להלן מדריך מקוצר להפעלת הקוד:

הדרך להפעיל ריצה אחת בודדת של זיהוי מטרות באמצעות המודל המסוים (סיגנל עם מודולציה בזמן) ושיטת הפתרון שלנו:

שורת הריצה הינה:

```
[successVec,resultHist,realHist,targets,targets_Coset] =
sim1(Ci,Q,L,P,snr_db,plot_fail_sim,numSims,is_full_sample,reduce_method,...
   sample_SubNyquist_factor,nu_pulses,less_p,same_reduce_pulses_B)
```

Inputs:

Ci => A vector with integers (Preferred to be prime). The number of integers will is T - the number of channels.

Q => An integer. Determine the ambiguity factor.

L => An integer. Determine the number of targets.

P => An integer. Determine the number of pulses in each cannel.

Snr_db => A number. Determine the noise ratio in dB.

Plot_fail_sim => A Boolean. When true: plots a graph for a failed iteration. When false: doesn't plot.

numSims => An integer. Determine the numbers of test iterations in the simulation.

is_full_sample => A Boolean. When true: takes all samples. When false: Solve the algorithm with less samples.

Reduce_method => An integer. Determine which samples to process in the Algorithm.
(0 – LPF , 1 – random , 2 – blocks , 3 – same k in all channels , 4 – not the same k in all channels)
We used only 1 (random)

sample_SubNyquist_factor => A number. Determine how many samples to process. (if full sample is 100 samples and sample_SubNyquist_factor is 2, we will process 50 samples only)

nu_pulses => An integer. Determine the new size of the slow time.

less_p => An integer. Determine how many pulses to send (should be less than P, random reduce)

same_reduce_pulses_B => A Boolean. When true: will correlate between the slow time reduction and the pulses reduction. When false: will do both reductions randomly and separately.

Outputs:

successVec => A [numSims x2] matrix. Contain information about the <numSims> iterations in the simulation.
First column: The max distance of a real target and a result target (after detection)
Second column: number of the targets who were recovered successfully.

resultHist => An [numSims x L x 2] matrix. Contain for every iteration and every target the cell in the **recovered** space-velocity matrix.

realHist => An [numSims x L x 2] matrix. Contain for every iteration and every target the cell in the **original** space-velocity matrix.

Targets => A struct. Contains for every **original** target 3 numbers:

t – The target's range (in time)

f – The target's frequency (phase from the Doppler shift) – symbolizes the velocity

a – The signal amplitude – not in use.

targets_Coset => A struct. Contains for every **recovered** target 3 numbers:

t – The target's range (in time)

f – The target's frequency (phase from the Doppler shift) – symbolize the velocity

a – The signal's amplitude – not in use.

ברירת המחדל להרצת סימולציה:

```
[successVec,resultHist,realHist,targets,targets_Coset] = sim1();
```

אשר שקולה להרצת הקוד הבא:

```
[successVec,resultHist,realHist,targets,targets_Coset] = sim1([31 79 ],2,5,100,inf,false ,50,1,1,1,100,100,1);
```

בכל איטרציה בסימולציה אנו מבצעים את הסכמה הבאה:

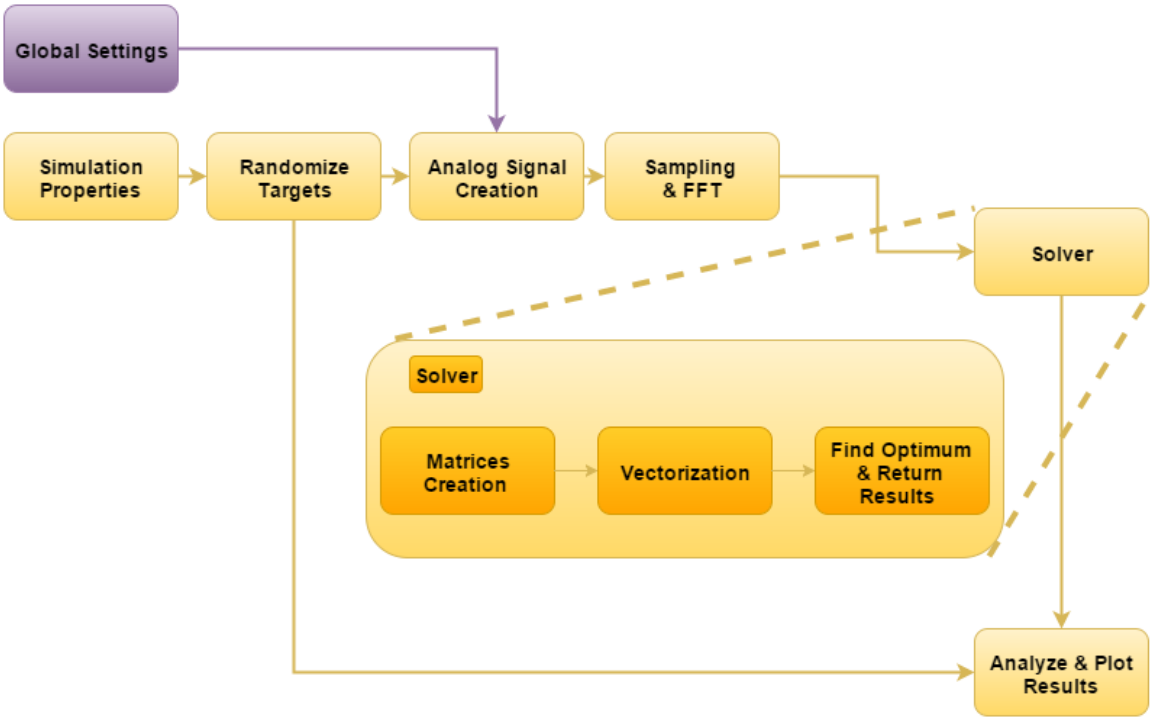

Figure 1

נציין ונרחיב מעט על הפונקציות העיקריות שאנו משתמשים בהן:

```
[g_coset] = global_settings(nu_pulses,P,L, Ci,Q,snr_db,is_full_sample,...
                reduce_method,sample_SubNyquist_factor,less_p,same_reduce_pulses_B);
```

The inputs are transferred from the sim1 input or the default values specified above.
The output is a struct that contain all kind of constants for the simulation.

```
[targets] = randomize_targets(g_coset)
```

Taking the global settings and return L randomize targets with a certain distance and velocity.
The struct targets are like specified above.

```
[x] = generate_analog_input_signal(g_coset, targets)
```

Taking the randomize target and the global settings and create (the output) an analog RX signals (here we can see our modulation).

```
[targets_Coset] = coset_nyquist(g_coset,x,targets);
```

Inputs:

G – global settings

X – the analog RX signal – a matrix, each row is a bucket.

Targets – the original randomize target – for equation check usage.

Output:

The recovered targets. Specified above.

```
[isSuccess,realHist(i,:,:),resultHist(i,:,:),successVec(i)]   =
                                    analyze_result(g_coset,targets,targets_Coset,i,plot_fail_sim);
```

Inputs:

g_coset - global settings

targets – original targets

targets_Coset – recovered targets

i – iteration number (the simulation is from several runs for statistic)

flot_fail_sim – like in sim1();

Outputs:

isSuccess – 0 if run failed, 1 if successful.

realHist(i,:,:),resultHist(i,:,:),successVec(i) – like in sim1() but for a single iteration.

לשאלות נוספות על הפעלת הקוד, ניתן לפנות לאחד מכתובות הדוא"ל הבאות:

shanlior@gmail.com

gal_winerich@hotmail.com