Számábrázolás

A 0 kivételével a valós számok fölírhatók $x = \sigma * B^k * \Sigma x_n * B^{-n}$ (1) alakban, ahol

- σ: előjel
- B: alap
- k: exponens/kitevő/karakterisztika
- Σ: 1-től ∞-ig
- Σ: mantissza
- $0 \geq x_n \geq B-1$

A gyakorlatban az (1)-beli szumma a technológiai korlátok következtében csak véges t értékig mehet, ami ábrázolási pontatlansághoz vezethet.

Hibák fajtái:

- öröklött: a bemenő adatok is hibával terheltek
- képlet: számítás során elkövetett hiba
- kerekítési: a gépi számok végességéből adódó pontatlanság

Számábrázolási módok

1. Fixpontos
   - exponens értéke rögzített
     - pl.: k = 0 esetén [0, 1]-beli számok ábrázolható segítségével

2. Binary Coded Decimals (BDC)
   - minden 10-es számrendszerbeli számjegyet bináris megfelelőjével kódol
   - pl. 73 → 0111 0011

3. Lebegőpontos
   - az ábrázható számok a 0-ra szimmetrikusak
   - egyszeres pontosság (4 byte)
     - exponens: 8 bit, mantissza: 24 bit
       - exponensen tárolható intervallum: [0, 255]
       - hogy a negatív exponenseket is tárolni tudjuk, az ábrázolt tartományt áttranszformáljuk a [-127, 128]-ba
     - a legkisebb és legnagyobb abszolútértékű ábrázolható értékek
       - $|r_{min}| = 2^{min(exponens)} * (1+min(mantissza)) = 2^{-127} * (1+0) = 2^{-127} \approx 10^{-38}$
       - $|r_{max}| = 2^{max(exponens)} * (1+max(mantissza)) = 2^{128} * (1+1-2^{-23}) \approx 2^{129}-2^{105} \approx 10^{38}$
   - dupla pontosság (8 byte)
     - exponens: 12 bit, mantissza: 52 bit
       - az ábrázolható tartomány az egyszeres pontosság analógiájára számítható
   - normalizált alakú számok: a mantissza első számjegye nemnulla
     - 2es számrendszer esetén a nemnulla érték szükségképpen 1, így annak ábrázolása nem hordozna információt → *implicitbit* elhagyása
     - denormalizáltak a 0, NaN, $+\infty$ és $-\infty$

Példa: 162,625d lebegőpontos ábrázolásban

$162{,}625 = 2^7+2^5+2^1+2^{-1}+2^{-3} = 2^8*(2^{-1}+2^{-3}+2^{-7}+2^{-9}+2^{-11})$

0	0	1	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1	1

- első bit: előjelbit (nemnegatív számok esetén 0, egyébként 1)
- mantissza: ($2^{-1}+2^{-3}+2^{-7}+2^{-9}+2^{-11}$) alapján, előjelbitet elhagyva
- exponens: (127+8)d = 135d = 10000111b

Matlab

Változók elnevezése: numerikus és alfabetikus karakterekkel. Első szimbólum szükségképpen alfabetikus; változónév maximális hossza: 63.
Változókkal kapcsolatos utasítások

clear (változónév$_1$, változónév$_2$, ... változónév$_n$): az argumentumban lévő változók törlése

clear: az összes változót törli (kivéve az előre definiáltakat)

Előre definiált változók:
- eps: gépi pontosság
- i, j: komplex számok képzetes része
- pi: π

Munkaterületről elérhető változók kilistázása
- who
- whos: részletesebb információkat kaphatunk a változóinkról
- save fajlnev: az aktuális munkamenet kimentése fajlnev.mat fájlba
- load fajlnev: a fajlnev.mat fájlban lévő munkamenet betöltése

Aritmetikai műveletek: +, -, *, \, /, ^, ’
- mátrixműveletek esetén ezek ’.’-tal való kombinációja: .*, .^, .’, .\, ./
  - elemenkénti műveletvégrehajtást tesz lehetővé
    - pl.: A=[3 2; -3 4];
      - >> B = A*A %% ans = [3 14; -21 10]
      - >> B = A.*A %% ans = [9 4; 9 16]

