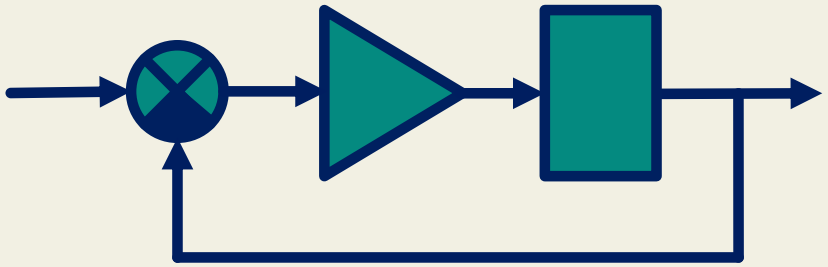
Szimulációs technológiák

NGB_IN040_1



- Alapfogalmak
- Példák:
  - » Példa1 Szoba hűlése, Euler módszerrel
  - » Példa2 Egytömegű lengőrendszer
  - » Példa3 RC kör
- Átviteli függvény
- Frekvenciatartomány
- Állapotgép

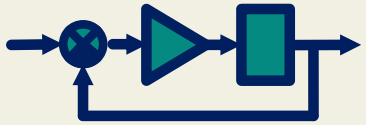
Elérhetőségek

Dr. ing. Claudiu Pozna
http://www.sze.hu/~pozna/

Horváth Ernő
http://www.sze.hu/~herno/

Tanszéki honlap
http://it.sze.hu

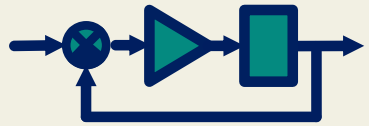
RS1.SZE.HU/~HERNO/



Arbejdsglæde

Danish for "work happiness, the feeling of happiness provoked by a satisfying job"

A feladat



Input

$$\dot{T} = \frac{\alpha(T_k - T)}{C}$$

$$T_0 = 32\ C^o$$

$$\alpha = 0{,}07$$

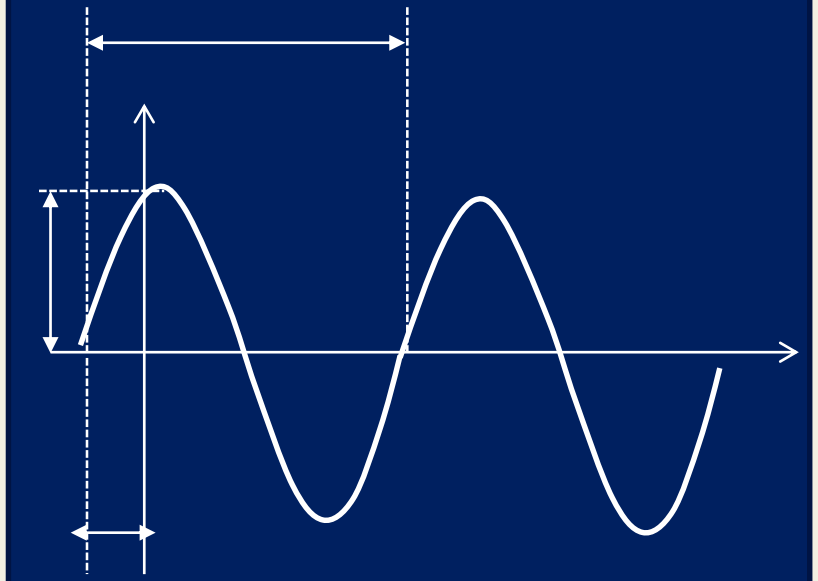
Runge-Kutta

f

Math

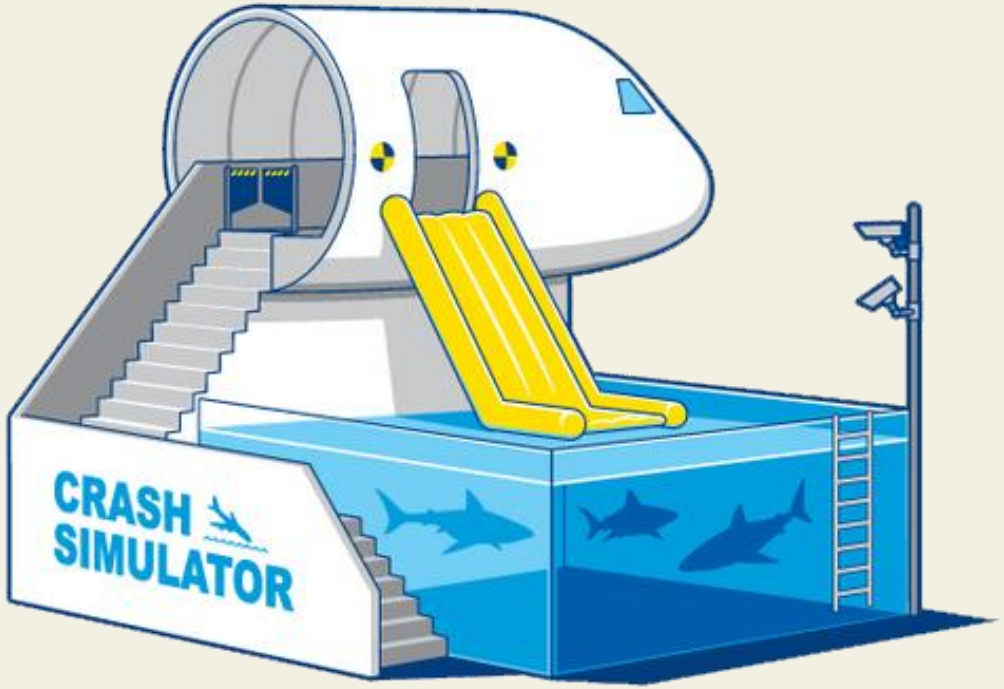
```
void numIntegralNeLap(double *x, double *y, int aSize, int *laps){
  int i, j, maxLap = 0;
  double *results;
  // az tolsó kör számának megállapítása
  for(i = 0; i < aSize - 1; i++)
    if(maxLap <= laps[i])
      maxLap = laps[i];
  // memória allokálása a körönként eredményeknek
  results = (double*)malloc(maxLap * sizeof(double));
  // az eredmények tömb nullázása
  for(j = 0; j < maxLap; j++) results[j] = 0;
  for(i = 0; i < aSize - 1; i++){
    for(j = 0; j <= maxLap; j++)
      if(laps[i] == j){
        results[j] += ((y[i+1] - y[i]) * (x[i] + x[i+1]) / 2.0);
      }
  }
  for(j = 1; j <= maxLap; j++)
    printf("%3d. kor: %7.1f meter hosszu\n", j, results[j]);

}
```

Output



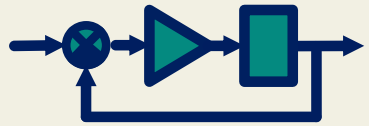
f

Mi is az a szimuláció?



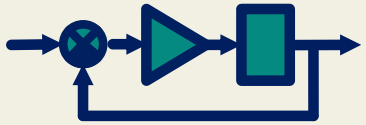
Source: Glennz

Szimuláció



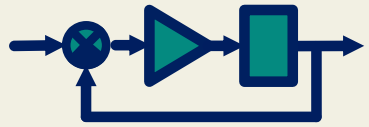
- Legegyszerűbb megfogalmazás: modellen végzett kísérlet.
- A modell esetünkben a valóság valamilyen szempontból absztrakt leképezése, leegyszerűsítése » számítási feladatok elvégzésének ideje

Modellezés



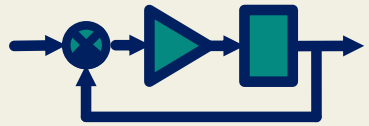
1. Problémafelvetés (megfogalmazás)
2. A probléma absztrakciója
3. A modell matematikai megfogalmazása
4. A modell matematikai alkalmazása
5. Kapott eredmények és a valóság összevetése

Matematikai alapok



- A **differenciálegyenletek** olyan egyenletek, melyekben az ismeretlen kifejezés egy differenciálható függvény.
  Az egyenletben az *ismeretlen függvény* különböző rendű deriváltjai szerepelnek, illetve egy- és többváltozós, de *ismert függvények*. Az ismeretlen függvény legmagasabb rendű deriváltjának a rendje » a differenciálegyenlet rendje.
- A problémák differenciálegyenletben való megfogalmazása a fizikában, mérnöki tudományokban, a közgazdaságtanban és még számos tudományban alapvető szerepet tölt be. [*wikipedia]

Matematikai alapok



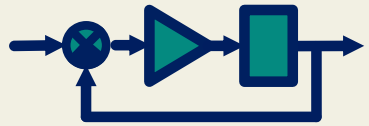
A **differenciálegyenletek** fajtái:

- Közönséges differenciálegyenlet (ODE) – a tárgyban főleg ezekkel foglakozunk, itt az egyenlet egy egyváltozós differenciálható függvényre van felírva
  - » n-edrendű differenciálegyenlet - a legmagasabb rendű derivált n-edrendű
  - » lineáris vagy nemlineáris
- Parciális differenciálegyenlet
- Algebro-differenciálegyenlet
- Integro-differenciálegyenletek

$$\frac{dy(t)}{dt} = f(t, y(t))$$

$$y(t_0) = y_0$$

Matematikai alapok



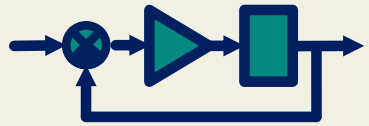
$F : (a, b) \times \boldsymbol{R^2} \rightarrow \boldsymbol{R}$
$f : (a, b) \times \boldsymbol{R} \rightarrow \boldsymbol{R}$

- Implicit alak: $F(x, y, y') = 0$ vagy $F(x, y, Dy) = 0$

- Explicit alak: $y(x)' = f(x, y(x))$ vagy $\frac{dy(x)}{dx} = f\big(x, y(x)\big)$

Lagrange jelölése y' , Leibniz jelölése: $\frac{dy}{dx}$, Newton: $\dot{y}$ (idő szerint)

Matematikai alapok



Adott:

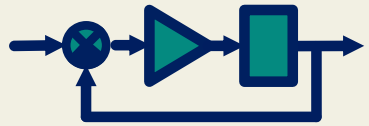
$$\frac{dy(t)}{dt} = f(t, y(t))$$

$y(t_0) = y_0$ (kezdeti érték)

Itt f adott függvény, amely kielégíti a Lipschitz-feltételt

Az differenciál egyenlet megoldása az $y(T)$, amely kielégíti a kezdeti feltételt is.

Eluler módszer - bevezetés



$$\frac{dy(t)}{dt} = f(t, y(t))$$

$y(t_0) = y_0$ (kezdeti érték)

$$\int_0^T \frac{dy(t)}{dt} = \int_0^T f(t, y(t))$$

- Integráljunk 0-tól T-ig

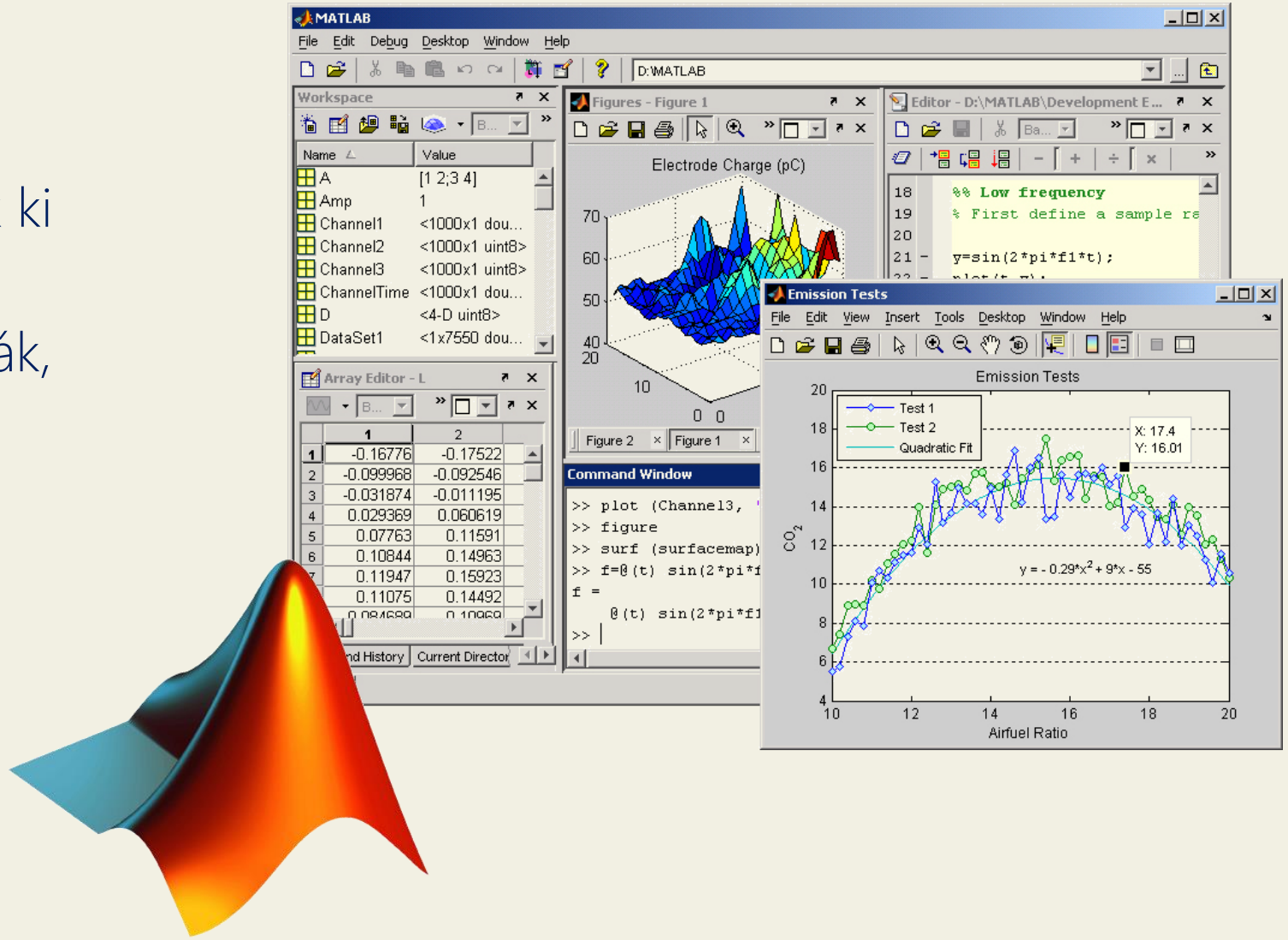
$$y(T) - y_{t_0} = \int_0^T f(t, y(t))$$

- Bontsuk véges Δt összegekre a t_0-tól T-ig terjedő intervallumot.

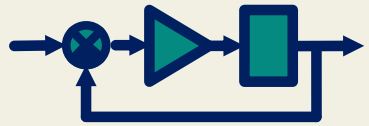
$$y(T) = y_{t_0} + \sum_{t=t_0}^{T} f(t, y(t))\Delta t$$

Alapok

- Mi a MATLAB?
- Mi a Simulink?
- Hogyan és miért alakultak ki és mire használhatók?
- Alapvető szintaktika, példák, alkalmazás stb.



MATLAB



A MATLAB egy speciális programrendszer, amely numerikus számítások elvégzésére lett kifejlesztve és emellett egy programozási nyelv. A The MathWorks által kifejlesztett programrendszer képes mátrix számítások elvégzésére, függvények és adatok ábrázolására, algoritmusok implementációjára és felhasználói interfészek kialakítására. Habár a szoftver kizárólag numerikus, a MuPAD csomag hozzáadásával képes matematikai kifejezéseket grafikusan is megjeleníteni.

2004-ben, több, mint 1 millió MATLAB felhasználó

[* wikipedia]

MATrix LABoratory

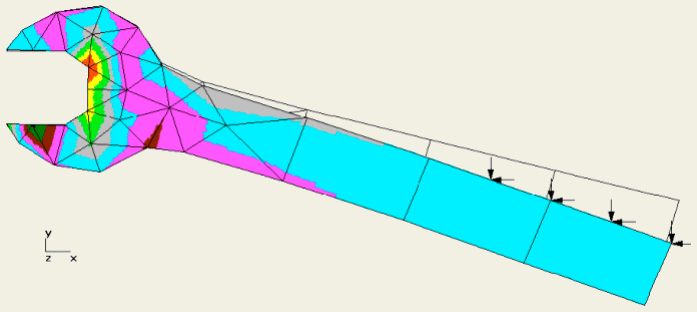
- Miért a mátrixok és a lineáris algebra?
- 1970-es évek, University of New Mexico
- Interpreteres interfész a LINPACK-hoz
- Lineáris egyenletrendszer, közös HPC gyökerek

1.
$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ \vdots \quad & \quad \vdots \quad \quad \vdots \quad \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &= b_m \end{aligned}$$

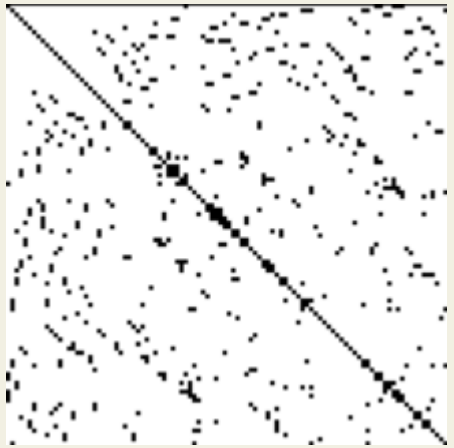
2.
$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

3. $A\mathbf{x} = \mathbf{b}$

4. $\mathbf{x} = A^{-1}\mathbf{b}$



- „merevségi mátrix”
- ritka mátrix
- hatékony megoldáshoz sokféle lineáris algebrai eszköz kell:
  - alapműveletek
  - egyes elemek, sorok, oszlopok manipulációja, stb.
  - invertálás
  - transzponálás
  - determináns
  - dekompozíciók
  - sajátérték-számítás
  - stb.



