


سوال (۱) 

ساختار شبکه هاپفیلد که در ادامه سوالات از آن استفاده خواهد شد:

N : number of neurons

θ_i : threshold of neuron i

K : number of patterns

P : List of patterns

$$P = [p_1, p_2, \dots, p_K]$$

x_i^k : input of neuron i in pattern k

$$p_k = [x_1^k, x_2^k, \dots, x_N^k]$$

$w_{i,j}$: weight between neuron i to j determined by Hebbian rule

$$w_{i,j} = \sum_{k=1}^K x_i^k x_j^k$$

$$w_{i,i} = 0$$

$$w_{i,j} = w_{j,i}$$

$u(i, t + 1)$: linear activation of neuron i at time $t + 1$

$$u(i, t + 1) = \sum_{j=1}^N w_{i,j} a(j, t) - \theta_i$$

$a(i, t + 1)$: sign activation of neuron i at time $t + 1$

$$a(i, t + 1) = \text{sign}(u(i, t + 1))$$

$E(i, t)$: Energy of neuron i at time t

$$E(i, t) = -a(i, t)u(i, t) = -a(i, t)\left(\sum_{j=1}^N w_{i,j} a(j, t) - \theta_i\right)$$

$E(t)$: Total Energy

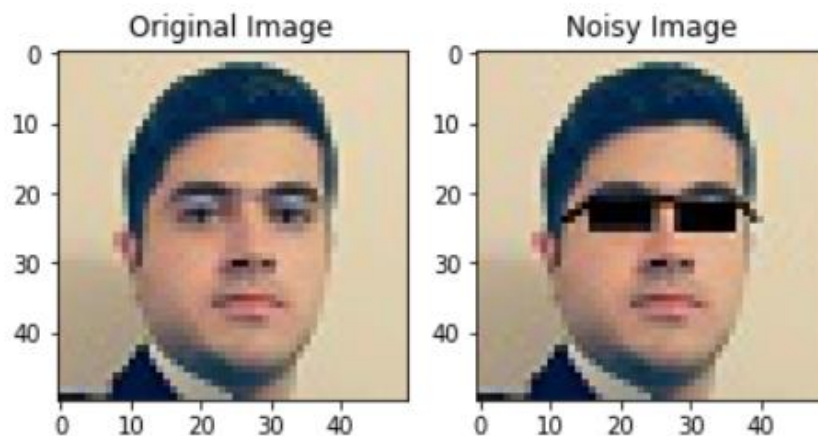
$$\begin{aligned} E(t) &= \sum_{i=1}^N E(i, t) = -\sum_{i=1}^N a(i, t) \left(\sum_{j=1}^N w_{i,j} a(j, t) - \theta_i \right) \\ &= -\sum_{i=1}^N \sum_{j=1}^N w_{i,j} a(i, t) a(j, t) + \sum_{i=1}^N a(i, t) \theta_i \end{aligned}$$

مراحل الگوریتم: ابتدا با توجه به لیست پترن ها و قاعده هبیلان ماتریس وزن را محاسبه می کنیم. سپس برای بررسی پایدار بودن ورودی جدید طی چند مرحله (Epochs) مقادیر a و انرژی را به دست می آوریم. این کار را تا زمانی ادامه می دهیم که یکی از سه حالت زیر رخ دهد:

۱. دو a آخر و پشت سر هم یکسان باشند. در این صورت در میان لیست a های ذخیره شده a یی که کمترین انرژی را دارد خروجی می دهیم.
۲. در لحظه اول a به دست آمده با ورودی برابر باشد. در این صورت ورودی پایدار بوده و خروجی نیز است.
۳. در صورتی که اتفاقات بالا نیفتد بایستی تمام Epoch ها را کامل برویم و پس از پایان از لیست a ها a یی که کمترین انرژی را دارد برگردانیم.

