discrete cosine transform.
idct                           - Inverse discrete cosine transform.
str2customreg                  - converts a custom regressor from string form to MCOS form.
minimize                       - Runs the optimization algorithm to minimize the cost and
minimize                       - Runs the optimization algorithm to minimize the cost and
angle2dcm                      - Create direction cosine matrix from rotation angles.
atmoscira                      - Use COSPAR International Reference Atmosphere 1986.
dcm2alphabeta                  - Convert direction cosine matrix to angle of attack and sideslip angle.
dcm2angle                      - Create rotation angles from direction cosine matrix.
dcm2latlon                     - Convert direction cosine matrix to geodetic latitude and longitude.
dcm2quat                       - Convert direction cosine matrix to quaternion.
dcm2rod                        - Convert direction cosine matrix to Rodrigues vector.
dcmbody2wind                   - Convert angle of attack and sideslip angle to direction cosine matrix.
dcmecef2ned                    - Convert geodetic latitude and longitude to direction cosine matrix.
quat2dcm                       - Convert quaternion to direction cosine matrix.
rod2dcm                        - Convert Rodrigues vector to direction cosine matrix.
acos                           - Inverse cosine, result in radians.
acosh                          - Inverse hyperbolic cosine.
acsc                           - Inverse cosecant, result in radians.
```

```
cos                          - Cosine of argument in radians.
cosh                         - Hyperbolic cosine.
csc                          - Cosecant of argument in radians.
cos_tr                       - function out = cos_tr(freq,mag,tinc,lastt)
fmlp                         - [basic,sol,cost,lambda,tnpiv] = fmlp(a,b,c,startbasic,tnpivmax)
linp                         - function [basic,sol,cost,lambda,tnpiv,flopcnt] = linp(a,b,c,startbasic)
mflinp                       - function [basic,sol,cost,lambda,tnpiv,flopcnt] = mflinp(a,b,c,startbasic)
mflp                         - [basic,sol,cost,lambda,tnpiv] = mflp(a,b,c,startbasic,tnpivmax)
validatorVehicleCostmap      - State validator based on 2-D costmap
aztilt2nedv                  - Azimuth and tilt to NED direction cosines
setCosts                     - Set up proportional transaction costs.
setCosts                     - Set up proportional transaction costs.
setCosts                     - Set up proportional transaction costs.
sdorectifier_cost            -
ddccostcomp                  - Utility to display the implementation cost of DDCs
bscost                       - Band Stop Cost function for order minimization w.r.t passband edge.
dct                          - Discrete cosine transform.
firrcos                      - Raised Cosine FIR Filter design.
idct                         - Inverse discrete cosine transform.
rcosdesign                   - Raised cosine FIR filter design
cost                         -
evalcost                     -
thiscost                     -
evalcost                     -
thiscost                     -
evalcost                     -
evalcost                     -
evalcost                     -
cost                         - fdesign.cost class
rcosine                      - Construct a raised cosine pulse shaping filter designer.
sqrtrcosine                  - Construct a square root raised cosine pulse shaping filter designer.
abstractpsrcosmin            - ABSTRACTPRCOSMIN   Construct an ABSTRACTPSRCOSMIN object.
abstractpsrcosnsym           - Construct an ABSTRACTPSRCOSNSYM object.
abstractpsrcosord            - ABSTRACTSPRCOSORD   Construct an ABSTRACTPSRCOSORD object.
psrcosmin                    - Construct an PSRCOSMIN object.
psrcosnsym                   - Construct an PSRCOSNSYM object.
psrcosord                    - Construct an PSRCOSORD object.
pssqrtrcosmin                - Construct an PSSQRTRCOSMIN object.
pssqrtrcosnsym               - Construct an PSSQRTRCOSNSYM object.
pssqrtrcosord                - Construct an PSSQRTRCOSORD object.
gencoswin                    - Returns one of the generalized cosine windows.
costheta                     - Compute cosine of angles of stable poles.
abstractrcosfir              - Abstract constructor produces an error.
abstractrcosmin              - Abstract constructor produces an error.
rcosmindesign                - Design the filter
abstractrcoswin              - Abstract constructor produces an error.
rcoswindesign                - Design a raised cosine filter
gausswin                     - WINDOWRCOS Construct a gausswin object
rcosmin                      - Construct a RCOSMIN object
rcoswin                      - Construct a RCOSWIN object
sqrtrcosmin                  - Construct a SQRTRCOSMIN object
sqrtrcoswin                  - Construct a SQRTRCOSWIN object
copyCosimDemoFiles           - COPYFILDEMOFILES(DEMONAME) copies source HDL files of HDL cosimulation
cosimBlock_TabChangedCB      - : callback when dialog tab changes.  This func must be
cosimWizard                  - Cosimulation Wizard.
hdlcosim                     - is a shorthand interface for hdlverifier.HDLCosimulation
hdlcosim_template            - is a cosimWizard generated function used for HDL
hdllink_block_init           - Mask helper function for Simulink HDL Link Cosim
cosim_fil_ml_wfa             - Cosimulation and FPGA-in-the-Loop in MATLAB-to-HDL Workflow
imdct                        - Inverse Modified Discrete Cosine Transform
mdct                         - Modified Discrete Cosine Transform
acos                         - Inverse cosine of codistributed array, result in radians
acosd                        - Inverse cosine of codistributed array, result in degrees
acosh                        - Inverse hyperbolic cosine of codistributed array
```