Alapvető szintaktika

- (G)UI
- alapműveletek
- adattípusok
- változók, workspace
- műveletek vektorokkal és mátrixokkal
  - indexek 1-től!
  - referencia
  - sorok, oszlopok, stb.
  - elemenkénti műveletvégzés
- megjelenítés
- függvények meghívása
- a szintaktika robusztussága „helyi dereferencia”, óriásképletek
- többféle megoldás ugyanarra a problémára (pl. size(A,1))
- IDE, m-fájlok
- programnyelvi elemek
  - if
  - for
  - függvények
  - stb.
- hibák értelmezése, debuging

Kitekintés

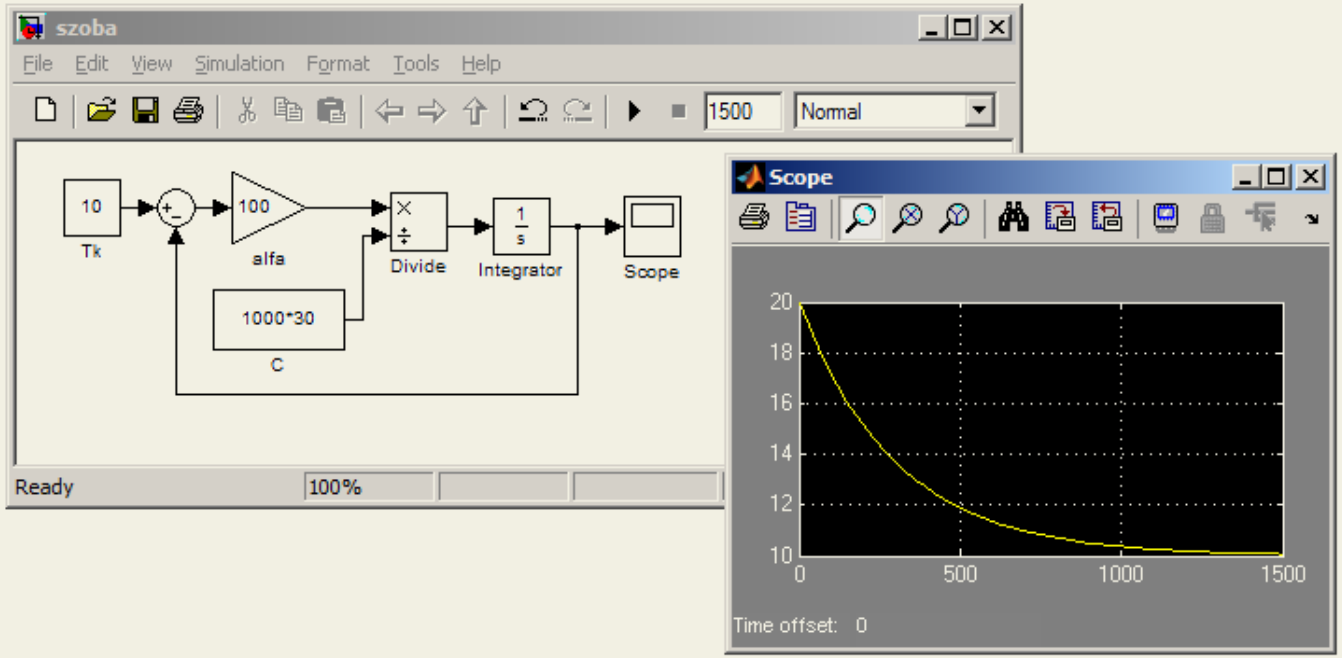
- Csak a felszínt kapargattuk, mivel a MATLAB moduláris (ld. később)
- Java alapú, de tartalmaz(hat) natív rendszerspecifikus komponenseket is (ezeket jellemzően C-ben írták)
- Ma már nem igaz, hogy a MATLAB csak interpreteres programnyelvként értelmezhető (ha programnyelvként akarjuk értelmezni), lefordítható
- Annak ellenére, hogy a gyökerek közösek (LINPACK), a közelmúltig nem bizonyult hatékonynak HPC feladatokra. Ez az utóbbi időben változott:
  - a lineáris algebrai „motort” optimalizálták, ill. részben visszatértek a teljesítményorientált natív implementációkhoz (LINPACK, és egyéb „leszármazottai:” LAPACK, stb.)
  - megjelentek a párhuzamosítást szolgáló megoldások
  - ezekkel együtt sem lett kimondottan HPC platform, de egyre nagyobb méretű problémákat lehet hatékonyan megoldani vele

Mi a Simulink?

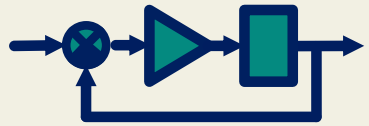
- a Simulink a MATLAB egyik toolbox-a a (sok közül)
- grafikus programozási nyelv (leginkább) fizikai rendszerek tranziens szimulációjához
- mi az, hogy tranziens fizikai rendszer
  - állandósult állapot: algebrai egyenlet, pl.: $0 = \frac{\alpha(T_k - T)}{C}$
  - időben változó állapot (tranziens): differenciálegyenlet, pl.: $\dot{T} = \frac{\alpha(T_k - T)}{C}$
- megoldás: integrálással (numerikusan!), pl.: Euler-módszer:

$$\boldsymbol{T_0}, \quad T_1 = T_0 + \frac{\alpha(T_k - T_0)}{C}\Delta t, \quad T_2 = T_1 + \frac{\alpha(T_k - T_1)}{C}\Delta t, \ldots \text{stb.}$$

- miért így néz ki? (analóg számítógép)



Példa 1: Szoba hűlése



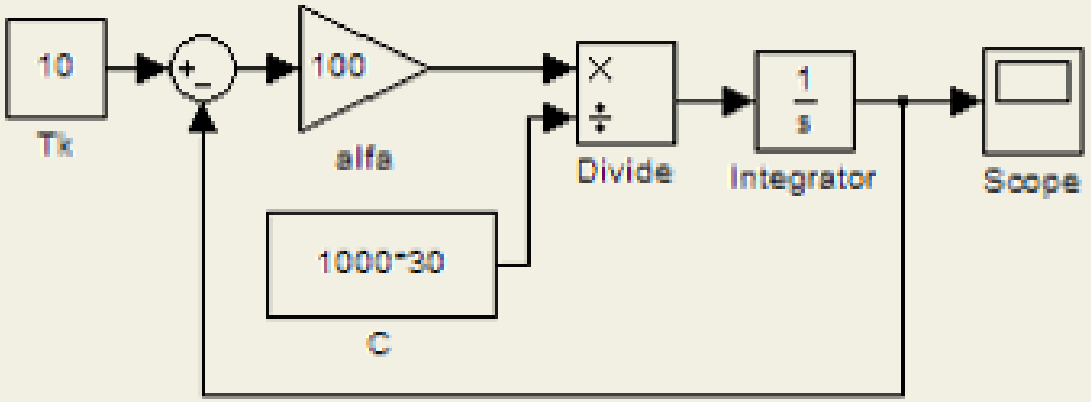
$$\dot{T} = \frac{\alpha(T_k - T)}{C}$$

- T_k- a környezet hőmérséklete
- C - hőkapacitás [J/C°]
- α - hőátadás
- Euler módszer, numerikus integrálás:

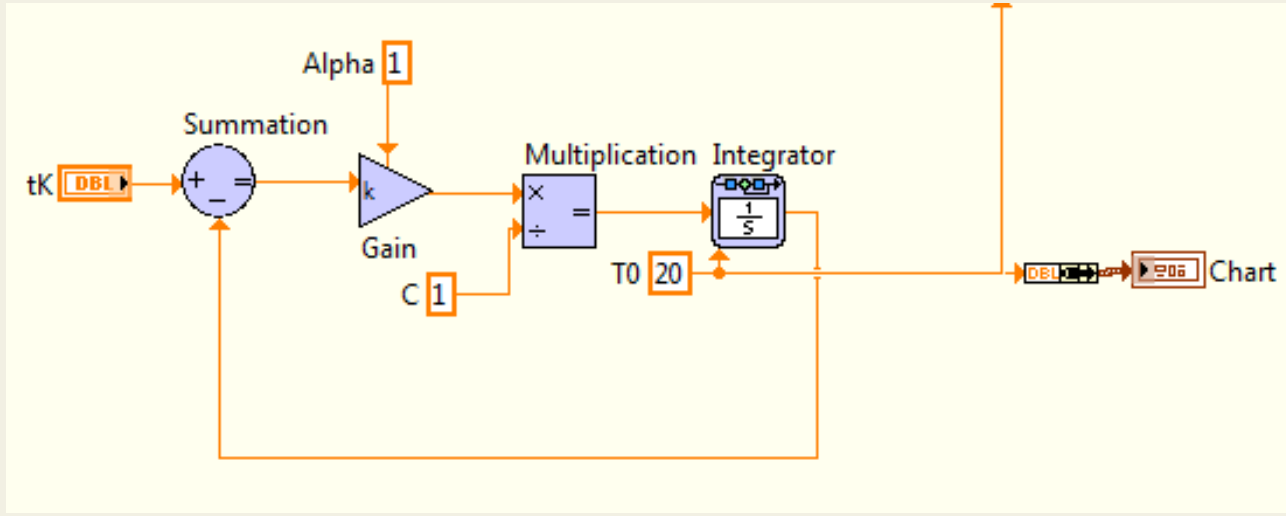
$$T_0, \qquad T_1 = T_0 + \frac{\alpha(T_k - T_0)}{C}\Delta t, \qquad T_2 = T_1 + \frac{\alpha(T_k - T_1)}{C}\Delta t, \ldots \text{stb.}$$

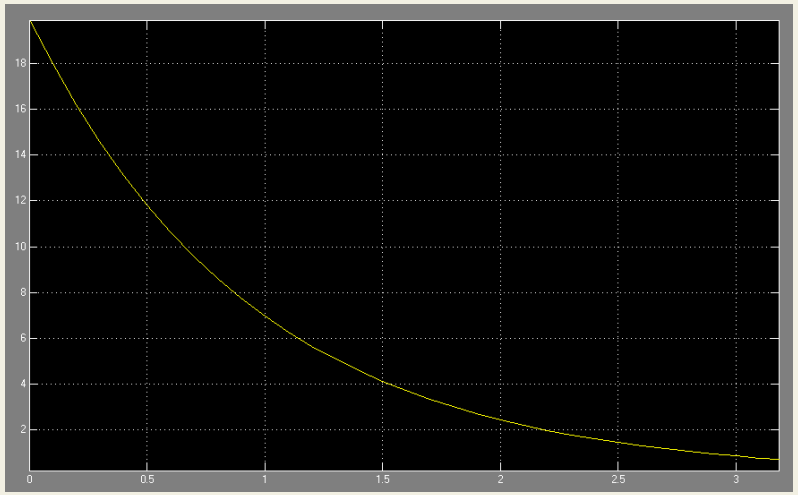
Példa 1: Szoba hűlése

MATLAB implementáció

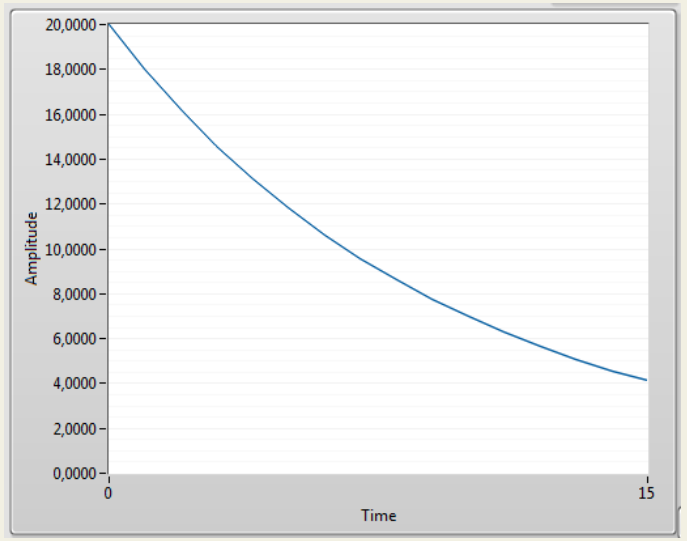


LabVIEW implementáció

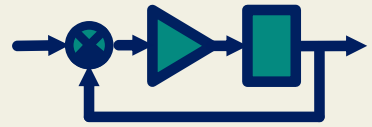




$$\dot{T} = \frac{\alpha(T_k - T)}{C}$$

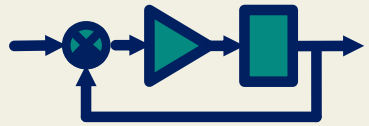


Közönséges differenciálegyenlet



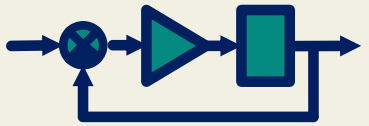
- Angolul: ordinary differential equation, ODE
- Ebben az esetben az egyenlet egy egyváltozós differenciálható függvényre van felírva.
- Az n-ed rendű közönséges **differenciálegyenlet általános megoldása** az a függvény, mely pontosan n számú egymástól független állandót (paramétert) tartalmaz, és azonosan kielégíti az adott differenciálegyenletet.
- A **numerikus integrálás** közelítő eljárás az integrál kiszámítására.

Runge-Kutta



- Egy függvény integráljának a közelítése a kvadratúra, a kvadratúra szabályok többsége az interpolált függvényből származtatható.
- A legegyszerűbb módszer a téglalap-szabály. A téglalap-szabály alkalmazásakor az integrálandó függvényt téglalapokkal közelítjük. Az interpolációs függvény konstans, és keresztül megy a keresett függvény egy-egy pontjain $((a+b)/2, f((a+b)/2))$.
- A Runge-Kutta módszerek numerikus megoldást adnak az közönséges differenciálegyenletekre. Legegyszerűbb válfaja az elsőrendű Euler módszer.

Euler módszer



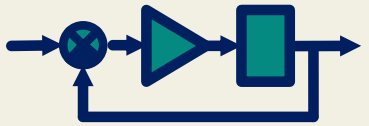
Az Euler-módszer a legegyszerűbb módszer differenciálegyenlet-rendszerek numerikus megoldására. Adott a következő:

$$\frac{dy(t)}{dt} = f(t, y(t)) \qquad y(t_0) = y_0$$

Bontsuk véges Δt összegekre a t_0-tól T-ig terjedő intervallumot.

$$y(T) = y_0 + \sum_{t=t_0}^{T} f(t, y(t))\Delta t$$

Euler módszer



Az Euler-módszer a legegyszerűbb módszer differenciálegyenlet-rendszerek numerikus megoldására. Adott a következő:

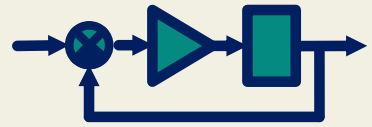
$$\frac{dy(t)}{dt} = f(t, y(t)) \qquad y(t_0) = y_0$$

Átalakítások után a képlet:

$$y(t + \Delta t) = y(t) + f(t, y(t))\Delta t$$

T_0, $T_1 = T_0 - k(T_0 - T_k)\Delta t$, $T_2 = T_1 - k(T_1 - T_k)\Delta t$, ... stb.

Euler módszer pszeudo kódja



T_0, $T_1 = T_0 - k(T_0 - T_k)\Delta t$, $T_2 = T_1 - k(T_1 - T_k)\Delta t$, ... stb.

Ismert paraméterek:

Δt: egy időlépés hossza

`f(t,y): y(t)` ismeretlen függvény deriváltja `t` időpillanatban

t_0: kezdeti időpillanat (gyakorlatban célszerű 0-át választani)

y_0: kezdeti érték a kezdeti időpillanatban

`T:` végső időpillanat

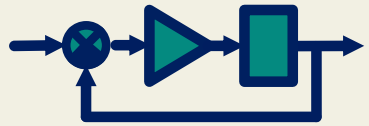
Inicializálás:

```
y := y0
t := t0
```

Eljárás:

```
Ciklus amíg t < T
  y := y + f(t,y) * Δt
  t := t + Δt
Ciklus vége
```

Szoba hűlése példa



$$\frac{dT(t)}{dt} = \frac{\alpha(T_k - T(t))}{C}$$

$$k = \frac{\alpha}{C}$$

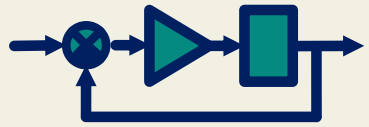
$$\frac{dT(t)}{dt} = -k(T(t) - T_k)$$

- T_0=**100 C°** kiindulási hőmérséklet
- T_K=**20 C°** környezeti hőmérséklet
- k=**0,07** hűlési konstans (hőkapacitás/hőátadás)
- A szimuláció **0s** és **60s** időköz alatt zajlik

A referencia számítása:

$$T(t) = T_k + (T_0 - T_k)e^{-kt}$$

Szoba hűlése példa



$$T_0, \quad T_1 = T_0 - k(T_0 - T_k)\Delta t, \quad T_2 = T_1 - k(T_1 - T_k)\Delta t, \ \ldots \text{stb.}$$

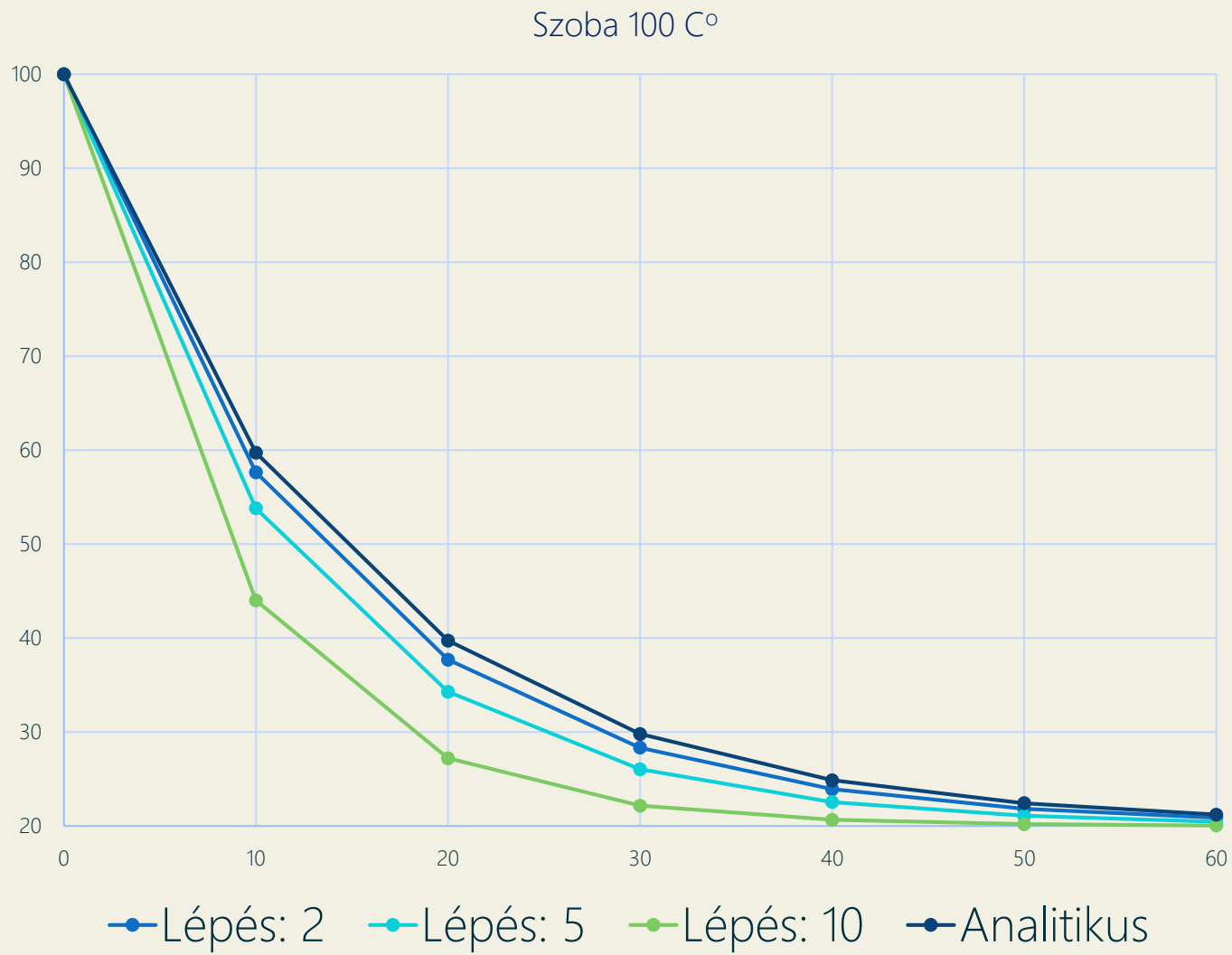
```
#define IDOKEZD 0    // idő kezdete
#define IDOVEG 60    // idő vége
#define KIINDHOM 100    // T0=100 C° kiindulási hőmérséklet
#define KORNYHOM 20    // T0=20 C° környzeti hőmérséklet
#define K -0.07    // k=0,07  hűlési konstans
```

```
void ivp_euler(deriv_fd f, double k, int lepes, int idokezd, int idoveg){
  int t = idokezd;
   printf("Lepes %2d: ", (int)lepes);
  do {
      if (t % 10 == 0)
      printf(FMT, k);
           k += lepes * f(t, k);
  } while ((t += lepes) <= idoveg);
  printf("\n");
}
```

