Analýza a informatizácia dynamických systémov

ERIK KUČERA | 3APIN

Rozdelenie predmetu

Diskrétne udalostné systémy

Cca 1 týždeň

Konečné automaty

Modelovanie výrobných, komunikačných a iných diskrétnych udalostných systémov (DES) pomocou konečných automatov

Spojité systémy

Cca 9 týždňov

Základy spojitých systémov, Laplaceova transformácia

Modelovanie spojitých systémov

Riadenie spojitých systémov, stabilita

Konečné automaty a modelovanie DES

ERIK KUČERA

ANALÝZA A INFORMATIZÁCIA DYNAMICKÝCH SYSTÉMOV | PREDNÁŠKA 2

SYSTÉM

- ***skupina objektov vydelená z univerza***

POCHOPENIE FYZIKÁLNYCH JAVOV

↓

SPRÁVANIE SA SYSTÉMU V ČASE

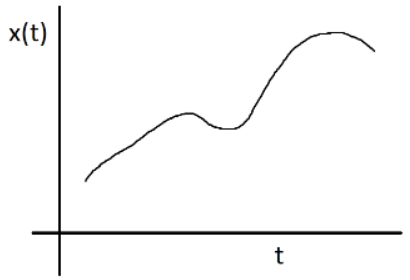
↓

RIADENIE SYSTÉMU?

MODELOVANIE SYSTÉMU

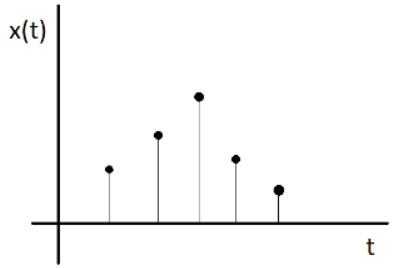
KLASIFIKÁCIA SYSTÉMOV

Spojitý systém



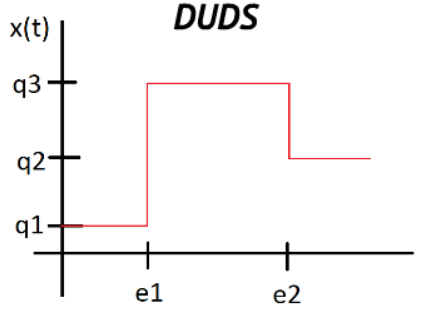
- varenie vody, napúšťanie nádrže...

Diskretizovaný systém



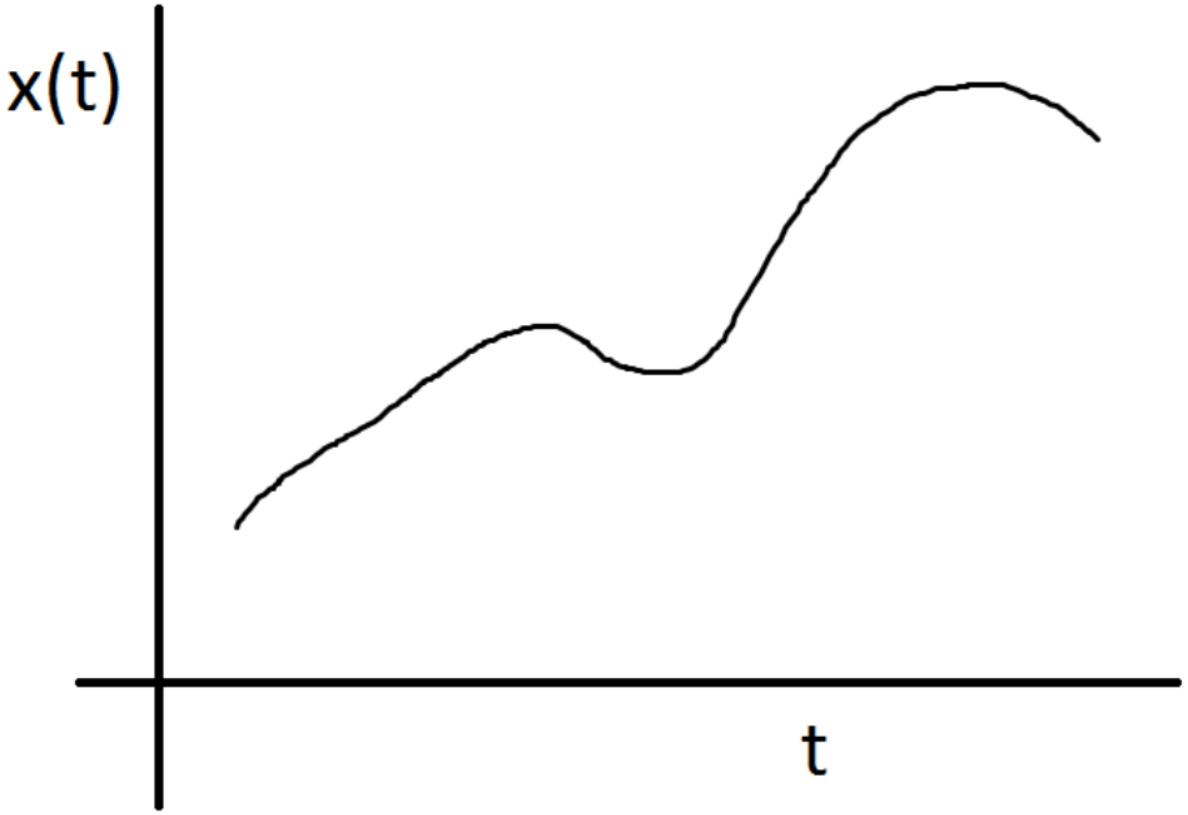
- sledovanie teploty vody v rôznych časových okamihoch

Diskrétny udalostný dynamický systém DUDS



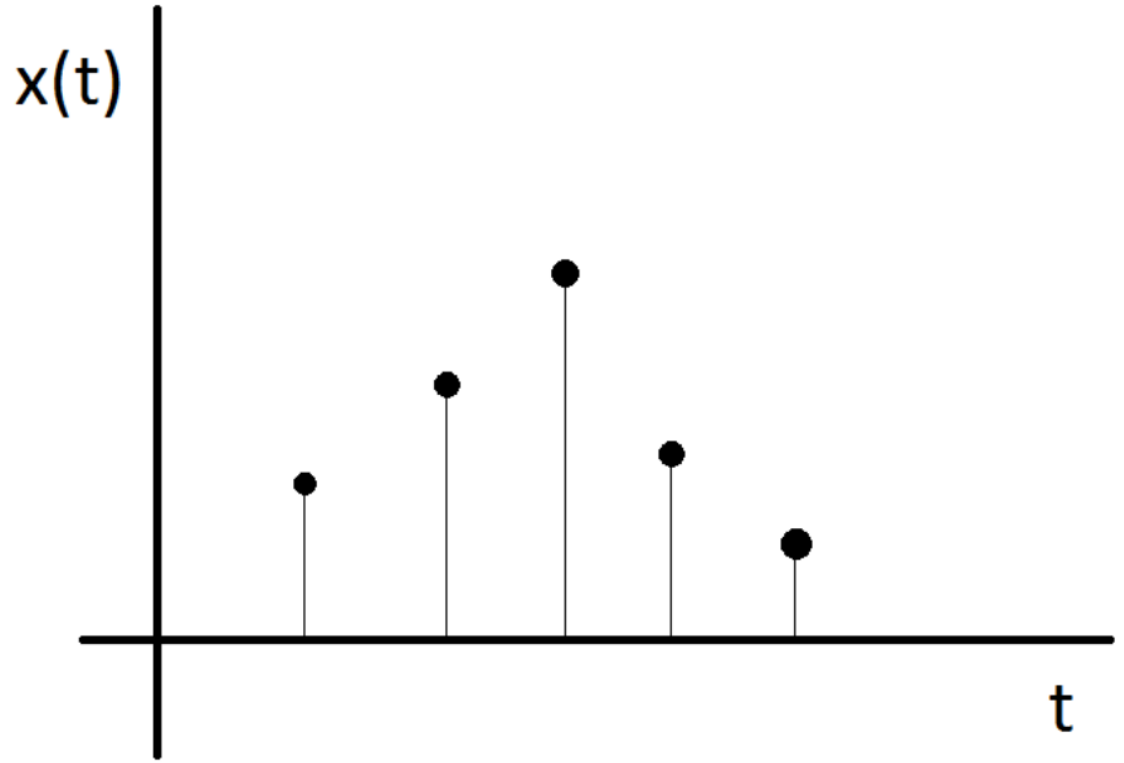
OBLASŤ NÁŠHO ZÁUJMU

Spojitý systém



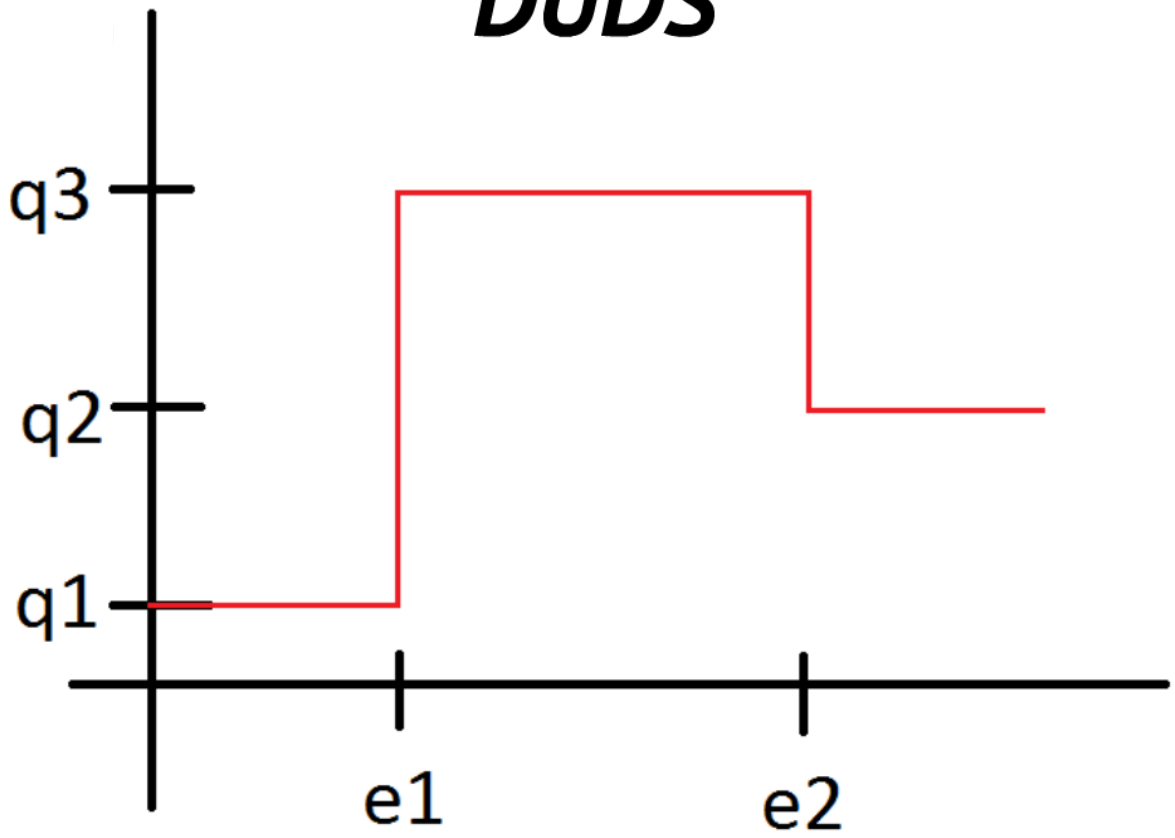
- varenie vody, napúšťanie nádrže...

Diskretizovaný systém



- sledovanie teploty vody v rôznych časových okamihoch

Diskrétny udalostný dynamický systém DUDS



OBLASŤ NÁŠHO ZÁUJMU

DUDS

Charakter diskrétnych dynamických udalostných systémov

- ***množina stavov, do ktorých sa môže systém dostať***
- ***množina udalostí, ktorým zodpovedajú prechody medzi stavmi***

Úlohou teórie DUDS je tvorba adekvátnych opisov - MODELOV, ktoré opisujú dynamiku daného DUDS.

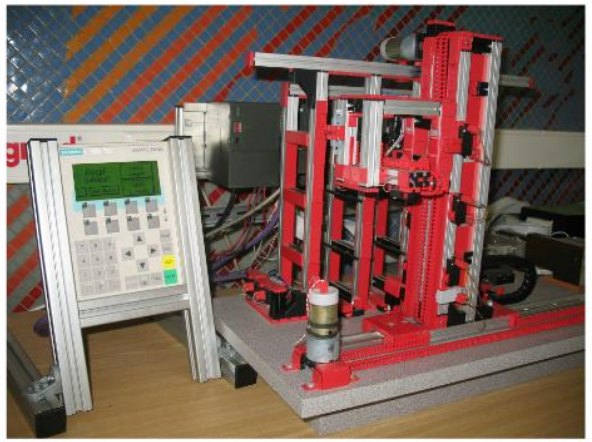
Rôzne označenia

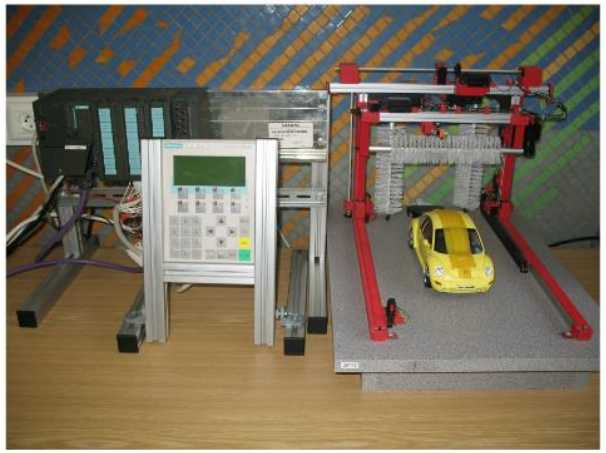
- Diskrétne udalostné systémy (DUS)
- Diskrétne udalostné dynamické systémy (DUDS)
- Udalostný systém (US)
- Discrete event systems (DES)
- Discrete event dynamic systems (DEDS)

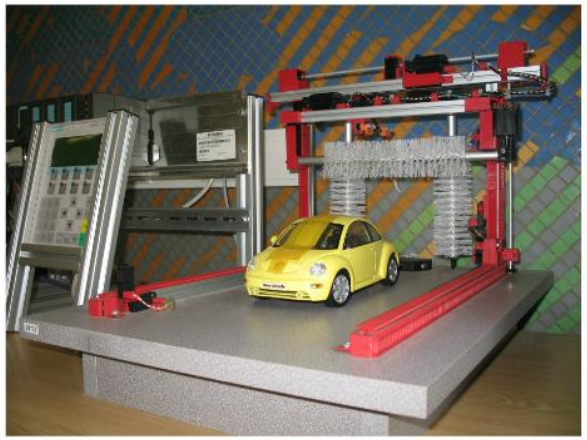
- Tieto označenia sú spravidla ekvivalentné

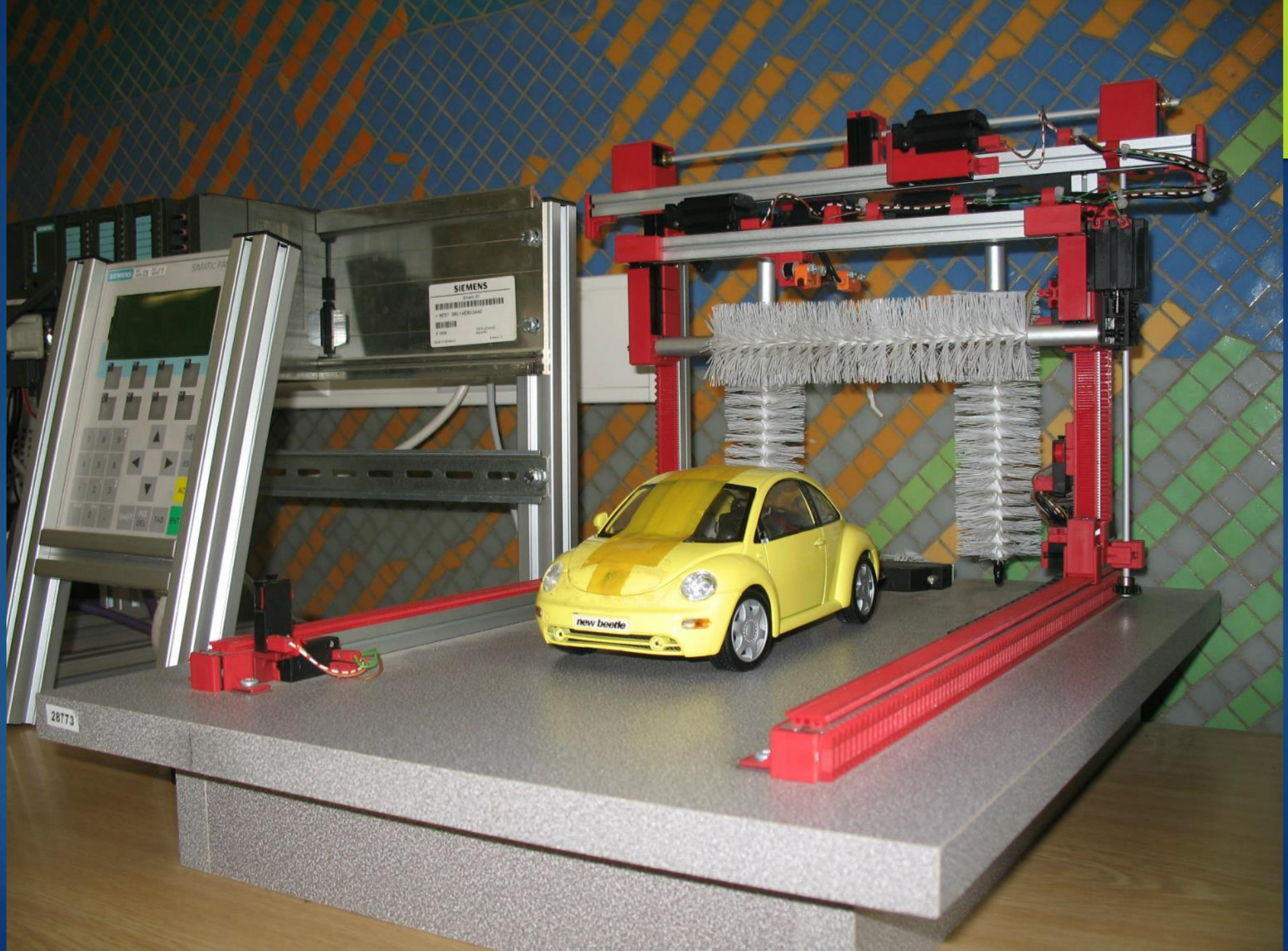
LABORATÓRIUM UDALOSTNÝCH SYSTÉMOV

URČENÉ NA VÝUČBU UDALOSTNÝCH SYSTÉMOV PRE PROGRAMY KYBERNETIKA A ROBOTIKA





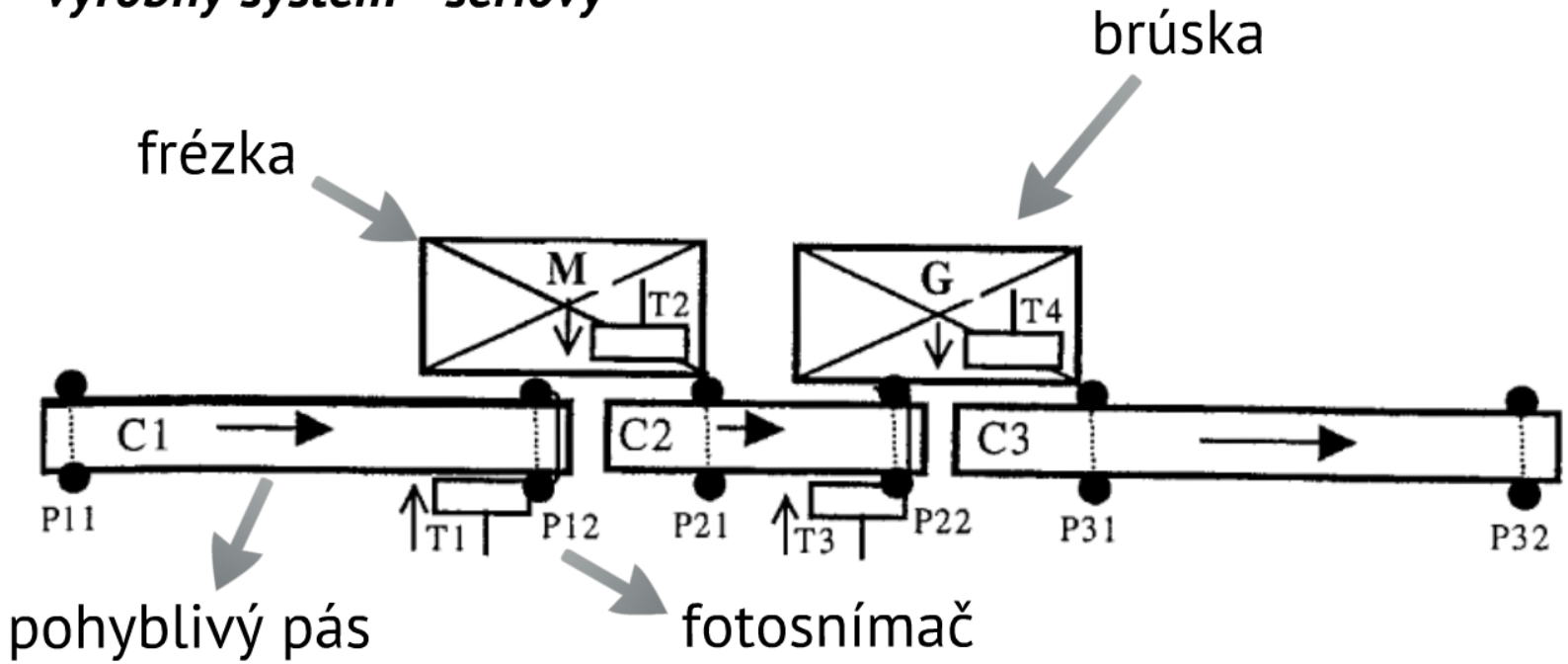



SIEMENS
new beetle
28773

DUDS

Príklady takýchto systémov

- ***výrobný systém - sériový***



DUDS

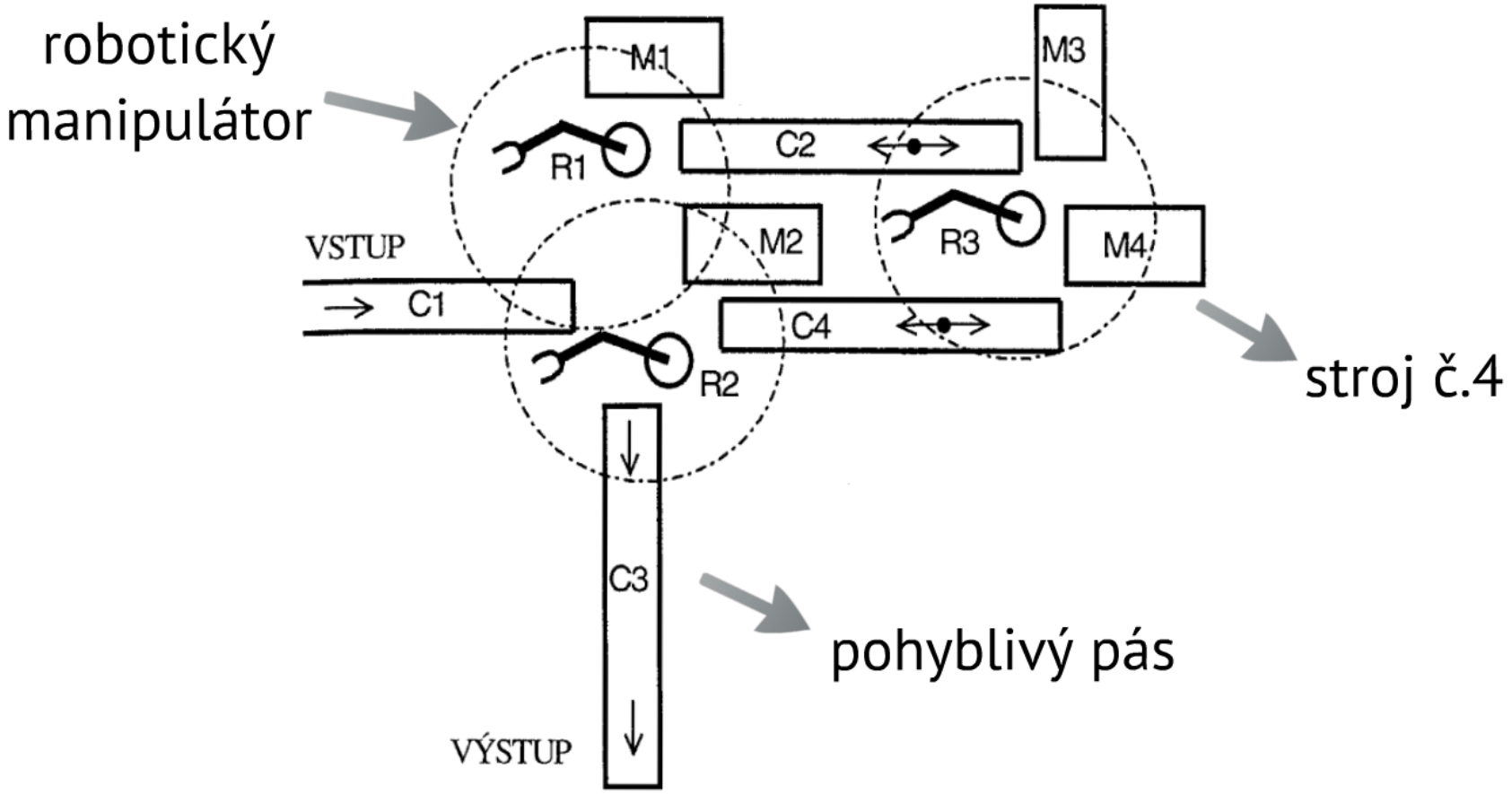
A čo keby bolo treba každý výrobok opracovať inak?

POŽADOVANÉ TECHNOLOGICKÉ POSTUPY

	Výrobok			
Operácia	**A**	**B**	**C**	**D**
1	M1	M3	M2	M1
2	M2 alebo M3	M2 alebo M4	M4	M3
3	M4	M3 alebo M4	M3 alebo M4	M3 alebo M4
4		M4		M2

DUDS

Riešenie - systém so sériovo-paralelnou štruktúrou



Vlastnosti DUDS

Ďalšie príklady DUDS:

- ***pružné výrobné systémy, číslicové počítače, dopravné systémy pozemné, vzdušné, vodné atď.***
- ***oblasti uplatnenia problematiky sú veľmi široké (nielen workflow!)***

Skúmanie vlastností DUDS:

- ***synchronizácia udalostí, súbežnosť, paralelizmus, mŕtve stavy systému, živosť systému, reverzibilnosť, dosiahnuteľnosť stavov, rozvrhovanie udalostí a iné...***

Modelovanie DUDS

Vybrané nástroje na modelovanie:

- ***konečné automaty***
- ***Grafcet***
- ***stavové diagramy***
- ***rebríkové diagramy***
- ***Petriho siete***

Matematické grafy a ich využitie pri špecifikácii systémov.

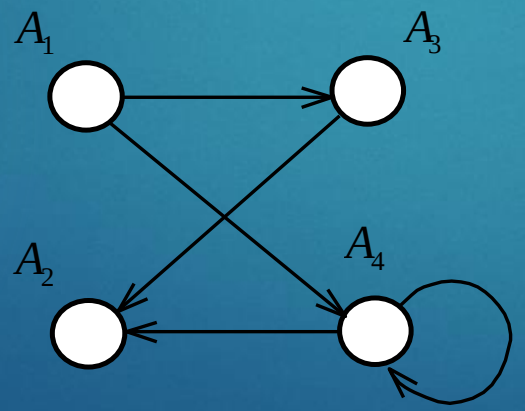
Definícia 1.

Jednoduchý neohodnotený orientovaný matematický graf je daný usporiadanou dvojicou

$$G = (A, R)$$

kde *A* je konečná neprázdna množina prvkov nazývaných uzlami grafu,
R je binárna relácia na *A* (môže byť aj prázdna).

Príklad.



Definícia 2.

Jednoduchý ohodnotený orientovaný matematický graf je 6-tica

$$G = (A, R, f_1, f_2, S_1, S_2)$$

kde A je konečná neprázdna množina uzlov

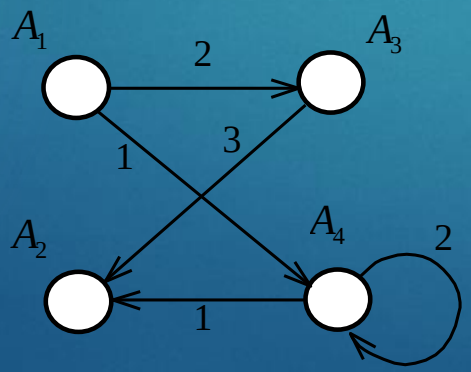
R je relácia na množine A, ktorá môže byť aj prázdna

f_1 je funkcia $A \to S_1$

f_2 je funkcia $R \to S_2$

S_1, S_2 sú množiny, pričom jedna, prípadne obe môžu byť prázdne

Príklad.



Definícia.

Deterministický konečný automat (DKA) je pätica

$$A = (\Sigma, Q, q_0, \delta, F)$$

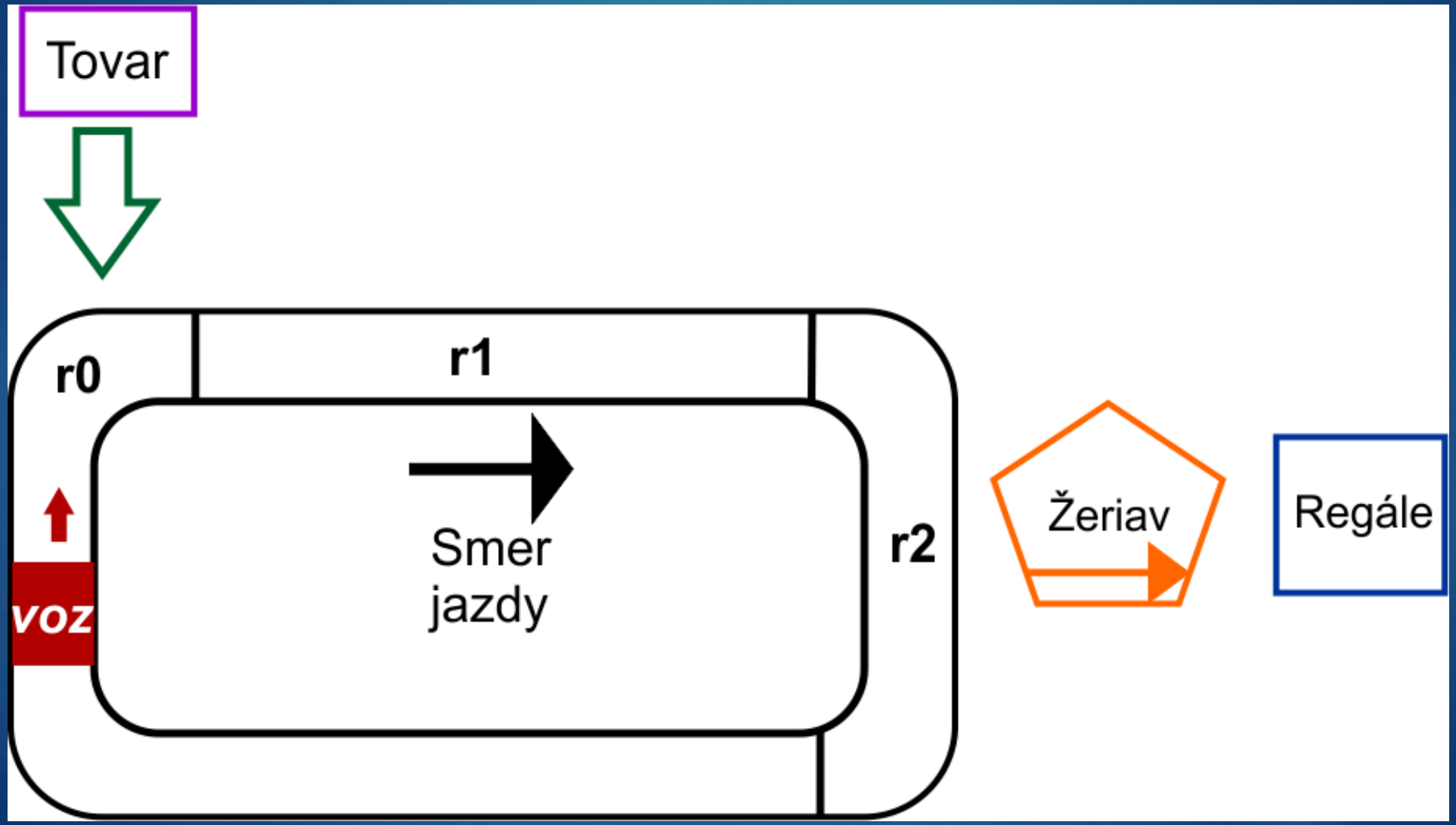
kde

1. Σ je neprázdna množina udalostí,
2. Q je neprázdna množina prvkov nazývaných stavy,
3. $q_0 \in Q$ počiatočný (alebo štartovací) stav,
4. δ je parciálna prechodová funkcia daná výrazom $\delta : Q \times \Sigma \to Q$,
5. F je množina cieľových stavov daná ako podmnožina Q: $F \subseteq Q$, kde F môže byť prázdna množina.

Konečný automat môžeme zapísať ako **usporiadanú päticu** $(Q, \sum, \sigma, q_0, F)$:

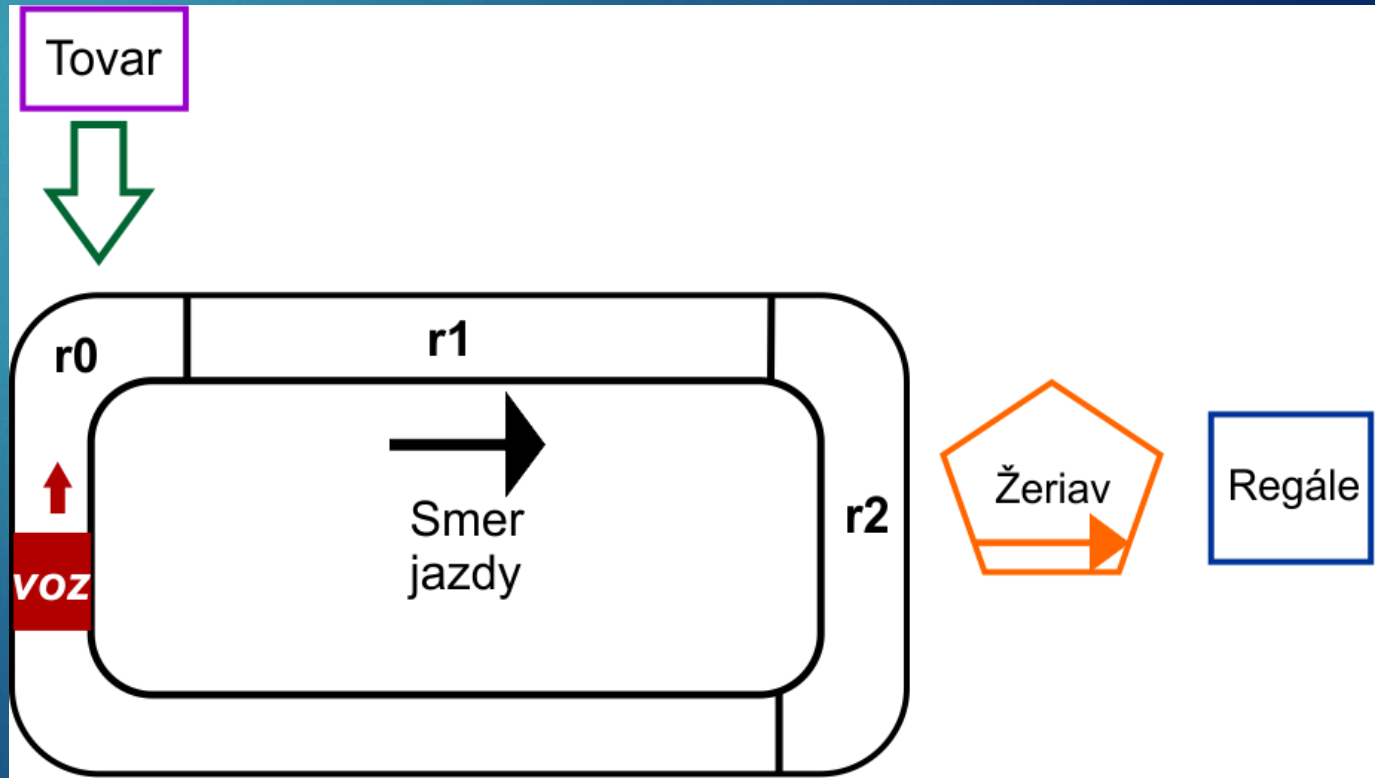
- Q je konečná a neprázdna **množina stavov**.
- $\sum$ je konečná **množina vstupných symbolov** (*abeceda*). Túto množinu môžeme tiež interpretovať ako množinu udalostí.
- σ je **prechodová funkcia** (túto je možné zapísať pomocou tzv. prechodovej tabuľky), ktorá popisuje pravidlá prechodov medzi stavmi. Má podobu $Q \times \sum \rightarrow Q$ (t.j. deterministický automat) alebo $Q \times \{ \sum \cup \varepsilon \} \rightarrow P(Q)$ (t.j. nedeterministický automat), kde $P(Q)$ je potenčná množina, t.j. množina všetkých podmnožín množiny Q.
- q_0 je **počiatočný stav**, $q_0 \in Q$
- F je množina **tzv. cieľových stavov,** $F \subseteq S$

Príklad 1 – sklad s 1 druhom tovaru

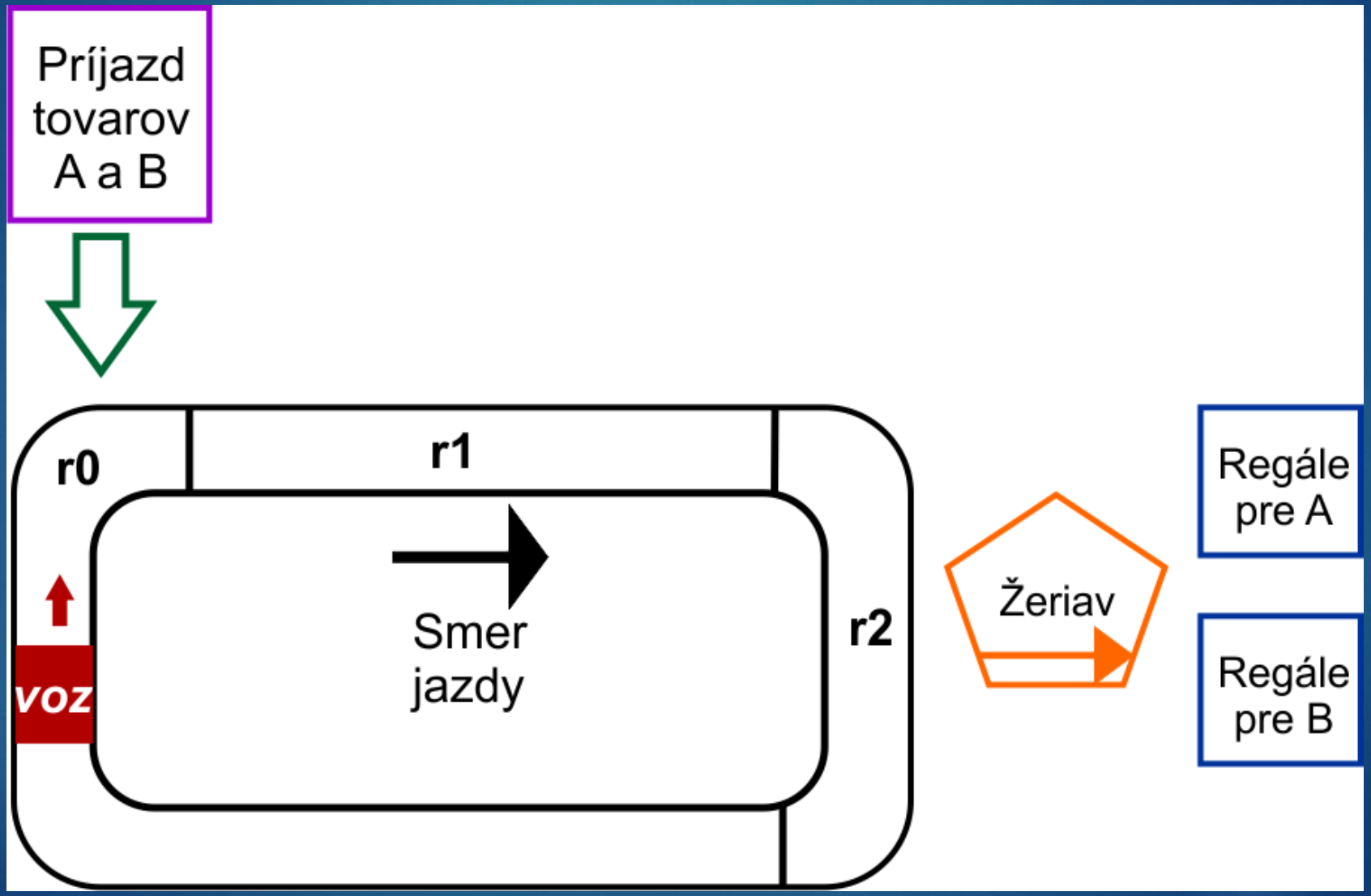


Príklad 1 – sklad s 1 druhom tovaru

- Do skladového systému prichádza jeden druh tovaru, ktorý je vykladaný pomocou pásového dopravníka do koľajového vozidla („voz“). Koľajové vozidlo sa následne presúva cez zónu r1 do zóny r2, kde zastaví. Tu je aktivovaný žeriav, ktorý prenesie tovar z vozidla do regálov. Po prenesení tovaru ide vozidlo naspäť do zóny r0 po ďalší tovar. Toto sa cyklicky opakuje.

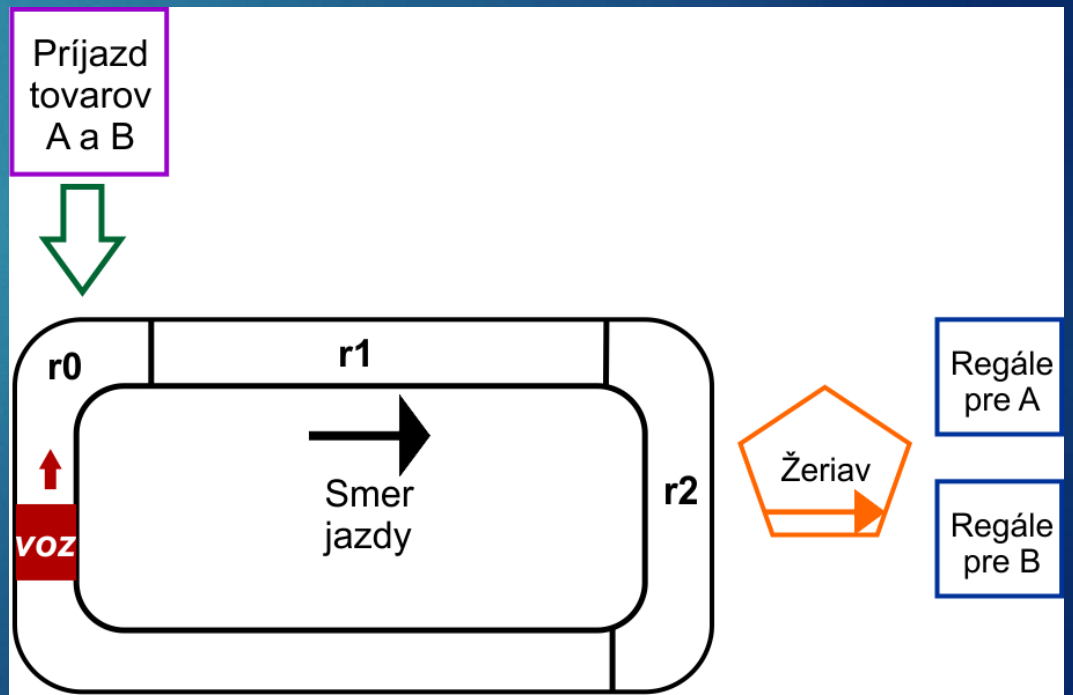


Príklad 1 – sklad s 2 druhmi tovaru



Príklad 1 – sklad s 2 druhmi tovaru

- Do skladového systému prichádzajú tovary A a B. Sled príchodu tovarov nie je vopred známy. Tovar, ktorý príde, je vykladaný pomocou pásového dopravníka do koľajového vozidla („voz“). Koľajové vozidlo sa následne presúva cez zónu r1 do zóny r2, kde zastaví. Tu je aktivovaný žeriav, ktorý prenesie tovar z vozidla do regálov. Ak ide o tovar A, tak bude vyložený do regálu A. Ak ide o tovar B, tak bude vyložený do regálu B. Po prenesení tovaru ide vozidlo naspäť do zóny r0 po ďalší tovar (A alebo B). Toto sa cyklicky opakuje.



Petriho siete

ERIK KUČERA

ANALÝZA A INFORMATIZÁCIA DYNAMICKÝCH SYSTÉMOV

| PREDNÁŠKA-CVIČENIE 2

Automat vs. Petriho sieť

Konečné automaty => množina stavov + prechodová funkcia

Prechodová funkcia priradí aktuálnemu stavu nový stav.

Aktuálny stav = okamžitý aktívny stav

Nový stav = nasledujúci stav - závisí od udalosti, ktorá sa v systéme vyskytla

Automat vs. Petriho sieť

Kde je teda problém?

V konečných automatoch existuje v jednom okamihu len jeden aktívny stav.

Dekompozícia systému na subsystém => žiada sa opísať aktivity subsystémov a ich vzťahy.

AUTOMAT SA STÁVA ŤAŽKOPÁDNYM
KAŽDÁ KOMBINÁCIA STAVOV POTREBUJE VLASTNÝ STAV V AUTOMATE

Automat vs. Petriho sieť

Nevýhody odstraňuje Petriho sieť

Hlavnou myšlienkou Petriho sietí je reprezentovať stavy subsystémov separátne => efektívne modelovanie distribuovaných aktivít systému.

Mnohé vlastnosti DUDS (synchronizácia, súbežnosť, ohraničenosť...) môžu byť veľmi dobre reprezentované a analyzované pomocou Petriho sietí.

Používajú sa nielen na modelovanie správania DUDS, ale aj pri návrhu ich riadenia.

Petriho sieť (Hrúz)

PN = (P, T, F, W, M_0)

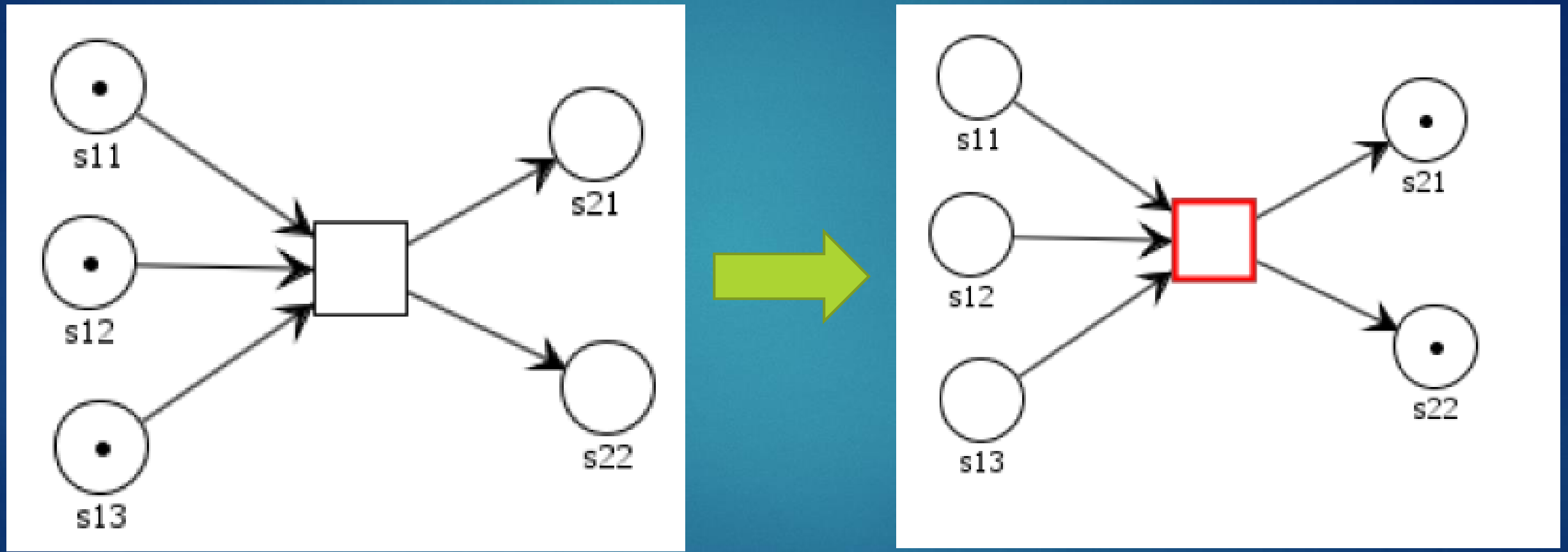
P = { p_1 , p_2, ... , p_n } - konečná neprázdna množina miest

T = { t_1 , t_2, ... , t_n } - konečná neprázdna množina prechodov

F sa nazýva toková relácia ($F = F_1 \cup F_2$), kde $F_1 \subseteq P \times T$ a $F_2 \subseteq T \times P$.

W je váhová funkcia $W: F \rightarrow I^+$, kde I^+ je množina kladných čísel.

M_0 - funkcia počiatočného označkovania $M_0 : P \rightarrow I^+ \cup \{0\}$


s11
s12
s13
s21
s22
s11
s12
s13
s21
s22

Farebné Petriho siete

ERIK KUČERA

ANALÝZA A INFORMATIZÁCIA DYNAMICKÝCH SYSTÉMOV

| PREDNÁŠKA-CVIČENIE 2

Farebné Petriho siete

Rozšírenie PN

- Petriho siete vyššej úrovne - rozšírenie klasických PN o možnosť popisu časových vzťahov alebo dátových typov
- Farebné Petriho siete (CPN) - značky v CPN majú svoju individualitu, údaj určitého typu - farbu
- Farba reprezentuje priradenie určitej hodnoty najrôznejších dátových typov
- Miesta, prechody a hrany - logické podmienky týkajúce sa farieb jednotlivých značiek

Farebné Petriho siete

Dôvody použitia

- CPN majú grafickú reprezentáciu
- Formalizmus CPN poskytuje aj hierarchický opis systému
- CPN ponúkajú interaktívnu simuláciu systému
- K CPN sú dostupné softvérové nástroje na ich konštrukciu, analýzu a simuláciu

Výpočtové prostredie MATLAB

ERIK KUČERA

ANALÝZA A INFORMATIZÁCIA DYNAMICKÝCH SYSTÉMOV | PREDNÁŠKA 2

Charakteristika MATLAB-u I

- Integrované výpočtové prostredie pre vedecko-technické výpočty
- Výkonné výpočtové jadro, ktoré je možné používať v príkazovom („ručnom“) režime alebo v programovom („automatickom“) režime
- Stovky hotových funkcií a nástrojov použiteľných pri tvorbe rôznych aplikácií a výpočtov (toolboxy)
- Univerzálny a jednoduchý programovací jazyk
- Modelovací a simulačný nástroj (Simulink)
- Nástroje na tvorbu finálnych, priamo spustiteľných aplikácií

Charakteristika MATLAB-u II

- Výkonné grafické nástroje pre zobrazovanie výsledkov v 2D a 3D priestore v statickom aj dynamickom (animovanom) režime
- Nástroje na tvorbu grafického rozhrania s používateľom
- Matlab dnes predstavuje celosvetový štandard pre tvorbu vedecko-technických výpočtových, modelovacích a simulačných aplikácií
- Použitie Matlabu môže významne zjednodušiť a urýchliť vývojové a programovacie práce

Aplikačné domény Matlabu (toolboxy)

- Matematika
- Algoritmy a programovanie
- Kybernetika, riadenie
- Spracovanie signálov
- Spracovanie obrazu
- Elektrotechnika
- Umelá inteligencia
- Mechanika, robotika
- Letectvo, kozmonautika
- Ekonómia a financie
- Virtuálna realita
- ...
- Používateľom vytvorené nástroje pre ľubovoľné aplikačné domény

Úvodný kurz Matlabu

- Prostredie - hlavné okno
- Údaje (dáta)
- Operácie s dátami
- Funkcie
- Programovanie
- Grafika
- Simulink
- Príklady

1.1 Prostredie

- Command Window – zadávanie príkazov, spúšťanie programov
- Workspace – zoznam premenných v pracovnom priestore (v pamäti)
- Current Directory – aktuálny adresár
- Command History – minulé príkazy
- Menu
- Help
- Tlačidlo Start
- Ďalšie typy okien – obrázky, animácie ...

Príkazový režim práce

>> príkaz1 + Enter

>> príkaz2 + Enter

>> ...

Programový režim práce

>> program + Enter

Command Window:

```
>> r=2.7

r =

    2.7000

>> v=8

v =

     8

>> objem_valca=pi*r^2*v

objem_valca =

  183.2177
```

Niektoré dôležité príkazy

clc – vymazanie okna

clear x – vymazanie premennej x

clear all – vymazanie všetkých premenných

who – zoznam premenných

whos – zoznam a veľkosti prem.

lookfor *heslo* – vyhľadá príkazy, v ktorých sa vyskytuje *heslo*

cd – zmena adresára

pwd – zobrazenie aktuálneho adresára

dir – obsah aktuálneho adresára

which *príklaz* – vyhľadá adresár, v ktorom sa nachádza *príkaz*

↑ - návrat k predchádzajúcim príkazom

Realizácia príkazu (programu)

>> abc

1. ak *abc* je premenná, zobrazí sa jej obsah
2. ak *abc* je názov príkazu/programu (m-file) v aktuálnom adresári, spustí ho
3. inak hľadá *abc* v inom adresári, kde má nastavené cesty a spustí ho

1.2 Dátové štruktúry

- **matice** (polia) - základný objekt v ML
  n-rozmerné polia, 2-rozmerné polia
- **vektory** - 1-rozmerné polia
- **skaláry** - 1-prvkové vektory
- **súbory** - dáta prenesené z prac. priestoru na iné médium

Zadanie skalárnej premennej →

Ak sa za príkazom napíše ;
výsledok sa nevypisuje →

Zadanie riadkového vektora →

```
>> skalar=1.78

skalar =

    1.7800

>> skalar=1.78;
>>
>> vektor=[1 2 3]

vektor =

     1     2     3

>> vektor=[1,2,3]

vektor =

     1     2     3

>>
```

Zadanie stĺpcového vektora →

```
>> vektor=[1;2;3]

vektor =

     1
     2
     3
```

alebo pomocou transpozície (x') →

```
>> vektor=[1,2,3]'

vektor =

     1
     2
     3

>> vektor'

ans =

     1     2     3

>>
```

Zadanie matice (2D-poľa)

alebo pomocou Enter na konci riadku

```
>> matica=[1 2 3;4 5 6;7 8 9]

matica =

     1     2     3
     4     5     6
     7     8     9

>> matica2=[1 2 3 4
5 6 7 8
9 10 11 12]

matica2 =

     1     2     3     4
     5     6     7     8
     9    10    11    12

>>
```

Inicializácia polí

```
>> a=zeros(3)

a =

     0     0     0
     0     0     0
     0     0     0

>> b=zeros(2,3)

b =

     0     0     0
     0     0     0

>> c=ones(2)*7

c =

     7     7
     7     7
```

```
>> d=rand(2,4)

d =

    0.6822    0.5417    0.6979    0.8600
    0.3028    0.1509    0.3784    0.8537

>> e=[]

e =

     []

>> f=2:1:7

f =

     2     3     4     5     6     7
```

```
>> a=[1 2 3 4;5 6 7 8;9 10 11 12]

a =

     1     2     3     4
     5     6     7     8
     9    10    11    12

>> a(2,3)

ans =

     7

>> a(1,:)

ans =

     1     2     3     4

>> a(:,1)

ans =

     1
     5
     9
```

Adresácia prvkov polí

Presnosť → help format

Veľkosť polí

size(pole);

length(vektor);

Viacrozmerné polia

```
>> P(3,2,2)=7

P(:,:,1) =

     0     0
     0     0
     0     0


P(:,:,2) =

     0     0
     0     0
     0     7

>> size(P)

ans =

     3     2     2
```

Zmena prvkov poľa

```
a =

     1     1     1     1
     1     1     1     1
     1     1     1     1
     1     1     1     1

>> a(3,2)=8

a =

     1     1     1     1
     1     1     1     1
     1     8     1     1
     1     1     1     1
```

```
a =

     1     1     1     1
     1     1     1     1
     1     1     1     1
     1     1     1     1

>> a(3,2:4)=8

a =

     1     1     1     1
     1     1     1     1
     1     8     8     8
     1     1     1     1
```

Spájanie polí

```
a =

     1     1
     1     1

>> b

b =

     2     2     2
     3     3     3

>> a=[a b]

a =

     1     1     2     2     2
     1     1     3     3     3
```

```
p =

     1     2     3

>> q

q =

     7     7     7

>> r=[p q]

r =

     1     2     3     7     7     7

>> s=[p;q]

s =

     1     2     3
     7     7     7
```

Špeciálne konštanty

pi = 3.1416

i, j - imaginárna časť komplexného čísla

Inf - nekonečno (napr. po delení nulou)

NaN - Not a Number (0/0 ...)

clock - čas

date - dátum

eps - najmenšie možné číslo

ans - výsledok operácie

Ukladanie dát do súborov

save *meno* x y A - uloží premenné x,y,A do súboru meno.mat

save *meno.qqq* x -ascii - uloží premennú x do súboru meno.qqq vo formáte ascii

save *meno* - uloží všetky premenné pracovného priestoru do súboru *meno.mat*

load *meno* - načíta všetky premenné zo súboru meno

1.3 Operácie s dátami

- základné matematické operácie
- maticové operácie
- knižnice funkcií
- používateľom vytvorené funkcie

Základné matematické operácie

premenná = výraz;

Výrazy obsahujú konštanty, premenné a operácie:

a+b, a-b, a*b, a/b, a^b, sqrt(.), ...

Príklad:

x=2^(5/(8+2)) → $x=2^{(5/10)}$

Maticové operácie

- súčet matíc: **A+B**
- súčin matíc: **A*B**
- násobenie matice konštantou: **A*const**
- pričítanie konštanty k matici: **A+const**
- inverzia matice: **inv(A)**
- násobenie matice maticou "po prvkoch": **A.*B**

...

Funkcie

y=funkcia(x);

- **základné matematické funkcie:**

  sin(x), cos(x), ... , exp(x), abs(x), ...

- **iné funkcie:**

  knižnice (toolbox-y) so stovkami funkcií → viď Help

- **používateľsky vytvorené funkcie**

Knižnice funkcií

- **Datafun** - analýza dát (help Datafun)
- **Polyfun** - interpolácia a polynómy
- **Graph2D, Graph3D** - grafy
- ...
- >> help
- >> demo

Príklad použitia funkcie max(x)

```
a =

    4.1865    8.3812    5.0281    1.9343    6.9790    4.9655    6.6023
    8.4622    0.1964    7.0947    6.8222    3.7837    8.9977    3.4197
    5.2515    6.8128    4.2889    3.0276    8.6001    8.2163    2.8973
    2.0265    3.7948    3.0462    5.4167    8.5366    6.4491    3.4119
    6.7214    8.3180    1.8965    1.5087    5.9356    8.1797    5.3408

>> max(a)

ans =

    8.4622    8.3812    7.0947    6.8222    8.6001    8.9977    6.6023

>> max(max(a))

ans =

    8.9977
```

Polynomiálna regresia - preloženie polynómu

napr. cez 7 bodov v rovine pomocou polynómu 4.stupňa v tvare

$$P(x)=a_4.x^4+a_3.x^3+a_2.x^2+a_1.x+a_0$$

Výsledkom sú koeficienty polynómu.

```
x =

     1     2     3     4     5     6     7

>> y

y =

     2     1     5     7     0     5     9

>> polyfit(x,y,4)

ans =

    0.1439   -2.0808    9.9167  -17.0014   10.7143
```

Výpočet koreňov polynómu

$y=10x^5+27x^4+12x^3+8x^2+2x+39$

```
>> korene=roots([10 27 12 8 2 39])

korene =

  -2.4389
  -0.7832 + 1.0044i
  -0.7832 - 1.0044i
   0.6527 + 0.7481i
   0.6527 - 0.7481i
```

Riešenie sústavy lineárnych rovníc

A – matica koeficientov lin. rovníc

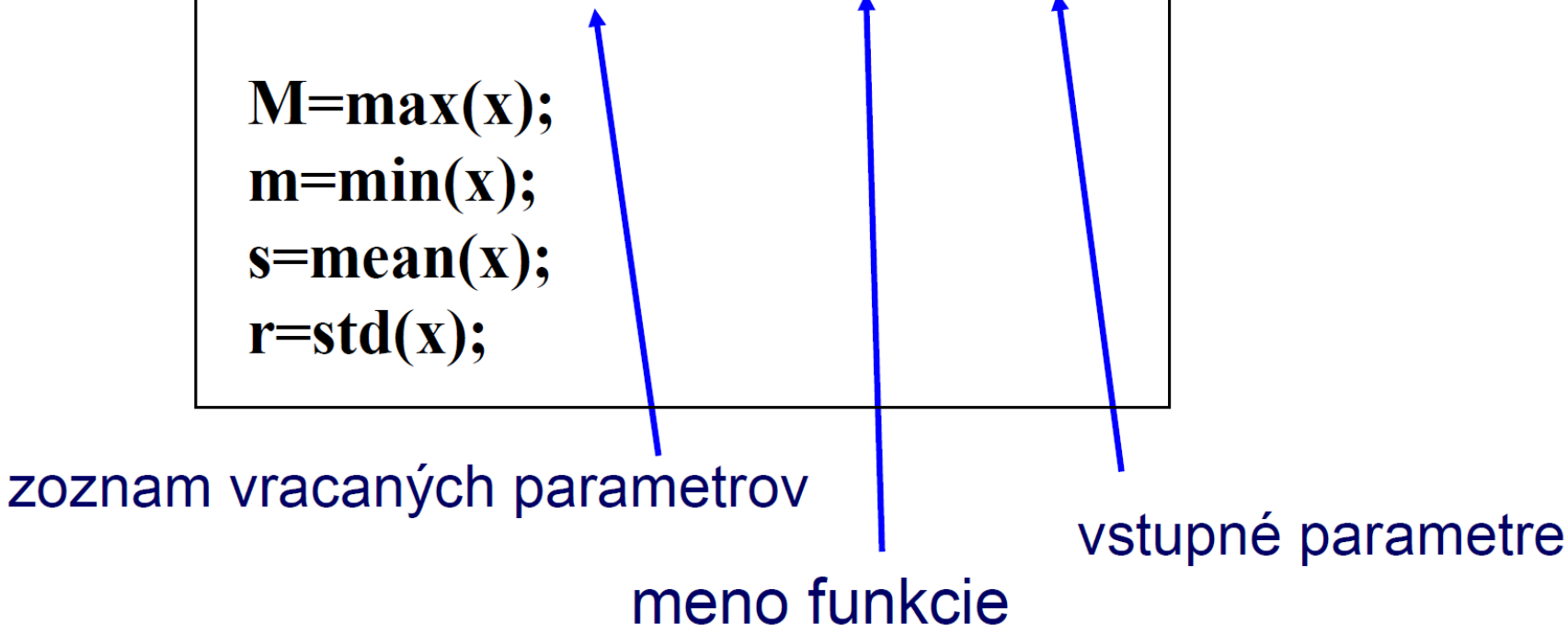
b – vektor pravých strán

```
A =

     2     4     6     1
     1     0     3     7
     2    -1    -2     8
     2     2     4     1

>> b

b =

    10
    15
     9
     2

>> x=inv(A)*b

x =

   -4.4478
    3.9254
    0.0746
    2.7463
```

Používateľom definované funkcie

Príklad:

```
function[M,m,s,r]=statistika(x)

M=max(x);
m=min(x);
s=mean(x);
r=std(x);
```




```
x = [2.3 0.01 7 -9.21 10 25.4 1.0 2.77];
[max, min, priem, rozpt] = statistika(x);
```

1.4 Grafy

- 2D grafy (→ help graph2D)
- 3D grafy (→ help graph3D)

2D grafy (obrázky)

plot(x,y) – kreslenie grafu

axis([x_d,x_h,y_d,y_h]) - zmena mierky osí

grid – nakreslenie rastru v grafe

hold – zafixovanie (odfixovanie) grafu pre viacnásobné kreslenie grafov (hold on, hold off)

xlabel('text') – označenie osi x

ylabel('text') – označenie osi y

title('text') – nadpis obrázku

gtext('text') – umiestnenie textu do obrázku

figure – nový obrázok

figure(n) – otvorenie/adresovanie obrázku číslo n

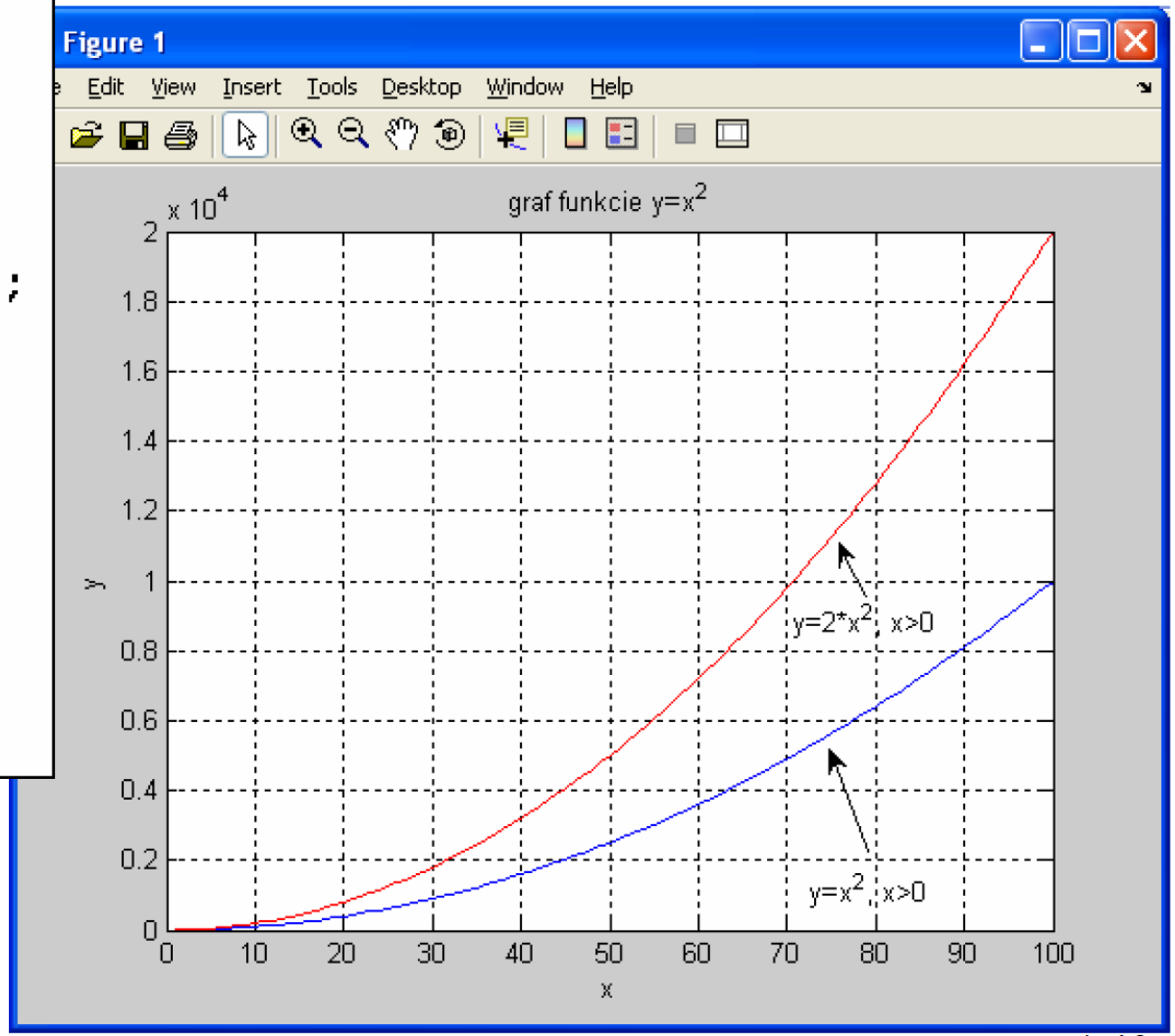
clf – vymazanie obrázku

subplot(abc) – rozdelí okno na maticu a x b podokien, adresuje podokno s poradovým číslom c

close(n)/close all – zavretie obrázku č. n / všetkých obr.

Príklad 2D grafu

```
>> x=1:1:100;
>> y=x.^2;
>> plot(x,y)
>> xlabel('x');
>> ylabel('y');
>> title('graf funkcie y=x^2');
>> grid
>> gtext('y=x^2, x>0')
>> hold
Current plot held
>> y2=2*x.^2;
>> plot(x,y2,'r')
```



plot (→ help plot)

plot(y) – vykreslenie vzoriek vektora y do grafu
plot(x,y) – vykreslenie závislosti y=f(x)
plot(x,y,'color') – definovanie farby grafu

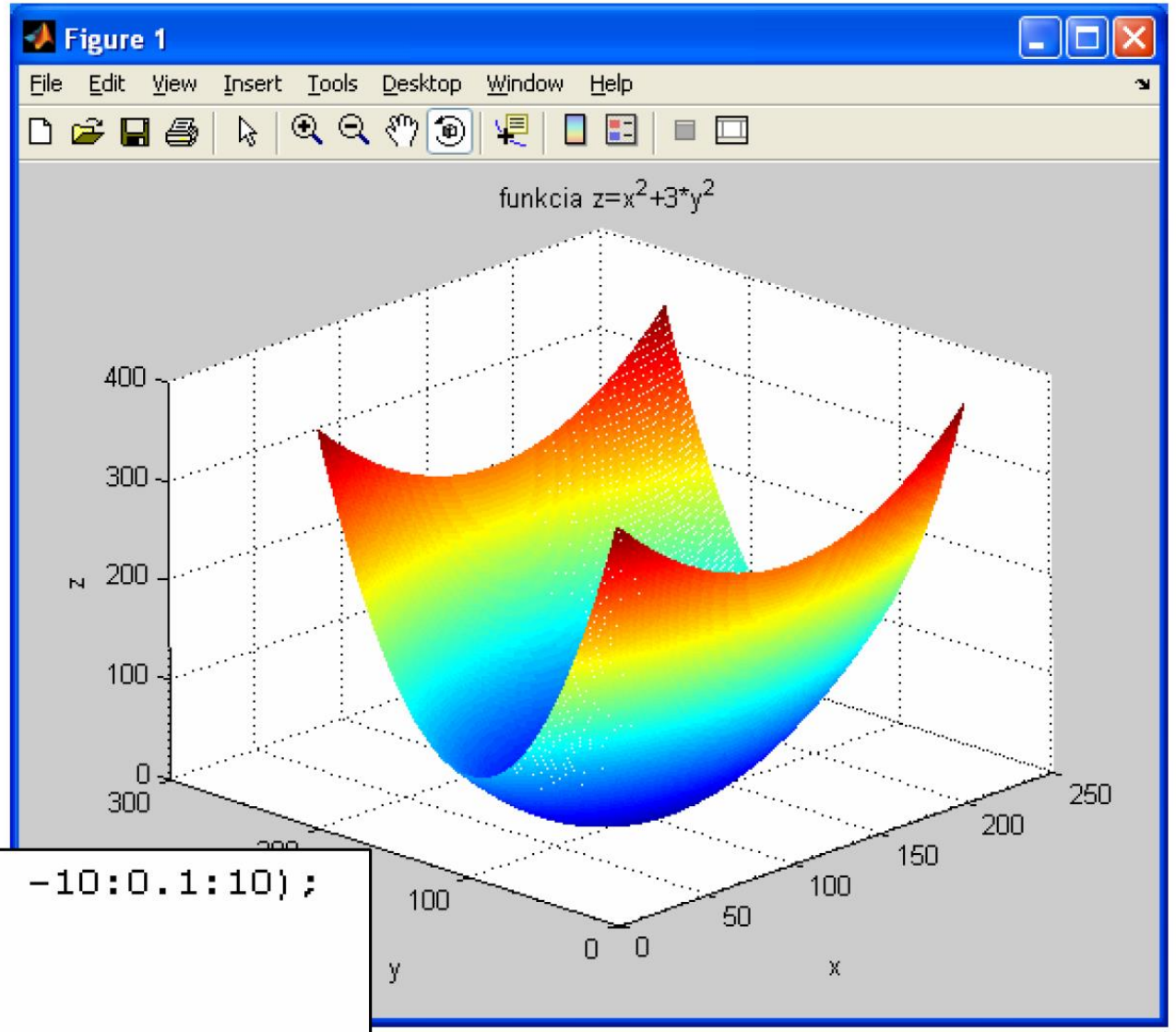
color: b-modrá (blue)
r-červená (red)
g-zelená (green)
y-žltá (yellow)
m-fialová (magentha)
c-bledomodrá (cyan)
k-čierna (black)

plot(x,y,'co') – definovanie farby grafu a symbolu
o-symbol (x,o,*,+, . ,d, …), c-farba, x,y - vektory

plot(x,y,'co') – vykreslenie bodu v rovine
o-symbol (x,o,*,+, .), c-farba, x,y – súradnice bodu

3D grafy

→ help graph3D

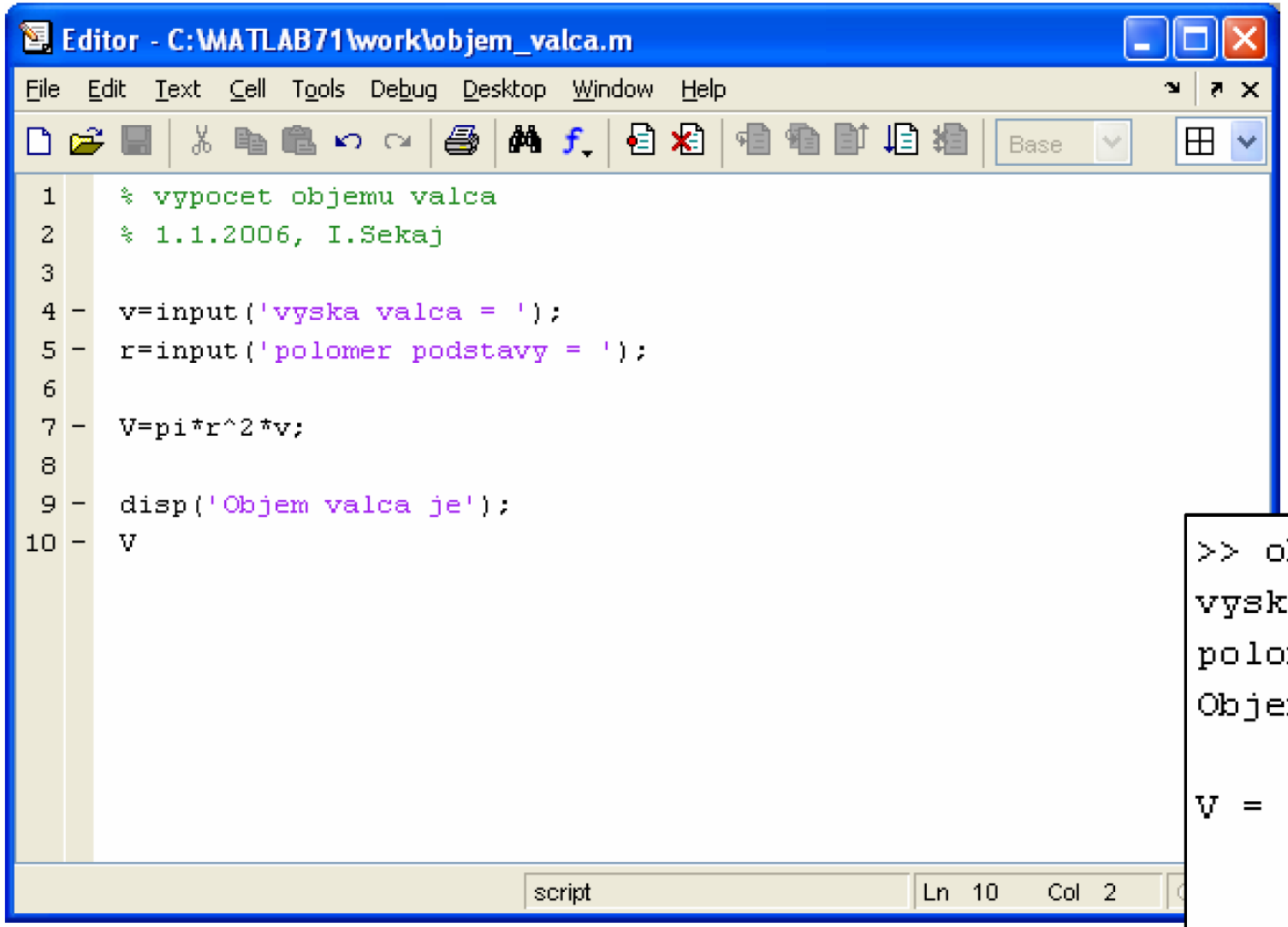



```
>> [x,y]=meshgrid(-10:0.1:10, -10:0.1:10);
>> z=x.^2+3*y.^2;
>> mesh(z)
>> xlabel('x')
>> ylabel('y')
>> zlabel('z')
>> title('funkcia z=x^2+3*y^2')
```

1.5 Programovanie
tvorba m-súborov (m-files)

- Tvorba používateľských programov,
- postupnosť príkazov ako v riadkovom režime,
- môžu byť použité príkazy, funkcie aj iné m-súbory,
- tvorba v integrovanom textovom editore (file/new/M-file)
- alebo v ľubovoľnom inom textovom editore, uložiť ako textový súbor – *meno.m*,
- spustiť z riadkového režimu: *>>meno*
  alebo z Matlab editora → *run*

Príklad programu




```
>> objem_valca
vyska valca = 2.5
polomer podstavy = 0.7
Objem valca je

V =

    3.8485

>>
```

Konštrukcie programovacieho jazyka Matlabu

- for, end
- if, else, elseif, end
- while, end
- break
- continue
- witch, case
- a iné ...

