EKONOMICKÁ UNIVERZITA V BRATISLAVE
FAKULTA HOSPODÁRSKEJ INFORMATIKY

Evidenčné číslo: 103003/I/2016/3781255981

IMPLEMENTÁCIA MARKOVOVÝCH ROZHODOVACÍCH PROCESOV

DIPLOMOVÁ PRÁCA

2016 **Bc. Zuzana Machová**

EKONOMICKÁ UNIVERZITA V BRATISLAVE
FAKULTA HOSPODÁRSKEJ INFORMATIKY

IMPLEMENTÁCIA MARKOVOVÝCH ROZHODOVACÍCH PROCESOV

DIPLOMOVÁ PRÁCA

Študijný program:	Informačný manažment
Študijný odbor:	Kvantitatívne metódy v ekonómii
Školiace pracovisko:	Katedra operačného výskumu a ekonometrie
Vedúci bakalárskej práce:	Ing. Marián Reiff, PhD.

Bratislava 2016 **Bc. Zuzana Machová**

Zadaanie zaverecnej práce

Čestné vyhlásenie

Čestne vyhlasujem, že záverečnú prácu som vypracovala samostatne, a že som uviedla všetku použitú literatúru.

Bratislava, apríl 2016

.....

Zuzana Machová

Poďakovanie

Poďakovanie patrí vedúcemu mojej diplomovej práce Ing. Marián Reiff, PhD., ktorý mi počas vypracovania poskytol cenné rady a pripomienky. Ďalej ďakujem mojej rodine a priateľom za podporu počas štúdia a vypracovávania tejto záverečnej práce.

ABSTRAKT

MACHOVÁ, Zuzana: *Implementácia Markovových rozhodovacích procesov*. – Ekonomická univerzita v Bratislave. Fakulta hospodárskej informatiky; Katedra operačného výskumu a ekonometrie. – Ing. Marián Reiff, PhD.. Bratislava: FHI EU, 2016, 65 s.

Cieľom diplomovej práce je tvorba programovej aplikácie na základe modelov Markovových rozhodovací procesov. Práca je rozdelená do štyroch kapitol. Obsahuje 30 obrázkov a 9 tabuliek. V prvá kapitola sa venuje problematike Markovových rozhodovacích procesov, základné vlastnosti a pojmy z oblasti Markovových procesov. Jedná sa o teóriu potrebnú k pochopeniu základných princípov a spôsobov výpočtu úloh. Taktiež čitateľa oboznamuje s užívateľským prostredím systému MATLAB. V ďalšej kapitole sa bližšie charakterizuje cieľ práce. Tretia kapitola je venovaná spôsobu tvorbu užívateľskej aplikácie v programe MATLAB. Záverečná kapitola je venovaná výsledkom práce, kde sa opisuje vytvorená aplikácia, ktorá je určená na riešenie úloh Markovových rozhodovacích procesov.

Kľúčové slová: Markovove rozhodovacie procesy, MATLAB, GUI, GUIDE, matica pravdepodobností prechodov, matica ocenení

ABSTRACT

MACHOVÁ, Zuzana: *Implementation of Markov decision process*. – University of Economics in Bratislava. Faculty of Economic Informatics; Department of Operation Research and Econometrics. – Ing. Marián Reiff, PhD. Bratislava: FHI EU, 2016, 65 s.

The main aim of master thesis is development of application based on Markov decision processes. The thesis is divided into four chapters. Contains 30 pictures and 9 tables. In first chapter is focused on Markov decision processes, main characteristics and concepts from the Markov processes. It is the theory needed for understanding the basic principles and methods for calculating tasks. Also, we provide hints to reader about user interface of MATLAB. In the next chapter we characterize more goal of thesis. Third part devotes to the methods of creating user application with MATLAB. Last chapter, work results we describe application which we developed for solving Markov decision processes.

Key words: Markov decision processes, MATLAB, GUI, GUIDE, transition probability matrix, reward matrix

Obsah

Zoznam obrázkov

Zoznam tabuliek

Úvod

„*Každý deň a každú minútu sa musím rozhodovať, čo urobím v najbližšej chvíli. Toto rozhodnutie za mňa nemôže spraviť nikto iný.*"

José Ortega y Gasset

Zamyslime sa nad spôsobom, akým učíme psa sadnúť. Zvyčajne to skúšame pomocou jednoduchého príkazu „sadni", ale vo väčšine prípadov to ani po niekoľkých pokusoch nefunguje. Potom, z ničoho nič, pes poslúchne príkaz a sadne si. Teraz by ste sa mali snažiť posilniť takéto dobré správanie nejakým typom odmeny, napríklad sušienkou, a časom si pes zvykne, že vždy keď poslúchne príkaz „sadni" dostane sušienku.

Mohli by sme teraz túto myšlienku túžby po odmene aplikovať pomocou matematického algoritmu? V skutočnosti je tu táto idea od vzniku počítačov a zložitosť úloh, ktoré počítače ovládajú sa vyvíjajú postupom času. V tejto diplomovej práci sa zameriame na matematický uhoľ pohľadu tohto problému a taktiež na vývoj programu, ktorý nám umožní rýchlejší výpočet daného matematického algoritmu.

Budeme sa zaoberať teóriou založenou na Markovovej teórií. Povedzme, že máme Markovov reťazec s diskrétnym časom, v najjednoduchšom prípade je každému kroku priradené rozdelenie pravdepodobností prechodov medzi jednotlivými stavmi tohoto stochastického procesu. Prechod medzi stavmi si môžeme predstaviť ako akciu, ktorú musí proces vykonať. Po každej akcii a prechode do ďalšieho stavu, proces ohodnotíme určitým ocenením. Problém nastáva v nájdení takej množiny akcií, nazývanej stratégia, ktorá maximalizuje celkovú odmenu prechodu medzi jednotlivými stavmi procesu. Proces s takýmito vlastnosťami nazývame Markovov rozhodovací proces (MRP).

Markovove rozhodovacie procesy, pomenované po ruskom matematikovi Andreyovi Markovovi [8]. Prvá zmienka o MRP sa objavila v 50-tych rokoch minulého storočia. V 60-tych rokoch 20. storočia naviazal na Bellmanov koncept dynamického programovania Howard [1], ktorý v roku 1960 odvodil algoritmus pre iteračný postup pre nájdenie optimálneho riadenia dynamického systému pracujúceho v nekonečnom časom horizonte. Poskytujú matematický rámec pre modelovanie komplexných, sekvenčných rozhodovacích problémov, ktoré vznikajú v operačnom výskume, manažmente, v oblasti financií, doprave, a mnohých ďalších odvetviach. Markovove rozhodovacie procesy tvoria stavy, akcie, pravdepodobnosti prechodov a ocenení. Proces sa vždy nachádza v nejakom stave

a poskytuje rozhodovateľovi všetky potrebné informácie pre výber z možných akcií v danom stave. Na takéto rozhodnutie proces reaguje prechodom do ďalšieho stavu na základe zvolenej akcie a získava odmenu (kladnú alebo zápornú). Hlavným cieľom je nájsť takú stratégiu, ktorá bude optimalizovať odmeny získané počas prechodu systémom.

V prvej kapitole sa snažíme čitateľa uviesť do problematiky Markovových rozhodovacích procesov. Definujeme základné vlastnosti a pojmy z oblasti diskrétnych a spojitých Markovových procesov. V práci zameriavame najmä na Markovove procesy s diskrétnym časom. Jedná sa o teóriu potrebnú k pochopeniu základných princípov a spôsobu výpočtu úloh. Taktiež čitateľa oboznámime s užívateľským prostredím systému MATLAB a stručne predstavíme M-súbory a riadiace štruktúry.

V nasledujúcej kapitole detailnejšie definujeme hlavné a čiastkové ciele práce. Hlavným cieľom diplomovej práce je tvorba programovej aplikácie na základe modelov Markovových rozhodovací procesov použiteľnú pri analýze zvoleného problému. Aplikáciu pre výpočet úloh MRP sme implementovali v systéme MATLAB.

Tretia kapitola je venovaná metodike práce a metódam skúmania. Čitateľa tu oboznamujeme s nástrojom GUIDE(z anglického Graphical User Interface Development Environment), ktorý je súčasťou MATLABu a slúži na tvorbu programových aplikácií. Pomocou GUIDE sme vytvorili programovú aplikáciu pre riešenie úloh MRP. Ďalej popisujeme postup pri tvorbe a programovaní GUI (z anglického Graphical User Interface) aplikácie.

Kapitola venovaná výsledkom práce je zároveň poslednou kapitolou, v ktorej opisujeme nami vytvorenú aplikáciu. Venujeme sa tu detailnému popisu užívateľského prostredia aplikácie a zároveň popisu niektorých častí zdrojového kódu. V závere kapitoly sme demonštrovali fungovanie aplikácie na probléme vodiča taxíka.

V závere sa zaoberáme zhodnotením práce a odporúčaniam pre ďalší vývoj programovej aplikácie, možnostiam pre jej rozvoj a udržiavanie aj v budúcnosti.

1 Súčasný stav riešenej problematiky doma a v zahraničí

V prvej kapitole sa budeme venovať základným pojmom potrebným pre pochopenie Markovových rozhodovacích procesov. Oboznámime sa taktiež s programovým prostriedkom MATLAB, ktorý je aj programovacím jazykom, ktorý využijeme pri tvorbe aplikácie.

1.1 Stochastické modely

Stochastické modely sú modely, v ktorých niektoré veličiny (premenné) sú náhodné, a ktoré používajú metódy počtu pravdepodobností. Sú teda založené na aplikácii počtu pravdepodobností.

Z hľadiska využívania počtu pravdepodobností sa v rámci operačného výskumu (operačnej analýzy) rozlišujú:

1. **Deterministické modely**, kde náhodné výkyvy veličín a vzťahov medzi nimi zanedbávame.
2. **Stochastické (pravdepodobnostné) modely**, definované vyššie.

Hlavným nástrojom na modelovanie stochastických procesov sú náhodné (stochastické) procesy. Priebeh náhodného procesu je pri každom jeho uskutočnení iný. Jednotlivé prípady uskutočnenia náhodných procesov nazývame **realizácia náhodného procesu.**

Stochastické procesy teda môžeme definovať ako množinu náhodných veličín $\{X_t, t \in T\}$ kde X_t reprezentuje množinu náhodných premenných v čase t a T je množina reálnych čísel, v ktorej sa parameter mení. V závislosti na charaktere množiny T sa stochastické procesy rozdeľujú na *procesy s diskrétnym časom* (T je spočítateľná množina), alebo na *procesy so spojitým časom* (T je interval).

Stochastický proces má často nasledujúcu štruktúru :

„Súčasný stav systému môže spadnúť do ktorejkoľvek z $m + 1$ vzájomne sa vylučujúcich kategórií nazývaných stavy. Tieto stavy sú označované $0, 1, 2, \dots, m$. Náhodná veličina X_t reprezentuje stav systému v čase *t*, preto jej jedinými možnými hodnotami sú $0, 1, 2, \dots, m$. Systém je pozorovaný v určitých časových intervaloch, $t = 0, 1, 2, \dots$. Teda, stochastický proces $\{X_t\} = \{X_0, X_1, X_2, \dots\}$ poskytuje matematickú reprezentáciu ako sa systém vyvíja v priebehu času. Tento typ procesu je označovaný ako **stochastický proces s diskrétnym časom** s konečným počtom stavov.“[1]

[1] Hiller, F., Lieberman, G.: *Introduction to Operation Research*. 2015. s.802

1.2 Markovove procesy

Závislosť jednotlivých stavov procesu skúma teória náhodných procesov spôsobom, pri ktorom vytvára modely špecifických náhodných procesov, ktoré sa vyznačujú istým typom závislosti medzi stavmi procesu v jednotlivých časových okamihoch. Najpodstatnejšiu úlohu v tejto súvislosti majú tzv. Markovove procesy.

Základnou vlastnosťou Markovových procesov je skutočnosť, že vývoj náhodného procesu do budúcnosti je určený iba súčasným stavom a nezáleží na tom, ako sa do tohoto stavu náhodný proces dostal. Pre popis správania systému používame teda podmienené pravdepodobnosti resp. pravdepodobnosti prechodu.

Stochastický proces $\{X_t, t \in T\}$, je **Markovov proces** ak pre $n = 1, 2, 3, \ldots$ a ľubovoľné hodnoty prametra $t \in T$ a pre ľubovoľné reálne čísla x, y platí rovnica:

$$P\{ X_{t_n} < y | X_{t_{n-1}} = x, X_{t_{n-2}} = x_{n-2}, \ldots, X_{t_1} = x_1, X_{t_0} = x_0 \} = \\ = P\{ X_{t_n} < y | X_{t_{n-1}} = x \} \tag{1}$$

pre všetky $x_{n-2}, \ldots, x_0$.

Pravdepodobnosť prechodu $p_{ij}(t_1, t_2)$ je podmienená pravdepodobnosť toho, že Markovov proces bude v čase t_2 v stave j, keď v čase t_1 bol v stave i, pričom $t_1 < t_2$,

$$p_{ij}(t_1, t_2) = P\{X_{t_2} = j | X_{t_1} = i\}. \tag{2}$$

Hovoríme, že Markovov proces $\{X_t, t \in T\}$ je ***homogénny***, ak pravdepodobnosť prechodu $p_{ij}(t_1, t_2)$ pre všetky $i, j \in I$ a $t_1, t_2 \in T$, závisí iba na rozdiele $t = t_2 - t_1$.

Ak $\{X_{t_1}, t \in T\}$ je homogénny Markovov proces s konečným počtom stavov $0,1,2, \ldots, n$ má limity,

$$\lim_{t \to \infty} p_{ij}(t) = p_j \tag{3}$$

pre všetky $i, j = 0,1, \ldots, n$.

1.3 Markovove reťazce

Najjednoduchším typom Markovových procesov sú Markovove reťazce, ktoré nadobúdajú konečný alebo nekonečný spočítateľný počet možných hodnôt (stavov). Majme náhodný proces s diskrétnym parametrom (časom). Množina T je konečná resp. spočítateľná množina. Predpokladajme, že $T = \{0,1, \ldots N\}$ resp. $T = \{0,1, \ldots, N, \ldots\}$. Nech množina I je

spočítateľná množina stavov, kde $i = 0,1,2, \ldots$. Hovoríme, že stochastický proces X_t je v čase t v stave $i \in I$, ak $X_t = i$.

Stochastický proces $\{X_t, t \in T\}$ je **Markovov reťazec**, ak pre všetky $t \in T$ je X_t diskrétna celočíselná náhodná veličina a súčasne platí,

$$P\{X_{t+1} = j | X_0 = k_0, X_1 = k_1, \ldots, X_{t-1} = k_{t-1}, X_t = i\} = \\ = P\{X_{t+1} = j | X_t = i\} = p_{ij} \tag{4}$$

pre všetky $t = 0, 1, 2, \ldots$ a $i, j, k_0, k_1, \ldots, k_{t-1}$.