Szoba hűlése példa

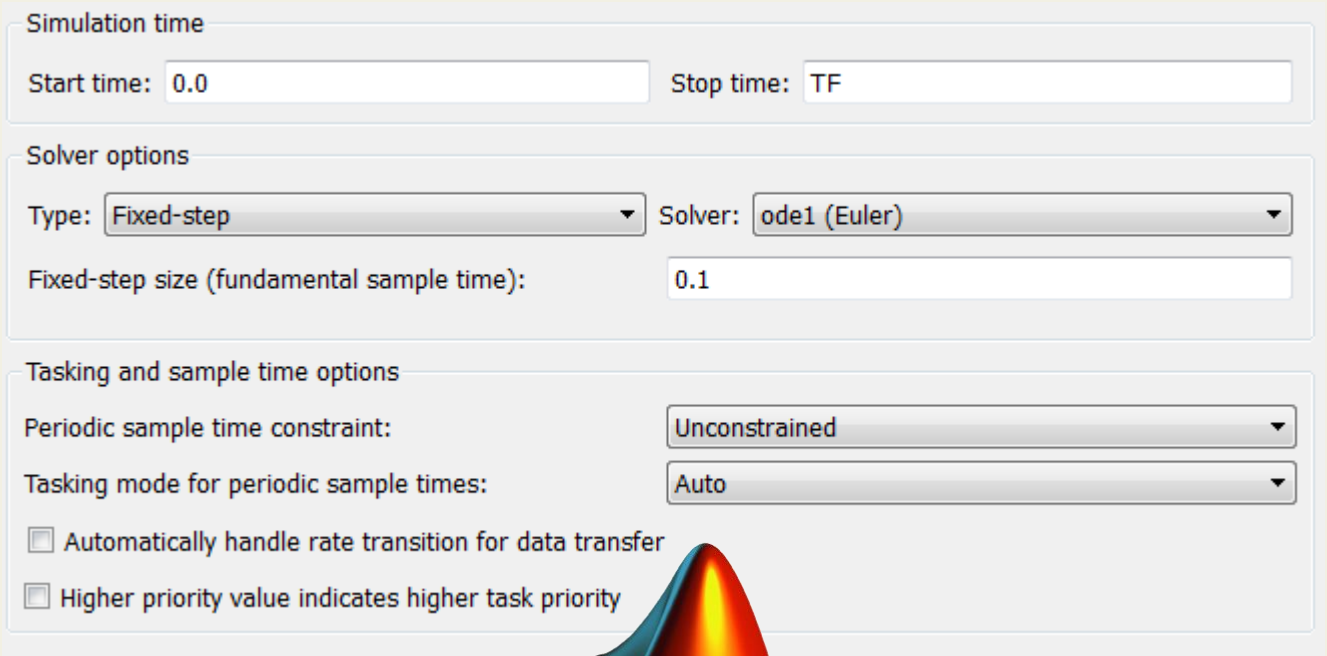
Minél nagyobb a lépésköz annál pontatlanabb az eredmény. Persze a futási idő nő.

$$\frac{dT(t)}{dt} = -k(T(t) - T_k)$$



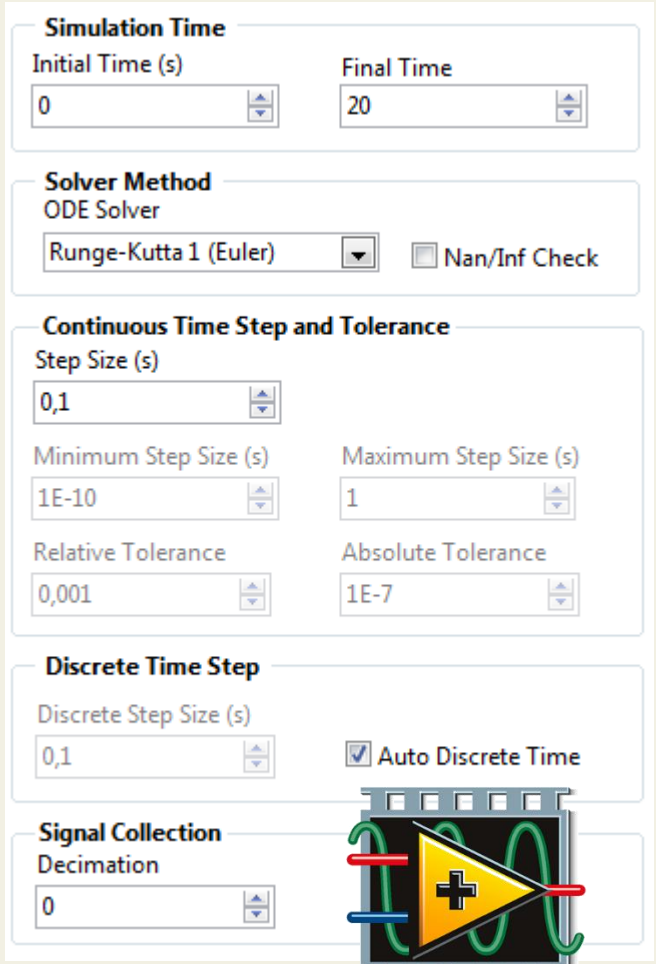
Példa 1: Szoba hűlése

MATLAB implementáció



$$\dot{T} = \frac{\alpha(T_k - T)}{C}$$

LabVIEW implementáció



Példa 2: egytömegű lengőrendszer

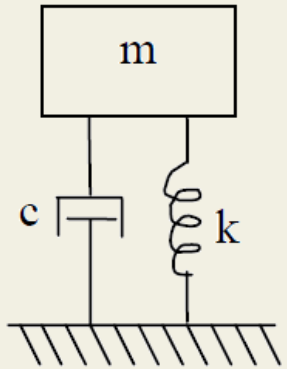
A modellezés 4 lépése:

- 1. absztrakció
- 2. matematikai megfogalmazás
- 3. Simulink-implementáció
- 4. ellenőrizzük, hogy nem számoltunk-e hülyeséget (validáció)

Valóság

1.

Absztrakt fizikai modell

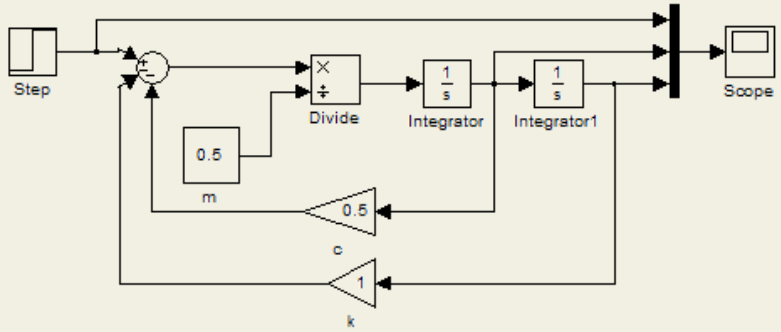


2.

Matematikai megfogalmazás

$$\ddot{x} = \frac{1}{m}\left[f(t) - c\dot{x} - kx\right]$$

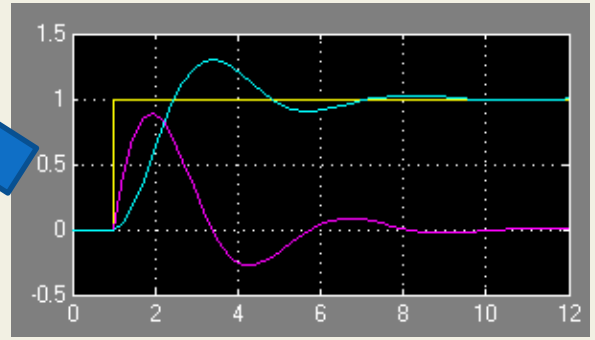
3.



Simulink implementáció

4.

?

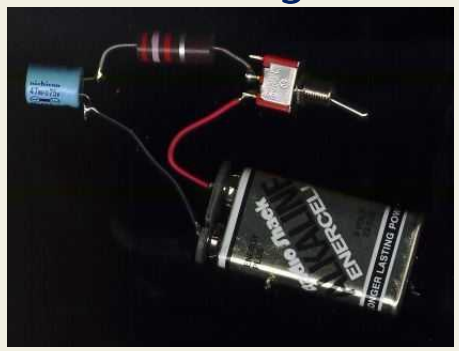


Példa 3: RC-kör

A modellezés 4 lépése:

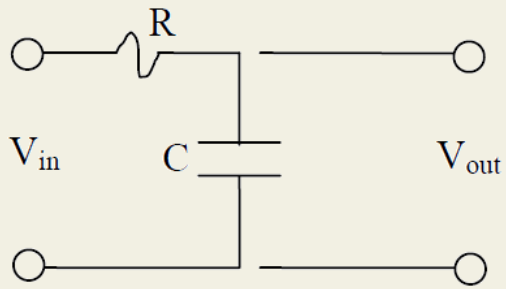
- 1. absztrakció
- 2. matematikai megfogalmazás
- 3. Simulink-implementáció
- 4. ellenőrizzük, hogy nem számoltunk-e hülyeséget (validáció)

Valóság



1.

Absztrakt fizikai modell

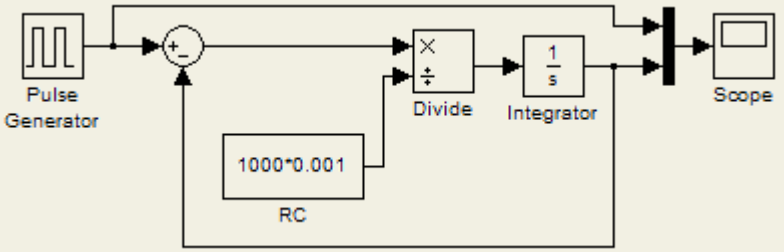


2.

Matematikai megfogalmazás

$$\dot{x} = \frac{1}{RC}\left[f(t) - x\right]$$

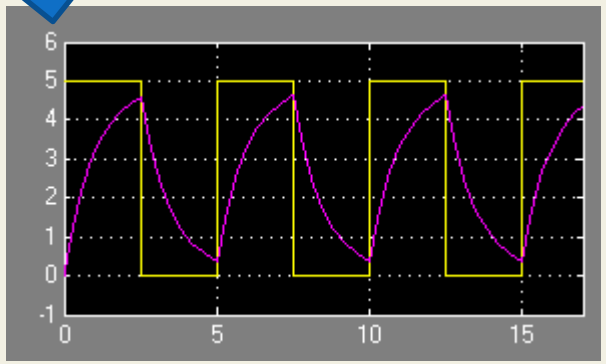
3.



Simulink implementáció

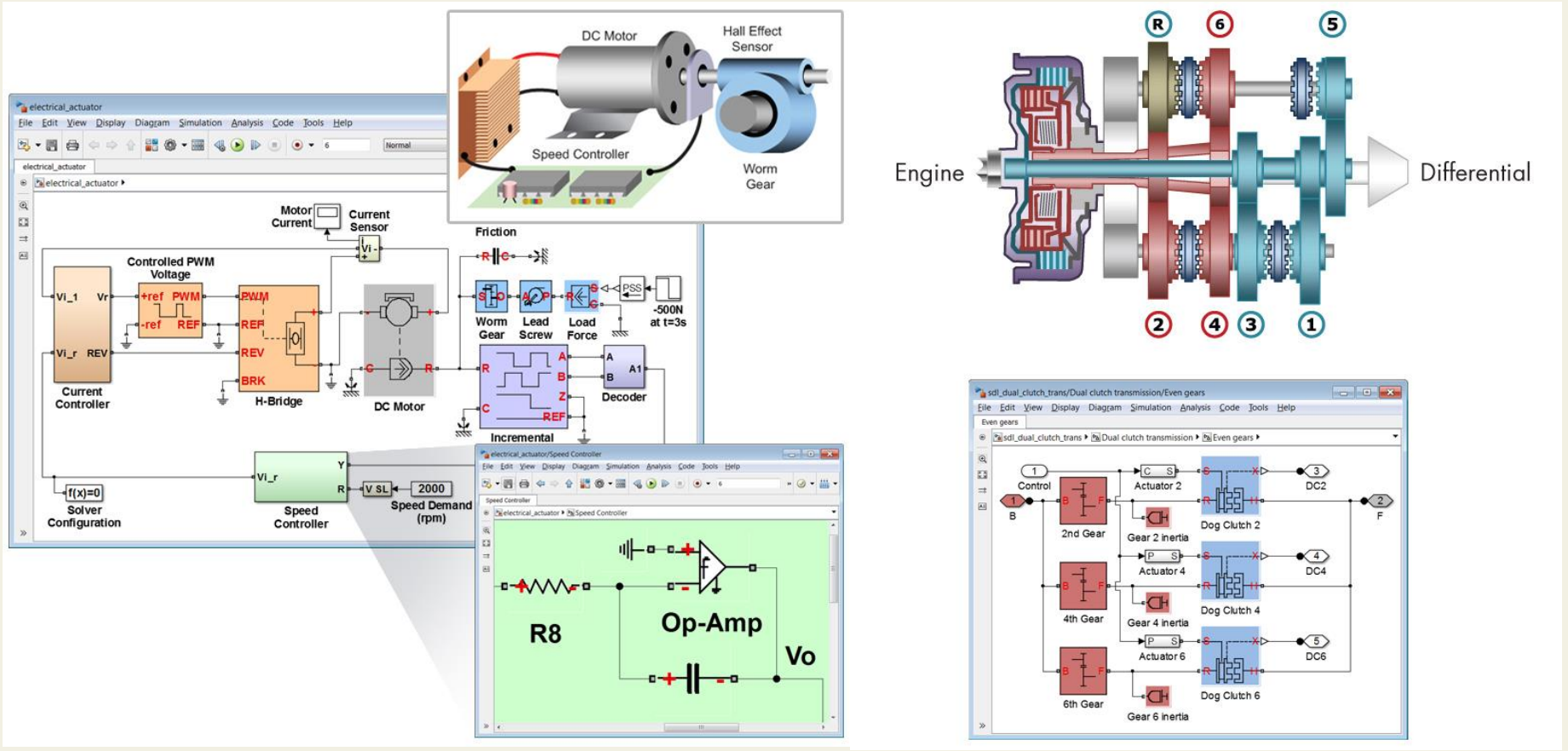
?

4.



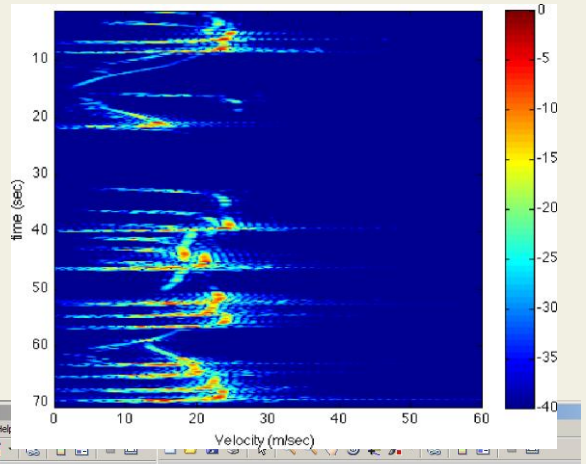
Kitekintés

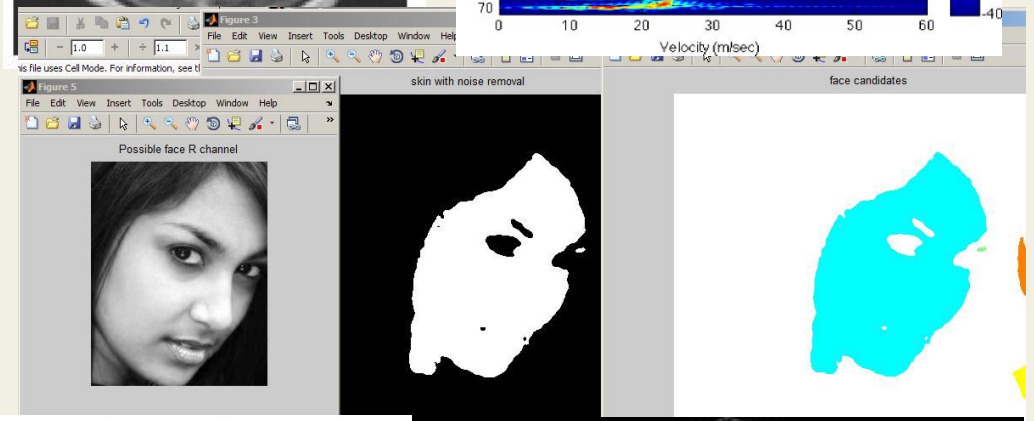
- nemlineáris modellek is készíthetők
- figyelembe vehetők a különböző adattípusok
- megvalósíthatók a hagyományos programszerkezetek (vö.: programozási nyelv!)
- kapcsolat… sokmindennel (ld. következő órák)
- a Simulink önmagában is moduláris
- tematikus, magas(abb absztrakciós) szintű toolboxok, modellek: SimElectronics, SimDriveline, stb.

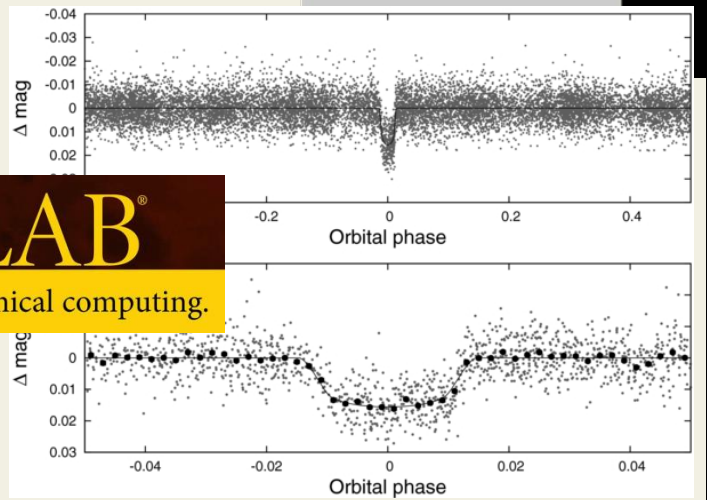


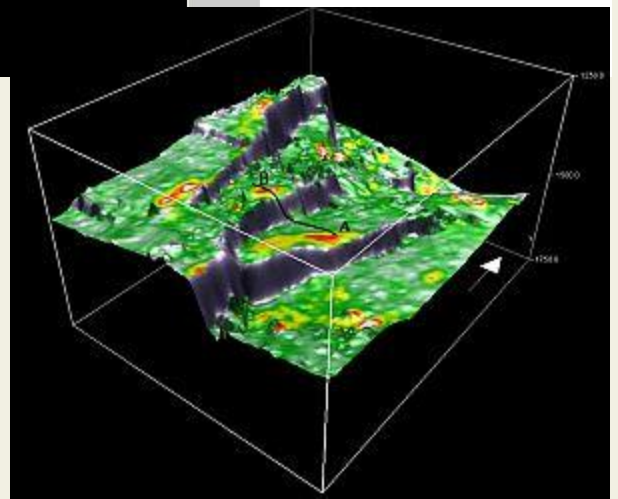
Alkalmazás - Mire használják a „többiek” a MATLABot?

- orvosi képalkotás
- képfeldolgozás, alakfelismerés
- mesterséges intelligencia
- rádiófrekvenciás jelfeldolgozás
- GIS
- csillagászat
- szoftverfejlesztés
- adatbányászat
- stb.
- és a felsorolt módszerek fejlesztésére! (ld. még: „balansz!”)



