Mikhail Grushko - BE130
Pset 2 - Problem 4

Table of Contents

Setup 1
Random vs evenly spaced 1
PV prediction accuracy test 3
Bonus 6

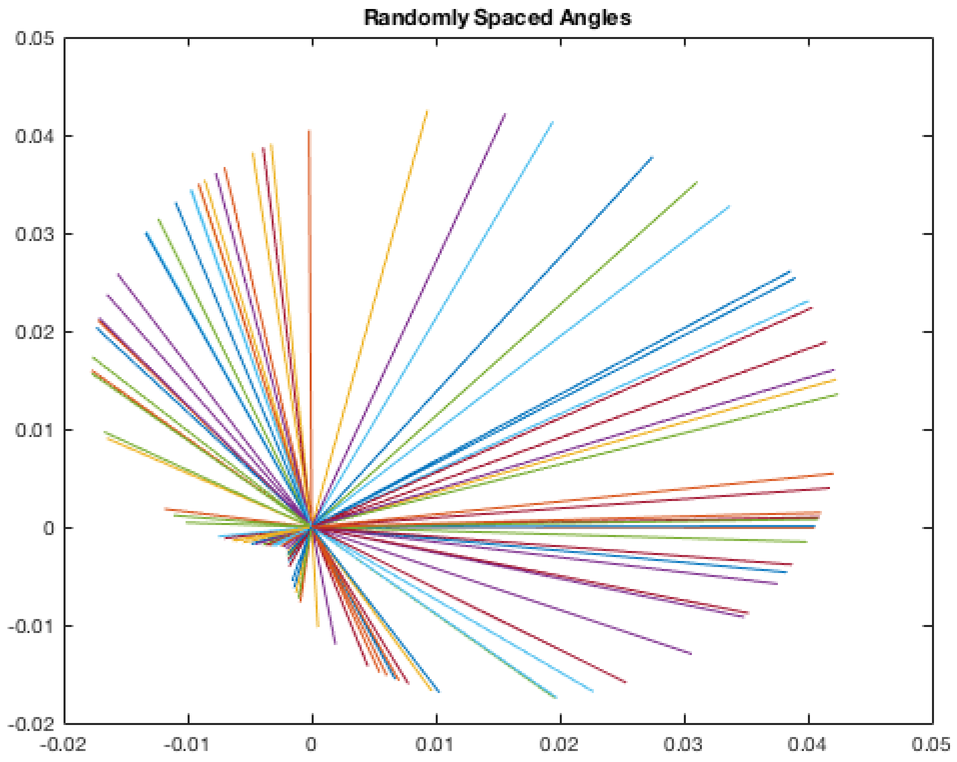
Setup

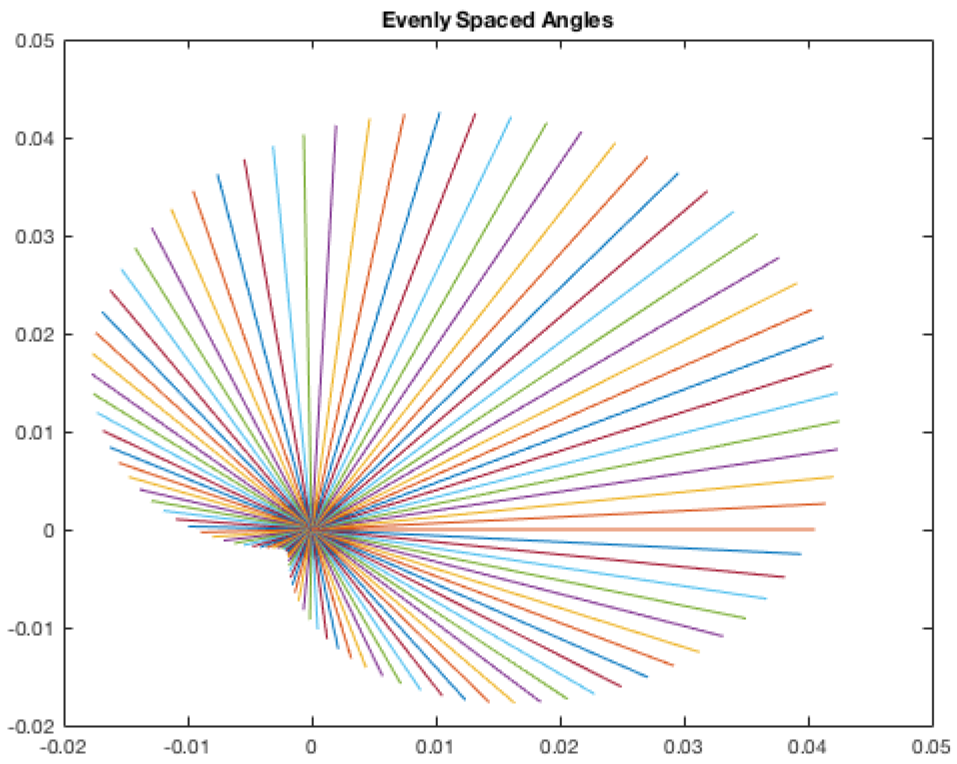
```
clc;
close all;
clear vars;
k1 = 0.022;
k0 = 0.025;
theta = pi/4;
```