Uvedená podmienka sa nazýva **Markovova vlastnosť**, podľa ktorej vývoj náhodného procesu do budúcnosti je určený iba súčasným stavom $X_t = i_t$ pričom nazáleží na tom, ako sa do tohto stavu náhodný proces dostal t. j. nie je závislý na minulých stavoch $X_{t-1}, X_{t-2} \ldots, X_1, X_0$.

Ak pravdepodobnosť prechodu závisí od času t, t. j. pravdepodobnosti prechodu $p_{ij}(t)$ sú vo všeobecnosti rozdielne, ide o ***nehomogénny*** Markovov reťazec.

Markovov reťazec, pre ktorý pravdepodobnosti $p_{ij}(t)$ nezávisia na t a $p_{ij}(t) = p_{ij}$ pre všetky i, j sa nazýva ***homogénny*** Markovov reťazec.

Ďalej sa budeme zaoberať iba prípadom homogénnych reťazcov.

Vo všeobecnosti môže systém prechádzať z ľubovoľného i-tého stavu do ľubovoľného j-tého stavu $(i, j = 1,2, \ldots, m)$. Preto je nutné poznať všetky pravdepodobnosti prechodu medzi jednotlivými stavmi. Jednotlivé pravdepodobnosti prechodov môžeme usporiadať do tzv. **matice pravdepodobností prechodov P**,

$$P = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1m} \\ p_{21} & p_{22} & \cdots & p_{2m} \\ \vdots & \vdots & & \vdots \\ p_{m1} & p_{m2} & \cdots & p_{mm} \end{bmatrix} \tag{5}$$

Matica pravdepodobností prechodov po jednom kroku je štvorcová a jej rozmer je rovný počtu stavov v množine I. Matice, pre prvky ktorej sú splnené podmienky:

$$p_{ij} \geq 0 \qquad i, j > 0$$

$$\sum_{j=1}^{\infty} p_{ij} = 1 \qquad i = 1, 2, \ldots,$$

sú stochastické matice. Matica (5) je teda stochastická matica.

1.3.1 Chapmanova-Kolmogorova rovnica

„Pri analýze dynamiky systémov sa nemožno obmedziť len na vývoj v jednom kroku. Pravdepodobnosť toho, že systém sa dostane zo stavu i do stavu j po n krokoch, sa nazýva **pravdepodobnosť prechodu po n krokoch** $\boldsymbol{p_{ij}^{(n)}}$. Rovná sa,

$$p_{ij}^{(n)} = P\{X_{n+m} = j | X_m = i\} \tag{6}$$

$n \neq 0$, $i, j \geq 0$.“[2]

Na výpočet pravdepodobností prechodu po n krokoch slúži **Chapmanova-Kolmogorova rovnica**:

$$P_{ij}^{(n+m)} = \sum_{k=0}^{\infty} P_{ij}^{(n)} P_{ij}^{(m)} \quad resp. \quad P^{(n+m)} = P^n . P^m \tag{7}$$

pre všetkyn, $m \geq 0$, všetky i, j. Prechod zo stavu i do stavu j sa uskutoční cez $n + m$ medzistavov jednotlivých ciest.

1.3.2 Klasifikácia stavov Markovho reťazca

Vlastnosťami jednotlivých stavov a z nich vytvorených submatíc a matíc pravdepodobností prechodov sa dá vysvetliť správanie sa systému, modelovateľného pomocou Markovových reťazcov.

V prvom rade môžeme stavy systému deliť **podľa možnosti prechodu**, resp. návratu po n krokoch.

1. Stav j je ***dosiahnuteľný*** zo stavu i, ak existuje také prirodzené n, že $p_{ij}^{(n)} > 0$. Ak takéto n neexistuje, potom stav j je nedosiahnuteľný zo stavu i.
2. Stav i je ***súsledný (komunikujúci)*** so stavom j, keď j je dosiahnuteľný z i a naopak. Majme ľubovoľné stavy $i, j, k \in I$ také, že j je dosiahnuteľný z i a k je dosiahnuteľný z j, potom k je dosiahnuteľný z i.
3. Stav i nazývame ***absorpčný***, ak existuje $p_{ii}^{(n)}$. V takomto prípade, po jeho dosiahnutí už neexistuje možnosť prechodu do ďalšieho stavu.

Ďalšou dôležitou klasifikáciou stavov je klasifikácia **podľa pravdepodobnosti návratu**, je to pravdepodobnosť, že ak proces začne v stave i tak po určitom čase sa znovu vráti do tohto stavu. Podľa pravdepodobnosti návratu delíme stavy na:

[2] Unčovský, L., Čemická K.: *Stochastické procesy a modely*.1992. s.31

1. ***Rekurentné (návratné, permanentné)***, stav, do ktorého sa proces počas prevádzky zaručene vráti, pravdepodobnosť návratu do rekurentného stavu je $p = 1$.
   Rekurentné stavy sa ďalej delia podľa periodickosti:
   - Stav i nazývame *periodický* s periódou d_i, ak komunikuje sám so sebou a najväčší spoločný deliteľ čísel $n \geq 1$, pre ktoré $p_{ii}^{(n)} > 0$, je d_i.
   - Ak $d_i = 1$ t. j. stav má periódu 1, hovoríme, že stav je *aperiodický*.
2. ***Transientné (prechodné, nenávratné)***, stav, do ktorého sa proces v priebehu prevádzky nemusí vrátiť, pravdepodobnosť návratu do transientného stavu je $p < 1$.