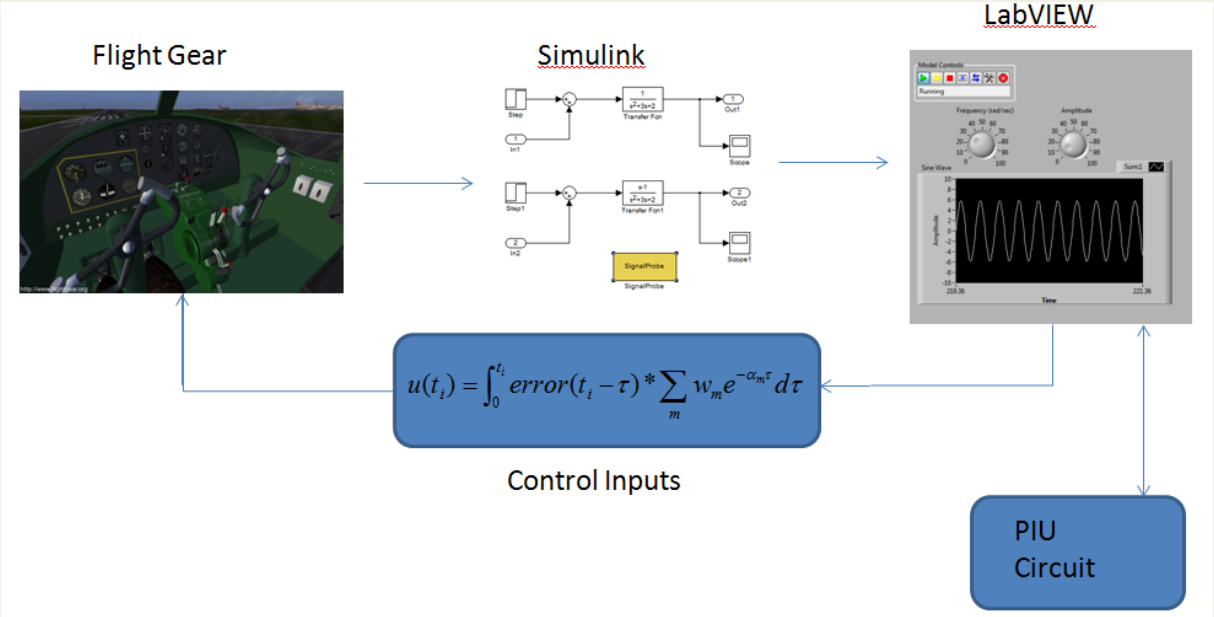


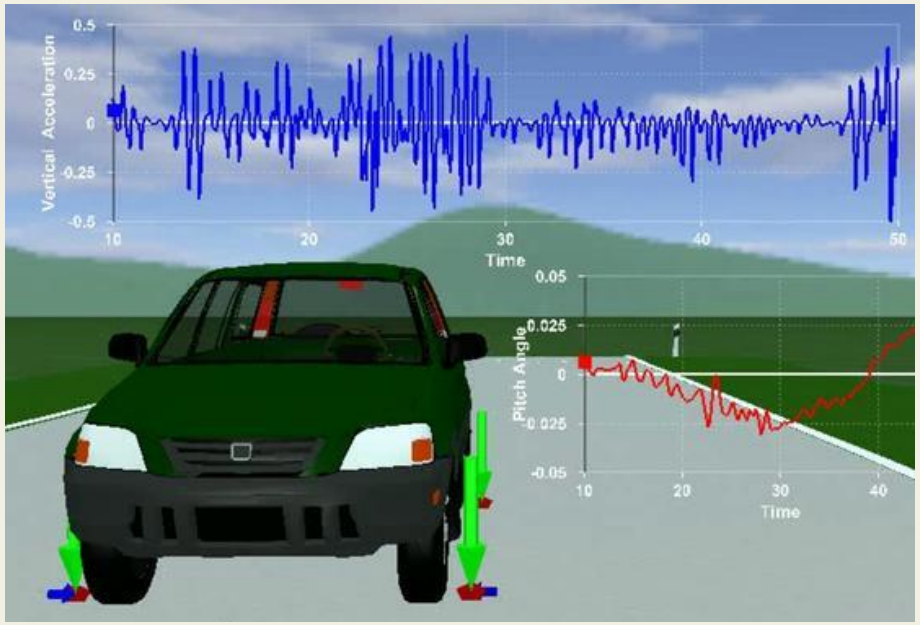




Mire használjuk „mi”?

- Tetszőleges (jellemzően inkább koncentrált paraméterű) fizikai rendszer modellezése a mérnöki gyakorlatban.
  A járműiparban jellemzően:
  - járműdinamikai modellezés
  - járműalrendszerek, járműmechatronikai rendszerek modellezése
- Irányítástechnika (!!!)
  - irányításelméleti kutatások
  - szabályzók, irányítási algoritmusok fejlesztése és együttes szimulációja a fizikai modellel (modellalapú fejlesztési folyamat)
  - (prototípus) irányítórendszerek és szoftverek tervezése és megvalósítása
- „Mindennapi”, tevékenységfüggetlen mérnöki feladatok:
  - mérési adatok feldolgozása
  - statisztikai vizsgálatok
  - rutinfeladatok automatizálása
  - stb.

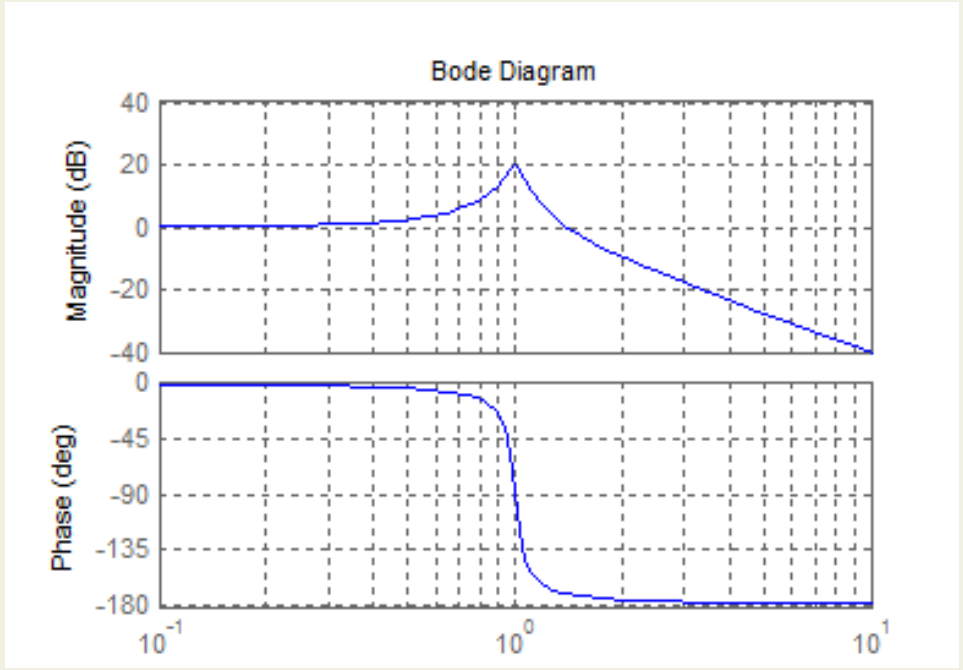




Irányítástechnikai alkalmazások

- teljes irányítástechnikai eszköztár
- átviteli függvények
- state-space
- diszkrét rendszerek
- szabályozótervezési algoritmusok
- irányítási rendszerek vizsgálata (stabilitás, frekvenciaátvitel, stb.)
- kapcsolódó vizualizációs módszerek
- stb.
- gyakorlatilag a teljes irányításelméleti vertikum megvalósítása rendelkezésre áll
- de facto szabvány (lehet, hogy van olyan probléma, amit nem ezzel oldanak meg, de biztos, hogy nincs olyan probléma, amit ezzel nem oldanak meg)

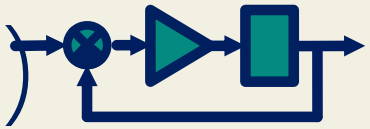
```
>> sysc=tf(1,[1 0.1 1])

Transfer function:
       1
---------------
s^2 + 0.1 s + 1

>> sysd=c2d(sysc,0.01)

Transfer function:
4.998e-005 z + 4.997e-005
-------------------------
  z^2 - 1.999 z + 0.999

Sampling time: 0.01
>>
>> bode(sysc)
```



Dokumentáció és „ökoszisztéma”

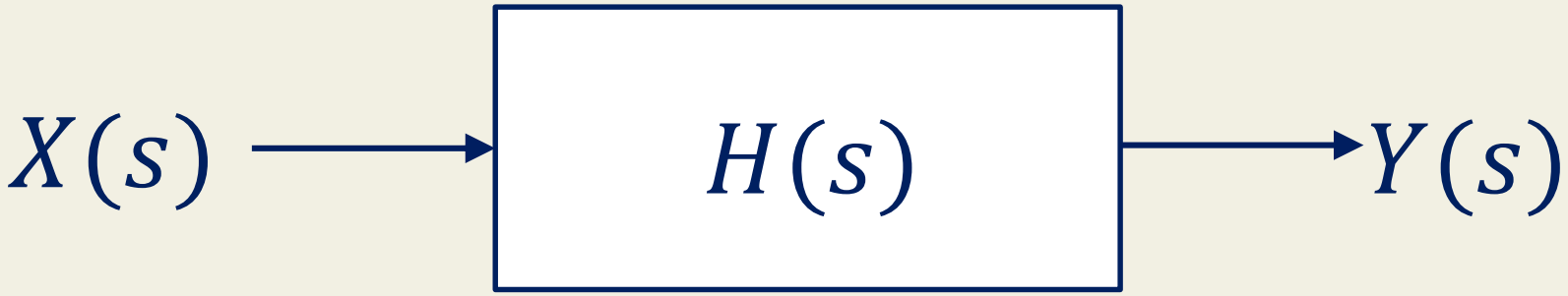
- nagyon jó help
  - monográfia szintű elvi leírások irodalmi hivatkozásokkal
  - példák, tutorialok
  - teljes referencia-kézikönyv
- több millió felhasználó (nagyon valószínű, hogy nem leszel egyedül a problémáiddal)
- „a katedrális és a bazár”, MathWorks File Exchange: http://www.mathworks.com/matlabcentral/fileexchange/
- fórumok
- kész megoldások, scriptek, modellek, gyakran publikációk mellékleteként
- nagyon jó interoperabilitás

Az átviteli függvény (transfer function)

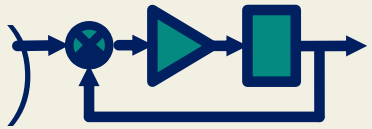


- Jelölése $\mathbf{H(s)}$, ritkábban $\mathbf{W(s)}$
- A rendszerek bemenete és kimenete közötti kapcsolat.

$$H(s) = \frac{Y(s)}{X(s)} = \frac{\mathcal{L}\{y(t)\}}{\mathcal{L}\{x(t)\}}$$



Az átviteli függvény (transfer function)



- Az általános képlet:

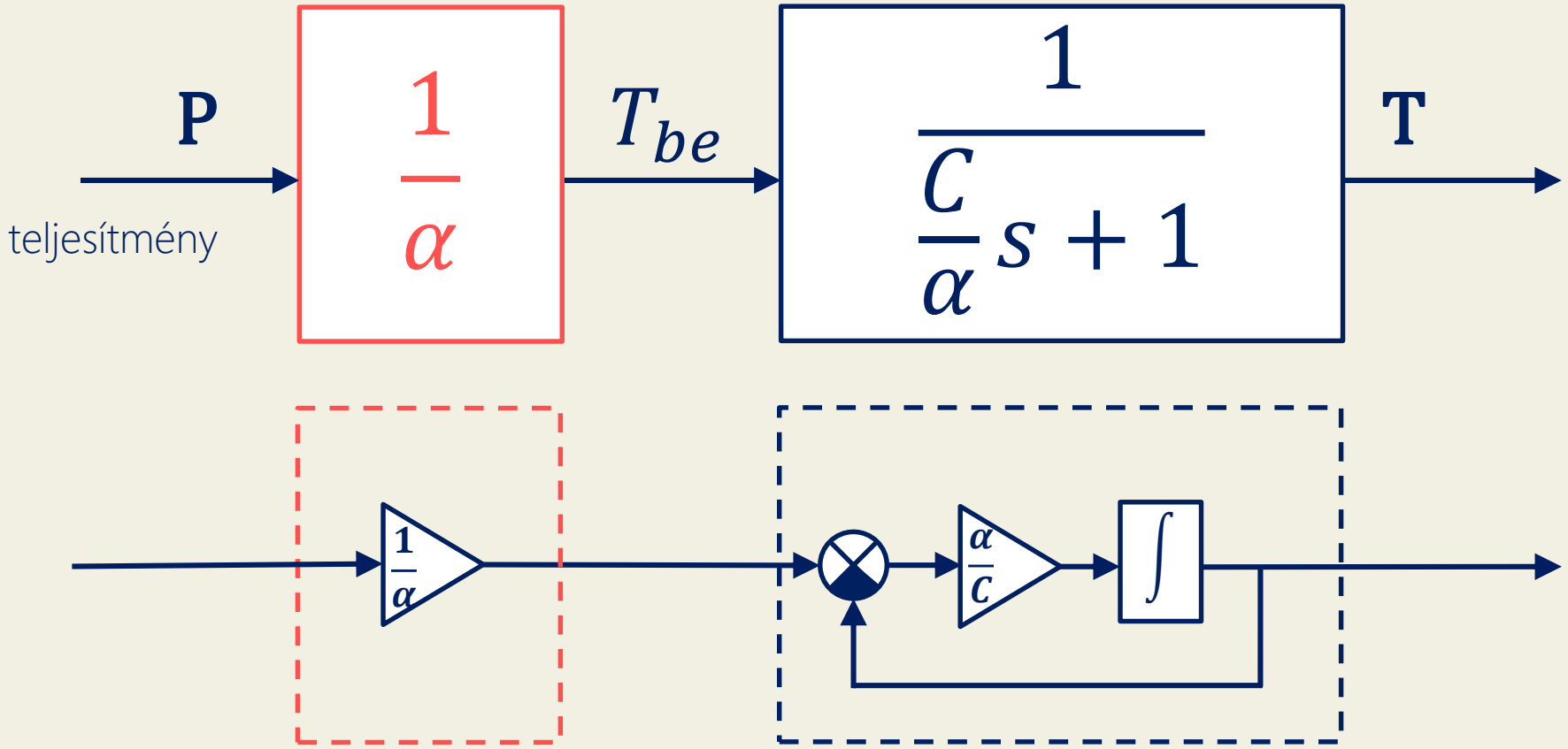
$$H(s) = \frac{Y(s)}{X(s)} = \frac{a_0 + a_1 s + a_2 s^2 \ldots + a_n s^n}{b_0 + b_1 s + b_2 s^2 \ldots + b_m s^m} \quad m \geq n$$

- Példák:

$$H_1(s) = \frac{s+1}{-2s^2 s + s - 3} \qquad H_2(s) = \frac{1}{\frac{C}{\alpha} s + 1}$$

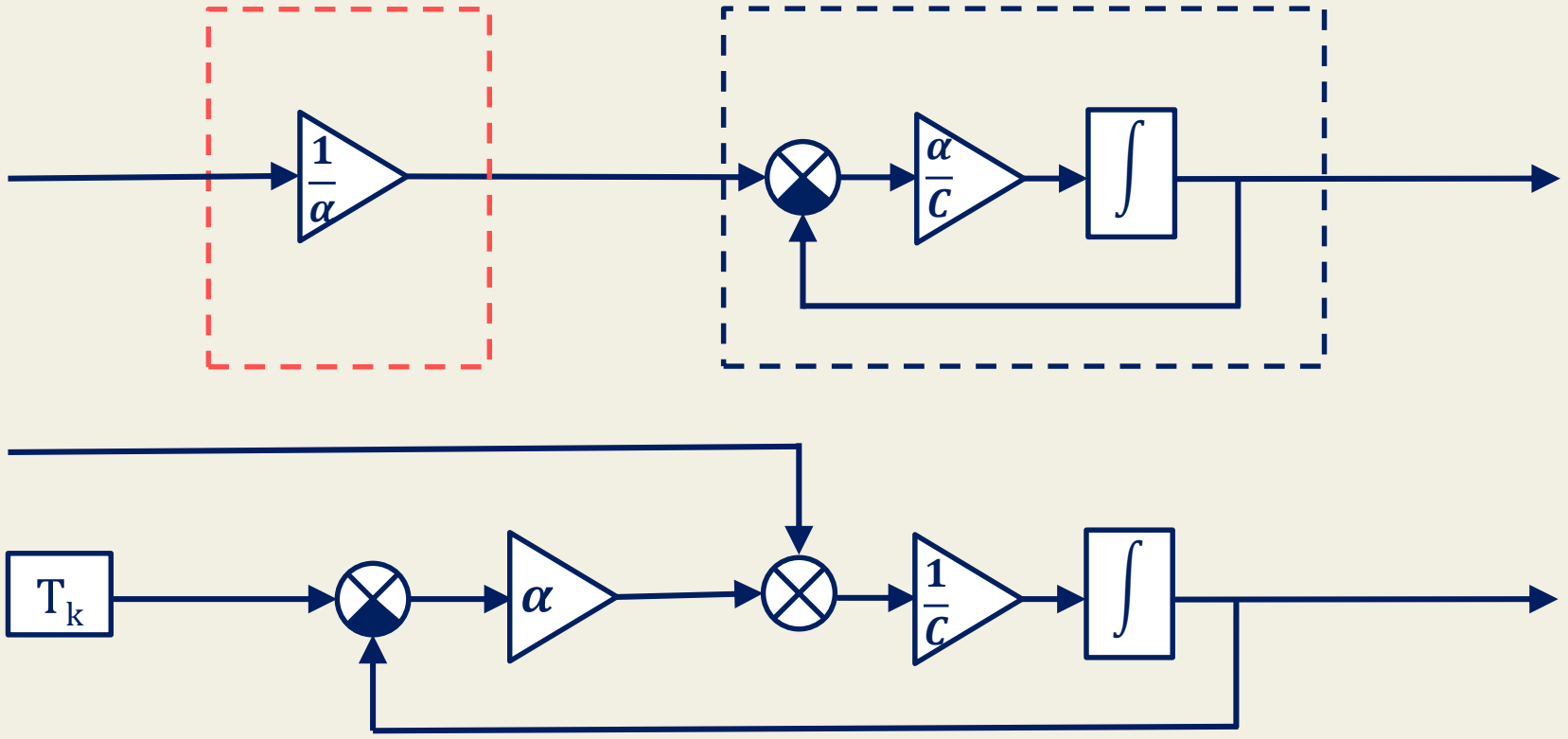
Az átviteli függvény (transfer function)

Vegyük a jól ismert szoba hűlése példát, de most szabályozzuk a hőmérsékletet is!

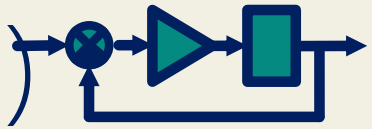


Az átviteli függvény (transfer function)

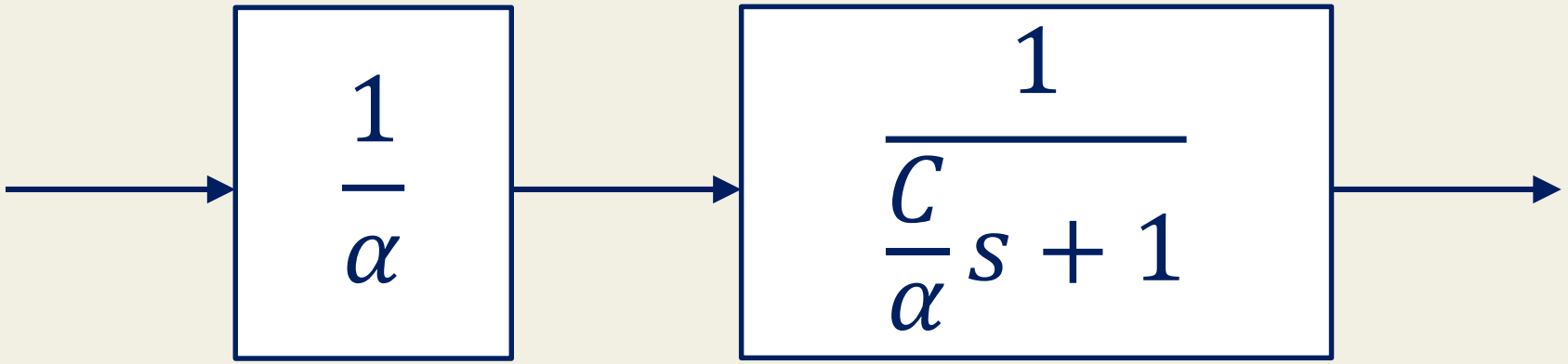
Vegyük a jól ismert szoba hűlése példát, de most szabályozzuk a hőmérsékletet is!

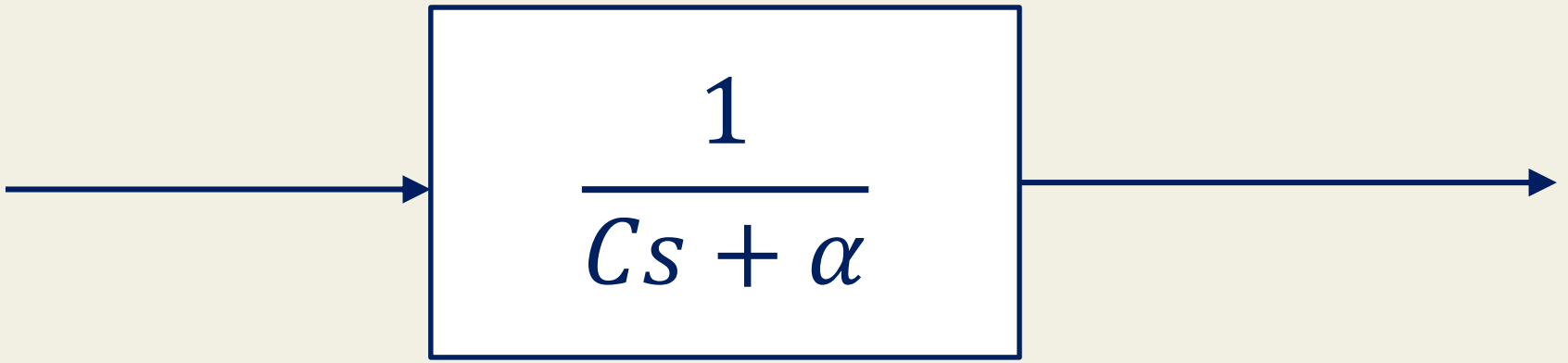


Az átviteli függvény (transfer function)

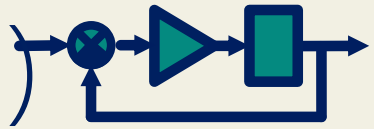


Vegyük a jól ismert szoba hűlése példát, de most szabályozzuk a hőmérsékletet is!

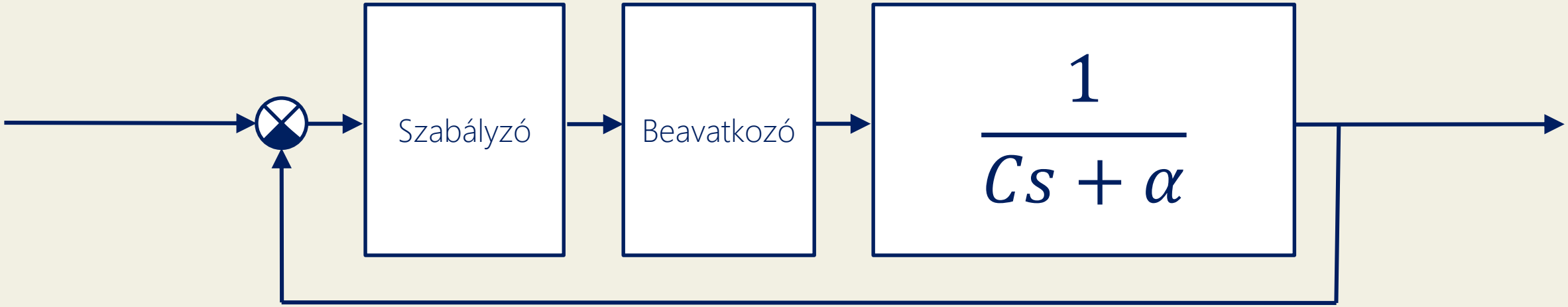




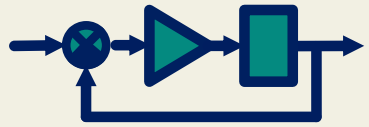
Az átviteli függvény (transfer function)



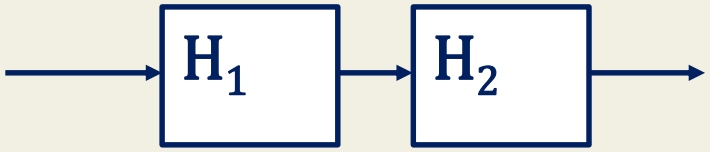
Vegyük a jól ismert szoba hűlése példát, de most szabályozzuk a hőmérsékletet is!



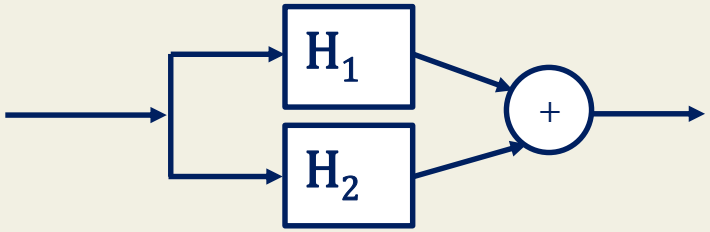
Simplifying block diagrams



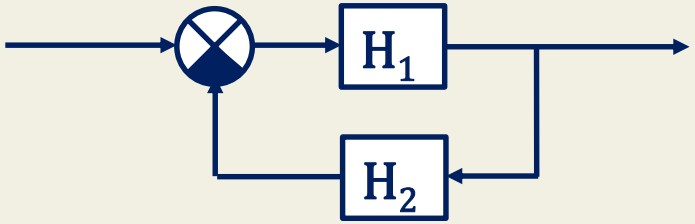
Hatásvázlat egyszerűsítés http://en.wikibooks.org/wiki/Control_Systems/Block_Diagrams



$$H = H_1 \cdot H_2$$

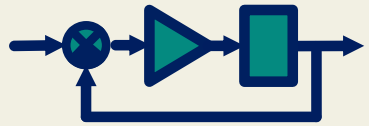


$$H = H_1 + H_2$$



$$H = \frac{H_1}{1 + H_1 H_2}$$

Diszkrét / Folytonos idő



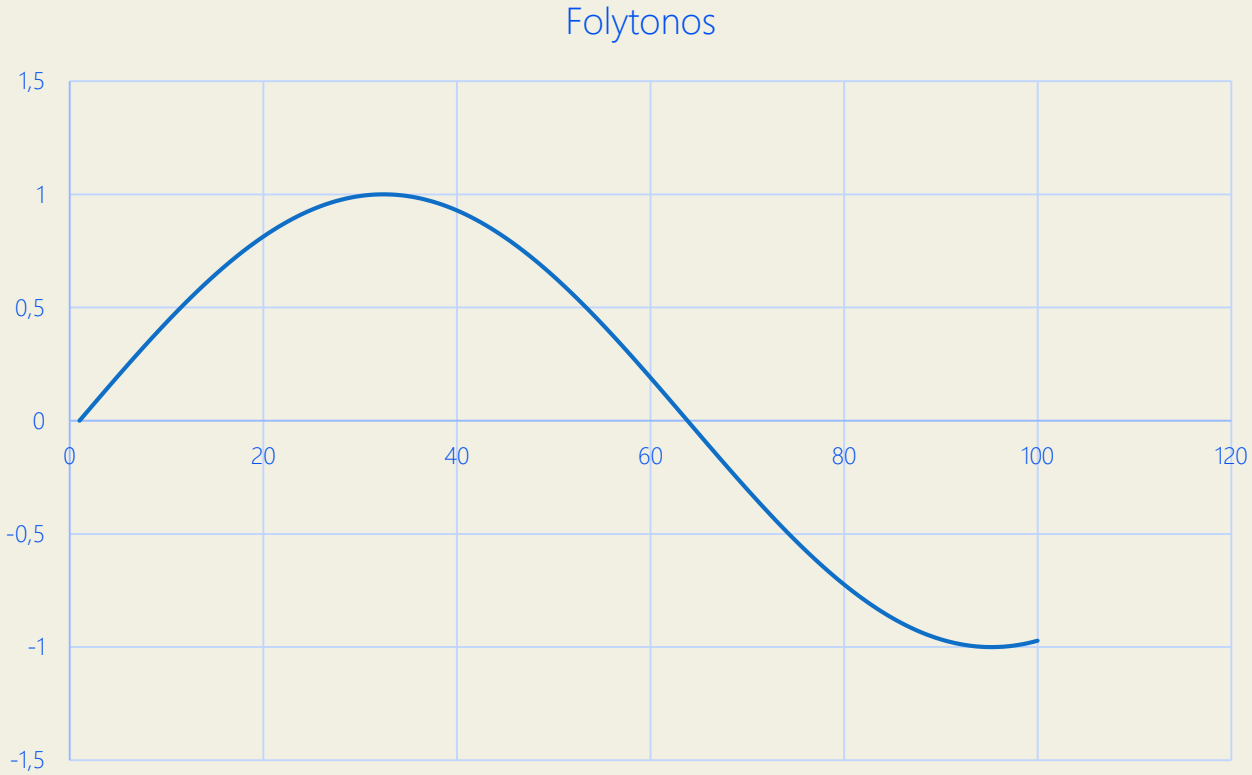
$$H(s) = \frac{Y(s)}{X(s)} = \frac{\mathcal{L}\{y(t)\}}{\mathcal{L}\{x(t)\}} = \frac{a_0 + a_1 s + a_2 s^2 \ldots + a_n s^n}{b_0 + b_1 s + b_2 s^2 \ldots + b_m s^m} \quad m \geq n$$

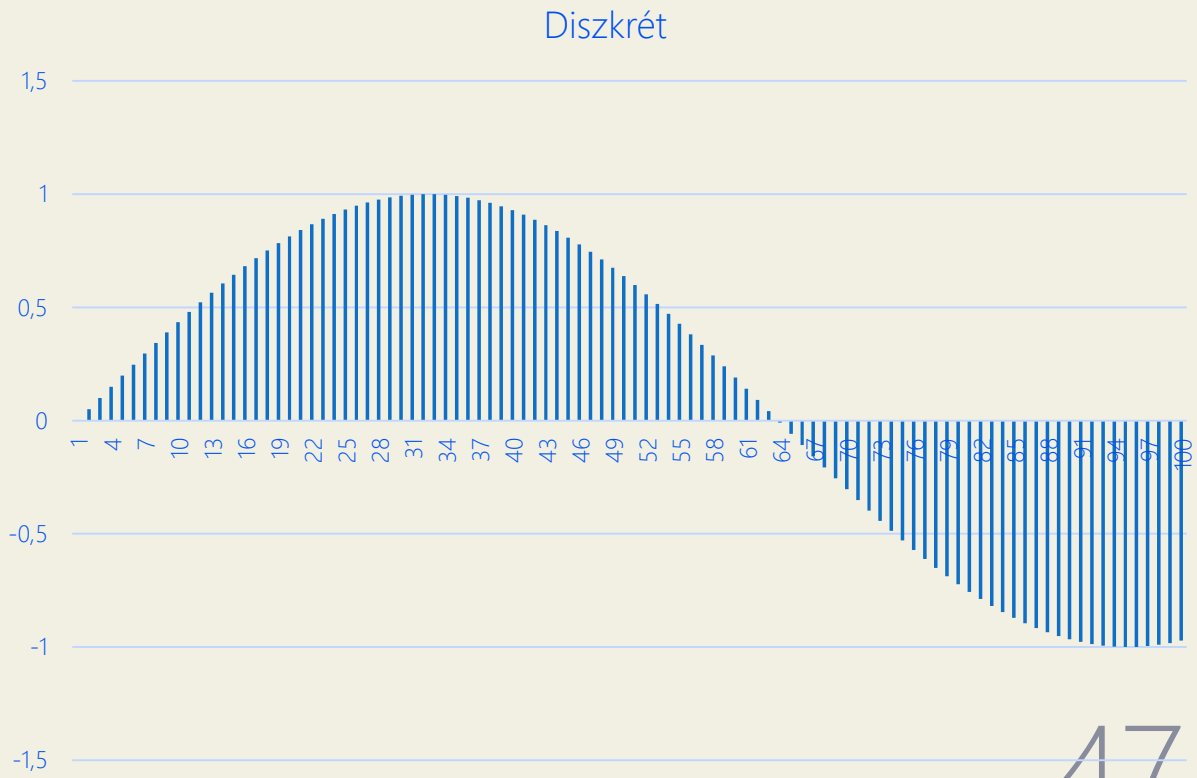
$$H(z) = \frac{Y(z)}{X(z)} = \frac{z\{y[k]\}}{z\{x[k]\}} = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2} \ldots + a_n z^{-n}}{b_0 + b_1 z^{-1} + b_2 z^{-2} \ldots + b_m z^{-m}} \quad m \geq n$$

$$H_1(z) = \frac{20z^{-1} - 30z^{-2} + 11{,}2z^{-3}}{(1 - 0{,}3z^{-1} - 0{,}58z^{-2} + 0{,}24z^{-3})} = \frac{20(z - 0{,}8)(z - 0{,}7)}{(z - 0{,}6)(z - 0{,}8)(z - 0{,}5)}$$

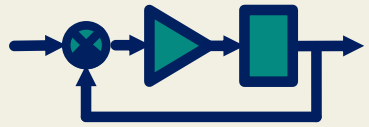
Diszkrét / Folytonos idő

Mintavételezés » Nyquist kritérium » $|f| \geq \frac{1}{2} f_s$

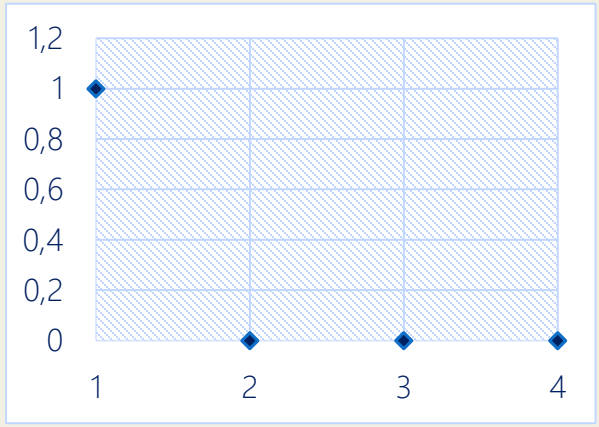




Diszkrét idejű jelek megadása

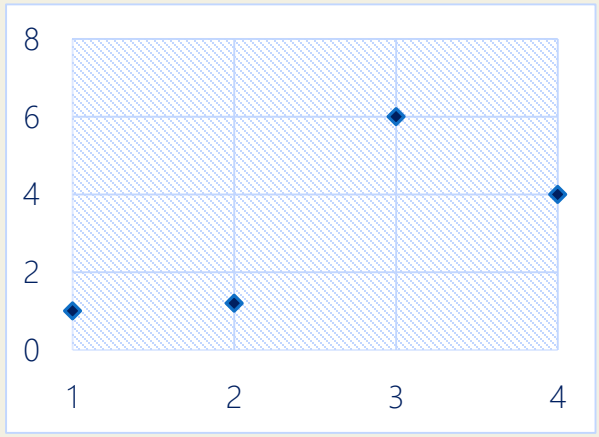


$x_1[k]$	0	1	2	3	…
x_1	1	0	0	0	…



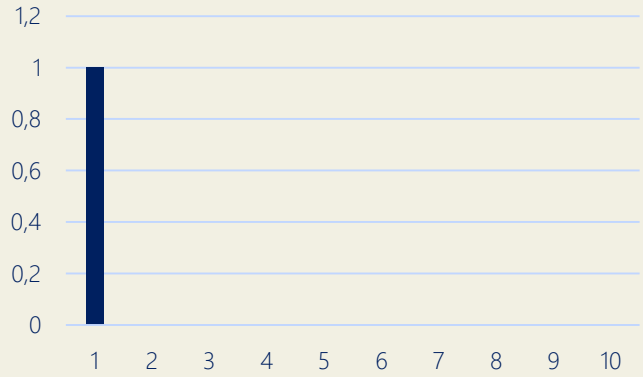
$$x_1[k] = \delta[k] = \begin{cases} 1, ha\ k=0 \\ 0, ha\ k\neq 0 \end{cases}$$

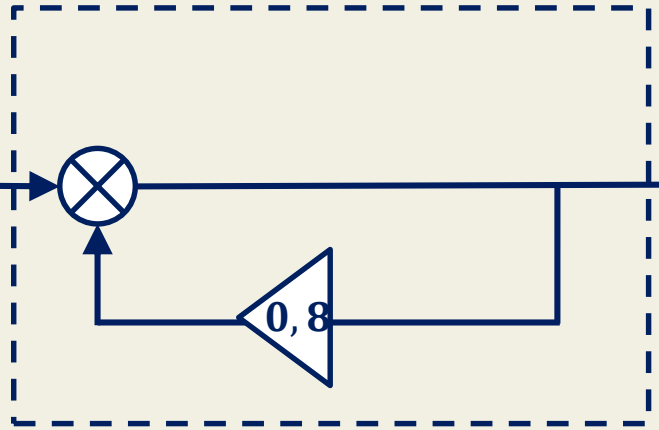
$x_2[k]$	0	1	2	3	…
x_2	1	1,2	6	4	…

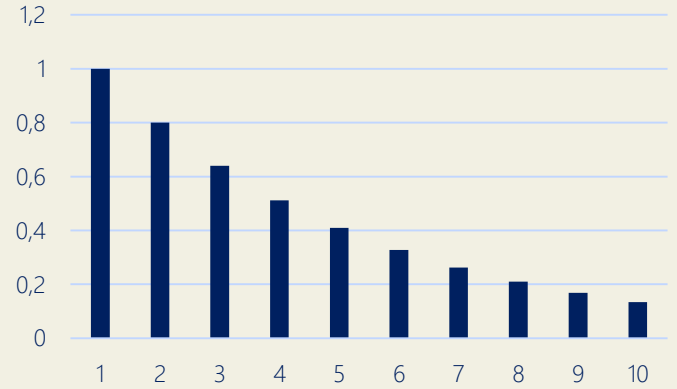


$$x_2[k] = \delta[0] + 1{,}2\delta[1] + 6\delta[2] \ldots$$

Algoritmusok

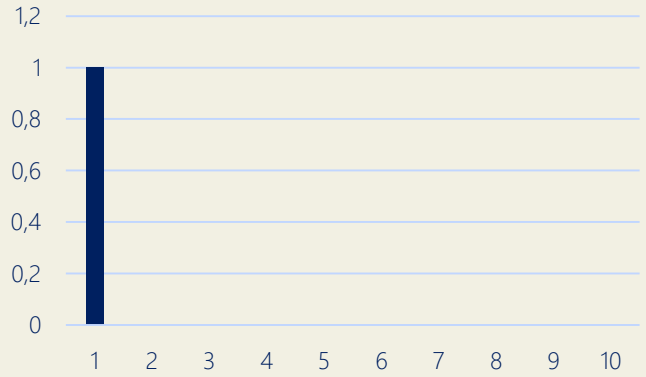


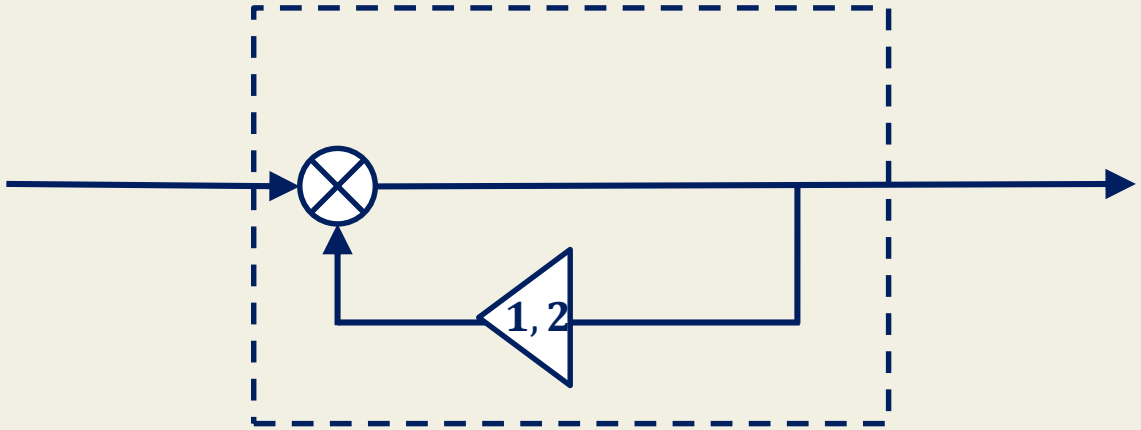


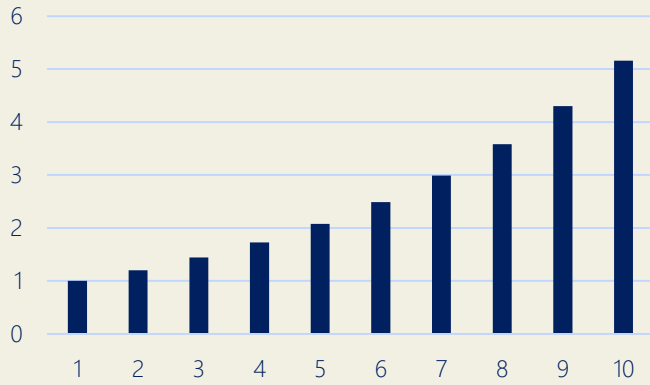



```
for(i=1; i<ELEMSZAM; i++)
 *(x+i)=*(y+i) + 0.8 * *(x+i-1);
```

