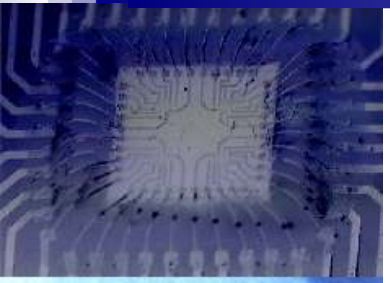
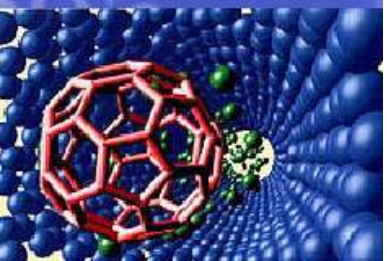






A funkcionalitás tervezése Celluláris Hullámszámítógépeken:

Template tervezés

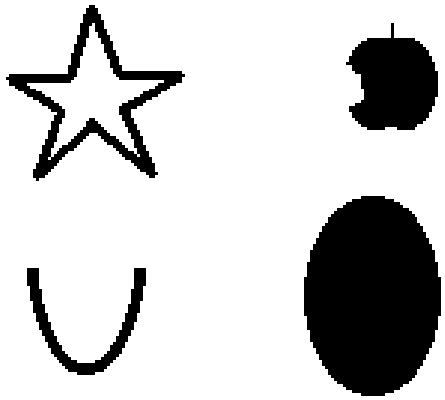
András Horváth

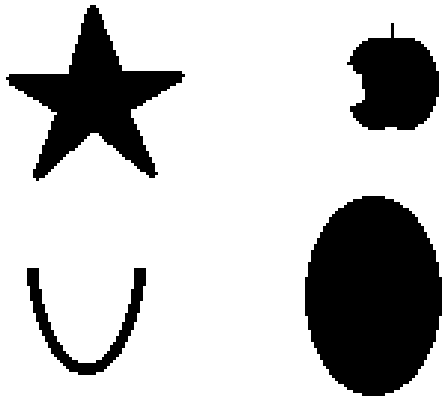

CNN
A Time Warner Company

Template-ek a gyakorlatban

Az eddigiek során már láttuk, hogy a templatek működnek és a gyakorlatban is használhatóak különböző feladatok megvalósítására

A template library-ből, egy könyvtárból töltöttük be őket





A tudomány korai története

A sötét középkor: Miszticizmus

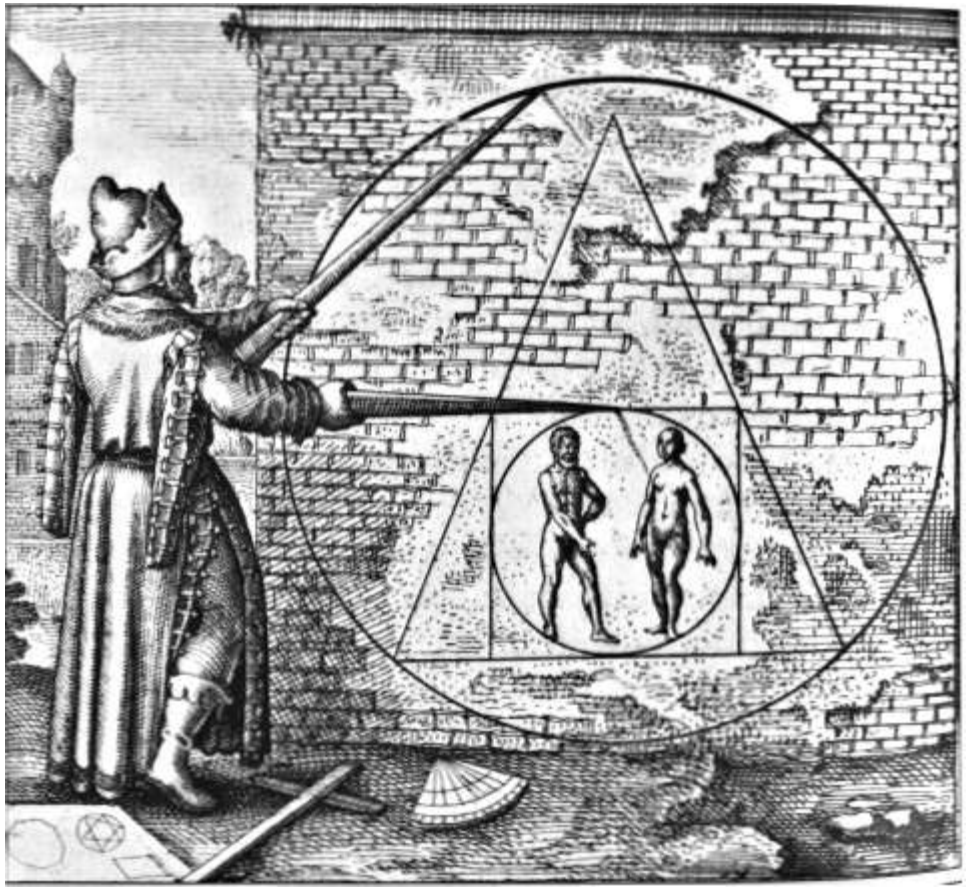
X eredménye Y

2+3=5

Tapasztalaton és

Megfigyelésen alapszik

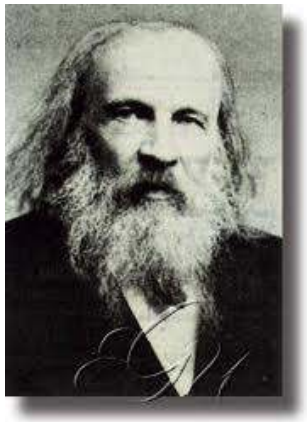
Nincs funkcionalitás



Alkémia, Taxonomiák és enciklopédiák, később törvények

Newton és a tudományos forradalom

Mengyelejev vs Bohr.