Random vs evenly spaced

```
% 100 randomly spaced
size = 100;
r100 = zeros(1, size);
thetapref = 2*pi*rand(1, size);


for i = 1 : length(r100)
    r100(i) = k1 * cos(thetapref(i) - theta) + k0;
end

PV = zeros(2, 100);

for i = 1 : size
    PV(1, i) = r100(i)*cos(thetapref(i));
    PV(2, i) = r100(i)*sin(thetapref(i));
end

figure;
plotv(PV); title("Randomly Spaced Angles")

% 100 evenly spaced

size = 100;
r100 = zeros(1, size);
```

```
thetapref = 2*pi*linspace(0, 1, size);


for i = 1 : length(r100)
    r100(i) = k1 * cos(thetapref(i) - theta) + k0;
end

PV = zeros(2, 100);

for i = 1 : size
    PV(1, i) = r100(i)*cos(thetapref(i));
    PV(2, i) = r100(i)*sin(thetapref(i));
end

figure;
plotv(PV); title('Evenly Spaced Angles')
```



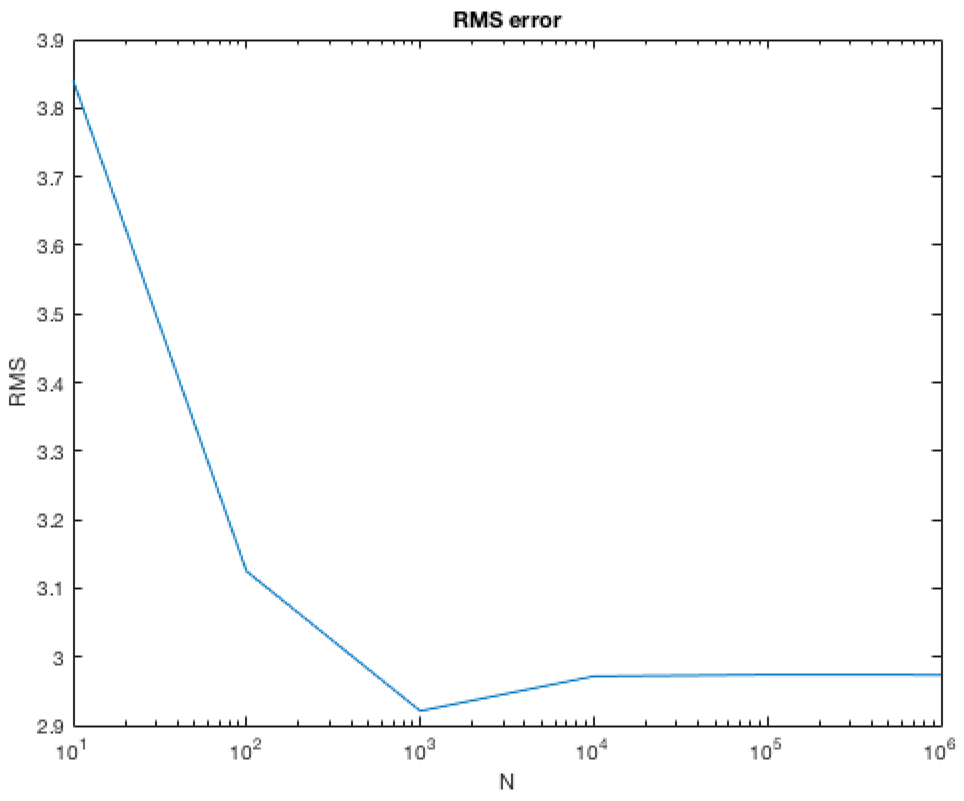


PV prediction accuracy test

```
% N = 10

size = 10;
r100 = zeros(1, size);
thetapref10 = 2*pi*rand(1, size);


for i = 1 : length(r100)
    r100(i) = k1 * cos(thetapref10(i) - theta) + k0;
end

PV10 = zeros(2, size);

for i = 1 : size
    PV10(1, i) = r100(i)*cos(thetapref10(i));
    PV10(2, i) = r100(i)*sin(thetapref10(i));
end

% N = 100

size = 100;
r100 = zeros(1, size);
thetapref100 = 2*pi*rand(1, size);
```

```
for i = 1 : length(r100)
    r100(i) = k1 * cos(thetapref100(i) - theta) + k0;
end

PV100 = zeros(2, size);

for i = 1 : size
    PV100(1, i) = r100(i)*cos(thetapref100(i));
    PV100(2, i) = r100(i)*sin(thetapref100(i));
end

% N = 1000
size = 1000;
r100 = zeros(1, size);
thetapref1000 = 2*pi*rand(1, size);


for i = 1 : length(r100)
    r100(i) = k1 * cos(thetapref1000(i) - theta) + k0;
end

PV1000 = zeros(2, size);

for i = 1 : size
    PV1000(1, i) = r100(i)*cos(thetapref1000(i));