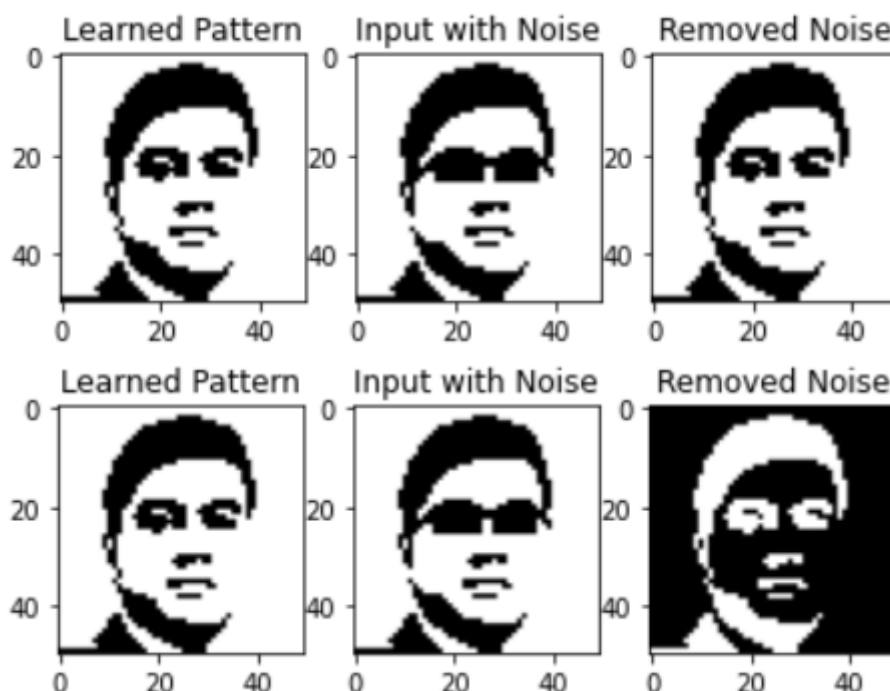
- ✓ در سوالات تمرین از اثر Threshold صرف نظر شده و مقدار آن برابر 0 می باشد ما هم در محاسبات خود آن را صفر در نظر می گیریم.
- ✓ پیاده سازی **کاملاً به صورت Vectorized** صورت گرفته و تنها در Epoch حلقه For استفاده شده است.

❖ تست شبکه هاپفیلد:



دو تصویر بالا ابتدا به صورت یک Vector که المان های آن ۱- یا ۱ هستند در آمده. تصویر سمت چپ به عنوان یک Pattern به شبکه داده شده و تصویر سمت راست به شبکه داده شده تا شبکه بتواند نویز آن را حذف کند.

اگر از تابع فعال سازی Sign استفاده کنیم مشاهده می کنیم شبکه واگرا می شود و خروجی در Epoch های زوج قرینه خروجی Epoch های فرد می شود.



این مشکل که در کوپیز شماره ۴ نیز پیش آمد را می توان با استفاده از تابع فعال سازی با فرم پیوسته حل کرد. بنابراین در پیاده سازی شبکه این امکان وجود دارد تا یکی از دو تابع فعال سازی **Tanh یا Sign** به عنوان هایپرپارامتر به شبکه داده شود.

❖ (۱.۱)

۴ پترن داریم. ابتدا ماتریس وزن متناسب با هر پترن را جداگانه مجاسبه می‌کنیم و سپس این ۴ ماتریس را با هم جمع می‌کنیم.

$$w_{i,j}^k = x_i^k x_j^k$$

$$p_1 = (1,1,1,1)$$

I/J	1	2	3	4
1	0	1	1	1
2	1	0	1	1
3	1	1	0	1
4	1	1	1	0

$$p_2 = (-1,-1,-1,-1)$$

I/J	1	2	3	4
1	0	1	1	1
2	1	0	1	1
3	1	1	0	1
4	1	1	1	0

$$p_3 = (1,1,-1,-1)$$

I/J	1	2	3	4
1	0	1	-1	-1
2	1	0	-1	-1
3	-1	-1	0	1
4	-1	-1	1	0

$$p_4 = (-1,-1,1,1)$$

I/J	1	2	3	4
1	0	1	-1	-1
2	1	0	-1	-1
3	-1	-1	0	1
4	-1	-1	1	0

$$w_{i,j} = \sum_{k=1}^K w_{i,j}^k$$

I/J	1	2	3	4
1	0	4	0	0
2	4	0	0	0
3	0	0	0	4
4	0	0	4	0

حال ورودی جدید را به مدل وارد می‌کنیم. و پایدار بودن آن را از طریق روابط زیر بررسی می‌کنیم.
 ✓ در حل این سوال به یک نکته توجه داریم: حاصل ضرب یک عدد مثبت در هر عددی علامت آن را تغییر نمی‌دهد.