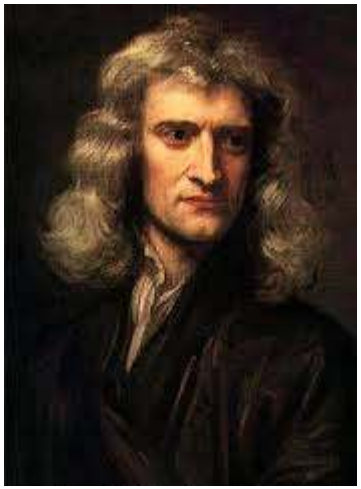
Scientific revolution

Egy probléma okának megtalálása:

Jó tervezésre is

3+3=6 és 2+4=6

A funkcionalitás a legfontosabb

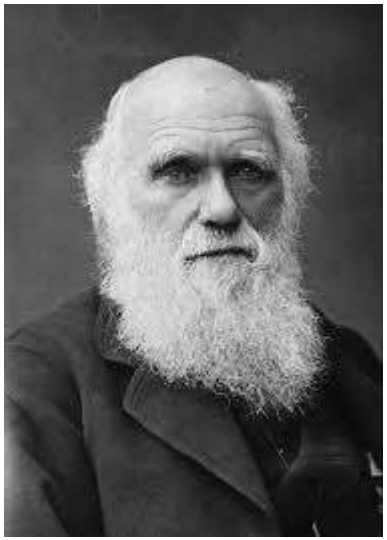


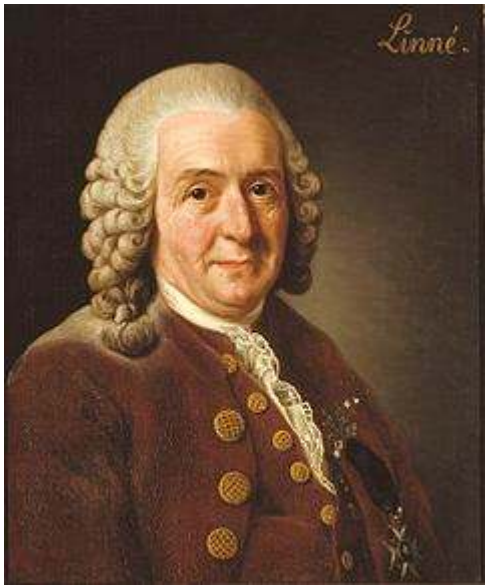
Megadják a törvények, szabályok

'hatáskörét' is.

Newton vs Kepler

Mendel, Darwin VS Linne





Miért?

Meg kell értenünk az alapvető jelenségeket/dinamikát, hogy képesek legyünk reprodukálni, (s ami még fontosabb megtrevezni) a funkciót.

Tudományosan a legfontosabb, hogy megértsünk, egy jelenséget, s megmagyarázzuk a 'legegyszerűbb', 'legáltalánosabb' magyarázattal.

Ha megértettük a dinamikát → Meg kell oldanunk az inverz problémát:

Ismerve a bemenetet és az elvárt kimenetet, meg tudjuk adni a tényleges dinamikát, ami az adott funkcionalitással bír?

Ha igen, akkor tudunk számítógépet tervezni

Funkcionalitás tervezése

Template tervezési módszerek:

- Intuitív template tervezés

- Adatbázis alapú tervezés, tanító algoritmusok (genetic algorithm)

Sosem lehetünk teljesen biztosak, hogy az elért funkció általános. :(

- Direkt tervezés, Direct design: Megértve a dinamikát, meg tudjuk oldani a hálózatot leíró differenciál egyenletet. Ennek segítségével meg tudjuk mondani, hogy milyen funkcióval bír egy template.

Ezáltal megoldhatjuk az inverz problémát is, és generálhatunk template-eket egy bemenet, kimenet párra.

Meg tudjuk oldani a diffegyeneletet?

NxM-es hálózat esetén:

NxM dimenziós diffegyenlet – Ha van visszacsatolás, mindegyik elem mindegyikkel összefügg

Elég bonyolult egy ilyen diffegyenletrendszer megoldása

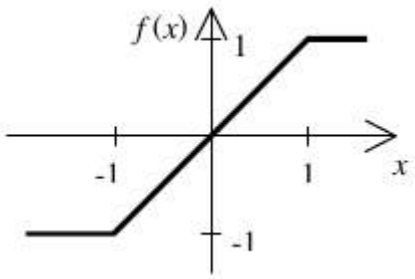
Nem propagáló CNN I.

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & a_{00} & 0 \\ 0 & 0 & 0 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} b_{-1-1} & b_{-10} & b_{-11} \\ b_{0-1} & b_{00} & b_{01} \\ b_{1-1} & b_{10} & b_{11} \end{bmatrix}, z = i$$

$$\dot{x}(t) = -x(t) + a_{00}\, y(t) + s$$

$$s = \sum_{C(kl) \in N_r(i,j)} \mathbf{B}_{ij,kl} u_{kl} + z;$$

$$y = f(x)$$



Nincs csatolás a feedback tempalte-ben

Az eredmény nem fog tovább terjedni. Csak a szomszédos elemektől függ (9 diffegyenletből álló rendszer).

(Az ilyen típusú template-ek mindig stabilak).

Feladat és template típusok

Az alábbi nem propagáló feladatosztályokat azonosíthatjuk:

- Lokális mintafelismerés

- Pixel változás felismerés

- Térbeli bináris logikai függvények, két bemenettel és egy kimenettel

Hogyan működik?