```
acsc                           - Inverse cosecant of codistributed array, result in radian
acscd                          - Inverse cosecant of codistributed array, result in degrees
acsch                          - Inverse hyperbolic cosecant of codistributed array
cos                            - Cosine of codistributed array in radians
cosd                           - Cosine of codistributed array in degrees
cosh                           - Hyperbolic cosine of codistributed array
cospi                          - Compute cos(X*pi) accurately.
csc                            - Cosecant of codistributed array in radians
cscd                           - Cosecant of codistributed array in degrees
csch                           - Hyperbolic cosecant of codistributed array
enginetradeoff_cost            - Compute controller cost based on sensor accuracy, actuator response, and
enginetradeoff_demopad         - Engine Design and Cost Tradeoffs
RBDSCostasLoop                 - Costas loop for RBDS application
hdlcoder_edalinks_cosimulation - HDL Verifier Cosimulation Model Generation in HDL Coder(TM)
hdlcoderrecon_m                - Image Reconstruction Using Cosimulation
hdlcoderreconcmds              - CTCOSIMCMDS - Creates Tcl commands for the Image Reconstruction model.
sdoExampleCostFunction         - An example cost function used by sdo.optimize or sdo.evaluate
gcExperimentCost               - Code for experiment cost definitions
evalcost                       - Evaluates cost at x
evalFC                         - Evaluates cost or constraint vector.
evalFCG                        - Evaluates cost variation induced by dxj by simulating gradient model.
cost                           - Various cost functions for minimization
gradient                       - Gradient of various cost functions for minimization
minimize                       - Runs the optimization algorithm to minimize the cost and estimate
minimize                       - Runs the optimization algorithm to minimize the cost and estimate
minimize                       - Runs the optimization algorithm to minimize the cost and estimate
minimize                       - Runs the optimization algorithm to minimize the cost and estimate
commdoc_rrc                    - Pulse Shaping Using a Raised Cosine Filter
commeqsim_computecostfcn       - EQ_COMPUTECOSTFCN Computes and plots MMSE Cost function for given
commeqsim_costfcnblk           - EQ_COSTFCNBLK is the open function block of the Cost Function block
commeqsim_costfcnconverg       - EQ_COSTFCNCONVERG plots convergency trajectory of MSE over its cost funct
sharedang2dcm                  - AEROBLKANG2DCM Aerospace Blockset Rotation Angles to Direction Cosine
shareddcm2ang                  - AEROBLKDCM2ANG Aerospace Blockset Direction Cosine Matrix to Rotation
cosineSimilarity               - Document similarities with cosine similarity
inflationCollisionChecker      - Configuration object for costmap collision checking.
vehicleCostmap                 - Costmap representing planning space around vehicle.
coshint                        - Hyperbolic cosine integral function
cosint                         - Cosine integral function.
fresnelc                       - Fresnel cosine integral.
coshint                        - Hyperbolic cosine integral function
cosint                         - Cosine integral function.
fresnelc                       - Fresnel cosine integral.
acos                           - Symbolic inverse cosine.
acosd                          - Inverse cosine, result in degrees.
acosh                          - Symbolic inverse hyperbolic cosine.
acsc                           - Symbolic inverse cosecant.
acscd                          - Inverse cosecant, result in degrees.
acsch                          - Symbolic inverse hyperbolic cosecant.
cos                            - Symbolic cosine function.
cosd                           - Cosine of argument in degrees.
cosh                           - Symbolic hyperbolic cosine.
coshint                        - Hyperbolic cosine integral function
cosint                         - Cosine integral function.
cospi                          - Compute cos(X*pi) accurately.
csc                            - Symbolic cosecant.
cscd                           - Cosecant of argument in degrees.
csch                           - Symbolic hyperbolic cosecant.
fresnelc                       - Fresnel cosine integral.
```

Números complexos

Definindo números complexos

a) Defina os números $z_1 = 2 + j3$, $z_2 = -2 + j3$, $z_3 = -2 - j3$ e $z_4 = 2 - j3$.