$$\text{if } a > 0 \text{ then } \text{sign}(ax) = \text{sign}(x)$$

x_1, x_2, x_3, x_4 and $x_i = \pm 1$ so $\text{sign}(x_i) = x_i$

$$u(i, t + 1) = \sum_{j=1}^N w_{i,j} a(j, t)$$

$$a(i, t + 1) = \text{sign}(u(i, t + 1))$$

$$E(t) = - \sum_{i=1}^N \sum_{j=1}^N w_{i,j} a(i, t) a(j, t)$$

	1	2	3	4
$t = 0$	x_1	x_2	x_3	x_4
$t = 1$	$\text{sign}(4x_2) =$ x_2	$\text{sign}(4x_1) =$ x_1	$\text{sign}(4x_4) =$ x_4	$\text{sign}(4x_3) =$ x_3
$t = 2$	$\text{sign}(16x_1) =$ x_1	$\text{sign}(16x_2) =$ x_2	$\text{sign}(16x_3) =$ x_3	$\text{sign}(16x_4) =$ x_4

با توجه به این که ردیف $t = 0$ با $t = 2$ برابر شد و صورت سوال که ذکر کرده بود "مینیمم‌های محلی شبکه‌ی هاپفیلد دقیقا همین ورودی‌ها باشند." پس در $t = 0$ شبکه پایدار بوده و به دنبال آن $t = 2$ نیز پایدار می‌شود. پس همه موارد لیست قابل ذخیره سازی می‌باشند.

ماتریس وزن ها نیز در بالاتر محاسبه شد.

خروجی کد نیز تمامی موارد بالا را ثابت می‌کند.

```
# Q1.2_graded
# Train on Question 1.1

hopfield = Hopfield(4)

X = np.array([
    [1, 1, 1, 1],
    [-1, -1, -1, -1],
    [1, 1, -1, -1],
    [-1, -1, 1, 1]
])

a = [-1, 1, -1, 1]

W = hopfield.train(X)
yp, acc, e = hopfield.predict(a, epochs=5, activation="sign")

print("Weight Matrix:", W)
print("Test pattern:", a)
print("Nearest pattern found:", yp)
print("Accuracy:", acc * 100, "%")
print("Energy:", e)
```

100% 5/5 [00:00<00:00, 204.34it/s]

```
Nearest pattern found in iteration: 2
Weight Matrix: [[0 4 0 0]
 [4 0 0 0]
 [0 0 0 4]
 [0 0 4 0]]
Test pattern: [-1, 1, -1, 1]
Nearest pattern found: [-1  1 -1  1]
Accuracy: 100.0 %
Energy: 16
```

❖ (۱.۲) همان طور که در تصویر زیر مشاهده می‌شود شبکه توانست با ۲ پترن داده شده پایداری ورودی سوم را نشان دهد و آن را به عنوان خروجی برگرداند. (داخل کلاس Hopfield ماتریس وزن ها محاسبه می‌شود و ...)

```
# Q1.2_graded
# Train on Question 1.2

hopfield = Hopfield(6)

X = np.array([
    [1, -1, 1, -1, 1, -1],
    [1, 1, 1, -1, -1, -1],
])

a = [1, 1, 1, -1, -1, -1]

hopfield.train(X)
yp, acc, e = hopfield.predict(a, epochs=5, activation="sign")

print("Test pattern:", a)
print("Nearest pattern found:", yp)
print("Accuracy:", acc * 100, "%")
print("Energy:", e)
```

0% 0/5 [00:00<?, ?it/s]

```
Input is already stable.
Nearest pattern found in iteration: 1
Test pattern: [1, 1, 1, -1, -1, -1]
Nearest pattern found: [ 1 1 1 -1 -1 -1]
Accuracy: 100.0 %
Energy: -28
```

قسمت بعدی سوال کمی ابهام دارد پس تمام حالات ممکن در ادامه بررسی خواهد شد. که احتمالا حالت دو مد نظر سوال بوده.