Ak všetky stavy v reťazci sú rekurentné, aperiodické a navzájom komunikujúce, hovoríme, že reťaz je **ergodická**.

Triedy stavov definujeme nasledujúcim spôsobom, stav $i \in I$ tvorí sám triedu, ak neexistuje iný stav s ním súsledný. Stavy, pre ktoré existuje súsledný stav sa rozdelia do množín vzájomne súsledných stavov. Každá takáto množina tvorí triedu stavov. Triedu stavov obsahujúcu stav i označíme C_i.

Množina stavov I je **uzavretá** ak žiadny so stavov mimo množiny I nie je súsledný zo žiadnym stavom v množine I.

1.4 Markovove rozhodovacie procesy

V každom stave Markovovho reťazca sa rozhodujeme o tom, ktorá z akcií (činností, alternatív) by sa mala vykonať v danom stave. Ak je proces v stave i v čase t a zvolila sa akcia a, nasledujúci stav systému sa určí pomocou pravdepodobnosti prechodu $p_{ij}(a)$.

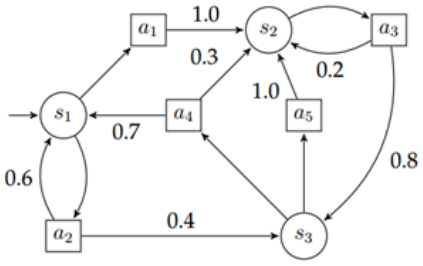
Ak X_t je stav procesu v čase t a a_t je akcia zvolená v čase t, tak platí:

$$P\{X_{t+1} = j | X_0, a_0, X_1, a_1, \dots, X_n = i, a_n = a\} = p_{ij}(a) \tag{8}$$

Markovov rozhodovací proces je definovaný ako štvorica $< \mathrm{S}, \mathrm{A}, \mathrm{P}, \mathrm{R} >$ kde:

- S je množina stavov,
- A je množina akcií,
- $P(i, a, j) = p_{ij}(a) = P\{X_{t+1} = j | X_0, a_0, X_1, a_1, \dots, X_n = i, a_n = a\}$,
  je pravdepodobnosť, že akcia a v stave i v čase t povedie v čase $t + 1$ do stavu j,
- $R(i, a, j) = r_{ij}(a)$ je okamžitý úžitok dosiahnutý po prechode zo stavu i do stavu j s pravdepodobnosťou prechodu $p_{ij}(a)$.

Obrázok 1: Príklad MRP



Zdroj: *Vlastné spracovanie*

Hlavnou úlohou MRP je nájsť **stratégiu** pre užívateľa, funkciu π, ktorá špecifikuje akciu, ktorú zvolí užívateľ v stave i. Cieľom je nájsť takú stratégiu π, ktorá bude maximalizovať kumulatívnu funkciu náhodných úžitkov, typicky očakávanú diskontovanú sumu za potenciálne nekonečnú dobu:

$$\sum_{t=1}^{\infty} \beta^t R(i_t, \pi(i_t), i_{t+1}) \tag{9}$$

kde β je diskontovaný faktor spĺňajúci podmienku $0 < \beta \leq 1$.

1.4.1 Ocenenia prechodov medzi stavmi

Doteraz sme sa zaoberali systémami popísanými Markovovými reťazcami, kde sa rozhodovalo len na základe pravdepodobností p_{ij}. Neuplatňovali sme hľadisko výnosov alebo strát (odmien), ktoré súvisí s jednotlivými krokmi. Predpokladajme, že každý prechod zo stavu i do stavu j je spojený s istou odmenou (ocenením) r_{ij}, vo forme výnosov $r_{ij} > 0$, či nákladov $r_{ij} < 0$. Ocenenie r_{ij}, môže byť konštantné pre všetky prechody, alebo môže byť klesajúcou alebo rastúcou funkciou n.

Celá množina odmien prechodu r_{ij} vytvára **maticu odmien R**, ktorá zodpovedá matici P:

$$R = \begin{bmatrix} r_{11} & r_{12} & \dots & r_{1m} \\ r_{21} & r_{22} & \dots & r_{2m} \\ \vdots & \vdots & & \vdots \\ r_{m1} & r_{m2} & \dots & r_{mm} \end{bmatrix} \tag{10}$$

Odmeny v Markovových reťazcoch umožňujú aj ocenenie alternatív, resp. variantov, ktoré prichádzajú do úvahy. Tento prístup vedie k modelom **Markovových rozhodovacích procesov**.