Ha nem tudjuk megérteni/elmagyarázni egy-egy létező template működését, esélyünk sincs rá,hogy új template-eket tervezzünk

Differenciál egyenlet megoldása driving-point plot-tal

Egy egyszerű egyenlet:

xdot=-(x+3)^2+4

0=-(x+3)^2+4

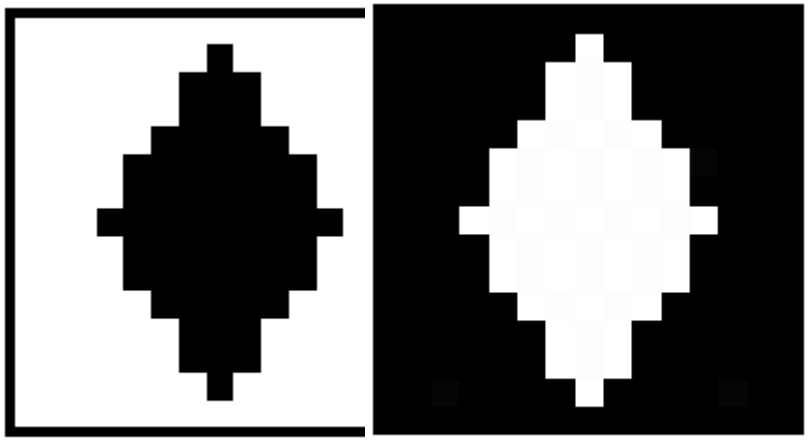
Mik lesznek az egyenlet fix-pontjai?

X0=-1 x1=-4

Hogyan működik?

Példa template - LOGNOT

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad z = \boxed{0}$$



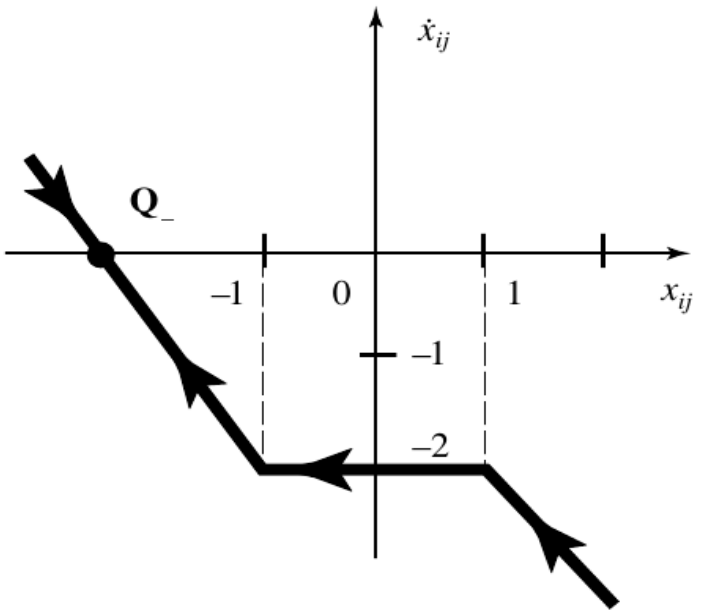
input $t = 0.5$

Hogyan működik?

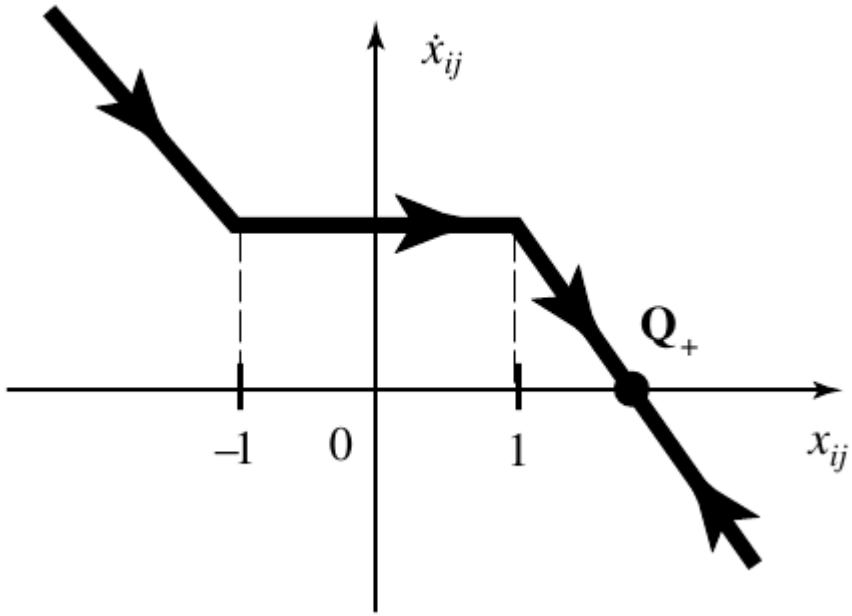
Példa template - LOGNOT

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad z = \boxed{0}$$

U=1



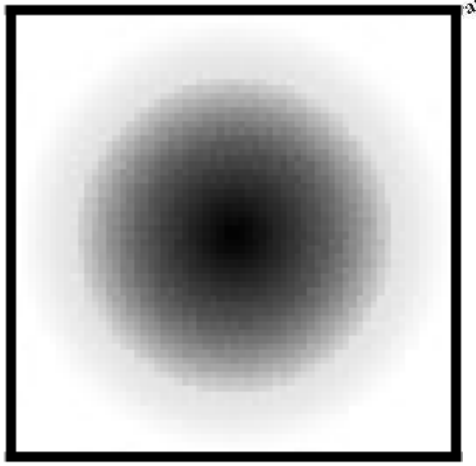
U=-1



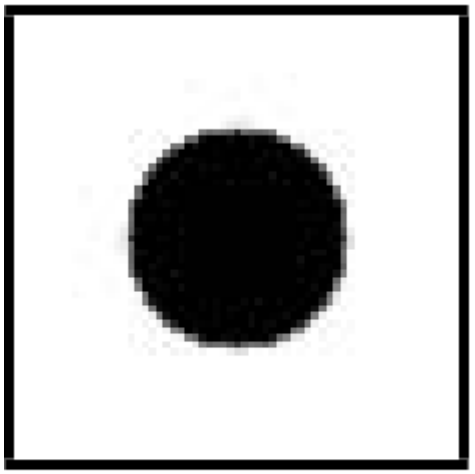
Hogyan működik?

Példa template - Threshold z=-0.4

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad z = \boxed{-z^*},$$



initial state

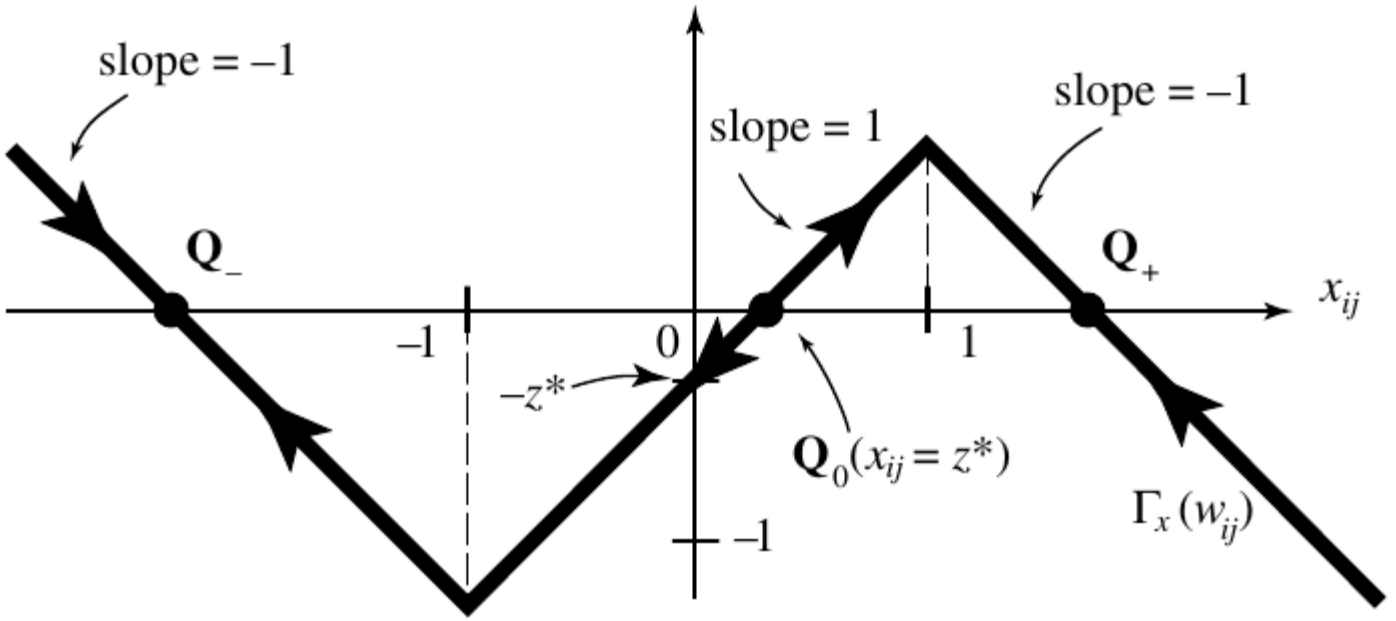


$t = 0.5$

Hogyan működik?

Példa template - Threshold z=-0.4

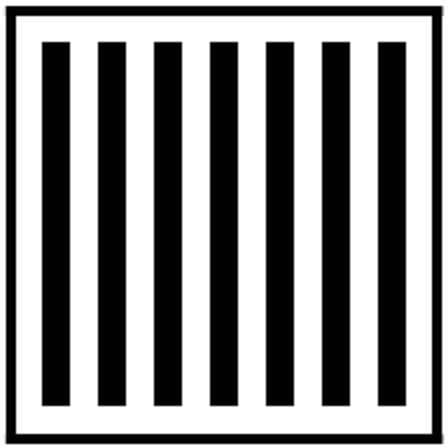
$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad z = \boxed{-z^*},$$



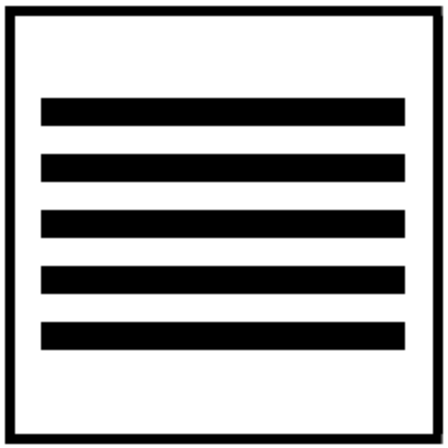
Hogyan működik?

Példa template - LOGOR

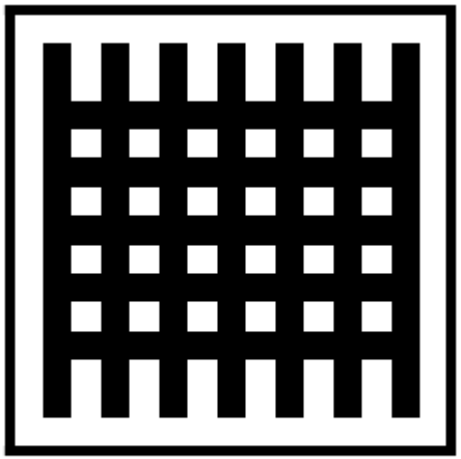
$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad z = \boxed{2}$$



input



initial state



$t = 0.5$

Hogyan működik?