حالت یک: پترن جدید فقط در X ظاهر شود: مشابه حالت بالا

```
# Q1.2_graded
# Train on Question 1.2
# With new pattern inside X
# Case 1

hopfield = Hopfield(6)

X = np.array([
    [1, -1, 1, -1, 1, -1],
    [1, 1, 1, -1, -1, -1],
    [-1, 1, 1, -1, -1, -1],
])

a = [1, 1, 1, -1, -1, -1]

hopfield.train(X)
yp, acc, e = hopfield.predict(a, epochs=5, activation="sign")

print("Test pattern:", a)
print("Nearest pattern found:", yp)
print("Accuracy:", acc * 100, "%")
print("Energy:", e)
```

0% 0/5 [00:00<?, ?it/s]

```
Input is already stable.
Nearest pattern found in iteration: 1
Test pattern: [1, 1, 1, -1, -1, -1]
Nearest pattern found: [ 1 1 1 -1 -1 -1]
Accuracy: 100.0 %
Energy: -38
```

حالت دو: پترن جدید فقط در a ظاهر شود: پایدار نمی‌شود.

```
# Q1.2_graded
# Train on Question 1.2
# With new pattern as a
# Case 2

hopfield = Hopfield(6)

X = np.array([
    [1, -1, 1, -1, 1, -1],
    [1, 1, 1, -1, -1, -1],
])

a = [-1, 1, 1, -1, -1, -1]

hopfield.train(X)
yp, acc, e = hopfield.predict(a, epochs=5, activation="sign")

print("Test pattern:", a)
print("Nearest pattern found:", yp)
print("Accuracy:", acc * 100, "%")
print("Energy:", e)
```

0% 0/5 [00:00<?, ?it/s]

```
Consecutive same pattern in iteration: 1
Nearest pattern found in iteration: 1
Test pattern: [-1, 1, 1, -1, -1, -1]
Nearest pattern found: [ 1  1  1 -1 -1 -1]
Accuracy: 83.33333333333334 %
Energy: -28
```

