Mechatronika szigorlat

Vári Gergő

2026. február 21.

Tartalomjegyzék

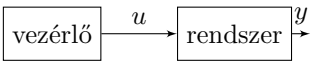
1 Mechatronika

1.1 Hasonlítsa össze a vezérlést és a szabályozást: a hatáslánc jellege, zavarjelekkel szembeni ellenállás, a mért mennyiség fajtája, reakció idő, illetve az irányításhoz felhasznált eszközök költsége szerint!

Vezérlés:

- nyílt hatáslánc: nincs visszacsatolás
- a rendszer belső jellemzőit/bemeneteit mérjük, valamint részben függ a külső feltételektől is, de a kimenetet (irányítani kívánt jellemzőt) nem mérjük
- logikai függvényt hozunk létre
- gyors reakció
- csak determinisztikus zavarok kezelésére képes
- vagy egyszerű (olcsó), de így kevés dolgot veszünk figyelembe
- vagy komplex (drága) és sok mindent figyelembe veszünk

vezérlés hatáslánca:



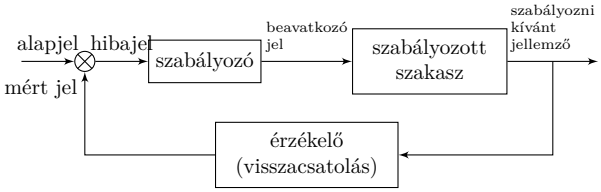
vezérlés fajtái:

- idővezérlés – pl. lámpa időzített fel-, lekapcsolása
- lefutó vezérlés
  - sorrendi – pl. csomagolás
  - feltétel

Szabályozás:

- zárt hatáslánc: van visszacsatolás
- az irányítani kívánt jellemző is tudja befolyásolni a folyamatot
- instabil rendszert is képes kezelni (stabilizálni)
- lassabb reakció, pontatlanabb
- képes kiküszöbölni a zavarjeleket
- alacsony műszaki komplexitás:
- mérés
- kivonás
- jelerősítés

szabályozás hatáslánca:



szabályozás fajtái:

- értéktartó – pl. hőmérséklet
- követő – pl. pálya
- kaszkád – több hurkú szabályozás
- állapotszabályozás – állapottér modell alapján szabályoz – többváltozós rendszerek – pl. inverz inga egyensúlyban tartása

1.2 Ismertesse az alábbi fogalmakat: linearitás, statikus/dinamikus rendszer, időinvariáns/idővariáns rendszer, folytonos/diszkrét idejű rendszer, koncentrált/elosztott paraméterű modell!

Linearitás:

- érvényes a szuperpozíció elve
- matematikai szemmel 2 feltétel:
  - számmal szorzás
  - hatások összege
- ha a rendszer u_i gerjesztésre y_i választ ad és u_j gerjesztésre y_j választ ad, akkor u_i és u_j lineáris kombinációjára y_i és y_j lineáris kombinációját adja

$$u = \lambda_i u_i + \lambda_j u_j \rightarrow y = \lambda_i y_i + \lambda_j y_j$$

- nem linearitás feloldása: munkaponti linearizálás

Statikus/dinamikus rendszer:

- statikus:
  - bármely pillanatban a bemenő jelek pillanatnyi értékei meghatározzák az adott pillanatban kimenő jelek értékeit – pl. lámpa fel-le kapcsolva
- dinamikus:
  - valós fizikai rendszerek működésének időbeli lefolyását is leírják, jellemzően idő szerinti differenciálegyenletek segítségével – pl. rezgéstani példák
  - memória jelleggel rendelkeznek

Időinvariáns/idővariáns rendszer:

- ha a rendszer $u(t)$ bemenetre $y(t)$ kimenetet ad és $u(t-\tau)$ bemenetre $y(t-\tau)$ kimenetet ad, akkor a rendszer időinvariáns – pl. fékrendszer: ahogy melegszik, változik a súrlódás
- „ma, holnap és két év múlva is ugyanúgy viselkedik”

Folytonos/diszkrét idejű rendszer:

- folytonos idejű:
  - $u(t)$ és $y(t)$ egy vizsgált $[T_a, T_b] \subseteq \mathbb{R}$ időintervallumban minden időpontban értelmezve van: $t \in [T_a, T_b] \subseteq \mathbb{R}$
- diszkrét idejű:
  - $u(t)$ és $y(t)$ csak diszkrét $t = T_k, k \in \mathbb{N}, t \in \{..., T_{-k}, ..., T_{-1}, T_0, T_1, ..., T_k, ...\}$ időpontok sorozatában van értelmezve, ahol: $T_{k-1} < T_k < T_{k+1}$
  - általában $T_k = k \cdot T_s$, ahol T_s a mintavételezési idő
  - $u[k] := u(T_k)$
  - $y[k] := y(T_k)$

Koncentrált/elosztott paraméterű modell:

- koncentrált paraméterű leírás:
  - a vizsgált valós fizikai rendszer összefüggéseit azok jellegétől függően egy adott térrészben összegezzük, vagy kiátlagoljuk, és egyetlen egyenlettel helyettesítjük

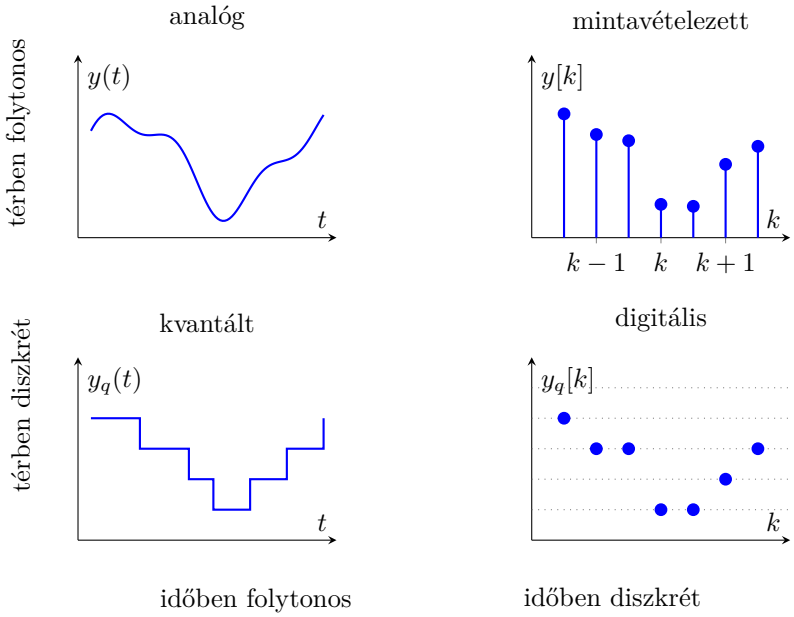
1.3 Hasonlítsa össze a rendszereket a leírt állapotváltozók (dimenzió) száma (véges/végtelen), valamint annak diszkrét/folytonos jellege szerint!

A rendszer dimenziója:

- a rendszer állapotváltozóinak száma
  - állapotváltozó: az állapot egyértelmű leírására szolgálnak
  - állapot: a múlt összesített hatása
- végtelen: végtelen számú állapotváltozóval írható le a rendszer
- véges: véges számú állapotváltozóval írható le a rendszer
  - $\underline{x}(t) \in \mathbb{R}^n$

Diszkrét/folytonos jelleg:

- diszkrét állapotú:
  - ha egy véges dimenziójú rendszer állapotváltozói véges számú értéket vehetnek fel, akkor a rendszer diszkrét állapotú
  - pl. digitális technika: 0 vagy 1
  - állapotautomaták:
    - az állapotbeli változások egy-egy esemény hatására ugrásszerűen mennek végbe
    - tipikusan szekvenciális hálózatok
- folytonos állapotú:
  - az állapotváltozók folytonos értéket vesznek fel
  - pl. sebesség



Diszkrét idejű folytonos értékű rendszer:

- differencia egyenletekkel írható le $\underline{x}[k+1] = \underline{\Phi x}[k] + \underline{\Gamma u}[k]$
- az állapotváltozók tetszőleges értéket vehetnek fel, de a diszkrét idő miatt ugrásszerűen mennek végbe

Folytonos idejű folytonos értékű rendszer:

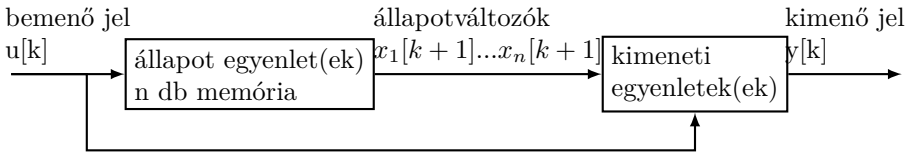
- differenciál egyenletekkel írható le $\underline{\dot{x}}(t) = \underline{Ax}(t) + \underline{Bu}(t)$
- állapotváltozók tetszőleges értéket vehetnek fel, folytonos idő miatt folyamatosan változnak

1.4 Írja fel a diszkrét idejű állapottér modell általános algebrai alakját, magyarázó ábrával szemléltesse! Ismertesse az összefüggésben szereplő együtthatók szerepét!

Állapottér modell:

- egyfajta matematikai modell
- állapotváltozók segítségével
- általános állapottér modell:

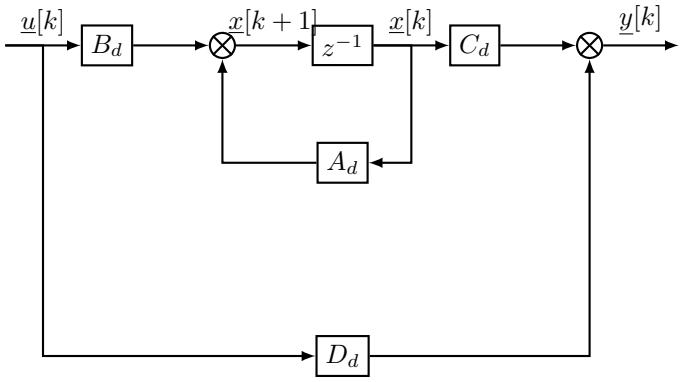
- lineáris idő invariáns (LTI) rendszerek diszkrét idejű állapottér modellje:



- MIMO (Multiple Input Multiple Output) diszkrét idejű állapotegyenlete:

$$\underline{x}[k+1] = \underline{\underline{A_d}}\underline{x}[k] + \underline{\underline{B_d}}\underline{u}[k]$$
$$\underline{y}[k] = \underline{\underline{C_d}}\underline{x}[k] + \underline{\underline{D_d}}\underline{u}[k]$$

  - $\underline{\underline{A_d}}$: állapotmátrix – a jelenlegi állapot hatása a következő állapotra
  - $\underline{\underline{B_d}}$: bemeneti mátrix – a jelenlegi bemenet hatása a következő állapotra
  - $\underline{\underline{C_d}}$: kimeneti mátrix – az állapot hatása a kimenetre
  - $\underline{\underline{D_d}}$: segédmátrix – a bemenet hatása közvetlenül a kimenetre
  - a mátrixok dimenziója függ a bemenetek és kimenetek számától, pl. egy SISO (Single Input Single Output) rendszernél: $\underline{\underline{B_d}} \in \mathbb{R}^{n\times 1}$, $\underline{\underline{C_d}} \in \mathbb{R}^{1\times n}$, $\underline{\underline{D_d}} \in \mathbb{R}^{1\times 1}$
  - hatásvázlat:



1.5 Írja fel a diszkrét rendszerek be-kimeneti modellezését reprezentáló ARMA-alakját!

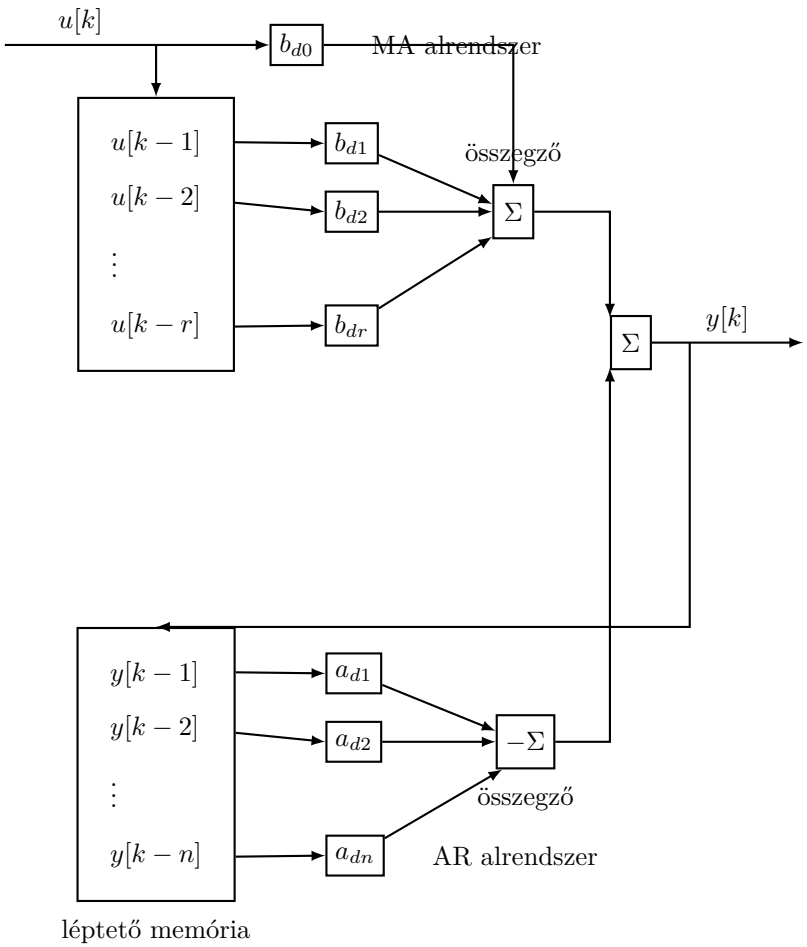
ARMA-alak:

- diszkrét idejű rendszer:
  - $u(t)$ és $y(t)$ csak diszkrét $t = T_k, k \in \mathbb{N}, t \in \{..., T_{-k}, ..., T_{-1}, T_0, T_1, ..., T_k, ...\}$ időpontok sorozatában van értelmezve, ahol: $T_{k-1} < T_k < T_{k+1}$
  - általában $T_k = k \cdot T_s$, ahol T_s a mintavételezési idő
  - $u[k] := u[T_k]$
  - $y[k] := y[T_k]$
- AutoRegresszív Mozgó Átlag
- AR – az $y[k]$ kimenő jel korábbi értékei hogyan hatnak vissza a kimenő jel aktuális értékeire
- MA – az $u[k]$ bemenő jel korábbi értékei (mozgó átlaga) milyen hatással bírnak az aktuális kimenetre
- mi a bemenő és kimenő jel összefüggésének felírására használjuk
- előnyös, mert közvetlenül látszik a memória jelleg, mely léptető (shift) memóriával valósítható meg
- általános alak:

$$\sum_{i=0}^{n} a_{di} y[k-i] = \sum_{i=0}^{r} b_{di} u[k-i]$$

- ezt átrendezve a kimenet új értékének számítása:

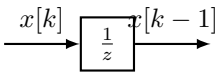
$$y[k] = \sum_{i=0}^{r} b_{di} u[k-i] - \sum_{i=1}^{n} a_{di} y[k-i], \text{ ahol } a_{d0} = 1\text{: az } y[k] \text{ együtthatója}$$



1.6 Ismertess diszkrét rendszerek z eltolási operátorral való képzését! Írja fel az ARMA-alakból az impulzusátviteli függvény képzését!

Eltolási operátor - z:

- egy adatot a jövőbe tol, ezt nem lehet mindig megvalósítani (csak egy múltbeli adatot tudunk a saját jövőjébe tolni), de az inverzének a megvalósítása könnyű
- az eltolási operátor inverze egy időkésleltető elem:



- az eltolási operátor alkalmazása:

$$\begin{aligned} x[k+i] &= z^i x[k] \\ z^{-i}x[k+i] &= x[k] \\ z^{-i}x[k] &= x[k-i] \end{aligned}$$

- az ARMA-modell algebrai alakja:

$$y[k] = \sum_{i=0}^{r} b_{di}u[k-i] - \sum_{i=1}^{n} a_{di}y[k-i]$$

- ARMA-modell az eltolási operátorral:

$$y[k] = \sum_{i=0}^{r} z^{-i}b_{di}u[k] - \sum_{i=1}^{n} z^{-i}a_{di}y[k]$$

- átrendezve:

$$\sum_{i=0}^{n} z^{-i}a_{di}y[k] = \sum_{i=0}^{r} z^{-i}b_{di}u[k] \quad a_{d0} = 1$$

- be- és kimeneti összefüggés eltolási operátorral, ahol $w(z)$ az impulzus átviteli függvény:

$$y(z) = \frac{\sum_{i=0}^{r} z^{-i}b_{di}}{\sum_{i=0}^{n} z^{-i}a_{di}}u(z) = w(z)u(z) \quad a_{d0} = 1$$

1.7 Ismertesse a diszkrét idejű konvolúció szerepét, összefüggését!

Konvolúció:

- bevezetünk ún. „egységnyi” jeleket – vizsgálójeleket
- ebből tudunk következtetni a rendszer tetszőleges bemenetre adott válaszára
- összetett bemenő jelek válaszának meghatározása komponensekre bontással:
  - a bemenőjelet vizsgálójelekből álló komponensekre bontjuk
  - meghatározzuk az elemi komponensek elemi hatását
  - az időinvariancia és a szuperpozíció elvének kihasználásával az elemi hatásokat összegezzük
- diszkrét idejű egységimpulzus:

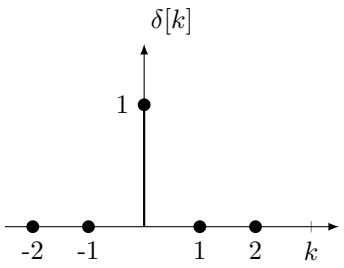
$$\delta[k] = \begin{cases} 0 \text{ ha } k \neq 0 \\ 1 \text{ ha } k = 0 \end{cases} \qquad k \in \mathbb{Z}$$

  - eltolás esetén:

$$\delta[k-K] = \begin{cases} 0 \text{ ha } k \neq K \\ 1 \text{ ha } k = K \end{cases} \qquad k, K \in \mathbb{Z}$$

  - szorzás esetén

$$Y_0\delta[k] = \begin{cases} 0 \text{ ha } k \neq 0 \\ Y_0 \text{ ha } k = 0 \end{cases} \qquad k \in \mathbb{Z}, \quad Y_0 \in \mathbb{R}$$



- diszkrét idejű egységugrás:

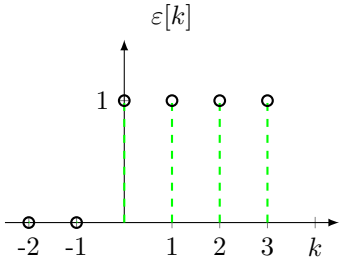
$$\varepsilon[k] = \begin{cases} 0 \text{ ha } k < 0 \\ 1 \text{ ha } k \geq 0 \end{cases} \qquad k \in \mathbb{Z}$$

  - eltolás esetén:

$$\varepsilon[k-K] = \begin{cases} 0 \text{ ha } k < K \\ 1 \text{ ha } k \geq K \end{cases}$$

  - szorzás esetén:

$$Y_0\varepsilon[k] = \begin{cases} 0 \text{ ha } k < 0 \\ Y_0 \text{ ha } k \geq 0 \end{cases} \qquad k, K \in \mathbb{Z} \quad Y_0 \in \mathbb{R}$$



- amikor egy diszkrét idejű rendszernek a bemenő jele egy diszkrét idejű egységimpulzus, akkor a kimenő jelet súlyfüggvénynek (vagy impulzusválasznak) nevezzük
  - jele: $w[k]$

- vizsgálójelek alapösszefüggései:

$$\sum_{i=-\infty}^{\infty} \delta[i] = 1$$

- belátható, hogy egy diszkrét idejű függvény konvolúciója az egységimpulzussal egy adott K érték mellett a függvény K helyen felvett értékét adja vissza

$$y[K] = \sum_{i=-\infty}^{\infty} y[i]\delta[K-i] = \sum_{i=-\infty}^{\infty} y[K-i]\delta[i] \quad K \in \mathbb{Z}$$

- az egységugrás értéke a k helyen kiszámolható az egységimpulzus függvényből

$$\varepsilon[k] = \sum_{i=-\infty}^{k} \delta[i] \quad k \in \mathbb{Z}$$

- az egységugrás két szomszédos helyen felvett értéknek különbsége megegyezik az egységimpulzus függvény megfelelő helyen felvett értékével

$$\delta[k] = \varepsilon[k] - \varepsilon[k-1] \quad k \in \mathbb{Z}$$

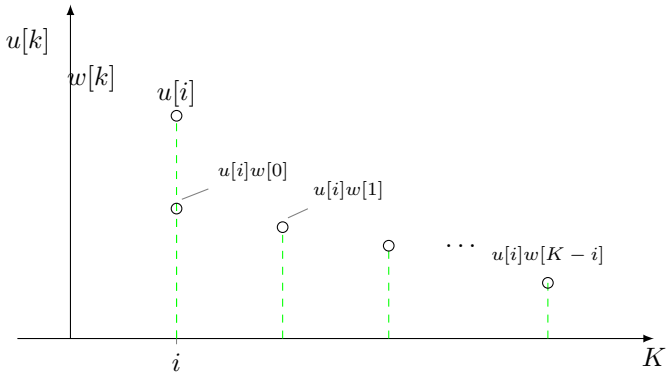
- egy diszkrét idejű rendszer kimenete k-adik időpillanatban a súlyfüggvénnyel és a bemenettel felírva:

$$y[k] = \sum_{i=0}^{k} w[k-i]u[i] \quad k \geq 0$$

- mivel $w[k-i]u[i]$ az i-edik időpillanat bemenetének a kimenetre való hatása

$$u[k = K] = u[k = K]\delta[(k-K) = 0]$$

$$y[k] = \sum_{i=0}^{k} w[k-i]u[i] \quad k > 0$$

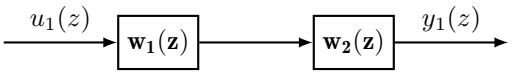


1.8 Adott két diszkrét idejű átviteli függvény. Vezesse le az eredő átviteli függvény összefüggését, ha a két átviteli függvény sorba, párhuzamosan, illetve visszacsatolva (pozitív és negatív) kapcsolódik egymáshoz. Mutassa be, a hatásvázlat átalakításának szabályait (elágazási pont áthelyezése tag mögé és tag elé, illetve összegzési pont áthelyezése tag mögé és tag elé)!

Hatásvázlat:

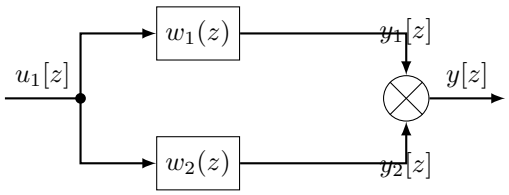
- rendszer működését lehet ábrázolni
- bal oldalt a bemenet(ek), jobb oldalt kimenet(ek)
- grafikus úton kapunk átviteli függvényt
- nyíl irányával jelezzük a jel haladási irányát
- műveletek:
  - elágazás
  - összegzés (negatív esetén az érintett negyed besatírozása)
- az impulzus átviteli függvényeket blokkban ábrázoljuk

Sorba kapcsolás:



$$u_1(z) \cdot w_1(z) \cdot w_2(z) = y_1(z)$$

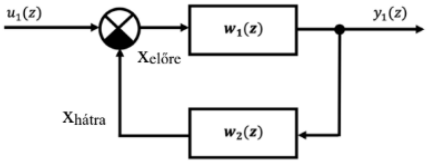
Párhuzamosan kapcsolás:



$$y(z) = y_1(z) + y_2(z) = w_1(z)u_1(z) + w_2(z)u_1(z)$$
$$y(z)(w_1(z) + w_2(z))u_1(z) = w_e(z)u_1(z)$$

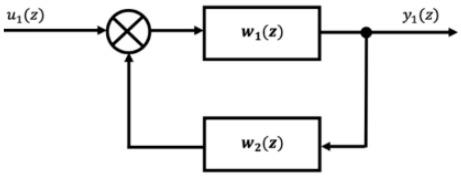
Visszacsatolás, elágazási pontok, összegzési pontok:

- negatív visszacsatolást szoktunk leggyakrabban alkalmazni



$$x_{\text{előre}}(z) = u_1(z) - x_{\text{hátra}}(z) = u_1(z) - w_2(z)y_1(z)$$
$$y_1(z) = x_{\text{előre}}(z)w_1(z) \rightarrow x_{\text{előre}}(z) = \frac{y_1(z)}{w_1(z)}$$
$$y_1(z) = u_1(z)w_1(z) - w_2(z)w_1(z)y_1(z)$$
$$\frac{y_1(z)}{u_1(z)} = \frac{w_1(z)}{1 + w_1(z)w_2(z)}$$

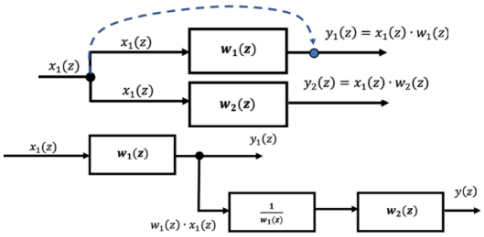
- pozitív visszacsatolás:



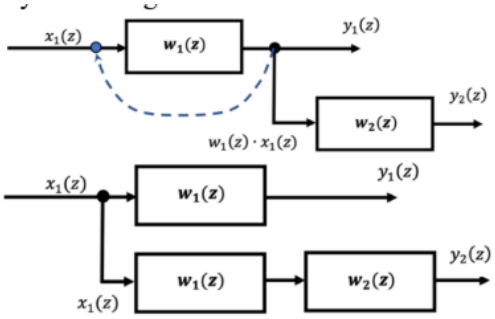
  - a negatív visszacsatoláshoz hasonlóan levezethető, az eredmény:

$$\frac{y_1(z)}{u_1(z)} = \frac{w_1(z)}{1 - w_1(z)w_2(z)}$$

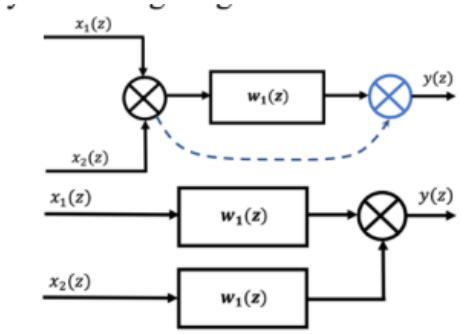
- elágazási pont áthelyezése a tag mögé:



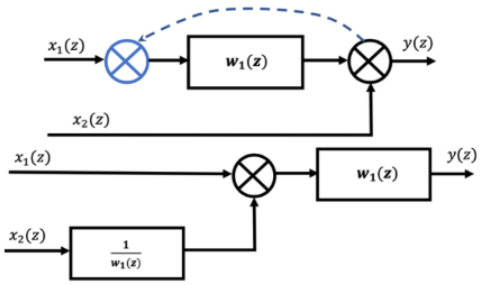
- elágazási pont áthelyezése a tag elé:



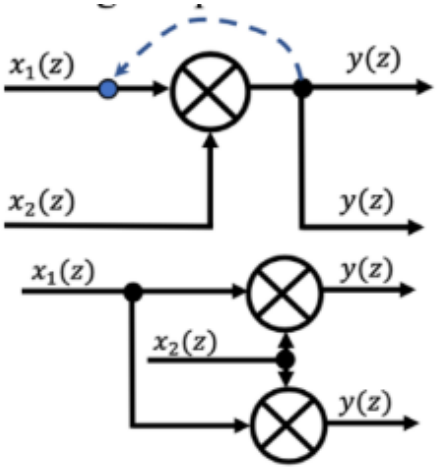
- összegzési pont áthelyezése a tag mögé:



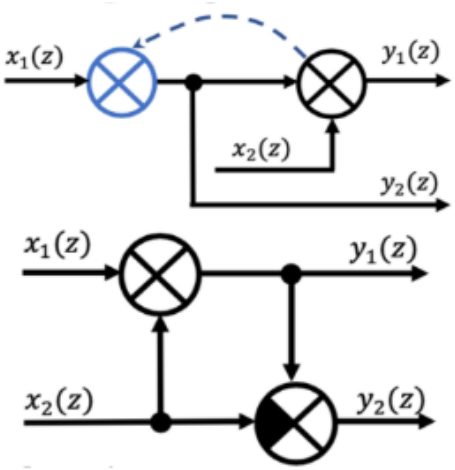
- összegzési pont áthelyezése a tag elé:



- elágazási pont áthelyezése összegzési pont elé:



- összegzési pont áthelyezése elágazási pont elé:



1.9 A lineáris idő invariáns (LTI) rendszerek diszkrét idejű állapottér modell felhasználásával, előre tartó Euler módszer segítésével vezesse le a folytonos idejű lineáris idő invariáns (LTI) rendszerek állapottér modellt!

Lineáris időinvariáns rendszerek diszkrét idejű állapottér modellje:

$$(1.) \quad \underline{x}[k+1] = \underline{\underline{A_d}}\,\underline{x}[k] + \underline{\underline{B_d}}\,\underline{u}[k]$$
$$(2.) \quad \underline{y}[k] = \underline{\underline{C_d}}\,\underline{x}[k] + \underline{\underline{D_d}}\,\underline{u}[k]$$

- az (1.)-es egyenletben vonjunk ki mindkét oldalból $\underline{x}[k]$-t (az előretartó Euler miatt):

$$\underline{x}[k+1] - \underline{x}[k] = \underline{\underline{A_d}}\,\underline{x}[k] + \underline{\underline{B_d}}\,\underline{u}[k] - \underline{x}[k]$$

- összevonás után osszunk le Δt-vel:

$$\frac{\underline{x}[k+1] - \underline{x}[k]}{\Delta t} = \frac{(\underline{\underline{A_d}} - \underline{\underline{I}})\,\underline{x}[k] + \underline{\underline{B_d}}\,\underline{u}[k]}{\Delta t}$$

- a folytonos idejű t_k diszkrét időben a k-adik időpillanatot jelenti, így tudunk helyettesíteni:

$$\frac{\underline{x}(t_{k+1}) - \underline{x}(t_k)}{\Delta t} = \frac{(\underline{\underline{A_d}} - \underline{\underline{I}})\,\underline{x}(t_k) + \underline{\underline{B_d}}\,\underline{u}(t_k)}{\Delta t}$$

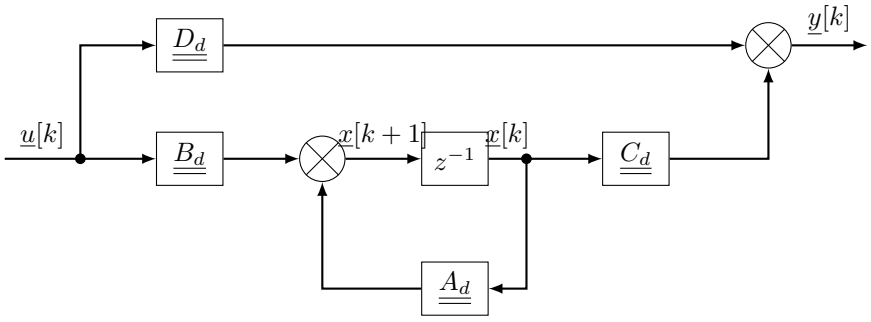
- szétbontva:

$$\frac{\underline{x}(t_{k+1}) - \underline{x}(t_k)}{\Delta t} = \frac{1}{\Delta t}(\underline{\underline{A_d}} - \underline{\underline{I}})\,\underline{x}(t_k) + \frac{1}{\Delta t}\underline{\underline{B_d}}\,\underline{u}(t_k)$$

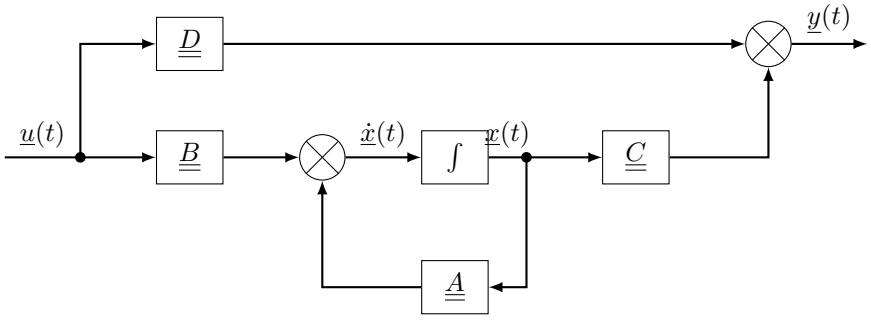
- vegyük a kifejezés határértékét, ha $\Delta t \to 0$:

$$\dot{\underline{x}}(t) = \underline{\underline{A}}\,\underline{x}(t) + \underline{\underline{B}}\,\underline{u}(t)$$
$$\underline{y}(t) = \underline{\underline{C}}\,\underline{x}(t) + \underline{\underline{D}}\,\underline{u}(t)$$

- diszkrét idejű állapottér modell hatásvázlata:



- folytonos idejű állapottér modell hatásvázlata:

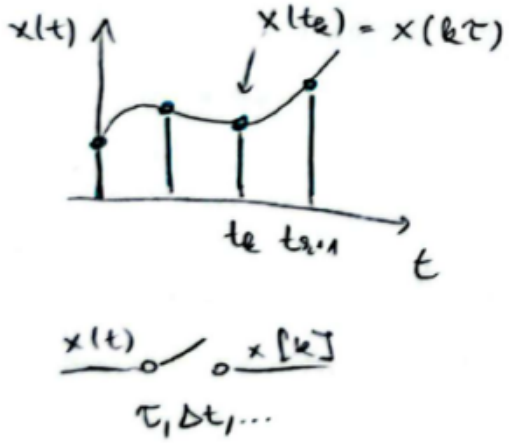


- így a mátrixok:

$$\underline{\underline{A}} = \frac{1}{\Delta t}(\underline{\underline{A_d}} - \underline{\underline{I}}), \quad \underline{\underline{B}} = \frac{1}{\Delta t}\underline{\underline{B_d}}, \quad \underline{\underline{C}} = \underline{\underline{C_d}}, \quad \underline{\underline{D}} = \underline{\underline{D_d}}$$

1.10 A z transzformáció összefüggését felhasználva ismertesse a Laplace transzformáció definícióját!

Mintavételezett jel:



- a mintavételezés eredményét **tömbös formában** lehet tárolni:

$x(0\tau)$	$x(1\tau)$	$x(2\tau)$	...

- ebből matematikai számsorozatot tudunk csinálni:

$$\{x_k\} = \{x_0, x_1, x_2, \dots\}$$

- állítsunk össze polinomot a sorozat elemeiből:

$$a_0 + a_1 x + a_2 x^2 \cdots = \sum_{k=0}^{N} a_k x^k$$

- általánosítás: $N \to \infty \to$ számsorozat $\to$ sor $\to$ hatványsor (függvénysor)

$$\sum_{k=0}^{\infty} a_k x^k \to \sum_{k=0}^{\infty} a_k z^{-k} \to \sum_{k=0}^{\infty} x[k] z^{-k} =: X(z)$$

$$x = z^{-1} \quad \text{Z-transzformáció}$$

- cél: $\Delta t = \tau \to 0$
  - tudjuk, hogy: $z = e^{\ln(z)}$

$$\sum_{k=0}^{\infty} x(k\tau) z^{-k\frac{\tau}{\tau}} \to \sum_{k=0}^{\infty} x(k\tau) e^{-\ln(z) k \frac{\tau}{\tau}}, \quad s := \ln(z)/\tau$$

- így:

$$\sum_{k=0}^{\infty} x(k\tau) e^{-sk\tau}, \quad k\tau \to t$$

- $\tau \to 0\,(\Delta t \to 0)$:

$$\int_0^{\infty} x(t) e^{-st} dt := X(s)$$

Így tehát a Laplace transzformáció:

$$F(s) = \mathcal{L}\{f(t)\} = \int_0^{\infty} f(t) e^{-st} dt$$

- $f(t)$ belépő függvény, tehát $t \in [0, \infty) \to f(t)\varepsilon(t)$

1.11 Igazolja a Dirac impulzus és az egységugrás függvény Laplace transzformáltjait!

Dirac impulzus:

- végtelen nagy impulzus végtelen kicsi idő alatt

$$\delta(t,\tau) = \begin{cases} 0, \text{ ha } t \neq \tau \\ \infty, \text{ ha } t = \tau \end{cases}$$

- MOGI – rendszertechnika jegyzet: 3.15:

$$\int_{-\infty}^{\infty} \delta(t)dt = 1$$

- a Dirac impulzus Laplace-transzformáltja:

$$\mathcal{L}\{\delta(t)\} = \int_0^{\infty} \delta(t)e^{-st}dt = e^{-s\cdot 0} = 1$$

Egységugrás:

$$\varepsilon(t) = \begin{cases} 0, \text{ ha } t < 0 \\ 1, \text{ ha } t \geq 0 \end{cases}$$

$$\mathcal{L}\{\varepsilon(t)\} = \int_0^{\infty} \varepsilon(t)e^{-st}dt = \int_0^{\infty} 1 \cdot e^{-st}dt = \left[-\frac{1}{s}e^{-st}\right]_0^{\infty} = 0 - \left(-\frac{1}{s}e^{0}\right) = \frac{1}{s}$$

1.12 Mutassa be, hogy az átviteli mátrix hogyan származtatható a lineáris idő invariáns (LTI) rendszerek folytonos idejű állapottér modellből!

Lineáris idő invariáns rendszerek folytonos idejű állapottér modellje:

$$\underline{\dot{x}}(t) = \underline{\underline{A}}\,\underline{x}(t) + \underline{\underline{B}}\,\underline{u}(t)$$
$$\underline{y}(t) = \underline{\underline{C}}\,\underline{x}(t) + \underline{\underline{D}}\,\underline{u}(t)$$

- első lépésben Laplace transzformáljuk az egyenleteket:

$$s\underline{X}(s) - \underline{x}(0) = \underline{\underline{A}}\,\underline{X}(s) + \underline{\underline{B}}\,\underline{U}(s)$$
$$\underline{Y}(s) = \underline{\underline{C}}\,\underline{X}(s) + \underline{\underline{D}}\,\underline{U}(s)$$

- az 1. egyenlet átrendezve:

$$(s\underline{\underline{I}} - \underline{\underline{A}})\underline{X}(s) = \underline{x}(0) + \underline{\underline{B}}\,\underline{U}(s)$$

- $\underline{X}(s)$ kifejezése:

$$\underline{X}(s) = (s\underline{\underline{I}} - \underline{\underline{A}})^{-1}\underline{x}(0) + (s\underline{\underline{I}} - \underline{\underline{A}})^{-1}\underline{\underline{B}}\,\underline{U}(s)$$

- $\underline{X}(s)$ visszahelyettesítve a kimeneti egyenletbe, majd rendezve:

$$\underline{Y}(s) = \underbrace{\underline{\underline{C}}(s\underline{\underline{I}} - \underline{\underline{A}})^{-1}\underline{x}(0)}_{\text{tranziens tag}} + \underbrace{(\underline{\underline{C}}(s\underline{\underline{I}} - \underline{\underline{A}})^{-1}\underline{\underline{B}} + \underline{\underline{D}})}_{\text{átviteli mátrix}}\underline{U}(s)$$

- Így az átviteli mátrix ($\underline{U}(s)$ együtthatója)

$$W(s) = \underline{\underline{C}}(s\underline{\underline{I}} - \underline{\underline{A}})^{-1}\underline{\underline{B}} + \underline{\underline{D}}$$

  - a tranziens tag eltűnik, az állandósult viselkedést az átviteli mátrix írja le

1.13 Ismertesse az egytárolós arányos tag súly és átmeneti függvényeit! Válaszában térjen ki az időállandó fogalmára!

Arányos egytárolós tag – PT1:

$$W(s) = \frac{A}{s - p_1} = \frac{K}{1 + sT}$$

Arányos egytárolós tag súlyfüggvénye:

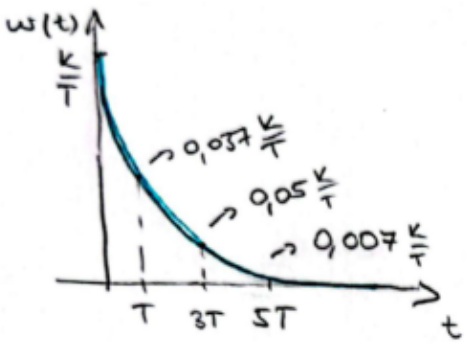
- az egységimpulzusra ($\delta(t)$-re) adott válasz
- $u(t) = \delta(t) \to U(s) = 1$

$$Y(s) = W(s)U(s) = \frac{K}{1 + sT} \cdot 1$$

- a kérdés: $y(t) = w(t) =?$

$$y(t) = w(t) = \frac{K}{T}\mathcal{L}^{-1}\left\{\frac{1}{s + \frac{1}{T}}\right\} = \frac{K}{T}e^{-\frac{t}{T}}\varepsilon(t)$$

- grafikusan ábrázolva:



Arányos egytárolós tag átmeneti függvénye:

- egységugrásra ($\varepsilon(t)$-re) adott válasz
- $u(t) = \varepsilon(t) \to U(s) = 1/s$

$$Y(s) = W(s)U(s) = \frac{K}{1 + sT}\frac{1}{s}$$

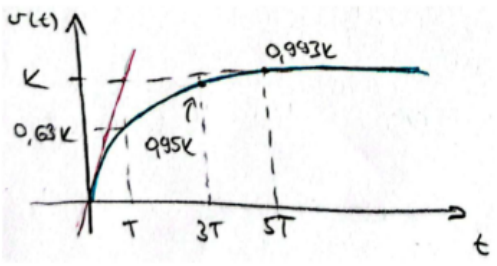
- a kérdés: $y(t) = v(t) =?$

$$y(t) = v(t) = \mathcal{L}^{-1}\left\{\frac{K}{(1 + sT)s}\right\} = K\mathcal{L}^{-1}\left\{\frac{1}{T(s + \frac{1}{T})s}\right\} = \frac{K}{T}\mathcal{L}^{-1}\left\{\frac{A}{s} + \frac{B}{s + \frac{1}{T}}\right\}$$

$$A = T, \quad B = -T$$

$$v(t) = K\mathcal{L}^{-1}\left\{\frac{1}{s} - \frac{1}{s + \frac{1}{T}}\right\} = K\left(1 - e^{-\frac{t}{T}}\right)\varepsilon(t)$$

- grafikusan ábrázolva:



T és K változók az átmeneti függvénynél:

- K meghatározza, hogy hova konvergál a rendszer
- T meghatározza, hogy mennyi idő alatt konvergál

Időállandó fogalma:

- Az átviteli függvény időállandós alakjából s együtthatói, melyet az átviteli függvény gyöktényezős alakjából tudunk átalakítani
- így az időállandó(k) a gyöktényezős alak – mely s-re nézve egy valós együtthatójú racionális törtfüggvény – **pólusai** valós részének reciprokának mínusz egyszerese:

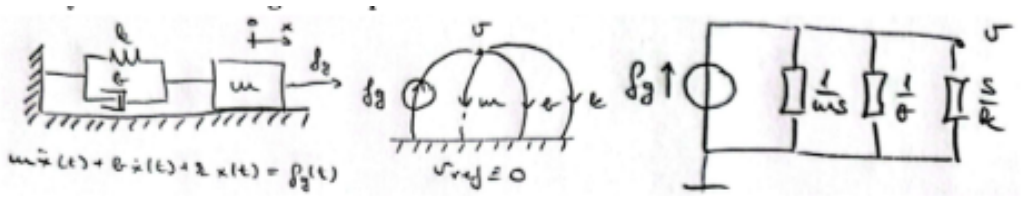
$$W(s) = \frac{b_r}{a_n} \frac{(s-z_1)(s-z_2)\dots(s-z_r)}{(s-p_1)(s-p_2)\dots(s-p_n)}$$

$$W(s) = \underbrace{\frac{b_r(-z_1)(-z_2)\dots(-z_r)}{a_n(-p_1)(-p_2)\dots(-p_n)}}_{\text{erősítés}} \frac{(1-\frac{1}{z_1}s)(1-\frac{1}{z_2}s)\dots(1-\frac{1}{z_r}s)}{(1-\frac{1}{p_1}s)(1-\frac{1}{p_2}s)\dots(1-\frac{1}{p_n}s)}$$

- időállandók: $T_1 = -\frac{1}{p_1}, T_2 = -\frac{1}{p_2}, \dots, T_n = -\frac{1}{p_n}$

1.14 Ismertesse az kéttárolós arányos tag súly és átmeneti függvényeit! Válaszában térjen ki a minőségi jellemzőkre!

Arányos kéttárolós tag – PT2 példával:



$$m\ddot{x}(t) + b\dot{x}(t) + kx(t) = f_g(t)$$

$$V(s) = \frac{1}{ms + b + \frac{k}{s}} F_g(s) = \frac{s}{ms^2 + bs + k} F_g(s)$$

- az elmozdulást keressük, Laplace tartományban az integrálási szabály miatt: $X(s) = \frac{1}{s} V(s)$

$$X(s) = \frac{1}{ms^2 + bs + k} F_g(s)$$

- **átviteli függvény:**

$$W(s) = \frac{1}{ms^2 + bs + k} = \frac{b_0}{a_2 s^2 + a_1 s + a_0} \quad \text{általános algebrai alak}$$

- **gyöktényezős alak:**

$$W(s) = \frac{b_0/a_2}{s^2 + \frac{a_1}{a_2} s + \frac{a_0}{a_2}} = \frac{1/m}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

  ω_n: csillapítatlan sajátkörfrekvencia, ζ: relatív csillapítási tényező

- **időállandós alak:**

$$W(s) = \frac{b_0/a_0}{\frac{a_2}{a_0} s^2 + \frac{a_1}{a_0} s + 1} = \frac{K}{T^2 s^2 + 2\zeta T s + 1}$$

  $T = \frac{1}{\omega_n}$ időállandó, K: erősítési tényező

átmeneti függvény:

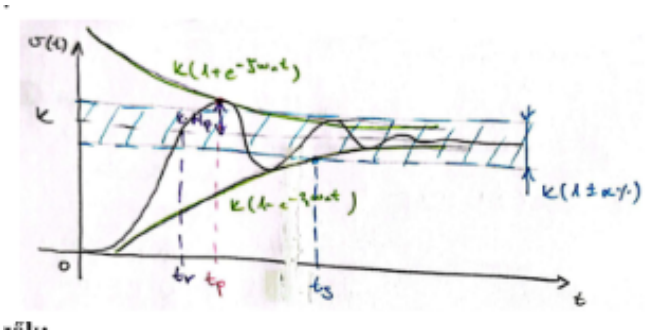
- $u(t) = \varepsilon(t) \to U(s) = 1/s$

$$Y(s) = \frac{K\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} U(s) \to v(t) = y(t) \dots$$

$$v(t) = K\left(1 - e^{-\zeta\omega_n t}\left(\cos(\omega_d t) + \frac{\zeta\omega_n}{\omega_d}\sin(\omega_d t)\right)\right)\varepsilon(t)$$

  ahol $\omega_d = \omega_n\sqrt{1-\zeta^2}$ a csillapított sajátkörfrekvencia.

- **ábrázolva:**



Minőségi jellemzők:

- **belengési idő (peak time):** t_p
  - az átmeneti függvény első maximumának eléréséhez szükséges idő

$$t_p = \frac{\pi}{\omega_d}$$

- **maximális túllövés (max. overshoot):** M_p
  - az átmeneti függvény első maximumának a K állandósult értékhez viszonyított értéke

$$M_p = \frac{v(t_p)}{v(\infty)} - 1 = e^{-\frac{\zeta\omega_n\pi}{\omega_d}} = e^{-\frac{\zeta\pi}{\sqrt{1-\zeta^2}}}$$

  - százalékos túllövés: $P.O. = M_p \cdot 100[\%]$

- **beállási idő (settling time):** t_s
  - az az időpillanat, amikor az átmeneti függvény eléri, és ezt követően már nem hagyja el a $v(\infty) = K$ állandósult érték körül definiált $K(1 \pm \alpha\%)$ szélességű sávot

$$t_s = \frac{1}{\zeta\omega_n} \ln\left(\frac{100}{\alpha}\right)$$

- **felfutási idő (rise time):** t_r
  - a $v(\infty) = K$ állandósult érték első eléréséhez szükséges idő

$$t_r = \frac{\pi - \arccos(\zeta)}{\omega_d}$$

A súlyfüggvény nem volt előadáson.

1.15 Vezesse le a lineáris idő invariáns (LTI) rendszerek folytonos idejű állapottér modelljének megoldását Laplace transzformáció segítségével!

- $u(t)$ bemenet, $y(t)$ kimenet, valamint az őket összekapcsoló folytonos idejű matematikai modell
- Laplace transzformációt végzünk
- $U(s)$ bemenet, $Y(s)$ kimenet, valamint az őket összekapcsoló Laplace tartománybeli matematikai modell
- $W(s)$ átviteli függvényt kirendezzük, $U(s)$-sel szorozzuk
- Inverz Laplace transzformáció
  - ismert elemekre bontás, melyeknek ismerjük az inverz Laplace transzformáltját
  - parciális törtekre bontás
  - polinomosztás – ha a nevező és a számláló fokszáma azonos

Példa:

- az állapottérmodellünk egy $\dot{x}(t)$ differenciálegyenlet, $x(0)$ kezdeti értékkel
- Laplace transzformáljuk:
$$\mathcal{L}\{\dot{x}(t)\} = sX(s) - x(0)$$
- fejezzük ki $X(s)$-t:
$$X(s) = \dots$$
- behelyettesítjük a kezdeti értékeket, illetve a többi megadott adatot
- inverz Laplace transzformáljuk:
$$x(t) = \mathcal{L}^{-1}\{X(s)\} = \dots$$
- megkapjuk a megoldást

1.16 Mutassa be a lineáris idő invariáns (LTI) rendszerek esetén folytonos idejű állapottér modell rendszermátrixának sajátértékei és a rendszer átviteli függvényének pólusai közötti összefüggést!

Ez a fejezet mesterséges intelligencia segítségével készült.

Egy folytonos idejű, lineáris időinvariáns (LTI) rendszer állapottér modellje:

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t)$$
$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t)$$

ahol $\mathbf{A}$ a rendszermátrix, $\mathbf{B}$ a bemeneti mátrix, $\mathbf{C}$ a kimeneti mátrix és $\mathbf{D}$ a közvetlen csatolási mátrix.

Végezzük el a Laplace-transzformációt zérus kezdeti feltételek mellett ($\mathbf{x}(0) = \mathbf{0}$):

$$s\mathbf{x}(s) = \mathbf{A}\mathbf{x}(s) + \mathbf{B}\mathbf{u}(s) \quad \Longrightarrow \quad (s\mathbf{I} - \mathbf{A})\mathbf{x}(s) = \mathbf{B}\mathbf{u}(s)$$
$$\mathbf{y}(s) = \mathbf{C}\mathbf{x}(s) + \mathbf{D}\mathbf{u}(s)$$

Az első egyenletből kifejezve az állapotvektort:

$$\mathbf{x}(s) = (s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}\mathbf{u}(s)$$

Behelyettesítve a kimeneti egyenletbe:

$$\mathbf{y}(s) = \left[\mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D}\right]\mathbf{u}(s)$$

Az átviteli függvény mátrix tehát:

$$\mathbf{H}(s) = \mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D}$$

Tudjuk, hogy az inverz mátrix a következőképpen számítható ki:

$$(s\mathbf{I} - \mathbf{A})^{-1} = \frac{\operatorname{adj}(s\mathbf{I} - \mathbf{A})}{\det(s\mathbf{I} - \mathbf{A})}$$

Az átviteli függvény pólusai azok az s értékek, ahol a nevező zérus, azaz:

$$\det(s\mathbf{I} - \mathbf{A}) = 0$$

Ez pontosan az $\mathbf{A}$ rendszermátrix karakterisztikus egyenlete. Az egyenlet megoldásai az $\mathbf{A}$ mátrix sajátértékei (λ_i).