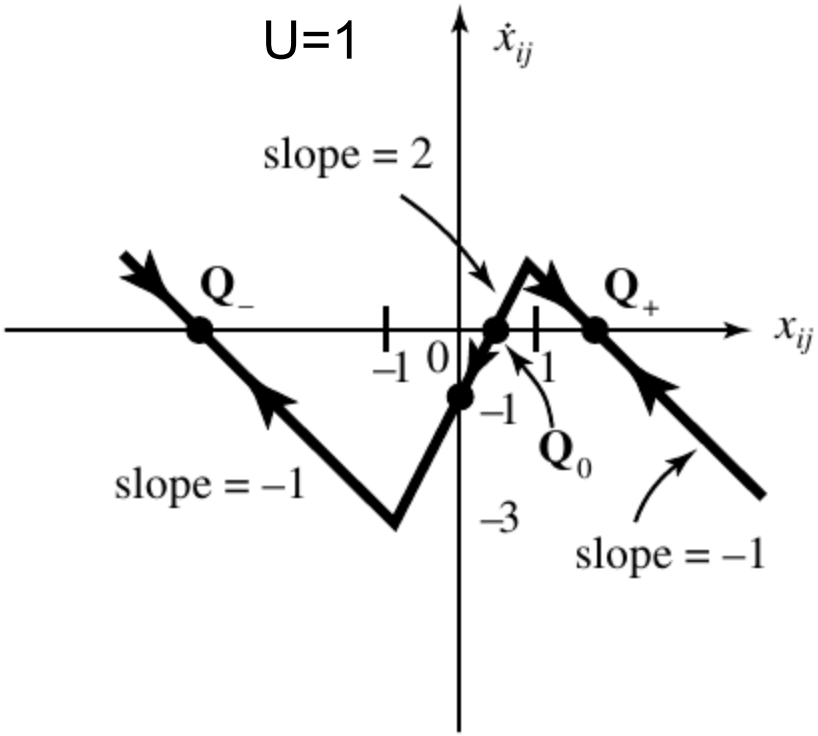
Példa template - LOGOR

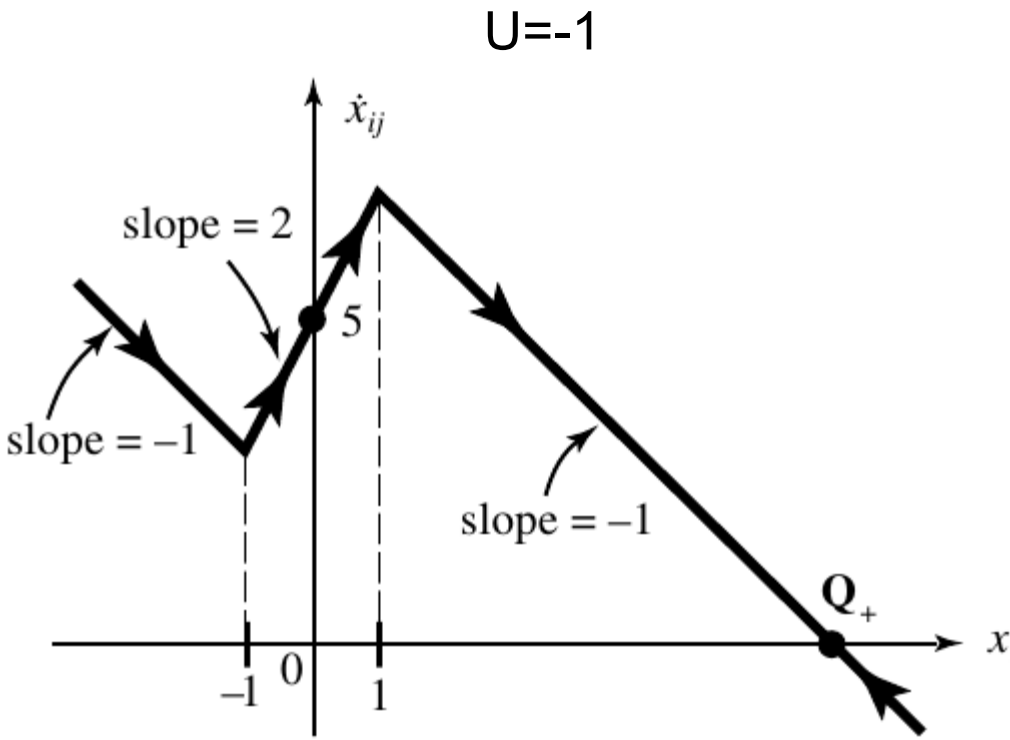
$$\mathbf{A} = \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 3 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} \qquad \mathbf{B} = \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 3 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} \qquad z = \boxed{2}$$

Hogyan működik?

Példa template - LOGOR

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad z = \boxed{2}$$





Template tervezés

Tervezzünk egy tempalte-et!

Tervezzük meg a FILLBLACK/FILLWHITE template-eket.

