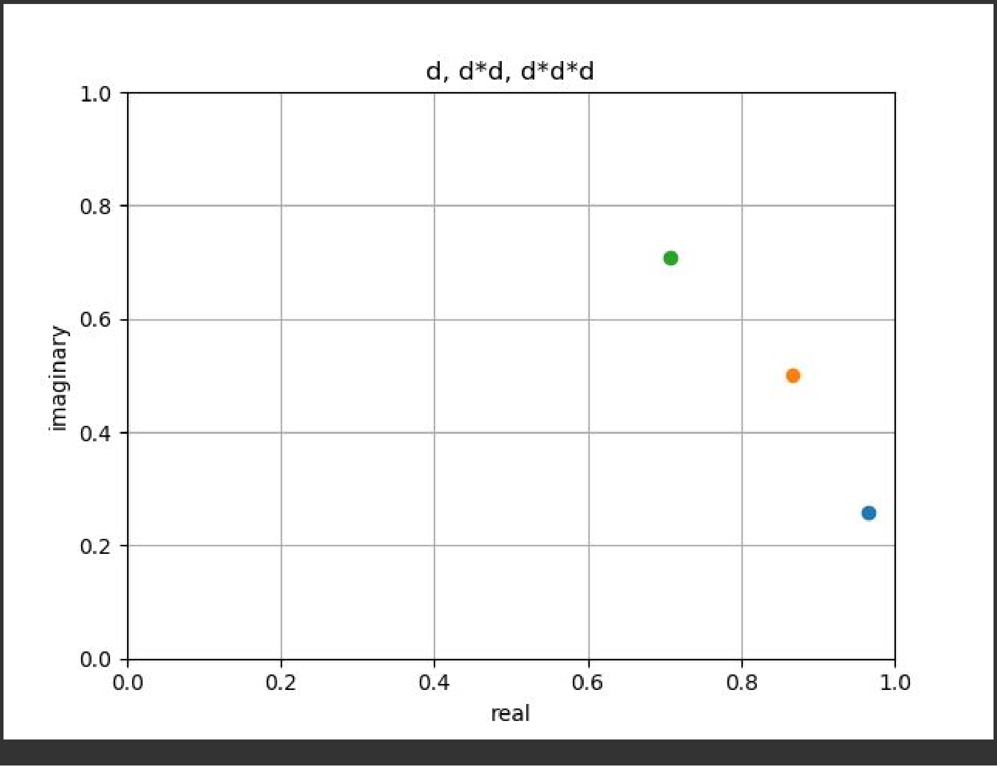
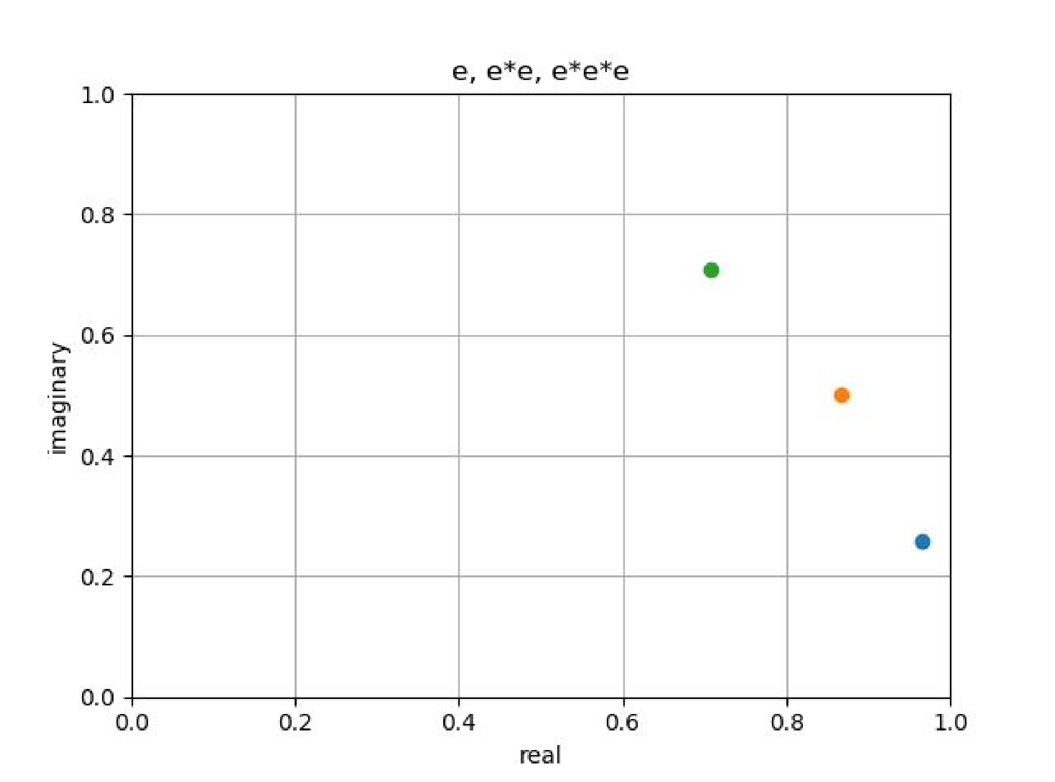
Problem 4

```
##===== 4d/e
d = cos(pi/12)+1.0im*sin(pi/12)
d2 = d*d
d3 = d2*d

d_x = [real(d) real(d2)  real(d3)]
d_y = [imag(d) imag(d2) imag(d3)]

plot(d_x,d_y,"o");
      title("d, d*d, d*d*d")
      xlabel("real")
      ylabel("imaginary")
      grid("true")
      ylim(0,1)
      xlim(0,1)
#
```



```
e = MathConstants.e;
e1 = e^(1.0im*pi/12);
e2 = e1*e1
e3 = e2*e1
```

```
e_x = [real(e1) real(e2)  real(e3)]
e_y = [imag(e1) imag(e2) imag(e3)]

plot(e_x,e_y,"o");
      title("e, e*e, e*e*e")
      xlabel("real")
      ylabel("imaginary")
      grid("true")
      ylim(0,1)
      xlim(0,1)
#
```



```
println("is e3 = d3  ?   ---->  ", e3==d3)
println("atan((sin(pi/4)/cos(pi/4))) =  ", atan((sin(pi/4)/cos(pi/4))))



julia>
is e3 = d3  ?   ---->  true
atan((sin(pi/4)/cos(pi/4))) =  0.7853981633974483
```