Algoritmusok

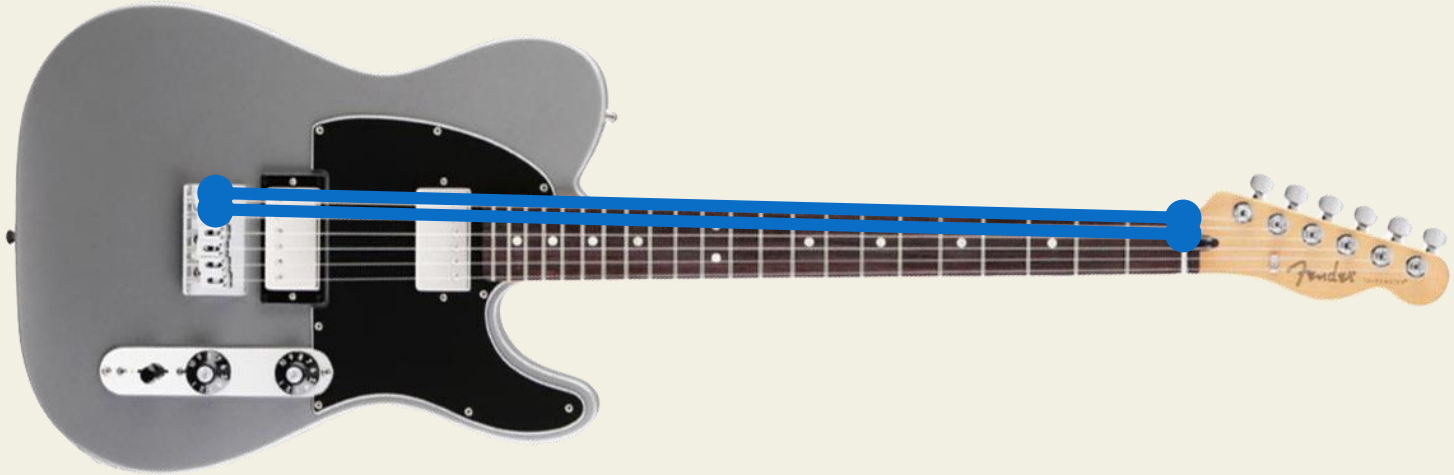




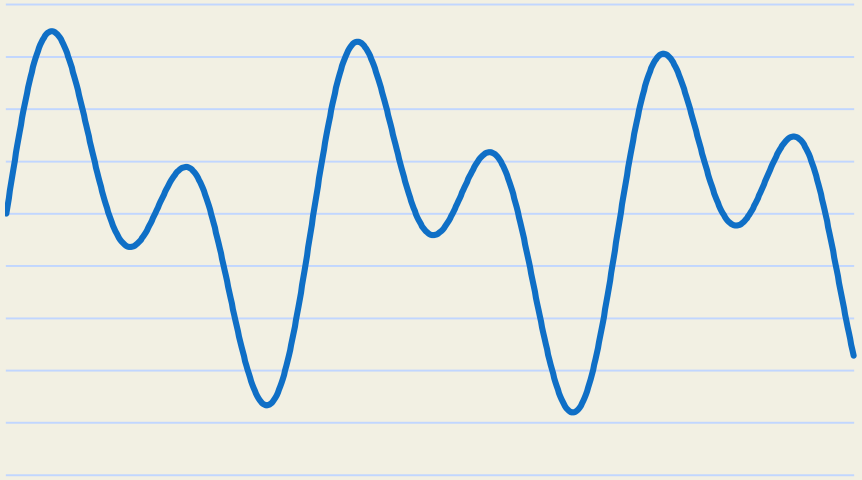


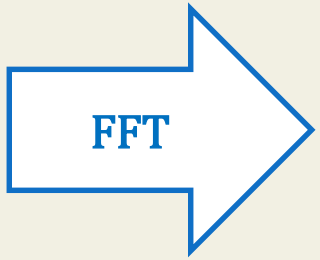

```
for(i=1; i<ELEMSZAM; i++)
 *(x+i)=*(y+i) + 1.2 * *(x+i-1);
```

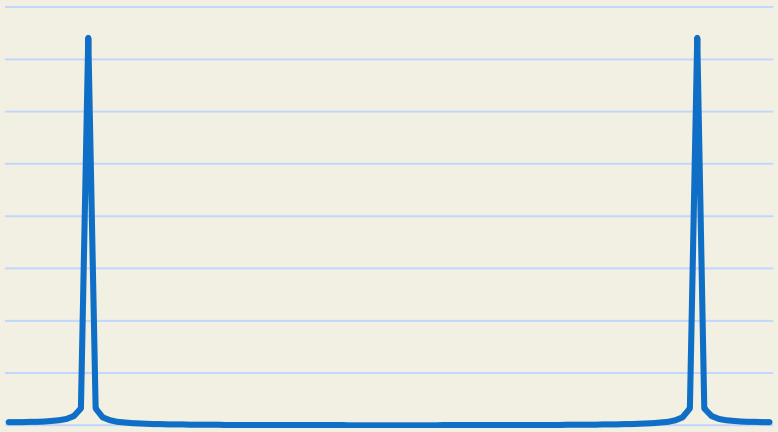
Jelek leírása frekvenciatartományban



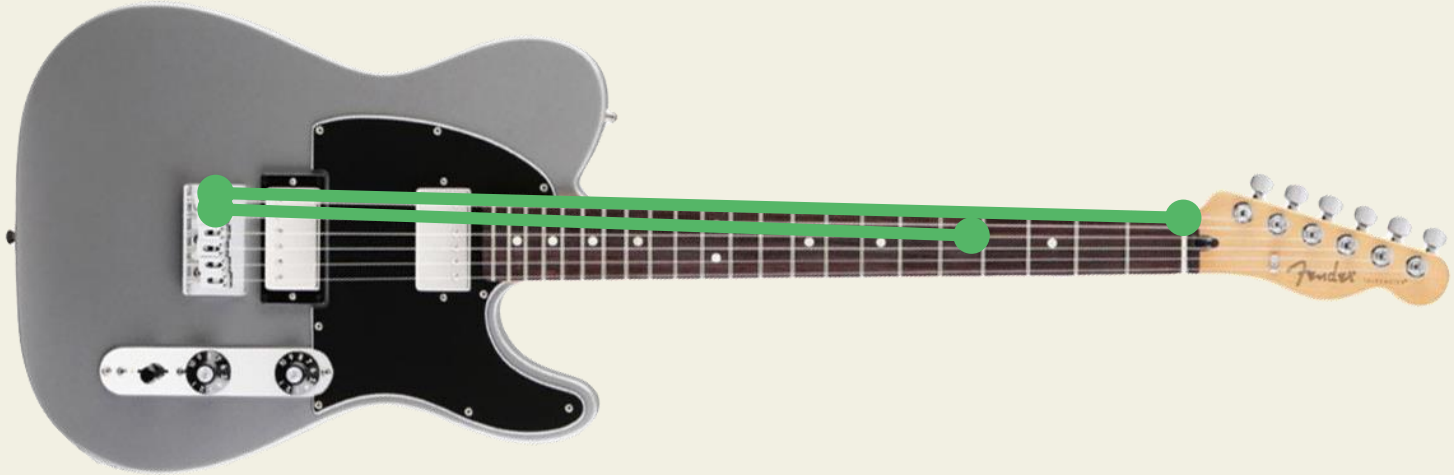
A gitár húrjainak megpendítése egy periodikus jelet állít elő.





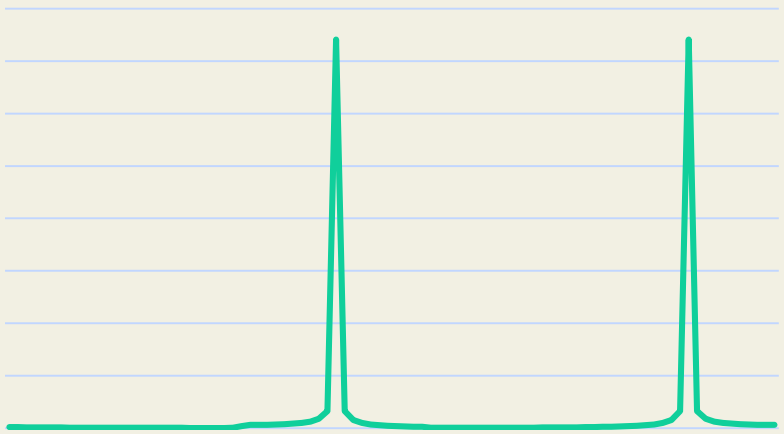


Jelek leírása frekvenciatartományban



A gitár húrjainak megpendítése egy periodikus jelet állít elő.

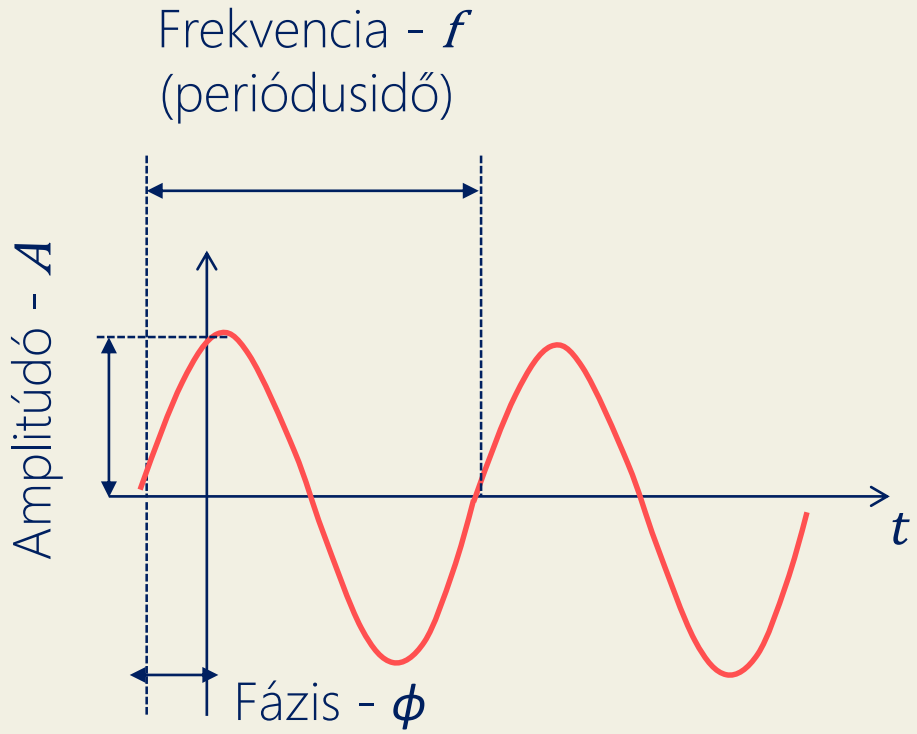
FFT



Jelek leírása frekvenciatartományban

Minden szinuszos periodikus jel leírható:

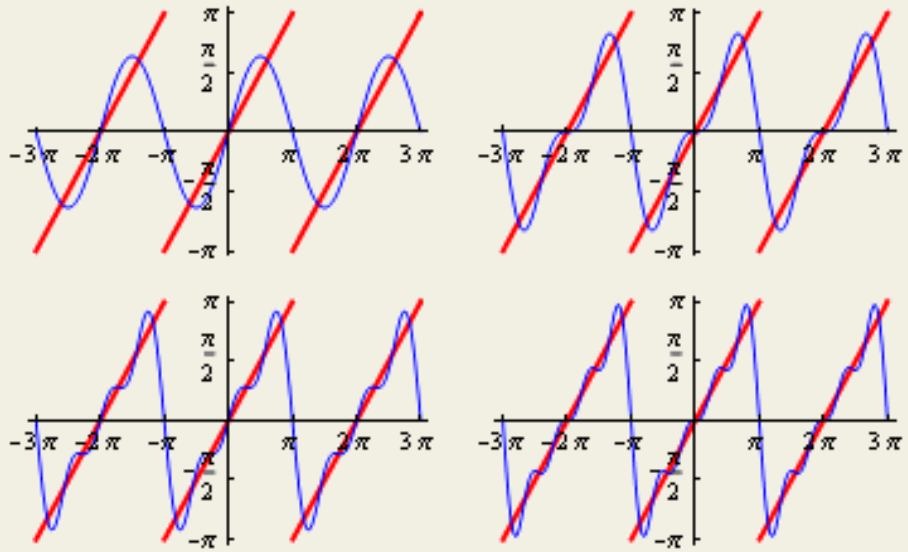
- A – amplitúdó
- f – frekvencia
- ϕ – fázis

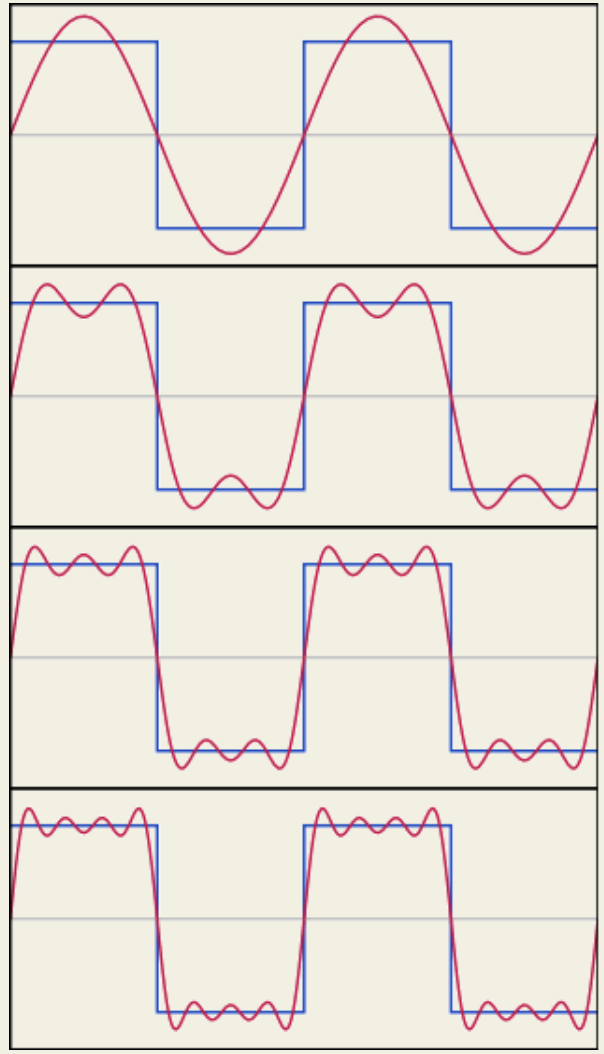


Jelek leírása frekvenciatartományban

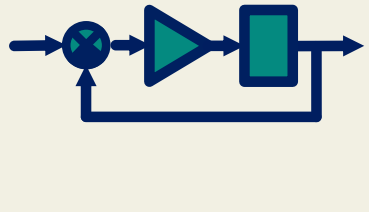
Minden periodikus jel leírható szinuszos jelek összegéből.

Ezt hívjuk Fourier sorfejtésnek.





Jelek leírása frekvenciatartományban

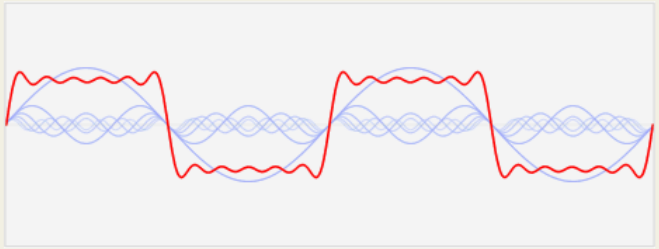


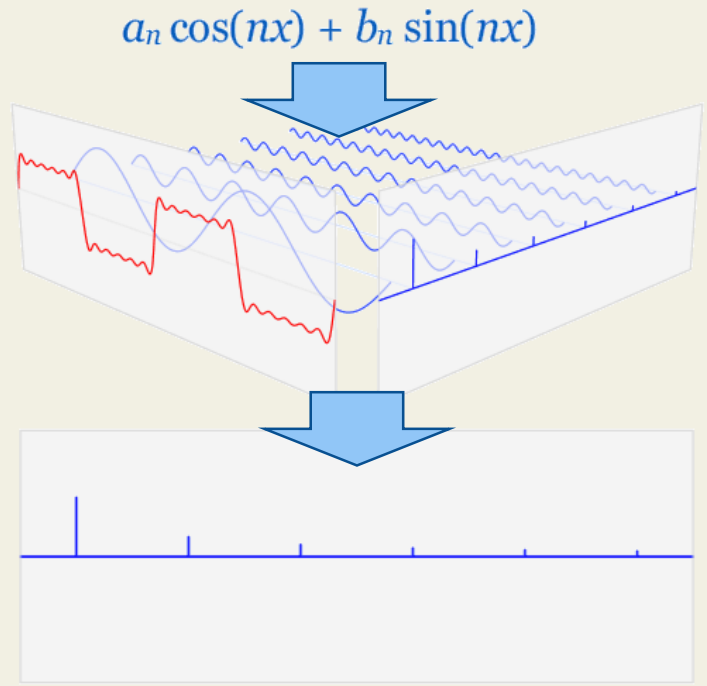
- **Nemperiodikus** jelek leírása: Fourier transzformáció
  - végtelen periódusú komponensek
  - folytonos leírás
  - eredmény: komplex frekvenciafüggvény
    - abszolútértéke: amplitúdó-frekvencia függvény (amplitúdóspektrum)
    - argumentuma: fázis-frekvencia függvény (fázisspektrum)

$$F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-j\omega t} dt$$

- Inverz Fourier-transzformáció:

$$F(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega) e^{j\omega t} d\omega$$





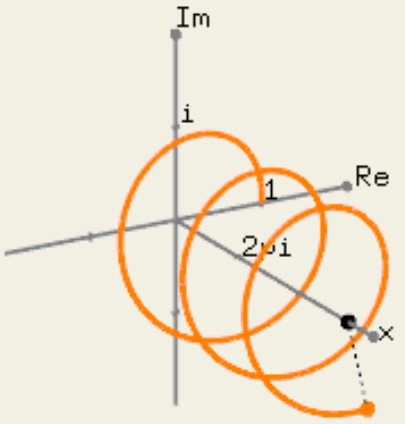
http://commons.wikimedia.org/wiki/File:Fourier_transform_time_and_frequency_domains.gif
http://en.wikipedia.org/wiki/Fourier_transform
admin. (2006, May 26). Lecture 5: Representation of Nonperiodic Functions. Retrieved January 07, 2011, from Free Online Course Materials — USU OpenCourseWare Web site: http://ocw.usu.edu/Electrical_and_Computer_Engineering/Signals_and_Systems/5_2node2.html.