FILLBLACK:

A kimeneti kép minden pixele fekete lesz a bemeneti kép pixeleitől függetlenül

FILLWHITE:

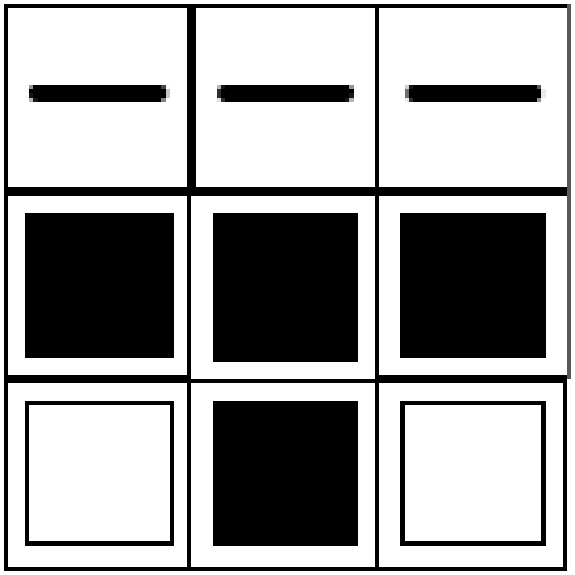
A kimeneti kép minden pixele fehér lesz a bemeneti kép pixeleitől függetlenül

Template tervezés

Tervezzük meg a FILLBLACK/FILLWHITE template-eket.

Minta felismerés

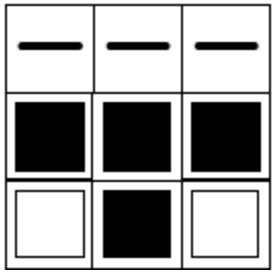
Tervezzünk template-et, ami a megadott mintát fogja detektáltni: A kimeneten csak azon pixelek lesznek feketék, ahol ez a minta a bemenet.



Minta felismerés

Az első feladat, hogy megtaláljuk a template sablonját (template-jét), a lergeszerűbb olyan formát, ami képes lehet megoldani ezt a feladatot. Itt a paraméterek fogják tükrözni, hogy mely értékek függenek egymástól.

A template első verziója:



$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & a & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 & 0 & 0 \\ b & b & b \\ -b & b & -b \end{bmatrix}, \quad z = i$$

Tervezzük meg driving-point plot-tal

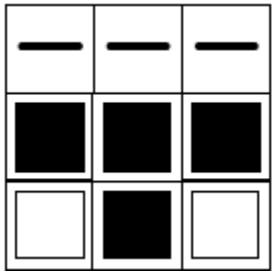
Minta felismerés

Tervezés igazságtáblázat segítségével (könnyen automatizálható)

Tervezzünk robosztus tempalte-et, elég,ha a 6-ból csupán 5 pixel megegyezik

A problémá átírható egy egyenlőtlenségrendszer megoldásává

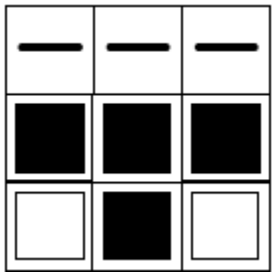
Nem kell megoldanunk a differenciálegyenletrendszert (csupán egy hagyományos algebrai egyenletrendszert.)



$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & a & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 & 0 & 0 \\ b & b & b \\ -b & b & -b \end{bmatrix}, \quad z = i$$

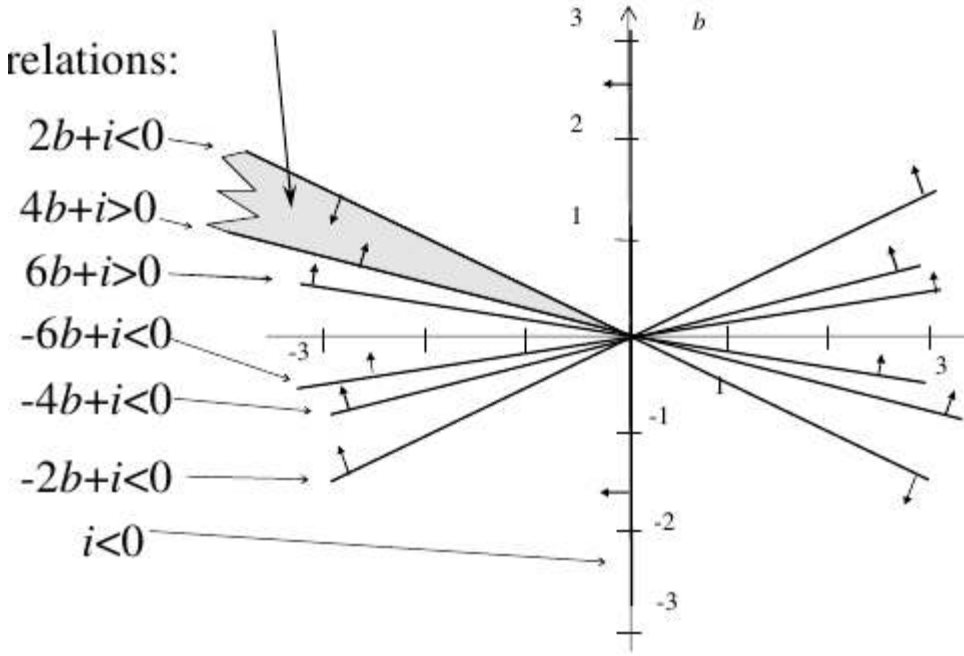
Minta felismerés

Az igazságtáblázat



$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & a & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 & 0 & 0 \\ b & b & b \\ -b & b & -b \end{bmatrix}, \quad z = i$$

# of matching pixels	*desired output*	*relation*
6	black (+1)	$6b+i>0$
5	black (+1)	$4b+i>0$
4	white (-1)	$2b+i<0$
3	white (-1)	$i<0$
2	white (-1)	$-2b+i<0$
1	white (-1)	$-4b+i<0$
0	white (-1)	$-6b+i<0$