>> help lang

```
>> help lang
  Programming language constructs.

  Control flow.
    if              - Conditionally execute statements.
    else            - Execute statement if previous IF condition failed.
    elseif          - Execute if previous IF failed and condition is true.
    end             - Terminate scope of control statements.
    for             - Repeat statements a specific number of times.
    while           - Repeat statements an indefinite number of times.
    break           - Terminate execution of WHILE or FOR loop.
    continue        - Pass control to the next iteration of a loop.
    switch          - Switch among several cases based on expression.
    case            - SWITCH statement case.
    otherwise       - Default SWITCH statement case.
    try             - Begin TRY block.
    catch           - Begin CATCH block.
    return          - Return to invoking function.
    error           - Display message and abort function.
    rethrow         - Reissue error.

  Evaluation and execution.
    eval            - Execute string with MATLAB expression.
    evalc           - Evaluate MATLAB expression with capture.
    feval           - Execute the specified function.
    evalin          - Evaluate expression in workspace.
    builtin         - Execute built-in function from overloaded method.
    assignin        - Assign variable in workspace.
    run             - Run script.
```

Niektoré typy podmienok pri vetvení programu

(priradenie: x=2)

podmienka rovnosti: if x==2 …
podmienka rôznosti: if x~=2 …
podmienka nerovnosti: if x>=2 …

logické AND: if x & y
logické OR: if x | y
negácia: if ~x

Cyklus typu: for, if, elseif, else

```
for i=1:100
    x(i)=(i-1)/10;
end;
plot(x);
```

```
x(1)=0;
for i=2:100
    if i<=25
        x(i)=x(i-1)+0.1;
    elseif i<=50
        x(i)=x(i-1)-0.1;
    elseif i<=75
        x(i)=x(i-1)+0.1;
    else
        x(i)=x(i-1)-0.1;
    end;
end;
plot(x);
```

Cyklus typu: while

```
x=0;
k=1;

figure(1);
clf;          % vymazanie predchadzajuceho obrazku
hold on;      % zafixovanie predchadzajucich plotov

while x<10  % pokial nie je splnena podmienka, bezi cyklus
    x=x+rand;    % pripocitanie nahodneho cisla < 1
    plot(k,x,'*');
    pause(0.2); % pauza 0.2 s
    k=k+1;
end;
```

Break, continue

```
x=0;

figure(1); % inicializacia obr.1
clf;       % vymazanie predchadzajuceho obrazku
hold on;   % zafixovanie predchadzajucich plotov

for k=1:1000
    r=rand;
    if r<0.1 continue; end; % preskoci danu iteraciu
    x=x+r;   % pripocitanie nahodneho cisla < 1
    plot(k,x,'*');  % vykreslenie jedneho znaku *
    pause(0.2); % pauza 0.2 s
    if x>=10 break; end;    % ak je x>10 vyskoc z cyklu for
end;
```

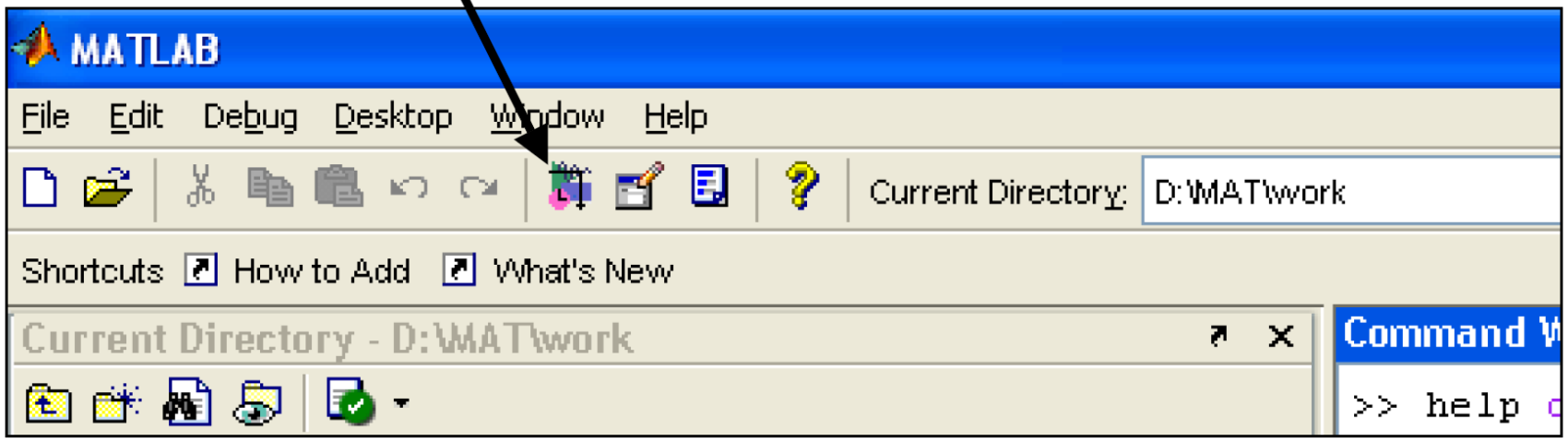
Switch, case, otherwise

```
d=zeros(1,10);
for n=1:100
    r=rand*10;
    z=ceil(r);
    switch(z);
        case 1
            d(1)=d(1)+1;
        case 2
            d(2)=d(2)+1;
        case 3
            d(3)=d(3)+1;
        case 4
            d(4)=d(4)+1;
        case 5
            d(5)=d(5)+1;
        case 6
            d(6)=d(6)+1;
        case 7
            d(7)=d(7)+1;
        case 8
            d(8)=d(8)+1;
        case 9
            d(9)=d(9)+1;
        otherwise
            d(10)=d(10)+1;
    end;
end;
bar([1 2 3 4 5 6 7 8 9 10],d);
d
```

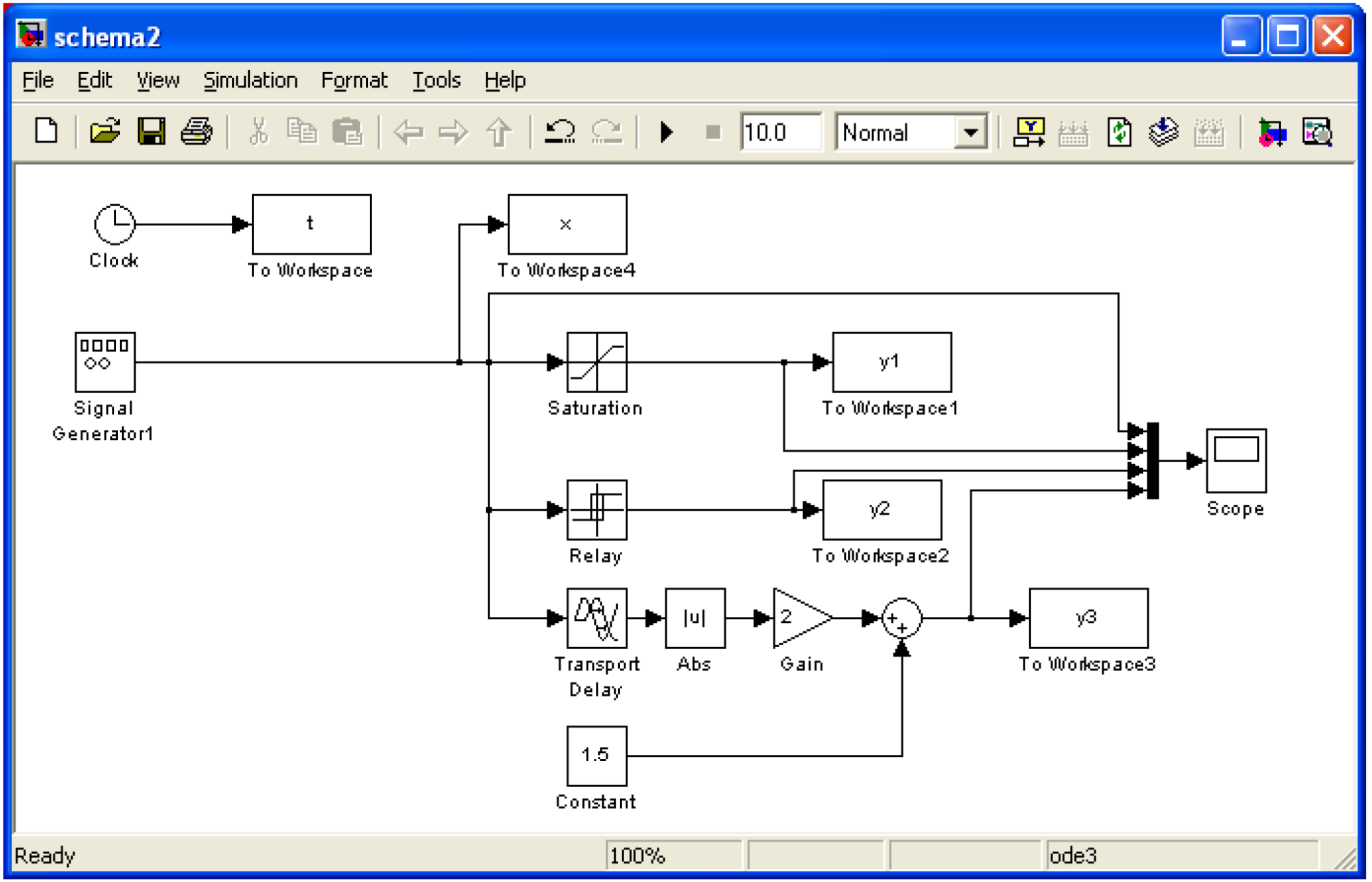
1.6 Simulink

- **Samostatný nástroj Matlabu na tvorbu modelov a realizáciu časových simulácií**
- **Rozsiahla knižnica rôznych typov objektov**
- **Vizualizácia**

- **spustenie: a) >> Simulink**
  b)



Príklad simulačného modelu v Simulinku



1.6 SIMULINK

1.7 Iné nástroje Matlabu

- GUI – grafické používateľské rozhranie
- VR-Toolbox – virtuálna realita
- RT-Toolbox - pripojenie na reálny svet (real-time)
- Matlab Compiler – prekladač z MATLABu do C++
- Matlab Web Server – sieťové aplikácie
- Paralell computing – rozdelenie výpočtu na viac PC
- a iné ...

Zdroje

- Danica Rosinová
- Ivan Sekaj
- Branislav Hrúz