Opakovacia hodina

Hodina je venovaná opakovaniu základných postupov pri tvorbe programov. Cieľom hodiny je aplikovanie nadobudnutých vedomostí o základoch riadenia toku programu, práci s maticami a tvorbe grafov.

Príklad 1

Rozpustenie Cu v zriedenom roztoku kyseliny dusičnej prebieha podľa nasledujúcej rovnice:

$$Cu + HNO_3 \rightarrow Cu(NO_3)_2 + NO + H_2O$$

Vypočítajte stechiometrické koeficienty rovnice.

Postup:
Zapíšeme bilancie jednotlivých atómov s tým, že reaktanty budú mať kladnú hodnotu a produkty zápornú. Keďže suma počtu atómov každého druhu musí byť rovná nule, bilančné rovnice sú v tvare:

$$a.Cu + b.HNO_3 \rightarrow c.Cu(NO_3)_2 + d.NO + e.H_2O$$

Cu	1.a	+	0.b	-	1.c	-	0.d	-	0.e	=	0
H	0.a	+	1.b	-	0.c	-	0.d	-	2.e	=	0
O	0.a	+	3.b	-	6.c	-	1.d	-	1.e	=	0
N	0.a	+	1.b	-	2.c	-	1.d	-	0.e	=	0

V tejto chvíli sme vygenerovali systém 4 rovníc, ale máme 5 neznámych. Musíme vygenerovať ďalšiu rovnicu. Dodatočná rovnica môže byť v tvare a=1. Takouto rovnicou postulujeme, že koeficient *a* v rovnici bude rovný 1.

Kontrolná otázka 1: Môže byť táto voľba koeficienta *a* ľubovoľná?
Odpoveď: V podstate áno (nesmie samozrejme mať hodnotu 0). Parameter *a* môže mať aj hodnotu 3,14, ale je jasné, že takáto voľba by nebola vôbec vhodná. Ak ako parameter použijeme hodnotu rovnú 1, nemôžeme nič pokaziť.

Kontrolná otázka 2: Možeme vytvoriť dodatočnú rovnicu v tvare b=1 alebo c=1?
Odpoveď: áno, táto voľba je len na nás. Výsledky v konečnom dôsledku musia byť rovnaké.

Upravený bilančný systém:

Cu	1.a	+	0.b	-	1.c	-	0.d	-	0.e	=	0
H	0.a	+	1.b	-	0.c	-	0.d	-	2.e	=	0
O	0.a	+	3.b	-	6.c	-	1.d	-	1.e	=	0
N	0.a	+	1.b	-	2.c	-	1.d	-	0.e	=	0
Dodatočná rovnica	1.a	+	0.b	-	0.c	-	0.d	-	0.e	=	1

V maticovom tvare

Q X P

$$\begin{pmatrix} 1 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & -2 \\ 0 & 3 & -6 & -1 & -1 \\ 0 & 1 & -2 & -1 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} a \\ b \\ c \\ d \\ e \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

Riešenie v MATLABE:

Do nového súboru vložíme nasledujúce riadky:

```
Q=[1 0 -1 0 0;0 1 0 0 -2;0 3 -6 -1 -1;0 1 -2 -1 -0;1 0 0 0 0]
P=[0;0;0;0;1]
X=Q\P
```

Výpis programu je nasledujúci:

```
Q =
1  0 -1  0  0
0  1  0  0 -2
0  3 -6 -1 -1
0  1 -2 -1  0
1  0  0   0  0

P =
0
0
0
0
1

X =
1.0000
2.6667
1.0000
0.6667
1.3333
```

Výsledok, ktorý sme takto získali je správny a môžeme ho dosadiť do rovnice:

$$1.0000Cu + 2.6667HNO_3 \rightarrow 1.0000Cu(NO_3)_2 + 0.6667NO + 1.3333H_2O$$

Akú má tento zápis nevýhodu? Klasickému chladnokrvnému anorganickému chemikovi by takýto zápis spôsobil srdcový infarkt. Musíme koeficienty určitým spôsobom upraviť tak, aby sme získali celočíselné hodnoty.
Matlab má implementovanú funkciu **rat**, ktorá aproximuje reálne číslo na racionálne.