Podstatným rozdielom medzi odmenami a pravdepodobnosťami je, že odmeny jednotlivých krokov sa spočítavajú s ohľadom na to, že závisia aj od príslušnej pravdepodobnosti prechodu p_{ij}. Výrazom $\boldsymbol{v_i^{(n)}}$ označíme **strednú hodnotu celkového očakávaného výnosu procesu**, ktorý je na začiatku sledovania v i-tom stave a uskutoční n prechodov,

$$v_i^{(n)} = \sum_{j=1}^{m} p_{ij}\, r_{ij} \tag{11}$$

kde $i = 1,2, \dots, n$.

Stredná hodnota sa v súvislosti s Markovovými rozhodovacími procesmi uvádza ***v rekurentnom tvare***, pre $i = 1,2, \dots, n;\ \ n = 1,2,3, \dots$

$$v_i^{(n)} = \sum_{j=1}^{m} p_{ij}(r_{ij} + v_j^{(n-1)}) \tag{12}$$

Pre $i = 1,2, \dots, n$ označme

$$q_i = \sum_{j=1}^{m} p_{ij}\, r_{ij} \tag{13}$$

potom dostaneme vzťah

$$v_i^{(n)} = q_i + \sum_{j=1}^{m} p_{ij}\, v_j^{(n-1)} \tag{14}$$

čo môžeme maticovo vyjadriť ako

$$v(n) = q + P.v(n-1) \tag{15}$$

Veličina $\boldsymbol{q_i}$ vyjadruje **strednú hodnotu bezprostredného výnosu**, t. j. výnosu pri prvom prechode zo stavu i.

Zo vzťahu (12) je zrejmé, že sa jedná o typický rekurentný vzťah dynamického programovania. Preto sa pri výpočtových strednej hodnoty očakávaného výnosu $v_i^{(n)}$ postupuje spätne od posledného obdobia $v_i^{(0)}$ smerom k začiatku.

Pre úspešný výpočet podľa vzťahu (14) je nutné stanoviť hodnotu veličiny $v_i^{(0)}$ pre $i = 1,2, \dots, n$. Nakoľko veličina $\boldsymbol{v_i^{(n)}}$ je stredná hodnota výnosu procesu, ktorá je v stave i a už neuskutoční žiadny prechod, alebo skončí v stave i. Väčšinou samozrejme nezáleží v akom stave proces skončí, a preto je typická hodnota pre stav $v_i^{(0)} = 0$ pre $i = 1,2, \dots, n$.

1.4.2 Meniace sa ocenenia prechodov

Ak celú maticu R vynásobíme určitým koeficientom (typické označenia β), dostávame z konštantného ocenenia prechodov, meniace sa ocenenie. Konštatné ocenenie je teda iba špeciálnym stavom meniaceho sa ocenenia, kde **koeficient $\boldsymbol{\beta}$** je rovný 1. Tento koeficient je nazývaný diskontným faktorom a môžeme ho teda vyjadriť pomocou úrokovej miery $\beta = 1/(1+i)$. Nemusí však slúžiť iba k výpočtu diskontovaných výnosov, možno mu priradiť aj pravdepodobnostný charakter. Môže označovať pravdepodobnosť, s ktorou dôjde k ďalšiemu opakovanému procesu. Pre obidva možné prípady je zhodná jeho vlastnosť a to, že $0 < \beta < 1$.

Pre prípad meniaceho sa ocenenia prechodov možno použiť vzorec výpočtu strednej hodnoty očakávaného výnosu procesu v tvare:

$$v_i^{(n)} = q_i + \beta \sum_{j=1}^{m} p_{ij}\, v_j^{(n-1)} \tag{16}$$

kde:

$$q_i = \sum_{j=1}^{m} p_{ij}\, r_{ij} \tag{17}$$

1.4.3 Rozhodovací proces s alternatívami

Doteraz sme predpokladali pre každý stav iba jednu možnú alternatívu a s ňou spojenú pravdepodobnosť a ocenenie. Pokiaľ však bude týchto alternatív daného stavu viac, povedzme k, potom každá z nich spravidla vyvolá inú pravdepodobnosť prechodu a s ňou príslušné ocenenie prechodu.

Nech D je množina rozhodnutí (alternatív) a pre každé rozhodnutie $k \in D$ je daná pravdepodobnosť prechodu p_{ij}^k a r_{ij}^k ocenenie prechodu pre k-té rozhodnutie. Úlohou je nájsť takú postupnosť rozhodnutí, pre ktorú stredná hodnota celkového očakávaného výnosu po n krokoch z *k*-tého rozhodnutia $v_i^{(n)}$ bude optimálna:

$$opt\ (v_i^{(n)}) = \max_{k \in D} \sum_{j=1}^{m} p_{ij}^k (r_{ij}^k + v_j^{(n-1)}) \tag{18}$$

pre $i = 1,2,\ldots,n;\ \ n = 1,2,3,\ldots.$

Ak označíme

$$q_i^k = \sum_{j=1}^{m} p_{ij}^k \, r_{ij}^k \tag{19}$$

potom predchádzajúci vzorec možno prepísať ako:

$$opt\ (v_i^{(n)}) = \max_{k \in D} \left[q_i^k + \sum_{j=1}^{m} p_{ij}^k \, v_j^{(n-1)}) \right] \tag{20}$$