Beépített függvények: sqrt, sin, cos, tan, sum, exp, lu, ……

Számítások során fölhasznált CPU-idő mérése: tic és toc utasításpár segítségével

Segítség kérése: help *függvénynév* formában

Mátrix elemeinek indexelése
- az indexek 1-től kezdődnek
- elemek elérése: ’()’ segítségével

Példa:

```
%3x3-as A mátrix definiálása
>> A = [1 2 3; 4, 5, 6; 7 8 9];
>> A(2,3) = 10        %A mátrix 2. sorának 3. elemének 10-re állítása; ans = [1 2 3; 4, 5, 10; 7 8 9]
>> A(2,3)             %A mátrix 2. sorának 3. eleme; ans = 10
>> A (5,5) = -3       %A mátrix 5. sorának 5. eleme -3 lesz; nem jelent gondot, hogy megelőzően ennél kisebb volt A
```

>> A(6,6) %lekérdezéskor túlindexelés esetén hibát kapunk
>> A(2:4) %A mátrix 2-4. sorának kezdő elemeit kapjuk ans = [4 7 0]
>> A(2:4, :) % A mátrix 2-4. sorait kapjuk vissza
>> A(2:4, 3:5) % A mátrix 2-4. sorainak 3-5. oszlopában lévő elemeit adja vissza ans = [10 0 0; 9 0 0; 0 0 0]
>> A(1:2:size(A,1), 1:2:size(A,2)) %A mátrix páratlan indexű sorainak páratlan indexű oszlopaiban lévő elemekből képzett mátrix; ans = [1 7 0; 3 9 0; 0 0 -3]

Gyakorlat: Generáljunk 3x4-es véletlen mátrixot -10 és +10 közé eső egész elemekkel!

% rand() [0,1]-beli egyenletes eloszlású pszeudorandomszámokat generál
>> R = rand(3, 4)
% így már -10 és + 10 között vagyunk, csak épp nem egészek az elemek
>> R = -10 + 20 * R
%különféle kerekítéseket kipróbálva
>> round(R)
>> ceil(R)
>> floor(R)
>> fix(R)

Feladat: A mártix 3. sorának véletlenszerűen kiválasztott oszlopaiból állítsunk elő s oszlopmátrixot!

>> rv = logical(round(rand(1, size(A,2)))) %% ceil vagy rand függvény használata nem lenne célravezető
>> s = A(3, rv) %% rv vektor 1-eseinek helyén lévő oszlopok kiválasztása a 3. sorból
>> s = s' %%transzponáljuk s-t, hogy oszlopvektort kapjunk

Vektorok/Mátrixok további létrehozási lehetőségei

>> v = a : b; %% [a, b]-beli egészeket tartalmazó sorvektort kapunk
>> v = a : h : b %% [a, b]-beli értékekből álló sorvektort kapunk, amelynek i-edik eleme az i-1-diknél minden esetben h-val nagyobb
- pl.: v = 10 : -0.3 : 9 %%ans = v = [10, 9.7, 9.4, 9.1]

>> v = linspace(a, b, n) %%[a,b]-beli értékek közül n darabot választ, hogy minden i~=j-re $|v_i - v_j| = (b-a)/(n-1)$
>> ones(n, m) %% n-szer m-es csupa 1-esekből álló mátrixot hoz létre
>> zeros(n, m) %% n-szer m-es csupa 0 álló mátrixot hoz létre
>> E = eye(n, m) %% n-szer m-es mátrixot hoz létre, melynek csupán az e_{ii} elemei 1-esek
>> rand(n,m) %% n-szer m-es mátrixot hoz létre, melynek elemei [0,1]-beli egyenletes eloszlást követő valósok
>> diag(v) %% diagonális mátrixot hoz létre, melynek főátlójában v vektor elemei szerepelnek
>> S = sparse(x,y,v) %% ritka mátrixot hoz létre, ami csupán az x_i-edik sorok y_i-edik pozícióin vesz föl 0-tól különböző, v_i értékeket
- full(S) %% teljes S mátrix kiíratása

>> B = [linspace(1, 5, 6); 5:-0.2:4;sin(pi), 1:5] %%az eddigiek kombinálása

Vektorok mint polinomok

A vektorokat értelmezhetjük, mint egy polinom csökkenő hatványkitevőjű változójához tartozó együtthatóinak sorozatát.
Pl. a $p(x) = -4x^2 + x^3 + 12 = 1x^3 - 4x^2 + 0x^1 + 12x^0$ polinomnak megfeleltethető vektor a p = [1 -4 0 12].