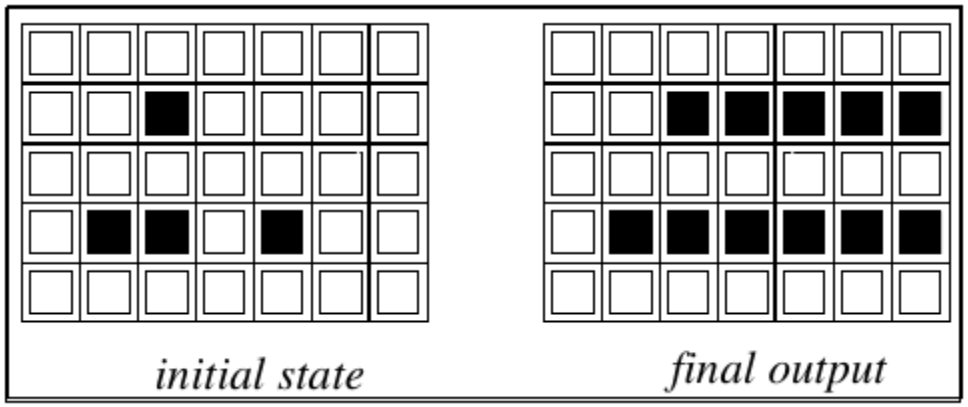
Propagáló template-ek

Shadow tempalte - árnyék generálás:

A feladat, hogy jobb oldali árnyékot generáljunk minden fekete pixelhez.

Egy olyan template-et kel lterveznünk, ami ha fekete pixelt 'észlel' a jobb oldali pixelt feketére változtatja...és így tovább és tovább

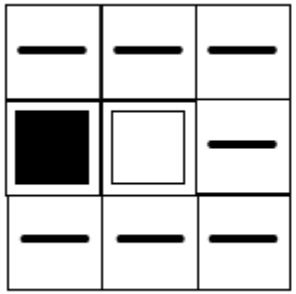
Ahhoz,hogy a tovaterjedést elérjük, az A template-et kell használni.



Tervezés driving-point plot-tal

Propagating templates

Template formula:



$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 \\ b & a & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad z = i$$

lokális pixel elrendezés	*várt kimenet*	*állapot*	*egyenlőt-lenség*
	fehér	inaktív	$-1-a-b+i<0$
	fekete	inaktív	$-1+a-b+i>0$
	fekete	inaktív	$-1+a+b+i>0$
	fekete	aktív	$1-a+b+i>0$

Igazságtáblázat minden lehetséges esetre

2^N lehetséges variáció (Ez esetben 4 lehetséges eset)

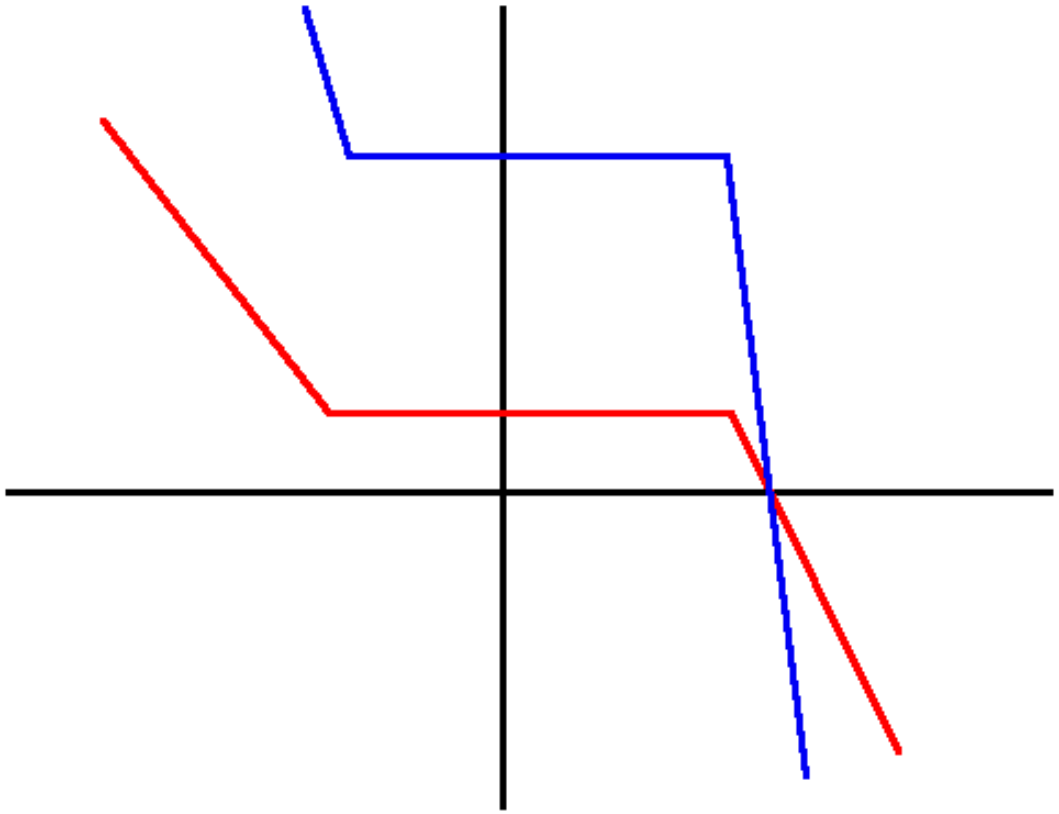
Propagating templates

Egy lehetséges megoldás

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 \\ 3 & 3 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad z = 3$$

Sebesség

Válasszuk azt a template-et, ahol a változás a lehető leggyorsabb



A maximális érték a chipen elérhető legnagyobb feszültség lesz. Ezáltal megnő az áramkör fogyasztása

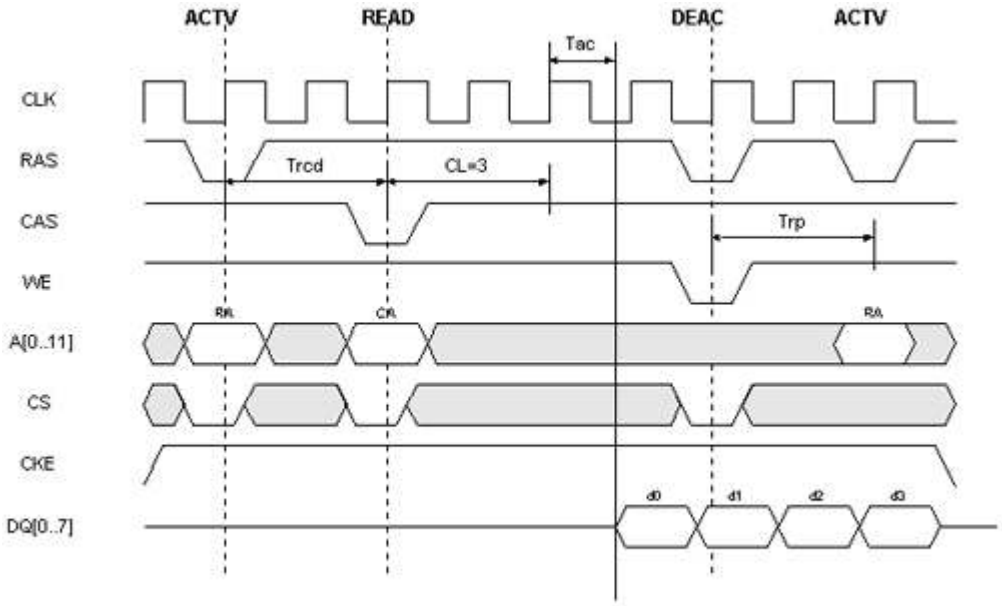
CNN előnye

Nem az órajelet kell növelni ,csak a tápot, a dinamika magától felgyorsul

1 State equation

$$\dot{x}_{ij} = -x_{ij} + \sum_{C(k,l)\in S_r(i,j)} A(i,j;k,l)y_{kl} + \sum_{C(k,l)\in S_r(i,j)} B(i,j;k,l)u_{kl} + z_{ij}$$

Hagyományos processzor esetében órajelet, frekvenciát kell növelni,s meg kell tartani a szinkronizációt



SDRAM burst read timing

Template optimalizáció

Ahogy ezt már láttuk, általában több (végtelen sok) lehetséges megoldáshoz jutunk (akár DP-plot-tal, akár az egyenlőtlenségrendszer megoldásával)

Melyik megoldást válasszuk?

Melyik a legjobb megoldás?

Mit jelent a 'legjobb'?

Létezik-e optimális template egy adott feladatra?

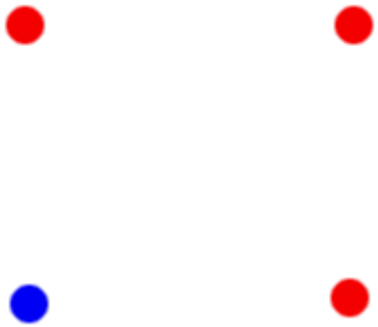
Milyen szempontból optimális

Sebesség, robosztusság, fogyasztás?

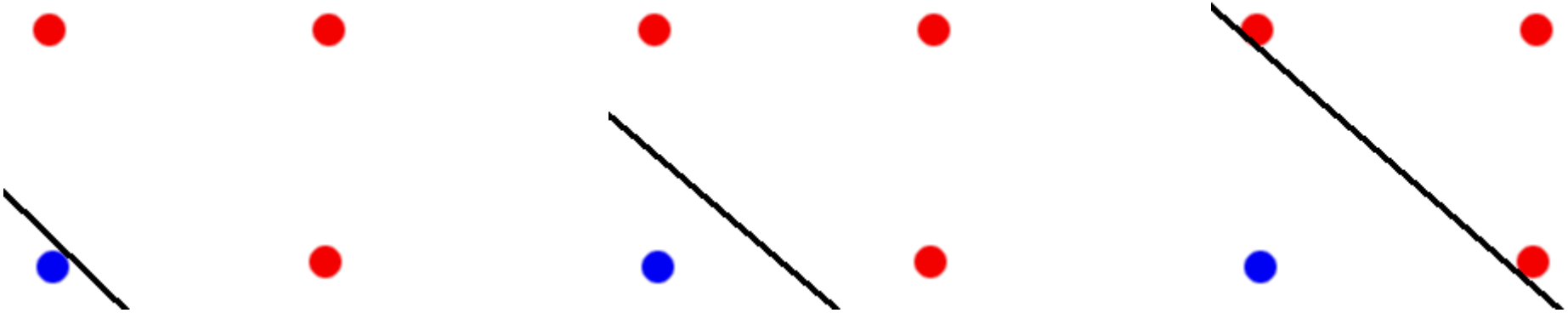
Robusztusság

Válasszuk az a template-et, ami legkevésbé érzékeny a zaj-ra

Egyszerű példa: lineáris szeparáció



Melyik a legjobb szeparáció?



A tempalte tervezés lépései

1, A feladat globális leírása

2,A lokális szabályok és formulák megállíptása

3, A legegyszerűbb template formula meghatározása

4, Igazságtáblázat elkészítése

4+5, Driving-point plot

5, Az egyenlőtlenségrendszer megoldása

6, Template optimalizáció