V našom prípade:

```
[citatel,menovatel]=rat(X)
```

```
citatel =
3
1
3
1
1

menovatel =
8
1
8
4
2
```

Od teraz si budeme pamätať !!!

Na prevod reálneho čísla na racionálne je možné použiť funkciu **rat**

Príklad použitia:

```
A=[0.5 0.3333333 0.66666666]
[cit,men]=rat(A)

A =

0.5000 0.3333 0.6667

cit =
1 1 2

men =
2 3 3
```

Ako je teraz možné využiť skutočnosť, že máme znalosti o racionálnych tvaroch hodnôt koeficientov? Najvhodnejšie by bolo vynásobenie celého vektora najmenším spoločným násobkom. No to by si opäť vyžadovalo vytvorenie vlastného programu na výpočet najmenšieho spoločného násobku. To je pre nás v tejto chvíli nie práve najvhodnejšie. Preto zostrojíme vlastný algoritmus, ktorý bude schopný upraviť koeficienty rovnice na celé čísla. Využijeme pritom znalosť racionálneho tvaru koeficientov. Budeme vyžadovať, aby hodnoty menovateľov všetkých koeficientov boli rovné jednotke (tým bude splnená podmienka celočíselných koeficientov).

Daný algoritmus je v podstate veľmi jednoduchý. Využíva cyklické násobenie všetkých prvkov vektora *X* maximálnou hodnotou vektora menovateľov dovtedy, pokiaľ všetky hodnoty v menovateli nie sú rovné 1 (v danom prípade je to zabezpečené podmienkou sum(men)~=5).

```
Q=[1 0 -1 0 0;0 1 0 0 -2;0 3 -6 -1 -1;0 1 -2 -1 -0;1 0 0 0 0];
P=[0;0;0;0;1];
X=Q\P;
[cit,men]=rat(X);
while(sum(men)~=5)
```

```
X=X.*max(men);
[cit,men]=rat(X);
end
disp(X);
```

Výsledok v hlavnom okne MATLABu je nasledujúci:

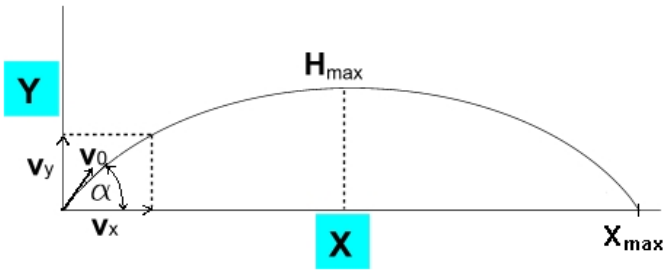
3.0000
8.0000
3.0000
2.0000
4.0000

Finálny tvar rovnice bude teda: $3Cu + 8HNO_3 \rightarrow 3Cu(NO_3)_2 + 2NO + 4H_2O$

Príklad 2

Výpočet trajektórie šikmého vrhu

Šikmý vrh (vo vákuu) je možné znázorniť nasledujúcou schémou:



Kde význam jednotlivých symbolov je nasledovný:

X	x-ová súradnica
Y	y-ová súradnica
V_0	počiatočná rýchlosť
Vx	rýchlosť v smere *x*
Vy	rýchlosť v smere *y*
H_{max}	maximálna dosiahnutá výška
X_{max}	dosiahnutá vzdialenosť pri dopade
α	uhol

Pre jednotlivé súradnice platí:

$X=v_x.t$ (1)
$Y=v_y.t - 1/2.g.t^2=v_0.\sin\alpha.t - 1/2.g.t^2$ (2)

$v_x=\cos\alpha.v_0$ (3)
$v_y=\sin\alpha.v_0 - g.t$ (4)

Čas dopadu

Veľmi dôležitý parameter, ktorý potrebujeme vypočítať je hodnota času, v ktorom predmet dopadne ***tD***. V bode dopadu má *Y*-ová súradnica nulovú hodnotu. To znamená, môžme ju získať riešením upravenej rovnice (2):

$0=v_0.\sin\alpha.t - 1/2.g.t^2$

Daná rovnica má dva korene: 0 a *tD*.