Iteračný postup na základe rekurentného vzťahu (18) môžeme použiť na určenie postupnosti optimálnych rozhodnutí pre $i = 1,2, \dots, m$ a $n = 1,2,3, \dots$. Iteračný postup nám v podstate dáva návod ako rekurzívne od konca horizontu vypočítať optimálne riadenie maximalizujúce výnosy $v_i^{(n)}$, tzv. **Bellmanov princíp**. Použitie tohto postupu predstavuje aplikáciu dynamického programovania, ktorý pre úlohy takéhoto typu spracoval Howard v knihe Dynamic probabilistic systems v r. 1960 [1]. Tento princíp optimality ilustrujeme na nasledujúcom jednoduchom príklade.

„Iteračný postup založený na rekurentnom vzorci (18) vyplýva z hodnoty celkového ocenenia každého kroku n s cieľom dosiahnuť jeho maximum. Pričom sa posúva späť po horizonte až k základnému obdobiu. Preto sa táto metóda nazýva **metóda iterácie podľa hodnoty** (podľa účelovej funkcie), prípadne len rekurentná metóda.“[3]

V práci [1] Howard navrhol aj druhú metódu, nazvanú metóda iterácie podľa stratégie (metóda iterácií). Táto metóda predpokladá dlhodobé fungovanie systému, ktoré oprávňuje použiť na výpočet strednej hodnoty výnosu po *n* krokoch rozklad matice P na dve matice a to na stacionárnu maticu F, ktorá je konštantná a maticu T, predstavuje transientnú zložku procesu a nazýva sa diferenciálnou maticou.

Metóda iterácie podľa stratégie teda pozostáva z dvoch fáz:

1. ***Fáza určovania hodnôt***, pri ktorej sa určenie optimálneho riešenia procesu redukuje na riešenie sústavy lineárnych rovníc v rámci iteračného postupu.
2. ***Fáza postupu zlepšovania stratégie***, kde každé ďalšie riešenie, ktoré je výsledkom iteračného cyklu zabezpečuje vyššiu hodnotu výnosu ako predchádzajúce.

Iteračný postup sa končí spravidla po menšom počte iterácií s riešením, ktoré zabezpečí najvyššiu možnú hodnotu výnosu vzhľadom na úlohu. Metóda iterácie podľa stratégie je v podstate optimalizačný postup, zameraný na konštrukciu matice pravdepodobnosti prechodu pre Markovov reťazec.

[3] Unčovský, L., Čemická K.: *Stochastické procesy a modely*.1992. s.44

Úlohu výberu optimálnej matice pravdepodobnosti prechodu Markovovho reťazca možno formulovať ako úlohu lineárneho programovania. Tieto úlohy sa nazývajú aj úlohy programovania v Markovových reťazcoch, alebo úlohy Markovovho programovania.

Systém môže byť v m stavoch, s prechodom zo stavu i do j sú spojené náklady c_{ij}, pričom $c_{ij} = -r_{ij}$. Ak systém funguje dostatočne dlho, možno predpokladať existenciu limitného rozdelenia stavov. Úlohou je minimalizovať náklady priradené k jednotlivým prechodom nájdením optimálnej matice pravdepodobností prechodu. K tomu treba nájsť vektor pravdepodobnosti prechodu $(p_i^k = p_{i1}^k, p_{i2}^k, \dots, p_{im}^k)$ pre $i = 1,2, \dots, m$ a $k \in D$ tak aby stredná hodnota nákladov

$$\sum_{i=1}^{m}\sum_{j=1}^{m} p_i^* p_{ij}^k c_{ij} \tag{21}$$

bola minimálna. Potom $p_1^*, p_2^*, \dots, p_m^*$, je príslušné limitné rozdelenie. Treba nájsť minimum účelovej funkcie (21) za podmienok:

$$\sum_{i=1}^{m} p_i^* p_i^k = p^*$$

$$\sum_{i=1}^{m} p_i^* = 1$$

$$p_i^* \geq 0$$

1.4.4 Problém výrobcu cukroviniek

Výrobca cukroviniek má v pláne sledovať úspešnosť svojho výrobku, ktorý bude predávať ešte 5 období (kvartálov). Výrobok je následne v každom sledovanom období hodnotený podľa úspešnosti ako:

- 1. stav – úspešný,
- 2. stav – normálny,
- 3. stav – neúspešný.

Predpokladajme, že úspešnosť v danom kvartály závisí len od úspešnosti v minulom kvartály a celá dynamika problému môže byť opísaná Markovovým reťazcom. V každom období a stave má výrobca cukroviniek možnosť zvoliť si jednu z nasledujúcich akcií:

- 1. rozhodnutie – nevykonať žiadnu akciu,

- 2. rozhodnutie – investovať do marketingu,
- 3. rozhodnutie – zvýšiť ceny.