    PV1000(2, i) = r100(i)*sin(thetapref1000(i));
end

% N = 10000
size = 10000;
r100 = zeros(1, size);
thetapref10000 = 2*pi*rand(1, size);


for i = 1 : length(r100)
    r100(i) = k1 * cos(thetapref10000(i) - theta) + k0;
end

PV10000 = zeros(2, size);

for i = 1 : size
    PV10000(1, i) = r100(i)*cos(thetapref10000(i));
    PV10000(2, i) = r100(i)*sin(thetapref10000(i));
end

% N = 100000
size = 100000;
r100 = zeros(1, size);
thetapref100000 = 2*pi*rand(1, size);


for i = 1 : length(r100)
```

```
    r100(i) = k1 * cos(thetapref100000(i) - theta) + k0;
end

PV100000 = zeros(2, size);

for i = 1 : size
    PV100000(1, i) = r100(i)*cos(thetapref100000(i));
    PV100000(2, i) = r100(i)*sin(thetapref100000(i));
end

% N = 1000000
size = 100000;
r100 = zeros(1, size);
thetapref1000000 = 2*pi*rand(1, size);


for i = 1 : length(r100)
    r100(i) = k1 * cos(thetapref1000000(i) - theta) + k0;
end

PV1000000 = zeros(2, size);

for i = 1 : size
    PV1000000(1, i) = r100(i)*cos(thetapref1000000(i));
    PV1000000(2, i) = r100(i)*sin(thetapref1000000(i));
end

diff10 = thetapref10 - theta;
diff100 = thetapref100 - theta;
diff1000 = thetapref1000 - theta;
diff10000 = thetapref10000 - theta;
diff100000 = thetapref100000 - theta;
diff1000000 = thetapref1000000 - theta;
RMS10 = (sqrt(mean((diff10).^2)));
RMS100 = (sqrt(mean((diff100).^2)));
RMS1000 = (sqrt(mean((diff1000).^2)));
RMS10000 = (sqrt(mean((diff10000).^2)));
RMS100000 = (sqrt(mean((diff100000).^2)));
RMS1000000 = (sqrt(mean((diff1000000).^2)));

RMS = [RMS10, RMS100, RMS1000, RMS10000, RMS100000, RMS1000000];
Ns = [10, 100, 1000, 10000, 100000, 1000000];
figure;
semilogx(Ns, RMS); title("RMS error"); xlabel("N"); ylabel("RMS")
```