حالت سه: پترن جدید هم در X و هم در a ظاهر شود: پایدار نمی‌شود.

```
# Q1.2_graded
# Train on Question 1.2
# With new pattern inside X, and as a
# Case 3

hopfield = Hopfield(6)

X = np.array([
    [1, -1, 1, -1, 1, -1],
    [1, 1, 1, -1, -1, -1],
    [-1, 1, 1, -1, -1, -1]
])

a = [-1, 1, 1, -1, -1, -1]

hopfield.train(X)
yp, acc, e = hopfield.predict(a, epochs=5, activation="sign")

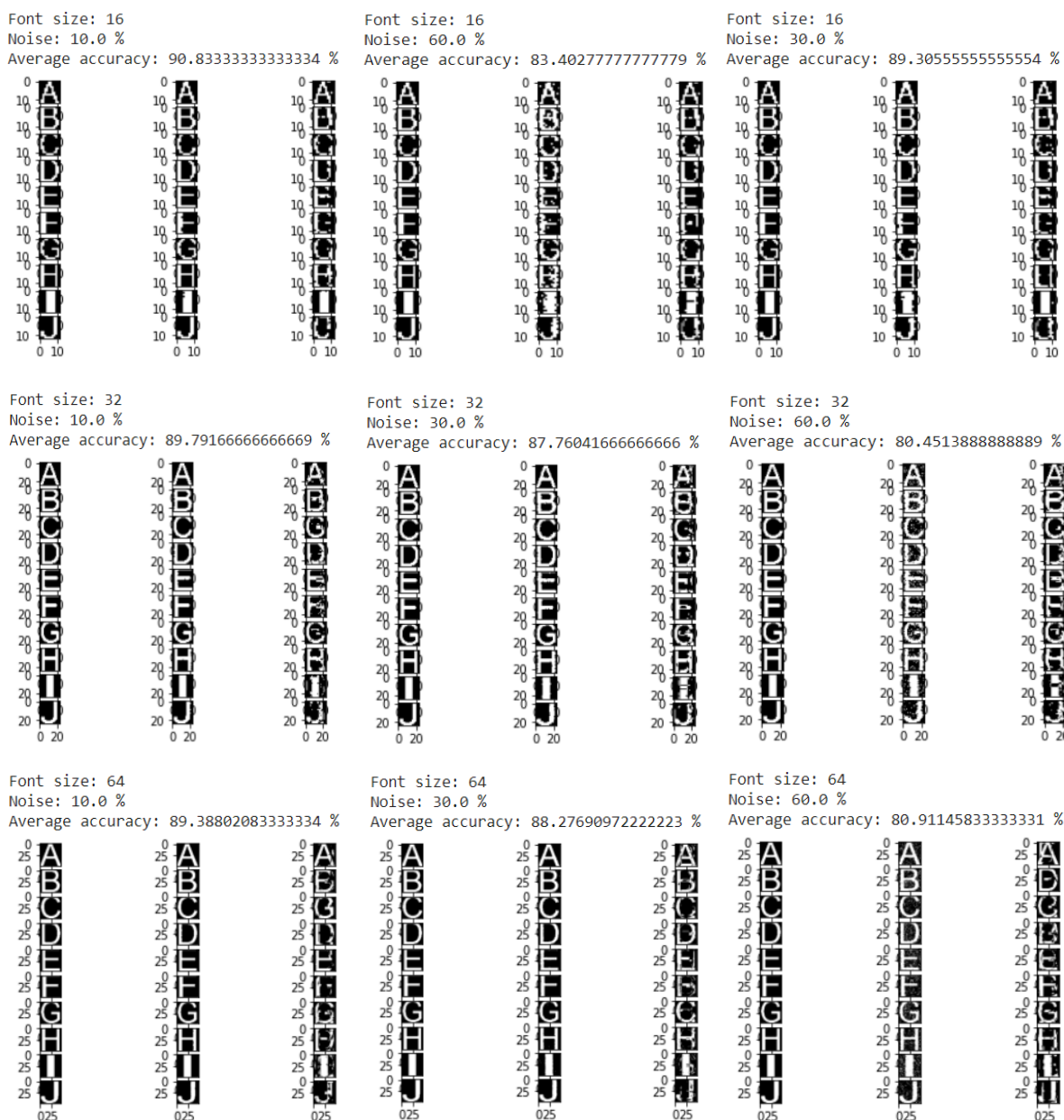
print("Test pattern:", a)
print("Nearest pattern found:", yp)
print("Accuracy:", acc * 100, "%")
print("Energy:", e)
```

0% 0/5 [00:00<?, ?it/s]

```
Consecutive same pattern in iteration: 1
Nearest pattern found in iteration: 1
Test pattern: [-1, 1, 1, -1, -1, -1]
Nearest pattern found: [ 1  1  1 -1 -1 -1]
Accuracy: 83.33333333333334 %
Energy: -38
```


❖ ۱.۳ نکات تکمیلی در داخل نوتبوک موجود است اینجا فقط به مقایسه خروجی ها می‌پردازیم. برای مشاهده دقیق‌تر زوم کنید.

font\noise	10%	30%	60%
16	90.83	89.30	83.40
32	89.79	87.76	80.45
64	89.38	88.27	80.91



نتیجہ:

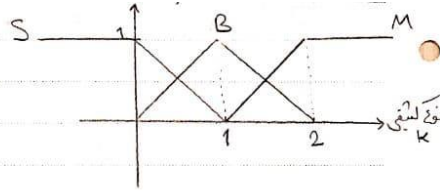
✓ با افزایش نويز دقت کاهش می یابد.

✓ با افزایش سایز فونت دقت به صورت کلی کاهش می‌یابد.

نوع کثیفی
Kind

عرق Sweat
لکه Blob
گل Mud

$0 \leq k \leq 2$



$$\mu_S(k) = \begin{cases} 1 & k < 0 \\ (1-k) & 0 \leq k < 1 \\ 0 & k \geq 1 \end{cases}$$

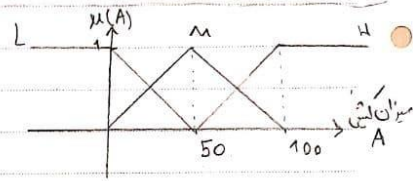
$$\mu_B(k) = \begin{cases} 0 & k < 0 \\ k & 0 \leq k < 1 \\ (2-k) & 1 \leq k < 2 \\ 0 & k \geq 2 \end{cases}$$

$$\mu_M(k) = \begin{cases} 0 & k < 1 \\ (k-1) & 1 \leq k < 2 \\ 1 & k \geq 2 \end{cases}$$

میزان کثیفی
Amount

کم Low
متوسط Medium
زیاد High

$0 \leq A \leq 100$



$$\mu_L(A) = \begin{cases} 1 & A < 0 \\ \frac{50-A}{50} & 0 \leq A < 50 \\ 0 & A \geq 50 \end{cases}$$

$$\mu_H(A) = \begin{cases} 0 & A < 50 \\ \frac{A-50}{50} & 50 \leq A < 100 \\ 1 & A \geq 100 \end{cases}$$

$$\mu_M(A) = \begin{cases} 0 & A < 0 \\ \frac{A}{50} & 0 \leq A < 50 \\ \frac{100-A}{50} & 50 \leq A < 100 \\ 0 & A \geq 100 \end{cases}$$

Rules Table:

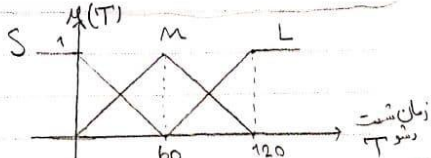
kind	Amount Low	Medium	High
Sweat	L	L	M
Blob	L	M	H
Mud	M	H	H

✓ مقدارهای جدول برای
زمان شست و شو (Wash Time)
می باشند.

زمان شست و شو
Time

کوتاه Short
متوسط Medium
طولانی Long

$0 \leq T \leq 120$
دقیقه



$$\mu_S(T) = \begin{cases} 1 & T < 0 \\ \frac{60-T}{60} & 0 \leq T < 60 \\ 0 & T \geq 60 \end{cases}$$

$$\mu_M(T) = \begin{cases} 0 & T < 0 \\ \frac{T}{60} & 0 \leq T < 60 \\ \frac{120-T}{60} & 60 \leq T < 120 \\ 0 & T \geq 120 \end{cases}$$

$$\mu_L(T) = \begin{cases} 0 & T < 60 \\ \frac{T-60}{60} & 60 \leq T < 120 \\ 1 & T \geq 120 \end{cases}$$

$k = 1.2 \Rightarrow \begin{cases} \mu_B(k) = 2 - 1.2 = 0.8 \\ \mu_M(k) = 1.2 - 1 = 0.2 \end{cases}$

مقاله
کتاب روش
MOM

$A = 27 \Rightarrow \begin{cases} \mu_L(A) = \frac{50-27}{50} = 0.46 \\ \mu_M(A) = \frac{27}{50} = 0.54 \end{cases}$

k = Blob	0.8	A = Low	0.46	0.46
k = Blob	0.8	A = Medium	0.54	0.54
k = Mud	0.2	A = Low	0.46	0.2
k = Mud	0.2	A = Medium	0.54	0.2

max 0.54
k = Blob
A = Medium

Washing Time: Medium = 60 min

Rules جدول

GMP: Fact: x is \hat{A}' : $R(x)$ (2.2 سوال)

Rule: if x is \hat{A} then y is \hat{B} : $R(x, y)$

Result: y is \hat{B} : $R(y) = R(x) \circ R(x, y)$ 15

Larsen: $\hat{B}'(y) = \sup_{x \in X} \min(\hat{A}'(x), R(x, y))$ $y \in Y$

• $\hat{B}'(y) = \sup_{x \in X} \min(A'(x), A(x)B(x))$ $D_x = \{3\}$ $R_y = \{-5, -4, \dots, -1\}$ 20