Aby sme získali hodnotu vzdialenosti dopadu ***Xmax***, dosadíme hodnotu *tD* do rovnice (1):

$X_{max}=v_x.tD$

Program v MATLABe schopný vypočítať čas dopadu a vzdialenosť dopadu je nasledovný:

```
g=9.81;
v0=25;
uhol=30;
tD_pomocny=roots([-0.5*g v0*sind(uhol) 0]);
disp('cas dopadu je:');
tD=max(tD_pomocny);
disp(tD);
disp('vzdialenost dopadu je:')
vx=cosd(uhol)*v0;
Xmax=vx*tD;
disp(Xmax);
```

Poznámka: goniometrické funkcie sin, cos, ... pracujú s rozmerom uhlov v radiánoch, písmeno *d* sa pridáva (sind, cosd, ...) za účelom použitia uhlov v stupňoch. Staršie verzie MATLABu funkcie sind, cosd, ... nepoznajú.

Maximálna výška a čas dosiahnutia maximálnej výšky

V bode ***Hmax*** je *Y*-ová zložka rýchlosti rovná nule. To využijeme pri výpočte času dosiahnutia maximálnej výšky. Dosadením nulovej hodnoty do rovnice (4) a následnou úpravou dostávame vzťah na výpočet času dosiahnutia maximálnej výšky ***tHmax***:

$v_y=0=\sin\alpha.v_0 - g.t$

úpravou dostávame:

$tH_{max}=\sin\alpha.v_0/g$

Hodnotu maximálnej výšky následne určíme dosadením času ***tHmax*** do rovnice (2):

$H_{max}= \sin\alpha.v_0.tH_{max} - 1/2.g.tH_{max}^2$

K programu pridáme nasledujúce riadky, ktoré vypočítajú hodnotu maximálnej výšky a čas, v ktorom sa maximálna výška dosiahne:

```
disp('cas v ktorom sa dosiahne maximalna vyska je:');
tH_max=v0*sind(uhol)/g;
```

```
disp(tH_max);
disp('maximalna vyska je:');
H_max=v0*sind(uhol)*tH_max-0.5*g*tH_max^2;
disp(H_max);
```

Posledným krokom je vykonanie výpočtu hodnôt *X* a *Y* v čase od 0 sekúnd po čas dopadu *tD*.

Celý program vyzerá nasledovne:

```
g=9.81;
v0=25;
uhol=30;
tD_pomocny=roots([-0.5*g v0*sind(uhol) 0]);
disp('cas dopadu je:');
tD=max(tD_pomocny);
disp(tD);
disp('vzdialenost dopadu je:')
vx=cosd(uhol)*v0;
Xmax=vx*tD;
disp(Xmax);
disp('cas v ktorom sa dosiahne maximalna vyska je:');
tH_max=v0*sind(uhol)/g;
disp(tH_max);
disp('maximalna vyska je:');
H_max=v0*sind(uhol)*tH_max-0.5*g*tH_max^2;
disp(H_max);
t_simulacie=linspace(0,tD,100);
for i=1:100
    t=t_simulacie(i);
    X(i)=vx*t;
    Y(i)=v0*sind(uhol)*t-0.5*g*t^2;
end
plot(X,Y)
title('Trajektoria sikmeho vrhu')
xlabel('Vzdialenost [m]');
ylabel('Vyska [m]');
```

Výsledný graf trajektórie: