FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY
UNIVERZITY KOMENSKÉHO V BRATISLAVE

Katedra aplikovanej matematiky a štatistiky

Všeobecný algebraický modelovací systém - GAMS

DIPLOMOVÁ PRÁCA

BRATISLAVA 2006 Jakub Drozdík

FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY
UNIVERZITY KOMENSKÉHO V BRATISLAVE

Odbor Matematika,
Špecializácia Ekonomická a finančná matematika

Všeobecný algebraický modelovací systém - GAMS

DIPLOMOVÁ PRÁCA

Jakub Drozdík

Vedúci diplomovej práce: doc. RNDr. Milan Hamala, CSc.

Bratislava 2006

Čestné prehlásenie

Čestne prehlasujem, že diplomovú prácu som vypracoval samostatne len s využitím teoretických vedomostí a s použitím uvedenej literatúry.

V Bratislave, 27. apríla 2006

Ďakujem vedúcemu diplomovej práce doc. RNDr. Milanovi Hamalovi, CSc. za cenné rady a pripomienky, ktoré mi pomohli pri písaní tejto práce.

Abstrakt:

Matematické modelovanie sa uplatňuje v rôznych oblastiach ľudského pôsobenia, napríklad v ekonómii, manažmente, finančnej matematike, inžinierstve, chemickom inžinierstve ... . Všeobecný algebraický modelovací systém - GAMS, ktorý je témou tejto diplomovej práce, umožňuje programovať rôzne typy optimalizačných úloh v ich prirodzenom matematickom zápise a následne ich riešiť pomocou rôznych „solverov“. GAMS si zároveň dokáže poradiť aj s mnohými úlohami, ktoré nemajú optimalizačný charakter (rôzne simulácie). Prvá časť diplomovej práce stručne, no zároveň obsažne, popisuje nástroje jazyka GAMS. Pomocou nich je možné naprogramovať rôzne úlohy, ako to dokumentuje druhá časť diplomovej práce obsahujúca matematické formulácie i formulácie v jazyku GAMS úloh lineárneho programovania, lineárneho regresného modelu, ktorého parametre sú odhadnuté pomocou metódy najmenších štvorcov, DEA modelu, CGE modelu a modelu Markowitzovej teórie portfólia. V prílohe je uvedený program riešiaci komplexný problém relatívne veľkých rozmerov, a to problém kalibrácie modelov na riadenie portfólia (model Markowitzovej teórie portfólia, model s kvadratickou funkciou užitočnosti) spolu so stručným popisom tohto programu.

Obsah

Predhovor

Témou mojej diplomovej práce je všeobecný algebraický modelovací systém GAMS (General Algebraic Modeling System), slúžiaci na modelovanie reálneho sveta. Je to vysoko-úrovňový programovací jazyk, ktorý nám umožňuje používať prirodzenú matematickú formuláciu modelov. Prostredníctvom jazyka GAMS je dostupných približne tridsať programových systémov na riešenie optimalizačných úloh („solverov"), a tým aj väčšina algoritmov v súčasnosti používaných na tieto účely. GAMS nachádza využitie v rôznych oblastiach ľudského pôsobenia. Napríklad v ekonómii (mikroekonómia, makroekonómia, medzinárodný obchod ...), manažmente, finančnej matematike, inžinierstve, chemickom inžinierstve, dokonca aj v lesníctve a v mnohých ďalších oblastiach vyžadujúcich si istý druh optimalizácie. GAMS vie však zvládnuť aj mnohé problémy, ktoré nemajú charakter optimalizačnej úlohy (napr. rôzne simulácie). Z uvedených faktov sme usúdili, že jazyk GAMS je veľmi užitočný nástroj na riešenie reálnych problémov, a preto sme si vytýčili nasledujúce **ciele diplomovej práce**:

1) Vypracovať stručný ale zároveň dostatočne obsažný popis jazyka GAMS

2) Prezentovať možnosti praktického využitia jazyka GAMS prostredníctvom rôznorodých úloh relatívne malých rozmerov

3) Spracovať jeden komplexný problém aplikovateľný v praxi

Naplneniu **prvého cieľa** je venovaná prvá časť diplomovej práce pozostávajúca z 15 kapitol. Kapitola 1 uvádza čitateľa do problematiky. Kapitola 2 obsahuje základné pravidlá písania programov v jazyku GAMS. Kapitoly 3-7 popisujú jednotlivé dátové typy jazyka GAMS. Kapitola 8 sa venuje príkazu na riešenie modelu (príkaz `solve`). Kapitoly 9-12 pojednávajú o zložitejších nástrojoch jazyka GAMS (podmienené príkazy, dynamické množiny, usporiadané množiny a riadiace štruktúry). Kapitola 13 podrobne rozoberá štruktúru výstupného súboru a možnosti upravovania jeho formátu. Kapitola 14 poskytuje informácie o príkaze `display`, pomocou ktorého vieme zobraziť hodnoty zo vstupného súboru[1] do súboru výstupného. Posledná kapitola, kapitola 15 poukazuje na existenciu nástrojov a možností jazyka GAMS, ktorým sme sa v predchádzajúcich kapitolách nevenovali z dôvodu obmedzenia na rozsah diplomovej práce, no schopnosť ovládať ich je užitočná a v niektorých prípadoch aj nevyhnutná. V tejto kapitole zároveň poskytujeme zdroje informácií k jednotlivým témam v podobe odkazov na konkrétne internetové stránky. Štruktúra prvej časti bola z väčšej miery prebraná z anglických príručiek získaných z internetu, no zároveň chceme zdôrazniť, že táto časť nie je doslovným prekladom anglických príručiek do slovenského jazyka. Pri písaní každej kapitoly sme sa totiž snažili použiť vlastný prístup k problematike, vrátane vlastných príkladov (iba tri špecifické príklady sú prebrané z literatúry).

Naplneniu **druhého cieľa** je venovaná druhá časť diplomovej práce pozostávajúca z kapitol 16-21. Kapitola 16 uvádza čitateľa do problematiky. Kapitola 17 sa venuje úlohám lineárneho programovania. Kapitola 18 popisuje lineárny regresný model a metódu najmenších štvorcov. Kapitola 19 približuje jeden z modelov používaných v DEA (Data Envelopment Analysis), a to vstupne orientovaný CCR model s konštantnými výnosmi z rozsahu. Kapitola 20 obsahuje komparatívno-statický CGE model uzavretej ekonomiky. Posledná kapitola, kapitola 21 sa zaoberá modelom Markowitzovej teórie portfólia. Koncepcia druhej časti bola vytvorená selekciou z širšieho okruhu modelov na základe zaujímavosti daného modelu, jeho aktuálnosti a v neposlednom rade, na základe širokospektrálnosti využitia jazyka GAMS v danom modeli. S väčšinou vybraných modelov som sa zoznámil počas štúdia na vysokej škole a zdrojom informácií mi boli prednášky, ktoré uvádzame na konci diplomovej práce v časti Použitá literatúra. CGE model som si z väčšej časti naštudoval sám z literatúry uvedenej v tej istej časti. Ešte by som chcel zdôrazniť, že programy napísané v jazyku GAMS k jednotlivým modelom som tvoril samostatne a ani k jednému z modelov som nemal k dispozícii hotový program.

[1] vstupným súborom sa v tomto prípade myslí súbor obsahujúci samotný program, nielen vstupné dáta

Naplneniu **tretieho cieľa** je venovaná Príloha 1. Vybrali sme si komplexný problém analyzovaný v inej diplomovej práci [6], patriacej do oblasti finančnej matematiky. V tejto práci sa testuje vplyv rôznych metód výpočtu očakávaného výnosu na vývoj reálnej hodnoty portfólia počas daného časového obdobia, pričom toto portfólio je najskôr zostavené pomocou modelu Markowitzovej teórie portfólia a potom pomocou modelu s kvadratickou funkciou užitočnosti na základe historických dát. V rámci každého z týchto dvoch modelov sú skúmané dve stratégie, a to stratégia buy & hold a stratégia pravidelného prehodnocovania portfólia. Naším cieľom bolo vytvoriť program v jazyku GAMS, pomocou ktorého by sa uvedené analýzy dali realizovať, pričom by umožňoval menenie parametrov takýchto analýz, napríklad historického obdobia, z ktorého vždy počítame očakávaný výnos, dňa začiatku investovania, periódy prehodnocovania portfólia, dĺžky obdobia, počas ktorého budeme ivestovať, ... Súčasne by v ňom boli naprogramované rôzne metódy počítania očakávaného výnosu s možnosťou jednoduchej voľby jednej z nich.

Poznámka:

Záujemcom o jazyk GAMS, ktorí s nim prichádzajú do kontaktu po prvý raz, odporúčame prečítať si najskôr podkapitolu 17.2, kde je detailne popísaný zápis jednoduchého modelu lineárneho programovania v jazyku. Podľa môjho názoru, aj neúplné pochopenie spomínanej podkapitoly, značne uľahčí čítanie prvej časti diplomovej práce.

Prvá časť:
Popis jazyka GAMS

1 Úvod do prvej časti

Čo je to GAMS a jeho základné vlastnosti:

Všeobecný algebraický modelovací systém (General Algebraic Modeling System - GAMS) je vysoko-úrovňový programovací jazyk, ktorý sa používa na modelovanie reálneho sveta. GAMS namodelované problémy nerieši, ale ich iba posúva kompatibilnému samostatnému programu na riešenie optimalizačných úloh (solveru), a teda slúži ako interface medzi užívateľom a približne tridsiatimi solvermi. Bol navrhnutý s cieľom spraviť modelovanie jednoduchším a prehľadnejším. Umožňuje nám používať prirodzenú matematickú formuláciu modelov, kým solvre si vyžadujú špeciálnu vstupnú formu, ktorá je navyše pre väčšinu z nich rozdielna. Úpravu do takéhoto tvaru zabezpečí GAMS a my sa môžeme vďaka tomu sústrediť na matematickú formuláciu modelu a nezaťažovať sa technickými detailmi. Vďaka algebraickej štruktúre modelu, a teda vďaka jeho prehľadnosti, môžeme v ňom robiť zmeny jednoducho a bezpečne. Ďalšou veľkou prednosťou jazyka GAMS je nezávislosť formulácie modelu od samotných dát. Z toho vyplýva, že problém školských rozmerov má identický tvar ako ten istý problém mnohonásobne väčších rozmerov. Navyše, ak potrebujeme konkrétny model opakovane riešiť, tak ho môžeme naformulovať vo všeobecnom tvare a až následne budeme zadávať aktuálne dáta. Keďže model a dáta sú v tomto prípade nielen logicky, ale aj fyzicky separované (sú napísané oddelene), môžeme dáta aktualizovať bez akéhokoľvek zásahu do správne naformulovaného modelu, a tým je vylúčené jeho prípadné narušenie. Čo sa týka zadávania dát, GAMS je navrhnutý tak, aby sme ich mohli zadať v čo najelementárnejšom tvare. To znamená, že všetky úpravy môžu byť robené v rámci modelu a navyše sa budú vyskytovať aj vo výstupe, aby sme mohli overiť ich správnosť. Týmto spôsobom sa vyhneme možným chybám v dôsledku obdržania zle upravených dát. Model naprogramovaný v jazyku GAMS nepotrebuje žiadnu sprievodnú dokumentáciu, pretože je „samo-vysvetľujúci“ (self-documenting). GAMS totiž umožňuje deklarovať až 31 znakové identifikátory a za každým z nich môže ešte nasledovať až 255 znakový vysvetľujúci text. Ten sprevádza príslušný identifikátor aj pri každom jeho výskyte vo výstupnom súbore. Navyše v GAMS-e dokážeme vložiť komentár na ktorékoľvek miesto v modeli a akokoľvek dlhý. Ďalšou prednosťou jazyka GAMS je, že pomocou neho vieme naformulovať veľmi široké spektrum typov modelov a mali by sme mať k dispozícii veľké množstvo v súčasnosti používaných algoritmov na ich riešenie. Z doteraz uvedeného vyplýva, že GAMS je veľmi vhodný na modelovanie rôznych reálnych problémov, a preto sa budeme v nasledujúcich kapitolách venovať popisu jeho jazyka.

Používaná syntax:

V syntaxe používame značenie „Backus-Naur form“ (BNF) kde:

[ ] značí, že prítomnosť obsahu uzavretého v týchto zátvorkách nie je povinná (je voliteľná)
{} značí, že takto uzavretá konštrukcia sa môže opakovať nula krát alebo viac krát
| vyjadruje logický operátor *alebo*, teda buď je prítomná konštrukcia na ľavo od | alebo tá na pravo

Dôležité pojmy:

zdrojový kód - napísaný súbor príkazov, ktorý sa môže následne skompilovať
vstupný súbor - súbor obsahujúci zdrojový kód
výstupný súbor - súbor obsahujúci výstup skompilovaného programu
identifikátor - symbol prislúchajúci k niektorému dátovému typu (meno parametra, premennej ...)
deklarácia - prvé zmienenie identifikátora v zdrojovom kóde umožňujúce jeho ďalšie používanie
definícia - identifikátor nadobúda konkrétne hodnoty
inicializácia - identifikátor nadobúda konkrétne hodnoty počas deklarácie
množina - množina indexov, prostredníctvom ktorých môžu byť definované identifikátory
indexová množina - matematický ekvivalent pojmu množina z jazyka GAMS
prvky - prvky množín, zodpovedajú indexom z matematickej formulácie modelu
kombinácia prvkov - spojenie *n* prvkov z *n* rôzne pomenovaných množín (napr.: $_{1\,2}$ v matici)
obor - zoznam množín na ktoré je identifikátor viazaný
overovanie oboru - overovanie príslušnosti použitých prvkov k množine z oboru identifikátora

Konvencie značenia:

Bežný matematický zápis prechádzania indexu cez indexovú množinu (i = 1 , 2 , ... , m) nahrádzame v tejto diplomovej práci matematickým zápisom lepšie korešpondujúcim s formuláciou modelu v jazyku GAMS ($i \in I$, pričom množina I by bola už skôr zadefinovaná ako $I = \{1, \dots m\}$). Napríklad nasledujúci blok ohraničení zapísaný v bežnom značení:

$$\sum_{j=1}^{n} a_{ij} x_j = b_i \qquad \text{, kde} \quad i = 1, 2, \dots, m$$

by sme v našom značení zapísali takto:

$$\sum_{j \in J} a_{ij} x_j = b_i \qquad \text{, kde} \quad i \in I$$

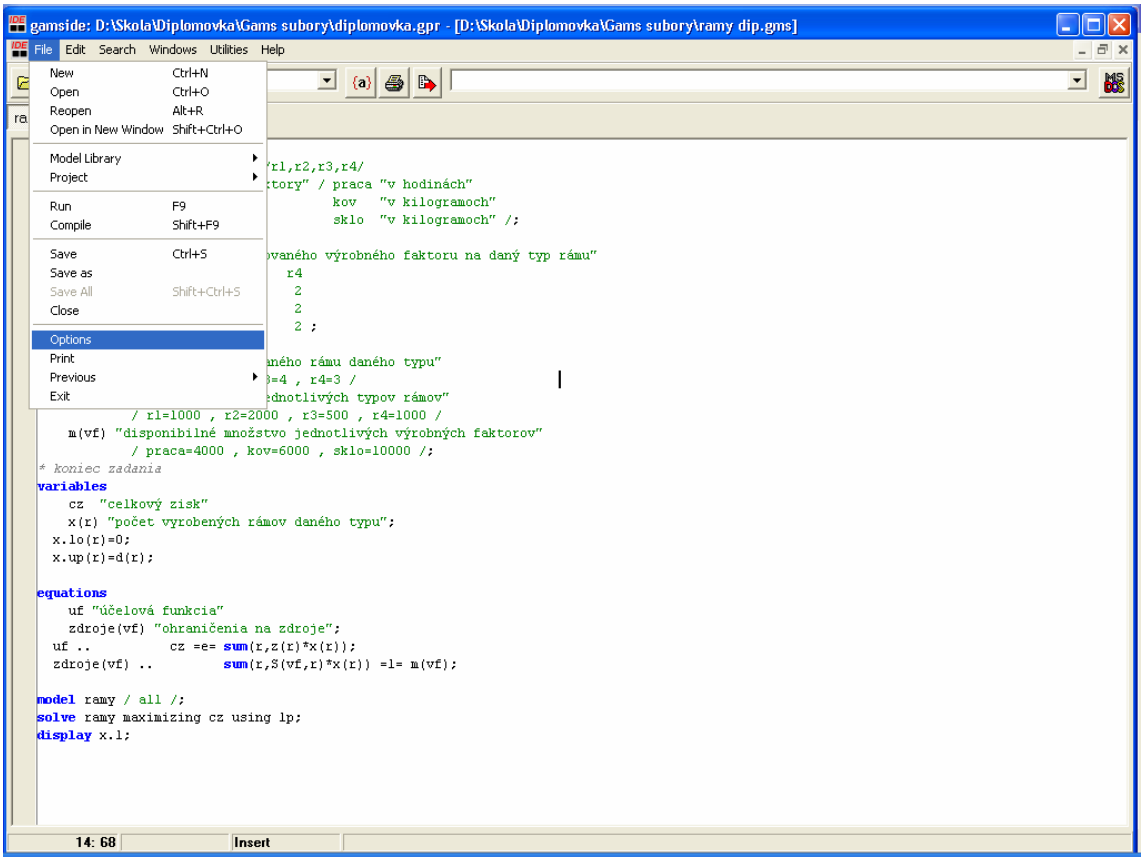
pričom množiny I a J by boli už skôr zadefinované ako $I = \{ 1, \dots, m \}$ a $J = \{ 1, \dots, n \}$ a zápis $i \in I$ je skrátenou verziou bežného matematického zápisu $\forall i \in I$.

Integrované vývojové prostredie (IDE) GAMS-u pre operačný systém Windows :

IDE (Integrated Development Environment) je v princípe rovnaké prostredie ako majú ostatné aplikácie pracujúce pod operačným systémom Windows. Voľne dostupnú demo verziu, postačujúcu na všetko, čo je uvedené v tejto práci, si môžete stiahnuť na internetovej adrese:

http://www.gams.com/download

V žiadnej z nasledujúcich kapitol sa už téme IDE nebudeme venovať, kvôli obmedzeniam na rozsah Diplomovej práce. Na vytvorenie predstavy o tom, ako toto prostredie vyzerá, uvádzame jeho ukážku:



2 Programy napísané v jazyku GAMS

Program napísaný v jazyku GAMS pozostáva z jedného alebo viacerých príkazov. Príkazy sa môžu nachádzať na hociktorom mieste v riadku. Je povolené písať viac príkazov do jedného riadku a príkaz môže pokračovať aj na viacerých riadkoch. Podľa potreby môžeme vkladať prázdne riadky. Z uvedeného vyplýva, že v jazyku GAMS neexistuje presne predpísaný formát. Užívateľ sa musí riadiť len nasledujúcimi pravidlami. Symbol musí byť zadeklarovaný ešte predtým, ako je používaný a musí mať priradené hodnoty ešte predtým, ako sa vyskytne na pravej strane priraďovacieho príkazu. Každý riadok môže obsahovať až 16 385 znakov. Ďalšou vlastnosťou jazyka GAMS je, že nerozlišuje malé a veľké písmená, teda `HOTovost` vyjadruje to isté ako `hotovOST`. Niektoré riadky zdrojového kódu nie sú súčasťou jazyka GAMS. Napríklad, ak sa na prvej pozícii v riadku nachádza znak `*`, znamená to, že daný riadok obsahuje komentár. Alebo, ak sa v tejto pozícii objaví znak `$`, tak daný riadok obsahuje príkazy modifikujúce nastavenia kompilátora. Každý príkaz v jazyku GAMS patrí do jednej z dvoch nasledujúcich skupín:

a) Deklaračné príkazy - napr.: `alias`, `parameter`, `variable`, `model ...`
b) Vykonávacie príkazy - napr.: `display`, `solve`, `for`, `option ...`

V jazyku GAMS existuje päť základných dátových typov, ktoré sú uvedené v nasledujúcom zhrnutí, pričom v prvom stĺpci sa nachádza anglický názov (kľúčové slovo pre daný dátový typ), v druhom slovenský preklad tohto názvu a v treťom jeho ekvivalent v matematickej formulácii modelu:

1)	`sets`	množiny	množiny indexov
2)	`parameters`	parametre	koeficienty a parametre modelu
3)	`variables`	premenné	neznáme, hľadáme ich optimálne hodnoty
4)	`equations`	rovnice	účelová funkcia a jednotlivé ohraničenia
5)	`models`	modely	účelová funkcia a uvažovaný súbor ohraničení

Vďaka dátovému typu model môžeme flexibilne meniť formuláciu modelu vypustením alebo pridaním niektorých ohraničení. Špeciálnymi dátovými typmi sú `scalars` (skaláre) a `tables` (tabuľky). Oba slúžia na zadávanie dát v špecifickej forme. Identifikátor je po deklarácii a definícii prostredníctvom týchto dvoch typov uložený do pamäti ako dátový typ parameter. Takže v modeli s ním následne pracujeme ako s parametrom. Deklarácia konkrétneho dátového typu sa dá všeobecne napísať takto:

kľúčové slovo dátového typu	*identifikátor*	*obor*	*text*
`variable`	`x`	`( i , j )`	`"prepravené množstvo" ;`

Pričom obor a text nie sú povinné. `i` a `j` sú identifikátory dátového typu množina.

Definícia identifikátora znamená, že sú mu priradené konkrétne hodnoty. Definícia v rámci deklarácie sa nazýva inicializácia (časť v `/ /`) a vyzerá nasledovne:

```
set s "množina všetkých číslic" /0,1,2,3,4,5,6,7,8,9/ ;
```

Každý program napísaný v jazyku GAMS pozostáva z nasledujúcich komponentov:

a) znaky	b) kľúčové slová	c) identifikátory	d) prvky
e) vysvetľujúci text	f) čísla	g) komentáre	h) oddeľovače

Znaky: Pri písaní programov v jazyku GAMS je povolené používať len znaky, ktoré sa nachádzajú v nasledujúcej tabuľke:

od A po Z	od a po z	od 0 po 9
&	"	#
*	=	?
@	>	;
\	<	'
:	-	/
,	( )	medzera
$	[ ]	_
.	{ }	!
+	%	^

Výnimkou sú komentáre a vysvetľujúci text, v ktorých smieme používať ľubovoľné znaky. Na tomto mieste by sme chceli upozorniť, že v jazyku GAMS sú všetky tri typy zátvoriek, ktoré sme uviedli v predchádzajúcej tabuľke, navzájom zameniteľné.

Kľúčové slová: Sú to slová rezervované pre konkrétny príkaz a inde sa používať nesmú (okrem komentárov a textu v úvodzovkách). V nasledujúcej tabuľke sa nachádza ich zoznam:

abort	ge	option	sos1	else
acronym	gt	options	sos2	semicont
acronyms	inf	or	sum	semiint
alias	integer	ord	system	file
all	le	parameter	table	files
and	loop	parameters	using	putpage
assign	lt	positive	variable	puttl
binary	maximizing	prod	variables	free
card	minimizing	sameas	xor	no
diag	model	scalar	yes	solve
display	models	scalars	repeat	for
eps	na	set	until	
eq	ne	sets	while	
equation	negative	smax	if	
equations	not	smin	then	

Identifikátory: Sú to mená množín, parametrov, premenných ... . Inak povedané symboly, ktoré môžu mať 31 znakov, musia začínať písmenom a môžu obsahovať iba písmená, číslice a znak _. Meno, použité pre jeden dátový typ, už nemôže byť použité pre iný.

Prvky: Sú to elementy množín, ktoré môžu mať 31 znakov, musia začínať písmenom alebo číslicou a obsahovať môžu len písmená, číslice, znak + a znak -. Ak sú však v úvodzovkách (nezáleží na tom v ktorých), môžu obsahovať ľubovoľné legálne znaky, ale aj diakritiku. Prvky nemajú numerickú hodnotu, teda 2005 nemá numerickú hodnotu 2005, a zároveň 05 je iný prvok ako 5.

Vysvetľujúci text: Za identifikátormi a jednoduchými prvkami môže nasledovať vysvetľujúci text, ktorý sa musí nachádzať celý na tom istom riadku ako identifikátor či prvok a nesmie presiahnuť 255 znakov. Čo sa týka využiteľnosti znakov, aj tu sú isté obmedzenia. Text nesmie začínať dvoma bodkami .. alebo znakom = a nesmie obsahovať znak , , znak / a znak ; . Tieto obmedzenia sa dajú úplne odstrániť pomocou úvodzoviek. Text je súčasťou jazyka. Na rozdiel od komentára ho kompilátor spracúva a poskytuje vo výstupnom súbore viazaný na jemu prislúchajúci identifikátor alebo prvok.

Čísla: Všetky čísla sú v jazyku GAMS uložené v pamäti ako reálne. Neexistuje tu rozdelenie na celočíselné a reálne dátové typy. GAMS poskytuje možnosť používania rozšírenej aritmetiky (INF, -INF, EPS, NA). Pre zápis veľmi veľkých čísel sa používa známa forma cisloecislo. Napríklad zápis v jazyku GAMS 3e9 je ekvivalentom matematického zápisu $3*10^9$.

Oddeľovače: Ako oddeľovač jednotlivých príkazov slúži znak `;`, pričom ak nasledujúci príkaz začína kľúčovým slovom, nemusí byť tento znak prítomný. Ďalšími oddeľovačmi sú znak `,` , ktorý oddeľuje jednotlivé údaje v zozname údajov a znak `/` , ktorým vymedzujeme obsah tohto zoznamu.

Komentáre: Časti programu, ktoré nie sú spracúvané kompilátorom, ale slúžia iba ako poznámky pre užívateľa. V jazyku GAMS existujú tri spôsoby ako vložiť komentár. Prvým je už spomínaná `*` na prvej pozícii riadku. Prostredníctvom druhého spôsobu môžeme napísať aj mnohoriadkový text. Ide o už skôr uvedenú konštrukciu `$ontext-$offtext`. Znak `$` musí byť na prvej pozícii riadku, `$ontext` na začiatku bloku textu a `$offtext` na jeho konci:

```
$ontext
Tu sa môže nachádzať ľubovoľne dlhý text.
No za posledným riadkom tohto textu musí nasledovať príkaz $offtext.
$offtext
```

Tretí spôsob nám umožňuje vložiť komentár spoločne s nejakým príkazom na jeden riadok. Najprv ho však musíme aktivovať pomocou `$inlinecom` (komentár vo vnútri riadku) alebo `$eolcom` (komentár na konci riadku) nasledovne:

```
$inlinecom { }
$eolcom #
x = 1;     # toto je komentár
y = 3;     { toto je tiež komentár }  z = 5;
```

3 Dátový typ množina

Dátový typ množina tvorí základný stavebný kameň jazyka GAMS. Z matematického hľadiska sa jedná o množiny indexov. Prvky prislúchajúce k identifikátorom dátového typu množina sú ekvivalentom indexov v matematickej notácii. Kým v matematickej notácii sa na definovanie parametrov, premenných a ohraničení modelu používajú množiny číselných indexov, v jazyku GAMS môžeme na tieto účely používať aj rôzne nečíselné indexy. V tejto kapitole si ukážeme spôsob deklarácie a inicializácie dátového typu množina v jazyku GAMS.

3.1 Jednoduché množiny

3.1.1 Deklarácia a definícia jednoduchej množiny

Syntax:

```
set(s) meno_mnoziny ["text"] [/prvok ["text"] {, prvok ["text"]} /]
    {, meno_mnoziny ["text"] [/prvok ["text"] {, prvok ["text"]} /] };
```

V zobrazení syntaxe bolo použité značenie „Backus - Naur form“, ktorého podrobný popis nájdete v kapitole 1 Úvod. Pravidlá pre mená množín a pre vysvetľujúci text sú popísané v kapitole 2 Programy napísané v jazyku GAMS v odseku Identifikátory. Namiesto príkazu `set` môžeme písať príkaz `sets`, pretože z hľadiska jazykového významu sú ekvivalentné. Teda `(s)` znamená, že `s` tam môže byť, ale aj nemusí. Dôvodom existencie príkazu `sets` je čitateľnosť modelu, ako uvidíme neskôr.

Majme napríklad množinu M , definovanú súborom jej prvkov:

$$M = \{ BA , PP , KE \}$$

V jazyku GAMS túto množinu zapíšeme takto:

```
set M /BA,PP,KE/;
```

Toto sa nazýva deklarácia (`set M`) a definícia množiny (`/BA,PP,KE/`), kde slovo `set` je kľúčové slovo, vyjadrujúce, že sa jedná o dátový typ množina. `M` je identifikátor množiny a `BA`, `PP`, `KE` sú jej prvkami. Funkciu oddeľovača prvkov plní čiarka, ale môže to byť aj koniec riadku:

```
set M /BA
       PP
       KE/;
```

Za každým identifikátorom a jednoduchým prvkom môže nasledovať vysvetľujúci text, ktorý smie byť uvádzaný v oboch druhoch úvodzoviek alebo aj bez nich. Na rozdiel od komentára, vysvetľujúci text kompilátor berie ako súčasť vstupu a používa ho vo výstupe na popis prislúchajúceho identifikátora. V našom príklade by vysvetľujúci text mohol vyzerať nasledovne:

```
set M "jednotlivé mestá" /BA 'Bratislava', PP Poprad, KE "Košice"/;
```

Zoberme prípad, keď parameter vstupujúci do modelu bude matica A typu 3×2 . Treba zadeklarovať a zadefinovať dve množiny indexov, *m* pre riadky a *n* pre stĺpce. Symbolicky:

$$A = \left(a_{i\,j} \right), \text{kde } i \in m = \{1 , 2 , 3\} \text{ a } \; j \in n = \{1 , 2\}$$

Takáto deklarácia a definícia sa dá urobiť pre každú množinu osobitne alebo pre všetky naraz. V spoločnej deklarácii sa, z dôvodu čitateľnosti, zvykne používať kľúčové slovo `sets` namiesto `set`, ako to dokumentuje náš príklad:

```
sets m "riadky" /1,2,3/
     n "stĺpce" /1,2/;
```

To isté sa dá zapísať aj takto:

```
sets m "riadky" /1,2,3/ , n "stĺpce" /1,2/;
```

V tomto príklade sú prvkami čísla, preto je na mieste pripomenúť, že prvky nemajú numerickú hodnotu, teda prvok `3` nemá numerickú hodnotu 3 a zároveň `03` by bol iný prvok ako `3`.

3.1.2 Operátor * a príkaz alias

V definícii množiny `m` z predchádzajúceho príkladu by sme mohli napísať namiesto `1`, `2`, `3` výraz `1*3`. Operátor `*` slúži na uľahčenie zadávania množín, ktorých prvky tvoria postupnosť. Napríklad `x3y*x10y` vytvorí postupnosť nasledujúcich prvkov {x3y, x4y, … , x9y, x10y}. Treba poznamenať, že `x03y*x10y` vyjadruje inú postupnosť, a to {x03y, x04y, … , x09y, x10y}.

Všeobecné pravidlá operátora * :

1. Mená oboch prvkov sa môžu líšiť iba v jednom slede číslic, ostatné znaky (vrátane číslic mimo tohto sledu) sa musia zhodovať.
2. Číslo vytvorené sledom číslic na ľavej strane operátora `*` musí byť menšie ako to na jeho pravej strane a samozrejme obe musia byť celé čísla.

Potom GAMS vytvorí z takto zadaného výrazu celočíselnú postupnosť od čísla na ľavej strane operátora po číslo na jeho pravej strane a pomocou nej následne skonštruuje postupnosť prvkov.

Príklady nesprávneho použitia operátora `*`:

```
2y2*x5y5    ,    p9d*p4d    ,    jk3*jr9
```

Predstavme si, že teraz potrebujeme maticu B typu 3×3. Vďaka použitiu príkazu `alias`, ktorý nám pre už zadeklarovanú množinu vytvorí nové mená, nemusíme deklarovať a inicializovať dve rovnaké množiny, ale iba jednu. Pri príkaze `alias` na poradí argumentov nezáleží a môže byť použitý aj na tvorbu viacerých nových mien pôvodnej množiny:

```
set n "riadky" /1*3/;
alias (n1,n,n2,n3);
```

Množiny `n1`, `n2` a `n3` budú iba iné mená pre množinu `n`, teda budú obsahovať prvky `1`, `2`, `3` a tak môžeme napríklad `n1` použiť pre stĺpce matice. Symbolicky:

$$B = \left(b_{ij} \right), \text{ kde } \; i \in n \text{ a } \; j \in n1$$

3.1.3 Podmnožina a overovanie oboru („domain checking“)

V modelovaní sa často vyskytuje potreba používať rôzne podmnožiny hlavnej množiny. Z hľadiska jazyka GAMS je podmnožina identifikátorom dátového typu množina. Môžeme ju zadeklarovať dvoma rôznymi spôsobmi. Ako samostatnú množinu, obsahujúcu žiadané prvky, alebo ako množinu viazanú na hlavnú množinu. Táto väzba je vyjadrené identifikátorom hlavnej množiny v obore podmnožiny. Odporúčame používať druhú alternatívu, pretože v tomto prípade je kompilátor schopný pomocou overovania oboru identifikátora („domain checking“) overiť, či sú podmienky podmnožiny splnené alebo nie. Zoberme si príklad s mestami. Z nejakých dôvodov budeme potrebovať rozdeliť mestá na Západ, Stred a Východ. Dá sa to realizovať napríklad takto:

```
sets M "mesta" /BA,PP,KE/
     Z(M) "západ" /BA/ , S "stred" /PP/ , V "východ" /KR/;
```

Naším cieľom bolo zostaviť množiny `Z`, `S`, `V`, ktoré sú zároveň podmnožinami `M`. Čo sa týka `Z` a `S`, tak sa nám to podarilo. Pričom `Z` je a `S` nie je v deklarácii viazaná na `M`. Množina `V` taktiež nie je viazaná v deklarácii na `M`, ale na rozdiel od `S`, prvok z `V` nepatrí do `M`. Dôvodom je preklep v mene prvku (`KR` namiesto `KE`). Takýmto spôsobom by vznikol nežiadúci model a dával by nežiaduce výsledky. Ak by sme však zmenili formuláciu množiny `V` na:

```
set V(M) "východ" /KR/;    ,
```

tak počas kompilácie by kompilátor vyhlásil chybu „domain violation for element“. Čo znamená, že prvok z `V` nepatrí do `M` a podľa deklarácie `V` (`V(M)`) by tam mal patriť.

3.2 Viacrozmerné množiny

V modelovaní je často potrebné kombinovať prvky z jednej množiny s prvkami inej množiny a na tento účel slúžia viacrozmerné množiny. Ich použitie si vysvetlíme na nasledujúcom príklade:

```
sets p "prvá množina" /a,b,c/ , d "druhá množina" /1,2/
     pd1(p,d) /a.1 , b.2/ , pd2(p,d) /c.1 , c.2/
     pd3(p,d) /a.2 , c.2/ , pd4(p,d) /a.1 , a.2 , b.1 , b.2/;
```

Ako vidíme, viacrozmerné (v našom prípade dvojrozmerné) množiny sú viazané na jednorozmerné (jednoduché) prostredníctvom ich mien. Teda aj tu sa uplatňuje „domain checking“. Prvky sa zlučujú pomocou znaku . , pričom ich poradie je identické s poradím v obore identifikátora viacrozmernej množiny (vidno to napríklad na `pd2`). Takto môžeme zlúčiť najviac 10 prvkov rôznych jednoduchých množín. Spojenie dvoch alebo viacerých prvkov budeme odteraz označovať pojmom kombinácia prvkov. Na nasledujúcom príklade si ukážeme ako sa dá kombinovanie prvkov značne uľahčiť:

```
sets pd2(p,d) /c.(1,2)/ , pd3(p,d) /(a,c).2/ , pd4(p,d) /(a,b).(1,2)/;
```

Výsledkom sú tie isté množiny ako v predchádzajúcom príklade. Použitý formát vyjadruje karteziánsky súčin množín, ktorý sa v matematickej notácií zapisuje nasledovne:

$$\{1,...,n\} \times \{1,...,m\} \times \quad ... \quad \times \{1,...,k\}$$

A jeho ekvivalentom v jazyku GAMS je takýto zápis, pričom tri bodky nie sú súčasťou tohto jazyka, sú len symbolické:

```
(1*n).(1*m). ... .(1*k)
```

4 Dáta

V prvej časti tejto kapitoly objasníme ako sa zadávajú dáta do modelu prostredníctvom jazyka GAMS a v druhej časti si vysvetlíme ako ich je možné ďalej upravovať.

4.1 Zadávanie dát: Dátové typy skalár, parameter, tabuľka

V jazyku GAMS sa dáta zadávajú do modelu pomocou troch dátových typov, a to:

`scalar`	skalár	vhodný pre jeden konkrétny údaj
`parameter`	parameter	vhodný pre dáta majúce formu zoznamu
`table`	tabuľka	vhodný pre dáta majúce formu tabuľky

Typ skalár a typ tabuľka sú len skratkou k typu parameter. V konečnom dôsledku každý identifikátor obsahujúci dáta je v pamäti uložený ako typ parameter a neskôr s ním môžeme pracovať ako s parametrom. Všetky dáta zadané prostredníctvom typu skalár a typu tabuľka vieme zadať aj pomocou typu parameter. Cieľmi takejto diferenciácie bolo uľahčenie zadávania dát a čitateľnosť modelu napísaného v jazyku GAMS. V tejto časti textu si vysvetlíme deklaráciu a definíciu všetkých troch typov.

4.1.1 Dátový typ skalár

Pomocou tohto typu môžeme zadať iba parameter dimenzie[2] 0, čo znamená, že sa neviaže na žiadnu množinu. K jednému identifikátoru bude prislúchať iba jedno konkrétne číslo. Môžeme ho buď inicializovať počas deklarácie, alebo ho identifikátoru priradiť pomocou priraďovacieho príkazu, ktorý bude vysvetlený podrobnejšie v časti 4.2.1.

Syntax:

```
scalar(s) meno_skalaru ["text"] [/ cislo /]
        { meno_skalaru ["text"] [/ cislo /] };
```

Pravidlá pre mená skalárov a pre vysvetľujúci text sú popísané v kapitole 2 Programy napísané v jazyku GAMS. `cislo` je numerická hodnota vzťahujúca sa k identifikátoru `meno_skalaru`.

Príklad:

```
scalar hotovost "množstvo peňazí v hotovosti v Sk" / 1500 /;
```

alebo

```
scalar hotovost "množstvo peňazí v hotovosti v Sk";
hotovost = 1500;
```

4.1.2 Dátový typ parameter

Dátový typ parameter slúži na zadávanie dát formou zoznamu. Dimenzia identifikátora tohto typu sa môže pohybovať v rozmedzí 0 až 10. Prednastavená hodnota parametra je 0 pre všetky elementy.

Syntax:

```
parameter(s) meno_parametra ["text"] [/  prvok[=]cislo
                                       {,prvok[=]cislo} /]
           { meno_parametra ["text"] [/  prvok[=]cislo
                                       {,prvok[=]cislo} /] };
```

Pravidlá pre mená parametrov a pre vysvetľujúci text sú popísané v kapitole 2 Programy napísané v jazyku GAMS. `prvok` je prvok (kombinácia prvkov) množiny (množín), na ktorú (na ktoré) je parameter viazaný. Parametru priraďujeme pre jednotlivé prvky (kombinácie prvkov) numerickú hodnotu `cislo`. Ako operátor priradenia sa môže použiť znak `=` alebo medzera. Oddeľovačom

[2] Dimenzia v tomto ponímaní vyjadruje počet jednoduchých množín, prostredníctvom ktorých je identifikátor daného typu definovaný. Skalár má preto dimenziu 0. Vektor má dimenziu 1. Matica má dimenziu 2.

jednotlivých priradení, ale aj identifikátorov je znak , alebo koniec riadku (tak ako v prípade množín). (s) na konci kľúčového slova parameter má tiež rovnaký význam ako v prípade množín.

Príklad: 0-rozmerný parameter (nie je viazaný na žiadnu množinu):

```
parameter hotovost "množstvo peňazí v hotovosti v Sk" /1500/;
```

Príklad: Jednorozmerné parametre (je viazaný na jednu množinu):

```
set z "zamestnanci" /Jano,Fero,Peter/;
parameters m(z) "mzdy zamestnancov" /Jano=5000 , Fero=6000 , Peter=5500/
           h(z) "odpracované hodiny zamestnancov" /Jano   50
                                                   Fero   40
                                                   Peter  45/;
```

Keďže identifikátor m je viazaný na množinu z (m(z)), pri kompilácii prebieha „domain checking“. Treba poznamenať, že parameter, ktorý bol inicializovaný aspoň pre jeden prvok jeho oboru, existuje pre všetky prvky tohto oboru (celá množina z) a má hodnotu nula, pokiaľ sme ju nezmenili v inicializácii alebo v priradení. Z toho vyplýva, že nulové hodnoty nemusíme v inicializácii vypisovať. Keby v našom príklade Peter nič nezarobil, mohli by sme to zapísať nasledovne:

```
parameter m(z) "mzdy zamestnancov" /Jano=5000 , Fero=6000/;
```

Príklad: Viacrozmerný parameter (je viazaný na viaceré množiny):

V nasledujúcom príklade vložíme dvojrozmerný parameter, a to maticu $A = \begin{pmatrix} 450 & 520 \\ 450 & 570 \\ 450 & 600 \end{pmatrix}$. V našom príklade m označuje množinu riadkových a n množinu stĺpcových indexov. Prvok a_{12} matice A je vyjadrený kombináciou prvku 1 z m a prvku 2 z n ako 1.2 . Parametru v inicializácii priradíme pre tento prvok numerickú hodnotu 520. Upozorňujeme, že poradie množín v obore identifikátora si môžeme zvoliť ľubovoľne, následne však musíme toto poradie dodržať v kombinácii prvkov. Teda prvý prvok každej kombinácie musí byť z množiny, ktorá je prvá v obore identifikátora atď. Ak by sa tak nestalo, „domain checking“ by zlyhal a kompilátor by vyhlásil chybu. Správne zadanie matice:

```
sets m "riadky" /1*3/
     n "stĺpce" /1,2/;
parameter A(m,n) "matica zadaná formou zoznamu"
            /1.1=450 , 1.2=520 , 2.1=450 , 2.2=570 , 3.1=450 , 3.2=600/;
```

Táto matica má všetky prvky prvého stĺpca rovnaké, a preto jej zápis v jazyku GAMS môžeme zjednodušiť pomocou operátora * a formátu ( ).( ), známych z kapitoly 3 Dátový typ množina:

```
parameter A(m,n) /(1*3).1=450 , 1.2=520 , 2.2=570 , 3.2=600/;
```

Analogicky sa pracuje s parametrami dimenzií vyšších ako 2. Výsledkom nasledujúceho príkazu bude trojrozmerné pole nadobúdajúce hodnotu 5 pre každú kombináciu prvkov z karteziánskeho súčinu množín i, j, k. Opäť používame operátor * a formát ( ).( ):

```
sets i /1*3/ , j /1*3/ , k /1*3/;
parameter u(i,j,k) /(1*3).(1*3).(1*3)=5/;
```

Presvedčili sme sa, že pomocou dátového typu parameter naozaj vieme vložiť skalárne (hotovost) i tabuľkové (A(m,n)) dáta.

4.1.3 Dátový typ tabuľka

Ako sme videli na predchádzajúcich príkladoch, zadávať maticu pomocou typu parameter je veľmi neprehľadné. V tomto prípade odporúčame použiť dátový typ tabuľka, ktorý poskytuje lepšiu orientáciu a čitateľnosť. Tento dátový typ je určený na vkladanie dát dimenzie od 2 po 10.

Syntax:

```
table meno_tabulky ["text"] EOL
                 prvok  { prvok }
        prvok    cislo  { cislo } EOL
      { prvok    cislo  { cislo } EOL };
```

Pravidlá pre mená tabuliek a pre vysvetľujúci text sú popísané v kapitole 2 Programy napísané v jazyku GAMS. `cislo` je numerická hodnota identifikátora patriaca ku kombinácii „riadkového“ a „stĺpcového“ prvku. Slovo `EOL` vyjadruje nutnosť prítomnosti konca riadku (end of line). Tabuľka je jediným dátovým typom v jazyku GAMS, ktorý nemá voľný formát deklarácie a inicializácie. Je nutné dodržiavať tvar uvedený v syntaxe, vrátane koncov riadku. Číselný údaj sa musí aspoň čiastočne prekrývať s jeho „stĺpcovým“ prvkom a nesmie sa prekrývať so žiadnym iným „stĺpcovým“ prvkom. Na rozdiel od ostatných dátových typov, v prípade tabuľky môžeme deklarovať a inicializovať iba jeden identifikátor. Ak nejaký prvok v inicializácii vynecháme, povedzme „stĺpcový“, potom všetky hodnoty vynechaného stĺpca budú 0. Vynechanie jedného konkrétneho číselného údaju je to isté, ako by sme tam napísali 0.

Na nasledujúcich príkladoch si ilustrujeme deklaráciu a inicializáciu dátového typu tabuľka:

Príklad:

Chceme vložiť maticu $A = \begin{pmatrix} 15 & 27 \\ 15 & 27 \\ 15 & 27 \end{pmatrix}$ pomocou dátového typu tabuľka. Dá sa to realizovať takto:

```
sets m "riadky" /1*3/
     n "stĺpce" /1,2/;
table A(m,n) "matica zadaná formou tabuľky"
                1     2
            1   15    27
            2   15    27
            3   15    27 ;
```

V tomto prípade môžeme zápis zjednodušiť pomocou operátora `*` na:

```
table A(m,n)
                1     2
          1*3   15    27 ;
```

Príklad: Teraz si uvedieme príklad tabuľky dimenzie tri. Tvorí sa analogicky, ibaže v riadkoch, respektíve v stĺpcoch sa nachádzajú kombinácie viacerých prvkov rôznych množín. Je len na užívateľovi ako ich pospája. Následne však musí dodržať poradie množín v obore identifikátora. Kompilátor si vyžaduje v obore najskôr riadkové množiny v poradí, v akom vystupujú v kombinácii riadkových prvkov a potom stĺpcové taktiež v poradí identickom s kombináciou stĺpcových prvkov. Ak by sme nedodržali poradie, kompilátor by vyhlásil chybu „domain violation for element“. Tento príklad zároveň dokumentuje flexibilitu v pomenúvaní prvkov množín za použitia úvodzoviek:

```
sets tovar "druh predávaného tovaru" /"zimné čiapky","plavky"/
     predajne "mestá, v ktorých tovar predávame" /"Bratislava","Košice"/
     stvrtrok "štvrťroky" /"1.štvrťrok" * "4.štvrťrok"/;
table poc_pred_ks(tovar,predajne,stvrtrok) "počet predaných kusov"
                          "1.štvrťrok" "2.štvrťrok" "3.štvrťrok" "4.štvrťrok"
 "zimné čiapky"."Bratislava"     500          25          200          700
 "zimné čiapky"."Košice"         440          30          150          650
 "plavky"."Bratislava"            80         700          400           70
 "plavky"."Košice"                40         800          200           25 ;
```

Tú istú tabuľku môžeme zadať aj pomocou znaku `+` nasledovne:

```
table poc_pred_ks (tovar,predajne,stvrtrok) "počet predaných kusov"
                                  "1.štvrťrok"  "2.štvrťrok"
      "zimné čiapky"."Bratislava"     500            25
      "zimné čiapky"."Košice"         440            30
      "plavky"."Bratislava"            80           700
      "plavky"."Košice"                40           800

                +                 "3.štvrťrok"  "4.štvrťrok"
      "zimné čiapky"."Bratislava"     200           700
      "zimné čiapky"."Košice"         150           650
      "plavky"."Bratislava"           400            70
      "plavky"."Košice"               200            25 ;
```

Príklad: Ďalej si uvedieme príklad na zjednodušenie vkladania tabuliek, ktorých dáta sú v špeciálnom tvare. Za týmto účelom budeme predpokladať, že všetky číselné vstupy tabuľky z predchádzajúceho príkladu budú mať rovnakú hodnotu, a to 200. Tabuľka s takouto štruktúrou by sa dala zadať nasledovným zjednodušeným spôsobom:

```
table poc_pred_ks(tovar,predajne, stvrtrok) "počet predaných kusov"
                                                    "1.štvrťrok" * "4.štvrťrok"
   ("zimné čiapky","plavky").("Bratislava","Košice")            200           ;
```

4.2 Algebraická úprava dát prostredníctvom dátového typu parameter

Dáta, ktoré sme zadali, si môžu ešte vyžadovať rôzne algebraické úpravy. Tie sa dajú realizovať pomocou priraďovacieho príkazu. Pozostáva z ľavej strany, priraďovacieho znaku `=` a z pravej strany. Ak nasledujúci príkaz nezačína kľúčovým slovom, tak musí byť ukončený znakom `;`. Na ľavej strane sa nachádza parameter, ktorému chceme priradiť nejakú hodnotu. Táto hodnota sa nachádza na pravej strane vo forme algebraického výrazu alebo funkcie dostupnej v jazyku GAMS. V prvej časti sa budeme venovať samotnému príkazu priradenia a v druhej tomu, čo všetko smie vystupovať na pravej strane a v akej forme.

4.2.1 Príkaz priradenia

V tejto časti textu využívame na vysvetlenie jednotlivých príkazov nasledujúcu formu:

`príkaz;`	vysvetlenie príkazu

Skalárne priradenie:

`scalar hotovost /1500/;`	identifikátor `hotovost` sme inicializovali na 1500
`hotovost=1300;`	priradili sme mu 1300
`hotovost=hotovost+300;`	zvýšili sme ho o 300, teda sme mu priradili 1600

Priradenie viacrozmernému parametru: Na ľavej strane priradenia musí byť v obore identifikátora aspoň toľko množín, koľko ich je na pravej a zároveň na pravej strane nesmie vystupovať množina, ktorá sa nenachádza v obore identifikátora na ľavej. Tieto pravidlá platia pre pravú stranu bez prítomnosti indexového operátora. Ako to funguje s ním uvidíme v časti 4.2.2 Výrazy. Na nasledujúcom príklade si vysvetlíme rôzne spôsoby priradenia. Najskôr si však zadeklarujeme a inicializujeme používané množiny a parametre:

```
sets m "riadky" /1*3/
     n "stĺpce" /1,2/;
parameters A(m,n) /(1*3).1=1 , 1.2=2 , 2.2=1 , 3.2=3/
           b(n) /1=5 , 2=3/;
```

`A('1','1')=6;`	prvku a_{11} matice A sme priradili 6
`A(m,'1')=2*A('1','1');`	prvému stĺpcu matice A sme priradili dvojnásobok prvku a_{11} (12)
`A(m,n)=6+A(m,n);`	každý prvok matice A sme zvýšili o 6
`A('2',n)=6-b(n);`	druhému riadku matice A sme priradili vektor pravej strany (1 , 3)

Všimnime si, že ak chceme uviesť v obore identifikátora namiesto celej množiny jeden konkrétny prvok, musíme ho dať do úvodzoviek (buď `' '` alebo `" "`).

4.2.2 Výrazy

V tejto časti sa zameriame na analýzu pravej strany priraďovacieho príkazu z hľadiska algebraických možností jazyka GAMS.

Štandardné aritmetické operácie: Na pravej strane sa môžu vyskytovať všetky štandardné aritmetické operácie. Pre súčet GAMS používa znak `+`, pre rozdiel `-`, pre súčin `*` a pre podiel `/`. Umocňovanie realizuje pomocou operátora `**`. Avšak výraz `x**n` je v ňom počítaný ako exp[n ln(x)], z čoho vyplýva nutnosť obmedzenia sa na kladné x. Na výpočet celočíselných mocnín reálneho základu sa používa funkcia `power(x,n)`, pričom základ je `x` a exponent je `n`. Priority vyššie uvedených operátorov v jazyku GAMS sú v súlade s matematickými konvenciami, no z dôvodu prehľadnosti môžeme používať zátvorky (buď `()` alebo `{}` alebo `[]`). Teda:

`x=6*2-3**2*2;` je to isté ako `x={6*2}-[(3**2)*2];`

Indexové operátory: V jazyku GAMS existujú nasledujúce štyri:

`sum`	sumácia cez riadiacu množinu indexov
`prod`	súčin cez riadiacu množinu indexov
`smax`	maximálna hodnota cez riadiacu množinu indexov
`smin`	minimálna hodnota cez riadiacu množinu indexov

Ich zápis je identický, a preto si uvedieme iba príklad sumácie:

`sum(J, b(J))` $\sum_{j \in J} b_j$

`sum((I,J), a(I,J))` $\sum_{\substack{i \in I \\ j \in J}} a_{ij}$

Sčítavať môžeme nielen jednotlivé identifikátory, ale aj algebraické výrazy:

`b(J)=sum(I, a(J,I)*x(I));` $b_j = \sum_{i \in I} a_{ji}\, x_i \qquad j \in J$

Funkcie: V tejto časti si vysvetlíme väčšinu z funkcií dostupných v jazyku GAMS:

Funkcia	Popis
`errorf(x)`	kumulatívna distribučná funkcia štandardného normálneho rozdelenia v bode x
`exp(x)`	e^x
`log(x)`	$\ln x$
`log10(x)`	$\log x$
`normal(x,y)`	náhodné číslo ~ $N(x, y)$
`uniform(x,y)`	náhodné číslo z rovnomerného rozdelenia medzi hodnotami x a y
`abs(x)`	$\lvert x \rvert$
`ceil(x)`	najmenšie celé číslo väčšie alebo rovné x
`floor(x)`	najväčšie celé číslo menšie alebo rovné x
`mapval(x)`	pre numerické hodnoty 0, inak rôzne od nuly (špeciálne symboly)
`max(x,y,...)`	maximum spomedzi x, y, . . .
`min(x,y,...)`	minimum spomedzi x, y, . . .
`mod(x,y)`	zvyšok po vydelení čísla x číslom y
`power(x,y)`	x^y, kde y musí byť celé číslo

Funkcia	Popis
round(x)	zaokrúhlenie x na najbližšie celé číslo
round(x,y)	zaokrúhlenie x na y desatinných miest, pričom $y = -2$ zaokrúhli x na stovky
sign(x)	vracia 1 ak $x > 0$, -1 ak $x < 0$ a 0 ak $x = 0$
sqr(x)	x^2
sqrt(x)	$\sqrt{x}$
trunc(x)	odstráni desatinný rozvoj čísla x a urobí z neho celé
arctan(x)	$\tan^{-1}(x)$, výsledok je v radiánoch
cos(x)	$\cos(x)$
sin(x)	$\sin(x)$

Rozšírená aritmetika - špeciálne symboly ako výsledky aritmetických operácií: Niekedy sa môže stať, že výsledkom algebraického výrazu sú špeciálne hodnoty. Všetky možné budú teraz uvedené (prvý stĺpec) a vysvetlené (tretí stĺpec). Číslo v zátvorke (druhý stĺpec) označuje hodnotu funkcie mapval pre prislúchajúci špeciálny symbol. Táto funkcia dáva nenulovú hodnotu len, ak je jej argumentom jeden z nasledujúcich špeciálnych symbolov:

INF	(6)	+∞, môžeme s ním pracovať ako s numerickou hodnotou (napr. priradenie)
-INF	(7)	-∞, môžeme s ním pracovať ako s numerickou hodnotou
NA	(5)	používa sa ak chýbajú niektoré dáta, výsledok ľubovoľnej operácie s NA je NA
UNDF	(4)	daná operácia nie je definovaná, užívateľ nemôže tento symbol používať
EPS	(8)	hodnota veľmi blízka, ale nie rovná nule

Ak potrebujeme používať číslo väčšie ako 10^{299}, mali by sme ho nahradiť symbolom INF, aby sme sa vyhli nedefinovaným operáciám. Aritmetické operácie obsahujúce symboly INF, -INF, EPS sú v jazyku GAMS definované v súlade s matematickou teóriou. Všetky, okrem UNDF, môžeme použiť na pravej strane priradenia. Význam symbolu NA spočíva v tom, že ak ním nahradíme hodnotu identifikátora pre chýbajúce dáta, môžeme sa pokúsiť model vyriešiť, pričom výsledok každej operácie obsahujúcej hodnotu NA bude NA.

5 Dátový typ premenná

Dátový typ premenná slúži na deklaráciu neznámych v modeli a na manipuláciu s nimi.

5.1 Deklarácia

Rozdiel v deklarácii premennej oproti predchádzajúcim typom je, že ju neinicializujeme.

Syntax:

```
[typ_premennej] variable(s) meno_premennej ["text"] {, meno_premennej ["text"]};
```

Pravidlá pre mená premenných a pre vysvetľujúci text sú popísané v kapitole 2 Programy napísané v jazyku GAMS. Na mieste slova `typ_premennej` sa môže nachádzať kľúčové slovo, vyjadrujúce konkrétny typ premennej. Ak žiadne neuvedieme, premenná bude voľná. V nasledujúcej tabuľke sú vymenované a popísané všetky dostupné typy premenných:

Typ premennej	Kľúčové slovo	Prednastavebá dolná hranica	Prednastavená horná hranica	Popis
voľná	`free`	`-inf`	`+inf`	Nie je ohraničená zdola ani zhora.
nezáporná	`positive`	`0`	`+inf`	Záporné hodnoty sú zakázané.
nekladná	`negative`	`-inf`	`0`	Kladné hodnoty sú zakázané.
binárna	`binary`	`0`	`1`	Môže nadobúdať iba 0 alebo 1.
celočíselná	`integer`	`0`	`100`	Môže nadobúdať iba celé čísla.
	`sos1`	`0`	`+inf`	Skupina premenných, z ktorých môže byť nenulová najviac jedna.
	`sos2`	`0`	`+inf`	Skupina premenných, z ktorých môžu byť nenulové najviac dve.
	`semicont`	`1`	`+inf`	Musí byť buď 0 alebo väčšia ako dolná. hranica.
	`semiint`	`1`	`100`	Musí byť celočíselná a buď 0 alebo väčšia ako dolná hranica.

Príklad: Teraz si uvedieme príklad deklarácie viacerých premenných rôznych typov:

```
variables
  z "zisk"
  x(t) "nakúpené množstvo jednotlivých tovarov, ktoré sú z množiny t"
  m(n,n1) "množ. investícii v i-tom mes. na j mes., kde i je z n a j z n1"
  p "množstvo nakúpeného piesku";
integer variable x;
positive variables m,p;
```

V tomto prípade sme najprv zadeklarovali všetky premenné spolu, a potom určili ich typ. Niekomu môže viac vyhovovať nasledujúci štýl, kde premenné deklarujeme po skupinách vytvorených na základe typu:

```
free variable
  z "zisk";
integer variable
  x(t) "nakúpené množstvá jednotlivých tovarov, ktoré sú z množiny t";
positive variables
  m(n,n1) "množ. investícii v i-tom mes. na j mes., kde i je z n a j z n1"
  p "množstvo nakúpeného piesku";
```

Všimnime si, že premenné, rovnako ako parametre, môžu byť viazané na množinu indexov. Napríklad premenná `x` je viazaná na množinu `t`, teda vektor `x` má toľko zložiek, koľko má `t` prvkov a ku každému prvku z `t` patrí jedna zložka vektoru `x`.

5.2 Atribúty premenných

Premenná obsahuje pre jeden konkrétny prvok viac atribútov. Všetky predchádzajúce dátové typy mali len jeden atribút, a to hodnotu. Jednej premennej môžeme priradiť viacero atribútov vďaka pridávaniu prípon k jej identifikátoru. Všetky dostupné atribúty sú vymenované v nasledujúcej tabuľke:

Atribút	Prípona
Dolná hranica	`.lo`
Horná hranica	`.up`
Konštantná hodnota	`.fx`
Hodnota	`.l`
Hraničná hodnota	`.m`
Škálovanie	`.scale`
Priorita vetvenia	`.prior`

Všetky tieto atribúty môže užívateľ modifikovať pomocou priraďovacieho príkazu. Prípona `.fx` plní tú istú funkciu ako priradenie rovnakej hodnoty do `.lo` a `.up`. Prípona `.l` obsahuje po vyriešení modelu optimálnu hodnotu premennej. Priradením nejakej hodnoty do `.l` pred spustením riešiaceho procesu volíme štartovací bod použitého algoritmu. Prípona `.prior` sa používa v modeloch celočíselného programovania (`MIP`).

Príklad: Uvedieme si príklad použitia rôznych atribútov v príkaze priradenia na ľavej ale i pravej strane. Najprv priradíme do `x`, deklarovaného vyššie, dolnú hranicu `1` pre všetky prvky množiny `t` a potom pre jeden konkrétny (`t1`) posunieme dolnú hranicu späť na `0`. Premennú `z` fixujeme na hodnotu `100000` a následne zadáme škálovaciu hodnotu `0.001`. To spôsobí, že z pohľadu solvera sa `z` v celom modeli nahradí fiktívnym `z1`, pričom `z1=z*0.001`, teda `100`. Ďalší príklad dokumentuje použitie atribútov na pravej strane priradenia. Ak by sa tento príkaz nachádzal až za príkazom `solve`, v priradení by sme počítali s už optimálnou hodnotou `z`. Teda po vyriešení modelu vieme pomocou atribútov `.l` a `.m` veľmi jednoducho vypočítať rôzne informatívne koeficienty:

```
x.lo(t)=1;
x.lo('t1')=0;
z.fx=100000;
z.scale=0.001;
scalar pz "počet zamestnancov" /100/;
scalar zpk "zisk per kapita (na hlavu), teda zisk/počet zamestnancov(pz)";
zpk=z.l/pz;
```

Identifikátor dátového typu premenná sa v príkaze `display`, na rozdiel od parametrov a množín, musí vyskytovať aj s niektorou z jeho prípon. Príkaz `display` slúži na zobrazenie hodnôt konkrétnych identifikátorov do výstupného súboru. Pomocou nasledujúceho zápisu zobrazíme dolné hranice premennej `x` pre všetky prvky množiny `t` a hodnotu premennej `z` v danom okamihu:

```
display x.lo,z.l;
```

6 Dátový typ rovnica

Dátový typ rovnica v sebe zahŕňa všetky typy ohraničení z matematickej formulácie modelu. Teda nielen rovnice, ako by sa mohlo zdať z názvu. Deklarácia a definícia tohto dátového typu musí prebehnúť oddelene.

6.1 Deklarácia

Deklarácia identifikátora dátového typu rovnica je podobná ako pre množiny alebo parametre.

Syntax:

```
equation(s) meno_rovnice ["text"] {, meno_rovnice ["text"]};
```

Pravidlá pre mená rovníc a pre vysvetľujúci text sú popísané v kapitole 2 Programy napísané v jazyku GAMS.

Príklad: V nasledujúcom príklade zadeklarujeme dva identifikátory typu rovnica, z čoho jeden bude skalárny a druhý bude viazaný na množinu indexov `vf`. Množina `vf` obsahuje jednotlivé výrobné faktory a ku každému z nich bude existovať jedno ohraničenie. Identifikátor `zdroje(vf)` bude teda v sebe zahŕňať toľko samostatných rovníc, koľko má `vf` prvkov. Pomocou `uf` bude neskôr definovaná účelová funkcia. Pre úplnosť uvádzame aj deklaráciu množiny `vf`:

```
set vf "jednotlivé výrobné faktory" /praca , kov , sklo/;
equations uf "účelová funkcia"
          zdroje(vf) "ohraničenia na zdroje";
```

6.2 Definícia

Pomocou definície identifikátorov typu rovnica vieme v jazyku GAMS skonštruovať účelovú funkciu a jednotlivé ohraničenia v súlade s matematickou formuláciou modelu.

Syntax:

```
meno_rovnice (obor) ..  vyraz typ_rovnice vyraz;
```

Slovo `obor` označuje obor identifikátora. Znaky `..` sú nutné na oddelenie mena od algebraického vyjadrenia. Dve slová `vyraz` vyjadrujú ľavú a pravú stranu rovnice a slovo `typ_rovnice` zase vzťah medzi nimi. Na mieste slova `typ_rovnice` sa môžu nachádzať nasledujúce symboly:

`=e=`	rovnosť, ľavá strana rovnice sa musí rovnať pravej strane
`=g=`	nerovnosť, ľavá strana rovnice musí byť väčšia alebo rovná pravej strane
`=l=`	nerovnosť, ľavá strana rovnice musí byť menšia alebo rovná pravej strane
`=n=`	nie je nutný žiaden vzťah medzi ľavou a pravou stranou rovnice

Príklad: Uvedieme definície rovníc, ktoré sme deklarovali v predchádzajúcej časti tejto kapitoly, pričom v algebraických výrazoch sa nachádzajú premenné `cz`, `x(r)` a parametre `z(r)`, `S(vf,r)`, `m(vf)`. Upozorňujeme, že premenné sa v definíciách rovníc uvádzajú bez prípon. Model, z ktorého sme tieto rovnice vybrali, môžete nájsť v podkapitole 17.2:

```
uf ..          cz =e= sum(r,z(r)*x(r));
zdroje(vf) ..         sum(r,S(vf,r)*x(r)) =l= m(vf);
```

V matematickej formulácii nerovnosť $x < 5 + y$ a nerovnosť $x - 5 - y < 0$ sú ekvivalentné. To isté platí v jazyku GAMS, preto aj v definícii rovníc je viac menej voľný formát konštrukcie oboch strán relačného vzťahu. Môžu sa tu nachádzať všetky druhy matematických výrazov použiteľné v priradení a tiež rôzne funkcie zabudované v jazyku GAMS. Upozorňujeme, že nie všetky funkcie, použiteľné v priradení, sa dajú zahrnúť do výrazu v rovnici. Napríklad funkcie vracajúce náhodné čísla sa nesmú vyskytnúť v rovnici za žiadnych okolností. Čo sa týka ostatných, závisí to od ich argumentu. Argument môže byť exogénny alebo endogénny. Exogénny je taký, ktorého hodnota je známa, napríklad parameter alebo aj nejaký atribút premennej (= premenná s nejakou príponou). Hodnota sa vypočíta len raz, a to na začiatku riešiaceho algoritmu. Naproti tomu, funkcia endogénneho argumentu

sa počíta v každej iterácii. Argument je neznámy, a teda obsahuje identifikátor typu premenná, ale bez akejkoľvek prípony. Ešte predtým, ako si ukážeme možnosti použitia funkcií v rovniciach, uvedieme ich nasledujúcu klasifikáciu v závislosti od ich spojitosti resp. nespojitosti a spojitosti resp. nespojitosti ich prvých derivácií:

Klasifikácia	Funkcia	Prvá derivácia	Príklad
Hladká	Spojitá	Spojitá	`exp`
Nehladká	Spojitá	Nespojitá	`abs`
Nespojitá	Nespojitá	Nespojitá	`sign`

Teraz uvedieme prehľad možností použitia jednotlivých funkcií v rovniciach v závislosti od typu argumentu, pričom v poslednom stĺpci sa nachádza typ modelu, ktorý vznikne zahrnutím rovnice obsahujúcej danú funkciu endogénneho argumentu. Význam jednotlivých skratiek nájdete v kapitole 7 Dátový typ model:

Funkcia	Klasifikácia	Exogénny argument	Endogénny argument	Typ modelu
`errorf(x)`	Hladká	Povolená	Povolená	NLP
`exp(x)`	Hladká	Povolená	Povolená	NLP
`log(x)`	Hladká	Povolená	Povolená	NLP
`log10(x)`	Hladká	Povolená	Povolená	NLP
`normal(x,y)`	-	Nepovolená	Nepovolená	
`uniform(x,y)`	-	Nepovolená	Nepovolená	
`abs(x)`	Nehladká	Povolená	Povolená	DNLP
`ceil(x)`	Nespojitá	Povolená	Nepovolená	
`floor(x)`	Nespojitá	Povolená	Nepovolená	
`mapval(x)`	Nespojitá	Povolená	Nepovolená	
`max(x,y...)`	Nehladká	Povolená	Povolená	DNLP
`min(x,y...)`	Nehladká	Povolená	Povolená	DNLP
`mod(x,y)`	Nespojitá	Povolená	Nepovolená	
`power(x,y)`	Hladká	Povolená	Povolená	NLP
`round(x)`	Nespojitá	Povolená	Nepovolená	
`round(x,y)`	Nespojitá	Povolená	Nepovolená	
`sign(x)`	Nespojitá	Povolená	Nepovolená	
`sqr(x)`	Hladká	Povolená	Povolená	NLP
`sqrt(x)`	Hladká	Povolená	Povolená	NLP
`trunc(x)`	Nespojitá	Povolená	Nepovolená	
`arctan(x)`	Hladká	Povolená	Povolená	NLP
`cos(x)`	Hladká	Povolená	Povolená	NLP
`sin(x)`	Hladká	Povolená	Povolená	NLP

6.3 Atribúty rovníc

Rovnice majú päť atribútov, a to `.l`, `.m`, `.lo`, `.up`, `.scale`. Nech zadáme rovnicu v akomkoľvek tvare, kompilátor ju upraví tak, že všetky neznáme sú na ľavej strane a všetky konštanty, vo forme jedného výsledného čísla, sú na pravej strane. Nasledujúca tabuľka obsahuje hodnoty atribútov rovníc podľa ich typu (`.l`, `.lo`, `.up` majú rovnaký význam ako pre premenné):

Typ rovnice	`.lo`	`.up`	`.l`
`=e=`	Pravá strana	Pravá strana	Ľavá strana
`=g=`	Pravá strana	`inf`	Ľavá strana
`=l=`	`-inf`	Pravá strana	Ľavá strana
`=n=`	`-inf`	`inf`	Hociktorá strana

Atribút `.m` obsahuje hraničnú hodnotu. Atribút `.scale` slúži na preškálovanie rovnice jeho hodnotou. Práca s atribútmi rovníc je založená na rovnakom princípe ako v prípade premenných.

7 Dátový typ model

Dátový typ model slúži na spojenie žiadaných rovníc do jedného súboru, čím sa vytvorí model v súlade s matematickou formuláciou. Do modelu môžeme zahrnúť ktorúkoľvek z rovníc, ktoré boli už skôr deklarované. Upozorňujeme, že nemuseli byť ešte definované.

7.1 Deklarácia a inicializácia

Syntax:

```
model(s) meno_modelu ["text"] [/all | meno_rovnice {, meno_rovnice}/]
      {, meno_modelu ["text"] [/all | meno_rovnice {, meno_rovnice}/] };
```

Pravidlá pre mená modelov a pre vysvetľujúci text sú popísané v kapitole 2 Programy napísané v jazyku GAMS. Medzi znakmi `//` sa nachádzajú buď mená jednotlivých rovníc, ktoré chceme zahrnúť do modelu, alebo slovo `all`. Použitím tohto špeciálneho slova dosiahneme, že v danom modeli sa budú nachádzať všetky doteraz deklarované rovnice.

Príklad: Teraz si uvedieme príklad skonštruovania dvoch rôznych modelov v jednom príkaze. Predpokladajme, že sa jedná o veľmi jednoduchý model, v ktorom vyrábame jeden tovar pomocou jedného výrobného faktoru. V prvom modeli poznáme dopyt po tomto tovare. V druhom predpokladáme, že všetko, čo vyrobíme, sa predá:

```
equations zisk "rovnica zisku z predaja vyrábaného tovaru"
          zdroj "mám len konečné množstvo výrobného faktoru"
          dopyt "dopytové ohraničenie";
models VsKD "model výroby s konečným dopytom" /all/
       VsND "model výroby s nekonečným dopytom" /zisk , zdroj/;
```

7.2 Klasifikácia modelov

Pomocou jazyka GAMS sa dajú riešiť rôzne typy modelov. Každý skonštruovaný model musí byť jedného z nasledujúcich typov:

`LP`	Linear programming
`NLP`	Nonlinear programming
`DNLP`	Nonlinear programming with discontinuous derivatives
`RMIP`	Relaxed mixed integer programming
`MIP`	Mixed integer programming
`RMINLP`	Relaxed mixed integer nonlinear programming
`MINLP`	Mixed integer nonlinear programming
`MPEC`	Mathematical programs with equilibrium constraints
`MCP`	Mixed complementarity problems
`CNS`	Constrained nonlinear system
`QCP`	Quadratically constrained program
`RMIQCP`	Relaxed mixed integer quadratically constrained program
`MIQCP`	Mixed integer quadratically constrained program

7.3 Atribúty modelov

Identifikátor dátového typu model obsahuje rôzne atribúty. Zoznam tých najdôležitejších:

Prípona	Popis	Kontroluje	Prednastavená hodnota	Globálne nastavenie
domlim	maximálny počet chýb „domain violation“	užívateľ	0	domlim
domusd	počet výskytov chyby „domain violation“	solver		
holdfixed	nahrádzanie fixovaných premenných	užívateľ	0	
0	nenahrádzajú sa v modeli			
1	nahrádzajú sa v modeli			
iterlim	maximálny počet iterácií	užívateľ	1000	iterlim
iterusd	použitý počet iterácií	solver		
limcol	počet stĺpcov zobrazených pre každý blok premenných	užívateľ	3	limcol
limrow	počet riadkov zobrazený pre každý blok rovníc	užívateľ	3	limrow
modelstat	kód vyjadrujúci stav modelu	solver		
1	OPTIMAL (optimálny)			
2	LOCALLY OPTIMAL (lokálne optimálny)			
3	UNBOUNDED (neohraničený)			
4	INFEASIBLE (neprípustný)			
5	LOCALLY INFEASIBLE (lokálne neprípustný)			
6	INTERMEDIATE INFEASIBLE (priebežne neprípustný)			
7	INTERMEDIATE NONOPTIMAL (priebežne neoptimálny)			
8	INTEGER SOLUTION (celočíselné riešenie)			
9	INTERMEDIATE NON-INTEGER (priebežne neceločíselný)			
10	INTEGER INFEASIBLE (celočíselne neprípustný)			
11	nepoužívaný kód			
12	ERROR UNKNOWN (neznáma chyba)			
13	ERROR NO SOLUTION (chyba – žiadne riešenie)			
numequ	počet jednotlivých rovníc v modeli	solver		
numinfes	počet „neprípustností“	solver		
numnopt	počet „neoptimálností“	solver		
numnz	počet prvkov ≠ 0 v matici koeficientov	solver		
numunbnd	počet neohraničených premenných	solver		
numvar	počet jednotlivých premenných v modeli	solver		
optca	absolútne kritérium na ukončenie pre MIP	užívateľ	0,0	optca
optcr	relatívne kritérium na ukončenie pre MIP	užívateľ	0,1	optcr
reslim	časový limit (v CPU sekundách) pre solver	užívateľ	1000	reslim
resusd	čas použitý na vyriešenie modelu (v CPU sekundách)	solver		

Prípona	Popis	Kontroluje	Prednastavená hodnota	Globálne nastavenie
scaleopt	nastavenie škálovania modelu	užívateľ	0	
solprint	nastavenie zobrazenia riešenia vo výstupe	užívateľ	1	solprint
solveopt	nastavenie zápisu výsledkov do výstupu	užívateľ	1	solveopt
solvestat	stav solvra	solver		
1	NORMAL COMPLETION			
2	ITERATION INTERRUPT			
3	RESOURCE INTERRUPT			
4	TERMINATED BY SOLVER			
5	EVALUATION ERROR LIMIT			
6	UNKNOWN			
7	(nepoužívaný)			
8	ERROR PREPROCESSOR ERROR			
9	ERROR SETUP FAILURE			
10	ERROR SOLVER FAILURE			
11	ERROR INTERNAL SOLVER ERROR			
12	ERROR POST-PROCESSOR ERROR			
13	ERROR SYSTEM FAILURE			
sysout	nastavenie vypisovania údajov od podsystému	užívateľ	0	sysout
workspace	veľkosť pracovnej pamäte (v MB)	užívateľ		work

Niektoré z uvedených atribútov podrobnejšie popíšeme, keď bude ich použitie aktuálne.

Príklad: Zmeníme povolený počet iterácií v modeli VsKD z prednastavených 1000 na 100:

```
VsKD.iterlim=100;
```

8 Príkaz solve

Ak máme zostavený model, všetky rovnice, parametre a množiny v ňom obsiahnuté sú definované, môžeme sa pokúsiť ho vyriešiť pomocou príkazu `solve`. Treba poznamenať, že samotný GAMS úlohu nerieši, ale len odovzdáva samostatnému riešiacemu programu. Takýchto programov je s jazykom GAMS kompatibilných asi 30, vďaka čomu by mali byť dostupné mnohé v súčasnosti používané algoritmy na riešenie optimalizačných úloh. Na konci tejto kapitoly ešte ukážeme ako nastaviť solvre pre jednotlivé typy modelov.

Syntax:

```
solve meno_modelu using typ_modelu maximizing | minimizing meno_premennej; |
solve meno_modelu maximizing | minimizing meno_premennej using typ_modelu;
```

Príklad: Použijeme model `VsKD` deklarovaný v úvode predchádzajúcej kapitoly. Aby sme nemuseli uvádzať celý model, budeme predpokladať, že všetky množiny, parametre, rovnice, sú už definované a všetky premenné deklarované, pričom premenná `z` vyjadruje hodnotu účelovej funkcie. Potom by príkaz `solve` vyzeral nasledovne (uvádzame obe možnosti syntaxe):

```
solve VsKD using LP maximizing z;
solve VsKD maximizing z using LP;
```

Slová `solve` a `using` sú kľúčové slová, `VsKD` je `meno_modelu`, `LP` je `typ_modelu`, `maximizing` je smer optimalizácie a `z` je `meno_premennej` (hodnota účelovej funkcie). Ak by sme chceli minimalizovať hodnotu účelovej funkcie, tak namiesto slova `maximizing` by sme napísali slovo `minimizing`. V jednom programe môžeme použiť aj viacero príkazov `solve`.

Príklad: V úlohe, ktorou sme sa zaoberali v predchádzajúcej kapitole, by sme chceli vyriešiť a mať zdokumentované vo výstupnom súbore oba modely (`VsKD` aj `VsND`). Dá sa to realizovať nasledovne:

```
solve VsKD using LP maximizing z;
solve VsND using LP maximizing z;
```

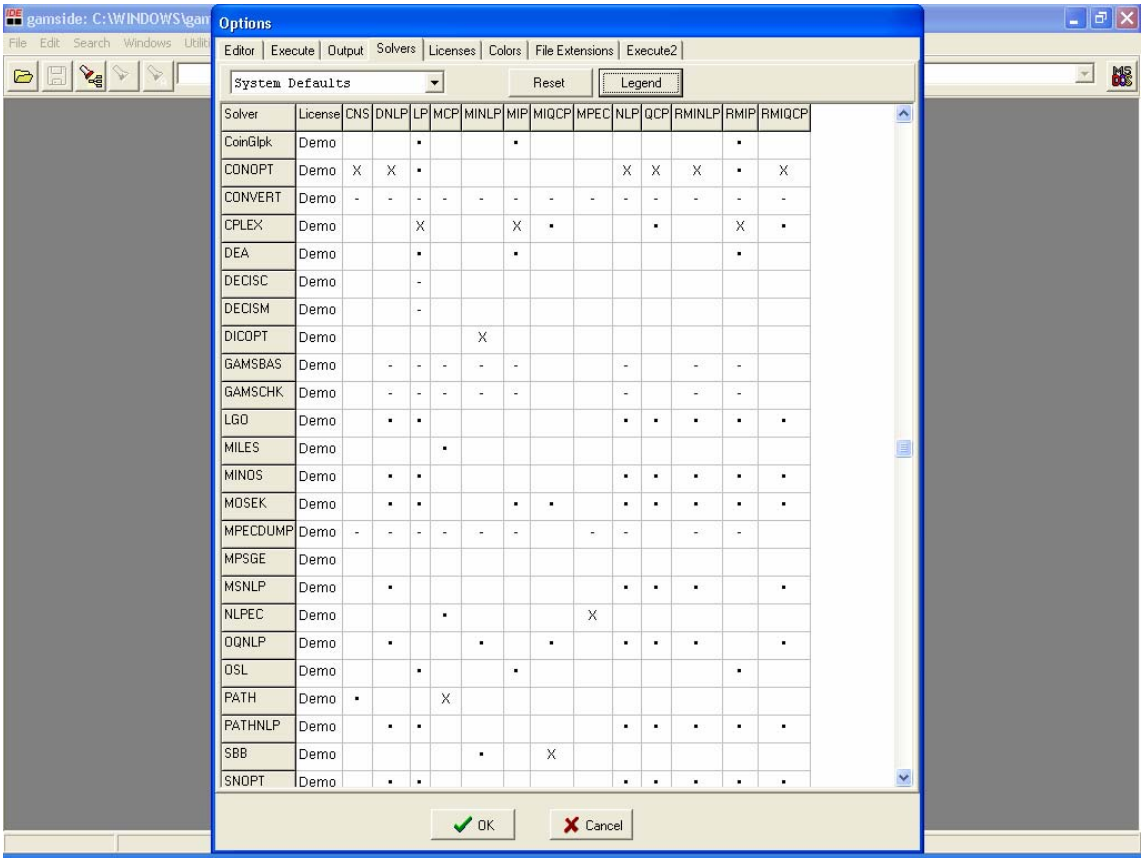
Podmienky platnosti príkazu solve:

a) Všetky rovnice vystupujúce v modeli boli už skôr zadefinované a premenná, vyjadrujúca hodnotu účelovej funkcie, figuruje v týchto rovniciach aspoň raz.
b) Premenná, vyjadrujúca hodnotu účelovej funkcie, je skalárna a voľná.
c) Vlastnosti každej rovnice súhlasia s vlastnosťami nami zvoleného typu modelu. Čím sa myslí, že ak je model napríklad typu `LP`, tak každá rovnica musí byť lineárna.
d) Všetky množiny a parametre vystupujúce v rovniciach už majú priradené hodnoty.

Činnosti spúšťané príkazom solve:

a) Upravenie modelu do tvaru vyžadovaného solverom.
b) Ladenie programu, vytvorenie dokumentácie modelu a jej zapísanie do výstupného súboru.
c) Overenie konzistencie hraníc a neprítomnosti neprípustných hodnôt (`UNDF`) v modeli.
d) V prípade, že sa pri vykonávaní predchádzajúcich krokov objavila chyba, nastáva prerušenie behu programu a poskytnutie popisu tejto chyby.
e) Odovzdávanie kontroly nad problémom solveru a čakanie na výsledky.
f) Prebranie výsledkov, vytvorenie dokumentácie, jej zapísanie do výstupného súboru, priradenie nových hodnôt poskytnutých solverom do atribútov `.l` a `.m` premenných i rovníc a v prípade, že v tejto fáze nastala chyba, poskytnutie jej popisu.

Teraz si ukážeme, ako nastaviť konkrétny solver pre daný typ modelu v prostredí pre Windows. Znak `x` znamená, že pre daný typ modelu sme zvolili solver prislúchajúci k tomuto riadku, teda napríklad pre `LP` sa bude používať CPLEX. Tento solver bude zároveň riešiť aj modely typu `MIP` a `RMIP`. Znak `.` vyjadruje dostupnosť daného solvera pre daný typ modelu. A znak - označuje solvery dostupné len cez príkaz `option`:


gamside: C:\WINDOWS\gam
File Edit Search Windows Utilit
Options
Editor Execute Output Solvers Licenses Colors File Extensions Execute2
System Defaults
Reset
Legend
Solver License CNS DNLP LP MCP MINLP MIP MIQCP MPEC NLP QCP RMINLP RMIP RMIQCP
CoinGlpk Demo . . .
CONOPT Demo X X . X X X . X
CONVERT Demo - - - - - - - - - - - - -
CPLEX Demo X . . X . .
DEA Demo . . .
DECISC Demo -
DECISM Demo -
DICOPT Demo X
GAMSBAS Demo - - - - - - - - - -
GAMSCHK Demo - - - - - - - - - -
LGO Demo . . . .
MILES Demo . .
MINOS Demo . . . .
MOSEK Demo . . . .
MPECDUMP Demo - - - - - - - - - - -
MPSGE Demo
MSNLP Demo . . . . . .
NLPEC Demo . X
OQNLP Demo . . . .
OSL Demo . . .
PATH Demo . X
PATHNLP Demo . . . .
SBB Demo . X
SNOPT Demo . . . .
OK
Cancel

9 Podmienené výrazy, podmienené priradenia a podmienené rovnice

Pri modelovaní reálneho sveta potrebujeme veľmi často zahrnúť do modelu rôzne druhy výnimiek. Napríklad, v modeli by bolo treba vynechať ohraničenia prislúchajúce k niektorým konkrétnym indexom, alebo by sme potrebovali vynechať niektorý index v rámci sumácie. Napríklad:

$$\sum_{\substack{i=1 \\ i \neq 5}}^{10} x_i$$

Takéto výnimky sa dajú realizovať pomocou podmienených príkazov. V prvej časti kapitoly sú popísané nástroje jazyka GAMS na konštrukciu logických podmienok. V druhej, následné využitie týchto podmienok na tvorbu podmienených priradení a podmienených rovníc.

9.1 Logické podmienky

Logické podmienky môžeme tvoriť v jazyku GAMS pomocou logických výrazov, numerických výrazov alebo ich vzájomnej kombinácie. Každý výraz nadobúda numerickú aj logickú hodnotu. Akú, to sa dozvieme v tejto časti kapitoly. Okrem toho si ukážeme, ako sa dá príslušnosť k množine použiť v úlohe logickej podmienky.

Logické výrazy: Môžu nadobúdať logickú hodnotu `True` alebo `False`. Ak `True`, numerická hodnota je 1. Ak `False`, numerická hodnota je 0. Logické výrazy konštruujeme pomocou relačných a logických operátorov. Všetky dostupné sú uvedené v nasledujúcich dvoch tabuľkách:

Relačné operátory		
Operátor		**Popis**
`lt`	`<`	menšie
`le`	`<=`	menšie rovné
`eq`	`=`	rovné
`ne`	`<>`	rôzne
`ge`	`>=`	väčšie rovné
`gt`	`>`	väčšie

Logické operátory	
Operátor	**Popis**
`and`	a
`or`	štandartné alebo
`xor`	vylučujúce alebo
`not`	negácia

Napríklad, matematickú podmienku $a \geq 5$ môžeme v jazyku GAMS zapísať buď ako `(a ge 5)`, alebo ako `(a >= 5)`. Tento výraz bude nadobúdať logickú hodnotu `True` (numerickú 1) pre $a \geq 5$ a hodnotu `False` (0) pre $a < 5$. Teraz si ešte uvedieme tabuľku numerických hodnôt generovaných logickými operátormi:

Čísla		Numerické hodnoty daných logických operácií			
a	**b**	**a and b**	**a or b**	**a xor b**	**not a**
0	0	0	0	0	1
0	rôzne od 0	0	1	1	1
rôzne od 0	0	0	1	1	0
rôzne od 0	rôzne od 0	1	1	0	0

Numerické výrazy: Výraz majúci numerickú hodnotu 0, nadobúda logickú hodnotu `False`. Výraz rôzny od 0 zasa hodnotu `True`. Teda napríklad výraz `(a-2)` nadobúda hodnotu `False` pre $a = 2$ a `True` pre $a \neq 2$.

Zmiešané výrazy: Tvoria sa kombináciou numerických a logických výrazov, za použitia numerických a logických operátorov. Pre priority operácií platia v jazyku GAMS matematické konvencie. V nasledujúcej tabuľke uvedieme tieto priority od najvyššej po najnižšiu. V druhej tabuľke sa nachádza niekoľko príkladov numerických, logických ale aj zmiešaných výrazov spolu s ich numerickými i logickými hodnotami:

Operácia	Operátor
umocňovanie	`**`
násobenie , delenie	`* , /`
sčítanie , odčítanie	`+ , -`
numerické relačné operátory	`< , <= , = , <> , >= , >`
negácia	`not`
a	`and`
štandardné alebo , vylučujúce alebo	`or , xor`

Výraz	Numerická hodnota	Logická hodnota
`(1<2) + (3<4)`	2	`True`
`(2<1) and (3<4)`	0	`False`
`(2*1) + ((5**2)-10)`	17	`True`
`(2*1) or ((5**2)-10)`	1	`True`
`(2 and 0) + (3*2<6)`	0	`False`
`(2 and 1) + (3*2<=6)`	2	`True`

Príslušnosť k množine: Príslušnosť k množine môžeme taktiež použiť ako výraz v logickej podmienke. Ak testovaný prvok patrí do uvedenej množiny, výraz nadobúda hodnotu `True`, ak nie, tak `False`. Ilustrujeme si to na nasledujúcom príklade:

```
set i /1*10/ , podmn_i (i) /2,5,7,8/;
```

Výraz `podmn_i(i)` nadobúda logickú hodnotu `True` (numerickú 1) pre prvky `2,5,7,8` a `False` (0) pre ostatné prvky množiny `i`.

9.2 Konštrukcia podmienených príkazov pomocou operátora $

Jednou z možností, ako tvoriť podmienené príkazy v jazyku GAMS, je použitie operátora `$` v kombinácii s výrazmi nadobúdajúcimi logickú hodnotu. Operátor `$` má v jazyku GAMS viacero funkcií. Jednou z nich je aj nahradenie logickej konštrukcie *if then else*, konštrukciou `$(výraz)`, ktorá by sa dala čítať ako: „Taký, že výraz nadobúda logickú hodnotu `True`“. Na mieste slova `výraz` sa nesmie vyskytovať premenná bez prípony. Premenná s príponou je povolená. Pomocou takýchto „dolárových podmienok“ môžeme obmedziť príkaz priradenia, definíciu rovnice alebo indexovú operáciu. Príklad použitia operátora `$`:

if (a > 2) then b = 3 ⇔ `b$(a>2) = 3;`

Teraz si ukážeme, ako zapísať podmienku vnorenú do podmienky. Môžeme to spraviť dvoma ekvivalentnými spôsobmi. Prvým je forma `$(podmienka1$(podmienka2))` a druhým (prehľadnejším) spôsobom zápis `$(podmienka1 and podmienka2)`. Napríklad:

if (a > 2) then if (c > 1) then b = 5 ⇔ `b$(a>2$(c>1)) = 5;`
if (a > 2 and c > 1) then b = 5 ⇔ `b$(a>2 and c>1) = 5;`

9.2.1 Podmienené priradenie

Všetky príklady z predchádzajúcej časti sú podmienenými priradeniami. Operátor `$` sa v nich nachádza na ľavej strane priraďovacieho príkazu. Vo všeobecnosti sa môže nachádzať na oboch stranách, pričom efekt môže byť rozdielny.

$ na ľavej strane: Ak je logická podmienka splnená, teda `výraz` nadobúda hodnotu `True`, tak sa priradenie zrealizuje. V opačnom prípade zostávajú hodnoty ľavej strany nezmenené. Napríklad nech `p(i)` je parameter obsahujúci numerickú hodnotu 8 pre všetky prvky z `i`. A nech `q(i)` je parameter s ľubovoľnými hodnotami, ktoré pre tento príklad nie sú podstatné. Priradenie:

```
p(i)$(q(i) ne 0) = 5;      ⇔      p(i)$q(i) = 5;
```

Po vykonaní jedného s týchto dvoch navzájom ekvivalentných priradení, bude mať `p` hodnotu 5 pre tie prvky, pre ktoré je `q` rôzne od 0. Hodnota `p` pre ostatné prvky z `i` ostane 8. Možnosť zápisu úspornejším spôsobom vyplýva z existencie logickej hodnoty numerických výrazov v jazyku GAMS.

$ na pravej strane: Jediný rozdiel oproti výskytu na ľavej strane je v tom, že priradenie sa vykoná vždy, bez ohľadu na splnenie podmienky. Ak logický výraz nadobúda hodnotu `True`, tak sa ľavej strane priradí hodnota pravej strany a ak `False`, tak sa ľavej strane priradí 0. Priradenie:

```
p(i) = 5$(q(i) ne 0);      ⇔      p(i) = 5$q(i);
```

Rozdiel oproti predchádzajúcemu priradeniu je v tom, že pre tie prvky, pre ktoré `p` obsahovalo hodnotu 8, bude teraz obsahovať hodnotu 0. Použitím operátora `$` na pravej strane vlastne tvoríme konštrukciu *if (podmienka) then (priradenie pravej strany) else (priradenie nuly)*. Ak by sme, v prípade nesplnenia podmienky, chceli namiesto nuly priradiť nejaký výraz, dá sa to pomocou operátora `$` spraviť nasledovne, pričom `a` je už inicializovaný skalár:

```
p(i) = 5$q(i) + 2*a$(not q(i));
```

Ak pre nejaký prvok z `i` bude hodnota `q(i)` rovná `0`, tak do `p` sa priradí výraz `0+2*a`, kde `0` zodpovedá numerickej hodnote výrazu `5$q(i)` a `2*a` numerickej hodnote výrazu `2*a$(not q(i))`.

Ďalšie príklady: Príklady na podmienku príslušnosti k množine:

```
p(i)$podmn_i(i) = q(i);
```

`i` a `podmn_i` sú jednoduché množiny, kým `p(i)` a `q(i)` sú parametre. Priradenie bude vykonané len pre tie prvky z `i`, ktoré patria aj do `podmn_i`. Nasledujúci príkaz je ekvivalentný s predchádzajúcim:

```
p(podmn_i) = q(podmn_i);
```

Teraz ukážeme ako to vyzerá, keď sa v priradení vyskytuje viacero množín. Nech `i` a `j` sú jednoduché množiny, `komb_i_j` je dvojrozmerná množina obsahujúca niektoré špecifické kombinácie prvkov z `i` a z `j`, ostatné symboly sú parametre. Definované sú nasledovne:

```
sets i /i1*i9/ , j /j1*j6/ , komb_i_j(i,j) /i2.j3,i5.j6,i7.j1,i8.j3,i5.j1/;
parameters  r(j) /j1*j6 = 2/
            a(i,j) /(i1*i9).(j1*j6)=8/
            b(i,j) /(i1*i9).(j1*j6)=5/;
```

O parametroch `a` respektíve `b` sa dá uvažovať ako o maticiach typu 9×6, kde všetky ich prvky majú hodnotu 8 respektíve 5. Podmienené priradenie by mohlo vyzerať nasledovne:

```
a(i,j)$komb_i_j(i,j) = b(i,j)*r(j);
```

Hodnota parametra by sa zmenila len pre kombinácie prvkov patriace do `komb_i_j`. Bola by 10, nie 8. Toto isté by sme mohli docieliť aj nasledujúcim príkazom:

```
a(komb_i_j(i,j)) = b(komb_i_j)*r(j);
```

Upozorňujeme, že ak by sme nenapísali identifikátor `komb_i_j` na ľavej strane aj s jeho oborom, kompilátor by vyhlásil chybu, lebo množina `j` by na ľavej strane priradenia nevystupovala, kým na pravej strane vystupuje.

9.2.2 Podmienené rovnice

V tejto časti si ukážeme dva hlavné spôsoby použitia operátora `$` na vynechanie niektorých ohraničení z definície rovnice. Prvým je operátor nachádzajúci sa v algebraickom výraze, teda na pravej strane od . . a druhým je operátor nachádzajúci sa na ľavej strane od . . .

$ na pravej strane od .. : Je to analógia `$` na pravej strane priradenia. Nech `i` je jednoduchá množina, `rov` je rovnica, `x` je premenná, `p` a `q` sú parametre:

```
rov(i) ..  p(i) =g= x(i)+q(i)$podmn_i(i);
```

Parameter `q(i)` sa bude pripočítavať len pre tie prvky z `i`, ktoré sú aj z `podmn_i`. Pre ostatné prvky sa bude pripočítavať 0.

$ na ľavej strane od .. : Je to analógia `$` na ľavej strane priradenia. Napríklad:

```
rov(i)$podmn_i(i) ..  p(i) =g= x(i)+q(i);
```

V tomto prípade existujú ohraničenia len pre tie prvky z `i`, ktoré sú aj z `podmn_i`. To isté by sa dalo docieliť nasledujúcim príkazom:

```
rov(podmn_i) ..  p(podmn_i) =g= x(podmn_i)+q(podmn_i);
```

9.2.3 Podmienené indexové operácie

Operátor `$` sa môže používať aj na obmedzenie indexových operácii. Takto upravené indexové operácie sa môžu vyskytnúť tak v priradeniach, ako aj v rovniciach.

V priradení: Budeme používať identifikátory, ktoré boli už v tejto kapitole spomínané:

```
r(j) = sum(i$komb_i_j(i,j),a(i,j));   ⇔   r(j) = sum(i,a(i,j)$komb_i_j(i,j));
```

Priradenie sumy sa vykoná len pre tie prvky z `j`, ktoré sa vyskytujú v kombináciách obsiahnutých v množine `komb_i_j` (t.j. pre prvky `j1,j3,j6`). Oba príkazy priradia do `r` súčet hodnôt parametra `a` cez tie prvky z `i`, ktoré sa vyskytujú v kombináciách obsiahnutých v množine `komb_i_j`. Napríklad do `r(j1)` sa priradí `a(i5,j1)+a(i7,j1)`. Pre ostatné prvky z `j` sa do `r` priradí 0, lebo sa jedná o priradenie s operátorom `$` na pravej strane.

V rovnici: Použijeme tie isté identifikátory. Navyše bude len rovnica `rov` definovaná tentoraz cez `j` a premenná `x` definovaná v tomto prípade cez všetky kombinácie prvkov z `i` s prvkami z `j`:

```
rov(j) ..  r(j) =g= sum( i$komb_i_j(i,j) , a(i,j)*x(i,j) );
```

Suma bude vystupovať len v ohraničeniach prislúchajúcich k tým prvkom z `j`, ktoré sa vyskytujú v kombináciách obsiahnutých v množine `komb_i_j` (t.j. prislúchajúcich k prvkom `j1,j3,j6`). Pre jednotlivé ohraničenia budú v sume vystupovať len tie hodnoty `a` a `x`, ktoré prislúchajú k prvkom z `i` vyskytujúcim sa v kombináciách obsiahnutých v množine `komb_i_j`. Napríklad pre `j1` to budú prvky `i5` a `i7`. Takto sme vygenerovali nasledujúce samostatné ohraničenia (hodnoty parametrov nie sú identické s hodnotami v ich inicializácii, ale sú modifikované predchádzajúcimi príkladmi):

```
---- rov  =G=

rov(j1)..  - 10*x(i5,j1) - 10*x(i7,j1) =G= -20 ; (LHS = 0)

rov(j3)..  - 10*x(i2,j3) - 10*x(i8,j3) =G= -20 ; (LHS = 0)

rov(j6)..  - 10*x(i5,j6) =G= -10 ; (LHS = 0)
```

Treba poznamenať, že pre všetky ostatné prvky z `j` prislúchajúce ohraničenia taktiež existujú a majú tvar `r(j) =g= 0`. Pre nezáporné `r(j)` sú automaticky splnené, a preto ich GAMS vo výstupe nezobrazuje. V prípade záporného `r(j)` vznikne neprípustný model.

10 Dynamické množiny

Všetky množiny, ktoré sme doteraz používali sa nazývajú statickými množinami. To znamená, že súbor ich prvkov sme raz inicializovali a neskôr sme ho už nemenili. Ako dynamické množiny charakterizujeme tie, ktorých súbor prvkov modifikujeme pomocou príkazu priradenia. Rozdelenie na statické a dynamické množiny je len istý druh charakterizácie, nie je to rozdelenie na dva rôzne dátové typy. Keďže sa jedná o ten istý dátový typ, spôsob deklarácie a inicializácie je v oboch prípadoch identický. Preto sa bude väčšina nových informácií, uvedených v tejto kapitole, týkať modifikácie súboru prvkov množiny pomocou priradenia.

10.1 Modifikácia súboru prvkov pomocou priradenia

Aj v tomto prípade platia všeobecné pravidlá príkazu priradenia. Jediný rozdiel oproti dátovým typom obsahujúcim nejaké numerické hodnoty (`parameters`, `variables` ...) je, že na pravej strane nesmie vystupovať aritmetický výraz.

Syntax:

```
meno_množiny (cely_obor | konkretny_prvok) = yes | no;
```

Slová `yes` a `no` sú kľúčové slová, vyjadrujúce, že daný prvok alebo skupina prvkov patrí, respektíve nepatrí do uvedenej množiny. Deklaráciu dynamických množín odporúčame viazať na už existujúce statické množiny. Takto deklarovaná množina má totiž všetky vlastnosti podmnožiny, uvedené v kapitole 3, prebieha „domain checking“, a teda dynamická množina môže obsahovať len prvky patriace do množiny, na ktorú je viazaná. Samozrejme je možné konštruovať aj samostatné dynamické množiny (neviažuce sa na statickú), no vystavujeme sa tým značnému riziku formulácie zlého modelu (rôzne preklepy).

Príklad: Jednorozmerné dynamické množiny

```
sets i "množina i" /i1*i9/
     podmn1_i(i) "prvá podmnožina množiny i" /i2,i5,i7/
     podmn2_i(i) "druhá podmnožina množiny i";
podmn1_i('i1')=yes;        podmn1_i('i5')=no;           podmn2_i(i)=yes;
podmn2_i('i9')=no;         podmn2_i(podmn1_i)=no;       podmn1_i('i8')=yes;
display podmn1_i,podmn2_i;
```

Po inicializácii sa v prvej podmnožine (`podmn1_i`) nachádzali prvky `i2`, `i5` a `i7`. Následkom úvodného priradenia pribudol prvok `i1`, potom sme odstránili prvok `i5`, následne sme do druhej podmnožiny priradili všetky prvky množiny `i`, prostredníctvom ďalšieho príkazu ubudol prvok `i9`, v poradí piate priradenie vyjme z druhej podmnožiny všetky prvky nachádzajúce sa v podmnožine prvej a posledné pridá do prvej prvok `i8`. Obsah dynamickej množiny môžeme meniť tak po jednom prvku, ako aj po celých množinách, či už statických alebo dynamických, čo je vidno aj z príkladu. Pre možnosť kontroly obsahu podmnožín uvádzame výstup generovaný príkazom `display`:

```
----     10 SET podmn1_i
i1,    i2,    i7,    i8
----     10 SET podmn2_i
i3 ,    i4 ,    i5 ,    i6 ,    i8
```

Príklad: Viacrozmerná dynamická množina

```
sets i /i1*i9/ , j /j1*j9/
     komb_i_j(i,j);
komb_i_j(i,j)=yes;
komb_i_j('i2','j5')=no;
komb_i_j('i7','j1')=no;
display komb_i_j;
```

V tomto príklade sme najprv do dynamickej množiny `komb_i_j` priradili všetky možné kombinácie prvkov z `i` s prvkami z `j` a potom sme z nich dve konkrétne vylúčili, a to `i2.j5`, `i7.j1`. Všetky princípy fungovania viacrozmerných statických množín uvedené v Kapitole 3 zostávajú v platnosti aj pre dynamické viacrozmerné množiny.

10.2 Množinové operátory

Dynamické množiny môžeme v modeloch využívať identickým spôsobom ako statické. Teda napríklad aj na konštrukciu podmienených priradení, podmienených rovníc a podmienených indexových operácií pomocou operátora `$`. Vo vnútri operátora `$` sa však môžu v súvislosti s množinami vyskytovať len logické (`not`, `and`, `or`, `xor`) a množinové operátory (zjednotenie, prienik, doplnok, odčítanie). Všetky štyri množinové operátory budú teraz uvedené:

Zjednotenie: Pre zjednotenie množín sa používa operátor `+`:

```
podmn3_i(i)=podmn1_i(i) + podmn2_i(i);
```

Ale dá sa zaobísť aj bez neho:

```
podmn3_i(i)=no;       podmn3_i(podmn1_i)=yes;       podmn3_i(podmn2_i)=yes;
```

Prienik: Pre prienik sa používa operátor `*`, ale aj v tomto prípade sa vieme bez neho zaobísť:

```
podmn3_i(i)=podmn1_i(i) * podmn2_i(i);
podmn3_i(i)=yes$(podmn1_i(i) and podmn2_i(i));
```

Doplnok: Pre doplnok sa používa logický operátor `not` a opäť sa vieme zaobísť aj bez neho:

```
podmn3_i(i)=not podmn1_i(i);
podmn3_i(i)=yes;       podmn3_i(podmn1_i)=no;
```

Odčítanie: Pre odčítanie sa používa operátor - a samozrejme, že to vieme spraviť aj bez neho:

```
podmn3_i(i)=podmn1_i(i) - podmn2_i(i);
podmn3_i(i)=yes$podmn1_i(i);       podmn3_i(podmn2_i)=no;
```

11 Usporiadané množiny a možnosti ich využitia

Pri doterajšom používaní množín nezáležalo na poradí prvkov v inicializácii. Ak by sme v ľubovoľnom príklade z už uvedených zmenili poradie prvkov v inicializácii, nespôsobili by sme v modeli žiadnu zmenu. Keby sme však to isté spravili s usporiadanou množinou, mohlo by to mať na formuláciu modelu negatívny dopad. V prvej časti tejto kapitoly si objasníme pojem usporiadanej množiny z hľadiska jazyka GAMS. V druhej si vysvetlíme, na čo slúžia operátory `ord` a `card` a v poslednej časti si ukážeme ako pracovať s operátormi „Lag“ a „Lead“.

11.1 Usporiadané množiny

Usporiadaná množina je taká množina, v ktorej nám záleží na poradí jej prvkov. Napríklad množina obsahujúca nejaké časové obdobie. Iba statická množina môže byť usporiadaná, dynamická nie. Je veľmi dôležité vysvetliť ako GAMS vytvára poradie prvkov. Vždy keď v inicializácii niektorej z množín, figurujúcej v modeli, uvedieme meno prvku, ktoré sa ešte v doterajších inicializáciách nevyskytlo, tak si ho GAMS uloží na posledné miesto takzvaného „Unique element list - entry order“. Ak sme meno tohto prvku už v niektorej z predchádzajúcich inicializácií použili, tak tento prvok už v zozname „Unique element list - entry order“ figuruje, a preto ho GAMS do zoznamu už nezapíše. Takto vznikne zoznam všetkých prvkov použitých v modeli. Na základe tohto zoznamu GAMS tvorí poradie prvkov v konkrétnej množine. Ak sa toto poradie zhoduje s poradím v inicializácii, jedná sa o usporiadanú množinu a ak nie, jedná sa o neusporiadanú množinu.

Príklad: Presný význam predchádzajúcich slov si môžeme uvedomiť prostredníctvom tohto príkladu:

```
sets t1 /1993,1994,1995,1996/
     t2 /1992,1996,2000,2004/
     t3 /2002,2003,2004,2005/;
```

Poradie v zozname „Unique element list - entry order“, ktorý môžeme zahrnúť do výstupu pomocou príkazu `$onuellist` umiestneného niekde pred deklaráciou prvej množiny, by vyzeralo nasledovne:

1993 1994 1995 1996 1992 2000 2004 2002 2003 2005

Teda v našom prípade by `t1` bola usporiadaná množina a `t2,t3` by boli neusporiadané, lebo napríklad poradie prvkov `t2` z pohľadu GAMS-u je `1996`, `1992`, `2000`, `2004`, čo nie je identické s poradím v inicializácii. Ak by sme chceli z `t2` spraviť usporiadanú množinu, stačí jej deklaráciu uviesť ako prvú. To ale zároveň spôsobí, že `t1` sa stane neusporiadanou. Niekto si môže položiť otázku: „Čo ak by sme potrebovali, aby boli všetky tri usporiadané?“ Dá sa to vyriešiť napríklad takto:

```
sets t1 /1993,1994,1995,1996/
     t2 /r1992,r1996,r2000,r2004/
     t3 /r-2002,r-2003,r-2004,r-2005/;
```

11.2 Operátory ord a card

Oba operátory sa používajú v súvislosti s množinami a vracajú celočíselnú hodnotu.

Operátor ord: Vracia pozíciu prvku v množine. Môžeme ho aplikovať iba na jednorozmernú (jednoduchú), statickú, usporiadanú množinu. Ak by dotyčná množina nespĺňala niektorú z týchto troch podmienok, kompilátor by vyhlásil chybu. Ako pracovať s operátorom `ord`, si ukážeme na nasledujúcom príklade. Zároveň uvádzame aj hodnoty parametra po vykonaní priradenia:

```
parameter pozicia(t1);
pozicia(t1)=ord(t1);          ⇒      1993  1,  1994  2,  1995  3,  1996  4
```

Operátor card: Vracia počet prvkov množiny. Môžeme ho aplikovať na ľubovoľnú množinu:

```
parameter pocet_prvkov;
pocet_prvkov=card(t2);        ⇒      4
```

11.3 Operátory „Lag“ a „Lead“

V modeloch sa často vyskytuje závislosť hodnoty identifikátora prislúchajúcej k nasledujúcemu prvku množiny indexov od hodnoty tohto identifikátora prislúchajúcej k aktuálnemu prvku tejto množiny indexov (`x(t+1)=2*x(t)`). Operátory „Lag“ a „Lead“ slúžia na zakomponovanie takejto závislosti do modelu formulovaného v jazyku GAMS. Množiny, na ktoré sú operátory aplikované, musia byť usporiadané. Tieto operátory môžu byť buď lineárne (`+`, `-`) alebo cyklické (`++`, `--`). Rozdiel medzi nimi je v tom, ako sa vysporiadavajú s okrajovými hodnotami. V tejto časti si vysvetlíme ich odlišnosti a zároveň ich použitie v priradeniach a rovniciach.

11.3.1 Operátory „Lag“ a „Lead“ v priradeniach

Môžeme ich používať na obidvoch stranách priradenia. V prípade lineárneho operátora výsledok závisí od strany použitia, kým v prípade cyklického nie. Cyklický operátor sa totiž s krajnými prvkami vysporiada pomocou vytvorenia „obojsmerného cyklu“ z dotyčnej množiny, v ktorom po poslednom prvku nasleduje prvý a prvému predchádza posledný. Napríklad mesiace v roku, po decembri nasleduje január a januáru predchádza december. Vďaka tomuto princípu sa v priradení nevyskytnú žiadne neexistujúce hodnoty a vykoná sa pre každý prvok z dotyčnej množiny (ak sa v ňom nenachádzajú nejaké iné obmedzenia (`$`)).

Lineárne „Lag“ a „Lead“ operátory na pravej strane priradenia:

```
set t /r2000*r2004/;
parameters a(t),b(t);
a(t)=1999+ord(t);
b(t)=-1;  b(t)=a(t-1);
```

Výsledok takéhoto priradenia do parametrov `a` a `b` je nasledovný:

```
----     24 PARAMETER a
r2000 2000,    r2001 2001,    r2002 2002,    r2003 2003,    r2004 2004
----     24 PARAMETER b
r2001 2000,    r2002 2001,    r2003 2002,    r2004 2003
```

Jediná zaujímavá otázka je, ako sa kompilátor vysporiadal s okrajovými hodnotami. Teda, čo sa stalo s hodnotou `b('r2000')`. Mala sa jej priradiť hodnota `a('r2000'-1)`, ktorá ale neexistuje. To spôsobí, že pravá strana bude mať logickú hodnotu `False`, a teda numerickú hodnotu 0, ktorá sa priradí do `b('r2000')`. Preto táto hodnota parametra `b` nevystupuje vo výstupe generovanom príkazom `display`.

Lineárne „Lag“ a „Lead“ operátory na ľavej strane priradenia:
Ak pridáme na koniec predchádzajúceho príkladu príkaz:

```
b(t)=-1;  b(t+1)=a(t);
```

tak zistíme, že všetky hodnoty parametra `b` budú rovnaké ako vo výstupe z predchádzajúceho príkladu, až na `b('r2000')`. Hodnota `b('r2000')` zostane tento krát -1, pretože prvý prvok, pre ktorý sa priradenie vykoná bude `'r2000'+1`, teda `r2001`. Zároveň posledný prvok `'r2004'+1` neexistuje, a teda priradenie sa nevykoná:

```
----     27 PARAMETER b
r2000   -1,    r2001 2000,    r2002 2001,    r2003 2002,    r2004 2003
```

Napravo od „Lag“ alebo „Lead“ operátora môže byť vo všeobecnosti ľubovolný výraz dávajúci celočíselný výsledok. Teda napríklad namiesto `b(t+1)` by sme mohli napísať `b(t+(1/2+sin(pi/6)))` a dostali by sme rovnaký výsledok, keďže výraz uvedený napravo od operátora `+` sa rovná 1.

Cyklické „Lag“ a „Lead“ operátory v priradení:
Ako sme už spomínali, v tomto prípade sa výsledky priradenia pre obe možné pozície operátora zhodujú. Dokumentuje to nasledujúci príklad:

```
set stvrtrok /s1*s4/;
parameters c(stvrtrok) /s1=10,s2=12,s3=9,s4=15/, d(stvrtrok);
d(stvrtrok)=c(stvrtrok--2);
d(stvrtrok++2)=c(stvrtrok);
```

Pri cyklických operátoroch (++, --) sa nevyskytujú v priradeniach neexistujúce prvky, ako to bolo v prípade lineárnych. Priradenie sa teda uskutoční pre všetky prvky, pre ktoré je definované. Zoberme si napríklad z druhého priradenia hodnotu `d('s4'++2)`. Táto hodnota existuje a je to `d('s2')`. Ak by bol operátor lineárny, neexistovala by a priradenie by sa neuskutočnilo. Ešte uvedieme výsledky oboch priradení:

```
----     32 PARAMETER d
s1  9,    s2 15,    s3 10,    s4 12
----     34 PARAMETER d
s1  9,    s2 15,    s3 10,    s4 12
```

11.3.2 Operátory „Lag“ a „Lead“ v rovniciach

Operátory sa môžu nachádzať na oboch stranách výrazu . . . V prípade výskytu neexistujúcej hodnoty, t. j. logickej hodnoty `False`, je tu istá analógia s podmienenými rovnicami z Kapitoly 9. Výrazy na pravej strane „Lag“ alebo „Lead“ operátora musia byť exogénne a dávať celočíselnú hodnotu.

Lineárne „Lag“ a „Lead“ operátory v rovniciach:
Uvedieme si naraz použitie na ľavej i pravej strane výrazu . . a porovnáme výsledné ohraničenia:

```
set t /r2000*r2004/;
variables x(t),y(t);
equation rov(t),rov1(t);
rov(t+1) ..  x(t+1) =e= x(t)  +y(t);
rov1(t)  ..    x(t) =e= x(t-1)+y(t-1);
```

Prvé existujúce ohraničenie z bloku `rov` bude prislúchať k prvku `'r2000'+1`, a teda k prvku `r2001`. V bloku `rov1` bude pre prvok `r2000` existovať ohraničenie, no keďže hodnoty `x('r2000'-1)` a `y('r2000'-1)` neexistujú, oba výrazy nadobúdajú logickú hodnotu `False`, a teda numerickú hodnotu 0. Z uvedeného vyplývajú nasledujúce samostatné ohraničenia, pričom uvádzame z každého bloku len prvé dve, keďže okrem rovnice prislúchajúcej k prvku `r2000` sa oba bloky zhodujú:

```
---- rov  =E=
rov(r2001)..  - x(r2000) + x(r2001) - y(r2000) =E= 0 ;
rov(r2002)..  - x(r2001) + x(r2002) - y(r2001) =E= 0 ;
REMAINING 2 ENTRIES SKIPPED

---- rov1  =E=
rov1(r2000)..  x(r2000) =E= 0 ; (LHS = 0)
rov1(r2001)..  - x(r2000) + x(r2001) - y(r2000) =E= 0 ;
REMAINING 3 ENTRIES SKIPPED
```

.

Cyklické „Lag“ a „Lead“ operátory v rovniciach:
Tak ako v priradení ani v rovniciach nezáleží na umiestnení operátora vzhľadom na výraz . . .
Uvažujme takýto príklad:

```
set t /r2000*r2004/;
variables x(t),y(t);
equation rov2(t),rov3(t);
rov2(t++1) ..  x(t++1) =e= x(t)   +y(t);
rov3(t)    ..     x(t) =e= x(t--1)+y(t--1);
```

Nami definované identifikátory dátového typu rovnice generujú nasledovné samostatné ohraničenia:

```
---- rov2  =E=
rov2(r2000)..  x(r2000) - x(r2004) - y(r2004) =E= 0 ;
rov2(r2001)..  - x(r2000) + x(r2001) - y(r2000) =E= 0 ;
REMAINING 3 ENTRIES SKIPPED

---- rov3  =E=
rov3(r2000)..  x(r2000) - x(r2004) - y(r2004) =E= 0 ;
rov3(r2001)..  - x(r2000) + x(r2001) - y(r2000) =E= 0 ;
REMAINING 3 ENTRIES SKIPPED
```

Ako vidíme, pri použití cyklických operátorov sa ohraničenia generované prvou definíciou rovnice zhodujú s ohraničeniami generovanými tou druhou. Z praktického hľadiska by sa tu žiadal skôr príklad obsahujúci časové obdobie, ktoré sa môže viac krát opakovať, ale naším cieľom bolo poskytnúť možnosť porovnania s lineárnymi operátormi, pri ktorých sme použili tú istú štruktúru.

12 Riadiace štruktúry (loop, for, while, if-else)

V tejto kapitole sa budeme zaoberať príkazmi, pomocou ktorých môžeme riadiť beh programu. V jazyku GAMS sú to nasledujúce štyri príkazy:

```
loop       for       while       if-else
```

V ich tele sa nesmie nachádzať deklarácia žiadneho dátového typu, ale ani definícia rovnice.

12.1 Príkaz loop

Príkaz `loop` nám umožňuje vykonať jeden alebo viacero príkazov pre každý prvok množiny.

Syntax:

```
loop (meno_množiny [$podmienka],
  prikaz {; prikaz}
);
```

Príkaz `loop` môže prechádzať aj cez viacero množín. V takom prípade by sme namiesto výrazu `meno_mnoziny` písali výraz `(meno_mnoziny,meno_mnoziny,...)`. Riadiaca množina indexov môže byť dynamická, ale upozorňujeme, že ju nemôžeme v tele príkazu modifikovať. Poradie v akom príkaz prebieha po jednotlivých prvkoch sa zhoduje s poradím prvkov v zozname „unique elements list".

Príklad: Tento príklad slúži na ilustráciu príkazu `loop`. Ako už vieme z Kapitoly 11, rovnaký výsledok dokážeme docieliť aj bez použitia tohto príkazu. Uvádzame formuláciu i výsledok:

```
set i /i1*i9/;
parameters a(i),b(i),c(i);
b(i)=round(uniform(1,10));   c(i)=round(uniform(1,10));
loop (i , a(i+1)=b(i)+c(i-1));

----      6 PARAMETER b
i1 3.000,    i2 9.000,    i3 6.000,    i4 4.000,    i5 4.000,    i6 3.000
i7 4.000,    i8 9.000,    i9 2.000

----      6 PARAMETER c
i1  6.000,    i2 10.000,    i3  6.000,    i4 10.000,    i5  8.000,    i6  2.000
i7  7.000,    i8  2.000,    i9  3.000

----      6 PARAMETER a
i2  3.000,    i3 15.000,    i4 16.000,    i5 10.000,    i6 14.000,    i7 11.000
i8  6.000,    i9 16.000
```

12.2 Príkaz for

Umožňuje opakovať viac krát blok príkazov, a to pre rôzne skalárne hodnoty.

Syntax:

```
for (meno_parametra=start to | downto end [by inkrementacia],
  prikaz {; prikaz}
);
```

Na rozdiel od príkazu `loop`, príkaz `for` prechádza cez skalárne hodnoty parametra a nie cez množiny. Slovo `start` vyjadruje počiatočnú hodnotu. Kľúčové slová `to` a `downto` udávajú smer pohybu hodnoty parametra, meniť sa bude o hodnotu uvedenú na mieste slova `inkrementacia` (ak ju neuvedieme, prednastavená hodnota = 1) pokiaľ bude hodnota parametra menšia alebo rovná hodnote `end`. Všetky tieto hodnoty môžu byť navyše ľubovoľné reálne čísla, len `inkrementacia` musí byť kladná.

Príklad: V nasledujúcich príkladoch je identifikátor `j` dátového typu parameter a je skalárny:

Príkaz	`for (j=-2.2 to 1.3 , display j);`	`for (j=-2.2 to 1.3 by 1.5 , display j);`

Výstup	`----  9 PARAMETER j = -2.200`
	`----  9 PARAMETER j = -1.200`
	`----  9 PARAMETER j = -0.200`
	`----  9 PARAMETER j = 0.800`
		`----  10 PARAMETER j = -2.200`
		`----  10 PARAMETER j = -0.700`
		`----  10 PARAMETER j = 0.800`

12.3 Príkaz while

Slúži na opakovanie bloku príkazov, kým je splnená daná podmienka.

Syntax:

```
while (podmienka ,
  prikaz {; prikaz}
);
```

Príklad: V tomto príklade sú identifikátory `j` a `k` skalárne parametre, už inicializované na `5` a `-3`:

```
while (k<1 , display j,k; j=j+1; k=k+1);

----     14 PARAMETER j                   =        5.000
            PARAMETER k                   =       -3.000

----     14 PARAMETER j                   =        6.000
            PARAMETER k                   =       -2.000

----     14 PARAMETER j                   =        7.000
            PARAMETER k                   =       -1.000

----     14 PARAMETER j                   =        8.000
            PARAMETER k                   =        0.000
```

12.4 Príkaz if-elseif-else

Ak potrebujeme, aby sa blok príkazov vykonal len ak je splnená istá podmienka, tak to môžeme docieliť pomocou príkazu `if`.

Syntax:

```
if (podmienka , prikazy ;
  { elseif condition , prikazy; }
    [ else prikazy; ]
);
```

Príklad: Komplexný príklad na ilustráciu celej konštrukcie príkazu:

```
if (f<=0,
  p(i)=-1;
  q(j)=-1;
  elseif ((f>0) and (f<1)),
    p(i)=p(i)**2;
    q(j)=q(j)**2;
    else
      p(i)=p(i)**3;
      q(j)=q(j)**3;
);
```

Ak `f` je menšie alebo rovné 0, tak `p` aj `q` budú mať pre všetky prvky hodnotu -1. Ak je `f` medzi 0 a 1, tak sa ich pôvodné hodnoty umocnia na druhú a ak je `f` väčšie alebo rovné 1, tak sa umocnia na tretiu.

13 Výstup

Výstup programu napísaného v jazyku GAMS nám poskytuje množstvo pomôcok na overenie správnosti formulácie modelu. Inak povedané, či model vygenerovaný kompilátorom sa zhoduje s našou predstavou. V tejto kapitole si postupne vysvetlíme všetky komponenty, ktoré sa môžu nachádzať vo výstupe a zároveň si povieme ako zariadiť, aby sa tam nevyskytovali, keď ich nepotrebujeme. Celá kapitola sa bude vzťahovať na nasledujúci vstupný súbor obsahujúci jednoduchý ilustračný model.

13.1 Ilustračný model

```
$title Rámy na obrazy
$ontext
Tu by sa mohlo nachádzať zadanie, ale my sme ho neuviedli, aby zbytočne
nezaberalo miesto. Je už totiž uvedené na inom mieste v tejto práci.
$offtext
$onsymxref onsymlist onuelxref onuellist
option limrow=2 , limcol=2;

sets r "typy rámov na obrazy" /R1,R2,R3,R4/
     vf "jednotlivé výrobné faktory" / praca "v hodinách"
                                       kov   "v kilogramoch"
                                       sklo  "v kilogramoch" /;

table mn(vf,r) "množstvo výrobného faktoru potrebného na daný typ rámu"
                  R1   R2   R3   R4
          praca    2    1    3    2
          kov      4    2    1    2
          sklo     6    2    1    2 ;

parameters z(r) "zisk z jedného predaného rámu daného typu"
                / R1=6 , R2=2 , R3=4 , R4=3 /
           d(r) "dopyt po jednotlivých typoch rámov"
                / R1=1000 , R2=2000 , R3=500 , R4=1000 /
           dm(vf) "disponibilné množstvo jednotlivých výrobných faktorov"
                / praca=4000 , kov=6000 , sklo=10000 /;
* koniec zadania

variables cz  "celkový zisk"
          x(r) "počet vyrobených rámov daného typu";
positive variable x;

equations uf "účelová funkcia"
          zdroje(vf) "ohraničenia na zdroje"
          dopyt(r) "dopytové ohraničenia pre jednotlivé typy rámov";
uf ..          cz =e= sum(r,z(r)*x(r));
zdroje(vf) ..         sum(r,mn(vf,r)*x(r)) =l= dm(vf);
dopyt(r) ..                          x(r)  =l= d(r);

model ramy / all /;
solve ramy maximizing cz using lp;
```

13.2 Kompilačný výstup

Je to výstup generovaný kompilátorom počas počiatočnej kontroly programu. Pozostáva z dvoch alebo v prípade výskytu kompilačných chýb, z troch častí, z takzvaného „echo print“, z vysvetľujúceho textu kompilačných chýb a z rôznych máp pamäte („maps“). V tejto časti si ukážeme „echo print“ a „maps“. Chybám sa budeme venovať neskôr, v samostatnej časti.

13.2.1 „Echo print“

Výstup vždy začína komponentom „Echo print“. Je to v podstate kópia zdrojového kódu s očíslovanými riadkami. Podľa prednastaveného formátu sa príkazy začínajúce znakom `$`, napísaným na prvom mieste v riadku, nezobrazujú. Výnimkou je situácia, ak by v týchto príkazoch nastala chyba. „Echo print“ sa vypína pomocou príkazu `$offlisting`. V našom modeli by vyzeral nasledovne:

```
   Tu by sa mohlo nachádzať zadanie, ale my sme ho neuviedli, aby zbytočne
   nezaberalo miesto. Je už totiž uvedené na inom mieste v tejto práci.
 7 option limrow=2 , limcol=2;
 8
 9 sets r "typy rámov na obrazy" /R1,R2,R3,R4/
10      vf "jednotlivé výrobné faktory" / praca "v hodinách"
11                                        kov   "v kilogramoch"
12                                        sklo  "v kilogramoch" /;
13
14 table mn(vf,r) "množstvo výrobného faktoru potrebného na daný typ rámu"
15                    R1   R2   R3   R4
16           praca     2    1    3    2
17           kov       4    2    1    2
18           sklo      6    2    1    2 ;
19
20 parameters z(r) "zisk z jedného predaného rámu daného typu"
21                 / R1=6 , R2=2 , R3=4 , R4=3 /
22             d(r) "dopyt po jednotlivých typoch rámov"
23                 / R1=1000 , R2=2000 , R3=500 , R4=1000 /
24             dm(vf) "disponibilné množstvo jednotlivých výrobných faktorov"
25                 / praca=4000 , kov=6000 , sklo=10000 /;
26 * koniec zadania
27
28 variables cz  "celkový zisk"
29           x(r) "počet vyrobených rámov daného typu";
30 positive variable x;
31
32 equations uf "účelová funkcia"
33           zdroje(vf) "ohraničenia na zdroje"
34           dopyt(r) "dopytové ohraničenia pre jednotlivé typy rámov";
35 uf ..          cz =e= sum(r,z(r)*x(r));
36 zdroje(vf) ..         sum(r,mn(vf,r)*x(r)) =l= dm(vf);
37 dopyt(r) ..                          x(r)  =l= d(r);
38
39 model ramy / all /;
40 solve ramy maximizing cz using lp;
```

13.2.2 „Symbol reference map“

Zobrazuje v abecednom poradí všetky identifikátory nachádzajúce sa vo vstupnom súbore. Každý z nich je nasledovaný prislúchajúcim dátovým typom a všetkými druhmi výskytov v zdrojovom kóde. Každý takýto výskyt je charakterizovaný číslom riadku, ktoré zodpovedá číslovaniu z komponentu „echo print“. Výraz `2*36` znamená, že daný symbol sa vyskytuje v riadku `36` dva krát. Komponent „symbol reference map“ môžeme aktivovať pomocou príkazu `$onsymxref`. Vyzerá takto:

```
Symbol Listing
SYMBOL      TYPE   REFERENCES

cz           VAR   declared       28 impl-asn       40      ref       35
                         40
d            PARAM declared       22  defined       23      ref       37
dm           PARAM declared       24  defined       25      ref       36
dopyt        EQU   declared       34  defined       37 impl-asn       40
                           ref    39
mn           PARAM declared       14  defined       14      ref       36
r            SET   declared        9  defined        9      ref       14
                          20      22       29       34     2*35     2*36
                        2*37  control       35       36       37
ramy         MODEL declared       39  defined       39 impl-asn       40
                           ref    40
uf           EQU   declared       32  defined       35 impl-asn       40
                           ref    39
vf           SET   declared       10  defined       10      ref       14
                          24      33     2*36  control       36
x            VAR   declared       29 impl-asn       40      ref       30
                          35      36       37
z            PARAM declared       20  defined       21      ref       35
zdroje       EQU   declared       33  defined       36 impl-asn       40
                           ref    39
```

Zoberme si napríklad symbol `x`. Z diagramu sa dá vyčítať, že je typu premenná (`VAR`), bol deklarovaný v riadku `29` (`declared`), hodnota mu bola implicitne priradená v riadku `40` prostredníctvom príkazu `solve` (`impl-asn`) a inak zmienený bol v riadkoch `30,35,36,37` (`ref`). Teraz uvedieme tabuľku všetkých skratiek a typov výskytu, ktoré sa môžu v tomto komponente nachádzať a ich popis:

Skratka	Dátový typ
VAR	premenná
PARAM	parameter
EQU	rovnica
SET	množina
MODEL	model

Typ výskytu	Popis
DECLARED	deklarácia identifikátora
DEFINED	definícia identifikátora
ASSIGNED	výskyt identifikátora na ľavej strane priradenia
IMPL-ASN	implicitné priradenie prostredníctvom príkazu `solve`
CONTROL	použitie množiny ako riadiaceho indexu
REF	iné výskyty identifikátora

13.2.3 „Symbol listing map“

V tomto komponente sú všetky identifikátory vystupujúce vo vstupnom súbore zoskupené podľa dátového typu a pri každom sa navyše nachádza vysvetľujúci text. Komponent „Symbol listing map“ môžeme aktivovať pomocou príkazu `$onsymlist`. Pre náš model by vyzeral nasledovne:

```
SETS
r             typy rámov na obrazy
vf            jednotlivé výrobné faktory

PARAMETERS
d             dopyt po jednotlivých typoch rámov
dm            disponibilné množstvo jednotlivých výrobných faktorov
mn            množstvo výrobného faktoru potrebného na daný typ rámu
z             zisk z jedného predaného rámu daného typu

VARIABLES
cz            celkový zisk
x             počet vyrobených rámov daného typu

EQUATIONS
dopyt         dopytové ohraničenia pre jednotlivé typy rámov
uf            účelová funkcia
zdroje        ohraničenia na zdroje

MODELS
ramy
```

13.2.4 „Unique element listing“

Pozostáva z dvoch častí. Prvou je zoznam jednotlivých prvkov vyskytujúcich sa vo vstupnom súbore, zoradených najskôr podľa zadávania v rámci zdrojového kódu a potom podľa abecedy. Druhou je zoznam výskytov týchto prvkov. Prvú môžeme aktivovať pomocou príkazu `$onuellist` a druhú pomocou `$onuelxref`:

```
Unique Element Listing


Unique Elements in Entry Order

1  R1          R2          R3          R4          praca       kov
7  sklo

Unique Elements in Sorted Order

1  kov         praca       R1          R2          R3          R4
7  sklo

ELEMENT   REFERENCES

kov       declared       11      ref       17       25
praca     declared       10      ref       16       25
R1        declared        9      ref       15       21       23
R2        declared        9      ref       15       21       23
R3        declared        9      ref       15       21       23
R4        declared        9      ref       15       21       23
sklo      declared       12      ref       18       25
```

13.2.5 Užitočné „dolárové“ príkazy

Ako sme už spomínali, pomocou operátora `$` vieme do modelu zakomponovať rôzne výnimky. Na tomto mieste je však jeho funkcia diametrálne odlišná. V tomto prípade sa jedná o rôzne modifikácie nastavení kompilátoru. Pripomíname, že znak `$` sa musí nachádzať na prvom mieste v riadku, aby mal význam modifikátora nastavení kompilátoru. V tejto časti vymenujeme a popíšeme najdôležitejšie „dolárové“ príkazy, ktoré možno používať v súvislosti s kompilačným výstupom:

Príkaz	Popis
`$offlisting`	vypne „echo print“
`$onlisting`	zapne „echo print“
`$offsymxref`	vypne „Symbol reference map“
`$onsymxref`	zapne „Symbol reference map“
`$offsymlist`	vypne „Symbol listing map“
`$onsymlist`	zapne „Symbol listing map“
`$offuelxref`	vypne „Unique element reference map“
`$onuelxref`	zapne „Unique element reference map“
`$offuellist`	vypne „Unique element listing map“
`$onuellist`	zapne „Unique element listing map“
`$title 'text'`	spôsobí, že daný text bude figurovať v hlavičke každej strany výstupného súboru

13.3 Výstup generovaný príkazom solve

13.3.1 „Equation listing“

Tento komponent výstupu zobrazuje jednotlivé rovnice so všetkými premennými na ľavej strane a všetkými konštantami, reprezentovanými jedným výsledným číslom, na strane pravej. Takto si môžeme veľmi jednoducho skontrolovať, či nami naprogramovaný model je v súlade s matematickou formuláciou. Prednastavene zobrazuje nanajvýš prvé 3 samostatné rovnice z každého bloku rovníc,

teda z každého identifikátora dátového typu rovnice. V našom modeli sme tento počet zmenili na dve pomocou príkazu `option limrow=2;` . Teda výstup by vyzeral nasledovne:

```
Equation Listing    SOLVE ramy Using LP From line 40


---- uf  =E=  účelová funkcia

uf..  cz - 6*x(R1) - 2*x(R2) - 4*x(R3) - 3*x(R4) =E= 0 ; (LHS = 0)


---- zdroje  =L=  ohraničenia na zdroje

zdroje(praca)..  2*x(R1) + x(R2) + 3*x(R3) + 2*x(R4) =L= 4000 ; (LHS = 0)

zdroje(kov)..  4*x(R1) + 2*x(R2) + x(R3) + 2*x(R4) =L= 6000 ; (LHS = 0)

REMAINING ENTRY SKIPPED


---- dopyt  =L=  dopytové ohraničenia pre jednotlivé typy rámov

dopyt(R1)..  x(R1) =L= 1000 ; (LHS = 0)

dopyt(R2)..  x(R2) =L= 2000 ; (LHS = 0)

REMAINING 2 ENTRIES SKIPPED
```

V prípade, že sme použili kľúčové slovo `all`, poradie blokov rovníc je identické s poradím ich deklarácie. Inak je determinované poradím v inicializácii modelu. Poradie ohraničení v rámci jedného bloku závisí od poradia prvkov v „Unique element list - entry order“.

Ak by bola niektorá rovnica nelineárna, zobrazenie komponentu „Equation listing“ by vyzeralo inak. Na mieste koeficientu danej premennej by sa v zátvorkách nachádzala vyčíslená hodnota prvej parciálnej derivácie rovnice podľa tejto premennej. Pre výpočet by sa použili aktuálne hodnoty (`.l`) jednotlivých premenných.

Príklad: Nasledujúci príklad by mal objasniť ako sa zobrazujú nelineárne rovnice:

```
variables x,y;
equation Eq1;
eq1 ..   4*sqr(x)*power(y,3)+9/x =e= 1;       x.l=3;       y.l=2;
```

Najskôr sa rovnica parciálne zderivuje podľa konkrétnej premennej (napríklad podľa `x` dostaneme výraz `8*x*power(y,3)+(-1)*(9/sqr(x))`), potom sa do výrazu dosadia aktuálne hodnoty premenných (za `x` sa dosadí `x.l`, teda `3` a za `y.l`, teda `2`) a vyčísli sa (`(191)`). Výstup by bol takýto:

```
Equation Listing    SOLVE eq Using NLP From line 8

---- Eq1  =E=
Eq1..  (191)*x + (432)*y =E= 1 ; (LHS = 291, INFES = 290 ***)
```

`LHS` vyjadruje aktuálnu hodnotu ľavej strany rovnice, `***` je upozornenie, že pre počiatočné hodnoty je rovnica neprípustná a `INFES` udáva „veľkosť neprípustnosti“.

13.3.2 „Column listing“

„Column listing“ zobrazuje jednotlivé premenné a k nim prislúchajúce koeficienty v rovniciach. Prednastavený počet samostatných premenných je 3. Opäť ho môžeme zmeniť, a to pomocou príkazu `option limcol=cislo;` , kde `cislo` vyjadruje tento počet. V našom modeli je `cislo` rovné 2:

```
Column Listing      SOLVE ramy Using LP From line 40

---- cz  celkový zisk
cz
                (.LO, .L, .UP = -INF, 0, +INF)
        1       uf

---- x  počet vyrobených rámov daného typu
x(R1)
                (.LO, .L, .UP = 0, 0, +INF)
       -6       uf
        2       zdroje(praca)
        4       zdroje(kov)
        6       zdroje(sklo)
        1       dopyt(R1)
x(R2)
                (.LO, .L, .UP = 0, 0, +INF)
       -2       uf
        1       zdroje(praca)
        2       zdroje(kov)
        2       zdroje(sklo)
        1       dopyt(R2)

REMAINING 2 ENTRIES SKIPPED
```

Poradie premenných je identické s poradím v deklarácii.

13.3.3 „Model statistics“

Je to posledný z komponentov výstupu zobrazujúcich model ešte pred začatím riešenia. Poskytuje informácie o veľkosti a nelineárnej zložitosti modelu. V našom príklade vyzerá nasledovne:

```
Model Statistics    SOLVE ramy Using LP From line 40

MODEL STATISTICS

BLOCKS OF EQUATIONS       3     SINGLE EQUATIONS        8
BLOCKS OF VARIABLES       2     SINGLE VARIABLES        5
NON ZERO ELEMENTS        21

GENERATION TIME      =        0.000 SECONDS    3.9 Mb  WIN216-141 Jan 24, 2005
```

`BLOCKS OF EQUATIONS` a `BLOCKS OF VARIABLES` vyjadrujú počet blokov rovníc a premenných, a teda počet identifikátorov daného dátového typu vyskytujúcich sa v naprogramovanom modeli. `SINGLE EQUATIONS` poskytuje informáciu o počte samostatných rovníc a `SINGLE VARIABLES` o počte samostatných premenných. Číslo prislúchajúce k `NON ZERO ELEMENTS` značí počet nenulových koeficientov v matici problému. Ak by sa jednalo o nelineárny model pribudli by štyri ukazovatele charakterizujúce zložitosť nelinearity, a to `NON LINEAR N-Z` (počet nelineárnych vstupov v matici problému), `DERIVATIVE POOL`, `CONSTANT POOL`, `CODE LENGTH`.

13.3.4 „Solve summary“

Tento komponent obsahuje informácie o procese riešenia modelu. Je rozdelený na dve časti. Prvú majú všetky solvre spoločnú a tá druhá je pre každý z nich špecifická. Prvá časť z nášho modelu:

```
S O L V E      S U M M A R Y

     MODEL   ramy                OBJECTIVE  cz
     TYPE    LP                  DIRECTION  MAXIMIZE
     SOLVER  CPLEX               FROM LINE  40

**** SOLVER STATUS     1 NORMAL COMPLETION
**** MODEL STATUS      1 OPTIMAL
**** OBJECTIVE VALUE             9200.0000

 RESOURCE USAGE, LIMIT          0.015      1000.000
 ITERATION COUNT, LIMIT         3         10000
```

Prečítať by sa dala takto: Riešime model ramy, ktorý je typu LP, pomocou solvru CPLEX, kde hodnotu účelovej funkcie (OBJECTIVE) vyjadruje premenná cz, ktorú maximalizujeme (MAXIMIZE). Príkaz solve prislúchajúci k tomuto riešeniu sa nachádza na riadku 40. Po vyriešení modelu, v 3 iteráciách (ITERATION COUNT) za 0.015 sekundy (RESOURCE USAGE), účelová funkcia nadobúda hodnotu 9200 (OBJECTIVE VALUE), pričom limit na počet iterácií bol 1000 a na čas trvania riešiaceho procesu 1000 sekúnd. Tieto limity sa dajú zmeniť pomocou príkazu option umiestneného niekde pred príkazom solve. Pre počet iterácií (xx) je to option iterlim=xx; a pre čas trvania riešiaceho procesu (yy) option reslim=yy;. Ak by boli limity prekročené, solver preruší riešenie. Ešte nám zostáva povedať, čo znamená SOLVER STATUS a MODEL STATUS. Za týmto účelom uvádzame nasledujúce dve tabuľky, ktoré zobrazujú, aké rôzne stavy môžu nastať:

SOLVER STATUS	Popis
1 NORMAL COMPLETION	solver skončil normálnym spôsobom, teda proces hľadania riešenia nebol prerušený
2 ITERATION INTERRUPT	proces bol prerušený z dôvodu prekročenia limitu iterácií
3 RESOURCE INTERRUPT	proces bol prerušený z dôvodu prekročenia limitu na jeho trvanie
4 TERMINATED BY SOLVER	z nejakého dôvodu solver nebol schopný pokračovať v procese
5 EVALUATION ERROR LIMIT	príliš veľa nedefinovaných operácií v nelineárnych výrazoch
6 UNKNOWN ERROR	všetky vyjadrujú istý druh neočakávaného zlyhania GAMS-u, solvra alebo ich vzájomnej komunikácie
8 ERROR PREPROCESSOR ERROR	
9 ERROR SETUP FAILURE	
10 ERROR SOLVER FAILURE	
11 ERROR INTERNAL SOLVER ERROR	
12 ERROR POST-PROCESSOR ERROR	
13 ERROR SYSTEM FAILURE	

MODEL STATUS	Popis
1 OPTIMAL	riešenie je optimálne (LP, RMIP)
2 LOCALLY OPTIMAL	riešenie je lokálne optimálne (NLP)
3 UNBOUNDED	riešenie je neohraničené (LP)
4 INFEASIBLE	neprípustnosť modelu (LP)
5 LOCALLY INFEASIBLE	lokálna neprípustnosť (NLP)
6 INTERMEDIATE INFEASIBLE	z istých dôvodov (napr.: presiahnutie limít) bol riešiaci proces prerušený a v tom momente bolo riešenie neprípustné
7 INTERMEDIATE NONOPTIMAL	to isté, ale riešenie bolo prípustné a nie optimálne
8 INTEGER SOLUTION	riešenie je celočíselné (MIP)
9 INTERMEDIATE NON-INTEGER	prerušený proces a riešenie nebolo celočíselné (MIP)
10 INTEGER INFEASIBLE	neexistuje celočíselné riešenie daného problému (MIP)
12 ERROR UNKNOWN	v oboch prípadoch nastala nejaká chyba a v dôsledku nej neexistuje žiadne riešenie daného problému
13 ERROR NO SOLUTION	

Druhou časťou je komponent „Solver report“. V našom modeli sme použili solver CPLEX a ten poskytuje takýto výstup:

```
GAMS/Cplex    Jan 26, 2005 WIN.CP.NA 21.6 027.030.041.VIS For Cplex 9.0
Cplex 9.0.2, GAMS Link 26

Optimal solution found.
Objective :        9200.000000
```

Táto časť výstupu sa líši pre jednotlivé solvre, a preto sa ňou nebudeme ďalej zaoberať.

13.3.5 „Solution listing“

Komponent „Solution listing“ nám poskytuje informácie o riešeniach modelu získaných od solvra. Môžeme ho vypnúť pomocou príkazu `option solprint=off;`. Pripomenieme, že po vyriešení modelu solver aktualizuje hodnoty identifikátorov pre prípony `.l`, `.m` a ak proces riešenia prebehol v poriadku, tak sú tieto hodnoty aj optimálnymi hodnotami. Pre náš model by komponent „Solution listing“ vyzeral nasledovne, pričom stĺpec `LOWER` resp. `UPPER` vyjadruje hodnotu identifikátora prislúchajúcu k prípone `.lo` resp. `.up`, stĺpec `LEVEL` k prípone `.l` a stĺpec `MARGINAL` k prípone `.m`. Znak . reprezentuje hodnotu 0:

```
                           LOWER     LEVEL     UPPER    MARGINAL

---- EQU uf                  .         .         .        1.000

  uf  účelová funkcia

---- EQU zdroje  ohraničenia na zdroje

         LOWER     LEVEL     UPPER    MARGINAL

praca     -INF   4000.000  4000.000     1.200
kov       -INF   6000.000  6000.000     0.400
sklo      -INF   8000.000 10000.000      .

---- EQU dopyt  dopytové ohraničenia pre jednotlivé typy rámov

      LOWER     LEVEL     UPPER    MARGINAL

R1     -INF   1000.000  1000.000     2.000
R2     -INF    800.000  2000.000      .
R3     -INF    400.000   500.000      .
R4     -INF       .     1000.000      .

                           LOWER     LEVEL     UPPER    MARGINAL

---- VAR cz                 -INF   9200.000     +INF       .

  cz  celkový zisk

---- VAR x  počet vyrobených rámov daného typu

      LOWER     LEVEL     UPPER    MARGINAL

R1      .     1000.000     +INF       .
R2      .      800.000     +INF       .
R3      .      400.000     +INF       .
R4      .         .        +INF     -0.200
```

Pre modely, ktorých riešenia nie sú optimálne, sa môžu za istými ohraničeniami nachádzať niektoré z nasledujúcich indikátorov:

Indikátor	Popis
NOPT	daný riadok alebo stĺpec je neoptimálny
INFES	daný riadok alebo stĺpec je neprípustný
UNBND	daný riadok alebo stĺpec spôsobuje neohraničenosť modelu

13.3.6 „Report summary“

Posledná časť výstupu generovaného príkazom `solve` je „Report summary“, ktorá zobrazuje počet riadkov či stĺpcov označených daným indikátorom (`NOPT`, `INFES`, `UNBND`). Pre náš model vyzerá takto:

```
**** REPORT SUMMARY :        0     NONOPT
                             0 INFEASIBLE
                             0  UNBOUNDED
```

13.3.7 File summary“

Je to posledná časť celého výstupu, ktorá nás informuje o adrese vstupného a výstupného súboru:

```
**** FILE SUMMARY

Input      C:\WINDOWS\gamsdir\Ramy na obrazy1.gms
Output     C:\WINDOWS\gamsdir\Ramy na obrazy1.lst
```

13.4 Chybové hlásenia

V jazyku GAMS sa môžu objaviť tri druhy chýb, a to kompilačné chyby, vykonávacie chyby a chyby vyskytujúce sa počas riešenia daného problému. Každému z týchto troch typov sa budeme venovať osobitne. Každá chyba je vo výstupe označená štyrmi hviezdičkami (`****`). Po prvej objavenej chybe sa pri najbližšej príležitosti beh programu zastaví. V jazyku GAMS môžu nastať stovky rozličných chýb, ktoré sú sprevádzané zrozumiteľným popisom, a preto si uvedieme len zopár z nich na ilustráciu spôsobu hlásenia chýb vo výstupnom súbore.

13.4.1 Kompilačné chyby

Ide o chyby zistené počas kompilácie programu. Niektoré typické príčiny výskytu kompilačných chýb:

a) použitie identifikátora, ktorý nebol ešte deklarovaný
b) zadanie indexov v zlom poradí
c) chýbajúca bodkočiarka na mieste, kde je nevyhnutná
d) preklepy v menách identifikátorov a prvkov množín

Ak sa v programe objaví kompilačná chyba, v komponente výstupu „Echo print“ sa pod miestom jej výskytu, v samostatnom riadku označenom `****`, zobrazí znak `$` a číslo tejto chyby. Na konci komponentu „Echo print“ sa nachádza zoznam všetkých čísel chýb a ich popis. Dokumentuje to nasledujúci príklad, pričom uvádzame len „Echo print“:

```
   1  set i /i1*i5/;
   2  parameter vektor(i);
   3  vektor('1')=10;
****           $170
   4  vktor(i)=14;
****      $140
   5  vektor('i3')=5
   6  vektor(i)=-1;
****       $409
Error Messages

140  Unknown symbol
170  Domain violation for element
409  Unrecognizable item - skip to find a new statement
       looking for a ';' or a key word to get started again

**** 3 ERROR(S)   0 WARNING(S)
```

Často sa stáva, že jedna chyba spôsobí celý rad ďalších, preto sa treba vždy zamerať na odstránenie prvej v poradí.

Špeciálnou skupinou kompilačných chýb sú chyby spojené s kompiláciou príkazu `solve`. Napríklad:

a) parameter nachádzajúci sa v rovnici neobsahuje ešte numerické hodnoty
b) naprogramovaný model nekorešponduje s typom modelu, ktorý sme uviedli v príkaze `solve`

V nasledujúcom príklade sa vyskytujú práve tieto dve:

```
   1  parameter v;
   2  variables x,y,z;
   3  equation rov,zz;
   4  zz ..  z =e= x - y;
   5  rov ..       sqr(x) + sqr(y) =l= v;
   6  model m /all/;
   7  solve m maximizing z using LP;
****                              $51,66,256
**** The following LP errors were detected in model m:
****  51 equation rov .. the function SQR is called with non-constant arguments
****  51 equation rov .. the function SQR is called with non-constant arguments
****  66 equation rov .. symbol "v" has no values assigned
Error Messages

 51  Endogenous function argument(s) not allowed in linear models
 66  The symbol shown has not been defined or assigned
       A wild shot: You may have spurious commas in the explanatory
       text of a declaration. Check symbol reference list.
256  Error(s) in analyzing solve statement. More detail appears
     Below the solve statement above

**** 3 ERROR(S)   0 WARNING(S)
```

V prípade kompilačných chýb v príkaze `solve` sa zobrazuje, popri zozname týchto chýb na konci „Echo print-u“, aj ich podrobnejší popis pod príkazom `solve`.

13.4.2 Vykonávacie chyby

Chyby zistené počas vykonávania jednotlivých príkazov. Najčastejšie ich spôsobujú nedefinované operácie, ako napríklad:

a) delenie nulou
b) logaritmus záporného čísla

Vďaka existencii špeciálneho symbolu `UNDF` môže GAMS, po objavení nedefinovanej operácie, pokračovať vo vykonávaní nasledujúcich príkazov, pričom hodnota každej nedefinovanej operácie bude `UNDF`. Túto hodnotu môžeme aj zobraziť pomocou príkazu `display`. Teda prebehne celá vykonávacia časť, avšak model nemožno riešiť, kým sa v programe nachádza akákoľvek chyba. Napríklad:

```
   1  parameter v/1/;
   2  variables x,y,z;
   3  x.l=log(v-1);
   4  equation rov,zz;
   5  zz ..  z =e= x - y;
   6  rov ..       sqr(x) + sqr(y) =l= v;
   7  model m /all/;
   8  solve m maximizing z using NLP;

COMPILATION TIME     =        0.000 SECONDS    2.2 Mb  WIN216-141 Jan 24, 2005

E x e c u t i o n

**** Exec Error at line 3: log: FUNC SINGULAR: x = 0

Setup/Generation    SOLVE m Using NLP From line 8

**** SOLVE from line 8 ABORTED
**** EXECERROR=1
```

13.4.3 Chyby počas riešenia

Chyby vyskytujúce sa počas vykonávania príkazu `solve`. Prvý druh sa nazýva Maticovými chybami („Matrix errors“). Jedná sa o chyby spojené s transformáciou problému do žiadaného vstupného formátu pre daný solver. Toto sú len dve takéto chyby z mnohých možných:

a) nekonzistentné (`.lo > .up`) alebo nedovolené (typ premennej `semicont` musí mať `.lo>0`) hranice
b) špeciálne symboly (`inf,-inf,NA`) sú používané ako koeficienty matice daného problému

V nasledujúcom príklade nie sú hranice premennej `y` konzistentné (`.lo > .up`):

```
1  parameter v/1/;
2  variables x,y,z;
3  equation rov,zz;
4  zz ..  z =e= x - y;
5  rov ..       sqr(x) + sqr(y) =l= v;
6  y.lo=0.5;   y.up=-0.5
7  model m /all/;
8  solve m maximizing z using NLP;


**** Matrix error - lower bound > upper bound
y   (.LO, .L, .UP = 0.5, 0, -0.5)

**** SOLVE from line 8 ABORTED, EXECERROR = 1
```

Druhým typom sú chyby, ktoré nastanú vo výpočtoch v samotnom riešiacom podsystéme (solver) a nie v GAMS-e. Môže to byť napríklad delenie nulou:

```
1  variable x,y;
2  equation rov;
3  rov ..  y =e= sqrt(10/x);
4  x.l = 10;
5  x.lo = 0;
6  model m /all/;
7  solve m maximizing y using NLP;

             S O L V E      S U M M A R Y

     MODEL   m                   OBJECTIVE  y
     TYPE    NLP                 DIRECTION  MAXIMIZE
     SOLVER  MINOS               FROM LINE  24

**** SOLVER STATUS     5 EVALUATION ERROR LIMIT
**** MODEL STATUS      7 INTERMEDIATE NONOPTIMAL
**** OBJECTIVE VALUE           3.1623E+149

 RESOURCE USAGE, LIMIT          0.016      1000.000
 ITERATION COUNT, LIMIT         0         10000
 EVALUATION ERRORS              2             0

 EXIT - Function evaluation error limit exceeded.

**** ERRORS/WARNINGS IN EQUATION rov
     2 error(s): div: FUNC SINGULAR: x/y, |y| <= 1e-150 (RESULT SET TO  0.1+300)

**** REPORT SUMMARY :        1     NONOPT ( NOPT)
                             0 INFEASIBLE
                             0  UNBOUNDED
                             2     ERRORS ( ****)
```

14 Príkaz display

Pomocou príkazu `display` môžeme vypísať do výstupného súboru akékoľvek údaje alebo text. Je určený skôr na kontrolu správnosti modelu, ako na dokumentáciu výsledkov, keďže poskytuje len obmedzené možnosti formátovania výstupu. Doteraz sme v práci používali iba prednastavený formát. V tejto kapitole sa budeme venovať popisu možností formátovania a použitia príkazu `display`.

14.1 Jednoduché použitie príkazu display

Syntax:

```
display identifikator | "text" {, identifikator | "text"};
```

Slovo `identifikator` vyjadruje identifikátor daného dátového typu bez jeho oboru. Ak sa jedná o premenné, rovnice alebo modely, treba používať aj konkrétnu príponu. Výstup generovaný príkazom `display` pozostáva z prvkov a k nim prislúchajúcich hodnôt. Treba poznamenať, že ak identifikátor obsahuje prednastavené hodnoty, tak tieto hodnoty nebudú vo výstupe príkazu `display` zobrazené. Slovo `"text"` vyjadruje ľubovoľný text uvedený v úvodzovkách.

Príklad: Nasledujúci príklad ilustruje použitie príkazu `display`:

```
sets s /s1*s4/ , t /t1*t4/;
parameters p(s),q(t);       p(s)=uniform(-9,9);      q(t)=uniform(-9,9);
variable v(s,t);       v.l(s,t)=p(s)+q(t);
display s,p,q," Nasledujú hodnoty 'v' ako výsledok priradenia v=p+q ",v.l;
```

Výstupný súbor bude obsahovať nasledujúcu sekciu prislúchajúcu k nášmu príkazu `display`:

```
----      8 SET s
s1,    s2,    s3,    s4

----      8 PARAMETER p
s1 -5.909,    s2  6.179,    s3  0.907,    s4 -3.580

----      8 PARAMETER q
t1 -3.740,    t2 -4.967,    t3 -2.703,    t4  6.413

----      8 Nasledujú hodnoty 'v' ako výsledok priradenia v=p+q
----      8 VARIABLE v.L
             t1          t2          t3          t4
s1        -9.649     -10.876      -8.612       0.504
s2         2.439       1.212       3.476      12.592
s3        -2.833      -4.060      -1.796       7.320
s4        -7.320      -8.547      -6.283       2.833
```

14.2 Prednastavená štruktúra výstupu príkazu display

V prípade viacrozmerných identifikátorov (napríklad `a(i,j,k,l,m)`) je prednastavená štruktúra výstupu uvedená v nasledujúcej tabuľke, pričom pre každú kombináciu indexov z množín zo stĺpca „Rovinné množiny“ sa vytvorí samostatná tabuľka, ktorej štruktúra je daná ďalšími dvoma stĺpcami:

Počet množín	Rovinné množiny	Riadkové množiny	Stĺpcové množiny
1	-	Forma zoznamu	1
2	-	1	2
3	-	1 , 2	3
4	1	2 , 3	4
5	1 , 2	3 , 4	5
6	1 , 2 , 3	4 , 5	6
7	1 , ... , 4	5 , 6	7
8	1 , ... , 5	6 , 7	8
9	1 , ... , 6	7 , 8	9
10	1 , ... , 7	8 , 9	10

Príklad: Nasledujúci príklad dokumentuje, ako funguje tento prednastavený formát príkazu `display`:

```
sets i /i1,i2/ , j /j1,j2/ , k /k1,k2/ , L /L1,L2/ , m /m1,m2/;
parameter a(i,j,k,L,m);       a(i,j,k,L,m)=uniform(-9,9);
display a;
```

Výstupný súbor bude obsahovať nasledujúcu sekciu prislúchajúcu k nášmu príkazu `display`:

```
----     20 PARAMETER a

INDEX 1 = i1  INDEX 2 = j1
                m1          m2
k1.L1       -7.792       0.004
k1.L2        8.966       1.417
k2.L1        8.840       4.721
k2.L2       -6.648       2.515

INDEX 1 = i1  INDEX 2 = j2
                m1          m2
k1.L1       -6.129      -4.499
k1.L2        3.041      -1.164
k2.L1       -2.525      -2.674
k2.L2       -6.633      -6.298

INDEX 1 = i2  INDEX 2 = j1
                m1          m2
k1.L1        1.604       5.956
k1.L2       -4.845       2.983
k2.L1        4.965      -3.534
k2.L2       -7.011       0.043

INDEX 1 = i2  INDEX 2 = j2
                m1          m2
k1.L1       -6.117       6.704
k1.L2       -4.228      -3.855
k2.L1        1.691       4.009
k2.L2        2.308      -0.652
```

14.3 Ovládacie prvky príkazu display

Ovládacie prvky príkazu `display` slúžia na stanovenie počtu riadkových a stĺpcových množín, ako aj na určenie počtu zobrazovaných desatinných miest vo výstupnom súbore. Ovládacie prvky môžu byť globálne, a teda platné pre všetky identifikátory zobrazené pomocou príkazu `display`, ale aj lokálne, a teda platné len pre konkrétny identifikátor.

Globálne ovládacie prvky: Globálne môžeme meniť len počet číslic, ktoré sa zobrazia za desatinnou čiarkou vo výstupe príkazu `display`. Forma zápisu je nasledovná:

```
option decimals=1;
```

To znamená, že všetky hodnoty, zobrazené pomocou príkazu `display`, budú vo výstupe zaokrúhlené na jedno desatinné miesto. Na mieste číslice 1 môže figurovať ľubovoľné celé číslo od 0 až po 8. Toto nastavenie môžeme pre konkrétne identifikátory zmeniť pomocou lokálnych ovládacích prvkov.

Lokálne ovládacie prvky: Pomocou nich vieme ovládať formát konkrétneho identifikátora.

Syntax:

```
option identifikator : desatinne : riadky : stlpce;
```

Kde `identifikator` vyjadruje identifikátor daného dátového typu, slovo `desatinne` vyjadruje počet desatinných miest (čísla 0 až 8), slovo `riadky` zas počet riadkových množín a slovo `stlpce` počet stĺpcových množín. Počet rovinných množín je (všetky – (`riadky` + `stlpce`)) a zároveň je aspoň 0.

Príklad: Parameter `a`, v tejto kapitole už zobrazený v prednastavenom formáte, teraz zobrazíme v troch rozdielnych formátoch pomocou nasledujúcich príkazov:

```
option a:0:3:2;      display "Prvý formát",a;
option a:2:2:2;      display "Druhý formát",a;
option a:0:0:4;      display "Tretí formát",a;
```

Výstupný súbor bude obsahovať nasledujúcu sekciu prislúchajúcu k našim príkazom `display`:

```
----     24 Prvý formát
----     24 PARAMETER a
                L1.m1       L1.m2       L2.m1       L2.m2
i1.j1.k1            2           5           9           6
i1.j1.k2            9           7           2           6
i1.j2.k1            2           3           6           4
i1.j2.k2            4           4           2           2
i2.j1.k1            6           8           3           6
i2.j1.k2            7           3           2           5
i2.j2.k1            2           8           3           3
i2.j2.k2            6           7           6           5

----     22 Druhý formát
----     22 PARAMETER a

INDEX 1 = i1
             L1.m1       L1.m2       L2.m1       L2.m2
j1.k1         1.54        5.00        8.98        5.63
j1.k2         8.93        7.10        2.05        6.12
j2.k1         2.28        3.00        6.35        4.48
j2.k2         3.88        3.81        2.05        2.20

INDEX 1 = i2
             L1.m1       L1.m2       L2.m1       L2.m2
j1.k1         5.71        7.65        2.85        6.33
j1.k2         7.21        3.43        1.88        5.02
j2.k1         2.28        7.98        3.12        3.29
j2.k2         5.75        6.78        6.03        4.71

----     26 Tretí formát
----     26 PARAMETER a
i1.j1.k1.L1.m1 2,    i1.j1.k1.L1.m2 5,    i1.j1.k1.L2.m1 9,    i1.j1.k1.L2.m2 6
i1.j1.k2.L1.m1 9,    i1.j1.k2.L1.m2 7,    i1.j1.k2.L2.m1 2,    i1.j1.k2.L2.m2 6
i1.j2.k1.L1.m1 2,    i1.j2.k1.L1.m2 3,    i1.j2.k1.L2.m1 6,    i1.j2.k1.L2.m2 4
i1.j2.k2.L1.m1 4,    i1.j2.k2.L1.m2 4,    i1.j2.k2.L2.m1 2,    i1.j2.k2.L2.m2 2
i2.j1.k1.L1.m1 6,    i2.j1.k1.L1.m2 8,    i2.j1.k1.L2.m1 3,    i2.j1.k1.L2.m2 6
i2.j1.k2.L1.m1 7,    i2.j1.k2.L1.m2 3,    i2.j1.k2.L2.m1 2,    i2.j1.k2.L2.m2 5
i2.j2.k1.L1.m1 2,    i2.j2.k1.L1.m2 8,    i2.j2.k1.L2.m1 3,    i2.j2.k1.L2.m2 3
i2.j2.k2.L1.m1 6,    i2.j2.k2.L1.m2 7,    i2.j2.k2.L2.m1 6,    i2.j2.k2.L2.m2 5
```

Posledný, špeciálny príkaz generoval výstup vo forme zoznamu. Spôsobila to naša požiadavka na 0 riadkových množín. V takomto prípade už tretia hodnota nemá význam počtu stĺpcových množín, ale počtu stĺpcov, na ktoré bude zoznam rozdelený. V našom prípade na 4 stĺpce.

15 Ďalšie možnosti jazyka GAMS

Cieľom tejto kapitoly je upozorniť na existenciu možností jazyka GAMS, ktorým sme sa nemohli bližšie venovať z dôvodu ohraničenia na rozsah diplomovej práce. Zároveň poskytneme konkrétne zdroje, z ktorých sa dá v prípade potreby podrobne oboznámiť s danou problematikou.

15.1 Integrated Development Environment (IDE)

V kapitole 1 Úvod sme uviedli ukážku tohto integrovaného vývojového prostredia. Viac priestoru táto téma nedostala, lebo podľa nášho názoru, naučiť sa, väčšine užívateľov postačujúce úkony, je pomerne jednoduché aj bez akéhokoľvek výkladu. Ak by však nastal nejaký problém, jeho riešenie by ste mohli nájsť na nasledujúcej adrese:

http://www.gams.com/dd/docs/bigdocs/gams2002/gamside.pdf [2]

15.2 Univerzálna množina

Univerzálna množina slúži na definovanie identifikátora prostredníctvom ľubovoľných množín. Zároveň môžeme takémuto identifikátoru priradiť hodnotu pre prvok, ktorý nebol nikde v programe spomenutý. Väčšinou sa tento typ množiny používa v súvislosti s prezentáciou výsledkov. Podrobnejšie informácie nájdete na:

http://www.gams.com/dd/docs/bigdocs/gams2002/sets.pdf [2]

15.3 Akronymy

Je to špeciálny dátový typ, umožňujúci použitie znakového reťazca ako hodnoty. Informácie na:

http://www.gams.com/dd/docs/bigdocs/gams2002/acronyms.pdf [2]

15.4 Výstup pomocou príkazu put

Pomocou príkazu `put` dokážeme vytvoriť výstup ľubovoľného formátu. Používa sa hlavne na dokumentáciu výsledkov. Podrobné informácie sa nachádzajú na nasledujúcich dvoch adresách:

http://www.gams.com/dd/docs/bigdocs/gams2002/output_via_put_commands.pdf [2]

http://www.gams.com/docs/gams/GAMSUsersGuide.pdf [1], Kapitola 15

15.5 Save and restart

Vďaka týmto črtám jazyka GAMS vieme program rozdeliť na viacero častí a spúšťať ich osobitne. Napríklad ak náš program obsahuje zdĺhavý výpočet, tak ho urobíme len raz a uložíme („save“). Pri každom ďalšom spustení programu môžeme použiť „restart“ a vyhnúť sa tak zdĺhavému výpočtu. Podrobné informácie nájdete na adrese:

http://www.gams.com/dd/docs/bigdocs/gams2002/save_restart.pdf [2]

15.6 Comand line parameters

Jedná sa o modifikáciu nastavení GAMS-u pomocou písania určitých parametrov do príkazového riadku alebo do riadku v GAMS IDE na to určeného. Zoznam týchto parametrov spolu s vysvetlením ich funkcie nájdete na adresách:

http://www.gams.com/dd/docs/bigdocs/gams2002/command_line_parameters.pdf [2]
http://www.gams.com/docs/gams/GAMSUsersGuide.pdf [1], Appendix B

15.7 Dolar control options

Jedná sa o modifikáciu nastavení GAMS-u pomocou operátora `$`. Zoznam týchto „dolárových“ príkazov spolu s vysvetlením ich funkcie nájdete na adresách:

http://www.gams.com/dd/docs/bigdocs/gams2002/dollarcommands.pdf [2]

http://www.gams.com/docs/gams/GAMSUsersGuide.pdf [1], Appendix C

15.8 Príkaz option

Jedná sa o modifikáciu nastavení GAMS-u pomocou príkazu `option`. Čo všetko sa dá v nastaveniach pomocou tohto príkazu zmeniť sa dozviete na adresách:

http://www.gams.com/dd/docs/bigdocs/gams2002/options.pdf [2]

http://www.gams.com/docs/gams/GAMSUsersGuide.pdf [1], Appendix D

15.9 Zahŕňanie externých súborov

Podrobné informácie o tejto tematike nájdete na adrese:

http://www.gams.com/dd/docs/bigdocs/gams2002/includes.pdf [2]

15.10 Používanie GDX (Gams Data eXchange) súborov

GAMS dokáže z týchto súborov načítavať a aj do nich zapisovať. Používanie GDX súborov je jedným z možných spôsobov importovania dát z rôznych programov respektíve exportovania dát do rôznych programov (napríklad Excel). Podrobné informácie nájdete na adrese:

http://www.gams.com/dd/docs/bigdocs/gams2002/gdxusage.pdf [2]

15.11 Prepojenie GAMS-u s rôznymi programami

Jedná sa o programy ako Excel, Access, SQL, Latex, Gnuplot, Matlab, MPS, Oracle... . Informácie na:

http://www.gams.com/dd/docs/bigdocs/gams2002/linkingtoother.pdf [2]
http://www.gams.com/~erwin/interface/interface.html [3]

15.12 Možnosti ovládania GAMS-u z externých programov

Jedná sa o prostredia programovacích jazykov Visual Basic, Delphi, C++, Java... . Informácie na:

http://www.gams.com/dd/docs/bigdocs/gams2002/externalincharge.pdf [2]
http://www.gams.com/~erwin/interface/interface.html [3]

Druhá časť:
Ilustrácia možností jazyka GAMS

16 Úvod do druhej časti

Cieľom druhej časti diplomovej práce je demonštrovať nástroje jazyka GAMS, diskutované v prvej časti. Tento cieľ napĺňame v kapitolách 17-21 prostredníctvom rôznorodých modelov relatívne malých rozmerov. Najskôr sa venujeme úlohám lineárneho programovania, potom prejdeme k lineárnemu regresnému modelu a metóde najmenších štvorcov, potom nasleduje DEA model, CGE model a na záver uvedieme model Markowitzovej teórie portfólia. DEA model uvedený v tejto časti je síce úlohou lineárneho programovania, ale z dôvodu výnimočnosti, všeobecnosti a praktickej použiteľnosti DEA modelov sme ho nezaradili do kapitoly Lineárne programovanie, ale venovali sme mu samostatnú kapitolu. Skoro všetky kapitoly sa skladajú z k nim prislúchajúceho všeobecného modelu a jedného alebo viacerých konkrétnych modelov. Všeobecný model obsahuje matematickú formuláciu, no väčšinou neobsahuje formuláciu v jazyku GAMS, pretože táto formulácia je uvedená v rámci konkrétneho modelu s konkrétnymi dátami. Konkrétny model vždy pozostáva zo zadania, matematickej formulácie, formulácie v jazyku GAMS a výsledkov poskytnutých GAMS-om. Inú štruktúru má iba kapitola CGE , pretože formulácia každého CGE modelu môže byť veľmi špecifická, a teda nemá význam hovoriť o všeobecnom CGE modeli. Matematické formulácie vyskytujúce sa v tejto časti diplomovej práce sú prispôsobené tak, aby korešpondovali s formuláciami v jazyku GAMS. Model Rámy na obrazy z kapitoly Lineárne programovanie je detailne vysvetlený, pretože má slúžiť hlavne záujemcom o jazyk GAMS, ktorí sa s ním stretávajú po prvýkrát (ako sme uviedli v úvode diplomovej práce).

V celej tejto časti používame v rámci zdrojového kódu nasledovnú konvenciu práce so zátvorkami:

`( )` – používame ich v súvislosti s aritmetickými operáciami
`{ }` – používame ich v súvislosti s kľúčovými slovami vyžadujúcimi si zátvorky (napr.: `sum`, `if`, `card`)
`[ ]` – používame ich v súvislosti s oborom identifikátora

Ilustrujeme to na nasledujúcom príklade:

```
y[i]=sum{j,(x[i,j]+b[j])/a[j]};
```

Inšpiráciou nám bol jazyk C++, v ktorom sa so zátvorkami pracuje na podobnom princípe.

17 Lineárne programovanie (LP)

Lineárne programovanie sa zaoberá riešením optimalizačných úloh, v ktorých buď maximalizujeme, alebo minimalizujeme lineárnu funkciu viacerých premenných (účelovú funkciu) pri lineárnych ohraničeniach. Každé ohraničenie musí byť buď v tvare „tupej“ nerovnosti, alebo v tvare rovnosti. Jednotlivé premenné môžu byť voľné alebo nezáporné.

17.1 Všeobecná úloha LP

Na to aby sme sformulovali všeobecnú úlohu lineárneho programovania, potrebujeme zadefinovať množiny indexov, parametre a premenné v nej vystupujúce.

17.1.1 Matematická formulácia úlohy LP

Hlavné množiny indexov sú dve. Prvou je množina indexov pre všetky premenné, pomenovali sme ju *I* a tou druhou je množina indexov pre všetky ohraničenia, pomenovali sme ju *J*. Vo formulácii budeme ešte potrebovať dve podmnožiny množiny *I* (*Iv* , *In*) a dve podmnožiny množiny *J* (*Jn* , *Jr*). Množina *Iv* obsahuje indexy prislúchajúce k voľným premenným a množina *In* obsahuje indexy prislúchajúce k nezáporným premenným. Množina *Jn* obsahuje indexy prislúchajúce k ohraničeniam v tvare nerovností a množina *Jr* obsahuje indexy prislúchajúce k ohraničeniam v tvare rovností.

Ako parametre všeobecnej úlohy lineárneho programovania vystupujú dva vektory (*c* , *b*) a jedna matica (*A*). Zložky vektora *c* predstavujú koeficienty jednotlivých premenných v účelovej funkcii a zložky vektora *b* predstavujú hodnoty pravých strán jednotlivých ohraničení. *A* je matica koeficientov jednotlivých premenných v jednotlivých ohraničeniach. Vo všeobecnosti vyzerajú nasledovne:

$$c = \left(c_i\right) \qquad b = \left(b_j\right) \qquad A = \left(a_{ji}\right) \qquad \text{, kde} \quad i \in I \quad \text{a} \quad j \in J$$

Vektor, ktorého zložkami sú jednotlivé premenné, sa označuje *x* a vo všeobecnosti vyzerá nasledovne:

$$x = \left(x_i\right) \qquad \text{, kde} \quad i \in I$$

Teraz už môžeme uviesť matematickú formuláciu všeobecnej úlohy lineárneho programovania, ktorú z dôvodu kompatibility s jazykom GAMS uvádzame v zložkovom tvare:

$$\begin{aligned} \max \quad & \sum_{i \in Iv} c_i x_i + \sum_{i \in In} c_i x_i \\ & \sum_{i \in Iv} a_{ji} x_i + \sum_{i \in In} a_{ji} x_i \le b_j \qquad , \quad j \in Jn \\ & \sum_{i \in Iv} a_{ji} x_i + \sum_{i \in In} a_{ji} x_i = b_j \qquad , \quad j \in Jr \\ & x_i \ge 0 \qquad , \quad i \in In \end{aligned}$$

17.1.2 Formulácia úlohy LP v jazyku GAMS

Uvádzame ilustráciu úlohy LP vo všeobecnom tvare naprogramovanej v užívateľskom prostredí jazyka GAMS. V zdrojovom kóde chýba smer optimalizácie, pretože v jazyku GAMS sa udáva až v príkaze `solve`, po dosadení konkrétnych dát:

```
$eolcom //

//=============================================================================
* DEKLARÁCIA MNOŽÍN, PARAMETROV, PREMENNÝCH A ROVNÍC
  sets
      I "množina indexov pre všetky premenné"
      J "množina indexov pre všetky ohraničenia";

  parameters
      A[J,I] "matica koeficientov jednotlivých premenných v jednotlivých ohraničeniach"
      b[J]   "vektor pravých strán ohraničení"
      c[I]   "vektor koeficientov jednotlivých premenných v účelovej funkcii";

  variables
      huf  "hodnota účelovej funkcie"
      x[I] "vektor jednotlivých premenných";

  equations
      uf       "účelová funkcia"
      ohr_n[J] "blok všetkých ohraničení v tvare nerovností"
      ohr_r[J] "blok všetkých ohraničení v tvare rovností"
      nezap[I] "blok všetkých nezáporných premenných";

//=============================================================================
* FORMULÁCIA ÚLOHY LINEÁRNEHO PROGRAMOVANIA VO VŠEOBECNOM TVARE
  sets
      Iv[I] "podmnožina množiny 'p' obsahujúca indexy pre voľné premenné"
      In[I] "podmnožina množiny 'p' obsahujúca indexy pre nezáporné premenné"
      Jn[J] "podmnožina množiny 'o' obsahujúca indexy pre ohraničenia v tvare nerovností"
      Jr[J] "podmnožina množiny 'o' obsahujúca indexy pre ohraničenia v tvare rovností";

    uf        ..  huf =e= sum{Iv,c[Iv]*x[Iv]} + sum{In,c[In]*x[In]};

    ohr_n[Jn] ..          sum{Iv,A[Jn,Iv]*x[Iv]} + sum{In,A[Jn,In]*x[In]} =l= b[Jn];

    ohr_r[Jr] ..          sum{Iv,A[Jr,Iv]*x[Iv]} + sum{In,A[Jr,In]*x[In]} =e= b[Jr];

    nezap[In] ..                                                  x[In]  =g=  0;

  model vs_LP /all/;
```

17.2 Konkrétna úloha LP – „ Rámy na obrazy “

17.2.1 Zadanie úlohy

Firma vyrába štyri typy rámov na obrazy, ktoré označíme R1, R2, R3, R4. Každý typ vyžaduje určité množstvo výrobných faktorov, a to odbornej práce, kovu a skla. Tieto sú zobrazené v tabuľke:

	R1	R2	R3	R4	Množstvo [ks]
PRÁCA [h]	2	1	3	2	4000
KOV [kg]	4	2	1	2	6000
SKLO [kg]	6	2	1	2	10000
Zisk [Sk]	6	2	4	3	
Dopyt [ks]	1000	2000	500	1000	

V tejto tabuľke sú v poslednom stĺpci uvedené množstvá jednotlivých výrobných faktorov, ktoré máme k dispozícii (disponibilné množstvá), v predposlednom riadku je zisk z predaja jedného kusu daného typu rámu a v poslednom riadku je maximálny dopyt po jednotlivých typoch rámov. Úlohou firmy je maximalizovať zisk.

17.2.2 Matematická formulácia úlohy

Najskôr si zvolíme množiny indexov, ktoré budeme v modeli používať. V našom prípade potrebujeme dve, a to množinu *r* obsahujúcu typy rámov a množinu *vf* obsahujúcu výrobné faktory:

$$r = \{ r1 , r2 , r3 , r4 \} \qquad vf = \{ praca , kov , sklo \}$$

Vytvoríme si maticu *S* spotrebovaných množstiev jednotlivých výrobných faktorov na daný typ rámu. Najskôr maticu uvedieme bez dosadenia konkrétnych číselných údajov, a potom po ich dosadení:

$$S = \begin{pmatrix} s_{praca\ r1} & s_{praca\ r2} & s_{praca\ r3} & s_{praca\ r4} \\ s_{kov\ r1} & s_{kov\ r2} & s_{kov\ r3} & s_{kov\ r4} \\ s_{sklo\ r1} & s_{sklo\ r2} & s_{sklo\ r3} & s_{sklo\ r4} \end{pmatrix} \quad \Leftrightarrow \quad S = \begin{pmatrix} 2 & 1 & 3 & 2 \\ 4 & 2 & 1 & 2 \\ 6 & 2 & 1 & 2 \end{pmatrix}$$

Teraz si ešte potrebujeme vytvoriť vektor ziskov z predaja jednej jednotky daného typu rámu, označili sme ho *z*, vektor dopytu po jednotlivých typoch rámov *d* a vektor disponibilných množstiev jednotlivých výrobných faktorov, ktorý sme nazvali *m*:

$$z = \begin{pmatrix} z_{r1} \\ z_{r2} \\ z_{r3} \\ z_{r4} \end{pmatrix} \quad d = \begin{pmatrix} d_{r1} \\ d_{r2} \\ d_{r3} \\ d_{r4} \end{pmatrix} \quad m = \begin{pmatrix} m_{praca} \\ m_{kov} \\ m_{sklo} \end{pmatrix} \quad \Leftrightarrow \quad z = \begin{pmatrix} 6 \\ 2 \\ 4 \\ 3 \end{pmatrix} \quad d = \begin{pmatrix} 1000 \\ 2000 \\ 500 \\ 1000 \end{pmatrix} \quad m = \begin{pmatrix} 4000 \\ 6000 \\ 10000 \end{pmatrix}$$

Našou úlohou je nájsť taký vektor výrobných množstiev jednotlivých typov rámov, ktorý nám zabezpečí maximálny zisk. Tento vektor neznámych si označíme ako *x*:

$$x = \begin{pmatrix} x_{r1} \\ x_{r2} \\ x_{r3} \\ x_{r4} \end{pmatrix}$$

Model v zložkovom tvare bude vyzerať nasledovne:

$$\max \quad \sum_{i \in r} z_i \, x_i$$

$$\sum_{i \in r} s_{ji} \, x_i \le m_j \qquad , \quad j \in vf$$

$$0 \le x_i \le d_i \qquad , \quad i \in r$$

17.2.3 Formulácia úlohy v jazyku GAMS

Množiny: (*r* a *vf* z matematickej formulácie) Množiny sa v jazyku GAMS definujú nasledovným spôsobom, pričom medzi znakmi `//` sa nachádzajú prvky množín oddelené čiarkou alebo koncom riadku a v úvodzovkách je vysvetľujúci text k jednotlivým identifikátorom a prvkom:

```
sets
    r  "typy rámov na obrazy"       / r1 , r2 , r3 , r4 /
    vf "jednotlivé výrobné faktory" / praca "v hodinách"
                                      kov   "v kilogramoch"
                                      sklo  "v kilogramoch" /;
```

Tabuľka: (*S* z matematickej formulácie) Jazyk GAMS nám pri zadávaní matice umožňuje zachovať jej formu. To, že matica `S` je definovaná prostredníctvom indexových množín `vf` a `r`, sa vyjadruje pomocou zápisu `S[vf,r]`:

```
table
    S[vf,r] "množstvo spotrebovaného výrobného faktora na daný typ rámu"
                r1   r2   r3   r4
         praca   2    1    3    2
         kov     4    2    1    2
         sklo    6    2    1    2 ;
```

Napríklad `S['sklo','r1']` má hodnotu `6`.

Parametre: (*z*, *d* a *m* z matematickej formulácie) Prvým parametrom je `z[r]`. Pre jednotlivé prvky množiny `r` sme mu priradili hodnoty zo zadania. Rovnako sme postupovali aj s dopytom (`d[r]`). Tretí parameter `m[vf]` sa viaže na množinu výrobných faktorov a tiež sme mu priradili hodnoty zo zadania:

```
parameters
    z[r]  "zisk z jedného predaného rámu daného typu"
            / r1=6 , r2=2 , r3=4 , r4=3 /
    d[r]  "dopyt po jednotlivých typoch rámov"
            / r1=1000 , r2=2000 , r3=500 , r4=1000 /
    m[vf] "disponibilné množstvo jednotlivých výrobných faktorov"
            / praca=4000 , kov=6000 , sklo=10000 /;
```

Tu sa končí vkladanie zadania úlohy a môžeme prejsť k formulácii modelu.

Premenné: (*x* z matematickej formulácie) Nezápornosť `x` sme zaručili tak, že sme do dolnej hranice premennej `x` priradili 0 pre všetky typy rámov `x.lo[r]=0`. Dopytové ohraničenie sme vyriešili priradením dopytu do hornej hranice premennej `x` pre každý typ rámu `x.up[r]=d[r]`. V každom modeli v jazyku GAMS potrebujeme ešte jednu voľnú skalárnu premennú, ktorá vyjadruje hodnotu účelovej funkcie. V našom modeli sme si ju nazvali `cz` ako celkový zisk z predaja rámov:

```
variables
    cz   "celkový zisk"
    x[r] "počet vyrobených rámov daného typu";
  x.lo[r]=0;
  x.up[r]=d[r];
```

Rovnice: V jazyku GAMS sa všetky druhy ohraničení spolu s účelovou funkciou jednotne nazývajú rovnicami. Prvú časť zápisu tvorí deklarácia identifikátorov a druhú definícia jednotlivých relačných vzťahov. Rovnica `uf` vyjadruje účelovú funkciu, ktorej hodnotu `cz` budeme maximalizovať a rovnica `zdroje[vf]` vyjadruje ohraničenia na množstvá jednotlivých výrobných faktorov. Pre každý prvok z množiny `vf` existuje jedno konkrétne ohraničenie:

```
equations
    uf         "účelová funkcia"
    zdroje[vf] "ohraničenia na zdroje";
  uf ..          cz =e= sum{r,z[r]*x[r]};
  zdroje[vf] ..         sum{r,S[vf,r]*x[r]} =l= m[vf];
```

Model: Ešte treba sformulovať model. My sme si ten náš pomenovali ramy a zahrnuli sme doň všetky definované rovnice prostredníctvom výrazu `all`. Namiesto neho by sme ich mohli aj vymenovať oddelené čiarkou alebo koncom riadku. Uvedieme oba možné príkazy na zostavenie modelu:

```
model ramy / all /;
model ramy / uf , zdroje /;
```

Príkaz na riešenie: Začína kľúčovým slovom `solve`, za ktorým nasleduje názov modelu (`ramy`), smer optimalizácie (`maximizing` alebo `minimizing`), optimalizovaná premenná (`cz`), slovo `using` a typ úlohy (`lp` – lineárne programovanie):

```
solve ramy maximizing cz using lp;
```

Príkaz „display“: Umožňuje nám zobraziť vo výstupnom súbore jednotlivé množiny, parametre, premenné, ... Nasledujúci príkaz zobrazí výsledné optimálne množstvá. Zápis `x.l` (`l` = level) je nutný, lebo premenná `x` obsahuje viacero číselných údajov (`x.lo`, `x.up`, ...):

```
display x.l;
```

Výstup by vyzeral takto, pričom neprítomnosť hodnoty pre prvok `R4` znamená, že hodnota je rovná `0`:

```
----     47 VARIABLE x.L  počet vyrobených rámov daného typu

R1 1000.000,    R2  800.000,    R3  400.000
```

Model v GAMS-e: Teraz uvedieme ako celý model vyzerá v prostredí jazyku GAMS pre operačný systém Windows:

```
sets
    r  "typy rámov na obrazy"       / r1 , r2 , r3 , r4 /
    vf "jednotlivé výrobné faktory" / praca "v hodinách"
                                      kov   "v kilogramoch"
                                      sklo  "v kilogramoch" /;

table
    S[vf,r] "množstvo spotrebovaného výrobného faktora na daný typ rámu"
                r1   r2   r3   r4
        praca    2    1    3    2
        kov      4    2    1    2
        sklo     6    2    1    2 ;

parameters
    z[r]  "zisk z jedného predaného rámu daného typu"
           / r1=6 , r2=2 , r3=4 , r4=3 /
    d[r]  "garantovaný odbyt jednotlivých typov rámov"
           / r1=1000 , r2=2000 , r3=500 , r4=1000 /
    m[vf] "disponibilné množstvo jednotlivých výrobných faktorov"
           / praca=4000 , kov=6000 , sklo=10000 /;

variables
    cz   "celkový zisk"
    x[r] "počet vyrobených rámov daného typu";
  x.lo[r]=0;
  x.up[r]=d[r];

equations
    uf         "účelová funkcia"
    zdroje[vf] "ohraničenia na zdroje";
  uf         ..  cz =e= sum(r,z[r]*x[r]);
  zdroje[vf] ..         sum(r,S[vf,r]*x[r]) =l= m[vf];

model ramy / all /;

solve ramy maximizing cz using lp;

display x.l;
```

17.2.4 Výsledky poskytnuté GAMS-om

Uvádzame výstup príkazu `display`:

```
----     39 VARIABLE x.L  počet vyrobených rámov daného typu

r1 1000.000,    r2  800.000,    r3  400.000
```

17.3 Konkrétna úloha LP – „ Investovanie “

17.3.1 Zadanie úlohy

Na začiatku každého z mesiacov 1, 2, 3 a 4 dostávame mzdu (príjmy) a platíme účty (výdavky). Peniaze, ktoré nám ostanú po týchto transakciách, môžeme na začiatku mesiaca investovať pri mesačnej úrokovej miere danej podľa dĺžky investovania v mesiacoch. Konkrétne údaje sú uvedené v tabuľkách. Úlohou je určiť stratégiu investovania, ktorá maximalizuje hotovosť na začiatku 5. mesiaca.

Mesiac	Príjmy	Výdavky
1	800	600
2	800	500
3	300	500
4	300	250

Počet mesiacov	Mesačná úroková sadzba
1	0,1 %
2	0,5 %
3	1 %
4	2 %

17.3.2 Matematická formulácia úlohy

Najskôr potrebujeme zadefinovať dve množiny indexov - *t* (čas) a *d* (dĺžka). Prvkami množiny *t* sú začiatky všetkých štyroch mesiacov. Prvky *d* vyjadrujú dĺžku časového obdobia v mesiacoch. Formálne sa tieto množiny môžu skladať z rovnakých prvkov, tak ako v našom modeli:

$$t = \{1, 2, 3, 4\} \qquad d = \{1, 2, 3, 4\}$$

Teraz už môžeme zadefinovať všetky parametre zo zadania. Sú nimi vektor príjmov *r*, vektor výdavkov *b* a vektor úrokových mier *y*. Prvé dva sú definované prostredníctvom množiny *t*, posledný prostredníctvom *d*:

$$r = (r_i) \qquad b = (b_i) \qquad y = (y_j) \qquad \text{, kde} \quad i \in t \quad \text{a} \quad j \in d$$

Po dosadení konkrétnych číselných údajov vyzerajú nasledovne:

$$r = \begin{pmatrix} 800 \\ 800 \\ 300 \\ 300 \end{pmatrix} \qquad b = \begin{pmatrix} 600 \\ 500 \\ 500 \\ 250 \end{pmatrix} \qquad y = \begin{pmatrix} 0{,}001 \\ 0{,}005 \\ 0{,}01 \\ 0{,}02 \end{pmatrix}$$

Neznámymi v modeli sú množstvá peňazí investované v *i* -tom mesiaci na *j* mesiacov, kde $i \in t$ a $j \in d$. Túto maticu sme nazvali *X*:

$$X = (x_{ij}) \qquad \text{, kde} \quad i \in t \quad a \quad j \in d$$

Nasleduje samotný model v zložkovom tvare. Náš model nie je celkom všeobecný, pretože sme ho zjednodušili pomocou nasledujúcej elementárnej úvahy. Maximalizujeme hotovosť na začiatku 5. mesiaca, a preto všetko čo nám po vyplatení výdavkov ostane, investujeme minimálne na jeden mesiac. Teda hotovosť na začiatku každého mesiaca, po všetkých transakciách, sa musí rovnať 0:

$$\max \quad \sum_{\substack{i \in t \\ j \in d \\ i+j=5}} x_{ij} (1 + y_j)^j$$

$$r_i + \sum_{\substack{k \in t \\ j \in d \\ k+j=i}} x_{kj} (1 + y_j)^j - b_i - \sum_{\substack{j \in d \\ i+j \le 5}} x_{ij} = 0$$

$$x_{ij} \ge 0 \qquad , \quad i \in t \quad \text{a} \quad j \in d$$

17.3.3 Formulácia úlohy v jazyku GAMS

Uvádzame ilustráciu modelu Investovanie naprogramovaného v užívateľskom prostredí jazyka GAMS:

```
  sets
      t "začiatok aktuálneho mesiaca" /1*4/;
    alias(t,d,tl);
$ontext
'd' vyjadruje dĺžku investovania a množina 'tl' je potrebná v súvislosti
s formuláciou modelu v jazyku GAMS (v matematickej formulácii sa nevyskytuje)
$offtext

  parameters
      r[t] "príjmy"        /1 =   800, 2 =   800, 3 =  300, 4 =  300/
      b[t] "výdavky"       /1 =   600, 2 =   500, 3 =  500, 4 =  250/
      y[d] "úroková miera" /1 = 0.001, 2 = 0.005, 3 = 0.01, 4 = 0.02/;

  variables
      x[t,d] "množstvo investícií v danom mesiaci na dané obdobie"
      cash   "množstvo peňazí na začiatku '5'-teho mesiaca";
  positive variable x;

  equations
      vf       "rovnica pre účelovú funkciu"
      cashz[t] "ohraničenie hotovosti na začiatku mesiaca po všetkých transakciách";

    vf ..        cash =e=  sum([t,d]$(ord(t)+ord(d)=card(t)+1),x[t,d]*(1+y[d])**ord(d));

    cashz[t] ..             r[t] + sum([tl,d]$(ord(tl)+ord(d)=ord(t)),x[tl,d]*(1+y[d])**ord(d))
                          - b[t] - sum(d$(ord(t)+ord(d)<=card(t)+1),x[t,d])
                     =e= 0;

  model investovanie /all/;

  solve investovanie using lp maximizing cash;

  display x.l,cash.l;
```

17.3.4 *Výsledky poskytnuté GAMS-om*

Uvádzame výstup príkazu `display`:

```
----   34 VARIABLE x.L  množstvo investícií v danom mesiaci na dané obdobie

            1           3           4

1                                200.000
2     199.800     100.200
4      50.000


----   34 VARIABLE cash.L = 369.772  množstvo peňazí na začiatku '5'-teho mesiaca
```

18 Lineárny regresný model (LRM)

Lineárny regresný model (LRM) vyzerá vo všeobecnosti takto:

$$y_i = \beta_0 + \sum_{j=1}^{k} \beta_j x_{ij} + \varepsilon_i \qquad (i = 1, \ldots, n),$$

kde y_i sú pozorované hodnoty skúmanej premennej, x_{ij} sú pozorované hodnoty jednotlivých vysvetľujúcich premenných a ε_i je stochastická zložka modelu pre *i* -te pozorovanie. Našou úlohou je odhadnúť parametre $\beta_0, \ldots, \beta_k$ metódou najmenších štvorcov (MNŠ). Takto získaný odhad bude mať najlepšie štatistické vlastnosti, ak ε_i spĺňa tzv. Gauss-Markove predpoklady. Tieto predpoklady neuvádzame z dôvodu obmedzenia na rozsah diplomovej práce. A zároveň nie sú z hľadiska naplnenia nášho cieľa (demonštrovať možnosti jazyka GAMS) až také podstatné.

18.1 Všeobecný LRM odhadnutý metódou najmenších štvorcov (MNŠ)

18.1.1 Matematická formulácia odhadovania parametrov LRM pomocou MNŠ

Začíname množinami indexov, ktoré sme si označili *p* (pozorovania) a *f* (faktory). Množina *p* obsahuje indexy jednotlivých pozorovaní a množina *f* obsahuje indexy jednotlivých vysvetľujúcich premenných (faktorov), pričom prvok 0 je výnimkou, ako uvidíme neskôr. Vo všeobecnosti vyzerajú takto:

$$p = \{1, \ldots, n\} \qquad f = \{0, \ldots, k\} \qquad , \quad k < n$$

Teraz si zadefinujeme vektor *y* obsahujúci hodnoty skúmanej premennej a maticu *X* obsahujúcu hodnoty vysvetľujúcich premenných v jednotlivých pozorovaniach. Výnimkou je jej prvý stĺpec so stĺpcovým indexom 0, tvorený vektorom jednotiek. Vo všeobecnosti by vyzerali takto:

$$y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \qquad X = \begin{pmatrix} x_{10} & x_{11} & \cdots & x_{1k} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n0} & x_{n1} & \cdots & x_{nk} \end{pmatrix} \quad \text{, kde} \quad \begin{pmatrix} x_{10} \\ \vdots \\ x_{n0} \end{pmatrix} = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$$

Neznámymi sú odhady regresných koeficientov $\beta_0, \ldots, \beta_k$, tvoriace vektor *b* (pre vektor odhadov sme nepoužili štandardné označenie $\hat{\beta}$, kvôli súladu s formuláciou v jazyku GAMS). Vo všeobecnosti vyzerá vektor *b* takto:

$$b = \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_k \end{pmatrix}$$

Ešte potrebujeme zadefinovať vektor reziduí *e*, ako rozdiel medzi vektorom *y* (pozorované hodnoty) a vektorom $\hat{y} = Xb$ (odhadnuté hodnoty parametra *y*). Vektor reziduí *e* je odhadom pre vektor chýb ε.

$$e = \begin{pmatrix} e_1 \\ \vdots \\ e_n \end{pmatrix} \quad \text{, kde} \quad i \in p$$

Neznáme b_j $(j \in f)$ vypočítame pomocou Metódy najmenších štvorcov, čo znamená, že budeme hľadať také parametre b_j, ktoré minimalizujú súčet štvorcov reziduí lineárnej regresie:

$$\min \quad \sum_{i \in p} e_i^2 = \sum_{i \in p} \left(y_i - \hat{y}_i \right)^2 = \sum_{i \in p} \left(y_i - \sum_{j \in f} x_{ij} b_j \right)^2$$

Po dosadení konkrétnych dát vyriešime takúto minimalizačnú úlohu v jazyku GAMS. Získame tak konkrétne parametre b_j, na základe ktorých odhadujeme skúmanú premennú y. V tejto fáze by prichádzalo na rad ekonometrické testovanie modelu pomocou rôznych štatistík a testov, ktoré však nie je témou našej diplomovej práce. Z tohto dôvodu sa obmedzíme len na výpočet niektorých vybraných štatistík:

„R-squared statistic“:

$$R^2 = \frac{\sum_{i \in p} \left(\hat{y}_i - \bar{y} \right)^2}{\sum_{i \in p} \left(y_i - \bar{y} \right)^2} \qquad \text{, kde} \quad \hat{y}_i = \sum_{j \in f} x_{ij} b_j \quad \text{a} \quad \bar{y} = \frac{1}{n} \sum_{i \in p} y_i$$

„Adjusted R-squared statistic“:

$$adj\, R^2 = 1 - \frac{n-1}{n-k} \left(1 - R^2 \right)$$

„Standard Error of regression“:

$$SE = \sqrt{\frac{\sum_{i \in p} e_i^2}{n-k}} \qquad \text{, kde} \quad e_i = y_i - \hat{y}_i$$

„Durbin-Watson statistic“:

$$DW = \frac{\sum_{\substack{i \in p \\ i \neq 1}} \left(e_i - e_{i-1} \right)^2}{\sum_{i \in p} e_i^2}$$

„Log likelihood“:

$$l = -\frac{n}{2} \left[1 + \ln 2\pi + \ln \left(\frac{1}{n} \sum_{i \in p} e_i^2 \right) \right]$$

„Akaike Information Criterion“:

$$AIC = -\frac{2l}{n} + \frac{2k}{n}$$

„Schwarz Criterion“:

$$SC = -\frac{2l}{n} + \frac{k \ln n}{n}$$

„F-statistic“:

$$F = \frac{R^2 / (k-1)}{(1-R^2) / (n-k)}$$

18.2 Konkrétny LRM – Griffinov model vplyvu prílevu zahraničného kapitálu na úspory

18.2.1 Matematická formulácia Griffinovho modelu

V 60-tych rokoch bol vo svetovej ekonomike rozšírený názor, že zvýšenie prílevu zahraničného kapitálu do ekonomiky vedie k jej zvýšenému rastu. Tento názor bol podložený predovšetkým výsledkami Harrod-Domarovho modelu ekonomického rastu, ktorý je akýmsi predchodcom Solowho modelu rastu. Výsledky tohto modelu indikovali, že zvyšovanie investícií zvyšuje ekonomický rast, pričom investície boli definované vzťahom $I = S + A$, kde S sú domáce úspory a A sú zahraničné úspory, t.j. prílev zahraničného kapitálu. Tento spôsob rastu ekonomiky bol v tom čase zaužívaný hlavne v rozvojových krajinách. V roku 1970 Griffin vyslovil teoretické pochybnosti o správnosti týchto výsledkov, t.j. že zahraničný kapitál bude mať takýto vplyv na danú ekonomiku. Zaznamenal, že výsledky Harrod-Domarovho modelu sú pravdivé, len ak domáce úspory nezávisia od zahraničného kapitálu. Argumentoval tvrdením, že aspoň nejakú časť domácich úspor je možné nahradiť zahraničným kapitálom, a preto istý rast v prílive zahraničného kapitálu bude znamenať menší rast investícií. V roku 1971 Griffin otestoval túto teóriu pomocou nasledujúceho lineárneho regresného modelu (LRM):

$$\left(\frac{S}{Y}\right)_i = \beta_1 + \beta_2 \left(\frac{A}{Y}\right)_i + \varepsilon_i \ ,$$

kde sa snažil ukázať nepriamu úmernosť medzi mierou národných úspor (S/Y) a mierou prílevu zahraničného kapitálu (A/Y), pričom Y predstavuje hrubý domáci produkt (HDP). Pri testovaní však zistil, že tento model nebol korektný, a preto ho vylepšil nasledovne:

Uvažoval spotrebu danú vzťahom

$$C = a + b(Y + A)$$

a keďže úspory sú definované ako

$$S = Y - C$$

tak po dosadení dostávame

$$S = (1-b)Y - a - bA$$

Po predelení Y máme

$$\left(\frac{S}{Y}\right) = (1-b) - a\left(\frac{1}{Y}\right) - b\left(\frac{A}{Y}\right)$$

Nahradením konštánt koeficientmi $\beta_1, \beta_2, \beta_3$ získame vylepšený Griffinov model v tvare:

$$\left(\frac{S}{Y}\right)_i = \beta_1 + \beta_2 \left(\frac{A}{Y}\right)_i + \beta_3 \left(\frac{1}{Y}\right)_i + \varepsilon_i$$

Našou úlohou bude nájsť odhady pre koeficienty $\beta_1, \beta_2, \beta_3$ pomocou metódy najmenších štvorcov a vypočítať štatistiky (uvedené v matematickej formulácii všeobecného LRM) vypovedajúce o kvalite

modelu. Používame prierezové dáta obsahujúce informácie o úsporách a príleve zahraničného kapitálu v 66-tich rozvojových krajinách za rok 1987:

Krajina	*A*/*Y*	1/*Y*	*S*/*Y*
1	11,8	0,000208	3
2	10,7	0,000175	10
3	9,3	0,000057	2
4	22,8	0,000901	12
5	40,9	0,000671	-10
6	25,2	0,000325	-6
7	16,2	0,000606	1
8	15,8	0,000483	7
9	18,6	0,000510	0
10	15,3	0,000870	8
11	21,1	0,000493	20
12	16,1	0,000463	5
13	7,2	0,000281	5
14	0,5	0,000003	38
15	57,0	0,000529	1
16	10,0	0,000813	6
17	0,7	0,000005	22
18	11,6	0,000476	5
19	7,3	0,001111	10
20	8,1	0,000637	4
21	16,1	0,000990	-2
22	7,0	0,000144	20
23	10,5	0,000122	6
24	2,4	0,000032	11
25	9,7	0,000444	5
26	29,4	0,003704	-73
27	0,3	0,000041	20
28	7,4	0,000199	4
29	7,5	0,000166	13
30	19,0	0,001190	14
31	1,8	0,000014	29
32	13,6	0,000212	6
33	7,1	0,000224	2

Krajina	*A*/*Y*	1/*Y*	*S*/*Y*
34	5,0	0,000191	22
35	2,2	0,000029	16
36	8,2	0,000234	-12
37	2,4	0,000060	14
38	4,9	0,000029	8
39	10,6	0,000330	17
40	2,5	0,000131	19
41	6,4	0,000283	13
42	1,1	0,000021	26
43	9,0	0,000211	8
44	7,0	0,000465	21
45	5,9	0,000350	23
46	3,4	0,000142	7
47	1,7	0,000079	15
48	1,8	0,000219	18
49	1,9	0,000094	17
50	2,9	0,000118	20
51	0,6	0,000016	23
52	0,2	0,000031	26
53	0,1	0,000053	21
54	0,6	0,000022	23
55	3,7	0,000676	29
56	12,0	0,000234	-3
57	5,3	0,000232	-3
58	2,9	0,000042	10
59	1,2	0,000032	37
60	0,1	0,000007	17
61	0,1	0,000003	23
62	0,2	0,000156	11
63	0,1	0,000014	10
64	0,0	0,000008	38
65	2,3	0,000286	34
66	0,0	0,000020	25

Toto bola konkrétna matematická formulácia modelu, no aj v tomto prípade je užitočné naprogramovať model vo všeobecnom tvare. Následne budeme vedieť vyriešiť, pomocou takto naprogramovaného modelu, akýkoľvek lineárny regresný model metódou najmenších štvorcov. A to len prostredníctvom aktualizácie množín indexov a dát. Všetko ostatné zostáva nezmenené. Teraz si zadefinujeme len konkrétne množiny a parametre modelu, pretože matematickú formuláciu všeobecného lineárneho regresného modelu sme už uviedli a nebudeme ju na tomto mieste opakovať.

V našom konkrétnom lineárnom regresnom modeli budú množiny p a f vyzerať nasledovne:

$$p = \{1, \ldots, 66\} \qquad f = \{0, 1, 2\}$$

Teraz by sme mali ešte uviesť konkrétne číselné hodnoty parametrov, ale nespravíme tak, keďže sa už nachádzajú v zadaní.

18.2.2 Formulácia Griffinovho modelu v jazyku GAMS

Uvádzame ilustráciu Griffinovho modelu odhadnutého metódou najmenších štvorcov naprogramovaného v užívateľskom prostredí jazyka GAMS:

```
$eolcom //

//=============================================================================================
* VŠEOBECNÝ MODEL LINEÁRNEJ REGRESIE ODHADNUTÝ POMOCOU MNŠ
*                Y = X*beta + epsilon
  sets
      p       "pozorovania"
      f       "množina charakterizujúca vysvetľujúce premenné (faktory) v modeli";
  parameters
      x[p,f] "hodnoty pozorovaní pre dané vysvetľujúce premenné"
      y[p]   "hodnoty pozorovaní pre skúmanú premennú";

  variables
      b[f]   "vektor koeficientov, ktoré sú odhadmi pre zložky vektora beta"
      h_uf   "hodnota účelovej funkcie = suma štvorcov reziduí";

  equation
      uf     "účelová funkcia";
    uf ..   h_uf =e= sum(p,sqr(sum(f,b[f]*x[p,f])-y[p]));

  model mns /all/;

//=============================================================================================
* ZADÁVANIE DÁT DO MODELU
* KONKRÉTNY MODEL LINEÁRNEJ REGRESIE    S/Y = b0 + b1*A/Y + b2*1/Y + epsilon
  sets
      f /0 "konštanta"
         1 "A/Y"
         2 "1/Y"      /
      p /1*66/;


* načítanie dát z Excelu (data_mns.xls) do parametra x
$call "Gdxxrw data_mns.xls par=x Rng=a2:d68 rdim=1 cdim=1"
$gdxin data_mns.gdx
$load x

* načítanie dát z Excelu (data_mns.xls) do parametra y
$call "Gdxxrw data_mns.xls par=y Rng=g3:h68 rdim=1"
$gdxin data_mns.gdx
$load y

//=============================================================================================
* RIEŠENIE MODELU, NAPOČÍTANIE VYBRANÝCH ŠTATISTÍK A ZOBRAZENIE VÝSLEDKOV
  solve mns using nlp minimizing h_uf;

  scalars
      n           "Počet prvkov množiny p, teda počet pozorovaní"
      k           "Počet prvkov množiny f";
    n=card(p);
    k=card(f);

  parameters
      odhad_y[p]  "odhadnutá hodnota y v danom pozorovaní (X*b)"
      rezidua[p]  "odchylky odhadnutých hodnôt y od reálnych hodnôt y"
      priemer_y   "aritmetický priemer odhadovanej premennej"
      sd_y        "smerodajná odchylka odhadovanej premennej"
      SST         "Sum of Squared Totals"
      SSE         "Sum of Squared Estimates"
      SSR         "Sum of Squared Residuals";
    odhad_y[p]  = sum(f,b.l[f]*x[p,f]);
    rezidua[p]  = y[p]-odhad_y[p];
```

```
  priemer_y   = sum(p,y[p])/n;
  SST         = sum(p,sqr(y[p]-priemer_y));
  sd_y        = sqrt(SST/(n-1));
  SSE         = sum(p,sqr(odhad_y[p]-priemer_y));
  SSR         = sum(p,sqr(rezidua[p]));

parameters
    R_2         "R-squared statistic"
    adj_R2      "adjusted R-squared statistic"
    SE          "Standard Error of regression"
    DW          "Durbin-Watson statistic"
    log_like    "hodnota funkcie log likelihood"
    AIC         "Akaike Information Criterion"
    SC          "Schwarz Criterion"
    F_stat      "F-statistic";
  R_2    = SSE/SST;
  adj_R2   = 1-((n-1)/(n-k))*(1-R_2);
  SE       = sqrt(SSR/(n-k));
  DW       = sum(p$(ord(p)>1),sqr(rezidua[p]-rezidua[p-1]))/SSR;
  log_like = -(n/2)*(1+log(2*pi)+log(SSR/n));
  AIC      = -2*log_like/n + 2*k/n;
  SC       = -2*log_like/n + k*log(n)/n;
  F_stat   =  (R_2/(k-1))/((1-R_2)/(n-k));

option decimals=6;
option b:3:0:1;
display
  h_uf.l,b.l
  R_2,adj_R2,SE,DW,log_like,AIC,SC,F_stat;
```

18.2.3 Výsledky poskytnuté GAMS-om

Uvádzame výstup príkazu `display`:

```
----     89 VARIABLE b.L  vektor koeficientov, ktoré sú odhadmi pre zložky
vektora beta

0     20.905
1     -0.404
2 -17377.549


----     89 PARAMETER R_2          =     0.567917  R-squared statistic
            PARAMETER adj_R2       =     0.554200  adjusted R-squared statistic
            PARAMETER SE           =    10.263923  Standard Error of regression
            PARAMETER DW           =     1.787653  Durbin-Watson statistic
            PARAMETER log_like     =  -245.804699  hodnota funkcie log likelihood
            PARAMETER AIC          =     7.539536  Akaike Information Criterion
            PARAMETER SC           =     7.639066  Schwarz Criterion
            PARAMETER F_stat       =    41.402677  F-statistic
```

19 Data Envelopment Analysis (DEA)

DEA analýza je jedným z prostriedkov ekonomického manažmentu. Dá sa pomocou nej určiť relatívna efektívnosť jednotlivých producentov vykonávajúcich rovnakú činnosť v rámci danej skupiny (napr. pobočky bánk). Činnosť producentov je charakterizovaná vstupmi a výstupmi, ktoré môžu byť merané v rozličných jednotkách. DEA analýza sa používa hlavne v prípadoch, keď efektivitu nemôžme merať ziskom, čo je prípad organizačných útvarov nevýrobného charakteru (štátna správa, školstvo, bankovníctvo, zdravotníctvo, ...).

19.1 Vstupne orientovaný CCR model s konštantnými výnosmi z rozsahu

19.1.1 Matematická formulácia CCR modelu

Najskôr si musíme zadefinovať tri indexové množiny používané v modeli. Jednu obsahujúcu jednotlivých producentov, nazvali sme ju *p*. Druhú obsahujúcu jednotlivé vstupy *vs* a tretiu obsahujúcu jednotlivé výstupy *vy*. Ak by sme formulovali model iba v jazyku GAMS, bolo by prehľadnejšie nazývať prvky množín menami producentov, názvami vstupov a výstupov. No keďže chceme model naformulovať aj v reči matematiky, opäť zvolíme číselné indexovanie:

$$p = \{1, \ldots, n\} \qquad vs = \{1, \ldots, m\} \qquad vy = \{1, \ldots, s\}$$

Teraz zadefinujeme maticu *X*, ktorej riadky sú tvorené vektormi vstupov jednotlivých producentov a maticu *Y*, ktorej riadky sú tvorené vektormi výstupov jednotlivých producentov:

$$X = (x_{ij}) \qquad Y = (y_{ik}) \qquad \text{, kde} \quad i \in p\,, \quad j \in vs \quad \text{a} \quad k \in vy$$

Neznámymi sú zložky vektora váh pre vstupy *u* a zložky vektora váh pre výstupy *v*:

$$u = (u_j) \qquad v = (v_k) \qquad \text{, kde} \quad j \in vs \quad \text{a} \quad k \in vy$$

Teraz, keď už máme zadefinované všetky potrebné symboly, môžeme pristúpiť k formulácii vstupne orientovaného CCR modelu pre *i* -teho producenta, pričom uvažujeme konštantné výnosy z rozsahu:

$$\begin{aligned} \max \quad & \sum_{k \in vy} v_k\, y_{ik} \\ & \sum_{k \in vy} v_k\, y_{lk} - \sum_{j \in vs} u_j\, x_{lj} \le 0 \qquad , \quad l \in p \\ & \sum_{j \in vs} u_j\, x_{ij} = 1 \\ & u_j \ge 0 \quad , \quad v_k \ge 0 \qquad , \quad j \in vs \quad \text{a} \quad k \in vy \end{aligned}$$

Pre každého producenta treba vyriešiť jednu takúto úlohu, teda spolu treba vyriešiť *n* optimalizačných úloh. Tým sa však naše modelovanie nekončí, pretože v DEA modeloch treba vždy riešiť aj duálnu úlohu, respektíve obálkový model, ktorý dostaneme z duálneho pridaním doplnkových premenných s cieľom mať ohraničenia v tvare rovností. Na vytvorenie obálkového modelu ešte potrebujeme zadefinovať vektory duálnych (*theta* , *lambda*) a doplnkových (*sx* , *sy*) premenných, pričom *theta* nie je viazaná na žiadnu množinu:

$$theta \qquad lambda = (lambda_i) \qquad sx = (sx_j) \qquad sy = (sy_k) \qquad \text{, kde} \quad i \in p,\ j \in vs \ \text{ a } k \in vy$$

Teraz už môžeme uviesť obálkový model:

$$\min \quad theta$$

$$\sum_{l \in p} x_{l\,j}\, lambda_l + sx_j = theta\, x_{i\,j} \qquad , \quad j \in vs$$

$$\sum_{l \in p} y_{l\,k}\, lambda_l - sy_k = y_{i\,k} \qquad , \quad k \in vy$$

$$lambda_l \geq 0 \quad , \quad sx_j \geq 0 \quad , \quad sy_k \geq 0 \qquad , \quad l \in p\,, \quad j \in vs \quad \text{a} \quad k \in vy$$

Tento model opäť vyriešime pre každého producenta s tým, že ak v i -tej úlohe budú všetky optimálne hodnoty zložiek vektorov sx a sy rovné nule, riešime navyše tzv. „slackový“ model. Symbolom $theta^*$ sme označili optimálnu hodnotu premennej $theta$ po vyriešení i- teho obálkového modelu. „Slackový“ model pre i -teho producenta vyzerá takto:

$$\max \quad \sum_{j \in vs} sx_j + \sum_{k \in vy} sy_k$$

$$\sum_{l \in p} x_{l\,j}\, lambda_l + sx_j = theta^*\, x_{i\,j} \qquad , \quad j \in vs$$

$$\sum_{l \in p} y_{l\,k}\, lambda_l - sy_k = y_{i\,k} \qquad , \quad k \in vy$$

$$lambda_l \geq 0 \quad , \quad sx_j \geq 0 \quad , \quad sy_k \geq 0 \qquad , \quad l \in p\,, \quad j \in vs \quad \text{a} \quad k \in vy$$

Po vyriešení „slackového“ modelu dostávame nové optimálne hodnoty zložiek vektorov *lambda*, *sx* a *sy*. Tým sme skompletizovali údaje potrebné na vyhodnotenie efektívnosti i -teho producenta. Teraz uvedieme schému postupu vyhodnocovania:

ak $theta \neq 1 \wedge (sx_j > 0 \vee sy_k > 0) \Rightarrow$ útvar je **neefektívny**, projektuje sa na **pseudoefektívnu hranicu**

ak $theta \neq 1 \wedge (sx_j = 0 \wedge sy_k = 0) \Rightarrow$ riešime „slackový“ model s dosadenou hodnotou $theta^*$, dostaneme nové hodnoty *sx* a *sy* $\Rightarrow$

$\Rightarrow$ ak $(sx_j > 0 \vee sy_k > 0) \Rightarrow$ útvar je **neefektívny**, projektuje sa na **pseudoefektívnu hranicu**

ak $(sx_j = 0 \wedge sy_k = 0) \Rightarrow$ útvar je **neefektívny**, projektuje sa na **efektívnu hranicu**

ak $theta = 1 \wedge (sx_j > 0 \vee sy_k > 0) \Rightarrow$ útvar je **pseudoefektívny**

ak $theta = 1 \wedge (sx_j = 0 \wedge sy_k = 0) \Rightarrow$ riešime „slackový“ model s dosadenou hodnotou $theta^*$, dostaneme nové hodnoty *sx* a *sy* $\Rightarrow$

$\Rightarrow$ ak $(sx_j > 0 \vee sy_k > 0) \Rightarrow$ útvar je **pseudoefektívny**

ak $(sx_j = 0 \wedge sy_k = 0) \Rightarrow$ útvar je **efektívny**

19.2 Konkrétny CCR model – Efektívnosť spoločností pre rozvod elektrickej energie

19.2.1 Zadanie úlohy

Našou úlohou je urobiť analýzu efektívnosti jednotlivých spoločností pre rozvod elektrickej energie na základe nasledujúcich dát:

Rozvodné spoločnosti	VSTUPY		VÝSTUPY	
	Počet zamestnancov	Operatívne náklady	MWh	Počet zákazníkov
1	464	352687	1182900	130570
2	1907	1077952	6879000	397300
3	1456	139140	1668374	218223
4	2185	1287669	8826400	802000
5	424	154045	345909	35534
6	1492	834093	844496	112120
7	1269	340563	3317800	234661
8	2220	627783	3279800	264000
9	394	521469	1661000	124000
10	1427	546064	7598000	154140
11	1541	2193869	6152300	160526
12	727	387121	2673360	160000
13	3820	2999753	9313200	1426375
14	585	128753	521400	60881
15	1187	434542	718200	101219

19.2.2 Matematická formulácia CCR modelu

Je takmer identická s formuláciou všeobecného CCR modelu, až na to, že n je rovné 15, m je rovné 2 a s je rovné 2.

19.2.3 Formulácia CCR modelu v jazyku GAMS

Uvádzame ilustráciu DEA modelu naprogramovaného v užívateľskom prostredí jazyka GAMS:

```
$eolcom //

  sets
      p  "množina producentov" /1*15/
      vs "množina vstupov"     /1,2/
      vy "množina výstupov"    /1,2/
      p1[p],p2[p]; // podmnožiny množiny p
    alias(p,pp);   // množina identická s množinou p

  table
      x[p,vs] "matica vstupov"
                1         2
         1     464      352687
         2    1907     1077952
         3    1456      139140
         4    2185     1287669
         5     424      154045
         6    1492      834093
         7    1269      340563
         8    2220      627783
         9     394      521469
        10    1427      546064
        11    1541     2193869
        12     727      387121
        13    3820     2999753
        14     585      128753
        15    1187      434542;
    x[p,'2']=x[p,'2']/1000;

  table
      y[p,vy] "matica vystupov"
                  1        2
         1    1182900   130570
         2    6879000   397300
         3    1668374   218223
         4    8826400   802000
         5     345909    35534
         6     844496   112120
         7    3317800   234661
         8    3279800   264000
         9    1661000   124000
        10    7598000   154140
        11    6152300   160526
        12    2673360   160000
        13    9313200  1426375
        14     521400    60881
        15     718200   101219;
    y[p,vy]=y[p,vy]/1000;

//=======================================================================================
* FORMULÁCIA PRIMÁRNEJ ÚLOHY
  positive variables
      u[p,vs] "vektor váh pre vstupy"
      v[p,vy] "vektor váh pre výstupy";
  free variable
      h_ufp   "hodnota účelovej funkcie primárnej úlohy";
  equations
      u_fp[p]     "účelová funkcia primárnej úlohy"
      efekt[pp,p] "podmienka zaručujúca že efektívnosť ostatných bude medzi 0 a 1"
      norm[p]     "normalizácia vstupov";
    u_fp[p1]    ..  h_ufp =e= sum(vy,v[p1,vy]*y[p1,vy]);
    efekt[p1,p] ..            sum(vy,v[p1,vy]*y[p,vy]) - sum(vs,u[p1,vs]*x[p,vs]) =l= 0;
```

```
   norm[pl]    ..            sum(vs,u[pl,vs]*x[pl,vs]) =e= 1;

  model prim /u_fp,efekt,norm/;

//=============================================================================================
* FORMULÁCIA OBÁLKOVÉHO MODELU
  positive variables
      lambda[pp,p] "duálna premenná"
      sx[p,vs]     "slacková premenná"
      sy[p,vy]     "slacková premenná";
  free variable
      theta[p]     "duálna premenná"
      h_ufd        "hodnota účelovej funkcie duálnej úlohy";

  equations
      u_fd[p]       "účelová funkcia duálnej úlohy"
      vstupy[p,vs]  "duálne ohraničenie pre vstupy"
      vystupy[p,vy] "duálne ohraničenie pre výstupy";
    u_fd[pl]       .. h_ufd =e= theta[pl];
    vstupy[pl,vs]  ..           sum(p,x[p,vs]*lambda[pl,p]) + sx[pl,vs] =e= theta[pl]*x[pl,vs];
    vystupy[pl,vy] ..           sum(p,y[p,vy]*lambda[pl,p]) - sy[pl,vy] =e= y[pl,vy];

  model dual /u_fd,vstupy,vystupy/;

//=============================================================================================
* FORMULÁCIA "SLACKOVEJ" ÚLOHY
  equations
      sum_slack[p] "súčet 'slackov' = hodnota účelovej funkcie"
      vs_dos[p,vs] "duálne ohraničenie pre vstupy s dosadenou hodnotou theta";
    sum_slack[pl] ..  h_ufd =e= sum(vs,sx[pl,vs]) + sum(vy,sy[pl,vy]);
    vs_dos[pl,vs] ..            sum(p,x[p,vs]*lambda[pl,p]) + sx[pl,vs] =e=
                                theta.l[pl] * x[pl,vs];

  model slack /sum_slack,vs_dos,vystupy/;

//=============================================================================================
* RIEŠENIE
  parameter
      ef[*]     "efektivita jednotlivých producentov"
      suma_s    "suma 'slackov' pre konkrétneho producenta"
      r[p,*]    "zaznamenané sumy 'slackov' pre jednotlivých producentov"
      vysl[p,*] "výsledky spolu s ich interpretáciou";
  scalar epsilon_t "akú thetu už považujem za 0" /0.0000001/
         epsilon_s "akú sumu už považujem za 0"  /0.0000001/;

  p2[p]=yes;
  loop(p2,
    pl[p]=no;
    pl[p2]=yes;
    solve prim maximizing h_ufp using lp;
    ef[p2]=h_ufp.l;
    solve dual  minimizing h_ufd using lp;
    suma_s=sum(vs,sx.l[p2,vs]) + sum(vy,sy.l[p2,vy]);
    r[p2,'suma pred riešením']=suma_s;
    if (theta.l[p2]>1+epsilon_t or theta.l[p2]<1-epsilon_t,     // ak theta<>1
      if (suma_s>epsilon_s or suma_s<-epsilon_s,                 // ak suma_s<>0
        vysl[p2,'neefektívny útvar,pseudo']=theta.l[p2];
        else                                                     // inak (t.j. suma_s=0)
          solve slack maximizing h_ufd using lp;                    // riešime slackový model
          suma_s=sum(vs,sx.l[p2,vs]) + sum(vy,sy.l[p2,vy]);
          r[p2,'suma po riešení']=suma_s;
          if (suma_s>epsilon_s or suma_s<-epsilon_s,                // ak suma_s<>0
```

```
          vysl[p2,'neefektívny útvar,pseudo']=theta.l[p2]
          else                                                    // inak (t.j. suma_s=0)
            vysl[p2,'neefektívny útvar,efekt']=theta.l[p2];
        };
      };
    };
    if (theta.l[p2]<=1+epsilon_t and theta.l[p2]>=1-epsilon_t,  // ak theta=1
      if (suma_s>epsilon_s or suma_s<-epsilon_s,                 // ak suma_s<>0
        vysl[p2,'pseudoefektívny útvar']=theta.l[p2];
        else                                                     // inak (t.j. suma_s=0)
          solve slack maximizing h_ufd using lp;                    // riešime slackový model
          suma_s=sum{vs,sx.l[p2,vs]} + sum{vy,sy.l[p2,vy]};
          r[p2,'suma po riešení']=suma_s;
          if (suma_s>epsilon_s or suma_s<-epsilon_s,               // ak suma_s<>0
            vysl[p2,'pseudoefektívny útvar']=theta.l[p2]
            else                                                   // inak (t.j. suma_s=0)
              vysl[p2,'efektívny útvar']=theta.l[p2];
          };
      };
    };
  };

  option decimals=8;
  display ef,u.l,v.l,theta.l,lambda.l,sx.l,sy.l,r,vysl;

//=======================================================================================================
* EXPORTOVANIE VÝSLEDKOV DO EXCELU
  execute_unload "dea.gdx" u,v,ef,vysl,r,theta,lambda,sx,sy;
  execute 'GDXXRW.EXE dea.gdx par=vysl rng=vysledky!a4 ';
  execute 'GDXXRW.EXE dea.gdx par=r rng=vysledky!f4 ';
  execute 'GDXXRW.EXE dea.gdx par=ef rng=vahy!a4 rdim=1 ';
  execute 'GDXXRW.EXE dea.gdx var=u.l rng=vahy!d3 ';
  execute 'GDXXRW.EXE dea.gdx var=v.l rng=vahy!h3 ';
  execute 'GDXXRW.EXE dea.gdx var=theta.l rng=dual!a4:b20 rdim=1 ';
  execute 'GDXXRW.EXE dea.gdx var=lambda.l rng=dual!e3:r20 ';
  execute 'GDXXRW.EXE dea.gdx var=sx.l rng=dual!a23 ';
  execute 'GDXXRW.EXE dea.gdx var=sy.l rng=dual!e23 ';
```

19.2.4 Výsledky poskytnuté GAMS-om

Tento krát uvedieme výsledky vo forme tabuľky z Excelu, pričom prázdne miesta predstavujú nulu a neprítomnosť niektorých riadkových indexov tabuľky znamená, že tieto riadky sú celé nulové:

Efektivita	
1	0,760993
2	0,791890
3	1,000000
4	1,000000
5	0,313128
6	0,212443
7	0,972377
8	0,569856
9	0,985773
10	1,000000
11	0,769442
12	0,813778
13	1,000000
14	0,508529
15	0,317163

Váhy vstupov		
	1	*2*
1	0,002155	
2	0,000524	
3	0,000494	0,002016
4	0,000105	0,000598
5	0,000950	0,003876
6	0,000204	0,000833
7	0,000313	0,001771
8	0,000173	0,000981
9	0,002538	
10		0,001831
11	0,000649	
12	0,001376	
13	0,000244	0,000023
14	0,000901	0,003674
15	0,000338	0,001378

Váhy výstupov		
	1	*2*
1	0,000022	0,005626
2	0,000089	0,000444
3		0,004582
4	0,000048	0,000716
5		0,008812
6		0,001895
7	0,000143	0,002122
8	0,000079	0,001175
9	0,000433	0,002148
10	0,000132	
11	0,000111	0,000549
12	0,000235	0,001164
13		0,000701
14		0,008353
15		0,003133

Theta	
1	0,760993
2	0,791890
3	1,000000
4	1,000000
5	0,313128
6	0,212443
7	0,972377
8	0,569856
9	0,985773
10	1,000000
11	0,769442
12	0,813778
13	1,000000
14	0,508529
15	0,317163

Lambda				
	3	***4***	***10***	***13***
1		0,092028		0,039796
2		0,413758	0,424718	
3	1,000000			
4		1,000000		
5	0,041738	0,032950		
6	0,013350	0,136168		
7	0,475541	0,127901	0,183669	
8	0,488500	0,172400	0,124128	
9		0,144963	0,050210	
10			1,000000	
11		0,057333	0,743124	
12		0,169785	0,154616	
13				1,000000
14	0,152790	0,034338		
15	0,116903	0,094399		

sx		
	1	***2***
1		30,513358
2		88,912541
9		299,967397
11		1208,436315
12		11,973727

sy		
	1	***2***
5	14,554106	
6	379,650133	
14	36,587942	
15	310,042242	

Teraz ešte uvedieme definitívne výsledky efektívnosti jednotlivých producentov:

Producenti	**Neefektívny útvar projektujúci sa na pseudoefektívnu hranicu**	**Neefektívny útvar projektujúci sa na efektívnu hranicu**	**Efektívny útvar**
3			1
4			1
10			1
13			1
7		0,972376	
8		0,569856	
9	0,985773		
12	0,813778		
2	0,791889		
11	0,769442		
1	0,760992		
14	0,508529		
15	0,317163		
5	0,313128		
6	0,212442		

20 Computable General Equilibrium (CGE)

CGE (Computable General Equilibrium) modely, modely vypočítateľnej všeobecnej rovnováhy, simulujú správanie sa ekonomických subjektov na jednotlivých trhoch. Jedná sa o makroekonomické modely založené na výsledkoch mikroekonomickej teórie. Tieto modely môžu byť komparatívno-statické, kde uvažujeme len jedno časové obdobie alebo dynamické, teda vyvíjajúce sa v čase. Prvé menované používajú jednoduchší matematický aparát a skúmajú sa nimi dopady exogénnych šokov v ekonomike v rámci jedného uvažovaného obdobia. Dynamické modely sú komplikovanejšie, no zároveň poskytujú väčšie možnosti modelovania šokov v ekonomike. Dátovou základňou CGE modelov je Matica spoločenských účtov (SAM - Social Accounting Matrix) obsahujúca všetky nominálne toky v ekonomike. CGE modely sa môžu zostavovať ako systém nelineárnych rovníc alebo ako úloha komplementárneho programovania.

Teraz popíšeme princíp, na ktorom je založená vypočítateľnosť CGE modelu. SAM matica nám popisuje ekonomiku v danom období a my predpokladáme, že táto ekonomika je v rovnováhe. Vďaka tomuto predpokladu vieme pomocou údajov zo SAM matice nakalibrovať neznáme parametre jednotlivých produkčných funkcií a funkcií užitočnosti tak, aby jediným riešením CGE modelu s nakalibrovanými parametrami, bez exogénnych zásahov, boli hodnoty SAM matice. Týmto spôsobom získame model zaručujúci rovnováhu na trhoch a môžeme skúmať dopady exogénnych šokov v ekonomike. Ďalším princípom tvorby CGE modelov je princíp relatívnosti cien. Ceny statkov majú v CGE modeloch význam iba ak sú relatívne, a preto sa cena jednej komodity stanoví ako „numeraire“. Ostatné ceny sú vyjadrené pomerne k „numeraire“.

20.1 Konkrétny CGE model – Komparatívno-statický CGE model pre uzavretú ekonomiku

V tejto práci sa budeme zaoberať komparatívno-statickým modelom pre uzavretú ekonomiku pozostávajúcu z troch produkčných sektorov, jednej reprezentatívnej domácnosti a vlády. Model zostavíme ako systém nelineárnych rovníc.

20.1.1 Predpoklady uvažovaného CGE modelu

Predpokladáme, že všetky subjekty sa správajú racionálne. Všetky trhy v ekonomike sú dokonale konkurenčné a zároveň sú v rovnováhe. Inak povedané, ekonomika je vo všeobecnej rovnováhe. Každý produkčný sektor vyrába jeden tovar pomocou dvoch výrobných faktorov, a to kapitálu a práce. Produkcia je popísaná produkčnou funkciou s konštantnými výnosmi z rozsahu. Spotreba domácností je popísaná funkciou užitočnosti s konštantnými výnosmi z rozsahu. Model je založený na neoklasických predpokladoch. Z toho vyplýva, že v danej ekonomike neexistuje nezamestnanosť. Vláda poskytuje sociálne dávky domácnostiam vo forme priameho transferu. Príjmy domácností tvoria výnosy z kapitálu, mzda a transfer od vlády. Príjmy vlády tvoria výnosy z kapitálu a daň z pridanej hodnoty. Všetky jednotky kapitálu sú rovnako produktívne a všetci pracovníci sú rovnako produktívni. Výrobné faktory sú v rámci sektorov dokonale mobilné, náklady na prepravu komodít sú nulové. Štruktúra obyvateľstva sa nemení. Kapitál sa neamortizuje a zároveň nedochádza k jeho akumulácii, pretože všetko, čo sa vyrobí sa spotrebuje domácnosťami a vládou.

20.1.2 Princíp zostavenie uvažovaného CGE modelu a jeho matematická formulácia:

Názvy množín indexov vystupujúcich v modeli sú nasledovné, pričom ich prvky uvádzame vo formulácii modelu v jazyku GAMS aj s ich popisom:

indexy	–	indexy vystupujúce v SAM matici
se	–	produkčné sektory
ko	–	produkované komodity

Matica SAM je tiež uvedená v zdrojovom kóde modelu. Pre SAM maticu platia dva základné princípy. Prvým je princíp input-output modelov, t.j. výdavky jedného subjektu sú príjmami iného subjektu. Druhým je princíp národného účtovníctva, t.j. suma dôchodkov určitého subjektu sa rovná

sume jeho výdavkov. Bližšie informácie o SAM matici môžete získať v [8]. Matica SAM je definovaná takto:

$$SAM = (sam_{ij}) \qquad \text{, kde} \quad i, j \in indexy$$

Princíp odvodenia jednotlivých rovníc modelu a ich definitívny tvar:

Teraz si uvedieme iba princíp odvodenia jednotlivých rovníc pre produkčné sektory, domácnosti a vládu, keďže ich samotné odvodenie nie je triviálne. Následne sformulujeme konkrétny CGE model obsahujúci už rovnice vo finálnej podobe. Zároveň uvedieme aj vzorce na kalibráciu jednotlivých parametrov produkčných funkcií a funkcií užitočnosti nášho modelu:

Produkčný sektor:

Používame Cobb-Douglasovu produkčnú funkciu:

$$Y = \gamma K^{\alpha K} L^{\alpha L}$$

Keďže predpokladáme dokonalú konkurenciu na všetkých trhoch a zároveň racionálnosť všetkých subjektov, jednotlivé produkčné sektory budú minimalizovať náklady ($rK + wL$) pri daných cenách a daných produkciách ($Y = f(K,L)$). Takáto úloha má pri uvažovaných predpokladoch jediné riešenie, a tak dostávame podmienený dopyt po kapitále a podmienený dopyt po práci. Zároveň z predpokladu dokonale konkurenčných trhov vyplýva podmienka nulového zisku. Z analýzy produkčného sektora sme získali nasledujúce tri podmienky rovnováhy, pričom $j \in se$, r je cena kapitálu, w je cena práce a P_j je cena tovaru:

$$K_j = K_j\left(Y_j, r, w\right) \quad , \quad L_j = L_j\left(Y_j, r, w\right) \quad , \quad P_j Y_j = rK_j + wL_j$$

Na základe údajov zo SAM matice, ktoré všade v tejto kapitole označujeme pruhom nad prislúchajúcim symbolom, nakalibrujeme parametre podľa nasledujúcich vzorcov, pričom všetky ceny položíme rovné 1, a teda údaje zo SAM matice vyjadrujú množstvá:

$$\alpha K_j = \frac{\overline{r}\,\overline{K}_j}{\overline{r}\,\overline{K}_j + \overline{w}\,\overline{L}_j} \quad , \quad \alpha L_j = \frac{\overline{w}\,\overline{L}_j}{\overline{r}\,\overline{K}_j + \overline{w}\,\overline{L}_j} \quad , \quad \gamma_j = \frac{\overline{Y}_j}{\overline{K}_j^{\alpha K_j}\,\overline{L}_j^{\alpha L_j}}$$

Po dosadení nakalibrovaných parametrov do rovníc pre K_j a L_j dostaneme prvé tri bloky rovníc modelu, ktoré vyzerajú nasledovne:

$$K_j = \frac{\overline{K}_j}{r}\,\frac{Y_j}{\overline{Y}_j}\,r^{\alpha K_j}\,w^{\alpha L_j} \quad , \quad L_j = \frac{\overline{L}_j}{w}\,\frac{Y_j}{\overline{Y}_j}\,r^{\alpha K_j}\,w^{\alpha L_j} \quad , \quad P_j Y_j = rK_j + wL_j$$

Ďalej pre produkčný sektor platí, že celkové množstvo použitého kapitálu a práce sa nesmie zmeniť. Navyše musí byť splnená aj ďalšia podmienka modelu, ktorá by sa dala slovne naformulovať ako: „Čo sa v ekonomike vyprodukuje, to sa v nej aj spotrebuje.“ Takto nám vznikli ďalšie dve rovnice a jeden blok rovníc modelu, pričom H_j je spotreba j -tej komodity domácnosťami a G_j je spotreba j -tej komodity vládou a Y_j je produkcia j -teho sektora:

$$\sum_{j \in se} \overline{K}_j = \sum_{j \in se} K_j \quad , \quad \sum_{j \in se} \overline{L}_j = \sum_{j \in se} L_j \quad , \quad Y_j = H_j + G_j$$

Domácnosti:

Používame Cobb-Douglasovu funkciu užitočnosti. Spotrebiteľ je racionálny, a preto maximalizuje svoju užitočnosť $u(H)$, kde H je vektor obsahujúci spotrebu jednotlivých komodít domácnosťami. Zároveň musí byť splnené rozpočtové ohraničenie $\sum_{i \in ko} P_i H_i = MH$, kde MH sú príjmy domácností.

Takáto úloha má, podobne ako pre produkčný sektor, jediné riešenie, a to tvorí ďalší blok rovníc nášho

modelu, pričom $i \in ko$, αH_i je podiel spotreby i -tej komodity na celkovej spotrebe domácností a dph_i je sadzba dane z pridanej hodnoty pre i -tu komoditu:

$$H_i = \frac{\alpha H_i \, MH}{P_i \left(1 + dph_i\right)}$$

Ďalšou rovnicou modelu je rovnica pre príjmy domácností, pričom *THG* vyjadruje transfer od vlády do domácností (sociálne dávky) a βK je časť celkovej zásoby kapitálu vlastnená domácnosťami:

$$MH = \sum_{j \in se} w L_j + \beta K \sum_{j \in se} r K_j + w \, THG$$

Vláda:

Podobne ako v prípade domácností, vláda maximalizuje svoju užitočnosť, danú Cobb-Douglasovou funkciou užitočnosti, pri rozpočtových ohraničeniach. Riešenie takejto úlohy tvorí blok rovníc nášho modelu, pričom $i \in ko$, G_i je spotreba i -tej komodity vládou, *MG* sú príjmy vlády a αG_i je podiel spotreby i -tej komodity na celkovej spotrebe vlády:

$$G_i = \frac{\alpha G_i \, MG}{P_i}$$

Poslednou rovnicou modelu je rovnica pre príjmy vlády, pričom výraz $\left(1 - \beta K\right)$ predstavuje časť celkovej zásoby kapitálu vlastnenú vládou:

$$MG = \left(1 - \beta K\right) \sum_{j \in se} r K_j + \sum_{i \in ko} P_i \, dph_i \, H_i - w \, THG$$

Model: Teraz skompletizujeme všetky rovnice potrebné na zostavenie modelu. Ich počet je 22:

$$K_j = \frac{\overline{K}_j}{r} \frac{Y_j}{\overline{Y}_j} r^{\alpha K_j} w^{\alpha L_j} \quad , \quad L_j = \frac{\overline{L}_j}{w} \frac{Y_j}{\overline{Y}_j} r^{\alpha K_j} w^{\alpha L_j} \quad , \quad P_j Y_j = r K_j + w L_j$$

$$\sum_{j \in se} \overline{K}_j = \sum_{j \in se} K_j \quad , \quad \sum_{j \in se} \overline{L}_j = \sum_{j \in se} L_j \quad , \quad Y_j = H_j + G_j$$

$$H_i = \frac{\alpha H_i \, MH}{P_i \left(1 + dph_i\right)} \quad , \quad MH = \sum_{j \in se} w L_j + \beta K \sum_{j \in se} r K_j + w \, THG$$

$$G_i = \frac{\alpha G_i \, MG}{P_i} \quad , \quad MG = \left(1 - \beta K\right) \sum_{j \in se} r K_j + \sum_{i \in ko} P_i \, dph_i \, H_i - w \, THG$$

$$j \in se \qquad \text{a} \qquad i \in ko$$

Premenné: Ešte uvedieme, ktoré symboly z rovní sú premennými modelu. Ich počet je 25:

$$Y_j \, , \, K_j \, , \, L_j \, , \, MH \, , \, H_i \, , \, MG \, , \, G_i \, , \, dph_i \, , \, P_i \, , \, w \, , \, r$$

<u>Počiatočné vyriešenie modelu:</u>

Pomocou počiatočného vyriešenie modelu si vieme overiť správnosť zápisu rovníc, ktoré obsahuje. Aby sme mohli model vyriešiť musíme si ešte stanoviť „numeraire“. V našom modely sme zafixovali premennú *w* na hodnotu 1. Ešte treba zafixovať aj premenné dph_i na hodnoty $\overline{dph}_i$ (sadzba dane z pridanej hodnoty pre i -tu komoditu, vypočítaná z údajov SAM matice). Týmto krokom sme docielili

rovnosť medzi počtom rovníc a počtom neznámych. Teraz už môžeme vyriešiť náš počiatočný model pomocou GAMS-u. V GAMS-e treba riešiť optimalizačnú úlohu, kde maximalizujeme konštantu pri ohraničeniach daných naším systémom nelineárnych rovníc.

Scenár zvýšenia DPH a sociálnych dávok:

V tomto scenári sa rozhodneme skúmať dopady zvýšenia sociálnych dávok domácnostiam o 20% a súčasného zvýšenia sadzieb DPH pre jednotlivé komodity v uvažovanej ekonomike. Takýchto scenárov by sa dalo vymyslieť mnoho, nám však stačí na ilustráciu možností jazyka GAMS len tento jeden. Výsledky scenára sa prezentujú ako percentuálne zmeny jednotlivých premenných, t.j.:

$$zmena\ premennej\ v\ \% \quad = \quad \frac{nova\ hodnota\ premennej \quad - \quad pociatocna\ hodnota}{pociatocna\ hodnota} \quad * \quad 100\ \%$$

20.1.3 Formulácia uvažovaného CGE modelu v jazyku GAMS

Uvádzame ilustráciu CGE modelu naprogramovaného v užívateľskom prostredí jazyka GAMS:

```
$eolcom //
  set indexy "indexy vystupujúce v SAM matici"     / po   "poľnohospodárstvo"
                                                     pr   "priemysel"
                                                     s    "služby"
                                                     K    "kapitál"
                                                     L    "práca"
                                                     DPH  "daň z pridanej hodnoty"
                                                     H    "domácnosti"
                                                     G    "vláda"/;
  alias(indexy,indexyl);

  set ko[indexy]   "podmnožina množiny indexy - komodity" / po , pr , s /;
  alias(ko,se); // "podmnožina množiny indexy - sektory"  / po , pr , s /

  table
      SAM[indexy,indexyl] "Social Accaunting Matrix - Matica spoločenských účtov"
                    po     pr      s      K     L    DPH      H      G
            po                                              40      5
            pr                                              45     15
            s                                               35     10
            K       10     30     20
            L       30     20     20
            DPH     5      10     5
            H                            40    70                  10
            G                            20           20                ;

  parameters
   // FIRMY
      p_Y[se]     "počiatočná produkcia v jednotlivých sektoroch"
      p_K[se]     "počiatočný dopyt po kapitále v jednotlivých sektoroch"
      p_L[se]     "počiatočný dopyt po práci v jednotlivých sektoroch"
```

```
      alfa_K[se]  "podiel kapitálu v produkčnej funkcii jednotlivých sektorov"
      alfa_L[se]  "podiel práce v produkčnej funkcii jednotlivých sektorov"
   // DOMÁCNOSTI
      p_p_H       "počiatočný príjem domácností"
      p_s_H[ko]   "počiatočná spotreba jednotlivých komodít v sektore domácností"
      alfa_H[ko]  "podiel jednotlivých komodít v spotrebe sektoru domácností"
   // VLÁDA
      p_p_G       "počiatočný príjem vlády"
      p_s_G[ko]   "počiatočná spotreba jednotlivých komodít v sektore vlády"
      alfa_G[ko]  "podiel jednotlivých komodít v spotrebe sektoru vlády"
   // OSTATNÉ PARAMETRE
      trans_H_G   "transfér od vlády do domácností - sociálne dávky"
      p_dph[ko]   "sadzba DPH pre jednotlivé komodity"
      p_TK        "celková zásoba kapitálu"
      p_TL        "celková ponuka práce"
      beta_K      "časť celkovej zásoby kapitálu vlastnená domácnosťami" ;
//=============================================================================================
* KALIBRÁCIA MODELU
   trans_H_G  = SAM["H","G"];
   p_K[se]    = SAM["K",se];
   p_L[se]    = SAM("L",se);
   p_Y[se]    = p_K[se] + p_L[se]  ;
   alfa_K[se] = p_K[se]/p_Y[se];
   alfa_L[se] = p_L[se]/p_Y[se];
   p_p_H      = SAM["H","K"] + SAM["H","L"] + trans_H_G;
   p_s_H[ko]  = SAM[ko,"H"] - SAM["DPH",ko];
   alfa_H[ko] = SAM[ko,"H"]/sum{se, SAM[se,"H"]};
   p_p_G      = SAM["G","K"] + SAM["G","DPH"] - trans_H_G;
   p_s_G[ko]  = SAM[ko,"G"];
   alfa_G[ko] = p_s_G[ko]/sum{se, p_s_G[se]};
   p_dph[ko]  = SAM["DPH",ko]/p_s_H[ko];
   p_TK       = sum{se, p_K[se]};
   p_TL       = sum{se, p_L[se]};
   beta_K     = SAM["H","K"]/p_TK;
//=============================================================================================
* KONŠTRUKCIA MODELU
  positive variables
      Y[se]     "produkcia v jednotlivých sektoroch"
      K[se]     "dopyt po kapitále v jednotlivých sektoroch"
      L[se]     "dopyt po práci v jednotlivých sektoroch"
      p_H       "príjem domácností"
      s_H[ko]   "spotreba jednotlivých komodít v sektore domácností"
      p_G       "príjem vlády"
      s_G[ko]   "spotreba jednotlivých komodít v sektore vlády"
      dph[ko]   "daň z pridanej hodnoty pre jednotlivé komodity"
      P[ko]     "cena jednotlivých komodít"
      w         "cena práce"
      r         "cena kapitálu" ;
  free variable
      h_uf      "hodnota fiktívnej účelovej funkcie" ;

  equations
      r_d_K[se]  "rovnica dopytu po kapitále v jednotlivých sektoroch"
      r_d_L[se]  "rovnica dopytu po práci v jednotlivých sektoroch"
      r_d_H[ko]  "rovnica dopytu po jednotlivých komoditách v sektore domácností"
      r_d_G[ko]  "rovnica dopytu po jednotlivých komoditách v sektore vlády"
      r_n_z[se]  "rovnica nulového zisku v jednotlivých sektoroch"
      r_r_t[ko]  "rovnica rovnováhy na trhoch s jednotlivými komoditami"
      r_r_t_L    "rovnica rovnováhy na trhu práce"
      r_r_t_K    "rovnica rovnováhy na kapitálovom trhu"
      r_p_H      "rovnica pre príjmy domácností"
      r_p_G      "rovnica pre príjmy vlády"
```

```
      uf         "fiktívna účelová funkcia" ;
    uf .. h_uf =E= 1;
    r_d_K[se] ..   K[se] =E= p_K[se]/r * Y[se]/p_Y[se] * w**alfa_L[se] * r**alfa_K[se];
    r_d_L[se] ..   L[se] =E= p_L[se]/w * Y[se]/p_Y[se] * w**alfa_L[se] * r**alfa_K[se];
    r_d_H[ko] ..   s_H[ko] =E= alfa_H[ko]/(P[ko]*(1+dph[ko])) * p_H;
    r_d_G[ko] ..   s_G[ko] =E= alfa_G[ko]/P[ko] * p_G;
    r_n_z[se] ..   P[se]*Y[se] =E= w*L[se] + r*K[se];
    r_r_t[ko] ..   Y[ko] =E= s_H[ko] + s_G[ko];
    r_r_t_L   ..   p_TL =E= sum{se,L[se]};
    r_r_t_K   ..   p_TK =E= sum{se,K[se]};
    r_p_H    .. p_H =E= sum{se, w*L[se]} + beta_K*sum{se,r*K[se]} + w*trans_H_G;
    r_p_G    .. p_G =E= (1-beta_K)*sum{se,r*K[se]} + sum{ko,P[ko]*dph[ko]*s_H[ko]} - w*trans_H_G

  model CGE /all/;
//=============================================================================================
* POČIATOČNÉ SPUSTENIE MODELU (OVERENIE SPRÁVNOSTI JEHO ZOSTAVENIA)
    w.fx = 1;
    dph.fx[se]=p_dph[se];
   // nasledujúcimi príkazmi určíme štartovací bod pre solver
    Y.l[se]   = p_Y[se];
    K.l[se]   = p_K[se];
    L.l[se]   = p_L[se];
    p_H.l     = p_p_H;
    s_H.l[ko] = p_s_H[ko];
    p_G.l     = p_p_G;
    s_G.l[ko] = p_s_G[ko];
    P.l[ko]   = 1;
    r.l       = 1;
  solve CGE using nlp maximizing h_uf;
//=============================================================================================
* SCENÁR ZVÝŠENIA VLÁDNYCH VÝDAVKOV NA SOCIÁLNU POLITIKU A ZÁROVEŇ ZVÝŠENIA DPH
    trans_H_G    = 1.2*trans_H_G;
    dph.fx["po"] = 1.05*p_dph["po"];
    dph.fx["pr"] = 1.1 *p_dph["pr"];
    dph.fx["s"]  = 1.1 *p_dph["s"];
  solve CGE using nlp maximizing h_uf;
//=============================================================================================
* VYHODNOTENIE PERCENTUÁLNYCH ZMIEN PREMENNÝCH MODELU
  parameters
      d_Y[se]    "percentuálna zmena produkcie v jednotlivých sektoroch"
      d_K[se]    "percentuálna zmena dopytu po kapitále v jednotlivých sektoroch"
      d_L[se]    "percentuálna zmena dopytu po práci v jednotlivých sektoroch"
      d_p_H      "percentuálna zmena príjmu domácností"
      d_s_H[ko]  "percentuálna zmena spotreby jednotlivých komodít v sektore domácností"
      d_p_G      "percentuálna zmena príjmu vlády"
      d_s_G[ko]  "percentuálna zmena spotreby jednotlivých komodít v sektore vlády"
      d_dph[ko]  "percentuálna zmena dph pre jednotlivé komodity"
      d_P[ko]    "percentuálna zmena ceny jednotlivých komodít"
      d_w        "percentuálna zmena ceny práce"
      d_r        "percentuálna zmena ceny kapitálu" ;
   d_Y[se]   = (Y.l[se]-p_Y[se])/p_Y[se] *100;
   d_K[se]   = (K.l[se]-p_K[se])/p_K[se] *100;
   d_L[se]   = (L.l[se]-p_L[se])/p_L[se] *100;
   d_p_H     = (p_H.l-p_p_H)/p_p_H *100;
   d_s_H[ko] = (s_H.l[ko]-p_s_H[ko])/p_s_H[ko] *100;
   d_p_G     = (p_G.l-p_p_G)/p_p_G *100;
   d_s_G[ko] = (s_G.l[ko]-p_s_G[ko])/p_s_G[ko] *100;
   d_dph[se] = (dph.l[se]-p_dph[se])/p_dph[se] *100;
   d_P[ko]   = (P.l[ko]-1) *100;
   d_w       = (w.l-1) *100;
   d_r       = (r.l-1) *100;
display d_Y , d_K , d_L , d_p_H , d_s_H , d_p_G , d_s_G , d_dph , d_P , d_w , d_r ;
```

20.1.4 Výsledky poskytnuté GAMS-om

Uvádzame výstup príkazu `display`, upravený tak, aby zaberal menej miesta:

```
-- PARAMETER d_Y  percentuálna zmena produkcie v jednotlivých sektoroch
     po  0.751,    pr -0.589,    s  -0.016

-- PARAMETER d_K  percentuálna zmena dopytu po kapitále v jednotlivých sektoroch
     po  1.020,    pr -0.447,    s   0.161

-- PARAMETER d_L  percentuálna zmena dopytu po práci v jednotlivých sektoroch
     po  0.662,    pr -0.800,    s  -0.193

-- PARAMETER d_p_H  =  1.549  percentuálna zmena príjmu domácností

-- PARAMETER d_s_H  perc. z. spotreby jednotlivých komodít v sektore domácností
     po  1.007,    pr -0.447,    s   0.296

-- PARAMETER d_p_G  = -1.129  percentuálna zmena príjmu vlády

-- PARAMETER d_s_G  perc. zmena spotreby jednotlivých komodít v sektore vlády
     po -1.041,    pr -0.918,    s  -0.953

-- PARAMETER d_dph  percentuálna zmena dph pre jednotlivé komodity
     po  5.000,    pr 10.000,    s  10.000

-- PARAMETER d_P  percentuálna zmena ceny jednotlivých komodít
     po -0.089,    pr -0.213,    s  -0.177

-- PARAMETER d_w   =  0.000  percentuálna zmena ceny práce
   PARAMETER d_r   = -0.354  percentuálna zmena ceny kapitálu
```

21 Model Markowitzovej teórie portfólia

Pomocou modelu Markowitzovej teórie portfólia sa zostavuje portfólio (akcie, dlhopisy, ...) s minimálnym rizikom za podmienky dosiahnutia aspoň nami stanoveného výnosu. My sa budeme v tejto kapitole zaoberať klasickým modelom Markowitzovej teórie portfólia, v ktorom sa očakávaný výnos jednotlivých zložiek portfólia stanovuje ako aritmetický priemer z historických dát a riziko portfólia reprezentuje jeho variancia.

21.1 Všeobecný tvar modelu Markowitzovej teórie portfólia

21.1.1 Matematická formulácia modelu

Aj v tomto prípade je potrebné najskôr zadefinovať množiny indexov. Všetky historické časové okamihy, z ktorých máme dáta (ceny akcií, dlhopisov, ...), budú obsiahnuté v množine nazvanej *t* a jednotlivé zložky finančného trhu (ďalej len zložky) bude obsahovať množina *n* (akcie, dlhopisy, ...):

$$t = \{0, \ldots, T\} \qquad n = \{1, \ldots, N\}$$

Teraz už môžeme definovať maticu cien zložiek *X*, ktorej riadkové indexy budú z množiny *t* a stĺpcové z množiny *n*:

$$X = (x_{ij}) \qquad \text{, kde} \quad i \in t \quad a \quad j \in n$$

Na zaznamenanie nami stanoveného výnosu portfólia použijeme skalár, ktorý sme nazvali *er*.

Na konštrukciu modelu Markowitzovej teórie portfólia je ešte potrebné vypočítať viaceré parametre. Jedná sa o maticu relatívnych mesačných výnosov jednotlivých zložiek *R*, kde riadkové indexy sú z množiny *t* a stĺpcové z *n*. Ďalej potrebujeme vektor priemerných mesačných výnosov jednotlivých zložiek *ar* viažuci sa na množinu *n* a variančno-kovariančnú maticu *COV* vyjadrujúcu kovariancie medzi jednotlivými zložkami:

$$R = (r_{ij}) \qquad ar = (ar_j) \qquad COV = (\mathrm{cov}_{kj}) \qquad \text{, kde} \quad i \in t \wedge i \neq 0, \quad j \in n \quad \text{a} \quad k \in n$$

Relatívny mesačný výnos *j* -tej zložky v *i* -tom mesiaci r_{ij} vypočítame podľa nasledujúceho vzorca:

$$r_{ij} = \frac{x_{ij} - x_{i-1\ j}}{x_{i-1\ j}}$$

Priemerný mesačný výnos *j* -tej zložky vypočítame podľa nasledujúceho vzorca, pričom $|t|$ vyjadruje počet prvkov množiny *t*:

$$ar_j = \frac{1}{|t| - 1} \sum_{\substack{i \in t \\ i \neq 0}} r_{ij}$$

Kovarianciu medzi *k* -tou a *j* -tou zložkou vypočítame podľa nasledujúceho vzorca:

$$\mathrm{cov}_{kj} = \frac{1}{|t| - 2} \sum_{\substack{i \in t \\ i \neq 0}} (r_{ik} - ar_k)(r_{ij} - ar_j)$$

Neznámymi hodnotami sú váhy jednotlivých zložiek v portfóliu. Vektor váh, viažuci sa na množinu *n*, sme si označili *w*:

$$w = (w_j) \qquad \text{, kde} \quad j \in n$$

Teraz už môžeme sformulovať model. Minimalizujeme varianciu portfólia, pričom očakávaný výnos portfólia musí byť väčší alebo rovný nami stanovenému výnosu a súčet váh sa musí rovnať 1:

$$\min \quad \sum_{\substack{k \in n \\ j \in n}} w_k \; w_j \; \mathrm{cov}_{k\,j}$$

$$\sum_{j \in n} w_j \; ar_j \geq er$$

$$\sum_{j \in n} w_j = 1$$

$$w_j \geq 0 \qquad , \quad j \in n$$

21.2 Konkrétny model Markowitzovej teórie portfólia

21.2.1 Zadanie úlohy

Je koniec decembra roku 2006 a my disponujeme istým obnosom, ktorý chceme investovať do akcií tak, aby sme dosiahli aspoň 10% ročný výnos. Na trhu sa obchoduje s akciami troch rôznych firiem. Tieto akcie si označíme ako a1, a2, a3. Historický vývoj ich cien je znázornený v nasledujúcej tabuľke, pričom historické obdobie je jeden rok a ceny sú vždy z konca prislúchajúceho mesiaca (0 predstavuje koniec decembra 2005):

Mesiac	a1	a2	a3
0	18.10	102.81	91.44
1	22.04	99.50	95.69
2	19.81	100.44	93.62
3	22.12	111.31	92.69
4	22.06	101.62	95.06
5	26.44	108.00	94.69
6	29.00	108.75	94.94
7	25.19	111.25	100.19
8	27.00	110.37	97.31
9	30.69	115.32	92.88
10	24.94	113.27	97.44
11	27.88	118.69	95.75
12	29.20	114.75	96.81

Našou úlohou je zistiť do ktorých akcií a akú časť z finančných prostriedkov máme investovať, aby náš očakávaný ročný výnos portfólia bol aspoň 10% pri minimálnom riziku.

21.2.2 Matematická formulácia modelu

Všetky mesiace zo zadania budú obsiahnuté v množine *t* z formulácie všeobecného modelu ($T = 12$) a jednotlivé akcie bude obsahovať množina *n* ($N = 3$):

$$t = \{0 \,, \ldots, 12\} \qquad n = \{1 \,, 2 \,, 3\}$$

Maticu cien akcií *X* nebudeme na tomto mieste uvádzať s konkrétnymi číselnými údajmi, keďže sa nachádza vo formulácii modelu v jazyku GAMS a zároveň aj v zadaní úlohy:

Na zaznamenanie nami stanoveného ročného výnosu 10% použijeme parameter *er*, ktorý ešte upravíme na mesačný výnos potrebný v modeli Markowitzovej teórie portfólia:

$$er := \frac{er}{12}$$

21.2.3 Formulácia modelu v jazyku GAMS

Uvádzame ilustráciu modelu Markowitzovej teórie portfólia naprogramovaného v užívateľskom prostredí jazyka GAMS:

```
sets
    t     "mesiace"          /0*12/
    n     "zložky portfólia" /1,2,3/
    tl[t] "množina 't' bez prvku 0";
  tl[t]=yes;
  tl['0']=no;
  alias(n,nl);

table
    x[t,n] "cena jednotlivých zložiek v jednotlivých časoch"
               1         2         3
        0    18.1     102.81    91.44
        1    22.04     99.5     95.69
        2    19.81    100.44    93.62
        3    22.12    111.31    92.69
        4    22.06    101.62    95.06
        5    26.44    108       94.69
        6    29       108.75    94.94
        7    25.19    111.25   100.19
        8    27       110.37    97.31
        9    30.69    115.32    92.88
       10    24.94    113.27    97.44
       11    27.88    118.69    95.75
       12    29.2     114.75    96.81   ;

scalar
    er "očakávaný ročný výnos" /0.1/;
  er=er/12;

parameters
    r[t,n]    "relatívny mesačný výnos v 'i'-tom mesiaci kde 'i' je z 't' "
  ar[n]=sum(tl,r[tl,n])/card(tl);
  cov[n,nl]=sum(tl,(r[tl,n]-ar[n])*(r[tl,nl]-ar[nl]))/(card(tl)-1);

positive variable
    w[n] "váhy jednotlivých zložiek v portfóliu";
free variable
    var  "variancia portfólia = hodnota účelovej funkcie";

equations
    vf_min "definícia účelovej funkcie"
    ere    "ohraničenie pre očakavaný výnos"
    sw     "podmienka jednotkového súčtu váh";
  vf_min ..   var =e= sum([n,nl],w[n]*w[nl]*cov[n,nl]);
  ere    ..           sum(n,w[n]*ar[n]) =g= er;
  sw     ..                 sum(n,w[n]) =e= 1;

model markowitz /vf_min,ere,sw/;

solve markowitz minimizing var using nlp;

display w.l;
```

21.2.4 Výsledky poskytnuté GAMS-om

Uvádzame výstup príkazu `display`:

```
----     62 VARIABLE w.L  váhy jednotlivých zložiek v portfóliu

1  0.055,    2  0.292,    3  0.653
```

Záver

V tejto diplomovej práci sme sa venovali teoretickému popisu a praktickému použitiu všeobecného algebraického modelovacieho systému GAMS.

Cieľom prvej časti diplomovej práce bolo vypracovať stručný ale zároveň dostatočne obsažný popis jazyka GAMS. Na základe presnej syntaxe ale aj jednoduchých príkladov sme priblížili štruktúru tohto programovacieho jazyka. Pomocou jeho nástrojov popísaných v prvej časti diplomovej práce je možné naprogramovať nielen jednoduchý ale aj značne komplikovaný model v jazyku GAMS.

Cieľom druhej časti diplomovej práce bolo prezentovať možnosti praktického využitia jazyka GAMS prostredníctvom rôznorodých úloh relatívne malých rozmerov. Široké spektrum využitia jazyka GAMS sme ukázali nielen na „jednoduchých" optimalizačných príkladoch, ktoré sa bežne vyučujú na vysokých školách s matematickým a ekonomickým zameraním na predmetoch finančná matematika, lineárne programovanie, ekonometria, štatistika ale aj na príkladoch, ktoré nie sú štandardnou učebnou látkou – DEA analýza, CGE model. Tieto modely sú súčasťou moderného trendu v ekonomickom modelovaní, no bežne sa im na prednáškach nevenuje veľa priestoru.