$$B'(-5) = \min(A'(3), A(3)B(-5)) = \min(1, 0.5 \times 0) = 0$$

$$B'(-4) = \min(A'(3), A(3)B(-4)) = \min(1, 0.5 \times 0.5) = 0.25$$

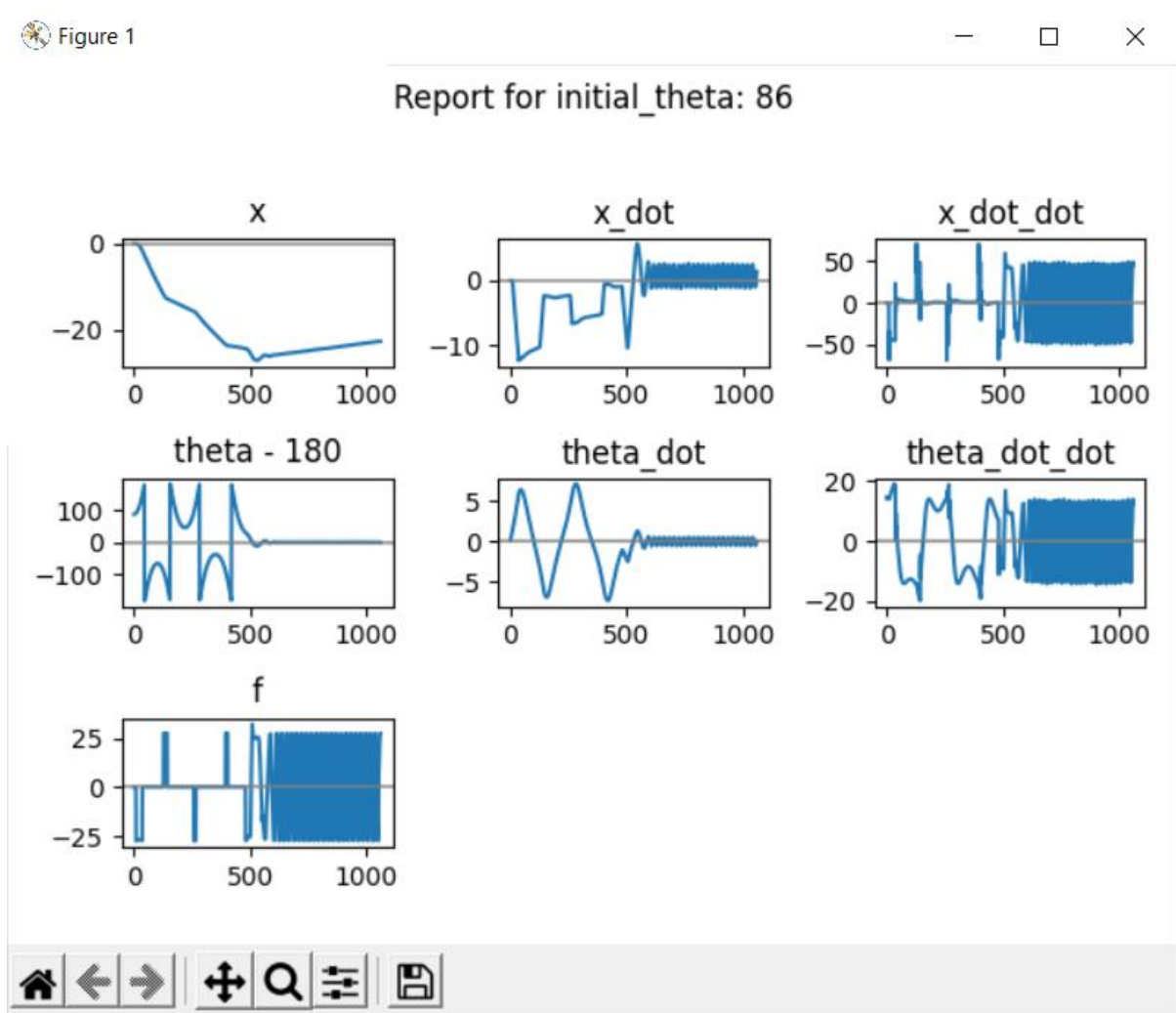
$$B'(-3) = \min(A'(3), A(3)B(-3)) = \min(1, 0.5 \times 1) = 0.5$$

$$B'(-2) = \min(1, 0.5 \times 0.67) = 0.335$$

$$B'(-1) = \min(1, 0.5 \times 0.33) = 0.165$$
 25

• $B'(y) = 0.5 \times B(y)$: $y \in R_y$ نصف

Figure 1



آونگ به طور کامل پایدار شده.

گاری در حدود Cycle 500 برای چند ثانیه پایدار شده است.