$\hat{f}$

Euler's identity (azonosság)

$$e^{i\pi} + 1 = 0$$

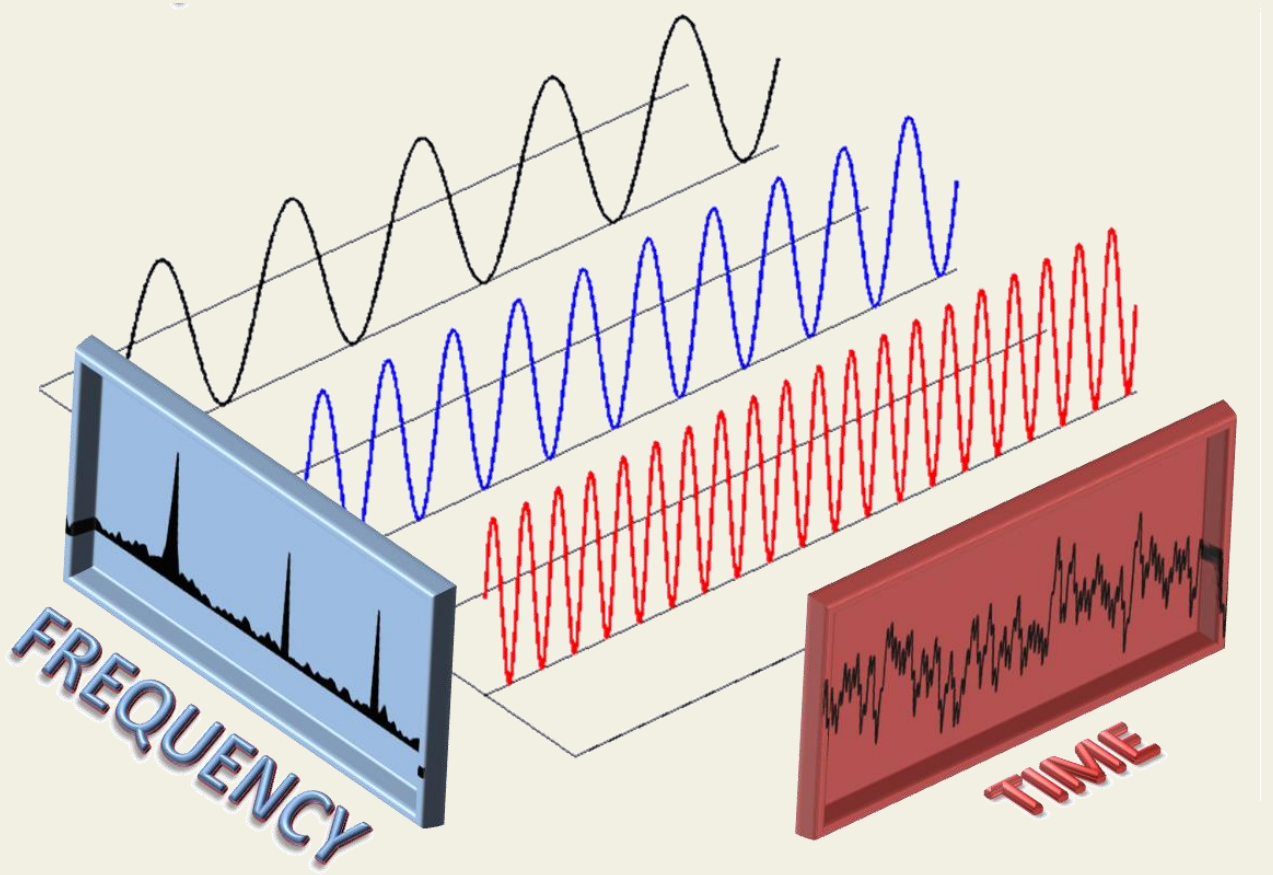
- **e** - Euler szám, a természetes logaritmus alapja
- **i** - képzetes rész, $\sqrt{-1}$
- **π** - pi



$$e^{ix} = \cos x + i \sin x$$

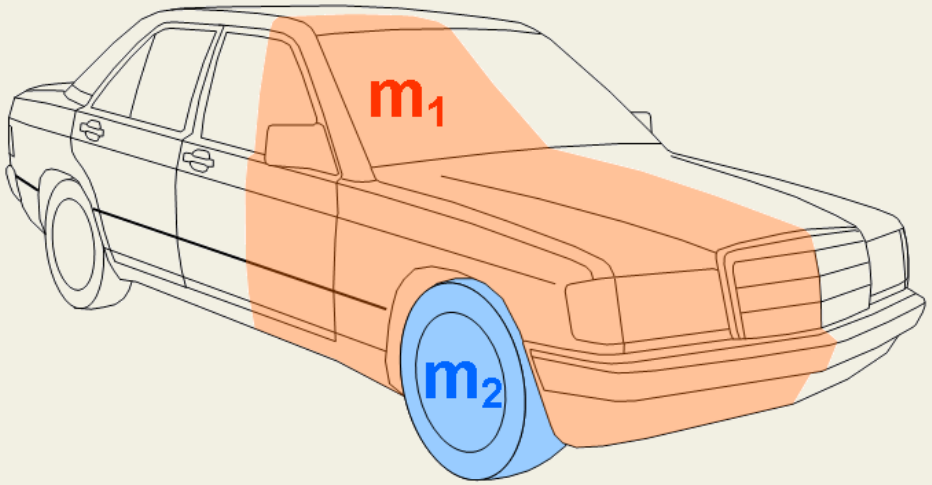
Jelek leírása frekvenciatartományban

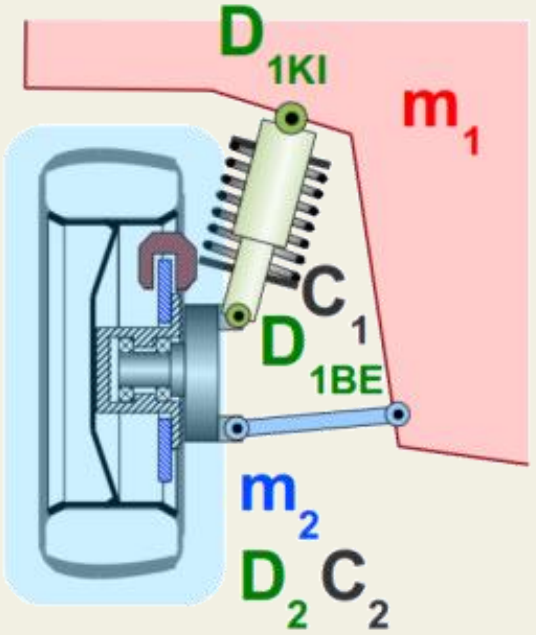
$$F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-j\omega t} dt$$

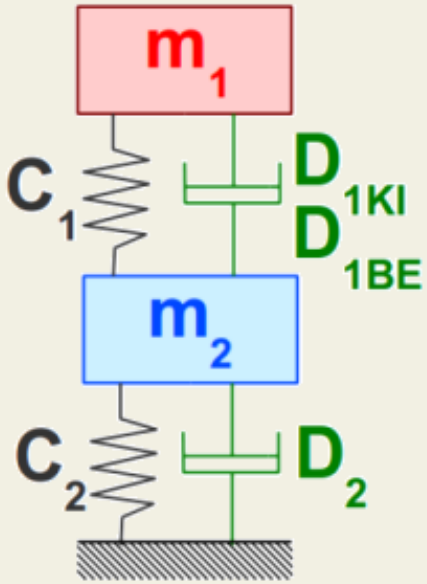


Két tömegű lengőrendszer

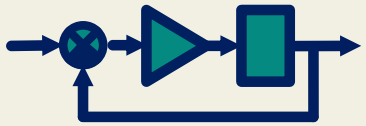
- Emlékeztetőül



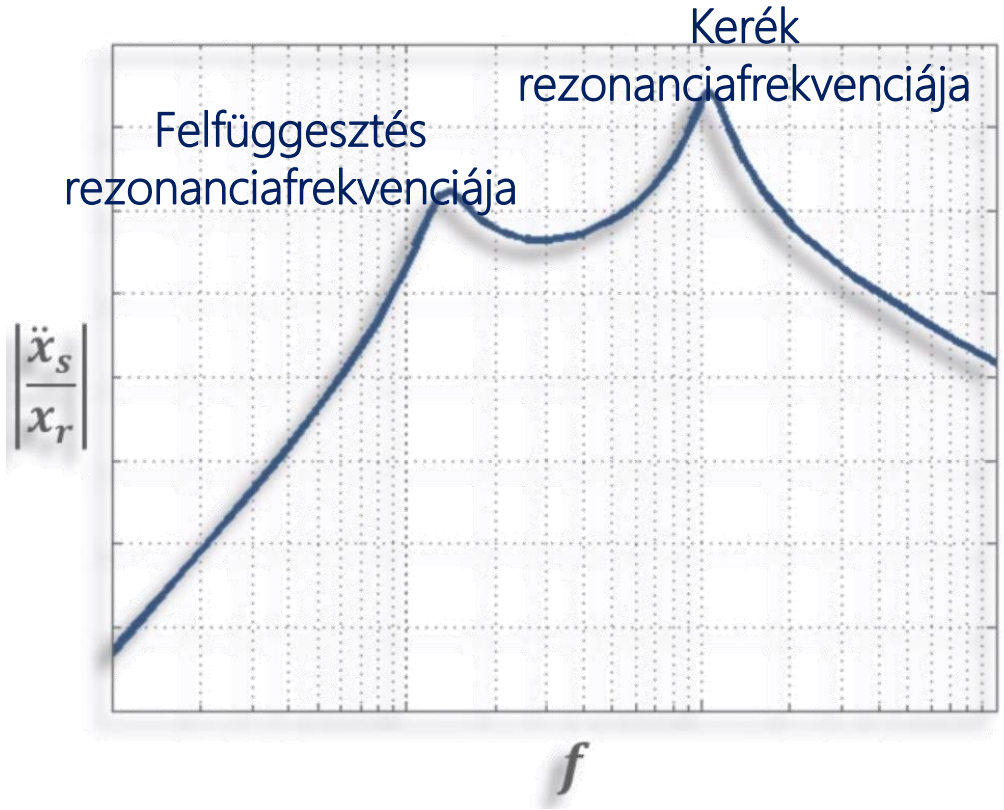




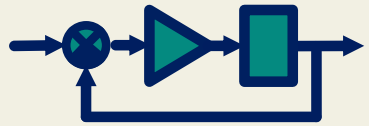
Két tömegű lengőrendszer

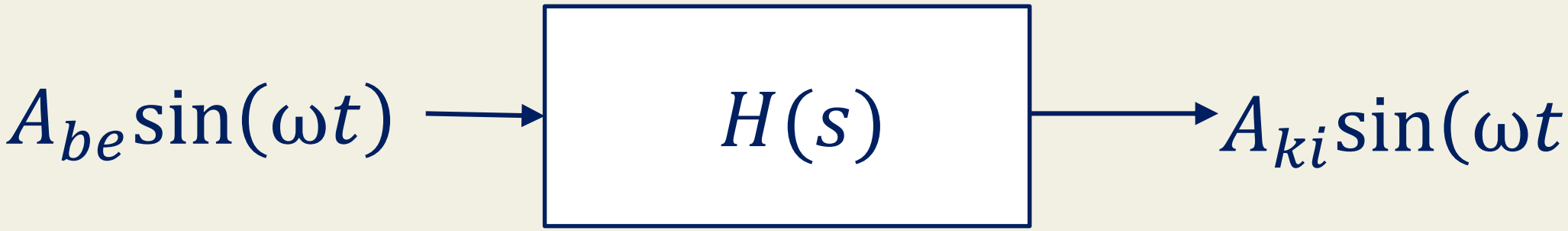


A bode diagram SISO rendszer átviteli karakterisztikájának ábrázolására szolgál.

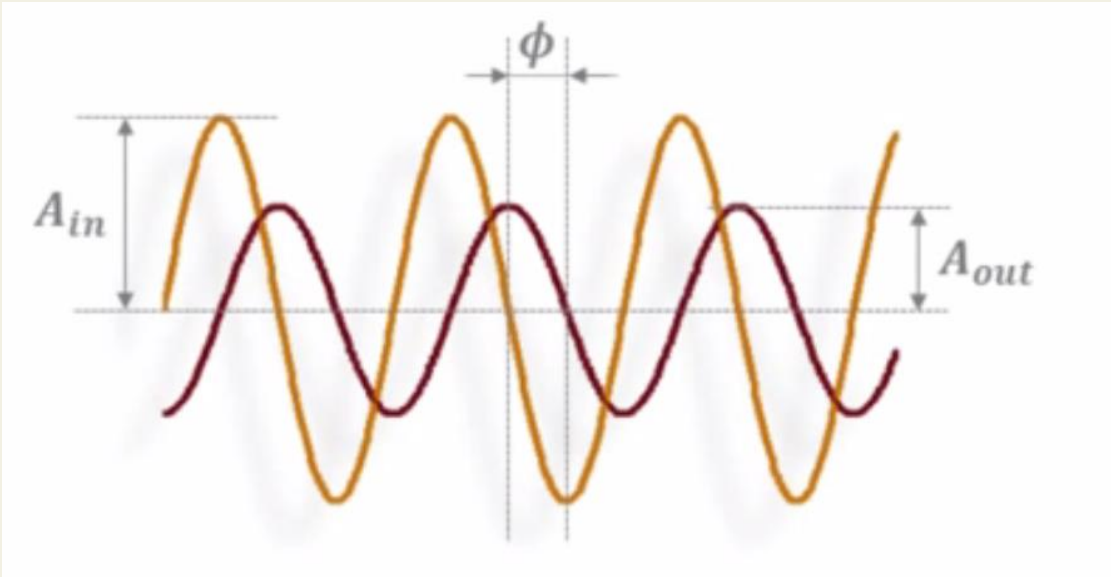


Bode diagram

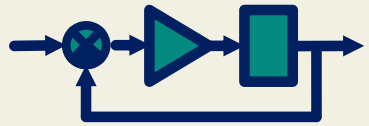




Az amplitúdó és a fázis változhat, a frekvencia nem!

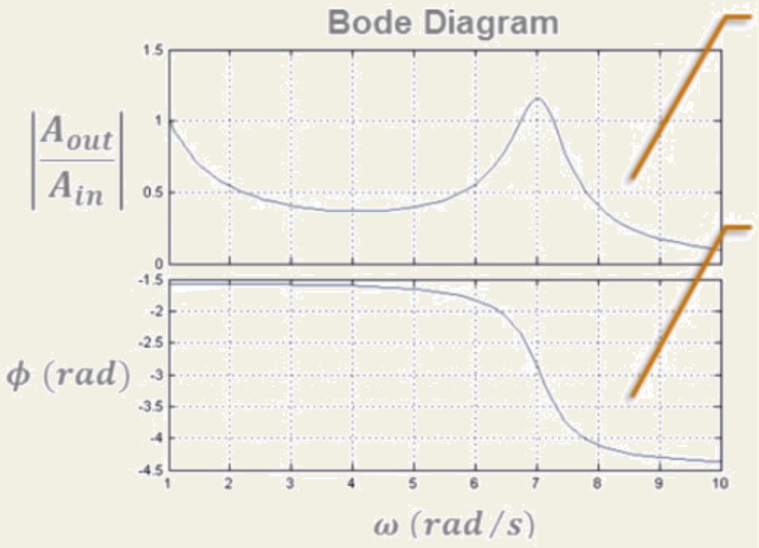


Bode diagram



$$A_{be}\sin(\omega t) \rightarrow \boxed{H(s)} \rightarrow A_{ki}\sin(\omega t + \phi)$$

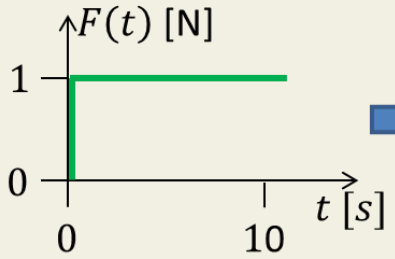
Az amplitúdó és a fázis változhat, a frekvencia nem!



Lineáris rendszer amplitúdóváltozása

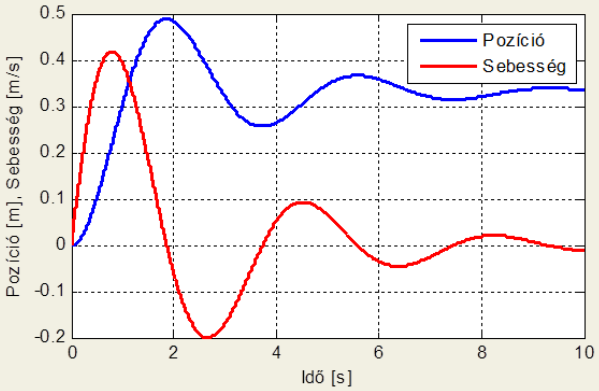
Lineáris rendszer fázisváltozása

Komplex frekvenciaátviteli függvény

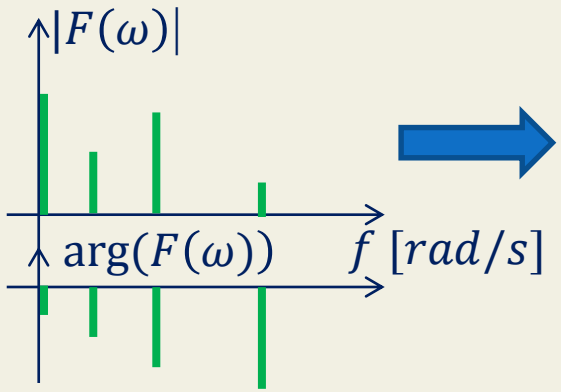


Gerjesztőerő
(bemenő időfüggvény)

Diff.egyenlet

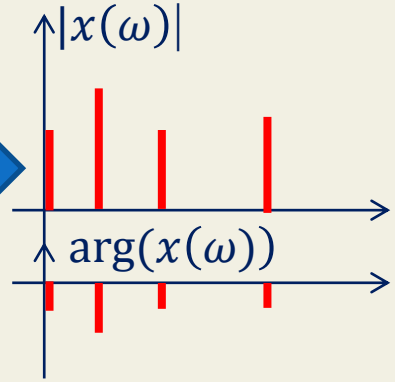


Mozgásmennyiségek
(kimenő időfüggvény)



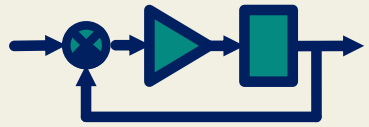
Gerjesztőerő
(komplex frekvenciafv.)

Komplex frekvenciaátviteli fv.



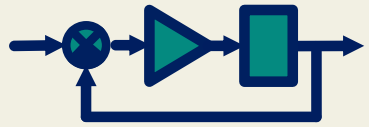
Mozgásmennyiségek
(komplex frekvenciafv.)

State machine - állapotgép



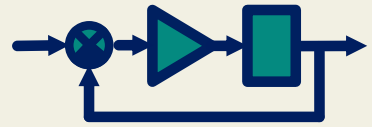
- Az állapotgép egy matematikai modell, amit gyakran használnak programfejlesztés, digitális áramkörök tervezése és egyéb mérnöki munka során.
- Példák:
  - » motorvezérlés
  - » kávégép
  - » szövegfeldolgozó
  - » robotkar vezérlése
  - » stb.
- A lényege, hogy az adott alrendszer csak egy állapotban lehet egyszerre, ebből az átmenetek és az egyes állapotok tevékenységei pontosan definiáltak.

State machine - állapotgép



- Bármilyen programnyelven megvalósítható, több leírása létezik:
  - » szöveges: állapot és tevékenységtáblás
  - » rajzos: UML és sok nem szabványos
- Mi egy C és LabVIEW nyelvi példát veszünk UML diagrammai
  - » http://en.wikipedia.org/wiki/Finite-state_machine
  - » http://en.wikipedia.org/wiki/UML_state_machine

Szövegfeldolgozó állapotgéppel



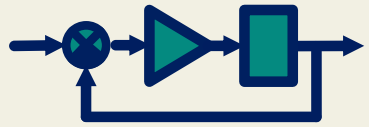
- Tegyük fel, hogy nem tudjuk, milyen hosszú egy sor
  - » nem olvashatunk tömbbe egy sort
  - » karakterenként olvassunk
- Szűrjük ki a /*-gal kezdődő és */-ig tartó kommenteket

```
Szöveg* valami /*komment ** és /*/ hh /*a*/ szöveg
```

4 állapotú state machine implementáció

```c
//state machine - állapotgép
#include <stdio.h>
typedef enum {normal,komment,cs_var,p_var}
allapotok;

int main(void){
  int c;
  allapotok a = normal;
  //CTR+Z-ig
  while((c=getchar())!=EOF){
    switch (a){
    case normal:
      if(c!='/')putchar(c);
      else a = cs_var;
      break;
    case cs_var:
      if(c=='*') a = komment;
      else{
        putchar('/');
        if(c!='/'){
          putchar(c);
          a=normal;
        }
      }
      break;
    case komment:
      if(c=='*') a = p_var;
      break;
    case p_var:
      if(c=='/') a = normal;
      else if(c!='*') a = komment;
      break;
    }
  }
}
```

Komment /* */ szűrő



Állapottábla

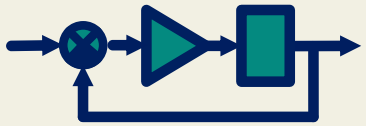
Állapot	'*'	'/'	egyéb
normál	normál	csillagra vár	normál
csillagra vár	megjegyzés	csillagra vár	normál
megjegyzés	perre vár	megjegyzés	megjegyzés
perre vár	perre vár	normál	megjegyzés

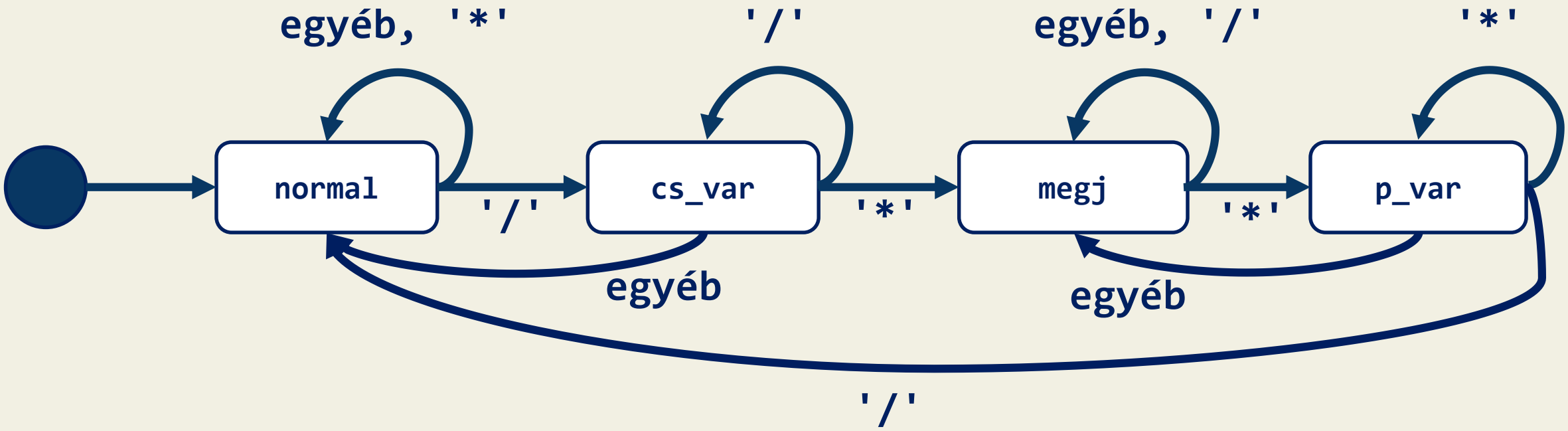
Tevékenységtábla

Állapot	'*'	'/'	egyéb
normál	másol	nem másol	másol
csillagra vár	nem másol	előző '/' kiírása, nem másol	előző '/' kiírása, másol

Forrás: A programozás alapjai - Pohl László, Budapest, 2010

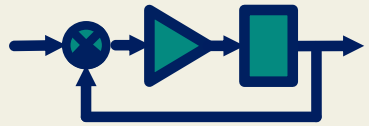
Komment /* */ szűrő





Az implementációba be lehet tenni egy plusz állapotot - végállapotot -, amibe bármikor elérhetünk EOF hatására.

Just in case! :)



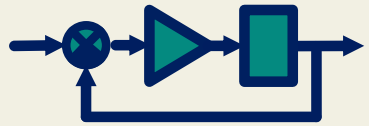
Görög betűk

- δ - delta
- θ - theta
- ω - omega
- σ - sigma
- μ - mű

Mátrixszorzás

$$\begin{bmatrix} 2 & 3 & 4 \\ 1 & 0 & 0 \\ 0 & 0 & 2 \end{bmatrix} \begin{bmatrix} 1000 \\ 100 \\ 10 \end{bmatrix} =$$

Just in case! :)

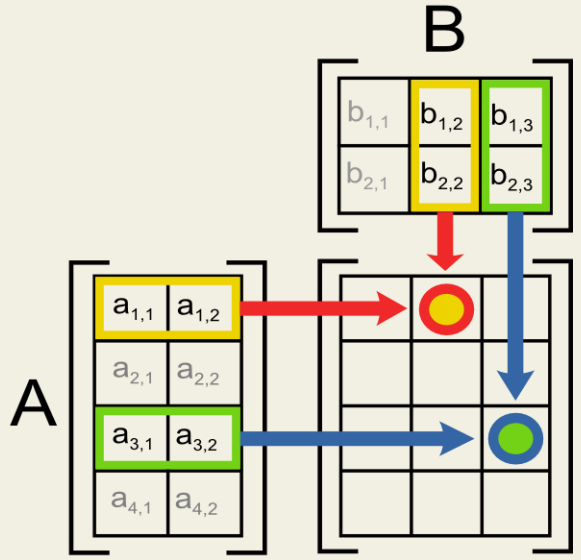


Görög betűk

- δ - delta
- θ - theta
- ω - omega
- σ - sigma
- μ - mű

Mátrixszorzás

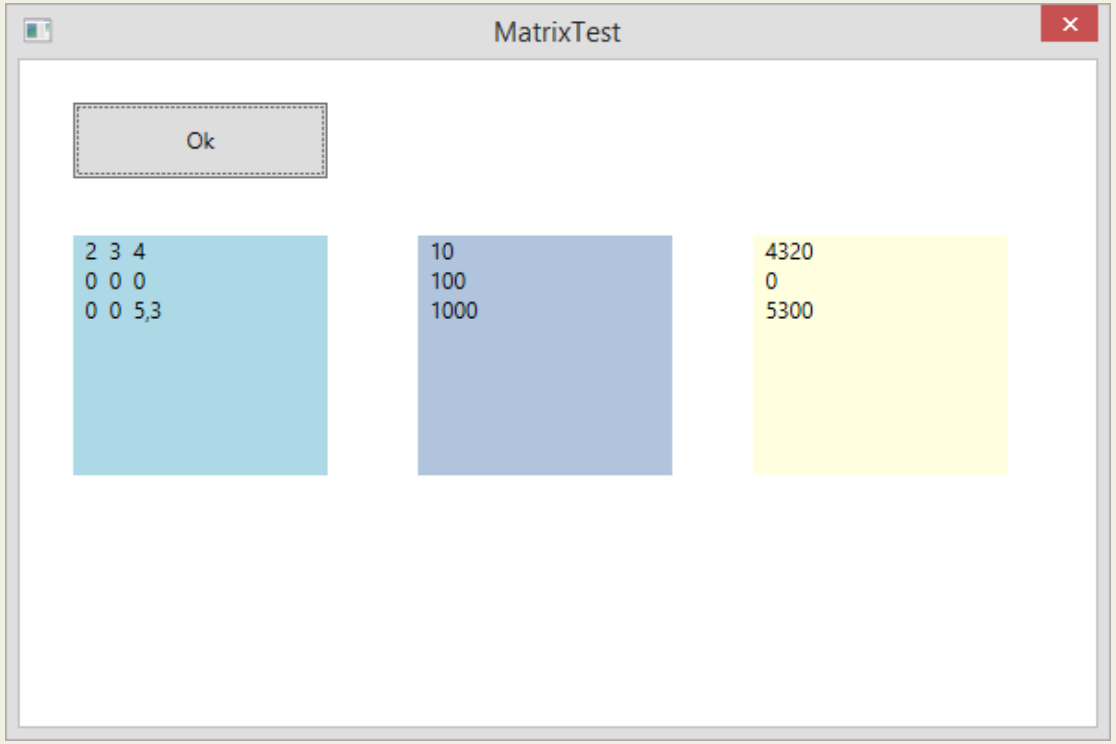
$$\begin{bmatrix} 2 & 3 & 4 \\ 1 & 0 & 0 \\ 0 & 0 & 2 \end{bmatrix} \begin{bmatrix} 1000 \\ 100 \\ 10 \end{bmatrix} = \begin{bmatrix} 2340 \\ 1000 \\ 20 \end{bmatrix}$$

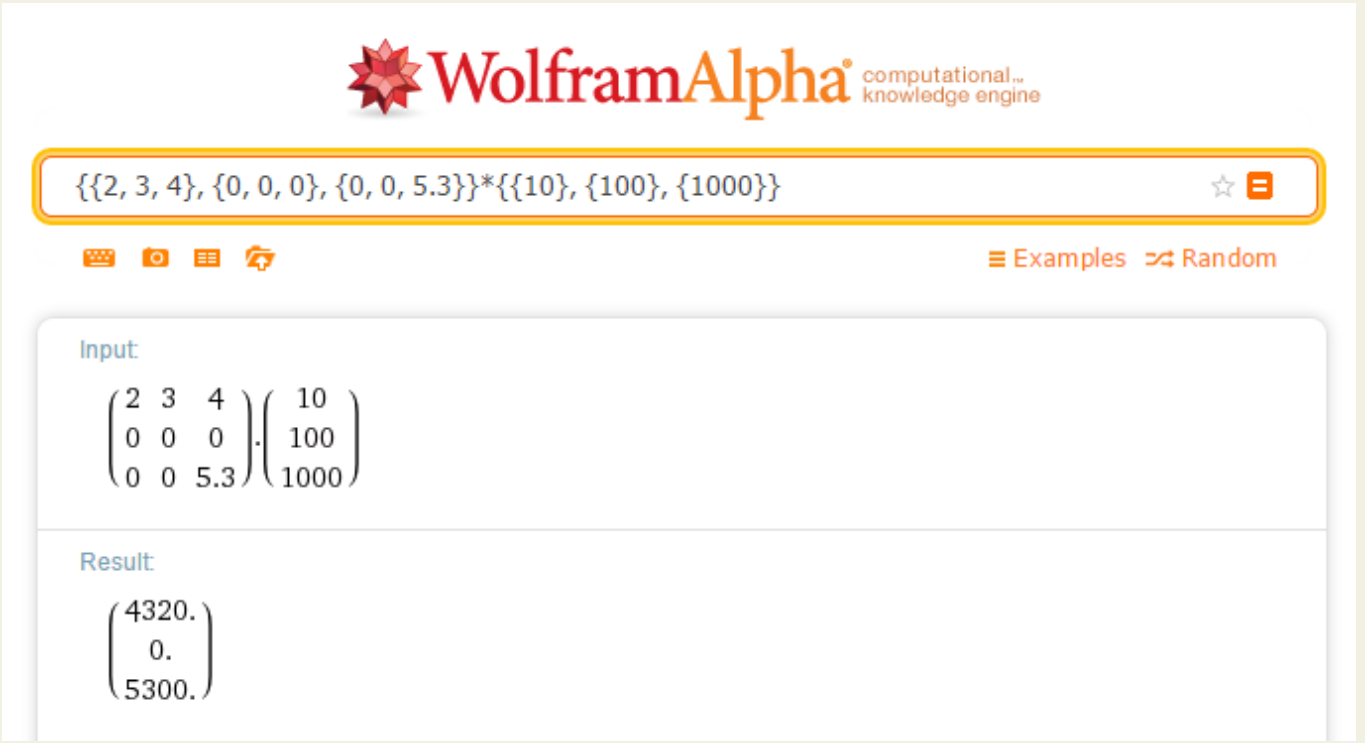


http://en.wikipedia.org/wiki/Matrix_(mathematics)

Mátrixok

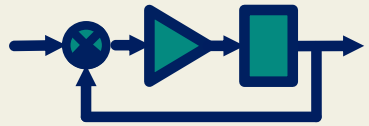
Mátrixszorzás bemutatása C# tesztprogramon keresztül





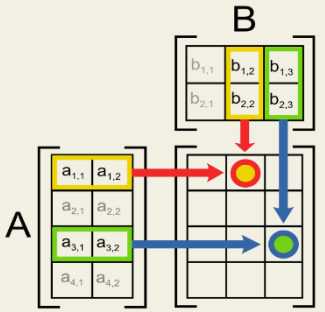
Ellenőrizzük: www.wolframalpha.com

Mátrixok



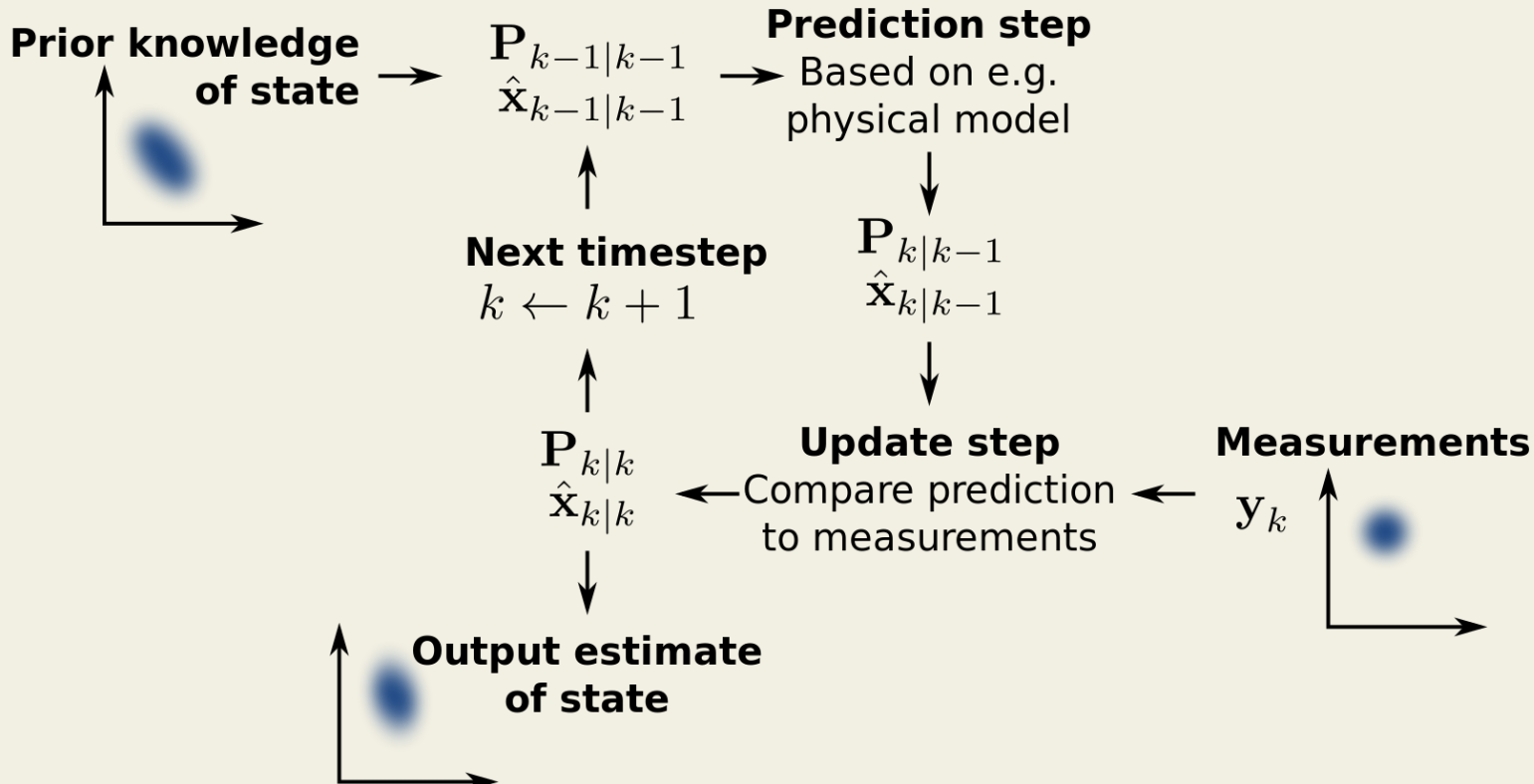
A és B mátrix szorzása, eredménye C mátrix, csak akkor lehetséges, ha A mátrix oszlopainak száma megegyezik a B mátrix sorainak számával.

```
if (a.GetLength(1) == b.GetLength(0))
{
  c = new float[a.GetLength(0), b.GetLength(1)];
  for (int i = 0; i < c.GetLength(0); i++)
   {
    for (int j = 0; j < c.GetLength(1); j++)
     {
       c[i, j] = 0;
       for (int k = 0; k < a.GetLength(1); k++) // vagy k<b.GetLength(0)
           c[i, j] = c[i, j] + a[i, k] * b[k, j];
     }
   }
}
```

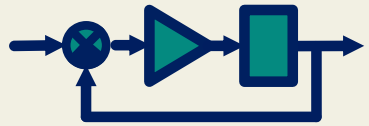


Kalman filter

- Kálmán Rudolf Emil (1930 -)
- Zajos bemenő adatok rekurzív mérésével egy pontosabb becslést ad a mérés tárgyának állapotáról.
  http://en.wikipedia.org/wiki/Kalman_filter



Bayes-tétel



Adott A és B események valószínűsége ($P(A)$ és $P(B)$), és a $P(B|A)$ feltételes valószínűség esetén fennáll a

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

egyenlőség.

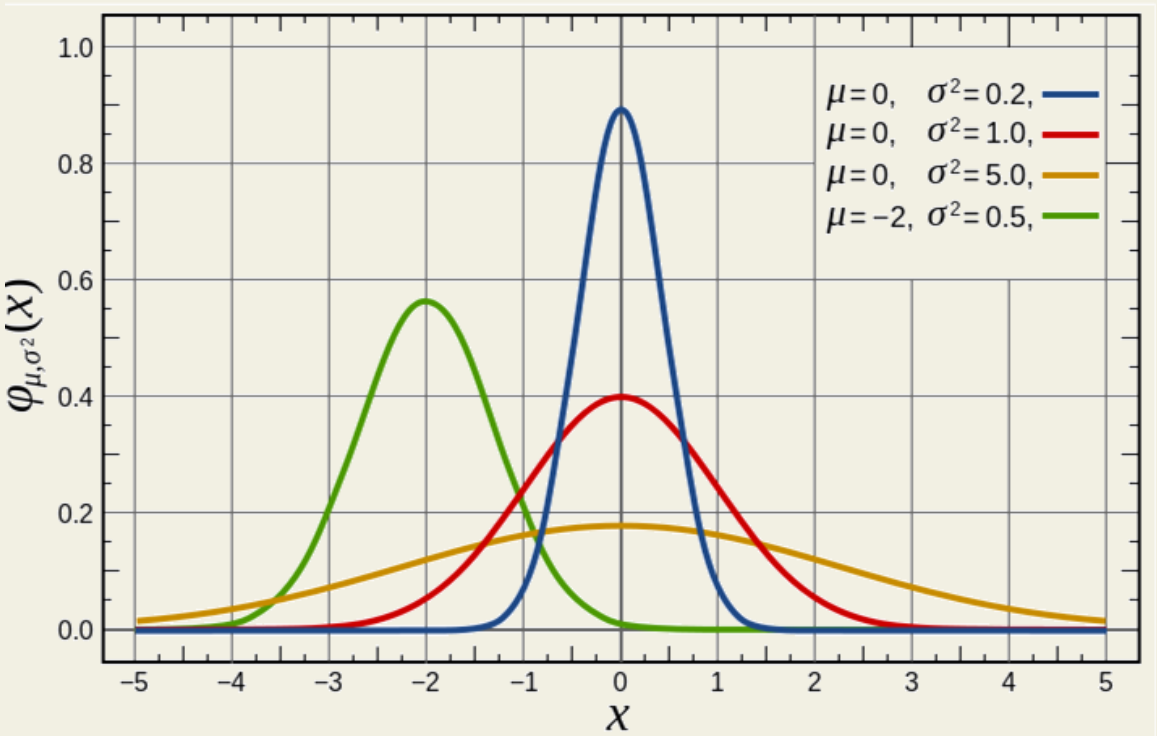
http://en.wikipedia.org/wiki/Bayes'_theorem

Normális eloszlás

Normal (Gaussian) distribution

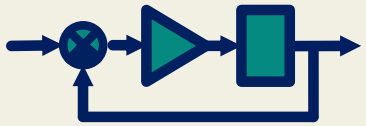
Sűrűségfüggvénye:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$



http://en.wikipedia.org/wiki/Normal_distribution

Felhasznált irodalom



- A C programozási nyelv - Az ANSI szerinti változat. B. W. Kernighan - D. M. Ritchie; Műszaki Könyvkiadó, 1995
- Kuslits M. jegyzetei
- A programozás alapjai - Pohl L., Budapest, 2010
- Differenciálegyenletek numerikus megoldása 2010, Pécsi Tudományegyetem Kollár B.
- Wikipedia
- Kollár B.: Differenciálegyenletek numerikus megoldása, 2010