Cieľom tretej časti diplomovej práce (uvedená v prílohe) bolo spracovanie jedného komplexného problému aplikovateľného v praxi. Vybrali sme si problém analyzovaný v [6], patriaci do oblasti finančnej matematiky. Tento model nám dal možnosť ukázať, že GAMS dokáže nielen vypočítať optimálne hodnoty maximalizačnej, resp. minimalizačnej úlohy pri určitých ohraničeniach, ale je vhodný aj na modelovanie problémov, pri ktorých je potrebné používať iteračné procesy, zložité logické vetvenia a logické podmienky, simulácie rôznych procesov, dynamickosť v rámci modelu a mnohé ďalšie charakteristiky komplikovaných modelov. Zároveň náš program dokumentuje dôležitú črtu jazyka GAMS, a tou je nezávislosť formulácie modelu od jeho dátovej základne. Program sme napísali tak, aby bolo možné prepočítavať model podľa užívateľom zadaných hodnôt – začiatku a dĺžky obdobia investovania, historického obdobia, ktoré berieme do úvahy pri výpočte očakávaného výnosu, periódy prehodnocovania portfólia, metódy výpočtu očakávaného výnosu, ... V tomto programe sme použili všetky nástroje jazyka GAMS diskutované v prvej časti diplomovej práce. Okrem nich sme využili aj nástroje, ktoré nie sú podrobne popísané, sú len spomenuté v kapitole 15 spolu s odkazom na literatúru obsahujúcu podrobnejšie informácie.

Vo vyspelých krajinách sveta sa všeobecný algebraický modelovací systém vyučuje na viacerých vysokých školách a bežne sa používa na rozmanité účely. Na Slovensku zatiaľ GAMS nie je príliš známy a používaný, i my sme sa s ním stretli po prvýkrát pri písaní diplomovej práce, ale predpokladáme, že táto skutočnosť sa v dohľadnej dobe zmení. Dúfame, že touto diplomovou prácou prispejeme aspoň malou mierou k jeho rozšírenejšiemu používaniu na Slovensku.

Literatúra

Použitá literatúra

Prvá časť diplomovej práce

[1] BROOKE A., KENDRICK D., MEERAUS A., RAMAN R.: *GAMS (A User's guide)*, GAMS Development Corporation, 1998, dostupné na internete: http://www.gams.com/docs/gams/GAMSUsersGuide.pdf

[2] MCCARL B. A.: *GAMS User Guide: 2004 (Version 21.3)*, 2004, dostupné na internete: http://www.gams.com/dd/docs/bigdocs/gams2002/mccarlgamsuserguide.pdf

[3] KALVELAGEN E.: *Interfacing GAMS with other applications*, GAMS Development Corporation, tento dokument je neustále aktualizovaný, dostupné na internete: http://www.gams.com/gamsapp/appmain.htm

[4] GAMS DEVELOPMENT CORPORATION: *GAMS GDX facilities and tools*, 2004, dostupné na internete: http://www.gams.com/dd/docs/gams/gdxutils.pdf

[5] GAMS DEVELOPMENT CORPORATION: *GAMS IDE (Documentation)*, 1997-2003, dostupné na internete: http://www.gams.com/docs/gamside.pdf

Druhá časť diplomovej práce

[6] HAJDUKOVÁ B.: *Kalibrácia modelu na riadenie portfólia*, diplomová práca, 2006, FMFI UK

[7] HALICKÁ M.: *DEA modely*, voliteľné prednášky, 4. roč., 2004, FMFI UK

[8] KOTOV M.: *Modely všeobecnej ekonomickej rovnováhy*, diplomová práca, 2002, FMFI UK

[9] KRIVANSKÁ K.: *Ekonometrické modelovanie zahraničného obchodu SR*, diplomová práca, 2002, FMFI UK

[10] MELICHERČÍK I.: *Finančná matematika I*, prednášky, 3. roč., 2004, FMFI UK

[11] PLESNÍK J.: *Lineárne programovanie*, prednášky, 2. roč., 2002, FMFI UK

[12] SEKEREŠ S.: *Teória statických a dynamických CGE modelov*, diplomová práca, 2006, FMFI UK

[13] ŠTULAJTER F.: *Štatistické metódy*, prednášky, 3. roč., 2003, FMFI UK

[14] TOMA V.: *Rozhodovacie techniky v operačnom manažmente*, voliteľné prednášky, 4. roč., 2004, FMFI UK

[15] WITKOVSKÝ V.: *Ekonometria*, prednášky, 3. roč., 2004, FMFI UK

[16] ZVÁRA K., ŠTĚPÁN J.: *Pravděpodobnost a matematická statistika*, 1997, MATFYZPRESS – Vydavatelství Matematicko-fyzikální fakulty Univerzity Karlovy

Literatúra súvisiaca s témou diplomovej práce

http://www.gams.com – oficiálna internetová stránka spoločnosti GAMS Development Corporation

http://www.gams.com/docs/brochure.pdf – oficiálna brožúra obsahujúca základné informácie o GAMS-e

http://www.gams.com/docs/intro.htm – internetová stránka obsahujúca základné informácie o GAMS-e

http://www.gams.com/docs/example.htm – ilustračný príklad modelovania v GAMS-e (dopravná úloha)

http://www.gams.com/modtype/index.htm – všetky typy modelov riešiteľné pomocou GAMS-u

http://www.gams.com/docs/release/release.htm – popis zmien v rámci rôznych verzií GAMS-u

http://www.gams.com/docs/privacy.pdf – dokument obsahujúci informácie o zabezpečení programov

http://www.gams.com/docs/gams/win-install.pdf – dokument obsahujúci informácie o inštalácii GAMS-u

http://www.gams.com/mccarl/useide.pdf – dokument obsahujúci informácie o prostredí GAMS-u (IDE)

http://www.gams.com/solvers/index.htm – internetová stránka obsahujúca zoznam solverov GAMS-u a súčasne odkazy na ich manuály vo formáte .pdf

Prílohy

Príloha 1 – Spracovanie komplexného problému

V tejto prílohe spracujeme komplexný problém analyzovaný v inej diplomovej práci [6], patriacej do oblasti finančnej matematiky. V práci sa testuje vplyv rôznych metód výpočtu očakávaného výnosu na vývoj hodnoty portfólia počas daného časového obdobia, pričom toto portfólio je najskôr zostavované pomocou modelu Markowitzovej teórie portfólia a potom pomocou modelu s kvadratickou funkciou užitočnosti na základe historických dát. Inými slovami, jedná sa o kalibráciu daných modelov. V rámci každého z nich sú skúmané dve stratégie, a to stratégia buy & hold a stratégia pravidelného prehodnocovania portfólia. Na tomto mieste uvádzame oba programy (ku každému modelu jeden) napísané v jazyku GAMS, pomocou ktorých sa uvedené analýzy dajú realizovať. Najskôr si priblížime postup kalibrácie v programoch, ktorý je pre oba modely rovnaký a potom sa budeme venovať týmto modelom jednotlivo.

Postup kalibrácie v programoch

Na to, aby sme mohli vysvetliť kalibráciu modelov v programoch, musíme najskôr popísať ako prebieha proces investovania v našom programe. Na začiatku máme k dispozícii historické dáta obsahujúce časové rady cien jednotlivých finančných aktív. Naše dáta sú však vygenerované náhodne a uvažujeme o nich ako o denných. Teraz si zvolíme deň, v ktorom po prvýkrát zostavíme portfólio na základe poznatkov o vývoji cien finančných aktív z minulosti, respektíve na základe hodnoty očakávaného výnosu jednotlivých aktív (môže byť vypočítaný rôznymi spôsobmi). Zvolením časovej bázy modelu určíme o aký výnos sa jedná (ročný, mesačný, desaťdenný, ...). Po zostavení prvého portfólia sa posunieme v čase o periódu prehodnocovania, pričom počas preskočeného obdobia sledujeme vývoj jeho hodnoty. Pri druhom zostavovaní sa náš program vetví na dve horeuvedené stratégie investovania. V stratégii buy & hold sa počiatočné portfólio už nebude meniť až do konca behu programu a v prípade druhej stratégie ho prehodnotíme na základe očakávaných výnosov vypočítaných z aktuálneho historického obdobia, tou istou metódou ako v prvej iterácii. Opäť sa posunieme o periódu prehodnocovania, pričom sledujeme hodnoty oboch portfólií. Uvedený proces opakujeme, pokiaľ sa nachádzame v rámci nami stanoveného obdobia. Po skončení behu programu si môžeme pozrieť výsledky výpočtov GAMS-u exportované do Excelu.

Na realizáciu uvedeného myšlienkového postupu sme potrebovali zadeklarovať a zadefinovať modifikovateľné parametre nachádzajúce sa na začiatku programov medzi riadkami označenými znakmi `#######…` . Všetky teraz popíšeme:

Skalár `start` predstavuje deň, v ktorom po prvýkrát zostavíme portfólio.

Skalár `hist` predstavuje počet dní dozadu od okamihu, v ktorom sa práve nachádzame, čím definujeme dĺžku historického obdobia uvažovaného pri výpočtoch očakávaného výnosu. Ak by sme chceli vždy počítať s celým dostupným historickým obdobím, treba do `hist` priradiť číslo presahujúce celkový počet časových okamihov, z ktorých máme dáta. V našom prípade ide o 1654 časových okamihov.

Skalár `dlzka` predstavuje dĺžku obdobia od prvého zostavenia portfólia po deň, keď ukončíme investovanie.

Skalár `baza` predstavuje časovú bázu modelu. Toto číslo hovorí o tom, koľkodenný očakávaný výnos budeme počítať z dátovej základne, ktorú v našom prípade tvoria denné dáta.

Skalár `pd` predstavuje počet obchodovacích dní v roku. Potrebujeme ho na prepočítavanie ročného výnosu na výnos podľa časovej bázy modelu a naopak.

Skalár `per_preh` predstavuje počet dní medzi dvoma po sebe nasledujúcimi prehodnocovaniami.

Skalár `m` predstavuje výšku počiatočných investícií.

Modifikáciou skalára `vynos` sa prepíname medzi naprogramovanými metódami výpočtu očakávaného výnosu podľa nasledujúcej schémy:

priemerný výnos => `vynos = 1`

komulovaný výnos	=> `vynos = 2`
kĺzavé priemery	=> `vynos = 3`
exponenciálna regresia	=> `vynos = 4`
faktorový model	=> `vynos = 5`
CAPM - indexy cez meny	=> `vynos = 6`
CAPM - indexy cez maturity	=> `vynos = 7`
AR(1) proces	=> `vynos = 8`

Všetko, čo sa týka výpočtov jednotlivých výnosov, je naprogramované v blokoch programu nachádzajúcich sa medzi riadkami označenými znakmi `******`... . Prvý takýto blok obsahuje deklarácie a definície potrebných množín, parametrov, ... a druhý konkrétne vzorce a modely na výpočet príslušných výnosov. Ešte by sme chceli upozorniť na výnimočnosť metódy faktorový model. Prostredníctvom netriviálnych zásahov do modelov, dokáže užívateľ meniť počet faktorov. Inak povedané, pomocou tohto programu vieme vyriešiť jednofaktorový, dvojfaktorový, ... , *n* -faktorový model, kde *n* je prirodzené číslo.

Hlavnou myšlienkou kalibrácie jednotlivých modelov je sledovanie vývoja hodnoty portfólia v závislosti od metódy výpočtu očakávaného výnosu, pričom ostatné parametre analýzy sú rovnaké. V citovanej diplomovej práci sa táto analýza robila len raz, a teda všetky parametre, okrem skalára `vynos`, boli zafixované. V našich programoch sú všetky parametre modifikovateľné, pretože sme ich chceli urobiť čo najflexibilnejšie a čo najvšeobecnejšie, aby v plnej miere vynikli možnosti jazyka GAMS.

Model Markowitzovej teórie portfólia

Vstupné parametre špecifické pre tento model:

Skalár `er` predstavuje očakávaný ročný výnos celého portfólia, ktorý si stanovíme podľa našej averzie k riziku.

Matematická formulácia modelu:

Matematická formulácia tohto modelu sa od formulácie modelu Markowitzowej teórie portfólia, z kapitoly 21, odlišuje len pridaním rozmeru času, a preto ju nebudeme na tomto mieste uvádzať.

Formulácia modelu v jazyku GAMS:

Teraz uvedieme zdrojový kód programu pre model Markowitzovej teórie portfólia:

```
$eolcom //


//#############################################################################
* ZADÁVANIE VSTUPNÝCH PARAMETROV MODELU
  scalars
      start    "deň začiatku investovania" /200/
      hist     "obdobie z ktorého máme historické dáta (v dňoch)" /100/
            // ak chcete celú históriu, stačí do "hist" priradiť číslo väčšie
            // ako počet dní v množine t
      dlzka    "dĺžka obdobia investovania (v dňoch)" /1000/
            // teda obdobia, počas ktorého buď držíme alebo prehodnocujeme
            // portfólio
      baza     "časová báza modelu (v dňoch)" /5/
            // vyjadruje na akej časovej báze (v dňoch) sa pracuje s dátami
      pd       "počet dní v roku" /250/
      per_preh "perióda prehodnocovania (v dňoch)" /5/
      m        "výška investícií" /1000000/
      vynos    "typ očakávaného výnosu" /1/                                  ;
            // parameter vypovedajúci o spôsobe počítania očakávaného výnosu

    //priemerný výnos            => vynos= 1
```

```
    //komulovaný výnos           => vynos= 2
    //kĺzavé priemery            => vynos= 3
    //exponenciálna regresia     => vynos= 4
    //faktorový model            => vynos= 5
    //CAPM - indexy cez meny     => vynos= 6
    //CAPM - indexy cez maturity => vynos= 7
    //AR(1) proces               => vynos= 8
  scalar
      er       "očakávaný ročný výnos" /0.1/;
//#############################################################################



$offsymxref offsymlist offuelxref offuellist onlisting
  options limrow=0,limcol=0,solprint=off,decimals=3;

//=============================================================================
//=============================================================================
***                  KONŠTRUKCIA MODELU MARKOWITZOVEJ TEÓRIE PORTFÓLIA

//=============================================================================
** MNOŽINY POUŽÍVANÉ V MODELI
  sets
      n      "zložky trhu"
      t      "množina časových okamihov (denné)"
      t_b[t] "množina časových okamihov podľa bázy modelu"
      orm[t] "okamih riešenia modelu";

//###############################################
$call "Gdxxrw data_priloha.xls set=n Rng=b1:u1 cdim=1"
$gdxin data_priloha.gdx
$load n
$call "Gdxxrw data_priloha.xls set=t Rng=a2:a1656 rdim=1"
$gdxin data_priloha.gdx
$load t
//###############################################

  parameter hod[t] "hodnota prvku z množiny t";
    hod[t]=ord[t]-1;
  alias{n,n1};  //vytvorili sme množiny identické s n
    t_b[t]=yes$(mod(hod[t],baza)=0 and hod[t]>0);

//=============================================================================
** DÁTA A MODIFIKÁCIE VSTUPNÝCH PARAMETROV
  parameter x[t,n] "pozorovania pre jednotlivé zložky v jednotlivých časoch";

//#########################################################
$call "Gdxxrw data_priloha.xls par=x Rng=a1:u1656 rdim=1 cdim=1"
$gdxin data_priloha.gdx
$load x
//#########################################################
    er = (er/pd)*baza;

//=============================================================================
** ZOSTAVENIE MODELU
  parameters
      r[t,n]      "očakávaný výnos jednotlivých akcií"
      cov[t,n,n1] "variančno-kovariančná matica";

  positive variable
      w[t,n]      "váhy jednotlivých zložiek v portfóliu";
  free variable
      var         "variancia portfólia (hodnota účelovej funkcie)";

  equations
      vf_min[t]      "definícia účelovej funkcie"
      ere[t]         "ohraničenie pre očakávaný vynos"
      sw[t]          "podmienka jednotkového súčtu váh";
```

```
    vf_min[orm] ..   var =e= sum{[n,n1],w[orm,n]*w[orm,n1]*cov[orm,n,n1]};
    ere[orm]    ..           sum{n,w[orm,n]*r[orm,n]} =g= er;
    sw[orm]     ..                    sum{n,w[orm,n]} =e= 1;

  model markowitz /vf_min,ere,sw/;

//=============================================================================
//=============================================================================
*** RIEŠENIE KONKRÉTNYCH PROBLÉMOV POMOCOU MODELU MARKOWITZOVEJ TEÓRIE PORTFÓLIA
***         (STRATÉGIE BUY & HOLD  A  REBALANCOVANIE PORTFOLIA)

//=============================================================================
** DEKLARÁCIA (PRÍPADNE DEFINICIA) PARAMETROV A MNOŽÍN POTREBNÝCH NA RIEŠENIE
*                               MODELU

  parameters
      rr[t,n]     "relatívny výnos"
      real_hod[t] "reálna hodnota portfólia v rôznych časoch pre buy and hold"
      ocak_hod[t] "očakávaná hodnota pre buy and hold"
      pom         "počet dní, ktoré uplynuli od posledného rebalancovania"
                 // parameter potrebný na výpočet hodnoty portfólia pri
                 // prehodnocovaní
      r_preh[t]   "reálna hodnota portfólia v rôznych časoch pri prehodnocovaní"
      o_preh[t]   "očakávaná hodnota pri prehodnocovaní";

  sets
      h[t]        "história pre výpočet očakávaného výnosu"
      hi_cov[t]   "história pre výpočet variančno-kovariančnej matice";

//*****************************************************************************
* DEKLARÁCIA PARAMETROV POTREBNÝCH PRE VÝPOČET OČAKÁVANÉHO VÝNOSU
  parameter   // pre kĺzavé priemery
      o_c[t,n] "očakávaná cena";

  parameters  // pre exp regresiu
      aa[n],bb[n] "koeficienty v exp regresii"
      hod_reg[t]  "hodnoty prvkov množiny h v rámci regresie"
      priem_cas   "priemer z hodnôt prvkov množiny h (hod_reg)"
      jj;

  sets        // pre faktorové modely
    //#######################
      f "faktory" /HDP,CPI,u/
    //#######################
      r_z[n];
    alias(f,f1);
  parameters
      HDP[t,n]          "hrubý domáci produkt"
      CPI[t,n]          "index spotrebiteľských cien"
      u[t,n]            "faktor nezamestnanosti"
      m_f[f,t,n]        "matica faktorov pre danú zložku"
      priem_fakt[f,n]   "stredná hodnota faktora"
      E_rr[n]           "stredná hodnota rr"
      cov_fakt[f,f1,n]  "vriačno-kovariančná matica faktorov"
      cov_r_fakt[f,n]   "vriačno-kovariančná matica faktoru a výnosu"
      fa[n]             "koeficient a";
  variables
      huf
      fb[f,n];
  equations
      uf
      sustava[f,n];
    uf             .. huf =e= 1;
    sustava[f,r_z] ..         sum{f1,cov_fakt[f,f1,r_z]*fb[f1,r_z]}
                           =e= cov_r_fakt[f,r_z];
  model koef_b /uf,sustava/;
```

```
//#########################################################
$call "Gdxxrw fgdp.xls par=HDP Rng=a1:u331 rdim=1 cdim=1"
$gdxin fgdp.gdx
$load HDP
$call "Gdxxrw fcpi.xls par=CPI Rng=a1:u331 rdim=1 cdim=1"
$gdxin fcpi.gdx
$load CPI
$call "Gdxxrw funem.xls par=u Rng=a1:u331 rdim=1 cdim=1"
$gdxin funem.gdx
$load u
    m_f['HDP',t,n] = HDP[t,n];
    m_f['CPI',t,n] = CPI[t,n];
    m_f['u',t,n]   = u[t,n];
//#########################################################


  parameters  // pre CAPM
      ri_men[t,n]     "výnosy indexu podľa meny"
      ri_mat[t,n]     "výnosy indexu podľa maturity"
      priem_indexu[n] "stredná hodnota výnosu z indexu";
    // + používame "E_rr"  z faktorovych modelov a bb z exp regresie

//#####################################################################
$call "Gdxxrw indexmena.xls par=ri_men Rng=a1:u331 rdim=1 cdim=1"
$gdxin indexmena.gdx
$load ri_men
$call "Gdxxrw indexmaturita.xls par=ri_mat Rng=a1:u331 rdim=1 cdim=1"
$gdxin indexmaturita.gdx
$load ri_mat
//#####################################################################

//*************************************************************************

    rr[t,n]$(t_b[t]) = (x[t,n]-x[t-baza,n])/x[t-baza,n];
    ocak_hod[t]$(hod[t]=start) = m;
    real_hod[t]$(hod[t]=start) = m;
    o_preh[t]$(hod[t]=start) = m;
    r_preh[t]$(hod[t]=start) = m;

//=========================================================================
** BUY & HOLD  A  PREHODNOCOVANIE (REBALANCOVANIE)
  scalar i /0/;

  for{i=0 to min(dlzka,(card{t}-1-start)),
    if{mod(i,per_preh)=0,
      orm[t] = yes$(hod[t]=start+i);
      h[t] = yes$(t_b[t] and hod[t]>=start+i-hist+baza and
                  hod[t]<=start+i and hod[t]>0);
      hi_cov[t] = yes$(t_b[t] and hod[t]>=start+i-250+baza and
                       hod[t]<=start+i and hod[t]>0);

    //*********************************************************************
    //priemerný výnos
      if{vynos=1,
        r[orm,n] = (1/card{h})*sum{h,rr[h,n]};
      };
    //---------------------------------------------------------------------
    //komulovaný
      if{vynos=2,
        r[orm,n] = (prod{h,(1+rr[h,n])}-1)/card{h};
      };
    //---------------------------------------------------------------------
    //kĺzavé priemery
      if{vynos=3,
        o_c[t+baza,n]$(hod[t]=start+i) =
          0.2*(-4*x[t-20,n]+ 11*x[t-15,n]-4*x[t-10,n]-14*x[t-5,n]+16*x[t,n]);
        r[t,n]$(hod[t]=start+i) = (o_c[t+baza,n]-x[t,n])/x[t,n];
      };
```

```
//-----------------------------------------------------------------------------
//exponenciálna regresia
  if{vynos=4,
    jj = 1;
    loop{t_b$(hod[t_b]>=start+i-hist+baza), hod_reg[t_b]=jj; jj=jj+1; };
    priem_cas = sum{h,hod_reg[h]}/card{h};
    bb[n] = exp( (sum{h,hod_reg[h]*log(x[h,n])}
                  -priem_cas*sum{h,log(x[h,n])})/
                 (sum{h,hod_reg[h]**2}-card{h}*priem_cas**2) );
    aa[n] = exp(sum{h,log(x[h,n])}/card{h}-log(bb[n])*priem_cas);
    o_c[t,n]$(hod[t]=start+i+baza) = aa[n]*power(bb[n],hod_reg[t]);
    r[t,n]$(hod[t]=start+i) = (o_c[t+baza,n]-x[t,n])/x[t,n];
  };
//-----------------------------------------------------------------------------
//faktorový model
  if{vynos=5,
    priem_fakt[f,n] = sum{h,m_f[f,h,n]}/card{h};
    E_rr[n] = sum{h,rr[h,n]}/card{h};
    cov_fakt[f,f1,n] = (1/(card{h}-1))*
      sum{h,(m_f[f,h,n]-priem_fakt[f,n])*(m_f[f1,h,n]-priem_fakt[f1,n])};
    cov_r_fakt[f,n] = (1/(card{h}-1))*
      sum{h,(rr[h,n]-E_rr[n])*(m_f[f,h,n]-priem_fakt[f,n])};
    loop{n,
      r_z[n1] = no;
      r_z[n] = yes;
      solve koef_b using lp maximizing huf;
    };
    fa[n] = E_rr[n]-sum{f,fb.l[f,n]*priem_fakt[f,n]};
    r[t,n]$(hod[t]=start+i) = fa[n]+sum{f,fb.l[f,n]*m_f[f,t+baza,n]};
    r[t,n]$(hod[t]=start+i) = r[t,n]*pd/baza;
  };
//-----------------------------------------------------------------------------
//CAPM - indexy cez meny
  if{vynos=6,
    priem_indexu[n] = sum{h,ri_men[h,n]}/card{h};
    E_rr[n] = sum{h,rr[h,n]}/card{h};
    bb[n] = sum{h,(rr[h,n]-E_rr[n])*(ri_men[h,n]-priem_indexu[n])}/
            sum{h,sqr(ri_men[h,n]-priem_indexu[n])};
    r[t,n]$(hod[t]=start+i) = rr[t,'u1']+bb[n]*(priem_indexu[n]-rr[t,'u1']);
  };
//-----------------------------------------------------------------------------
//CAPM - indexy cez maturity
  if{vynos=7,
    priem_indexu[n] = sum{h,ri_mat[h,n]}/card{h};
    E_rr[n] = sum{h,rr[h,n]}/card{h};
    bb[n] = sum{h,(rr[h,n]-E_rr[n])*(ri_mat[h,n]-priem_indexu[n])}/
            sum{h,sqr(ri_mat[h,n]-priem_indexu[n])};

    r[t,n]$(hod[t]=start+i) = rr[t,'u1']+bb[n]*(priem_indexu[n]-rr[t,'u1']);
  };
//-----------------------------------------------------------------------------
//AR(1) proces
  if{vynos=8,
    fa[n] = sum{t$(h[t] and hod[t]<>start+i),rr[t,n]*rr[t+baza,n]}/
            sum{h$(hod[h]<>start+i),sqr(rr[h,n])};
    r[t,n]$(hod[t]=start+i) = fa[n]*rr[t,n];
  };
//*****************************************************************************

  cov[orm,n,n1] = (1/(card{hi_cov}-1))*
    sum{hi_cov,(rr[hi_cov,n]-r[orm,n])*(rr[hi_cov,n1]-r[orm,n1])};
  solve markowitz minimizing var using nlp;
};
ocak_hod[t]$(hod[t]=start+i+1) = m*(1+ere.l[t-(i+1)]*(i+1)/baza);
real_hod[t]$(hod[t]=start+i+1) =
  m*(1+sum{n,w.l[t-(i+1),n]*((x[t,n]-x[t-(i+1),n])/x[t-(i+1),n])});
pom = mod(i,per_preh)+1;
o_preh[t]$(hod[t]=start+i+1) = r_preh[t-pom]*(1+pom*ere.l[t-pom]/baza);
```

```
    r_preh[t]$(hod[t]=start+i+1) =
      r_preh[t-pom]*(1+sum{n,w.l[t-pom,n]*((x[t,n]-x[t-pom,n])/x[t-pom,n])});
  };

//=============================================================================
//=============================================================================
***                       ZAZNAMENANIE VÝSLEDKOV
***        A ICH EXPORTOVANIE DO EXCELU SPOLU SO VSTUPNÝMI PARAMETRAMI

  parameter vysledky[*,*],vahy[t,n],vstupne_parametre[*];

    vysledky[t,'očakávaná hodnota reb'] = o_preh[t];
    vysledky[t,'reálna hodnota reb']    = r_preh[t];
    vysledky[t,'reálna hodnota b&h']    = real_hod[t];
    vahy[t,n]                           = w.l[t,n];
    vstupne_parametre['historické obdobie (v dňoch)']         = hist;
    vstupne_parametre['začiatok']                             = start;
    vstupne_parametre['počet dní v roku']                     = pd;
    vstupne_parametre['dĺžka investičného obdobia']           = dlzka;
    vstupne_parametre['očakávaný ročný výnos portfólia']      = er/baza*pd;
    vstupne_parametre['časová báza modelu (v dňoch)']         = baza;
    vstupne_parametre['perióda prehodnocovania']              = per_preh;
    vstupne_parametre['výška počiatočných investícií']        = m;
    vstupne_parametre['typ výnosu']                           = vynos;

  Execute_Unload "markvysledky.gdx",vstupne_parametre,t,n,vysledky,vahy;
  Execute 'GDXXRW.EXE markvysledky.gdx par=vstupne_parametre rng=a2 rdim=1';
  Execute 'GDXXRW.EXE markvysledky.gdx par=vysledky rng=d12';
  Execute 'GDXXRW.EXE markvysledky.gdx par=vahy rng=j12';
```

Model s kvadratickou funkciou užitočnosti

Vstupné parametre špecifické pre tento model:

Skalár `d` predstavuje transakčné náklady (sadzba).

Pomocou skalára `koef_rizika` definujeme hodnotu parametra `alfa` v okamihoch prehodnocovania portfólia.

Matematická formulácia modelu:

Účelová funkcia v i -tom časovom okamihu:

$$\max \quad \sum_{j=1}^{N}(1-d)x_{ij}\left(1+r_{ij}\right)h_{i+p\,j}-\frac{1}{2}\alpha_i\left(\sum_{j=1}^{N}(1-d)x_{ij}\left(1+r_{ij}\right)h_{i+p\,j}\right)^2-\frac{1}{2}\alpha_i\sum_{j=1}^{N}\sum_{k=1}^{N}(1-d)^2\,x_{ij}\,x_{ik}\,\rho_{jk}\,h_{i+p\,j}\,h_{i+pk}$$

Prvky, cez ktoré prechádza index i, sú determinované vstupnými parametrami modelu (`start`, `dlzka`, `per_preh`), symbol p predstavuje skalár `baza` z formulácie v jazyku GAMS a N je celkový počet finančných aktív („potenciálnych zložiek portfólia“). d sú už spomínané transakčné náklady, x_{ij} je cena j -tej zložky trhu v i -tom časovom okamihu, r_{ij} je výnos j -tej zložky trhu v i -tom časovom okamihu, $h_{i+p\,j}$ je premenná predstavujúca držané množstvo z j -tej zložky trhu do $i+p$ -teho časového okamihu, po transakciách v i -tom časovom okamihu. Dôvod časového posunu vyplýva z ohraničení. α_i je parameter determinujúci bod z definičného oboru, v ktorom účelová funkcia nadobúda svoje maximum. ρ_{jk} je kovariancia medzi j -tou zložkou a k -tou zložkou trhu.

Ohraničenia v i -tom časovom okamihu:

$$h_{ij}+b_{ij}-s_{ij}=h_{i+p\,j} \qquad \left(j=1,\ldots,N\right)$$

$$\sum_{j=1}^{N}(1-d)x_{ij}\,s_{ij} = \sum_{j=1}^{N}(1+d)x_{ij}\,b_{ij} \qquad \text{ak} \quad i \neq start$$

$$m = \sum_{j=1}^{N}(1+d)x_{ij}\,b_{ij} \qquad \text{ak} \quad i = start$$

Prvé ohraničenie definuje premennú $h_{i+p\,j}$ pre j -tu zložku ako množstvo, ktoré sme mali pred transakciami, teda v i -tom časovom okamihu, plus nakúpené množstvo b_{ij} v i -tom časovom okamihu mínus predané množstvo s_{ij} v i -tom časovom okamihu. Dostávame množstvo, ktorým budeme disponovať počas celého obdobia do ďalšieho prehodnocovania. Práve z tohto ohraničenia vyplýva logická potreba časového posunu premennej *h*. Za druhým ohraničením je idea tzv. samofinancovania, teda nakupovať môžeme len za objem peňazí získaný z predaja. Keď sme na začiatku investovania, ľavú stranu (objem peňazí získaný z predaja) v tomto ohraničení nahradí skalár *m* predstavujúci počiatočné investície.

Formulácia modelu v jazyku GAMS:

Teraz uvedieme zdrojový kód programu pre model s kvadratickou funkciou užitočnosti:

```
$eolcom //


//#############################################################################
* ZADÁVANIE VSTUPNÝCH PARAMETROV MODELU
  scalars
      start    "deň začiatku investovania" /200/
      hist     "obdobie z ktorého máme historické dáta (v dňoch)" /100/
            // ak chcete celú históriu, stačí do "hist" priradiť číslo väčšie
            // ako počet dní v množine t
      dlzka    "dĺžka obdobia investovania (v dňoch)" /1000/
            // teda obdobia, počas ktorého buď držíme alebo prehodnocujeme
            // portfólio
      baza     "časová báza modelu (v dňoch)" /5/
            // vyjadruje na akej časovej báze (v dňoch) sa pracuje s dátami
      pd       "počet dní v roku" /250/
      per_preh "perióda prehodnocovania (v dňoch)" /5/
      m        "výška investícií" /1000000/
      vynos    "typ očakávaného výnosu" /1/                                    ;
            // parameter vypovedajúci o spôsobe počítania očakávaného výnosu

    //priemerný výnos            => vynos= 1
    //komulovaný výnos           => vynos= 2
    //kĺzavé priemery            => vynos= 3
    //exponenciálna regresia     => vynos= 4
    //faktorový model            => vynos= 5
    //CAPM - indexy cez meny     => vynos= 6
    //CAPM - indexy cez maturity => vynos= 7
    //AR(1) proces               => vynos= 8
  scalars
      d           "transakčné náklady" /0.003/
      koef_rizika "koeficient určujúci veľkosť averzie k riziku (alfa)" /1.05/;
//#############################################################################



$offsymxref offsymlist offuelxref offuellist
  options limrow=3,limcol=0,solprint=on,decimals=3;

//=============================================================================
//=============================================================================
***           KONŠTRUKCIA MODELU S KVADRATICKOU FUNKCIOU UŽITOČNOSTI
```

```
//===========================================================================
** MNOŽINY POUŽÍVANÉ V MODELI
  sets
      n      "zložky trhu"
      t      "množina časových okamihov (denné)"
      t_b[t] "množina časových okamihov podľa bázy modelu"
      orm[t] "okamih riešenia modelu";

//##############################################
$call "Gdxxrw data_priloha.xls set=n Rng=b1:u1 cdim=1"
$gdxin data_priloha.gdx
$load n
$call "Gdxxrw data_priloha.xls set=t Rng=a2:a1656 rdim=1"
$gdxin data_priloha.gdx
$load t
//##############################################

  parameter hod[t] "hodnota prvku z množiny t";
    hod[t]=ord{t}-1;
  alias(n,n1);  //vytvorili sme množiny identické s n
    t_b[t]=yes$(mod(hod[t],baza)=0 and hod[t]>0);

//===========================================================================
** DÁTA A MODIFIKÁCIE VSTUPNÝCH PARAMETROV
  parameter x[t,n] "pozorovania pre jednotlivé zložky v jednotlivých časoch";

//#########################################################
$call "Gdxxrw data_priloha.xls par=x Rng=a1:u1656 rdim=1 cdim=1"
$gdxin data_priloha.gdx
$load x
//#########################################################

//===========================================================================
** ZOSTAVENIE MODELU
  parameters
      r[t,n]      "očakávaný výnos jednotlivých akcií"
      cov[t,n,n1] "variančno-kovariančná matica"
      alfa[t]     "averzia k riziku";

  positive variables
      b[t,n] "kúpené množstvo danej zložky v aktuálnom čase"
      s[t,n] "predané množstvo danej zložky v aktuálnom čase"
      h[t,n] "počet kusov po transakcii";

  free variables
      euwt "stredná hodnota užitkovej funkcie";

  equations
      vf_max[t] "účelová funkcia"
      ib[t,n]   "rovnica definujúca premennú h"
      cfa[t]    "nakúpiť môžem len za množstvo peňazí aké som získal z predaja";

    vf_max[t]$(orm[t]) .. euwt =e= sum{n,(1-d)*x[t,n]*(1+r[t,n])*h[t+baza,n]}
                                 - 0.5*alfa[t]*sqr(sum{n,(1-d)*x[t,n]*
                                                   (1+r[t,n])*h[t+baza,n]})
                                 - 0.5*alfa[t]*sum{[n,n1],sqr(1-d)*x[t,n]*
                                                   x[t,n1]*h[t+baza,n]*
                                                   h[t+baza,n1]*cov[t,n,n1]};

    ib[t,n]$(orm[t])  ..      h.l[t,n]+b[t,n]-s[t,n]
                          =e= h[t+baza,n];
    cfa[orm]          ..      m$(hod[orm]=start)+sum{n,(1-d)*x[orm,n]*s[orm,n]}
                          =e= sum{n,(1+d)*x[orm,n]*b[orm,n]};

  model uzitkova_funkcia /vf_max,ib,cfa/;

//===========================================================================
```

```
//=============================================================================
***       RIEŠENIE KONKRÉTNYCH PROBLÉMOV POMOCOU PREDCHÁDZAJÚCEHO MODELU
***          (STRATÉGIE BUY & HOLD  A  REBALANCOVANIE PORTFOLIA)

//=============================================================================
** DEKLARÁCIA (PRÍPADNE DEFINÍCIA) PARAMETROV A MNOŽÍN POTREBNÝCH NA RIEŠENIE
*                                MODELU

  parameters
      rr[t,n]     "relatívny výnos"
      real_hod[t] "reálna hodnota portfólia v rôznych časoch pre buy and hold"
      ocak_hod[t] "očakávaná hodnota pre buy and hold"
      pom         "počet dní, ktoré uplynuli od posledného rebalancovania"
                // parameter potrebný na výpočet hodnoty portfólia pri
                // prehodnocovaní
      r_preh[t]   "reálna hodnota portfólia v rôznych časoch pri prehodnocovaní"
      o_preh[t]   "očakávaná hodnota pri prehodnocovaní"

  sets
      hi[t]       "história pre výpočet očakávaného výnosu"
      hi_cov[t]   "história pre výpočet variančno-kovariančnej matice";

//*****************************************************************************
* DEKLARÁCIA PARAMETROV POTREBNÝCH PRE VÝPOČET OČAKÁVANÉHO VÝNOSU
  parameter   // pre kĺzavé priemery
      o_c[t,n] "očakávaná cena";

  parameters  // pre exp regresiu
      aa[n],bb[n] "koeficienty v exp regresii"
      hod_reg[t]  "hodnoty prvkov množiny h v rámci regresie"
      priem_cas   "priemer z hodnôt prvkov množiny h (hod_reg)"
      jj;

  sets        // pre faktorové modely
    //########################
      f "faktory" /HDP,CPI,u/
    //########################
      r_z[n];
    alias{f,f1};
  parameters
      HDP[t,n]          "hrubý domáci produkt"
      CPI[t,n]          "index spotrebiteľských cien"
      u[t,n]            "faktor nezamestnanosti"
      m_f[f,t,n]        "matica faktorov pre danú zložku"
      priem_fakt[f,n]   "stredná hodnota faktora"
      E_rr[n]           "stredná hodnota rr"
      cov_fakt[f,f1,n]  "vriačno-kovariančná matica faktorov"
      cov_r_fakt[f,n]   "vriačno-kovariančná matica faktoru a výnosu"
      fa[n]             "koeficient a";
  variables
      huf
      fb[f,n];
  equations
      uf
      sustava[f,n];
    uf             .. huf =e= 1;
    sustava[f,r_z] ..         sum{f1,cov_fakt[f,f1,r_z]*fb[f1,r_z]}
                           =e= cov_r_fakt[f,r_z];
  model koef_b /uf,sustava/;

//#########################################################
$call "Gdxxrw fgdp.xls par=HDP Rng=a1:u331 rdim=1 cdim=1"
$gdxin fgdp.gdx
$load HDP
$call "Gdxxrw fcpi.xls par=CPI Rng=a1:u331 rdim=1 cdim=1"
$gdxin fcpi.gdx
$load CPI
$call "Gdxxrw funem.xls par=u Rng=a1:u331 rdim=1 cdim=1"
```

```
$gdxin funem.gdx
$load u
    m_f['HDP',t,n] = HDP[t,n];
    m_f['CPI',t,n] = CPI[t,n];
    m_f['u',t,n]   = u[t,n];
//#########################################################

  parameters  // pre CAPM
      ri_men[t,n]      "výnosy indexu podľa meny"
      ri_mat[t,n]      "výnosy indexu podľa maturity"
      priem_indexu[n]  "stredná hodnota výnosu z indexu";
    // + používame "E_rr" z faktorovych modelov a bb z exp regresie

//#####################################################################
$call "Gdxxrw indexmena.xls par=ri_men Rng=a1:u331 rdim=1 cdim=1"
$gdxin indexmena.gdx
$load ri_men
$call "Gdxxrw indexmaturita.xls par=ri_mat Rng=a1:u331 rdim=1 cdim=1"
$gdxin indexmaturita.gdx
$load ri_mat
//#####################################################################

//*************************************************************************

    rr[t,n]$(t_b[t]) = (x[t,n]-x[t-baza,n])/x[t-baza,n];
    h.l[t,n] = 0;
    ocak_hod[t]$(hod[t]=start) = m;
    real_hod[t]$(hod[t]=start) = m;
    r_preh[t]$(hod[t]=start) = m;
    o_preh[t]$(hod[t]=start) = m;

//=========================================================================
** PREHODNOCOVANIE (REBALANCOVANIE)
  scalar i /0/;

  for{i=0 to min(dlzka,(card{t}-1-start)),
    if{mod(i,per_preh)=0,
      orm[t] = yes$(hod[t]=start+i);
      hi[t] = yes$(t_b[t] and hod[t]>=start+i-hist+baza and
                   hod[t]<=start+i and hod[t]>0);
      hi_cov[t] = yes$(t_b[t] and hod[t]>=start+i-250+baza and
                       hod[t]<=start+i and hod[t]>0);

    //*********************************************************************
    //priemerný výnos
      if{vynos=1,
        r[orm,n] = (1/card{hi})*sum{hi,rr[hi,n]};
      };
    //---------------------------------------------------------------------
    //komulovaný
      if{vynos=2,
        r[orm,n] = (prod{hi,(1+rr[hi,n])}-1)/card{hi};
      };
    //---------------------------------------------------------------------
    //kĺzavé priemery
      if{vynos=3,
        o_c[t+baza,n]$(hod[t]=start+i) =
          0.2*(-4*x[t-20,n]+ 11*x[t-15,n]-4*x[t-10,n]-14*x[t-5,n]+16*x[t,n]);
        r[t,n]$(hod[t]=start+i) = (o_c[t+baza,n]-x[t,n])/x[t,n];
      };
    //---------------------------------------------------------------------
    //exponenciálna regresia
      if{vynos=4,
        jj = 1;
        loop{t_b$(hod[t_b]>=start+i-hist+baza), hod_reg[t_b]=jj; jj=jj+1; };
        priem_cas = sum{hi,hod_reg[hi]}/card{hi};
        bb[n] = exp( (sum{hi,hod_reg[hi]*log(x[hi,n])}
                     -priem_cas*sum{hi,log(x[hi,n])})/
```

```
                    (sum{hi,hod_reg[hi]**2}-card{hi}*priem_cas**2) );
        aa[n] = exp(sum{hi,log(x[hi,n])}/card{hi}-log(bb[n])*priem_cas);
        o_c[t,n]$(hod[t]=start+i+baza) = aa[n]*power(bb[n],hod_reg[t]);
        r[t,n]$(hod[t]=start+i) = (o_c[t+baza,n]-x[t,n])/x[t,n];
      };
    //-------------------------------------------------------------------------
    //faktorový model
      if{vynos=5,
        priem_fakt[f,n] = sum{hi,m_f[f,hi,n]}/card{hi};
        E_rr[n] = sum{hi,rr[hi,n]}/card{hi};
        cov_fakt[f,f1,n] = (1/(card{hi}-1))*
          sum{hi,(m_f[f,hi,n]-priem_fakt[f,n])*(m_f[f1,hi,n]-priem_fakt[f1,n])};
        cov_r_fakt[f,n] = (1/(card{hi}-1))*
          sum{hi,(rr[hi,n]-E_rr[n])*(m_f[f,hi,n]-priem_fakt[f,n])};
        loop{n,
          r_z[n1] = no;
          r_z[n] = yes;
          solve koef_b using lp maximizing huf;
        };
        fa[n] = E_rr[n]-sum{f,fb.l[f,n]*priem_fakt[f,n]};
        r[t,n]$(hod[t]=start+i) = fa[n]+sum{f,fb.l[f,n]*m_f[f,t+baza,n]};
        r[t,n]$(hod[t]=start+i) = r[t,n]*pd/baza;
      };
    //-------------------------------------------------------------------------
    //CAPM - indexy cez meny
      if{vynos=6,
        priem_indexu[n] = sum{hi,ri_men[hi,n]}/card{hi};
        E_rr[n] = sum{hi,rr[hi,n]}/card{hi};
        bb[n] = sum{hi,(rr[hi,n]-E_rr[n])*(ri_men[hi,n]-priem_indexu[n])}/
                sum{hi,sqr(ri_men[hi,n]-priem_indexu[n])};
        r[t,n]$(hod[t]=start+i) = rr[t,'u1']+bb[n]*(priem_indexu[n]-rr[t,'u1']);
      };
    //-------------------------------------------------------------------------
    //CAPM - indexy cez maturity
      if{vynos=7,
        priem_indexu[n] = sum{hi,ri_mat[hi,n]}/card{hi};
        E_rr[n] = sum{hi,rr[hi,n]}/card{hi};
        bb[n] = sum{hi,(rr[hi,n]-E_rr[n])*(ri_mat[hi,n]-priem_indexu[n])}/
                sum{hi,sqr(ri_mat[hi,n]-priem_indexu[n])};
        r[t,n]$(hod[t]=start+i) = rr[t,'u1']+bb[n]*(priem_indexu[n]-rr[t,'u1']);
      };
    //-------------------------------------------------------------------------
    //AR(1) proces
      if{vynos=8,
        fa[n] = sum{t$(hi[t] and hod[t]<>start+i),rr[t,n]*rr[t+baza,n]}/
                sum{hi$(hod[hi]<>start+i),sqr(rr[hi,n])};
        r[t,n]$(hod[t]=start+i) = fa[n]*rr[t,n];
      };
    //*************************************************************************

      cov[orm,n,n1] = (1/(card{hi_cov}-1))*
        sum{hi_cov,(rr[hi_cov,n]-r[orm,n])*(rr[hi_cov,n1]-r[orm,n1])};
      alfa[orm] = 1/(koef_rizika*r_preh[orm]);
      solve uzitkova_funkcia maximizing euwt using nlp;
    };
    pom = mod(i,per_preh)+1;
    o_preh[t]$(hod[t]=start+i+1) =
      sum{n,(1-d)*x[t-pom,n]*(1+pom*r[t-pom,n]/baza)*h.l[t-(pom-baza),n]};
    r_preh[t]$(hod[t]=start+i+1) = sum{n,(1-d)*x[t,n]*h.l[t-(pom-baza),n]};
    ocak_hod[t]$(hod[t]=start+i+1) =
      sum{n,(1-d)*x[t-(i+1),n]*(1+(i+1)*r[t-(i+1),n]/baza)*h.l[t-(i+1-baza),n]};
    real_hod[t]$(hod[t]=start+i+1) = sum{n, (1-d)*x[t,n]*h.l[t-(i+1-baza),n]};
  };

//===========================================================================
//===========================================================================
***                        ZAZNAMENANIE VÝSLEDKOV
***        A ICH EXPORTOVANIE DO EXCELU SPOLU SO VSTUPNÝMI PARAMETRAMI
```

```
parameter vysledky[*,*],vahy[t,n],vstupne_parametre[*];

vahy[t,n] = ((1-d)*x[t,n]*h.l[t+baza,n]/r_preh[t+1])$(r_preh[t+1]);
vysledky[t,'očakávaná hodnota reb'] = o_preh[t];
vysledky[t,'reálna hodnota reb']    = r_preh[t];
vysledky[t,'reálna hodnota b&h']    = real_hod[t];
vstupne_parametre['historické obdobie (v dňoch)']         = hist;
vstupne_parametre['začiatok']                             = start;
vstupne_parametre['počet dní v roku']                     = pd;
vstupne_parametre['dĺžka investičného obdobia']           = dlzka;
vstupne_parametre['časová báza modelu (v dňoch)']         = baza;
vstupne_parametre['perióda prehodnocovania']              = per_preh;
vstupne_parametre['výška počiatočných investícií']        = m;
vstupne_parametre['typ výnosu']                           = vynos;

Execute_Unload "uzitkovavysledky.gdx",vstupne_parametre,vahy,t,n,vysledky;
Execute 'GDXXRW.EXE uzitkovavysledky.gdx par=vstupne_parametre rng=a2 rdim=1';
Execute 'GDXXRW.EXE uzitkovavysledky.gdx par=vysledky rng=d12';
Execute 'GDXXRW.EXE uzitkovavysledky.gdx par=vahy rng=j12';
```

Príloha 2 – Výstupné súbory modelov z druhej časti

V tejto prílohe poskytneme výstupné súbory k štyrom modelom z druhej časti. Jedná sa o modely „Rámy na obrazy“ a „Investovanie“ z kapitoly 17 LP, o Griffinov model z kapitoly 18 LRM a o model Markowitzovej teórie portfólia z kapitoly 21. Výstupné súbory zvyšných dvoch konkrétnych modelov (CGE, DEA) neuvádzame, pretože sú veľmi dlhé (20 a 120 strán) a na ilustráciu štruktúry výstupu dostatočne postačujú štyri horeuvedené modely.