Összefüggés: Az LTI rendszer átviteli függvényének pólusai megegyeznek a rendszermátrix sajátértékeivel (feltéve, hogy a rendszer minimálrealizáció, azaz nem történik pólus-zérus kiejtés az állapotváltozók közötti irányíthatatlanság vagy megfigyelhetetlenség miatt). Ez az összefüggés alapvető a rendszer stabilitásának vizsgálatánál: a rendszer akkor és csak akkor aszimptotikusan stabil, ha az $\mathbf{A}$ mátrix összes sajátértékének valós része negatív.

1.17 Ismertesse a frekvencia-átviteli függvény fogalmát, illetve annak megjelenítési módjait (Bode diagram)!

A kérdés, hogy egy folytonos idejű időinvariáns rendszer milyen választ ad harmonikus gerjesztésre:

- szinuszos vizsgáló jelet használunk:

$$u(t) = A\sin(\omega t) \to U(s) = \frac{A\omega}{s^2 + \omega^2}$$

- harmonikus gerjesztés esetén a válasz (állandósult értéke):

$$y(t) = \tilde{A}(\omega)\sin(\omega t - \varphi(\omega))$$

  $\tilde{A}(\omega)$: frekvenciafüggő amplitúdó, $\varphi(\omega)$: frekvenciafüggő fáziseltolás

- formális helyettesítés: $s \to j\omega$ komplex szám
  - **frekvencia-átviteli függvény az átviteli függvényből:**

$$W(j\omega) = W(s)|_{s=j\omega}$$

- a komplex szám nagysága:

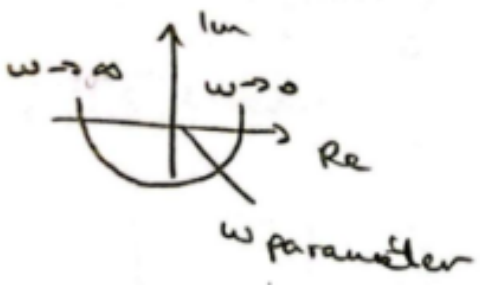
$$|W(j\omega)| = \frac{A_{ki}}{A_{be}} = A_r, \quad \text{ahol } A_{ki} = \tilde{A},\ A_{be} = A,\ A_r : \text{amplitúdó arány}$$

- fáziskülönbség:

$$\Delta\varphi = \varphi_{ki} - \varphi_{be} = \arg(W(j\omega)) = \arctan\left(\frac{\mathrm{Im}\{W(j\omega)\}}{\mathrm{Re}\{W(j\omega)\}}\right)$$

Frekvencia-átviteli függvény megjelenítése:

- **Nyquist-diagram:**



- **Bode diagram(ok):**
  a) amplitúdó – körfrekvencia diagram (teljesítmény arány)
  b) fázisszög – körfrekvencia diagram (fáziskülönbség)

Amplitúdó – körfrekvencia diagram:

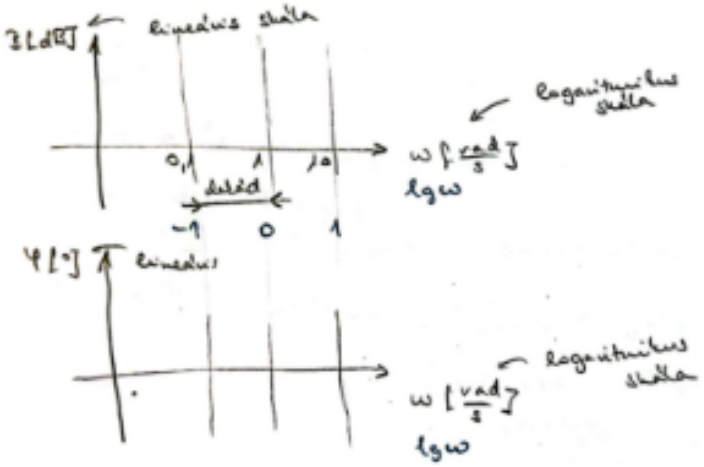
$$A_r \to P_r = A_r^2$$

$$B = \lg(P_r) = \lg(A_r^2)\ [B]\ \text{(bel)}$$

$$B = 10\lg(P_r)\ [dB]\ \text{(decibel)}$$

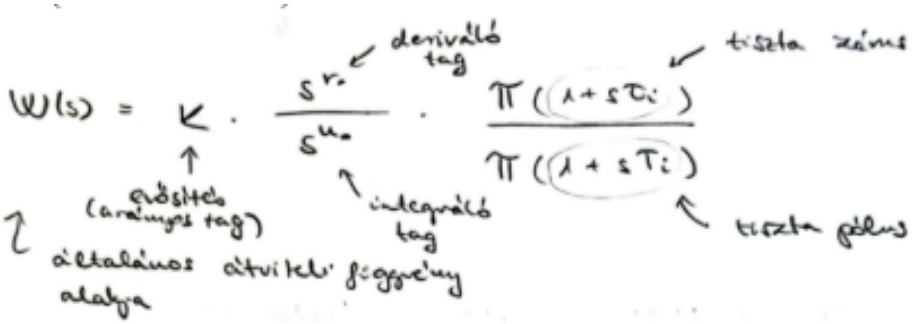
$$B = 10\lg(A_r^2) = 20\lg(A_r)\ [dB]$$

Diagramok:

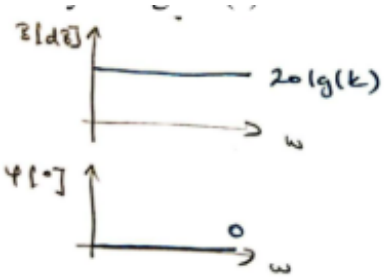


- a frekvencia-átviteli függvény szorzattagokra bontható
- a diagram ezeknek a tagoknak a diagramjának az összegzésével kapható meg, bizonyítás:
  - komplex szám másik alakja: $W(j\omega) = re^{i\varphi}$
  - szorzatokra bontható: $W(j\omega) = W_1(j\omega)W_2(j\omega)$
  - $W(j\omega) = r_1 e^{i\varphi_1} r_2 e^{i\varphi_2} = r_1 r_2 e^{i(\varphi_1+\varphi_2)}$
  - $\lg(r_1 r_2) = \lg(r_1) + \lg(r_2)$

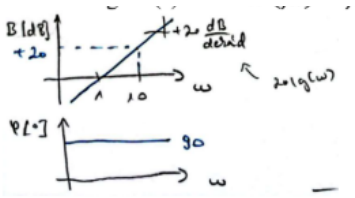
Az alapelemek (általános alak):



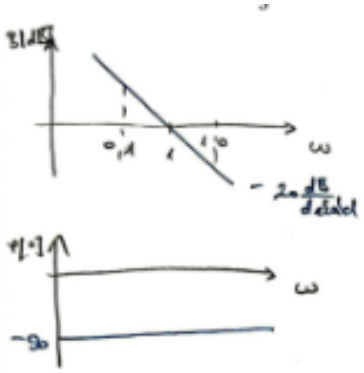
- **Arányos tag:** $W(s) = K$



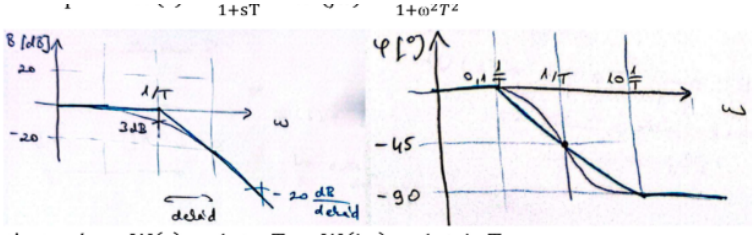
- **Deriváló tag:** $W(s) = s \to W(j\omega) = j\omega$



- **integráló tag:** $W(s) = \frac{1}{s} \rightarrow W(j\omega) = \frac{1}{j\omega} = -j\frac{1}{\omega}$



- **tiszta pólus:** $W(s) = \frac{1}{1+sT} \rightarrow W(j\omega) = \frac{1-j\omega T}{1+\omega^2 T^2}$

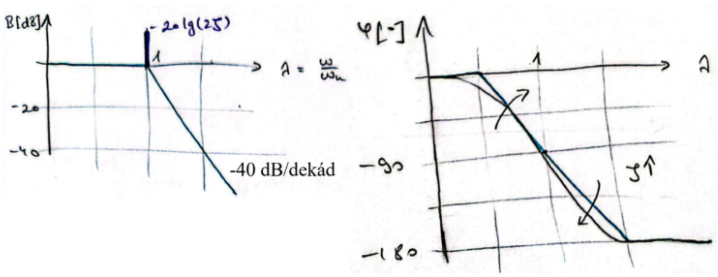


- **tiszta zérus:** $W(s) = 1 + sT \rightarrow W(j\omega) = 1 + j\omega T$
  - tiszta pólus esetén kapott eredmények tükrözése y tengelyre
- **komplex póluspár:** $W(s) = \frac{\omega_n^2}{s^2+2\zeta\omega_n s+\omega_n^2} \rightarrow W(j\omega) = \frac{\omega_n^2}{-\omega^2+2\zeta\omega_n\omega j+\omega_n^2}$

$$W(j\omega) = \frac{1}{1 - \frac{\omega^2}{\omega_n^2} + \frac{2\zeta\omega j}{\omega_n}}$$

  - frekvenciahányados: $\lambda = \frac{\omega}{\omega_n}$

$$W(j\omega) = \frac{1 - \lambda^2 - 2\zeta\lambda j}{((1-\lambda^2)^2 + 4\zeta^2\lambda^2)^2}$$



1.18 Vezesse le a Fourier sorfejtésének alakja komplex alakját!

Ez a fejezet mesterséges intelligencia segítségével készült.

A periodikus $f(t)$ függvény (T periódussal) valós alakú Fourier-sora:

$$f(t) = a_0 + \sum_{k=1}^{\infty} \left(a_k \cos(k\omega_0 t) + b_k \sin(k\omega_0 t)\right)$$

ahol $\omega_0 = \frac{2\pi}{T}$.

Használjuk fel az Euler-formulákat:

$$\cos(k\omega_0 t) = \frac{e^{jk\omega_0 t} + e^{-jk\omega_0 t}}{2}, \quad \sin(k\omega_0 t) = \frac{e^{jk\omega_0 t} - e^{-jk\omega_0 t}}{2j} = -j\frac{e^{jk\omega_0 t} - e^{-jk\omega_0 t}}{2}$$

Behelyettesítve a sorba:

$$f(t) = a_0 + \sum_{k=1}^{\infty} \left[a_k \frac{e^{jk\omega_0 t} + e^{-jk\omega_0 t}}{2} - jb_k \frac{e^{jk\omega_0 t} - e^{-jk\omega_0 t}}{2}\right]$$

$$f(t) = a_0 + \sum_{k=1}^{\infty} \left[\frac{a_k - jb_k}{2} e^{jk\omega_0 t} + \frac{a_k + jb_k}{2} e^{-jk\omega_0 t}\right]$$

Definiáljuk a komplex Fourier-együtthatókat (c_k): - $c_0 = a_0$ - $c_k = \frac{a_k - jb_k}{2}$ ha $k > 0$ - $c_{-k} = \frac{a_k + jb_k}{2}$ (ami éppen c_k konjugáltja, ha $f(t)$ valós)

Ekkor a szummázás kiterjeszthető a negatív indexekre is:

$$f(t) = \sum_{k=-\infty}^{\infty} c_k e^{jk\omega_0 t}$$

Az együtthatók kiszámítása: Tudjuk, hogy $a_k = \frac{2}{T}\int_T f(t)\cos(k\omega_0 t)dt$ és $b_k = \frac{2}{T}\int_T f(t)\sin(k\omega_0 t)dt$. Ebből c_k ($k \neq 0$):

$$c_k = \frac{1}{2}(a_k - jb_k) = \frac{1}{T}\int_T f(t)\left(\cos(k\omega_0 t) - j\sin(k\omega_0 t)\right)dt$$

$$c_k = \frac{1}{T}\int_T f(t) e^{-jk\omega_0 t} dt$$

Ez az összefüggés érvényes $k = 0$ esetén is, hiszen $c_0 = a_0 = \frac{1}{T}\int_T f(t)dt$.

Végeredmény: A komplex Fourier-sor alakja és az együtthatók:

$$f(t) = \sum_{k=-\infty}^{\infty} c_k e^{jk\omega_0 t}, \quad c_k = \frac{1}{T}\int_{-T/2}^{T/2} f(t) e^{-jk\omega_0 t} dt$$

1.19 A Fourier sorfejtés miként általánosítható nem periodikus (lecsengő) függvényekre? Ismertesse a Fourier transzformáció származtatását!

Ez a fejezet mesterséges intelligencia segítségével készült.

A Fourier-sor periodikus függvények felírására alkalmas. Ha egy függvény nem periodikus (lecsengő), tekinthetjük úgy, mint egy olyan periodikus függvényt, amelynek periódusideje a végtelenbe tart ($T \to \infty$).

Kiindulva a komplex Fourier-sorból:

$$f(t) = \sum_{k=-\infty}^{\infty} c_k e^{jk\omega_0 t}, \quad c_k = \frac{1}{T} \int_{-T/2}^{T/2} f(t) e^{-jk\omega_0 t} dt$$

Helyettesítsük be c_k-t a sorba:

$$f(t) = \sum_{k=-\infty}^{\infty} \left[\frac{1}{T} \int_{-T/2}^{T/2} f(\tau) e^{-jk\omega_0 \tau} d\tau \right] e^{jk\omega_0 t}$$

Mivel $\omega_0 = \frac{2\pi}{T}$, ezért $\frac{1}{T} = \frac{\omega_0}{2\pi}$. Jelöljük $\Delta\omega = \omega_0$-t:

$$f(t) = \frac{1}{2\pi} \sum_{k=-\infty}^{\infty} \left[\int_{-T/2}^{T/2} f(\tau) e^{-j(k\Delta\omega)\tau} d\tau \right] e^{j(k\Delta\omega)t} \Delta\omega$$

Ahogy $T \to \infty$, úgy $\Delta\omega \to d\omega$, a diszkrét frekvenciaértékek ($k\Delta\omega$) pedig folytonos változóvá (ω) válnak. A szumma ekkor integrállá alakul:

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} f(\tau) e^{-j\omega\tau} d\tau \right] e^{j\omega t} d\omega$$

Definiáljuk a belső integrált mint a függvény Fourier-transzformáltját:

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

Ekkor az eredeti függvény visszakapható az inverz Fourier-transzformációval:

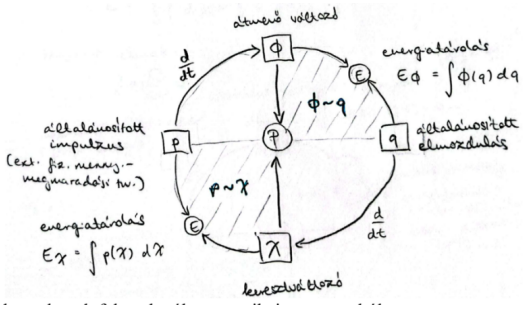
$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(j\omega) e^{j\omega t} d\omega$$

Összefoglalva: A Fourier-transzformáció a Fourier-sor általánosítása nem periodikus jelekre. Míg a Fourier-sor diszkrét frekvencia-spektrumot (vonalas spektrum) eredményez, addig a Fourier-transzformáció folytonos spektrumot ad. A transzformáció létezésének feltétele a Dirichlet-feltételek teljesülése, legfontosabb ezek közül az abszolút integrálhatóság: $\int_{-\infty}^{\infty} |f(t)| dt < \infty$.

1.20 Ismertesse a következő fogalmakat (adja meg a definícióját és rövid értelmezését): extenzív és intenzív fizikai mennyiségek, átmenő és keresztváltozók, energiatárolók (átmenő és keresztváltozóval) és disszipatív elemek (kétpólusok), csatolt kétpólus elem (transzformátor és girátor)!

- **extenzív fizikai mennyiség:** olyan mennyiség, melynek a rendszer egészét jellemző értéke megegyezik az őt alkotó részrendszereket jellemző értékek összegével (pl. tömeg, térfogat, töltés).
- **intenzív fizikai mennyiség:** a rendszer egészét jellemző értéke egyensúly esetén megegyezik a rendszert alkotó részrendszerek értékeivel (pl. hőmérséklet, nyomás).
- **átmenő változók (Φ):** extenzív fizikai mennyiség rátája (d/dt), általában megmaradási törvény is tartozik hozzá.
- **keresztváltozók (χ):** intenzív fizikai mennyiségek (különbsége) vagy általános elmozdulás (q) rátája.

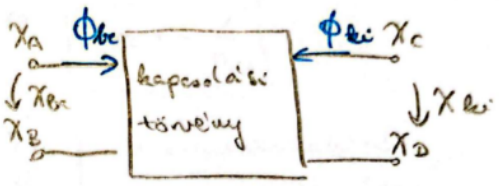
Állapot-tetraéder (Tetrahedron of state):



A rendszerelemek feloszthatók energetikai szempontból:

- **energiatárolók:** energiát tárolnak, az elem valamilyen extenzív mennyiség energiáját halmozza fel, lehet kapacitív vagy induktív.
- **disszipatív elemek:** veszteséget modellezik, a valós rendszerek modellezéséhez szükségesek.
- **energiaátalakítók:** csatolt kétpólus, négypólus elem.
- **energiaforrások**.

Energiaátalakítók:



- **veszteségmentes:** $P_{be} = P_{ki}$, előjelkonvenció miatt $P_{be} + P_{ki} = 0 \implies \chi_{be}\Phi_{be} + \chi_{ki}\Phi_{ki} = 0$
- **lineáris, statikus rendszer** → algebrai egyenlet:

$$\begin{bmatrix} \chi_{be} \\ \Phi_{be} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} \begin{bmatrix} \chi_{ki} \\ \Phi_{ki} \end{bmatrix}$$

$$\chi_{be} = c_{11}\chi_{ki} + c_{12}\Phi_{ki}$$
$$\Phi_{be} = c_{21}\chi_{ki} + c_{22}\Phi_{ki}$$

- **behelyettesítve a teljesítmény-egyenletbe:**

$$(c_{11}\chi_{ki} + c_{12}\Phi_{ki})(c_{21}\chi_{ki} + c_{22}\Phi_{ki}) + \chi_{ki}\Phi_{ki} = 0$$

$$c_{11}c_{21}\chi_{ki}^2 + (1 + c_{11}c_{22} + c_{12}c_{21})\chi_{ki}\Phi_{ki} + c_{12}c_{22}\Phi_{ki}^2 = 0$$

- **2 nem triviális megoldás:**

1. lehetőség: $c_{12} = c_{21} = 0$, ekkor $(1 + c_{11}c_{22}) = 0 \implies c_{22} = -\frac{1}{c_{11}}$

- energiaátalakító: **transzformátor**
- azonos típusú változók között teremt kapcsolatot (pl. villamos transzformátor, hatómű, DC motor)

$$\begin{bmatrix} \chi_{be} \\ \Phi_{be} \end{bmatrix} = \begin{bmatrix} c_{11} & 0 \\ 0 & -\frac{1}{c_{11}} \end{bmatrix} \begin{bmatrix} \chi_{ki} \\ \Phi_{ki} \end{bmatrix}$$

2. lehetőség: $c_{11} = c_{22} = 0$, ekkor $(1 + c_{12}c_{21}) = 0 \implies c_{21} = -\frac{1}{c_{12}}$

- energiaátalakító: fordítóváltó, **girátor**
- eltérő típusú változók között teremt kapcsolatot (pl. giroszkóp, munkahenger)

$$\begin{bmatrix} \chi_{be} \\ \Phi_{be} \end{bmatrix} = \begin{bmatrix} 0 & c_{12} \\ -\frac{1}{c_{12}} & 0 \end{bmatrix} \begin{bmatrix} \chi_{ki} \\ \Phi_{ki} \end{bmatrix}$$

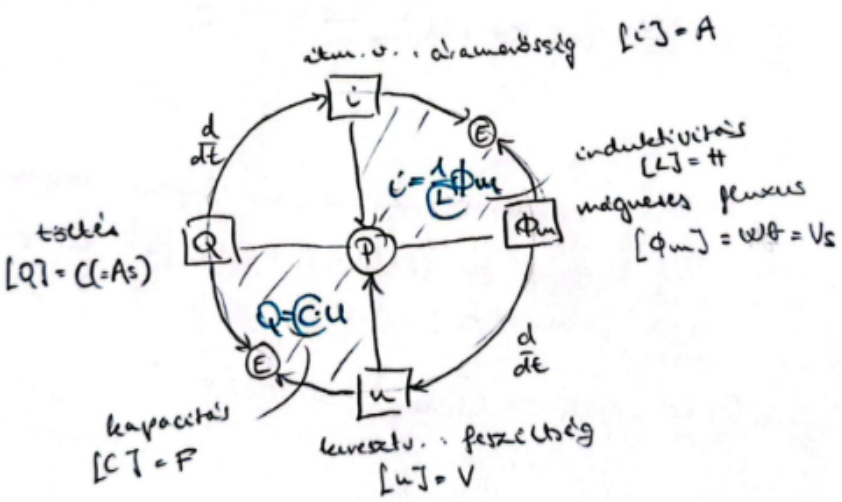
1.21 Adja meg az villamos rendszer (kapcsolt elektromechanikai), haladó és forgómozgású mechanikai rendszerek és az áramlástechnikai (pneumatikus és hidraulikai) rendszerek koncentrált paraméterű leírása esetén az átmenő és keresztváltozó típusát, valamint az energiatárolókat (amennyiben léteznek) és disszipatív elemeket.

Villamos rendszer:

- ϕ: i – áramerősség [A]
- χ: u – feszültség [V]
- energiatárolók:

$$E_\chi = \int Q du = \int C u du = \frac{1}{2} C u^2$$
$$E_\phi = \int i d\phi_m = \int \frac{1}{L} \phi_m d\phi_m = \frac{1}{2L} \phi_m^2 = \frac{1}{2} L i^2$$

- disszipatív elem:
  - ellenállás
  - $i = \frac{1}{R} u$
  - $[R] = \Omega$
- további elemek:
  - L – induktivitás [H]
  - C – kapacitás [F]
  - ϕ_m – mágneses fluxus [Wb] = [Vs]
  - Q – töltés [C] = [As]

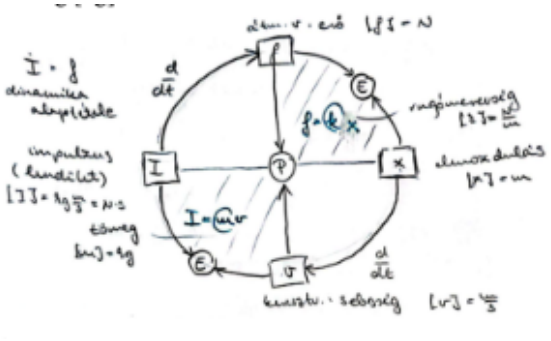


Mechanikai haladó:

- ϕ: f – erő [N]
- χ: v – sebesség [m/s]
- energiatárolók:

$$E_\chi = \int Idv = \int mvdv = \frac{1}{2}mv^2$$
$$E_\phi = \int fdx = \int kxdx = \frac{1}{2}kx^2$$

- disszipatív elem:
  - viszkózus csillapítási tényező
  - $f = bv$
  - $[b] = \mathrm{Ns/m}$
- további elemek:
  - I – lendület [Ns] = [kg m/s]
  - x – elmozdulás [m]
  - k – rugómerevség [N/m]
  - m – tömeg [kg]



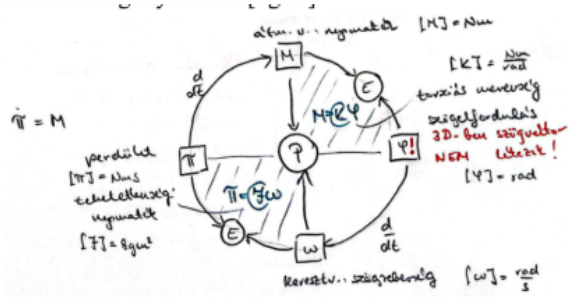
Mechanikai forgó:

- ϕ: M – nyomaték [Nm]
- χ: ω – szögsebesség [rad/s]
- energiatárolók:

$$E_\chi = \int \pi d\omega = \int J\omega d\omega = \frac{1}{2}J\omega^2$$
$$E_\phi = \int Md\varphi = \int k\varphi d\varphi = \frac{1}{2}k\varphi^2$$

- disszipatív elem:
  - viszkózus csillapítási tényező
  - $M = B\omega$
  - $[B] = \mathrm{Nms/rad}$
- további elemek:
  - π – perdület [Nms]
  - φ – szögelfordulás [rad]
  - k – torziós rugómerevség [Nm/rad]
  - J – tehetetlenségi nyomaték [$\mathrm{kgm^2}$]

$\dot{\pi} = M$ $[J] = \text{kgm}^2$ $[k] = \frac{\text{Nm}}{\text{rad}}$ 3D-ben szögvektor NEM létezik! $[\varphi] = \text{rad}$



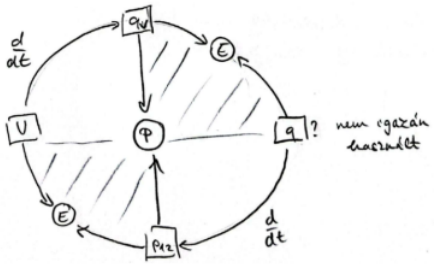
Áramlástechnikai:

- ϕ: q_v – tömegáram/térfogatáram
- χ: p_{12} – nyomás
- energiatárolók:

  fluid kapacitás: $q_v = C_f \frac{dp_{12}}{dt}$

  fluid induktivitás: $q_v = \int_{T_1}^{T_2} \frac{1}{L_f} p dt$

- disszipatív elem:
  - fluid ellenállás
  - $q_v = \frac{1}{R_f} p_{12}$
- áramlástechnikai (fluid) rendszerek:
  - hidraulikus: nem összenyomható közeg
  - pneumatikus: összenyomható közeg, légköri nyomásnál nagyobb nyomás
  - akusztikus: összenyomható közeg, légköri nyomás



nem igazán használt [q?]

hidraulikus rendszer kapacitás:

- nyitott tartály
- tápnyomás: $p_t = \rho g h$
- $q_v = \frac{dV}{dt} = Av \rightarrow V = Ah \rightarrow \frac{dV}{dt} = A\frac{dh}{dt}$
  $\frac{dp}{dt} = \rho g \frac{dh}{dt} \rightarrow \frac{dp}{dt} = \rho g \frac{1}{A}\frac{dV}{dt}$
  $\boxed{q_v = \frac{A}{\rho g}\frac{dp_{12}}{dt}}$ hidraulikus kapacitás C_f

	Pneumatikus	Hidraulikus
kapacitás	$C_f = \frac{V}{\kappa p_1}$	$C_f = \frac{A}{\rho g}$
induktivitás	nincs	$L_f = \frac{\rho l}{A}$
disszipatív elem	R_f	R_f

1.22 Mutassa be, milyen módszerekkel határozható meg a kereszt illetve átmenő változók értékei különféle források figyelembevétele esetén!