Problem 5

```
##===== 5a
M = [0 1;-1 0];
      M_eig = eigvals(M);
# this doesn't work for some reason:  diag(eigvals(M))
 lambda = [M_eig[1] 0; 0  M_eig[2]];
```

```
V = eigvecs(M)

julia> V
2×2 Array{Complex{Float64},2}:
 0.707107+0.0im          0.707107-0.0im
      0.0+0.707107im     0.0-0.707107im
```

```
V_i = inv(V)

julia> V_i
2×2 Array{Complex{Float64},2}:
 0.707107-0.0im  -0.0-0.707107im
 0.707107+0.0im   0.0+0.707107im
```

```
println("M eigenvalues =  [", M_eig[1], "]\n               [", M_eig[2],
"]");
println("is   M == V*lambda*V_i  true?   ------>  ", M == V*lambda*V_i)
#

M eigenvalues =  [0.0 + 1.0im]
                  [0.0 - 1.0im]

is   M == V*lambda*V_i  true?   ------>  true
```

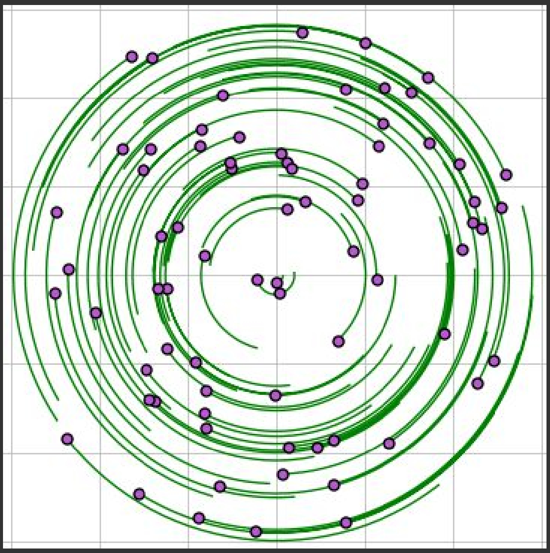
```
##===== 5b
M1_2 = V*sqrt(lambda)*V_i
println("M1_2  =   [", M1_2[1,1], "   ", M1_2[1,2], "]\n         [",
M1_2[2,1], "   ", M1_2[2,2], "]")
#
M1_2  =   [0.7071067811865476 + 0.0im   0.7071067811865475 + 0.0im]
```

```
          [-0.7071067811865475 + 0.0im    0.7071067811865476 + 0.0im]
```

```
#=== 5c
      theta in radians for M = pi/2
      theta in radians for M1_2 = pi/3
=#
```

```
##====== 5d
include("animate_matrix.jl")
animate_matrix(M) # 90 degree rotation
animate_matrix(M1_2) # 45 degree rotation

#= M & M1_2 are rotational matrixes - M rotates points by 90 degrees
clockwise, M1_2 rotates them by 45 degrees clockwise =#
```

M



M 1/2