Polinomokkal Matlabban végezhető műveletek

```
>> p = [1 -4 0 12];
>> polyval(p, 5)      %% p polinom 5-ben vett helyettesítési értékét adja meg
>> r = roots(p)       %% p polinom gyökeit határozza meg
>> q = poly(r)        %% meghatározza azon p polinomot, melynek gyöke r
>> conv(p, q)         %% p polinom szorozva q polinommal  ans = 1.0000, -8.0000, 16.0000, 24.0000, -96.0000, -0.0000, 144.0000
>>deconv([1 0 0 -1], [1 -1]);        %% x^3-1 polinom és x-1 hányadosa
```

m. fájlok

Matlab utasításaink sorozatát .m végződésű szöveges fájlba menthetjük. Amennyiben fájlunk a *function visszatérési_érték(ek) = függvénynév(argumentum_lista)* sorral kezdődik, majd azt függvénynév.m néven mentjük el, úgy egy a későbbiekben Matlab munkamenetből is meghívható függvényt készítettünk. A függvényben használt változók lokálisak.
Abban az esetben, ha .m fájlunk első sorának felépítése nem igazodik az előbb leírtakhoz, úgy egyszerűen végrehajtandó parancsok sorozatát definiálhatjuk .m fájlunkban. Ebben az esetben a műveletek az aktuális munkamenet változóin hajtódnak végre, értékük módosítása hatással lesz a továbbiakra is.
Mindkét típusú .m fájlra találtok mintát a honlapon: a függvénytípusúra a fibonacci sorozat n-edik tagját kiszámítók (fiboRec.m, fibo.m, fastFibo.m), a nem függvényt definiáló fájlra pedig a selectRows.m.

Vezérlési szerkezetek:

- if
- while
- for
- break: segítségével kiléphetünk egy ciklusmagból

A vezérlési szerkezetekre példát a honlapon lévő .m-fájlokban találhattok. Az if, while és a for utasításokban egyaránt közös, hogy azokat az end kulcsszóval kell lezárni.

Feltételes kifejezések:

- Pl.: x<=8
- Használható operátorok: ~(tagadás), & (logikai és), | (logikai vagy), <, >, <=, >=, ==

Függvények ábrázolása

- plot(x, y, 'kirajzolás_módja')
  - x: a kiértékelési pontokat tartalmazó vektor
  - y: a függvény x_i pontokban vett kiértékelési értéke

    - ’kirajzolás_módja’ sztringgel határozhatjuk meg a megjelenítés egyes paramétereit, mint pl. az egyenes típusa, vagy színe
        - ’g*’ hatására pl. zöld színű csillagok jelzik majd az ábrázolt függvényt
- fplot(’fuggvenynev’, [a b])
    - hatására ’fuggvenynev’ függvény megjelenik [a b] intervallumon ábrázolva (alapértelmezés szerint 25 pontban történő kiértékelés alapján)
- grid
    - az utasítás kiadásával berácsozhatjuk a megjelenített grafikon hátterét
- Példa: a honlapon multiPlot.m néven futó fájlban látható
    - szerepel benne a subplot utasítás, ami hatására egy ablakban több függvény is külön-külön ábrázolható
    - a példában a plot utasítás harmadik argumentumaként megfigyelhető a kirajzolás módjának módosítási lehetőségei

PLUSZPONT szerzési lehetőség (csoportonként az első 5 helyes beküldőnek):

A honlapról letölthető selectRows.m fájl a munkamenetben létrehozott M mátrix soraiból választ ki véletlenszerűen sorokat 500 alkalommal, majd a konzolon megjeleníti az egy alkalommal átlagosan kiválasztott sorok számát.

Feladat:

1) Válaszold meg, hogy a rand() függvénnyel egyenletesen generált véletlen számok ellenére miért 0 mégis az átlagosan kiválasztott sorok száma! (Gyakorlaton is elhangzott)
2) A selectRows.m egyetlen sorának módosításával érd el, hogy az M mátrixból *véletlenszerűen* kiválasztott sorok száma M összes sorának 20%-a legyen (vagyis egy 1000*1000-es M mátrixból átlagosan 200 sor kerüljön kiválasztásra)!
3) Az előbbiekben módosított sor megváltoztatása, hogy az mind a négy MATLAB-os kerekítési eljárás használata mellett hasonlóan viselkedjen (ti. sorok 20%-át választja ki)!

A megoldások (amik az 1) kérdésre adott válaszból és 2) és 3) pontokban leírtakat megvalósító sorokból állnak) a következő gyakorlatig küldhetitek be, EHA-kód, név és a gyakorlatotok időpontjának társaságában.

Lineáris egyenletrendszerek megoldása és LU-felbontás

Általános alak: $A\mathbf{x} = \mathbf{b}$

Példa:

$2x1 - x2 + 2x3 = 6$
$4x1 - x2 + 6x3 = 20$
$6x1 - 7x2 + 8x3 = 16$

Ebben az esetben A = [2 -1 2; 4 -1 6; 6 -7 8] és b = [6; 20; 16].

Az első sor -2-szeresének, ill. -3-szorosának a második, ill. harmadik sorhoz való hozzáadása az első oszlop főátlója alatti elemeinek kinullázódását eredményezi. A művelethez tartozó eliminációs mátrix:

$$M_1 = \begin{matrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -3 & 0 & 1 \end{matrix}$$

$$M_1*A = \begin{matrix} 2 & -1 & 2 \\ 0 & 1 & 2 \\ 0 & -4 & 2 \end{matrix}$$

A következő (második) oszlop főátló alatti elemei az M2 eliminációs mátrixszal tehetők nullává.

$$M_2 = \begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 4 & 1 \end{matrix}$$

$$M_2*M_1*A = \begin{matrix} 2 & -1 & 2 \\ 0 & 1 & 2 \\ 0 & 0 & 10 \end{matrix}$$

M_2*M_1*A már felső trianguláris mátrix, azaz $M_2*M_1*A = U$. Az egyenlőséget balról (M_2*M_1) inverzével szorozva $A = (M_2*M_1)^{-1}*U = M_1^{-1}*M_2^{-1}*U = L*U$-t kapjuk. Az eliminációs mátrixok jó tulajdonságait kihasználva:

$$L = M_1^{-1} * M_2^{-1} = \begin{matrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 0 & 1 \end{matrix} * \begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -4 & 1 \end{matrix} = \begin{matrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & -4 & 1 \end{matrix}$$

$A = L*U$ –t behelyettesítve $A\mathbf{x} = \mathbf{b}$-be, $L*U*\mathbf{x} = \mathbf{b}$ adódik. Bevezetve y mesterséges változót, először $L*\mathbf{y} = \mathbf{b}$-t, majd $U*\mathbf{x} = \mathbf{y}$-t megoldva megkapjuk az eredeti egyenlőségrendszer **x** megoldását.

1. lépés

$$\begin{array}{lllllllll} 1*y_1 & + & 0*y_2 & + & 0*y_3 & = & 6 & \rightarrow & y_1 = 6 \\ 2*y_1 & + & 1*y_2 & + & 0*y_3 & = & 20 & \rightarrow & y_2 = 8 \\ 3*y_1 & - & 4*y_2 & + & 1*y_3 & = & 16 & \rightarrow & y_3 = 30 \end{array}$$

$$y = \begin{matrix} 6 \\ 8 \\ 30 \end{matrix}$$

2. lépés

$$\begin{array}{lllllllll} 2*x_1 & - & 1*x_2 & + & 2*x_3 & = & 6 & \rightarrow & x_1 = 1 \\ 0*x_1 & + & 1*x_2 & + & 2*x_3 & = & 8 & \rightarrow & x_2 = 2 \\ 0*x_1 & + & 0*x_2 & + & 10*x_3 & = & 30 & \rightarrow & x_3 = 3 \end{array}$$

Vegyük észre, hogy az első lépés után adódó $y = [6\ 8\ 30]^T$ megoldásvektor megkapható az egyenletrendszer kezdeti jobboldalának vektorából is (**b**), az együtthatómátrixon elvégzett átalakításokat végrehajtva rajta $\mathbf{y} = M_2*M_1*\mathbf{b}$ vektorral.