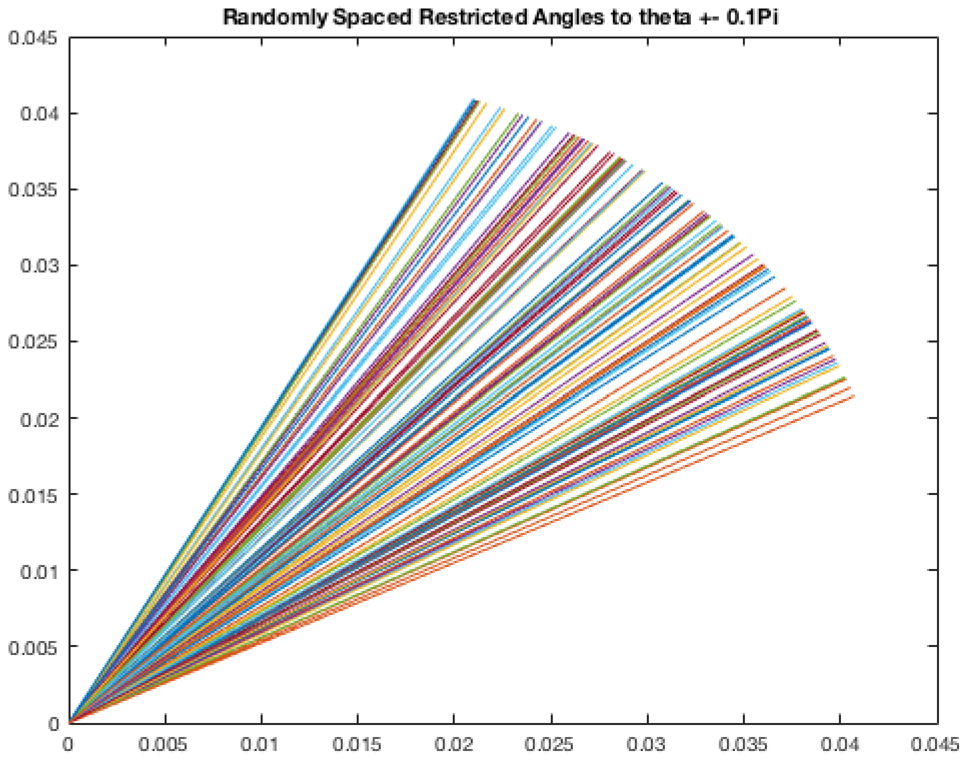


Bonus

```
% The RMS decreases with the larger sample size, due to the law of
 large numbers in statistics. In order to even further
% decrease RMS, one could completely cut off angles that are a certain
% distance away from the desired theta. Look below for the graph of
 the new
% weighting system. Also, the calculation below

% 100 randomly spaced
size = 100;
r100 = zeros(1, size);
theta = theta;
thetapref = theta + 0.2*pi*(rand(1, size) - 0.5);


for i = 1 : length(r100)
    r100(i) = k1 * cos(thetapref(i) - theta) + k0;
end

PV = zeros(2, 100);

for i = 1 : size
    PV(1, i) = r100(i)*cos(thetapref(i));
    PV(2, i) = r100(i)*sin(thetapref(i));
```

```
end

figure;
plotv(PV); title("Randomly Spaced Restricted Angles to theta +- 0.1Pi
 ")

diff = thetapref - theta;
RMS = (sqrt(mean((diff10).^2)))
```

```
RMS =

    3.8411
```



Published with MATLAB® R2017b