Konkrétna úloha LP – „ Rámy na obrazy “

```
GAMS Rev 141  Intel /MS Window
26.04.06 00:42:28 Page 1

G e n e r a l   A l g e b r a i c   M o d e l i n g   S y s t e m
C o m p i l a t i o n


   1
   2
   3    sets
   4        r  "typy rámov na obrazy"       / r1 , r2 , r3 , r4 /
   5        vf "jednotlivé výrobné faktory" / praca "v hodinách"
   6                                          kov   "v kilogramoch"
   7                                          sklo  "v kilogramoch" /;
   8
   9    table
  10        S[vf,r] "množstvo spotrebovaného výrobného faktora na daný typ rámu"
  11                     r1   r2   r3   r4
  12               praca    2    1    3    2
  13               kov      4    2    1    2
  14               sklo     6    2    1    2 ;
  15
  16    parameters
  17        z[r]  "zisk z jedného predaného rámu daného typu"
  18                / r1=6 , r2=2 , r3=4 , r4=3 /
  19        d[r]  "garantovaný odbyt jednotlivých typov rámov"
  20                / r1=1000 , r2=2000 , r3=500 , r4=1000 /
  21        m[vf] "disponibilné množstvo jednotlivých výrobných faktorov"
  22                / praca=4000 , kov=6000 , sklo=10000 /;
  23
  24    variables
  25        cz   "celkový zisk"
  26        x[r] "počet vyrobených rámov daného typu";
  27      x.lo[r]=0;
  28      x.up[r]=d[r];
  29
  30    equations
  31        uf         "účelová funkcia"
  32        zdroje[vf] "ohraničenia na zdroje";
  33      uf         ..  cz =e= sum{r,z[r]*x[r]};
  34      zdroje[vf] ..         sum{r,S[vf,r]*x[r]} =l= m[vf];
  35
  36    model ramy / all /;
  37
  38    solve ramy maximizing cz using lp;
  39
  40    display x.l;


COMPILATION TIME     =        0.000 SECONDS    3.2 Mb  WIN216-141 Jan 24, 2005
```

```
GAMS Rev 141  Intel /MS Window
26.04.06 00:42:28 Page 2
G e n e r a l   A l g e b r a i c   M o d e l i n g   S y s t e m
Equation Listing    SOLVE ramy Using LP From line 38


---- uf  =E=  účelová funkcia

uf..  cz - 6*x(r1) - 2*x(r2) - 4*x(r3) - 3*x(r4) =E= 0 ; (LHS = 0)


---- zdroje  =L=  ohraničenia na zdroje

zdroje(praca)..  2*x(r1) + x(r2) + 3*x(r3) + 2*x(r4) =L= 4000 ; (LHS = 0)

zdroje(kov)..  4*x(r1) + 2*x(r2) + x(r3) + 2*x(r4) =L= 6000 ; (LHS = 0)

zdroje(sklo)..  6*x(r1) + 2*x(r2) + x(r3) + 2*x(r4) =L= 10000 ; (LHS = 0)

GAMS Rev 141  Intel /MS Window
26.04.06 00:42:28 Page 3
G e n e r a l   A l g e b r a i c   M o d e l i n g   S y s t e m
Column Listing      SOLVE ramy Using LP From line 38


---- cz  celkový zisk

cz
                (.LO, .L, .UP = -INF, 0, +INF)
        1       uf


---- x  počet vyrobených rámov daného typu

x(r1)
                (.LO, .L, .UP = 0, 0, 1000)
       -6       uf
        2       zdroje(praca)
        4       zdroje(kov)
        6       zdroje(sklo)

x(r2)
                (.LO, .L, .UP = 0, 0, 2000)
       -2       uf
        1       zdroje(praca)
        2       zdroje(kov)
        2       zdroje(sklo)

x(r3)
                (.LO, .L, .UP = 0, 0, 500)
       -4       uf
        3       zdroje(praca)
        1       zdroje(kov)
        1       zdroje(sklo)

REMAINING ENTRY SKIPPED
GAMS Rev 141  Intel /MS Window
26.04.06 00:42:28 Page 4
G e n e r a l   A l g e b r a i c   M o d e l i n g   S y s t e m
Model Statistics    SOLVE ramy Using LP From line 38


MODEL STATISTICS

BLOCKS OF EQUATIONS       2     SINGLE EQUATIONS        4
BLOCKS OF VARIABLES       2     SINGLE VARIABLES        5
NON ZERO ELEMENTS        17
```

```
GENERATION TIME      =        0.016 SECONDS    3.9 Mb  WIN216-141 Jan 24, 2005


EXECUTION TIME       =        0.016 SECONDS    3.9 Mb  WIN216-141 Jan 24, 2005
GAMS Rev 141  Intel /MS Window
26.04.06 00:42:28 Page 5
G e n e r a l   A l g e b r a i c   M o d e l i n g   S y s t e m
Solution Report     SOLVE ramy Using LP From line 38


               S O L V E      S U M M A R Y

     MODEL   ramy               OBJECTIVE  cz
     TYPE    LP                 DIRECTION  MAXIMIZE
     SOLVER  CONOPT             FROM LINE  38

**** SOLVER STATUS     1 NORMAL COMPLETION
**** MODEL STATUS      1 OPTIMAL
**** OBJECTIVE VALUE             9200.0000

 RESOURCE USAGE, LIMIT          0.000      1000.000
 ITERATION COUNT, LIMIT         5         10000


    C O N O P T 3   Intel /MS Window version 3.14E-015-053
    Copyright (C)   ARKI Consulting and Development A/S
                    Bagsvaerdvej 246 A
                    DK-2880 Bagsvaerd, Denmark

 Using default options.


 ** Optimal solution. There are no superbasic variables.


 CONOPT time Total                            0.000 seconds
   of which: Function evaluations             0.000 =  0.0%
             1st Derivative evaluations       0.000 =  0.0%

 Work length =    0.02 Mbytes
    Estimate =    0.02 Mbytes
    Max used =    0.01 Mbytes

                            LOWER          LEVEL          UPPER         MARGINAL

---- EQU uf                   .              .              .             1.0000

  uf  účelová funkcia

---- EQU zdroje  ohraničenia na zdroje

             LOWER          LEVEL          UPPER         MARGINAL

praca        -INF         4000.0000      4000.0000         1.2000
kov          -INF         6000.0000      6000.0000         0.4000
sklo         -INF         8000.0000     10000.0000          .

                            LOWER          LEVEL          UPPER         MARGINAL

---- VAR cz                 -INF         9200.0000        +INF             .

  cz  celkový zisk

---- VAR x  počet vyrobených rámov daného typu

          LOWER          LEVEL          UPPER         MARGINAL
```

```
r1          .          1000.0000      1000.0000         2.0000
r2          .           800.0000      2000.0000          .
r3          .           400.0000       500.0000          .
r4          .              .          1000.0000        -0.2000


**** REPORT SUMMARY :        0     NONOPT
                             0 INFEASIBLE
                             0  UNBOUNDED
GAMS Rev 141  Intel /MS Window
26.04.06 00:42:28 Page 6
G e n e r a l   A l g e b r a i c   M o d e l i n g   S y s t e m
E x e c u t i o n


----     40 VARIABLE x.L  počet vyrobených rámov daného typu

r1 1000.000,    r2  800.000,    r3  400.000


EXECUTION TIME       =        0.031 SECONDS    2.9 Mb  WIN216-141 Jan 24, 2005


USER: GAMS Development Corporation, Washington, DC   G871201/0000CA-ANY
      Free Demo,  202-342-0180,  sales@gams.com,  www.gams.com   DC0000


**** FILE SUMMARY

Input      D:\Skola\Diplomovka\Gams subory\ramy dip.gms
Output     D:\Skola\Diplomovka\Gams subory\ramy dip.lst
```

Konkrétna úloha LP – „ Investovanie “

```
GAMS Rev 141  Intel /MS Window
26.04.06 00:44:12 Page 1
G e n e r a l   A l g e b r a i c   M o d e l i n g   S y s t e m
C o m p i l a t i o n


   1
   2    sets
   3        t "začiatok aktuálneho mesiaca" /1*4/;
   4      alias{t,d,t1};
      'd' vyjadruje dĺžku investovania a množina 't1' je potrebná v súvislosti
      s formuláciou modelu v jazyku GAMS (v matematickej formulácii sa nevyskytuje)
   9
  10    parameters
  11        r[t] "príjmy"        /1 =   800, 2 =   800, 3 =  300, 4 =  300/
  12        b[t] "výdavky"       /1 =   600, 2 =   500, 3 =  500, 4 =  250/
  13        y[d] "úroková miera" /1 = 0.001, 2 = 0.005, 3 = 0.01, 4 = 0.02/;
  14
  15    variables
  16        x[t,d] "množstvo investícií v danom mesiaci na dané obdobie"
  17        cash   "množstvo peňazí na začiatku '5'-teho mesiaca";
  18    positive variable x;
  19
  20    equations
  21        vf       "rovnica pre účelovú funkciu"
  22        cashz[t] "ohraničenie hotovosti na začiatku mesiaca po všetkých transakciách";
  23
```

```
  24      vf       ..  cash =e=  sum{[t,d]$(ord{t}+ord{d}=card{t}+1),x[t,d]*(1+y[d])**ord{d}};
  25
  26      cashz[t] ..             r[t] +
sum{[t1,d]$(ord{t1}+ord{d}=ord{t}),x[t1,d]*(1+y[d])**ord{d}}
  27                             - b[t] - sum{d$(ord{t}+ord{d}<=card{t}+1),x[t,d]}
  28                         =e= 0;
  29
  30    model investovanie /all/;
  31
  32    solve investovanie using lp maximizing cash;
  33
  34    display x.l,cash.l;
  35
  36


COMPILATION TIME     =        0.000 SECONDS    3.2 Mb  WIN216-141 Jan 24, 2005
GAMS Rev 141  Intel /MS Window
26.04.06 00:44:12 Page 2
G e n e r a l   A l g e b r a i c   M o d e l i n g   S y s t e m
Equation Listing    SOLVE investovanie Using LP From line 32


---- vf  =E=  rovnica pre účelovú funkciu

vf..  - 1.08243216*x(1,4) - 1.030301*x(2,3) - 1.010025*x(3,2) - 1.001*x(4,1) + cash =E= 0 ; (LHS
= 0)


---- cashz  =E=  ohraničenie hotovosti na začiatku mesiaca po všetkých transakciách

cashz(1)..  - x(1,1) - x(1,2) - x(1,3) - x(1,4) =E= -200 ; (LHS = 0, INFES = 200 ***)

cashz(2)..  1.001*x(1,1) - x(2,1) - x(2,2) - x(2,3) =E= -300 ; (LHS = 0, INFES = 300 ***)

cashz(3)..  1.010025*x(1,2) + 1.001*x(2,1) - x(3,1) - x(3,2) =E= 200 ; (LHS = 0, INFES = 200 ***)

REMAINING ENTRY SKIPPED

GAMS Rev 141  Intel /MS Window
26.04.06 00:44:12 Page 3
G e n e r a l   A l g e b r a i c   M o d e l i n g   S y s t e m
Column Listing      SOLVE investovanie Using LP From line 32


---- x  množstvo investícií v danom mesiaci na dané obdobie

x(1,1)
                (.LO, .L, .UP = 0, 0, +INF)
       -1       cashz(1)
        1.001   cashz(2)

x(1,2)
                (.LO, .L, .UP = 0, 0, +INF)
       -1       cashz(1)
        1.01    cashz(3)

x(1,3)
                (.LO, .L, .UP = 0, 0, +INF)
       -1       cashz(1)
        1.0303  cashz(4)

REMAINING 7 ENTRIES SKIPPED

---- cash  množstvo peňazí na začiatku '5'-teho mesiaca

cash
                (.LO, .L, .UP = -INF, 0, +INF)
```

```
        1       vf

GAMS Rev 141  Intel /MS Window
26.04.06 00:44:12 Page 4
G e n e r a l   A l g e b r a i c   M o d e l i n g   S y s t e m
Model Statistics    SOLVE investovanie Using LP From line 32


MODEL STATISTICS

BLOCKS OF EQUATIONS       2     SINGLE EQUATIONS        5
BLOCKS OF VARIABLES       2     SINGLE VARIABLES       11
NON ZERO ELEMENTS        21


GENERATION TIME      =        0.016 SECONDS    3.9 Mb  WIN216-141 Jan 24, 2005



EXECUTION TIME       =        0.016 SECONDS    3.9 Mb  WIN216-141 Jan 24, 2005
GAMS Rev 141  Intel /MS Window
26.04.06 00:44:12 Page 5
G e n e r a l   A l g e b r a i c   M o d e l i n g   S y s t e m
Solution Report     SOLVE investovanie Using LP From line 32


               S O L V E      S U M M A R Y

     MODEL   investovanie        OBJECTIVE  cash
     TYPE    LP                  DIRECTION  MAXIMIZE
     SOLVER  CONOPT              FROM LINE  32

**** SOLVER STATUS     1 NORMAL COMPLETION
**** MODEL STATUS      1 OPTIMAL
**** OBJECTIVE VALUE              369.7724

 RESOURCE USAGE, LIMIT          0.000      1000.000
 ITERATION COUNT, LIMIT         4         10000


    C O N O P T 3   Intel /MS Window version 3.14E-015-053
    Copyright (C)   ARKI Consulting and Development A/S
                    Bagsvaerdvej 246 A
                    DK-2880 Bagsvaerd, Denmark

 Using default options.


 ** Optimal solution. There are no superbasic variables.


 CONOPT time Total                            0.000 seconds
   of which: Function evaluations             0.000 =  0.0%
             1st Derivative evaluations       0.000 =  0.0%

 Work length =    0.02 Mbytes
    Estimate =    0.02 Mbytes
    Max used =    0.01 Mbytes

                           LOWER          LEVEL          UPPER         MARGINAL

---- EQU vf                   .              .              .            1.0000

  vf  rovnica pre účelovú funkciu

---- EQU cashz  ohraničenie hotovosti na začiatku mesiaca po všetkých transakciách

         LOWER          LEVEL          UPPER         MARGINAL
```

```
1      -200.0000      -200.0000      -200.0000        -1.0824
2      -300.0000      -300.0000      -300.0000        -1.0303
3       200.0000       200.0000       200.0000        -1.0293
4       -50.0000       -50.0000       -50.0000        -1.0010

---- VAR x  množstvo investícií v danom mesiaci na dané obdobie

           LOWER          LEVEL          UPPER         MARGINAL

1.1          .              .            +INF           -0.0511
1.2          .              .            +INF           -0.0428
1.3          .              .            +INF           -0.0511
1.4          .           200.0000        +INF             .
2.1          .           199.8002        +INF             .
2.2          .              .            +INF           -0.0193
2.3          .           100.1998        +INF             .
3.1          .              .            +INF           -0.0273
3.2          .              .            +INF           -0.0192
4.1          .            50.0000        +INF             .

                          LOWER          LEVEL          UPPER         MARGINAL

---- VAR cash              -INF          369.7724        +INF             .

  cash  množstvo peňazí na začiatku '5'-teho mesiaca


**** REPORT SUMMARY :        0     NONOPT
                             0 INFEASIBLE
                             0  UNBOUNDED
GAMS Rev 141  Intel /MS Window
26.04.06 00:44:12 Page 6
G e n e r a l   A l g e b r a i c   M o d e l i n g   S y s t e m
E x e c u t i o n


----     34 VARIABLE x.L  množstvo investícií v danom mesiaci na dané obdobie

            1           3           4

1                               200.000
2      199.800     100.200
4       50.000


----     34 VARIABLE cash.L               =      369.772  množstvo peňazí na začiatku '5'-teho
mesiaca


EXECUTION TIME       =        0.078 SECONDS    2.9 Mb  WIN216-141 Jan 24, 2005


USER: GAMS Development Corporation, Washington, DC   G871201/0000CA-ANY
      Free Demo,  202-342-0180,  sales@gams.com,  www.gams.com   DC0000


**** FILE SUMMARY

Input      D:\Skola\Diplomovka\Gams subory\investment dip.gms
Output     D:\Skola\Diplomovka\Gams subory\investment dip.lst
```

Konkrétny LRM – Griffinov model

```
GAMS Rev 141  Intel /MS Window
26.04.06 00:42:02 Page 1
G e n e r a l   A l g e b r a i c   M o d e l i n g   S y s t e m
C o m p i l a t i o n


   1
   2
   4
   5
//=============================================================================================
   6  * VŠEOBECNÝ MODEL LINEÁRNEJ REGRESIE ODHADNUTÝ POMOCOU MNŠ
   7  *                Y = X*beta + epsilon
   8    sets
   9        p      "pozorovania"
  10        f      "množina charakterizujúca vysvetľujúce premenné (faktory) v modeli";
  11    parameters
  12        x[p,f] "hodnoty pozorovaní pre dané vysvetľujúce premenné"
  13        y[p]   "hodnoty pozorovaní pre skúmanú premennú";
  14
  15    variables
  16        b[f]   "vektor koeficientov, ktoré sú odhadmi pre zložky vektora beta"
  17        h_uf   "hodnota účelovej funkcie = suma štvorcov reziduí";
  18
  19    equation
  20        uf     "účelová funkcia";
  21      uf ..   h_uf =e= sum{p,sqr(sum{f,b[f]*x[p,f]}-y[p])};
  22
  23    model mns /all/;
  24
  25
//=============================================================================================
  26  * ZADÁVANIE DÁT DO MODELU
  27  * KONKRÉTNY MODEL LINEÁRNEJ REGRESIE    S/Y = b0 + b1*A/Y + b2*1/Y + epsilon
  28    sets
  29        f /0 "konštanta"
  30           1 "A/Y"
  31           2 "1/Y"      /
  32        p /1*66/;
  33
  34  * načítanie dát z Excelu (data_mns.xls) do parametra x
--- LOAD  x = 1:x
  38
  39  * načítanie dát z Excelu (data_mns.xls) do parametra y
--- LOAD  y = 1:y
  43
  44
//=============================================================================================
  45  * RIEŠENIE MODELU, NAPOČÍTANIE VYBRANÝCH ŠTATISTÍK A ZOBRAZENIE VÝSLEDKOV
  46    solve mns using nlp minimizing h_uf;
  47
  48    scalars
  49        n           "Počet prvkov množiny p, teda počet pozorovaní"
  50        k           "Počet prvkov množiny f";
  51      n=card{p};
  52      k=card{f};
  53
  54    parameters
  55        odhad_y[p]  "odhadnutá hodnota y v danom pozorovaní (X*b)"
  56        rezidua[p]  "odchylky odhadnytých hodnôt y od reálnych hodnôt y"
  57        priemer_y   "aritmetický priemer odhadovanej premennej"
  58        sd_y        "smerodajná odchylka odhadovanej premennej"
  59        SST         "Sum of Squared Totals"
  60        SSE         "Sum of Squared Estimates"
  61        SSR         "Sum of Squared Residuals";
  62      odhad_y[p]  = sum{f,b.l[f]*x[p,f]};
```

```
  63      rezidua[p]  = y[p]-odhad_y[p];
  64      priemer_y   = sum{p,y[p]}/n;
  65      SST         = sum{p,sqr(y[p]-priemer_y)};
  66      sd_y        = sqrt(SST/(n-1));
  67      SSE         = sum{p,sqr(odhad_y[p]-priemer_y)};
  68      SSR         = sum{p,sqr(rezidua[p])};
  69
  70    parameters
  71        R_2         "R-squared statistic"
  72        adj_R2      "adjusted R-squared statistic"
  73        SE          "Standard Error of regression"
  74        DW          "Durbin-Watson statistic"
  75        log_like    "hodnota funkcie log likelihood"
  76        AIC         "Akaike Information Criterion"
  77        SC          "Schwarz Criterion"
  78        F_stat      "F-statistic";
  79      R_2    = SSE/SST;
  80      adj_R2   = 1-((n-1)/(n-k))*(1-R_2);
  81      SE       = sqrt(SSR/(n-k));
  82      DW       = sum{p$(ord{p}>1),sqr(rezidua[p]-rezidua[p-1])}/SSR;
  83      log_like = -(n/2)*(1+log(2*pi)+log(SSR/n));
  84      AIC      = -2*log_like/n + 2*k/n;
  85      SC       = -2*log_like/n + k*log(n)/n;
  86      F_stat   =  (R_2/(k-1))/((1-R_2)/(n-k));
  87
  88    option decimals=6;
  89    option b:3:0:1;
  90    display
  91      h_uf.l,b.l
  92      R_2,adj_R2,SE,DW,log_like,AIC,SC,F_stat;
GAMS Rev 141  Intel /MS Window
26.04.06 00:42:02 Page 2
G e n e r a l   A l g e b r a i c   M o d e l i n g   S y s t e m
Include File Summary


   SEQ   GLOBAL TYPE      PARENT   LOCAL  FILENAME

     1        1 INPUT          0       0  D:\Skola\Diplomovka\Gams subory\mns dip.gms
     2       35 CALL           1      35  Gdxxrw data_mns.xls par=x Rng=a2:d68 rdim=1 cdim=1
     3       36 GDXIN          1      36  D:\Skola\Diplomovka\Gams subory\data_mns.gdx
     4       40 CALL           1      40  Gdxxrw data_mns.xls par=y Rng=g3:h68 rdim=1
     5       41 GDXIN          1      41  D:\Skola\Diplomovka\Gams subory\data_mns.gdx


COMPILATION TIME     =        1.547 SECONDS    3.2 Mb  WIN216-141 Jan 24, 2005
GAMS Rev 141  Intel /MS Window
26.04.06 00:42:02 Page 3
G e n e r a l   A l g e b r a i c   M o d e l i n g   S y s t e m
Equation Listing    SOLVE mns Using NLP From line 46


---- uf  =E=  účelová funkcia

uf..  (1542)*b(0) + (1716.8)*b(1) - (0.225490286958529)*b(2) + h_uf =E= 0 ; (LHS = -24367, INFES
= 24367 ***)

GAMS Rev 141  Intel /MS Window
26.04.06 00:42:02 Page 4
G e n e r a l   A l g e b r a i c   M o d e l i n g   S y s t e m
Column Listing      SOLVE mns Using NLP From line 46


---- b  vektor koeficientov, ktoré sú odhadmi pre zložky vektora beta

b(0)
                (.LO, .L, .UP = -INF, 0, +INF)
    (1542)      uf
```

```
b(1)
                (.LO, .L, .UP = -INF, 0, +INF)
    (1716.8)    uf

b(2)
                (.LO, .L, .UP = -INF, 0, +INF)
      (-0.2255) uf


---- h_uf  hodnota účelovej funkcie = suma štvorcov reziduí

h_uf
                (.LO, .L, .UP = -INF, 0, +INF)
        1       uf

GAMS Rev 141  Intel /MS Window
26.04.06 00:42:02 Page 5
G e n e r a l   A l g e b r a i c   M o d e l i n g   S y s t e m
Model Statistics    SOLVE mns Using NLP From line 46


MODEL STATISTICS

BLOCKS OF EQUATIONS       1     SINGLE EQUATIONS        1
BLOCKS OF VARIABLES       2     SINGLE VARIABLES        4
NON ZERO ELEMENTS         4     NON LINEAR N-Z          3
DERIVATIVE POOL          12     CONSTANT POOL         161
CODE LENGTH            1244


GENERATION TIME      =        0.031 SECONDS    3.9 Mb  WIN216-141 Jan 24, 2005


EXECUTION TIME       =        0.031 SECONDS    3.9 Mb  WIN216-141 Jan 24, 2005
GAMS Rev 141  Intel /MS Window
26.04.06 00:42:02 Page 6
G e n e r a l   A l g e b r a i c   M o d e l i n g   S y s t e m
Solution Report     SOLVE mns Using NLP From line 46


               S O L V E      S U M M A R Y

     MODEL   mns                 OBJECTIVE  h_uf
     TYPE    NLP                 DIRECTION  MINIMIZE
     SOLVER  CONOPT              FROM LINE  46

**** SOLVER STATUS     1 NORMAL COMPLETION
**** MODEL STATUS      2 LOCALLY OPTIMAL
**** OBJECTIVE VALUE             6636.9308

 RESOURCE USAGE, LIMIT          0.063      1000.000
 ITERATION COUNT, LIMIT         8         10000
 EVALUATION ERRORS              0             0


    C O N O P T 3   Intel /MS Window version 3.14E-015-053
    Copyright (C)   ARKI Consulting and Development A/S
                    Bagsvaerdvej 246 A
                    DK-2880 Bagsvaerd, Denmark

 Using default options.


 ** Optimal solution. Reduced gradient less than tolerance.


 CONOPT time Total                            0.063 seconds
```

```
   of which: Function evaluations            0.000 =  0.0%
             1st Derivative evaluations      0.000 =  0.0%
             Directional 2nd Derivative      0.000 =  0.0%

 Work length =    0.10 Mbytes
    Estimate =    0.10 Mbytes
    Max used =    0.06 Mbytes

                            LOWER          LEVEL          UPPER         MARGINAL

---- EQU uf                   .              .              .             1.0000

  uf  účelová funkcia

---- VAR b  vektor koeficientov, ktoré sú odhadmi pre zložky vektora beta

         LOWER          LEVEL          UPPER         MARGINAL

0        -INF           20.9046        +INF            EPS
1        -INF           -0.4037        +INF            EPS
2        -INF       -17377.5493        +INF            EPS

                            LOWER          LEVEL          UPPER         MARGINAL

---- VAR h_uf                -INF         6636.9308        +INF             .

  h_uf  hodnota účelovej funkcie = suma štvorcov reziduí


**** REPORT SUMMARY :        0     NONOPT
                             0 INFEASIBLE
                             0  UNBOUNDED
                             0     ERRORS
GAMS Rev 141  Intel /MS Window
26.04.06 00:42:02 Page 7
G e n e r a l   A l g e b r a i c   M o d e l i n g   S y s t e m
E x e c u t i o n


----     91 VARIABLE h_uf.L                =  6636.930818  hodnota účelovej funkcie = suma
štvorcov reziduí

----     91 VARIABLE b.L  vektor koeficientov, ktoré sú odhadmi pre zložky vektora beta

0     20.905
1     -0.404
2 -17377.549


----     91 PARAMETER R_2                  =      0.567917  R-squared statistic
            PARAMETER adj_R2               =      0.554200  adjusted R-squared statistic
            PARAMETER SE                   =     10.263923  Standard Error of regression
            PARAMETER DW                   =      1.787653  Durbin-Watson statistic
            PARAMETER log_like             =   -245.804699  hodnota funkcie log likelihood
            PARAMETER AIC                  =      7.539536  Akaike Information Criterion
            PARAMETER SC                   =      7.639066  Schwarz Criterion
            PARAMETER F_stat               =     41.402677  F-statistic


EXECUTION TIME       =        0.000 SECONDS    2.9 Mb  WIN216-141 Jan 24, 2005


USER: GAMS Development Corporation, Washington, DC   G871201/0000CA-ANY
      Free Demo,  202-342-0180,  sales@gams.com,  www.gams.com   DC0000


**** FILE SUMMARY
```

```
Input      D:\Skola\Diplomovka\Gams subory\mns dip.gms
Output     D:\Skola\Diplomovka\Gams subory\mns dip.lst
```

Konkrétny model Markowitzovej teórie portfólia

```
GAMS Rev 141  Intel /MS Window
26.04.06 00:41:38 Page 1
G e n e r a l   A l g e b r a i c   M o d e l i n g   S y s t e m
C o m p i l a t i o n


   1
   2
   3    sets
   4        t     "mesiace"          /0*12/
   5        n     "zložky portfólia" /1,2,3/
   6        t1[t] "množina 't' bez prvku 0";
   7      t1[t]=yes;
   8      t1['0']=no;
   9      alias{n,n1};
  10
  11    table
  12        x[t,n] "cena jednotlivých zložiek v jednotlivých časoch"
  13                     1         2         3
  14              0    18.1     102.81    91.44
  15              1    22.04     99.5     95.69
  16              2    19.81    100.44    93.62
  17              3    22.12    111.31    92.69
  18              4    22.06    101.62    95.06
  19              5    26.44    108       94.69
  20              6    29       108.75    94.94
  21              7    25.19    111.25   100.19
  22              8    27       110.37    97.31
  23              9    30.69    115.32    92.88
  24             10    24.94    113.27    97.44
  25             11    27.88    118.69    95.75
  26             12    29.2     114.75    96.81   ;
  27
  28    scalar
  29        er "očakávaný ročný výnos" /0.1/;

  30      er=er/12;
  31
  32    parameters
  33        r[t,n]    "relatívny mesačný výnos v 'i'-tom mesiaci kde 'i' je z 't' "
  34        ar[n]     "priemerný mesačný výnos"
  35        cov[n,n1] "variančno-kovariančná matica";
  36      r[t,n]$(ord{t}>1)=(x[t,n]-x[t-1,n])/x[t-1,n];
  37      ar[n]=sum{t1,r[t1,n]}/card{t1};
  38      cov[n,n1]=sum{t1,(r[t1,n]-ar[n])*(r[t1,n1]-ar[n1])}/(card{t1}-1);
  39
  40    positive variable
  41        w[n] "váhy jednotlivých zložiek v portfóliu";
  42    free variable
  43        var  "variancia portfólia = hodnota účelovej funkcie";
  44
  45    equations
  46        vf_min "definícia účelovej funkcie"
  47        ere    "ohraničenie pre očakavaný výnos"
  48        sw     "podmienka jednotkového súčtu váh";
  49      vf_min ..   var =e= sum{[n,n1],w[n]*w[n1]*cov[n,n1]};
  50      ere    ..           sum{n,w[n]*ar[n]} =g= er;
```

```
  51      sw     ..                 sum{n,w[n]} =e= 1;
  52
  53    model markowitz /vf_min,ere,sw/;
  54
  55    solve markowitz minimizing var using nlp;
  56
  57    display w.l;


COMPILATION TIME     =        0.000 SECONDS    3.2 Mb  WIN216-141 Jan 24, 2005
GAMS Rev 141  Intel /MS Window
26.04.06 00:41:38 Page 2
G e n e r a l   A l g e b r a i c   M o d e l i n g   S y s t e m
Equation Listing    SOLVE markowitz Using NLP From line 55


---- vf_min  =E=  definícia účelovej funkcie

vf_min..  (0)*w(1) + (0)*w(2) + (0)*w(3) + var =E= 0 ; (LHS = 0)


---- ere  =G=  ohraničenie pre očakavaný výnos

ere..  0.0483983412556312*w(1) + 0.0104128660620014*w(2) + 0.00525663640372498*w(3) =G=
0.00833333333333333 ;

      (LHS = 0, INFES = 0.00833333333333333 ***)


---- sw  =E=  podmienka jednotkového súčtu váh

sw..  w(1) + w(2) + w(3) =E= 1 ; (LHS = 0, INFES = 1 ***)

GAMS Rev 141  Intel /MS Window
26.04.06 00:41:38 Page 3
G e n e r a l   A l g e b r a i c   M o d e l i n g   S y s t e m
Column Listing      SOLVE markowitz Using NLP From line 55


---- w  váhy jednotlivých zložiek v portfóliu

w(1)
                (.LO, .L, .UP = 0, 0, +INF)
       (0)      vf_min
        0.0484  ere
        1       sw

w(2)
                (.LO, .L, .UP = 0, 0, +INF)
       (0)      vf_min
        0.0104  ere
        1       sw

w(3)
                (.LO, .L, .UP = 0, 0, +INF)
       (0)      vf_min
        0.0053  ere
        1       sw


---- var  variancia portfólia = hodnota účelovej funkcie

var
                (.LO, .L, .UP = -INF, 0, +INF)
        1       vf_min

GAMS Rev 141  Intel /MS Window
26.04.06 00:41:38 Page 4
```

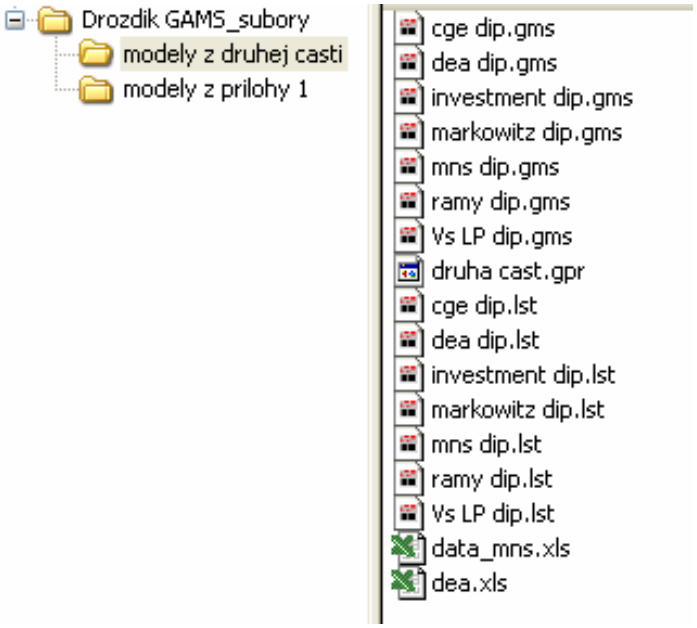
```
G e n e r a l  A l g e b r a i c  M o d e l i n g  S y s t e m
Model Statistics    SOLVE markowitz Using NLP From line 55


MODEL STATISTICS

BLOCKS OF EQUATIONS       3     SINGLE EQUATIONS        3
BLOCKS OF VARIABLES       2     SINGLE VARIABLES        4
NON ZERO ELEMENTS        10     NON LINEAR N-Z          3
DERIVATIVE POOL          11     CONSTANT POOL          20
CODE LENGTH             117


GENERATION TIME      =        0.000 SECONDS    3.9 Mb  WIN216-141 Jan 24, 2005


EXECUTION TIME       =        0.000 SECONDS    3.9 Mb  WIN216-141 Jan 24, 2005
GAMS Rev 141  Intel /MS Window
26.04.06 00:41:38 Page 5
G e n e r a l  A l g e b r a i c  M o d e l i n g  S y s t e m
Solution Report     SOLVE markowitz Using NLP From line 55


               S O L V E     S U M M A R Y

     MODEL   markowitz           OBJECTIVE  var
     TYPE    NLP                 DIRECTION  MINIMIZE
     SOLVER  CONOPT              FROM LINE  55

**** SOLVER STATUS     1 NORMAL COMPLETION
**** MODEL STATUS      2 LOCALLY OPTIMAL
**** OBJECTIVE VALUE                0.0004

 RESOURCE USAGE, LIMIT          0.094      1000.000
 ITERATION COUNT, LIMIT         8         10000
 EVALUATION ERRORS              0             0


    C O N O P T 3   Intel /MS Window version 3.14E-015-053
    Copyright (C)   ARKI Consulting and Development A/S
                    Bagsvaerdvej 246 A
                    DK-2880 Bagsvaerd, Denmark

 Using default options.


 ** Optimal solution. Reduced gradient less than tolerance.


 CONOPT time Total                            0.094 seconds
   of which: Function evaluations             0.000 =  0.0%
             1st Derivative evaluations       0.000 =  0.0%

 Work length =    0.03 Mbytes
    Estimate =    0.03 Mbytes
    Max used =    0.01 Mbytes

                            LOWER          LEVEL          UPPER         MARGINAL

---- EQU vf_min              .              .              .             1.0000
---- EQU ere                0.0083         0.0091        +INF            .
---- EQU sw                 1.0000         1.0000         1.0000         0.0008

  vf_min  definícia účelovej funkcie
  ere  ohraničenie pre očakavaný výnos
  sw  podmienka jednotkového súčtu váh

---- VAR w  váhy jednotlivých zložiek v portfóliu
```

```
         LOWER          LEVEL          UPPER         MARGINAL

1          .             0.0548        +INF            EPS
2          .             0.2920        +INF             .
3          .             0.6532        +INF            EPS

                           LOWER          LEVEL          UPPER         MARGINAL

---- VAR var               -INF            0.0004        +INF             .

  var  variancia portfólia = hodnota účelovej funkcie


**** REPORT SUMMARY :        0     NONOPT
                             0 INFEASIBLE
                             0  UNBOUNDED
                             0     ERRORS
GAMS Rev 141  Intel /MS Window
26.04.06 00:41:38 Page 6
G e n e r a l   A l g e b r a i c   M o d e l i n g   S y s t e m
E x e c u t i o n


----     57 VARIABLE w.L  váhy jednotlivých zložiek v portfóliu

1 0.055,    2 0.292,    3 0.653


EXECUTION TIME       =        0.000 SECONDS    2.9 Mb  WIN216-141 Jan 24, 2005


USER: GAMS Development Corporation, Washington, DC   G871201/0000CA-ANY
      Free Demo,  202-342-0180,  sales@gams.com,  www.gams.com   DC0000


**** FILE SUMMARY

Input      D:\Skola\Diplomovka\Gams subory\markowitz dip.gms
Output     D:\Skola\Diplomovka\Gams subory\markowitz dip.lst
```

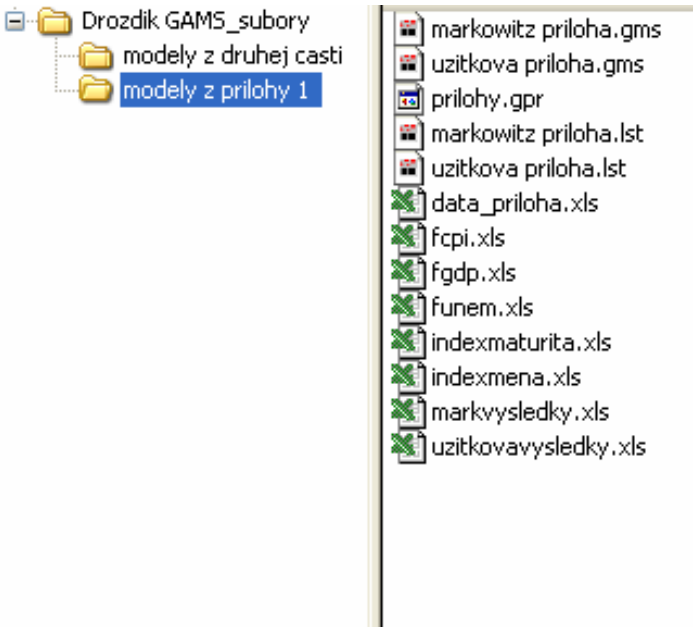
Príloha 3 – CD obsahujúce súbory všetkých programov uvedených v tejto práci

Štruktúra CD:

Ak chcete skompilovať niektorý z uvedených programov, tak si najskôr otvorte gamsovský projekt z príslušného adresára (modely z druhej casti – druha cast.gpr , modely z prilohy 1 – prilohy.gpr) a až potom, v rámci daného projektu, žiadaný súbor s príponou .gms z toho istého adresára.



– Program k CGE modelu
– Program k DEA modelu
– Program k modelu Investovanie
– Program k Markow. modelu
– Program k Griffinovmu LRM
– Program k modelu Rámy na ...
– Program všeob. úlohy LP
– Gamsovský projekt
– Výstup k ...
– Výstup k ...
– Výstup k ...
– Výstup k ...
– Výstup k ...
– Výstup k ...
– Výstup k ...
– Vstupné dáta pre Griffinov LRM
– Výsledky DEA modelu (export)



– Program k Markow. modelu
– Program k modelu s f. užitočnosti
– Gamsovský projekt
– Výstup ...
– Výstup ...
– Vstupné dáta pre oba modely
– Vstupné dáta pre faktor CPI
– Vstupné dáta pre faktor HDP
– Vstupné dáta pre faktor nezamest.
– Vstupné dáta indexov podľa mat.
– Vstupné dáta indexov podľa meny
– Výsledky Markowitzovho modelu
– Výsledky modelu s f. užitočnosti