Feladatmegoldás:

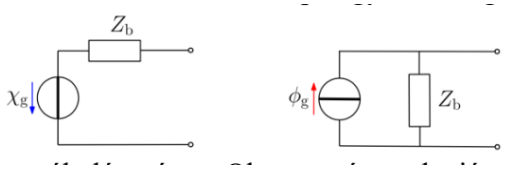
- ha kapunk egy rendszert, hogy oldjuk meg a feladatot?
- első lépés: struktúragráfot készítünk
- a struktúragráf alapján elkészítjük az impedanciahálózatot (itt már Laplace tartományban vagyunk)
- a struktúragráfot redukáljuk
- miután a legegyszerűbb alakra hoztuk, felmerül a kérdés, hogy mit is kell kiszámolnunk, és ez milyen módszerrel lehetséges
- több módszer van, a forrás és a keresett változó típusától függ, hogyan tudjuk megoldani a feladatot

A lehetséges módszerek:

Forrás	Keresett mennyiség	Számítás módja
ϕ_g	χ_i	a) csomóponti potenciálok módszere b) feszültségosztó + forráscsere (Thevenin tétel)
ϕ_g	ϕ_i	a) hurokáramok módszere b) áramosztó
χ_g	χ_i	a) csomóponti potenciálok módszere b) feszültségosztó
χ_g	ϕ_i	a) hurokáramok módszere b) áramosztó + forráscsere (Norton tétel)

Forráscsere:

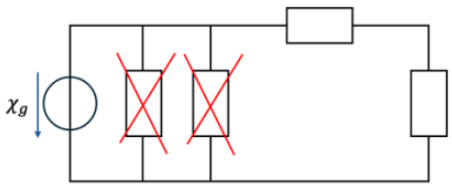
- akkor lehet szükség rá, ha a keresett mennyiség és a forrás típusa különböző
- a Thevenin illetve Norton ekvivalens képe egy tetszőleges hálózatnak:



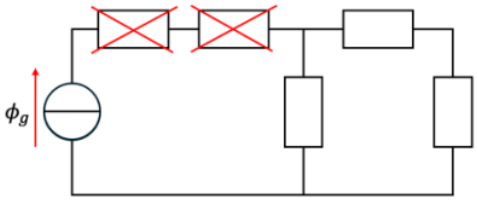
- forráscserénél az általánosított Ohm-törvény alapján a belső ellenállással tudjuk kiszámolni a forráscsere utáni forrásokat ($\chi_g = Z_b\phi_g$)

Elanyagolható impedanciák:

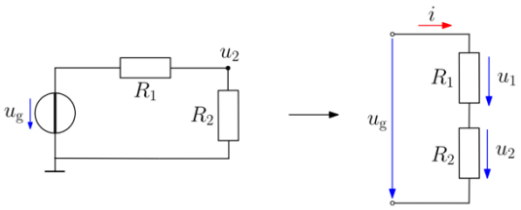
- keresztváltozó forrással párhuzamosan kapcsolt impedanciák:



- átmenő változó forrással sorosan kapcsolt impedanciák:



Feszültségosztó:



- u_2-t keressük
- tudjuk, hogy ugyanaz az áram folyik R_1-en és R_2-n: $i = i_1 = i_2$
- a feszültség pedig megoszlik az ellenállásokon: $u_g = u_1 + u_2$
- az Ohm-törvényt alkalmazva ezekre az egyenletekre:

$$\frac{u_1}{R_1} = \frac{u_2}{R_2}$$

- rendezzük át az egyenletet

$$u_1 = \frac{R_1}{R_2} u_2$$

- helyettesítsünk be u_g-be:

$$u_g = \frac{R_1}{R_2} u_2 + u_2 = \frac{R_1 + R_2}{R_2} u_2$$

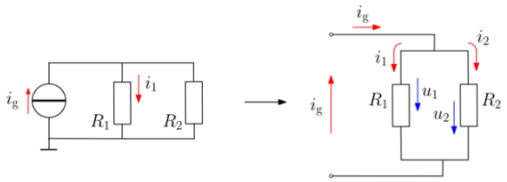
- átrendezve a keresett u_2 feszültség:

$$u_2 = \frac{R_2}{R_1 + R_2} u_g$$

- ez felírható az általánosított Ohm-törvénnyel is:

$$\chi_2 = \frac{R_2}{R_1 + R_2} \chi_g$$

Áramosztó:



- ismert a forrás, az ellenállások értékei és keressük i_1-et
- tudjuk, hogy a párhuzamosan kapcsolt ellenállásokon ugyanaz a feszültség esik:

$$u = u_1 = u_2$$

- az áram pedig az ellenállásoknak megfelelően megoszlik:

$$i_g = i_1 + i_2$$

- felírva az Ohm-törvényt:

$$R_1 i_1 = R_2 i_2 \rightarrow i_2 = \frac{R_1}{R_2} i_1$$

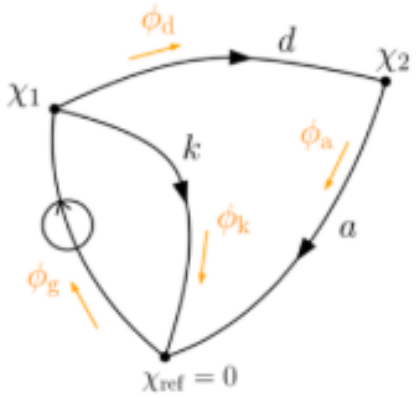
- i_g-be való behelyettesítés és rendezés:

$$i_g = i_1 + \frac{R_1}{R_2} i_1 = \frac{R_1 + R_2}{R_2} i_1 \rightarrow i_1 = \frac{R_2}{R_1 + R_2} i_g$$

- általános Ohm-törvénnyel:

$$\phi_1 = \frac{Z_2}{Z_1 + Z_2} \phi_g$$

Csomóponti potenciálok módszere:



- minden csomópontra felírjuk a Kirchhoff csomóponti törvényt: $\sum i = 0$

$$\chi_1 : \phi_d + \phi_k - \phi_g = \frac{\chi_1 - \chi_2}{Z_d} + \frac{\chi_1 - \chi_3}{Z_k} - \phi_g = 0$$

$$\chi_2 : \phi_a - \phi_d = \frac{\chi_2 - \chi_3}{Z_a} - \frac{\chi_1 - \chi_2}{Z_d} = 0$$

$$\chi_3 : \phi_g - \phi_a - \phi_k = \phi_g - \frac{\chi_2 - \chi_3}{Z_a} - \frac{\chi_1 - \chi_3}{Z_k} = 0$$

- tudjuk, hogy a referencia 0, így leegyszerűsödik az egyenletünk
- az egyenletet átalakítva, majd megoldva:

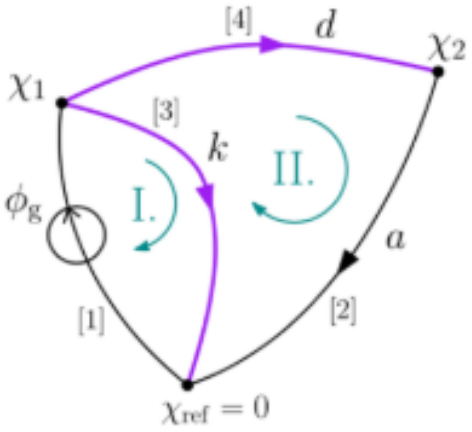
$$(Z_d + Z_a) \cdot \chi_2 = Z_a \cdot \chi_1$$

$$\chi_1 = \frac{Z_d + Z_a}{Z_a} \cdot \chi_2$$

- innen már csak vissza kell helyettesíteni és kifejezni a keresett változót:

$$\chi_2 = \frac{Z_k \cdot Z_a}{Z_a + Z_k + Z_d} \cdot \phi_g \qquad \chi_1 = \frac{Z_d + Z_a}{Z_a} \cdot \frac{Z_k \cdot Z_a}{Z_a + Z_k + Z_d} \cdot \phi_g = \frac{Z_k(Z_a + Z_d)}{Z_a + Z_k + Z_d} \cdot \phi_g$$

Hurokáramok módszere:



- feszítőfát kell választani: olyan hurokmentes részgráf, mely minden csomópontot tartalmaz
- a feszítőfa ágai és a kötőágak hurkokat határoznak meg, melyek irányait az irányított kötőágak szabnak meg
- minden hurokra felírjuk a Kirchhoff huroktörvényt: miszerint az egy hurkon belül a feszültségek előjeles összege 0: $\sum u = 0$

- felírjuk a hurokegyenleteket (pozitív irány: az ág iránya egyezik a hurokéval):

$$\mathrm{I} : \chi_g + \chi_k = 0$$

$$\mathrm{II} : \chi_a - \chi_k + \chi_d = 0$$

- kifejezzük a feszültségeket a hurokáramok segítségével, az általános Ohm-törvény felírásával:

$$Z = \frac{\chi}{\phi}$$

$$\chi_d = Z_d \cdot \phi_d = Z_d \cdot \phi_{\mathrm{II}}$$
$$\chi_a = Z_a \cdot \phi_a = Z_a \cdot \phi_{\mathrm{II}}$$
$$\chi_k = Z_k \cdot \phi_k = Z_k \cdot (\phi_{\mathrm{I}} - \phi_{\mathrm{II}})$$

- visszaírva a hurokegyenletbe a 2-es ág esetén:

$$\mathrm{II} : Z_a \cdot \phi_{\mathrm{II}} - Z_k \cdot (\phi_{\mathrm{I}} - \phi_{\mathrm{II}}) + Z_d \cdot \phi_{\mathrm{II}} = Z_a \cdot \phi_{\mathrm{II}} - Z_k \cdot \phi_{\mathrm{I}} + Z_k \cdot \phi_{\mathrm{II}} + Z_d \cdot \phi_{\mathrm{II}} = 0$$

$$(Z_a + Z_k + Z_d) \cdot \phi_{\mathrm{II}} = Z_k \cdot \phi_{\mathrm{I}}$$

- az 1-es ágban a forrás előírja az abban az ágban folyó áramot, $\phi_{\mathrm{I}} = \phi_g$, így:

$$\phi_{\mathrm{II}} = \frac{Z_k}{Z_a + Z_k + Z_d} \cdot \phi_g$$

- mivel a hurokáramok ismertek, kiszámolhatók belőlük a feszültségek az általános Ohm-törvénnyel

$$\chi_2 = \chi_a = \frac{Z_a \cdot Z_k}{Z_a + Z_k + Z_d} \cdot \phi_g$$

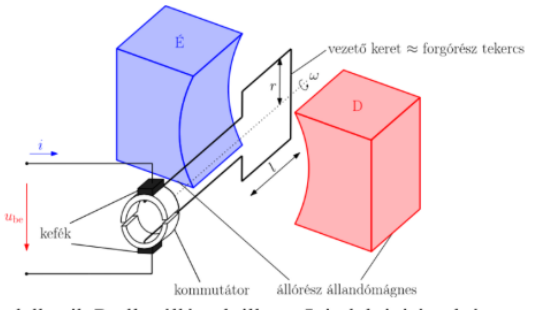
$$\chi_1 = \chi_k = Z_k \cdot \left(1 - \frac{Z_k}{Z_a + Z_k + Z_d}\right) \cdot \phi_g = \frac{Z_k(Z_a + Z_d)}{Z_a + Z_k + Z_d} \cdot \phi_g$$

- az éleken folyó áramok pedig kiszámolhatók a már ismert hurokáramok előjeles összegeiből

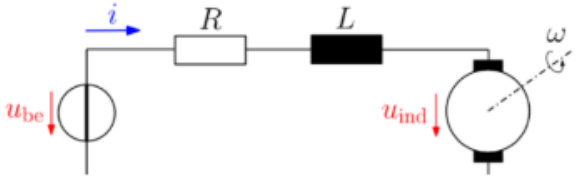
1.23 Egy adott, tanult példa (egyenáramú motor) kapcsán ismertesse a struktúra gráf és az impedancia hálózat felrajzolásának lépéseit. Milyen feltételek teljesülése esetén és hogyan lehet csatolt kétpólus elemmel összekapcsolt rendszereket egy oldalra redukálni? Válaszában térjen ki a rendszerek közötti átjárásokat biztosító fizikai összefüggésekre is!

Egyenáramú motor:

- villamos energiát mechanikus energiává képes alakítani, vagy fordítva: generátor
- leggyakrabban állómágnest tartalmaz, az ez által létrehozott mezőben: tekercselt forgórész



- a tekercs rendelkezik R ellenállással, illetve L induktivitással, így a villamos hálózattal a következő módon írható le a rendszer:



- a huroktörvényt alkalmazva felírható a rendszerre a következő egyenlet:

$$u_{\text{be}}(t) = Ri(t) + L\frac{di(t)}{dt} + u_{\text{ind}}(t)$$

- a fizikai összefüggésekből levezethető, hogy:

$$u_{\text{ind}} = k_e\omega(t), \quad \text{valamint} \quad M_{\text{vill}}(t) = k_m i(t)$$

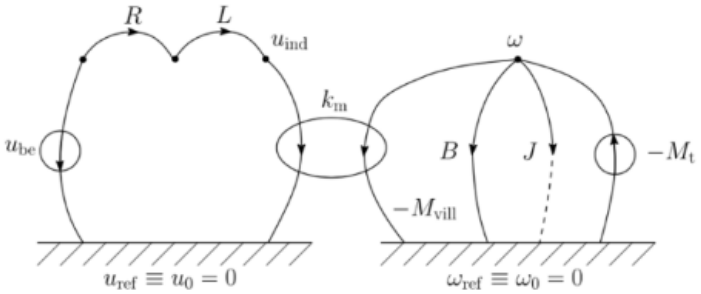
  - ahol k_e a motor sebességállandója, k_m pedig a nyomatékállandó
  - a két mennyiség értéke SI-ben azonos

- írjuk fel a motorra a dinamika alaptételét:

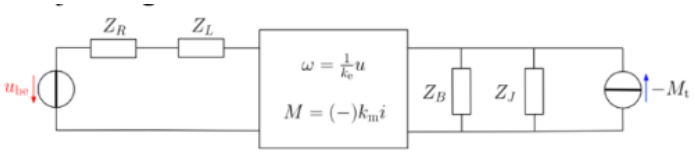
$$J\frac{d\omega(t)}{dt} = M_{\text{vill}}(t) - B\omega(t) - M_t(t)$$

  - J: a motor tehetetlenségi nyomatéka
  - B: viszkózus csillapítási tényező
  - $M_t(t)$: terhelő nyomaték

DC motor struktúragráfja és impedanciahálózata:



- M_t nyomaték előjele negatív, mivel a terhelés csökkenti a fordulatszámot



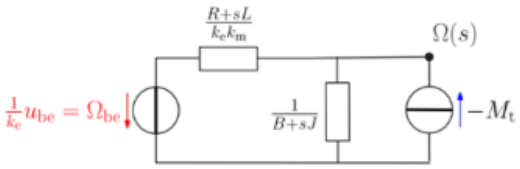
- az impedanciák értékei (Laplace tartományban):
  - $Z_R = R$
  - $Z_L = sL$
  - $Z_B = \frac{1}{B}$
  - $Z_J = \frac{1}{Js}$
- akkor lehet a kapcsolt kétpólus oldalait redukálni, ha:
  - veszteségmentes: $P_{\text{be}} = P_{\text{ki}}$, előjelkonvenció miatt $P_{\text{be}} + P_{\text{ki}} = 0$
  - lineáris, statikus rendszer – algebrai egyenletekkel leírható
  - a transzformátor egyenletei:

$$\begin{bmatrix} \chi_{12} \\ \phi_1 \end{bmatrix} = \begin{bmatrix} c_{11} & 0 \\ 0 & -\frac{1}{c_{11}} \end{bmatrix} \begin{bmatrix} \chi_{34} \\ \phi_2 \end{bmatrix} \implies \begin{bmatrix} u_{\text{ind}} \\ i \end{bmatrix} = \begin{bmatrix} k_e & 0 \\ 0 & -\frac{1}{k_m} \end{bmatrix} \begin{bmatrix} \omega \\ M_{\text{vill}} \end{bmatrix}$$

  - impedanciák:

$$Z_{\text{mech}} = \frac{\Omega(s)}{M(s)} = \frac{\frac{1}{k_e}U(s)}{k_m I(s)} = \frac{1}{k_e k_m}\frac{U(s)}{I(s)} = \frac{1}{k_e k_m} Z_{\text{vill}}$$

- a soros és párhuzamos impedanciákat össze tudjuk vonni:
  - $Z_{e,RL} = R + sL$
  - $Z_{e,BJ} = \frac{1}{B+Js}$
- redukálás:



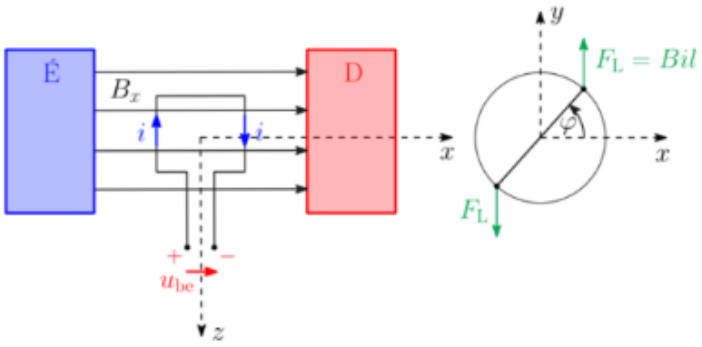
- innen már a tanult módszerek segítségével tudunk számolni

Fizikai összefüggések:

- **indukált feszültség:**
  - a mágneses fluxus változásából írhatjuk le - **Faraday törvény:**

$$u_{\text{ind}}(t) = \frac{d\Phi(t)}{dt} = B_x \frac{dA(t)}{dt} = B_x \frac{d(l \cdot 2r \sin\varphi(t))}{dt} = (B_x l \cdot 2r \cos\varphi(t))\omega(t)$$

  - az indukált feszültséget és a szögsebességet összekötő tagot elnevezzük sebességállandónak, jele: k_e
- **motor villamos nyomatéka:**
  - a Lorentz-erőből adódó forgatónyomatékkal számolható
  - arányos az áramerősséggel



- a Lorentz-erő:

$$\underline{F} = i(\underline{l} \times \underline{B}) = i \begin{bmatrix} 0 \\ 0 \\ -l \end{bmatrix} \times \begin{bmatrix} B_x \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ -iB_x l \\ 0 \end{bmatrix}$$

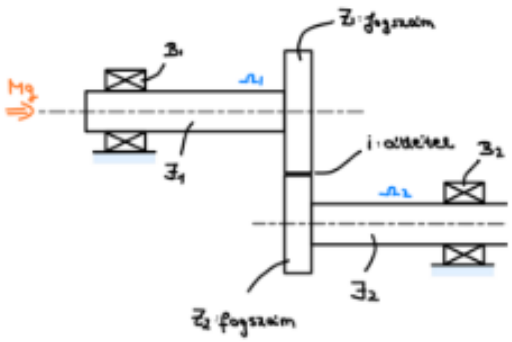
- a Lorentz-erőből számolt nyomaték:

$$\underline{M} = \underline{r} \times \underline{F} = \begin{bmatrix} r\cos\varphi \\ r\sin\varphi \\ 0 \end{bmatrix} \times \begin{bmatrix} 0 \\ F \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ rF\cos\varphi \end{bmatrix}$$

- $M_{\text{vill}} = 2M = (2B_x l \cdot r\cos\varphi(t))i$
- az áramot és a nyomatékot összekötő tagot elnevezzük nyomatékállandónak, jele: k_m

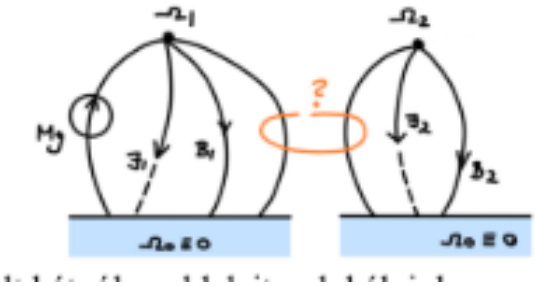
1.24 Egy adott, tanult példa (fogaskerékhajtómű, fogaskerék-fogasléc) kapcsán ismertesse a struktúra gráf és az impedancia hálózat felrajzolásának lépéseit. Milyen feltételek teljesülése esetén és hogyan lehet csatolt kétpólus elemmel összekapcsolt rendszereket egy oldalra redukálni? Válaszában térjen ki a rendszerek közötti átjárásokat biztosító fizikai összefüggésekre is!

Fogaskerék-hajtómű:

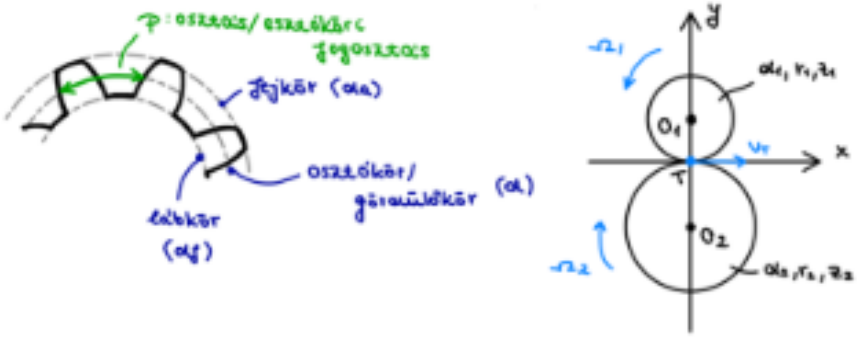


Ide már azt rakunk, amit akarunk. (pl. tengelykapcsoló stb.)

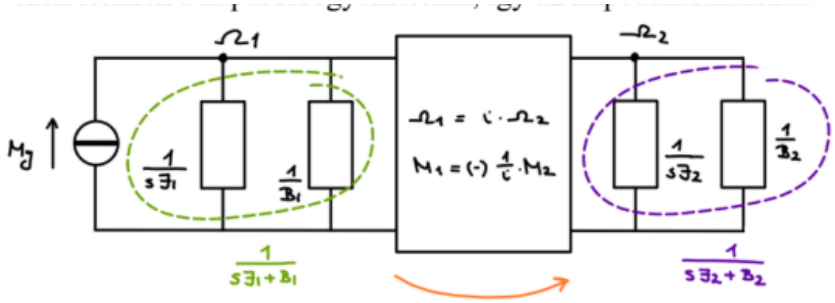
- **struktúragráf:**



- akkor lehet a kapcsolt kétpólus oldalait redukálni, ha:
  - veszteségmentes: $P_{\text{be}} = P_{\text{ki}}$, előjelkonvenció miatt $P_{\text{be}} + P_{\text{ki}} = 0$
  - lineáris, statikus rendszer - algebrai egyenletekkel leírható
- **fizikai összefüggések a kapcsolóegyenlethez:**
  - $d\pi = pz \rightarrow d = \frac{p}{\pi} z$
  - a kerületi sebességek: $v_T = \Omega_1 r_1 = \Omega_2 r_2 \rightarrow \Omega_1 = \frac{r_2}{r_1}\Omega_2 = \frac{z_2}{z_1}\Omega_2 = i\Omega_2$
  - nyomaték: $M = fr \rightarrow \left.\begin{matrix} M_1 = f_T r_1 \\ M_2 = -f_T r_2 \end{matrix}\right\} M_1 = -\frac{r_1}{r_2} M_2 = -\frac{1}{i} M_2$



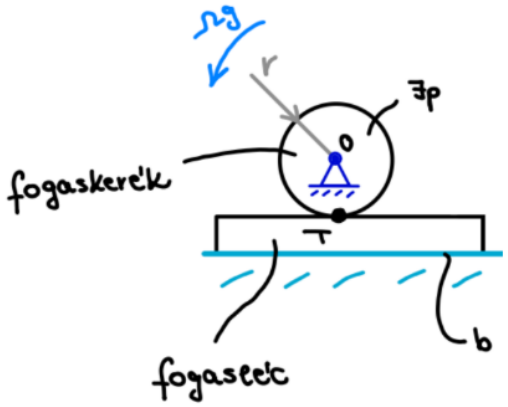
- ezek lesznek a kapcsolóegyenleteink, így az impedanciahálózat:



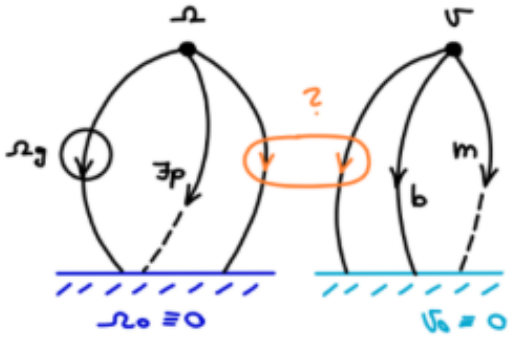
- **redukált impedanciák:**

$$Z_{\text{forg1}} = \frac{\Omega_1}{M_1} = \frac{i\Omega_2}{\frac{1}{i}M_2} = i^2\frac{\Omega_2}{M_2} = i^2 Z_{\text{forg2}}$$

Fogaskerék – fogasléc:



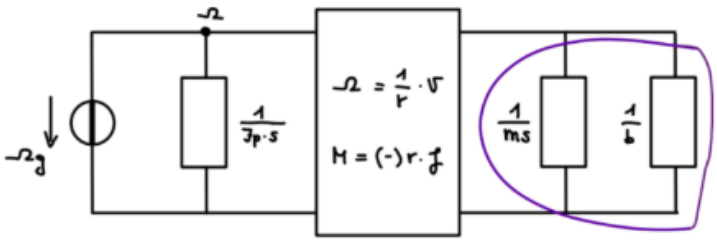
- **struktúragráf:**



- **fizikai összefüggés:**

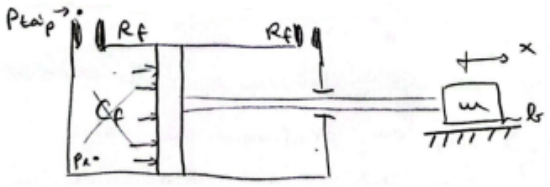
$$\Omega_g = \frac{1}{r}v_T, \quad M = -r f_T$$

- **így az impedanciahálózatunk:**

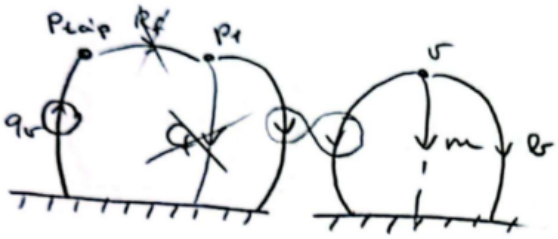


1.25 Egy adott, tanult példa (hidraulikus és pneumatikus munkahenger) kapcsán ismertesse a struktúra gráf és az impedancia hálózat felrajzolásának lépéseit. Milyen feltételek teljesülése esetén és hogyan lehet csatolt kétpólus elemmel összekapcsolt rendszereket egy oldalra redukálni? Válaszában térjen ki a rendszerek közötti átjárásokat biztosító fizikai összefüggésekre is!

Hidraulikus munkahenger:



- nincs C_f mert a tartály zárt, valamint nem összenyomható a közeg
- **struktúragráf:**

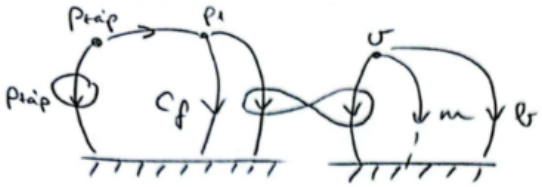


- **Megjegyzés:** átmenő változó generátorral sorosan kapcsolt impedancia elhanyagolható: R_f
- akkor lehet a kapcsolt kétpólus oldalait redukálni, ha:
  - veszteségmentes: $P_{\text{be}} = P_{\text{ki}}$, előjelkonvenció miatt $P_{\text{be}} + P_{\text{ki}} = 0$
  - lineáris, statikus rendszer – algebrai egyenletekkel leírható
- **fizikai összefüggések:** $p_{12} = (-)\frac{f}{A}, q_v = Av$
- **így a girátor:**

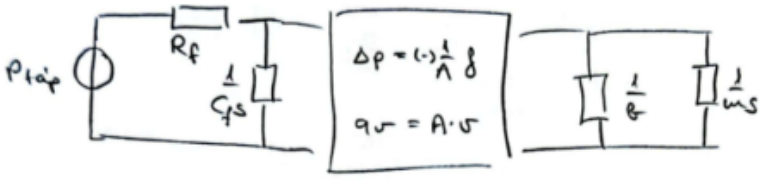
$$\begin{bmatrix} p_{12} \\ q_v \end{bmatrix} = \begin{bmatrix} 0 & (-)\frac{1}{A} \\ A & 0 \end{bmatrix} \begin{bmatrix} v \\ f \end{bmatrix}$$

Pneumatikus munkahenger:

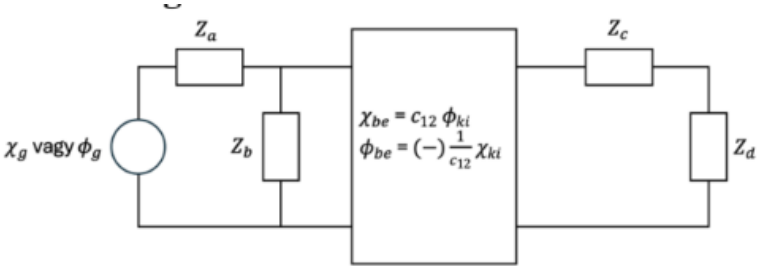
- mivel a gáz összenyomható: van C_f
- a kompresszor tápnyomást állít elő → keresztváltozó forrás
- **struktúragráf:**



- a fizikai összefüggések ugyanazok, mint a hidraulikus rendszernél
- **impedanciahálózat:**



Impedanciahálózat redukálása girátorral:



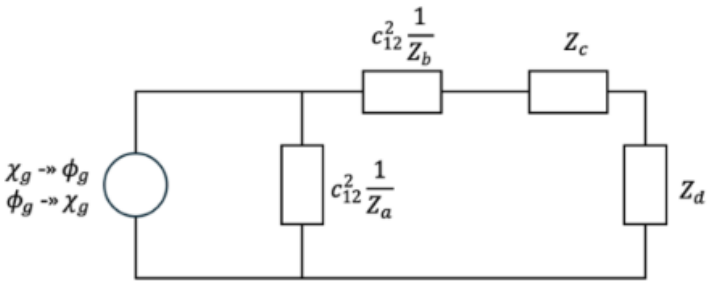
- **forrás redukálása:**

$$\left.\begin{aligned}\chi_{\text{ki}} &= c_{12}\phi_{\text{be}}\\ \phi_{\text{ki}} &= \tfrac{1}{c_{12}}\chi_{\text{be}}\end{aligned}\right\}\text{a forrás típusa is megváltozik}$$

- **impedanciák redukálása:**

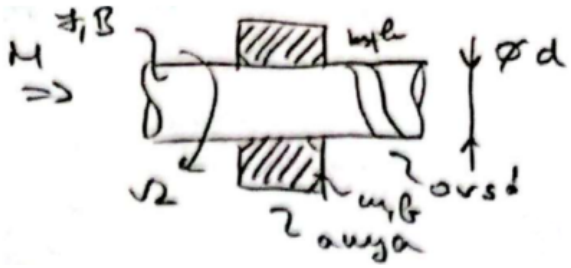
$$Z_{\text{ki}} = \frac{\chi_{\text{ki}}}{\phi_{\text{ki}}} = \frac{c_{12}\phi_{\text{be}}}{\frac{1}{c_{12}}\chi_{\text{be}}} = c_{12}^2\frac{1}{Z_{\text{be}}}$$

  - az impedanciák kapcsolása is megváltozik: soros ↔ párhuzamos



1.26 Egy adott, tanult példa (golyósorsó és vonóelem) kapcsán ismertesse a struktúra gráf és az impedancia hálózat felrajzolásának lépéseit. Milyen feltételek teljesülése esetén és hogyan lehet csatolt kétpólus elemmel összekapcsolt rendszereket egy oldalra redukálni? Válaszában térjen ki a rendszerek közötti átjárásokat biztosító fizikai összefüggésekre is!

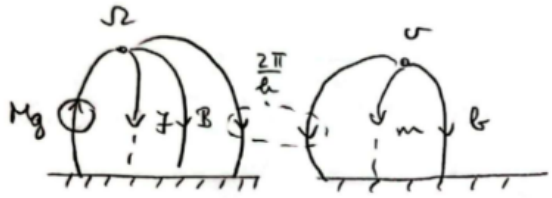
Golyóorsó:



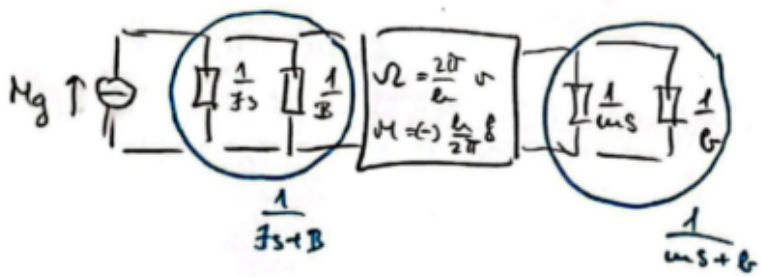
- **fizikai összefüggés:** amíg az orsó egy teljes fordulatot (2π radián) megtesz, addig az anya h-t halad rajta: $\varphi = \frac{2\pi}{h}x \to \frac{d}{dt} \to \omega = \frac{2\pi}{h}v$
- akkor lehet a kapcsolt kétpólus oldalait redukálni, ha:
  - veszteségmentes: $P_{\text{be}} = P_{\text{ki}}$, előjelkonvenció miatt $P_{\text{be}} + P_{\text{ki}} = 0$
  - lineáris, statikus rendszer – algebrai egyenletekkel leírható
- **így a transzformátor kapcsolóegyenletei:**

$$\omega = \frac{2\pi}{h}v, \quad M = (-)\frac{h}{2\pi}f$$

- **struktúragráf:**



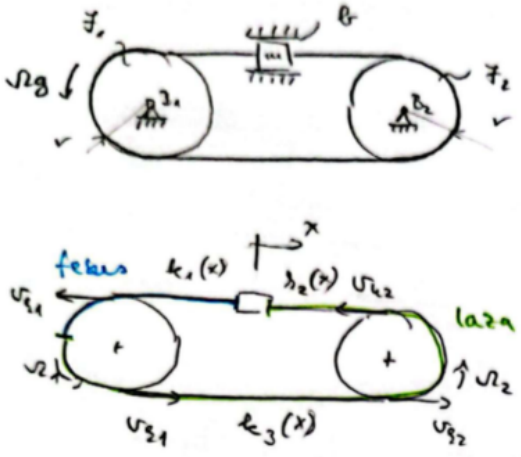
- **impedanciahálózat:**



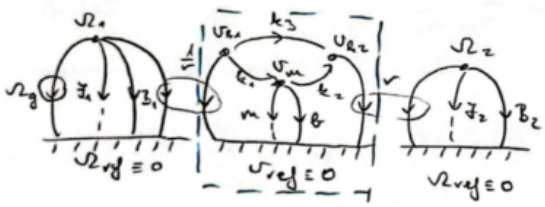
- **impedanciák redukálása:**

$$Z_{\text{forgó}} = \frac{\Omega}{M} = \left(\frac{2\pi}{h}\right)^2 \frac{v}{f} = \left(\frac{2\pi}{h}\right)^2 Z_{\text{haladó}}$$

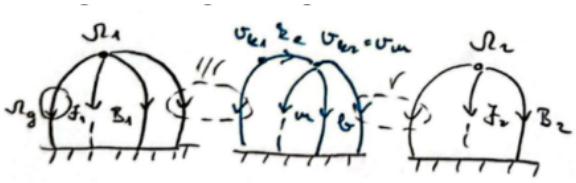
Vonóelem:



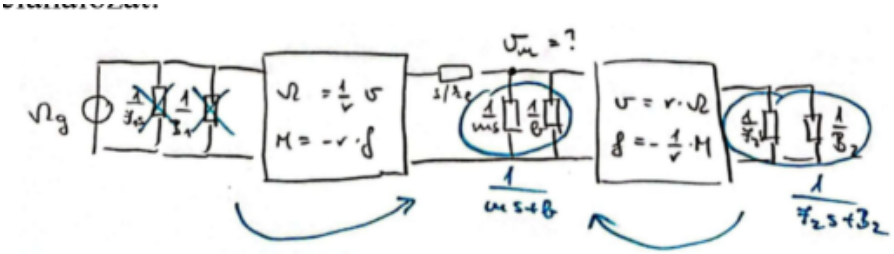
- **fizikai összefüggés:**
- de mivel az ágaknak van rugómerevségük, muszáj két transzformátorral dolgozni: $\Omega_1 = \frac{1}{r_1} v_1$, majd $v_2 = r_2 \Omega_2$
- a húzott ág rugómerevségét megkülönböztetjük a tehetetlentől, de ezek a struktúragráfon összevonhatók: $k_e = k_T + k_{\text{sz}}$ (párhuzamos rugók eredője)
- **struktúragráf:**



- v_m-et közelítsük v_{k2}-vel



- így lesz egy eredő rugómerevségünk: k_3 párhuzamosan van kötve soros k_1 és k_2-vel
- **impedanciahálózat:**



- **impedanciák redukálása:**

$$Z_{\text{haladó}} = \frac{r\Omega}{\frac{1}{r}M} = r^2 Z_{\text{forgó}}$$

2 Informatika

2.1 A számítástudomány alapjai. Turing gép. Eljárások, algoritmusok.

Elméleti számítástudomány: matematikai
Számítástechnika: elméleti és gyakorlati megvalósítás, technológia
Turing gép: elméleti „számítógép". Részei:

- **végtelen, cellákra osztott szalag**
  - egy cellában lehet szimbólum, vagy lehet üres
  - az adatok, műveletek, illetve eredmények cellái véges számúak, ezeken túl a szalag üres
- **Író/olvasó fej:**
  - egyszerre egy cellával foglalkozik
  - írhat, olvashat, törölhet
  - a szalagon jobbra/balra lépkedhet (változtatás nélkül)
- **Vezérlőegység:**
  - állapotai számozva
  - véges állapot (véges állapotú automata)
  - helyettesítési táblázat adja meg a működést
    - állapot + művelet + adat → új állapot + eredmény + fejmozgás

Turing gép:

- matematikailag 5-10 elemből álló szabályhalmaz
- informatikailag:
  - szalag = memória
  - vezérlőegység = CPU
  - fej = busz
- alkalmas rekurzióra → veremtár (stack)

Eljárások, algoritmusok:

- emberi nyelven megfogalmazott feladat, cselekménysorozat
- megoldási eljárás („algoritmus jelölt"), program írható
- egy megoldási eljárás akkor **algoritmus**, ha **bármilyen** bemenet esetén **véges számú lépés** után eredményt kapunk (A Turing-gép megáll)
- az algoritmusra nem létezik formális matematikai definíció

2.2 A számítógép architektúrák alapjai. Boole függvények. Logikai kapuk. Kombinációs és szekvenciális logikai hálózatok. Tárolók: S-R, J-K, D.

Neumann elvek (1945):

- teljesen elektronikus működés
- kettes számrendszer használata
- szekvenciális művelet végrehajtás
- adatok és programok a belső memóriában
- univerzális felhasználás: Turing-gép
- öt funkcionális egység:
  - aritmetikai egység
  - központi vezérlőegység
  - memóriák
  - bemeneti-,
  - és kimeneti egységek

Boole függvények:

- olyan matematikai függvények, melyek vagy 1, vagy 0 értéket vesznek fel
- bemenetei és kimenetei is logikai változók
- két független bemenet (A és B) esetén:

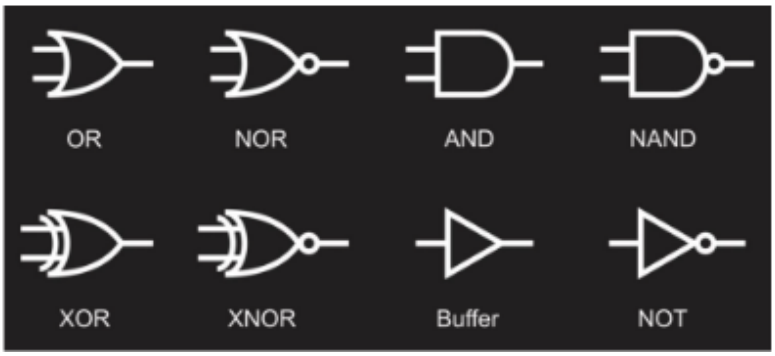
A	**B**	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

- ismert logikai műveletek: negálás, és, vagy, kizáró vagy (ezekre van műveleti jel is C-ben)
- f1: „és”, f7: „vagy”, f6: „kizáró vagy”, f8: „nem vagy”, f14: „nem és”, f0: „azonosan 0”, f15: „azonosan 1”.
- n bemenet esetén n db változó van, mindegyiknek 2 értéke, szóval a 2^n bemeneti kombinációhoz 2 elemet rendelünk, így 2^{2^n} db függvény lesz
- **De Morgan azonosságok:**

$$\overline{(A \wedge B)} = \bar{A} \vee \bar{B}$$
$$\overline{(A \vee B)} = \bar{A} \wedge \bar{B}$$

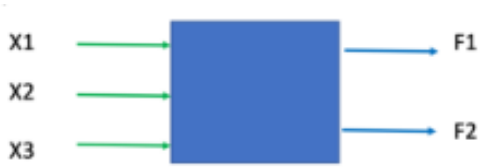
Logikai kapuk:

- logikai „építőkockák”, melyek alapműveleteket valósítanak meg
- összekapcsolásukkal jöhet létre az aritmetikai művelet



Kombinációs logikai hálózat:

- kimenete(i) csak a bemenet(ek)től függenek
- minden kimenetet egy függvény ír le: $F_1(X_1, X_2, \ldots, X_n)$



Példa: 1 bites fél összeadó

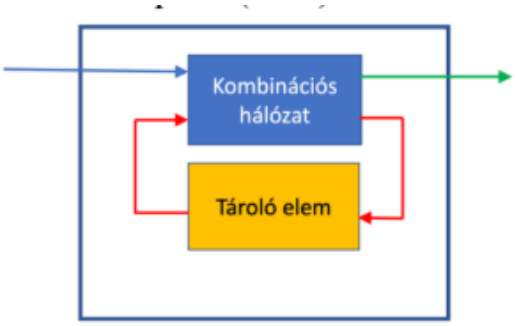
- bemenetek: A és B
- kimenet: Y és C (carry)-átvitel

A	B	Y	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

- Függvények:
- Y = A xor B
- C = A and B

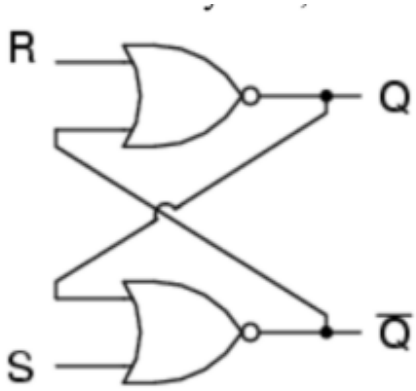
Szekvenciális logikai hálózat:

- függ a **bemenetektől** és a **hálózat belső állapotától**
- aszinkron: nincs ütemező órajel
- szinkron: csak órajelnél vált állapotot (CPU)



S-R tároló:

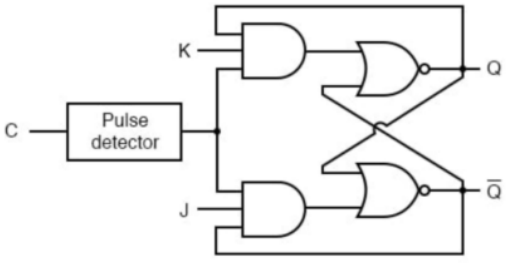
- S: set, 1-re állítja a kimenetet
- R: reset, 0-ra
- S = 0 és R = 0, akkor tartja az értékét (memória)
- S = 1 és R = 1 érvénytelen, mert kimenet és a kimenet negáltja is 0



S	R	Q	$\bar{Q}$
0	0	latch	latch
0	1	0	1
1	0	1	0
1	1	0	0

J-K tároló:

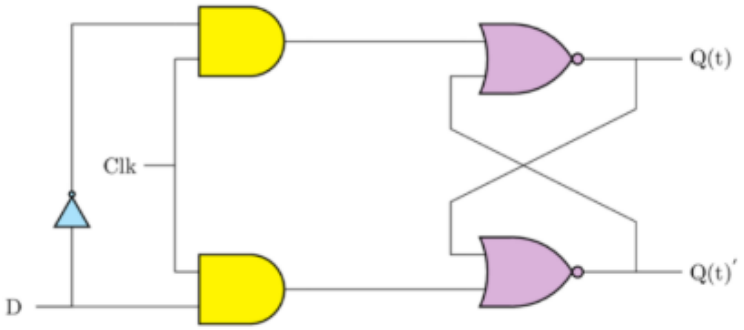
- működése az S-R tárolóéhoz hasonló, viszont ez egy szinkron működésű szekvenciális logikai hálózat
- J: set, 1-re állít
- K: reset, 0-ra állít
- S = 0, R = 0, tartja az értékét (memória)
- S = 1, r = 1, megváltoztatja az értékét az előző kimenet negáltjára



C	J	K	Q	$\bar{Q}$
↑	0	0	latch	latch
↑	0	1	0	1
↑	1	0	1	0
↑	1	1	toggle	toggle
x	0	0	latch	latch
x	0	1	latch	latch
x	1	0	latch	latch
x	1	1	latch	latch

D tároló:

- Az S-R tárolót kiegészítjük „és” kapukkal, valamint egy léptető órajellel: Clk
- Az egyetlen bemenetet kettéágaztatjuk, egyik felét negáljuk → nem lehet S és R egyszerre 1
- „statikus RAM”
- gyors elérés
- bonyolult felépítés
- CPU regiszterei, cache



2.3 A számítógép felépítése. Memóriák. CPU részei. Utasítás ciklus. Szubrutinhívás. Interrupt. Közvetlen memória hozzáférés.

Hardver:

- elektronikus/mechanikus alkatrészek összessége
- kézbe vehető
- szoftver nélkül nem üzemképes

Szoftver:

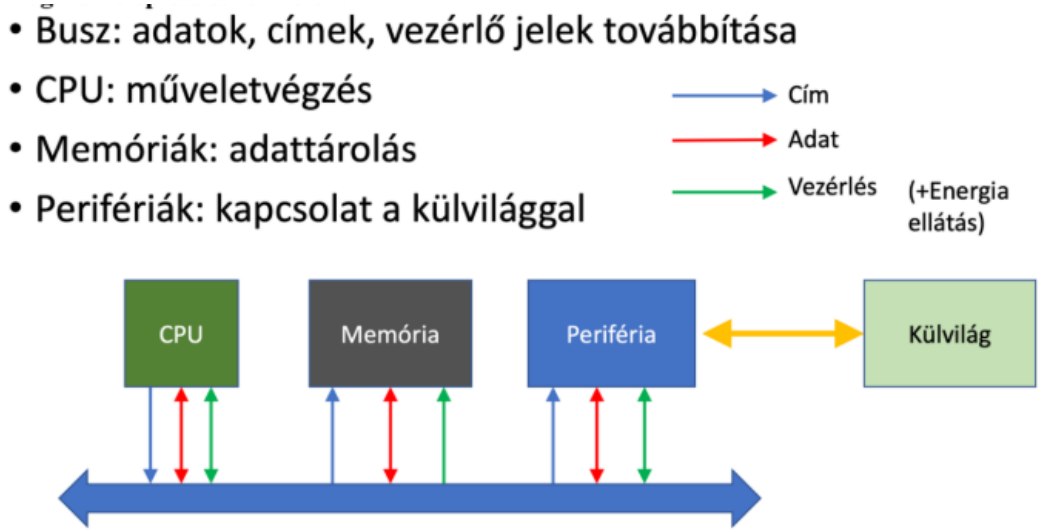
- számítógépet működtető programok

Firmware:

- a kettő együtt
- olyan szoftver, mely egy hardverben található
- pl PC-nél bekapcsoláskor elinduló ROM-BIOS az alaplap egyik integrált áramkörébe töltött program

Logikai felépítés és funkciók:

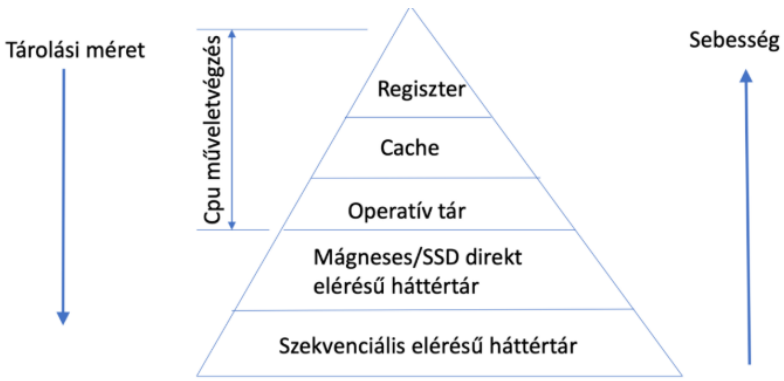
- **Busz:** adatok, címek, vezérlő jelek továbbítása
- **CPU:** műveletvégzés
- **Memóriák:** adattárolás
- **Perifériák:** kapcsolat a külvilággal



- busz: párhuzamos (manapság soros) jelköteg
- busz szélességek: egyszerre átvihető adat mérete
- adatbusz: 8, 16, 32, 64 bit, szinte mindig akkumulátor regiszter mérete
- címbusz: memória max méretét adja meg

Memóriák – tárolók hierarchia szintjei:

- **regiszterek:**
  - a CPU belső tárolói
  - nagyon gyors hozzáférés az adatokhoz
  - közvetlenül a CPU-ban találhatóak
  - utasítások végrehajtása
  - adatok köztes tárolása
- **cache:**
  - kis méretű, gyors, közel a processzorhoz
  - az aktuálisan leggyakrabban használt adatokat, utasításokat tárolja
  - cél: memória-hozzáférési idő csökkentése, rendszer teljesítményének javítása
- **operatív tár** (RAM – random access memory):
  - adatok, utasítások átmeneti tárolása
  - nagyobb kapacitás, mint a cache és regiszterek, de lassabb hozzáférés
  - a futó programok számára ideiglenesen tárolódnak itt adatok
- **Mágneses / SSD direkt elérésű háttértár:**
  - mágneses lemezeket (merevlemezeket) vagy szilárdtest meghajtókat (SSD) használ
  - nagyobb kapacitás
  - adatok hosszútávú tárolása
- **Szekvenciális elérésű háttértár**
  - szalag típusú adathordozók
  - adatok egymás után találhatóak
  - ilyen sorrendben történik a hozzáférés
  - lassabb



CPU:

- központi feldolgozó egység
- operatív memóriából olvassa be a program utasításait és az adatokat
- az utasításokat dekódolja, végrehajtja
- eredmények operatív memóriában
- buszok vezérlése: címek kiküldése, vezérlőjelek, memória/periféria címzése, adat bekérése/kiküldése buszon (olvasás/írás)
- órajel működteti → szekvenciális
- gyorsítani lehet: cache, utasítás végrehajtó egység számának növelése

CPU részei:

- regiszterek:
  - processzoron belüli memória
  - adatok, címek, műveletek eredményei
  - az utasítások a busz nélkül elérik
- ALU
  - aritmetikai-logikai egység
  - aritmetikai műveletek: összeadás, kivonás, a++, a–, 2-es komplemens képzés, néha egész számok szorzása/osztása, bitek eltolása
    - lebegőpontos műveletekhez szoftver, vagy társ(-, vagy co)processzor
  - logikai műveletek: negálás, és, vagy, kizáró vagy, összehasonlítás: kivonás, eredménye csak flag-ekben jelenik meg

Utasítás ciklus:

- végrehajtandó utasítás címe programszámláló regiszterben
- utasítás beolvasása a memóriából (fetch)
- utasítás dekódolása, ha vannak, akkor paraméterek beolvasása memóriából
- utasítás végrehajtása, eredmény tárolása (ha van)
- következő utasítás címének megadása: programszámláló regiszter a végrehajtott utasítás hosszával inkrementálódik

Szubrutinhívás:

- egy korábbi kódrészlet újrahívása
- a következő utasítás címe veremtárba kerül, a program oda fog visszatérni
- újra használható a kód

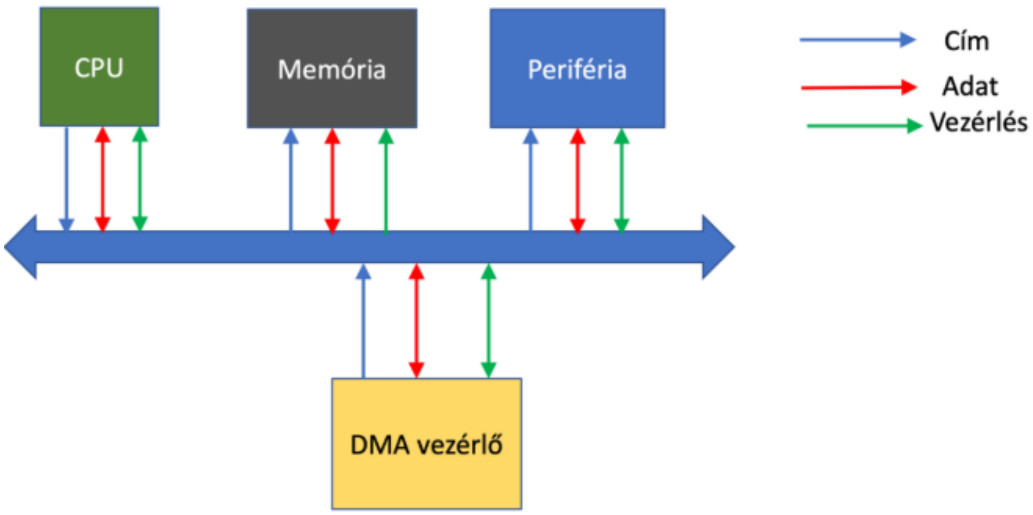
Interrupt:

- megszakítja az adott utasítást
- eltárolja, hogy hova kell visszatérjen
- megszakítás kezelése (ISR – interrupt service routine)
- visszatérés oda, ahol félbe lett szakítva

Közvetlen memória hozzáférés:

- DMA – direct memory access
- a művelet közben nincs nagy számítási igény: buszt kell vezérelni, címet növelni 1-gyel
- speciális áramkör: DMA vezérlő

- a CPU megmondja a DMA vezérlőnek, hogy honnan, hova, hány bájtot
- a DMA vezérlő elkéri a buszt a CPU-tól, ha annak épp nincs szüksége rá
- DMA vezérlő átvisz 1 bájtot a perifériából a saját adatregiszterébe
- a DMA vezérlő által elkért buszon átvisz 1 bájtot a memóriába (írás esetén fordítva)
- DMA vezérlő visszaadja a buszt
- DMA újra elkéri a buszt
- az utolsó bájt átvitele után nem kéri el a buszt, IRQ (interrupt request) -val jelzi, hogy az átvitel kész
- busz arbitáció: busz elkérése



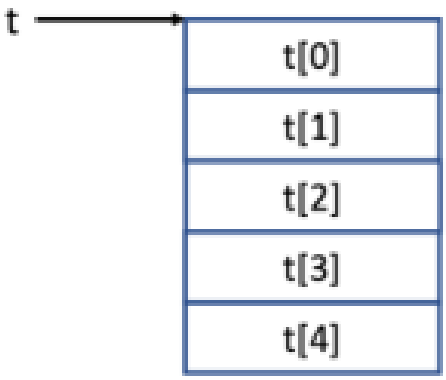
2.4 Adatszerkezetek. Tömbök, kapcsolt listák, gráf, fa, verem, sor.

Pointer:

- nem adatot, hanem címet tárol
- az adatot indirekt módon tudjuk elérni
- típusa: ilyen típusú adatot tartalmaz a cím, amire a pointer mutat

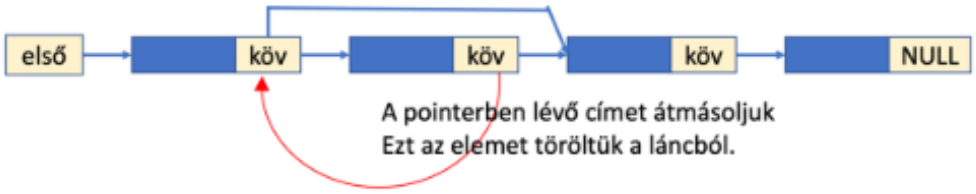
Adatokat különböző adattárolókban lehet tárolni, melyek más-más módon működnek, attól függően érdemes kiválasztani, melyiket szeretnénk használni, hogy mire használjuk őket.
Tömb:

- azonos típusú adatok
- az adatok indexelve vannak (számozva)
- az index segítségével férhetünk hozzá bármelyik adathoz (**kicímzés** lehetséges!!!!)
- egymást követő memóriacímeken van tárolva
- deklarálásnál meg van szabva, hány elemű
- egy adott elemhez konstans idővel, bejárás nélkül férünk hozzá
- beszúrás/törlés nem hatékony
- vannak lineárisak és többdimenziósak is: mátrixok
- rendezetlen tömbben lineáris keresés: $O(n)$
- rendezett tömbben bináris keresés (intervallumfelezés): $O(\log_2 n)$

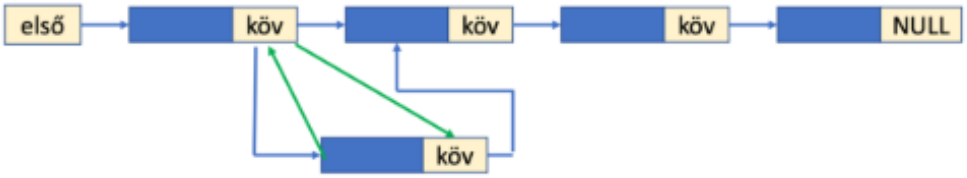


Kapcsolt lista:

- Adatok olyan sorrendje, ahol a sorrendet a pointerek határozzák meg
- beszúrás/törlés hatékony
- bejárás, valamint **egy elem keresése** is lineáris bonyolultságú
- lehet egy,- és kétirányú lánc
- végjel: NULL pointer
- az adatok sorrendje a memóriában tetszőleges, csak a pointereket változtatjuk

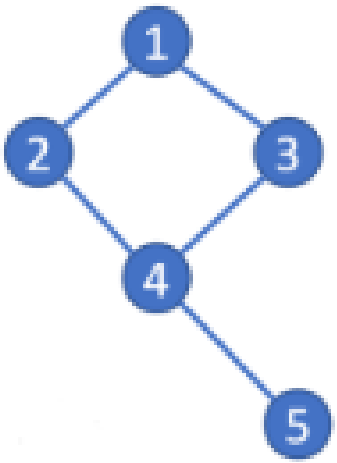


Beszúrás:



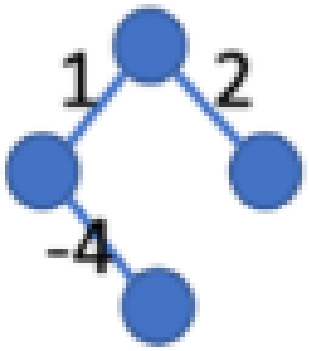
Gráf:

- sorszámozott csomópontok és az azokat összekötő élek halmaza
- szomszédok: összekötött csomópontok
- deg(u): egy adott csomópont foka a befutó élek száma
- ha deg(u) = 0, akkor a csomópont izolált pont
- egyik csomópontból (v_0) egy másikba (v_n) haladó élek halmaza az út
- P($v_0, \ldots, v_n$) út zárt, ha $v_0 = v_n$
- egy út egyszerű, ha minden pontja különböző
- kör: 3-nál hosszabb egyszerű, zárt út
- összefüggő: olyan gráf, melynek bármelyik 2 pontja közt létezik út
- teljes: minden csomópont mindegyik másikkal össze van kötve
- címkézett: az élekhez súlyokat rendelünk
- súlyozott: nemnegatív címkék
- irányított: az éleknek iránya van
- tárolásuk: szomszédsági mátrix: $a_{ij} = 1$, ha i-ből j-be halad él, egyébként 0



1. ábra: *
Egyszerű gráf

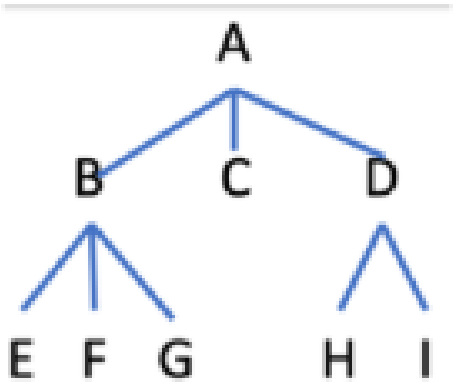
2. ábra: *
Teljes gráf



3. ábra: *
Súlyozott gráf

Fa:

- Köröket nem tartalmazó gráf
- elemek véges (T) halmaza, mely
  - tartalmaz egy kitüntetett gyökérelemet
  - a többi elem nemnulla diszjunkt részfája T-nek
- tárolás:
  - info(k) – az elem adatai
  - gyermek – az első gyermek indexe
  - testvér – az első testvér indexe
- bináris fa: egy szülőnek max 2 leszármazottja lehet

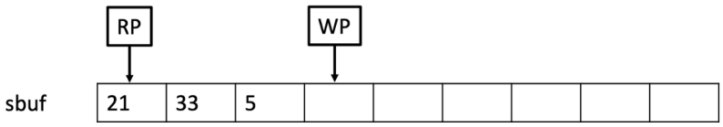


Veremtár/stack:

- LIFO: Last In First Out
- Push: elemet a verembe rak
- Pop: elemet leemel a verem tetejéről
- alkalmazások: függvényhívás, rekurzió, böngésző „vissza”, szövegszerkesztő „undo” gombja
- SP: stack pointer: a verem tetejét mutatja

Sor/Queue:

- FIFO: First In First Out
- 2 mutató: WP, RP (write, read)
- írás/olvasás előtt/után a mutatót növelni kell



2.5 Algoritmusok. Bejárás, keresés, rendezés. Algoritmusok bonyolultsága. Rekurzió.

Eljárások, algoritmusok:

- emberi nyelven megfogalmazott feladat, cselekménysorozat
- megoldási eljárás („algoritmus jelölt"), program írható
- egy megoldási eljárás akkor **algoritmus**, ha **bármilyen** bemenet esetén **véges számú lépés** után eredményt kapunk (A Turing-gép megáll)
- az algoritmusra nem létezik formális matematikai definíció

Algoritmusok bonyolultsága:

- ha tudjuk, hogy egy problémára létezik algoritmus, meg kell vizsgálni a megvalósíthatóságát: a bemenetek számának függvényében a tárhely és lépésszám (számítási idő) szükségletet: bonyolultságot – határértékszámítás
- konstans:
  - az algoritmus futási ideje állandó marad az input méretétől függetlenül
  - $O(1)$
  - pl. a tömb egy adott elemének lekérése
- logaritmikus:
  - az algoritmus futási ideje logaritmikusan növekszik az input méretével
  - $O(\log N)$
  - pl. bináris keresés rendezett tömbben: $O(\log_2 N)$
- lineáris:
  - az algoritmus futási ideje lineárisan növekszik az input méretével
  - $O(N)$
  - pl. egy elem keresése N elemű tömbben: $O(N)$
- polinomiális:
  - az algoritmus futási ideje polinomiálisan (exponenciálisan) növekszik az input méretével
  - $O(N^2)$
  - pl. bubble sort algoritmus: $O(N^2)$
- faktoriális:
  - az algoritmus futási ideje faktoriálisan növekszik az input méretével
  - $O(N!)$
  - pl. egy N elemű sorozat összes permutációjának előállítása: $O(N!)$

Bejárás:
Adott adattárolókat különböző módon lehet bejárni, azaz minden elemükhöz hozzáférni, melyeknek más-más komplexitásuk lehet

- tömb: mivel az adatok egymás után helyezkednek el a memóriában, végig csak egyesével kell növelni a pointert, komplexitás: $O(n)$
- kapcsolt lista: szintén lineáris bonyolultság: $O(n)$
- fa: rekurzióval

Keresés:
Adott értékű elemet keresünk az adattárolóban, ezt többféle módon lehet attól függően, hogy az adott adattároló rendezett-e vagy sem

- tömb: alapvetően lineáris komplexitású: $O(n)$

- ha rendezett a tömb, akkor lehet intervallumfelezéssel keresni, így a bináris keresés komplexitása logaritmikus: $O(\log_2 n)$
- a kapcsolt listában szintén lineáris komplexitással lehet keresni: $O(n)$

Rendezés:

- tömbök rendezése lehetséges többféle módon: a legalapabb, leglassabb $O(n^2)$ komplexitású a bubble sort
- lehet quick sort algoritmust is alkalmazni, mely egy rekurzívan alkalmazott függvény
- az intervallumot mindig két új intervallumra osztja és az elemeket besorolja a megfelelő intervallumba (ezek diszjunkt halmazok)
- rekurzívan meghívja magát
- egészen addig ismétlődik, amíg 1-1 elem nem marad az összes részhalmazban
- átlagos komplexitása: $O(n \cdot \log(n))$, a legrosszabb eset: $O(n^2)$

Rekurzió:

- a műveletet saját magával definiáljuk
- pl. faktoriális rekurzióval:
  - $0! = 1$
  - ha n nem 0, akkor $n! = n \cdot (n-1)!$
- a rekurzív algoritmus saját magát indítja újra működés közben, a veremtár segítségével
- szükség van valamilyen kilépési feltételre, különben megtelik a veremtár
- fa adatszerkezet bejárása csak rekurzióval lehetséges

2.6 Az adatbázisok alapjai. Adatmodellezés. Kapcsolatok típusai. Relációs adatbázismodell. Relációk jellemzői. A relációs algebra műveletei. SQL alapok, lekérdezések.

Adatbázisok:

- rekord: összetartozó adatok egy példányhoz
- adattételekből áll
- összetartozó adatokat tárol
- **adatmodell:** a valós világ elemeinek és köztük lévő kapcsolatok leképzése adatokra
  - egész pontosan: az azokról meglévő ismereteinket képezzük le az adatokra
- szűkebb értelemben: a leírásra szolgáló adatok szerkezete
- az adatmodell általánosan tartalmazza:
  - az alapelemeket
  - az alapelemekkel végzett műveleteket
  - integritási kényszereket:
    - adatszerkezet: adatok tárolására szolgáló elemi adatok rendszere
    - művelet: a megjelenítési igényeknek megfelelően (pl. indexek)
    - integritási kényszer: ellentmondás mentességet biztosító feltételek, megszorítások

Egyed – kapcsolat modell:

- egyed: a világ egy megkülönböztetett objektuma
- egyedek egy halmaza – rekordtípus
- egyedelőfordulás-halmaz: azonos, megkülönböztető tulajdonsággal jellemzett egyedelőfordulások gyűjteménye
- megkülönböztethetőség biztosítása:
  - természetes módon (ujjlenyomat)
  - mesterségesen (OM azonosító, személyi, TAJ, stb)
- kapcsolat: egyedek közt fennálló viszony
- kapcsolattípus előfordulás halmaz: két egyedelőfordulás halmaz, illetve az egyedelőfordulások között fennálló viszony
- kapcsolat típusa:
  - 1:1 (házasság)
  - 1:N (oktató – csoport)
  - N:M (ingatlan – nyilvántartó)
- tulajdonság: egyedek, illetve kapcsolatok azon jellemzői, melyek az egyedtípust definiálják / az egyedtípusok közti kapcsolatok jellemzői (szőke ember)
- tulajdonságérték: az egyed-, vagy kapcsolatelőfordulásokhoz tartozó konkrét adat (szőke)
- értékhalmaz: a tulajdonság lehetséges értékei (szőke, barna, fekete, vörös)

Reláció:

- n darab halmaz direkt szorzatának részhalmaza
- névvel azonosítjuk
- minden sor különböző benne
- kulcs: a reláció bármely elemét egyértelműen azonosítja

- fokszám: a relációt alkotó halmazok (értelmezési tartományok) száma
- kardinalitás: a reláció elemeinek száma
- attribútum: egyes elemekben a tényezők konkrét értéke
- az egyes elemeknek nincs sorrendje
- **műveletek:**
  - egyesítés
  - metszet
  - különbség
- **csonkító műveletek:**
  - vetítés (projection): tényezők (értelmezési tartományok) kiemelése
  - kiválasztás (select): elemek kiválasztása
- **kapcsoló műveletek:**
  - join: kapcsoló tényezők kapcsolnak
  - Descartes szorzat: mindent mindennel (direkt szorzat)
- a reláció ábrázolható kétdimenziós adattáblával
- oszlopok: tulajdonságok
- a táblát névvel azonosítjuk (reláció neve)
- sorok: reláció előfordulások halmaza – nincs két azonos sor
- tulajdonságok nem lehetnek összetettek

Kulcsok:

- segítségükkel megkülönböztetjük a sorokat (nem lehetnek ugyanolyanok)
- szuperkulcs: sorokat megkülönböztető oszlophalmaz
- kulcs: minimális elemszámú szuperkulcs
- elsődleges kulcs: a megkülönböztetésre választott kulcs
  - egyedi érték kell legyen
  - nem lehet benne NULL
- külső kulcs: 1:N és M:N kapcsolatok leírására
  - másik táblázat elsődleges kulcsa által felvett érték, vagy NULL lehet

Lekérdezések:

- `SELECT [ALL | DISTINCT | TOP n [PERCENT] ] { <kifejezés lista> | * }`
- `FROM { <táblázat név> > } [másodnév] [, ...]`
- `[WHERE <kiválasztási feltétel>]`
- `[GROUP BY <oszlopnév lista>]`
- `[HAVING <kiválasztási feltétel>]`
- `[ORDER BY {<oszlop név> | <egész áll>} [ASC|DESC] [, ...] ]`
- pl. kik kaptak elégtelent?
- `SELECT neptun FROM vizsga WHERE jegy=1;`
- Ahhoz, hogy a nevet is megtudjuk, kell a másik tábla is
- `SELECT hallgato.nev, vizsga.jegy FROM hallgato, vizsga, WHERE hallgato.neptun = vizsga.neptun and jegy=1;`

2.7 Az operációs rendszer céljai, feladatai. Folyamatok kommunikációja. Ütemezési algoritmusok az operációs rendszerben. Termelő-fogyasztó probléma. Postaláda kezelés. Szemaforok.

Operációs rendszer:

- egy program (rendszer), amely közvetítő szerepet játszik a számítógép felhasználója és a számítógép hardvere között
- felhasználói programok végrehajtása, felhasználói feladatmegoldás könnyítése
- a számítógép rendszer használatának kényelmesebbé tétele
- a számítógép hardver kihasználásának hatékonyabbá tétele
- koordinálja és vezérli a hardver erőforrások különböző felhasználók különböző programjai által történő használatát
- **alkalmazói-programok:** hogyan legyenek felhasználva a rendszer erőforrások a felhasználók számítási problémáinak megoldásához
- felhasználók: emberek, gépek, más számítógépek
- részei:
  - Kernel
  - Shell
  - Utility-k

Kernel:

- memória rezidens főprogram
- folyamat kezelés
- memória kezelés
- háttértár kezelés
- I/O kezelés
- fájl kezelés
- védelmi rendszer
- hálózat elérés támogatása

Folyamat kezelés:

- folyamat: egy végrehajtás alatt lévő program
- erőforrásokra van szüksége
- operációs rendszer feladata:
  - folyamat létrehozása, törlése
  - folyamat felfüggesztése, újraindítása (multitaszk)
  - eszközök biztosítása a folyamatok szinkronizációjához, kommunikációjához
- folyamat általában műveletek végrehajtása meghatározott sorrendben
- folyamat elkezdődik és befejeződik
- a következő részművelet végrehajtása akkor kezdődhet, ha az előző befejeződött
- op. rsz. több folyamatból áll
- hatékonyabb erőforrás kihasználás
- feladat végrehajtás gyorsítása

- többféle feladat egyidejű végrehajtása

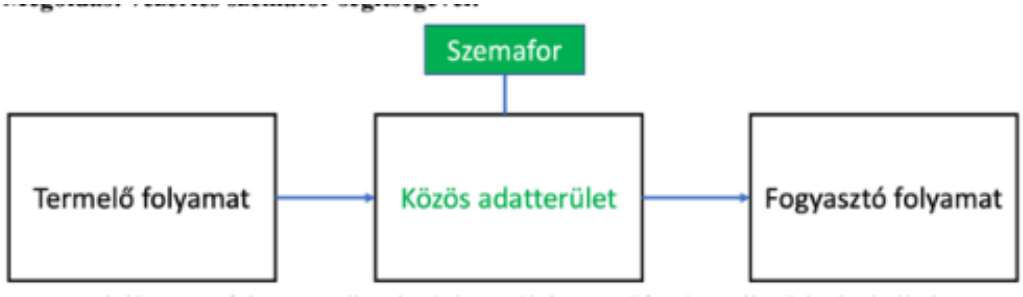
Folyamatok típusai:

- független: egymás működését nem befolyásolják
- versengő: nem ismerik egymást, de közös erőforrásokon kell osztozzanak
- együttműködő: ismerik egymást, együtt dolgoznak egy feladat megoldásán, információt cserélnek

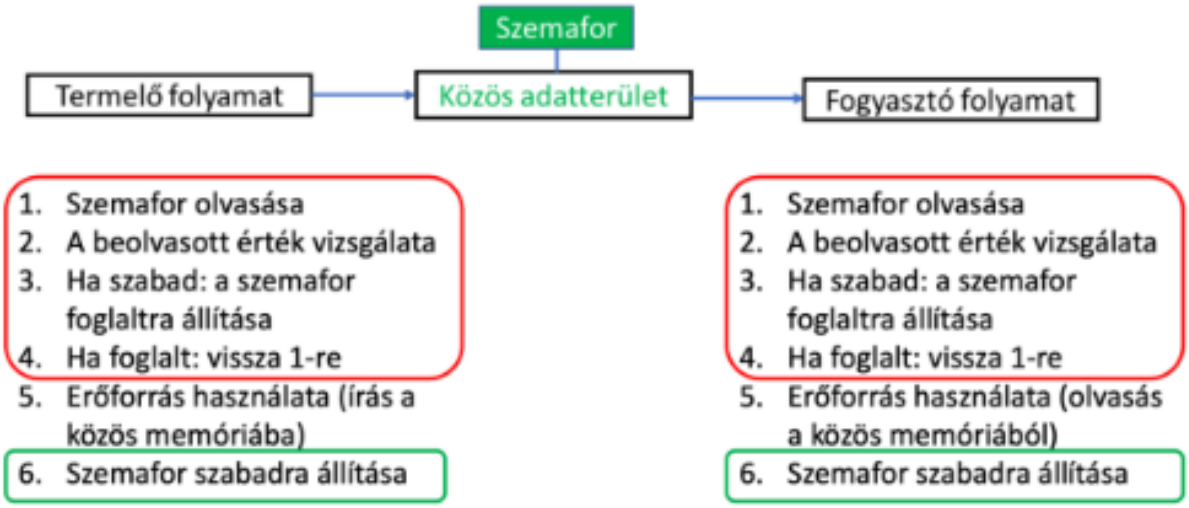
Több egymással párhuzamosan futó folyamat gyakran kommunikál közösen használt memóriaterületek segítségével. Ezek a területek nem érhetők el egyidejűleg a folyamatok számára. Az egyidejű hozzáférés kizárása **szemaforok** (bináris, nem bináris) segítségével történik.
Termelő-fogyasztó probléma:

- a közös adatterületet egyszerre csak egy folyamat használhatja
- kölcsönös kizárás esete nem csak közös memória esetén lép fel (pl. nyomtató közös használata)

Megoldás: vezérlés szemafor segítségével:



- mielőtt egy folyamat elkezdené használni az erőforrást, ellenőriznie kell, hogy az szabad-e
- csak akkor kezdheti használni, ha a szemafor szabadot jelzett, egyébként vár

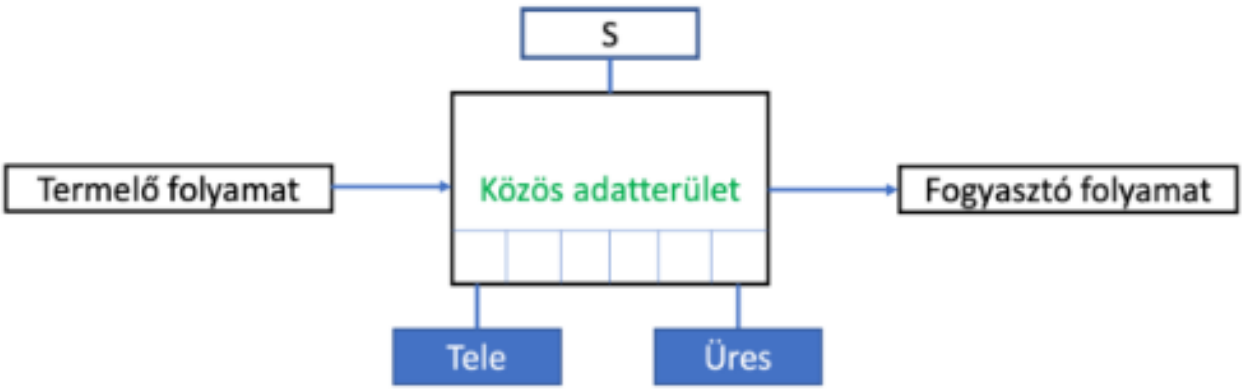


- kritikus szekció, kritikus szakasz, kritikus régió
- oszthatatlan művelet: P primitív, V primitív

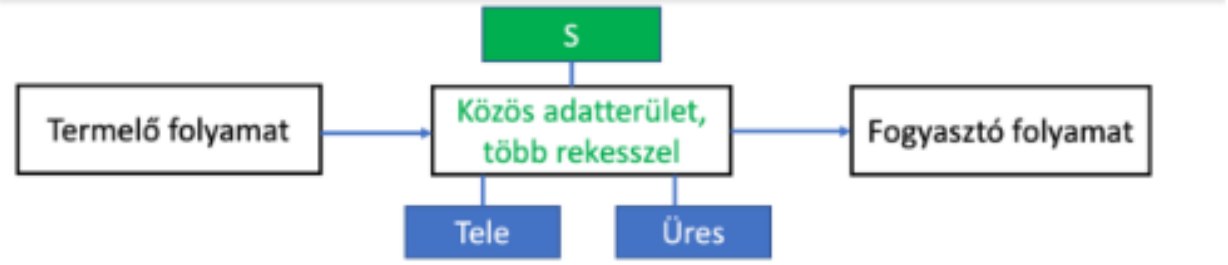
- P primitív: foglaltra állítás
- V primitív: szabadra állítás

Postaláda kezelés:

- postaláda: olyan közös adatterület, ahová egynél több üzenet írható



- újabb szemaforok a vezérléshez:
  - Tele
  - Üres
- ezekben egész számot tárolunk
- 3 db szemafor a vezérléshez:
  - S: kölcsönös kizárást megvalósító szemafor (0 = foglalt, 1 = szabad)
  - Tele: tele helyek száma, nem bináris
  - Üres: üres helyek száma, nem bináris
- P primitív: a paraméterül kapott szemafor értékének eggyel csökkentése („foglalt”)
- V primitív: a paraméterül kapott szemafor értékének eggyel növelése („szabad”)



1. P(Üres);
2. P(S);
3. Írás a memóriába;
4. V(S);
5. V(Tele);

1. P(Tele);
2. P(S);
3. Olvasás a memóriából
4. V(S);
5. V(Üres);

Pszeudokód:

Termelő:

- P(Üres);
- P(S);
- Írás a memóriába;
- V(S);
- V(Tele);

Fogyasztó:

- P(Tele);
- P(S);
- Olvasás a memóriából;
- V(S);
- V(Üres);

2.8 Holtpont az operációs rendszerben. Holtpont kezelése. Holtpont észlelése. Holtpont megelőzés. Bankár algoritmus.

Folyamat kezelés:

- folyamat: egy végrehajtás alatt lévő program
- erőforrásokra van szüksége
- operációs rendszer feladata:
  - folyamat létrehozása, törlése
  - folyamat felfüggesztése, újraindítása (multitaszk)
  - eszközök biztosítása a folyamatok szinkronizációjához, kommunikációjához
- folyamat általában műveletek végrehajtása meghatározott sorrendben
- folyamat elkezdődik és befejeződik
- a következő részművelet végrehajtása akkor kezdődhet, ha az előző befejeződött

Ha több folyamat is ugyanazt az erőforrást használja, ügyelni kell a holtpont elkerülésére.
Holtpont: a folyamatok nem tudnak folytatódni.

- pl. ha két folyamat egymásra vár, mert pont a másik folyamat erőforrására lenne szüksége, de azt meg éppen ő foglalja

Holtpont kezelése:

- nem teszünk semmit (strucc algoritmus)
- detektálás és feloldás: észrevesszük, ha holtpont van és megpróbáljuk feloldani
- megelőzés: struktúrájában holtpontmentes rendszer tervezése

Holtpont detektálása, példa:

- legyen 1 erőforrás, ebből 10 példány
- legyen 4 folyamat: P1, P2, P3, P4
- a pillanatnyi helyzet:
- 9 erőforrás le van foglalva, 1 szabad

	Lefoglalva	**Igény**
P1	4	4
P2	1	0
P3	3	4
P4	1	1

- P2 fut (mivel nem igényel több erőforrást), P1, P3, P4 várakoznak
- P2 lefut, 1 erőforrás felszabadul (szabad: 2)
- P4 lefut (igénye 1), még 1 erőforrás felszabadul (szabad: 3)
- P1 (igény 4) és P3 (igény 4) közül egyik se tud futni, nincs (elég) szabad erőforrás, így holtpontra kerülnek

Holtpont feloldása, példa:

- radikális megoldás: a holtpontban érintett összes folyamatot felszámoljuk
- kíméletes megoldás:
- erőforrás átmenetileg mentés, kiosztás másnak, majd visszaállítás
- legkevesebb folyamat felszámolása
- prioritásos folyamatoknál: alacsonyabb prioritású folyamat mentése

- Hol tartunk a folyamatban?
- Folyamat és erőforrás visszaállíthatósága szükséges a kíméletes feloldáshoz

Holtpont megelőzése: bankár algoritmus:

- biztonságosan tervezett az a folyamatokat és erőforrásokat tartalmazó rendszer, amelyben:
  - létezik a folyamatoknak (legalább egy) olyan sorrendje, mely szerint végrehajtva őket, azok maximális erőforrás igénye is kielégíthető
  - a biztonságos rendszerben nem lehetséges holtpont kialakulása
- az ellenőrzést **bankár algoritmus**sal végezzük a folyamat indítása és az erőforrás foglalás előtt

Bankár algoritmus példa:

- egy rendszerben 3 erőforrás van: E1 10 db, E2 5 db, E3 7 db
- ebben a rendszerben 5 folyamat fut: P1, P2, P3, P4, P5

Max igény (1. lépés)

	E1	**E2**	**E3**
P1	7	5	3
P2	3	2	2
P3	9	0	2
P4	2	2	2
P5	4	3	3

Aktuálisan foglalt (2. lépés)

	E1	**E2**	**E3**
P1	0	1	0
P2	3	0	2
P3	3	0	2
P4	2	1	1
P5	0	0	2

- Biztonságos ez az állapot?
- 3. lépés: igény = max igény – foglalt

Max	**E1**	**E2**	**E3**	**Foglal**	**E1**	**E2**	**E3**	**igény**	**E1**	**E2**	**E3**
P1	7	5	3	P1	0	1	0	P1	7	4	3
P2	3	2	2	P2	3	0	2	P2	0	2	0
P3	9	0	2	P3	3	0	2	P3	6	0	0
P4	2	2	2	P4	2	1	1	P4	0	1	1
P5	4	3	3	P5	0	0	2	P5	4	3	1

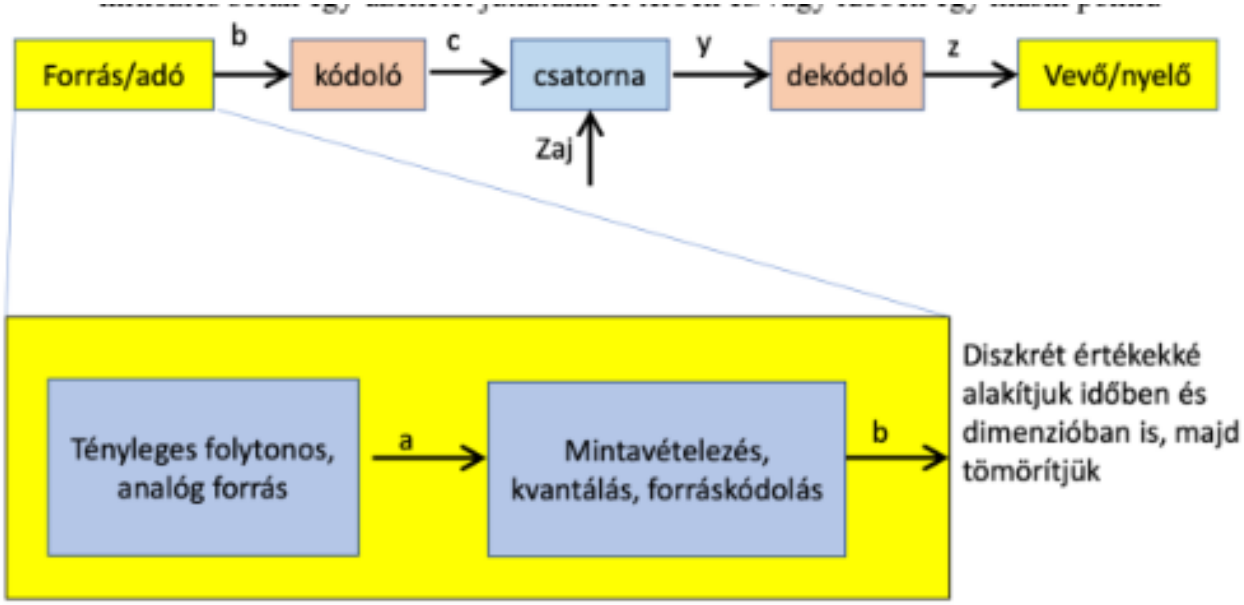
- 4. lépés: szabad erőforrások számának meghatározása:
- E1: 10-8=2, E2: 5-2=3, E3: 7-7=0
- a készlet vektorban: (2,3,0)
- 5. lépés: a készletből kielégíthető valamelyik folyamat igénye?
- igény <= készlet?
- P2 folyamat ilyen (0,2,0 <= 2,3,0)
- 6. lépés: P2-t lefuttatjuk
- P2 lefutása után az általa eddig foglalt erőforrások felszabadulnak: (3,0,2)
- 7. lépés: új készlet: (2,3,0) + (3,0,2) = (5,3,2)

- 8. lépés: ha még van folyamat, vissza az 5. lépésre
- 5. lépés: készlet (5,3,2) igény kielégíthető-e?
- P5 (4,3,1) folyamat ilyen
- 6. lépés: P5-t (0,0,2) lefuttatjuk
- 7. lépés: új készlet: (5,3,2) + (0,0,2) = (5,3,4)
- 8. lépés: ha még van folyamat, vissza az 5. lépésre
- 5. lépés: készletből (5,3,4) igény kielégíthető-e?
- P4 (0,1,1) folyamat ilyen
- 6. lépés: P4-t (2,1,1) lefuttatjuk.
- 7. lépés: új készlet: (7,4,5).
- 5. lépés: P1(0,1,0) lefuthat. (Igény: 7,4,3 <= 7,4,5)
- 7. lépés: (7,5,5),
- 5. lépés: P3(6,0,0) igény lefuttat. (3,0,2) új készlet:
- 7. lépés (10,5,7)
- 8. lépés: találtunk 1 sorrendet, amely elkerüli a holtpontot, biztonságos állapotban vagyunk.

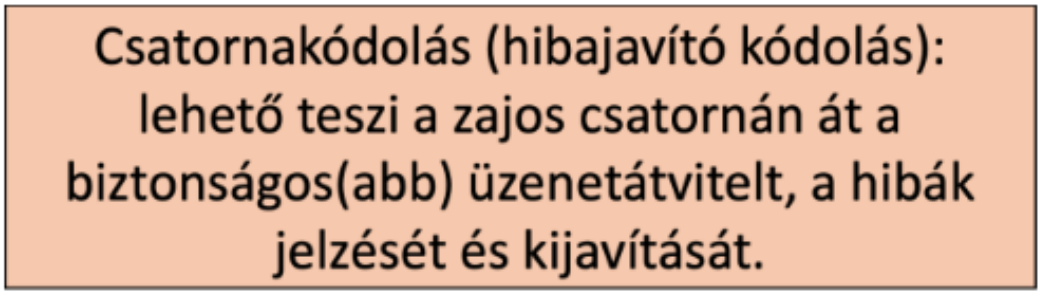
2.9 Shannon hírközlési modellje. Forráskódolás, prefix kód.

Shannon hírközlés modellje:

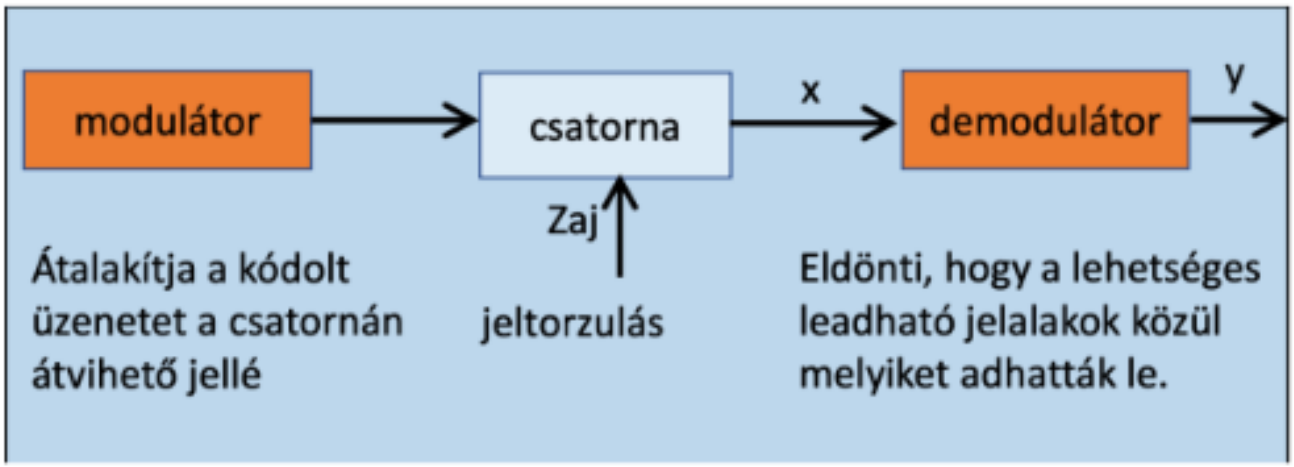
- hírközlés során egy üzenetet juttatunk el térben és/vagy időben egy másik pontra



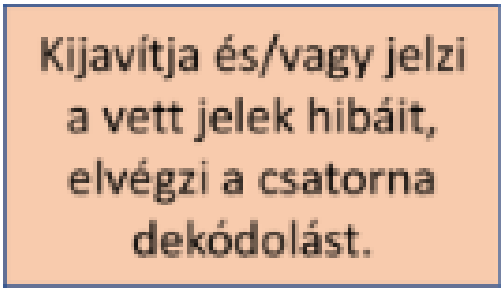
- **Forrás/adó:**
  - Tényleges folytonos, analóg forrás $\xrightarrow{a}$ Mintavételezés, kvantálás, forráskódolás $\xrightarrow{b}$
  - Diszkrét értékekké alakítjuk időben és dimenzióban is, majd tömörítjük
- **kódoló:**



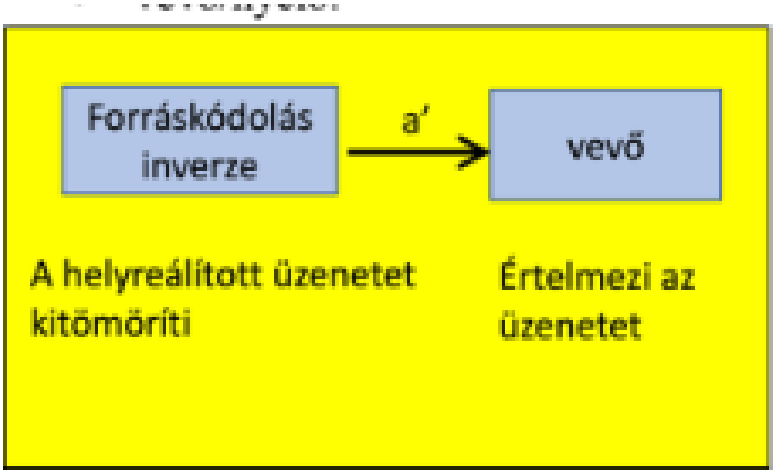
- **Csatornakódolás (hibajavító kódolás):** lehetővé teszi a zajos csatornán át a biztonságos(abb) üzenetátvitelt, a hibák jelzését és kijavítását.
- **csatorna:**



- modulátor → csatorna (Zaj, jeltorzulás) → demodulátor
- Átalakítja a kódolt üzenetet a csatornán átvihető jellé
- Eldönti, hogy a lehetséges leadható jelalakok közül melyiket adhatták le.
- **dekódoló:**



- Kijavítja és/vagy jelzi a vett jelek hibáit, elvégzi a csatorna dekódolást.
- **vevő/nyelő:**



- Forráskódolás inverze $\xrightarrow{a'}$ vevő
- A helyreállított üzenetet kitömöríti
- Értelmezi az üzenetet

Információelmélet:

- információ: valamely véges számú, előre ismert esemény közül annak a megnevezése, hogy melyik következett be
- az információ mértéke azonos azzal a bizonytalansággal, amit megszüntet
- **Hartley:** m számú, azonos valószínűségű esemény közül egy megnevezésével nyert információ: $I = \log_2 m$
- vagyis $\log_2 m$ db eldöntendő kérdéssel azonosítható egy elem
- **Shannon:** minél váratlanabb egy esemény, bekövetkezése annál több információt jelent
- Legyen $A = \{A_1, A_2, ..., A_m\}$ esemény-halmaz, az A_1 esemény valószínűsége p_1, az A_m esemény valószínűsége p_m. Ekkor az A_i esemény megnevezésével nyert információ: $I(A_i) = \log_2 \frac{1}{p_i} = -\log_2 p_i$
- Ha $p_i = 1/m$ az összes i-re, akkor visszakapjuk a Hartley féle definíciót.

Forráskódolás:

- A forrás kimenetén véges sok elemből álló $A = \{A_1, A_2, ..., A_n\}$ halmaz elemei jelenhetnek meg. Az A halmaz elnevezése: forrásábécé.
- Üzenet: az A forrásábécé betűiből képzett, véges $A^{(1)}, A^{(2)}, ..., A^{(m)}$ sorozatok
- A lehetséges üzenetek halmaza: $\mathcal{A}$
- a kódolt üzenetek egy $B = \{B_1, B_2, ..., B_s\}$ szintén véges halmaz elemeiből épülnek fel, ahol B elnevezése: kódábécé
- kódszavak: B elemeiből képzett véges hosszúságú $B^{(1)}, B^{(2)}, ..., B^{(m)}$ sorozatok
- a lehetséges kódszavak halmaza: $\mathcal{B}$
- az f: $A \to \mathcal{B}$ és F: $\mathcal{A} \to \mathcal{B}$ függvényeket forráskódnak nevezzük
- az f leképezés a forrás minden egyes szimbólumához rendel egy kulcsszót
- egyértelműen dekódolható kód: egy f forráskód egyértelműen dekódolható, ha minden egyes B-beli sorozatot csak egy féle A-beli sorozatból állít elő
  - feltétele: f és F is invertálható legyen
- az állandó hosszúságú kódok egyértelműen dekódolhatóak, de nem mindig gazdaságosak: ASCII kód: 'A' $\leftrightarrow$ 65

Prefix kód:

- a lehetséges kódszavak közül egyik sem folytatása a másiknak, bármely kódszó végéről levágva bármekkora szegmenst, nem kapunk másik kódszót
- a prefix kód egyértelműen dekódolható, de létezik nem prefix is, ami egyértelmű
- pl. $A = \{a, b, c, d\}$, $B = \{0, 1\}$, $f(a) = 0$, $f(b) = 01$, $f(c) = 011$, $f(d) = 0111$, nem prefix, de egyértelműen dekódolható, mert 0-nál az új kód kezdődik
- legrövidebb átlagos szóhosszúságú prefix kód: **Huffman-kód**

2.10 Hálózati kommunikáció, OSI/ISO modell. Hálózati elsőbbségi elvek. Az interneten használt kommunikációs protokollok. IP cím, maszkolás, DNS rendszer.

Hálózatok:

- számítógépeket gyakran kötjük adathálózatba
- erőforrás megosztás
- csoportmunka
- kommunikációs platform
- történet: a meglévő telefonhálózatot használták adatátvitel céljára
- ma adatátvitelre használt hálózaton telefonálunk, nézünk „TV”-t

OSI/ISO (open systems interconnection) modell:

1. fizikai: összeköttetés, áramkörök
2. adatkapcsolati: hibajavítás, forgalomvezérlés
3. hálózat: ismétlés, darabolás, blokk, hálózati cím
4. szállítási: topológiamentes adatszállítás érpár közt
5. viszonylati (együttműködési): párbeszéd, kommunikáció fel-, és újraépítése
6. megjelenítési (ábrázolási): szintakszis alkalmazások irányában
7. alkalmazási: felhasználó programja (pl. böngésző, facebook app)

Hálózati cím: egyértelműen meghatározza, kitől jön és kinek megy az adat. Hálózaton belül egyedi.
Hálózat típusok:

- PAN: personal area network
- LAN: local area network
- WAN: wide area network
- topológiák: busz, csillag, gyűrű

(Ethernet) hálózat elemei:

- repeater/hub: jelerősítő (csillag közepén)
- bridge/switch: cím/port párosítások megtanulása, forgalom irányítása
- router: útvonalválasztó – címváltoztatás is történik

Hálózati elsőbbségi elvek:

- egy hálózaton több node is található, el kell dönteni, hogy ki kapcsolhat adásra – közlekedési analógia

1. minden node kaphat időszeletet, ha szabad a csatorna, foglalt csatornánál véletlenszerű ideig vár, majd újravizsgál: kis terhelésnél hatékony
    a. ethernet ezt használja
2. fix időszelet áll rendelkezésre az adáshoz
    a. az időszelet lehet azonos az összes node-ra (token-ring)
    b. lehet változtatható: sok kis cella közlekedik, néhány a node-é
    c. mindig megvan a sávszélesség: nemzetközi vonalakhoz

Logikai elrendezések:

- kliens/szerver modell: központi erőforrású informatikai rendszer
- peer-to-peer modell: minden node rendelkezhet megosztott erőforrással (pl filecserélő hálózat)

Az interneten használt kommunikációs protokollok:

- RFC: request for comments
  - IETF és más szervezetek adják ki
  - dokumentumtípus, protokollokat, szabványokat írnak le benne
  - HTTP: Hypertext Transfer Protocol

IP cím (RFC -791):

- minden internetes protokollt használó host-nak egy, 32 bit-es (= 4 byte (v4)) címe van
- a címet decimálisan ábrázoljuk, pontokkal elválasztva, pl. 152.66.25.4
- hálózati maszk: direktben elérhető gépek számításához
- ARP protokoll használata: IP↔MAC address (LAN szegmensen belül)
- Gateway/átjáró: annak a node-nak a címe, amelyik a maszkon kívüli címek eléréséhez kell (router)
- legyen gépünk ip-je 152.66.25.4, maszk: 255.255.255.0, átjáró: 152.66.25.254
- a forrás és célcímet a maszkkal bináris ÉS művelettel számítjuk

Küldjünk csomagot a 152.66.25.13 célcímre.

```
152. 66. 25. 4 &        152. 66. 25. 13 &
255.255.255.0           255.255.255.0
----------------        ----------------
152.66.25.0             152.66.25.0 ugyanaz jött ki (== igaz)
```

A másik gép közvetlenül (etherneten) elérhető
Kérdezzünk le a 217.20.130.97-címről:

```
152. 66. 25. 4 &        217. 20.130. 97 &
255.255.255.0           255.255.255.0
----------------        ----------------
152.66.25.0             217. 20.130.0 nem ugyanaz (==hamis).
```

A kérést az átjárónak címezzük (152.66.25.254) (etherneten), majd ő továbbítja

- bővítés (címtartomány betelik): ip v6: 8 db 16 bites szám hexadecimálisan, kettőspontok között, 0-k elhagyhatóak
- első 4 szám: hálózat, második 4 szám: ezen belüli gép
- pl. facebook IPv6 címe: 2a03:2880:f10c:83:face:b00c:0:25de

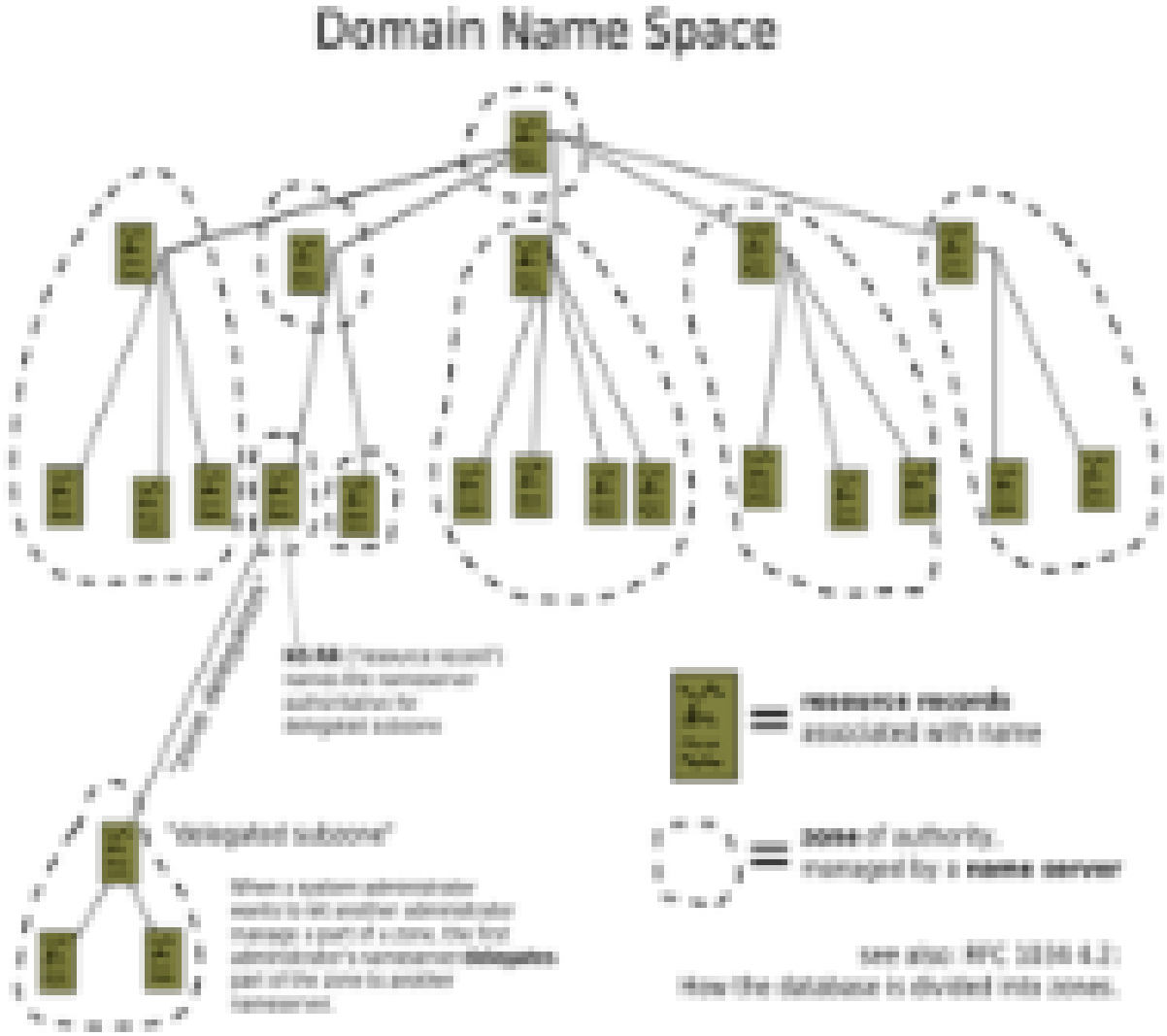
DNS rendszer:

- domain name system
- az emberek számára nehezen megjegyezhetőek az IP címek
- a DNS név alapján találja meg a keresett szerver IP címét
- a gépeket névvel látják el, ezekhez kapcsolódnak az IP címek
- hierarchikus adatszerkezet alakul ki
- a felhasználó beírja egy cím nevét
- ha a cache nem ismeri a címet, akkor le kell kérdezi a helyi DNS kiszolgálót
- ez megkérdezi a gyökér DNS-szervert, az válaszol, majd a helyi DNS kérdez le újra az válaszban adott szervertől
- megkapja a keresett IP címet

Keresés: mogi.bme.hu oldal ip címe ?

- TLD szerver: hu:→ ns.nic.hu

- ns.nic.hu: bme.hu → ns.bme.hu
- ns.bme.hu: mogi.bme.hu→delta.inflab.bme.hu
- delta.inflab.bme.hu→centos.mogi.bme.hu
- (ip: 152.66.24.184)
- reverse lookup: ip címből tud nevet mondani



2.11 Az objektum fogalma, objektum-orientált elvek. Az osztály fogalma. Struktúrák. Tagfüggvények. Konstruktor. Destruktor. Statikus tagok. Barátság, friend függvények.

Objektum orientált elvek:

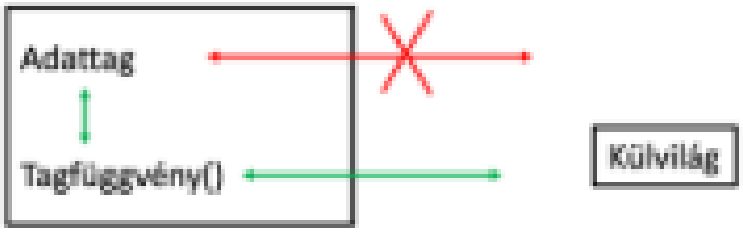
- egységbe zárás (encapsulation)
- adatrejtés (data hiding)
- öröklődés (inheritance)
- sokalakúság (polymorphism)

Egységbe zárás:

- objektum: az adatok (változók) és az azokon dolgozó programrészletek (függvények) egy helyen szerepeljenek
- c++ -ban struct kibővítve függvényekkel → class (osztály)
- az osztály tartalmaz
  - adattagokat (tulajdonságok)
  - tagfüggvényeket (metódus)
    - belső, külső kifejtés (külsőnél hatókör operátor – „::”)
- tagfüggvény a this→ pointeren keresztül éri el a példány adatait

Adatrejtés:

- probléma: a kódban lévő változók szabadon változtathatóak értékadással, hibás tartalom előfordulhat
- megoldás: adatrejtés
- az osztály tagjait csak az osztály tagfüggvényei érik el



- **módosítók (három féle elérés módosító):**
  - private: csak az osztály tagfüggvényei érik el
  - public: a külvilág is eléri (adatrejtés sérült)
  - protected: csak az osztály és annak leszármazottjai érik el
- a módosító addig érvényes, míg egy másik felül nem írja
- class esetén a private az előre bekapcsolt
- struct esetén a public
- szokásos metódusok: getter, setter
  - csak így hívjuk őket, konvenció
  - set beállít, get kiolvas

Öröklődés:

- egy osztály felhasználható arra, hogy egy új, leszármazott osztályt hozzunk létre belőle

- hierarchia alakítható ki
- leszármazottban benne van az ősosztály összes adattagja és metódusa
- plusz adattagokkal és metódusokkal bővíthető
- adattagot, metódust törölni nem lehet

Sokalakúság:

- az örökléssel létrehozott osztály máshogy viselkedhet, mint az ősosztály
- megvalósítás: ugyanolyan nevű és paraméterű metódust tartalmaz, a törzsében más utasításokkal

Konstruktor:

- a példány adatainak inicializálását végzi
- nincs típusa
- neve ugyanaz, mint az osztálynak
- lehet overload-ja (mert egy függvény)
- ha nem írjuk meg, a fordító biztosít 2 konstruktort:
  - alapméretezett üres
  - másoló
- statikus példány is meghívja
- ha a konstruktor egyik paraméterét úgy hívjuk, mint valamelyik adattag, akkor a this→ pointer segít az adattag elérésben

```
osztaly(int adat) { this->adat = adat; }
```

- ha a konstruktorban csak kezdeti értékmegadás van paraméterből, akkor egyszerűsíthető:

```
osztaly(int kezdoadat) : adat(kezdoadat) {}
```

- ilyenkor is ki kell írni a függvény törzsét! {}, különben úgy érzi, hogy nincs kifejtve
- lehet abban az esetben is használni, ha ugyanúgy hívjuk az adattagot, mint a paramétert:

```
osztaly(int adat): adat(adat) {}
```

- ha egy paramétere van a konstruktornak, értékadásként is lehet írni, de ez letiltható az explicit szóval:

```
osztaly o3 = 30; // ha csak 1 paraméter van, így is lehet.
explicit osztaly(int adat) { this->adat = adat; }
```

- másoló konstruktor:
- új példányt hoz létre már létező példányból úgy, hogy az adatterületet a megadott példányéból másolja le
- ahhoz, hogy ne változtassa az egyik dinamikus adattagot tartalmazó osztály változtatása a másik dinamikus adattagot tartalmazóosztályt, referenciákkal kell megírni a másoló konstruktort

Destruktor:

- példány törlésekor biztosítja, hogy a példány ne hagyjon memóriaszemetet maga után
- alapvetően biztosítja a fordító, de ha a konstruktorba dinamikus memóriafoglalás volt, meg kell írni, felszabadítva az ott lefoglalt memóriát
- neve ugyanaz, mint az osztálynak
- nincs típusa
- nincs paramétere
- neve előtt egy ∼ jelzi, hogy destruktor
- nem célszerű kézzel hívni

- statikus példány az őt tartalmazó blokk záró kapcsosánál szűnik meg, dinamikus a delete-nél

Statikus adattagok:

- értéke minden példányban ugyanaz
- memóriában is csak egyszer szerepel
- file szinten és példányban is megadható
- akkor is létezik, ha egyetlen példány sincs az osztályból
- statikus tag elérése osztály::tag
- :: a névtér operátor

```
class atvalto
{
    static double mm;
public:
    static double inchtomm(double inch)
    {
        return atvalto::mm * inch;
    }
};

double atvalto::mm = 25.4; // mm/inch

int main()
{
    double inch,mm;
    cout << "Hany inch a bicikli kerek atmeroje? ";
    cin >> inch;
    mm = atvalto::inchtomm(inch);
    cout << "vagyis " << mm << " mn" << endl;
}
```

Barátság, friend függvények:

- kibúvót jelent az adatrejtés elve alól
- a barátait mindig az osztály válogatja meg, nem a barátok döntik el
- a barátként definiált függvény, vagy osztály függvénye eléri az adatokat
- 3 féle barátság definiálható:
  - egy külső, önálló függvény
  - egy másik osztály
  - egy másik osztály valamelyik metódusa

2.12 Operátorok túlterhelése az objektum orientált programozásban. C++ IO, new, delete operátorok túlterhelésének szabályai. Osztály hierarchiák.

Operátorok:

- osztályhoz tudunk definiálni olyan műveletsort (metódust), mely valamely operátor (műveleti jel) hatására megy végbe
- ezt a metódust operator overload-nak nevezzük
- az operator metódus neve operator és a műveleti jel, amit definiálunk
- nem átdefiniálható:
  - `::` hatókör operátor
  - `.` tag kiválasztás
  - `.*` tag kiválasztás pointerrel
  - `?:` háromoperandusú
  - `sizeof()` objektum mérete
  - `typeid()` objektum típusa
- nem lehet általunk definiált műveleti jelet használni
- egyoperandusúból nem lehet kétoperandusút csinálni (a++ -ból a++2 -t)
- **Kétoperandusú operátor túlterhelése:**
  - pl `+`, `-`, `/`, `*`, `&`, `|`
  - tagfüggvénnyel: a bal oldali operandus az osztály maga, a jobb oldali a függvény paraméter:

    ```
    osztaly operator+(const osztaly& jobb);
    ```

  - barát függvénnyel: a barát függvénynek két paramétere van, a két operandus, először a bal, majd a jobb oldali:

    ```
    friend osztaly operator-(const osztaly&egyik, const osztaly&masik);
    ```

  - kommutatív műveletet megadni a sorrend megadása miatt csak barát függvénnyel lehet:

    ```
    friend complex operator*(const complex& c, const double a); // c*a
    friend complex operator*(const double a, const complex& c); // a*c
    ```

- **Egyoperandusú operátor túlterhelése:**
- pl. `++`
- állhat az operandus előtt (előtag) és után (utótag)
- előtagot tagfüggvénnyel paraméter nélkül, baráttal egy paraméterrel terhelünk túl:

  ```
  osztaly operator ++(); // ++osztalyként lesz meghívva
  friend osztaly operator++(const osztaly& o);
  ```

- utótagnál egy plusz int típusú dummy paraméterrel jelezzük az utótagot:

  ```
  osztaly operator ++(int dummy); // osztaly++-ként lesz meghívva
  friend osztaly operator++(const osztaly& o, int dummy);
  ```

- **Értékadó operátor túlterhelése:**
  - kell, ha a fordító által biztosított gyári adatterület lemásoló nem megfelelő
  - ha másoló konstruktor kellett, ez is kell
  - típusra mutató referenciát kell visszaadni

    ```
    osztaly& operator=(const osztaly& masik);
    ```

- **New operátor túlterhelése:**
  - a new operátor void* visszatérő értékkel, size_t első paraméterrel írandó:

    ```
    void * operator new(size_t n);
    ```

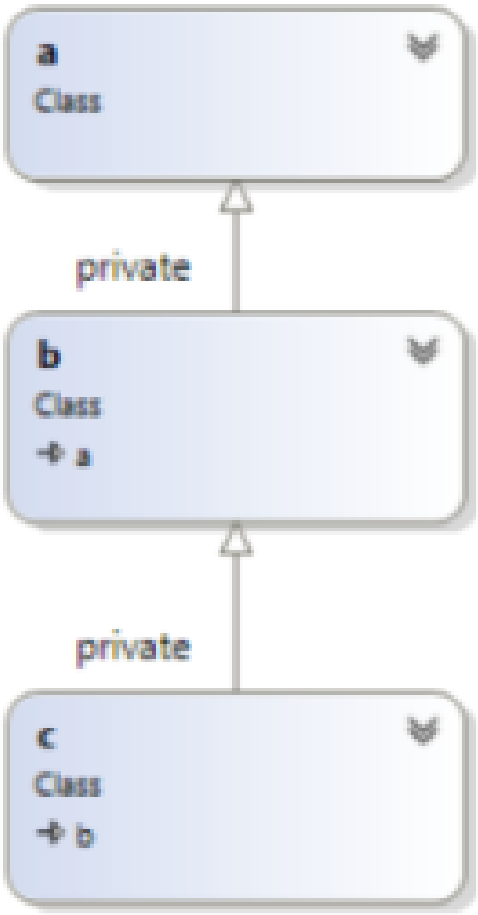
- **Delete operátor túlterhelése:**
  - a delete operátor túlterhelése két paraméterrel írandó: az első a törlendő példányra mutató void*, a második a méret:

    ```
    void operator delete(void *p, size_t n);
    ```
- **I/O operátorok túlterhelése:**
  - ahhoz, hogy az osztályunkat rá lehessen küldeni a cout-ra, a cout-hoz készítenünk kell egy `operator<<()` függvényt
  - cout-hoz nem tudunk tagfüggvényt írni (valaki már megírta az osztályt), így célszerű barát függvényt írni
  - nem a `cin` és `cout`-hoz írjunk barát függvényt, hanem az `istream`-hez és `ostream`-hez, így fájlokhoz is jó lesz
  - az I/O operátorok túlterhelése csak stream-re mutató referenciát visszaadó függvény lehet
  - célszerű, ha az osztályunk barátja

    ```
    friend istream & operator>>(istream &is, osztaly& o); // cin-hez
    friend ostream & operator<<(ostream &os, const osztaly& o); // cout-hoz
    ```
  - ha barát, akkor az osztályban kell megírni a fejlécét

Osztály hierarchiák:

- a leszármazottból újabb osztályok származtathatóak
- felrajzolható az osztályok gráfja
- a hierarchia megtervezése sok időt vesz igénybe
- ez a lánc nem többszörös öröklődés:
- „C”-nek „b” közvetlen bázis osztálya, „a” közvetett bázis osztálya



2.13 Öröklődés, egységbe zárás az objektum-orientált programozásban. Protected osztálytagok. Kompozíció. Aggregáció. Többszörös öröklődés.

Öröklődés:

- egy osztály felhasználható arra, hogy egy új, leszármazott osztályt hozzunk létre belőle
- hierarchia alakítható ki
- leszármazottban benne van az ősosztály összes adattagja és metódusa
- plusz adattagokkal és metódusokkal bővíthető
- adattagot, metódust törölni nem lehet
- ősben lévő kód újrahasznosul, nem kell újra megírni
- az ős változtatása benne lesz a leszármazottban is
- a meglévő kód bővíthető és változtatható is
- `protected` tagok: maradjon meg az adatrejtés elve, de a leszármazottak el tudják érni az ős adatait
- öröklődnek:
  - metódusok, adattagok, operátorok
- nem öröklődnek:
  - konstruktorok
  - destruktorok
  - barát függvények
  - túlterhelt = operátor

Sokalakúság:

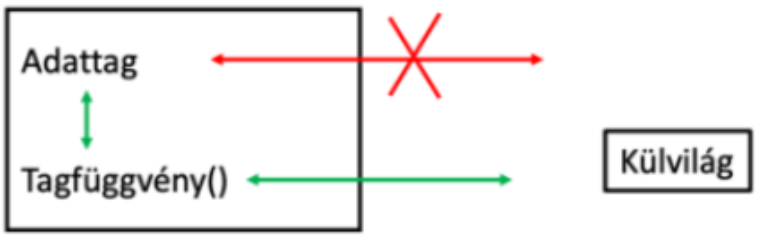
- az örökléssel létrehozott osztály máshogy viselkedhet, mint az ősosztály
- megvalósítás: ugyanolyan nevű és paraméterű metódust tartalmaz, a törzsében más utasításokkal

Egységbe zárás:

- objektum: az adatok (változók) és az azokon dolgozó programrészletek (függvények) egy helyen szerepeljenek
- c++ -ban `struct` kibővítve függvényekkel -> `class` (osztály)
- az osztály tartalmaz
  - adattagokat (tulajdonságok)
  - tagfüggvényeket (metódus)
    - belső, külső kifejtés (külsőnél hatókör operátor – „`::`”)
- tagfüggvény a `this->` pointeren keresztül éri el a példány adatait

Adatrejtés:

- probléma: a kódban lévő változók szabadon változtathatóak értékadással, hibás tartalom előfordulhat
- megoldás: adatrejtés
- az osztály tagjait csak az osztály tagfüggvényei érik el

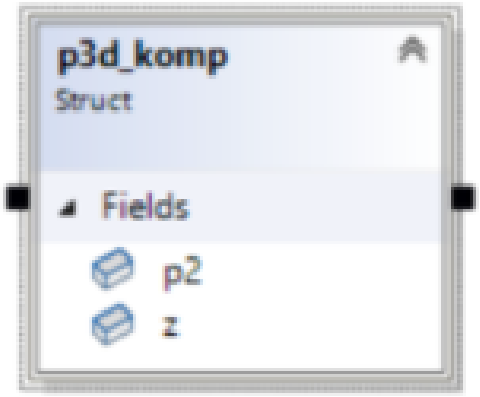


- módosítók (három féle elérés módosító):
  - `private`: csak az osztály tagfüggvényei érik el
  - `public`: a külvilág is eléri (adatrejtés sérült)
  - `protected`: csak az osztály és annak leszármazottjai érik el
- a módosító addig érvényes, míg egy másik felül nem írja
- `class` esetén a `private` az előre bekapcsolt
- `struct` esetén a `public`
- szokásos metódusok: getter, setter
  - csak így hívjuk őket, konvenció
  - `set` beállít, `get` kiolvas

Kompozíció:

- ha egy osztályban statikus adattagként egy másik osztály szerepel
- a másik osztály példánya önállóan nem létezik, csak az első osztály példányában van

```
struct p2d // nem kell public:
{
    double x, y;
};
struct p3d_komp
{
    p2d p2; // kompozíció: statikus adattagja van
    double z; // plusz adattag
};
```



- példányosítás és használat:

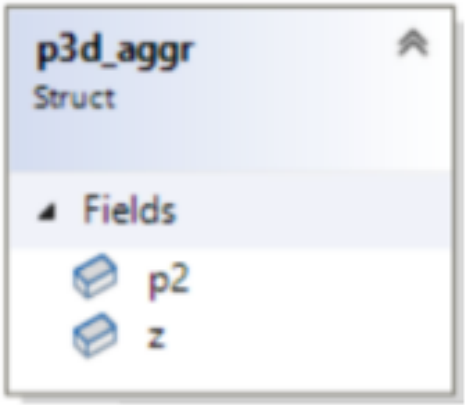
```
p3d_komp p3k;
p3k.p2.x = 0;
p3k.p2.y = 1;
p3k.z = 2;
```

- az `x` és `y` adattagok előtt kell a beépített osztály neve (`p2`)
- a beépített osztályból nem kell önálló példány

Aggregáció:

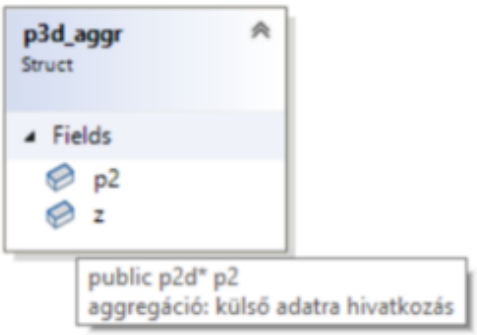
- ha egy osztályban külső adattagként (pointerrel, vagy referenciával) másik osztályra hivatkozás szerepel
- a másik osztály példányának önállóan léteznie kell, csak akkor tudunk rá mutató referenciát/pointert megadni

```
struct p3d_aggr
{
    p2d *p2; // aggregáció: külső adatra hivatkozás
    double z;
};
```



- példányosítás és használat:

```
p3d_aggr p3a;
p2d p2p; // önálló 2d-s pont
p2p.x = 3;
p2p.y = 4; // meglévő pont címe kell!
p3a.p2 = &p2p; // így kap értéket a pointer.
p3a.z = 5;
```

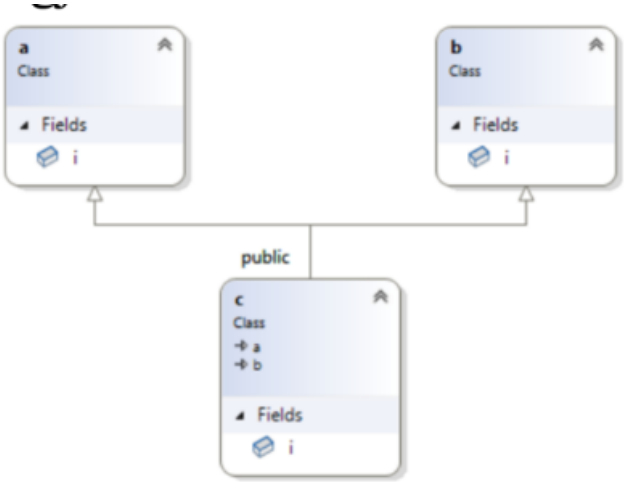


- ha `p3a` dinamikus példány lett volna, akkor kitörlése után `p2p` még létezne
- `p3a`-ban a `p2.x` és `p2.y` változtatása `p2p`-t is változtatja

Többszörös öröklődés:

- multiple inheritance
- öröklődés típusmódosítói:
  - `public` öröklődés:
    - `public` -> `public`
    - `protected` -> `protected`
    - `private` -> nem elérhető
  - `protected`/`private` öröklődés
    - `public` -> `protected`/`private`
    - `protected` -> `protected`/`private`
    - `private` -> nem elérhető
  - ha nem írjuk ki, akkor az öröklődés típusa `private` lesz
- ha a leszármazott osztály másképp viselkedik, mint az ős: polimorfizmus
- egy leszármazottnak több őse is lehet: `class e : a, d { };`
- c++ -ban a jele a vessző:
- ha módosító van, mindkét ős elé ki kell írni
- ha az ősökben azonos nevű tagok szerepelnek, akkor a leszármazottban a fordító nem tudja, melyikről van szó
- ilyenkor a hatókör operátor segít `::`
- legyen az örökösnek is ugyanolyan nevű adattagja:

```
class a
{ public: int i; };
class b
{ public: int i; };
class c : public a, public b
{ public: int i; // ez kicsit túlzó hülyeség
};
cOsztalyuDuplanOroklodottPeldany.a::i = 1;
cOsztalyuDuplanOroklodottPeldany.b::i = 2;
cOsztalyuDuplanOroklodottPeldany.i = 3;
```



2.14 Polimorfizmus az objektum-orientált programozásban. Virtuális alaposztályok. Abstract osztály. Általánosított osztályok.

2.15 Standard Template Library a C++-ban. Tárolók. Bejárók. Algoritmusok. Függvényobjektumok.

2.16 Fuzzy halmazok alapjai, műveletek fuzzy halmazokon.

2.17 Fuzzy következtető módszer, defuzzifikációs módszerek.

2.18 Aggregációs operátorok, általános hatványközép, OWA.

2.19 A .net rendszer részei: GC, CIL, assembly-k. Esemény vezérelt programok felépítése windows alatt.

2.20 Grafikus adattárolás (vektor, raszter), alkatrész modellezési módszerek.

2.21 3D->2D vetítési algoritmusok, a window-viewport transzformáció.

2.22 Görbe közelítési módszerek: természetes spline, Bezier, Catmull-Rom görbék.

2.23 Láthatóság, árnyalás, megvilágítás, színmodellek, anyagmodellek.

2.24 Képfeldolgozás, konvolúció, élkeresés, szegmentálás, alakfelismerés.

2.25 Neurális hálózatok alapjai, a Perceptron, a Perceptron tanítása.

2.26 Felügyelt és felügyelet nélküli tanulás. Mesterséges neurális hálózatok.

2.27 Evolúciós algoritmusok, evolúció stratégiák, genetikus programozás.

2.28 Az ”M” nyelv (Matlab) jellegzetességei: változók, vektorok és mátrixok, feltételes végrehajtás, ciklusok, számtani sorozatok, függvény definíció, diagram rajzolás.

2.29 Ismertesse az alábbi, mechatronikában tanult elvek programmal történő megvalósítását: állapotgép, ARMA modell.