```
% clear all; close all
% clc
% z1=
% z2=
% z3=
% z4=
```

Plotando os números complexos na representação cartesiana

b) Plote os números graficamento no plano complexo

```


 %axis([XMIN XMAX YMIN YMAX])
```

Operações de módulo e fase

c) Calcule o módulo e fase dos números z_1, z_2, z_3 e z_4 utilizando as funções *abs* e *angle*. Calcule o módulo de z_1 e z_2 utilizando as funções *sqrt* e *conj.* Calcule a fase de z_1 e z_2 utilizando as funções *atan2* para rad e para graus por meio de fator de conversão e *atan2d*

```
% disp('--------c')
% mod1=
% mod2=
% mod3=
% mod4=
% phase1=
% phase2=
% phase3=
% phase4=
%
% %Outras formas
% mod1=sqrt
% mod1=sqrt
% mod2=
% mod2=
%
% phase1=atan2
% phase1d=atan2
% phase1d=angle
% phase1d=atan2d
%
% phase2=
% phase2d=
% phase1d=
% phase2d=
```

d) Determine e avalie a fase dos números complexos z_1, z_2, z_3 e z_4 pela função *atan e atand.* Corrija a discrepância obtida e avalie com o resultado esperado.

```
% disp('-------- d (erro de atan)')
% phase1=atan
% phase1d=atand
%
% phase2=atan
% phase2d=atand
%
% phase3=atan
% phase3d=atand
%
% phase4=atan
% phase4d=atand
```

Observação: Quando estiver usando funções trignométrica inversas (por ex. tg^{-1}) deve estar atento ao ângulo. Se o número complexo estiver localizado no ***segundo ou terceiro quadrante*** a resposta calculada estará deslocada de $180°$. A resposta correta é obtida somando ou subtraindo $180°$ do valor encontrado.

```
% disp('------correção de atan')
% phase2=
% phase2d=
%
% phase3=
% phase3d=
```

Convertendo representação cartesiana para a forma polar

e) Converta o z_1 para a forma polar por meio do comando *cart2pol*

```
% [z1_rad, z1_mag]=cart2pol
```

Convertendo representação polar para a forma cartesiana

f) Converta o número $z_5 = \sqrt{2}\,e^{-j\frac{\pi}{4}}$ para coordenadas cartesianas

```
% [z5_real, z5_imag]=pol2cart
```

Plot de número complexo na forma polar

g) Plot os números complexos em um diagrama polar utilizando as funções *compass* e *polarscatter* e *polarplot*

```
% figure
% compass(z1,'b')
% hold on
%
%
% figure
% polarscatter(z1_rad,z1_mag,75,'filled')
% hold on
```

```
%
%
% figure
% polarplot([z1 z2 z3 z4],'x','MarkerSize',14)
```

Operações básicas

h) Faça a operação entre os números complexos: $z_6 = z_1 + z_4$ e $z_7 = z_1 + z_2$e plote os resultados

```
% z6=z1+z4;
% figure
% compass(z6,'b')
% hold on
%
%
% legend('z_6','z_1','z_4',"Location","bestoutside")
%
% z7=z1+z2;
% figure
% compass(z7,'b')
% hold on
%
%
% legend('z_7','z_1','z_2',"Location","bestoutside")
```

i) Faça a operação $z_8 = z_4 * e^{j\pi}$ e plote o resultado

```
%Usando formula de Euler
% y=abs(1)*(cos(pi)+j*sin(pi))
```