V nasledujúcich tabuľkách (Tabuľka 1-3) sú hodnoty pravdepodobností prechodov a ocenení prechodov pri voľbe jedného z daných rozhodnutí. Rozhodnutie investície do marketingu, predstavuje zvýšenie pravdepodobnosti lepšieho predaja výrobku, za cenu zvýšenia nákladov na výrobu. Zvýšením cien sa výnosy výrobcu zvýšia, avšak za cenu zvýšenia pravdepodobností prechodov do horších kategórií.

Tabuľka 1: 1. rozhodnutie - nevykonať žiadnu akciu

a) Pravdepodobnosti prechodov

$p_{ij}(1)$		j		
		1	2	3
i	1	0,35	0,55	0,10
	2	0,15	0,60	0,25
	3	0,10	0,40	0,50

b) Ocenenia prechodov

$r_{ij}(1)$		j		
		1	2	3
i	1	9	7	2
	2	6	2	-1
	3	-1	-4	-12

Zdroj: *Vlastné spracovanie*

Tabuľka 2: 2. rozhodnutie - investovať do marketingu

a) Pravdepodobnosti prechodov

$p_{ij}(2)$		j		
		1	2	3
i	1	0,45	0,50	0,05
	2	0,30	0,50	0,20
	3	0,15	0,60	0,03

b) Ocenenia prechodov

$r_{ij}(2)$		j		
		1	2	3
i	1	6	5	-1
	2	4	-0,25	-4
	3	-2	-6	-14

Zdroj: *Vlastné spracovanie*

Tabuľka 3: 3. rozhodnutie - zvýšiť cenu

a) Pravdepodobnosti prechodov

$p_{ij}(3)$		j		
		1	2	3
i	1	0,30	0,50	0,20
	2	0,10	0,50	0,40
	3	0,00	0,35	0,65

b) Ocenenia prechodov

$r_{ij}(3)$		j		
		1	2	3
i	1	12,15	9	2
	2	7	3	1
	3	0	-4	13

Zdroj: *Vlastné spracovanie*

Stredné hodnoty výnosov po prvom kvartály podľa vzťahu (19) sú vypočítané v nasledujúcej tabuľke (Tabuľka 4):

Tabuľka 4: Očakávané výnosy po 1. prechode

$q_{ij}(1)$		r		
		1	2	3
i	1	7,2	5,15	8,545
	2	1,85	0,275	2,6
	3	-7,7	-4,25	7,05

Zdroj: *Vlastné spracovanie*

Najskôr si ilustrujeme situáciu, ktorá by nastala ak by sme reťazec neriadili, to znamená, že by sme nevykonali žiadnu akciu (1. rozhodnutie). Podľa vzťahu (12) vypočítame očakávané výnosy po 5-tich kvartáloch (Tabuľka 5).

Tabuľka 5: Výnosy po 5 obdobiach

n	*0*	*1*	*2*	*3*	*4*	*5*
$v_1(n)$	0	7,20	9,97	10,84	11,06	11,05
$v_2(n)$	0	1,85	2,12	2,09	2,01	1,90
$v_3(n)$	0	-7,70	-10,09	-10,90	-11,23	-11,41

Zdroj: *Vlastné spracovanie*

V tabuľke 6 môžeme vidieť vypočítané hodnoty $v_i(n)$ a k nim prislúchajúce rozhodnutia $d_i(n)$ pre plán predaja 5 kvartálov. Rozhodnutia $d_i(n)$ predstavujú optimálne riadenie systému, ktoré maximalizuje výnosy pre $i=1,2,3$ a $n=1,2,...,6$.

Tabuľka 6: Optimálne výnosy a rozhodnutia

n	*0*	*1*	*2*	*3*	*4*	*5*
$v_1(n)$	0	8,55	10,55	11,54	12,42	13,31
$v_2(n)$	0	1,85	2,39	3,27	4,15	5,03
$v_3(n)$	0	-7,40	-6,86	-6,10	-5,23	-4,36
$d_1(n)$	-	3	3	3	1	1
$d_2(n)$	-	1	1	2	2	2
$d_3(n)$	-	2	2	2	2	2

Zdroj: *Vlastné spracovanie*

Z dosiahnutých výsledkov je zrejmé, že ak je výrobok úspešný, t. j. 1.stav, je vhodné najskôr nevykonať žiadnu akciu a až keď sa blíži k ukončeniu predaja, je výhodné zvýšiť ceny. Vidíme, že ak je v stave 2, t. j. výrobok sa zle predáva je vhodné vždy investovať do reklamy, v prípade neutrálnych výsledkov. Ku koncu predaja sa od tejto stratégie ustúpi a nevykonáva sa žiadna akcia. V tomto prípade sa prostriedky vynaložené na reklamu už zrejme nevrátia. Avšak, pre ktorýkoľvek východiskový stav dosiahneme lepší výsledok hospodárenia ako keby sme reťazec neriadili.

1.5 Úvod do MATLABu

MATLAB (Matrix Laboratory = Maticové laboratórium) je vysokovýkonný programový prostriedok na vedecko-technické výpočty, ktorý dnes nachádza uplatnenie v mnohých vedeckých odboroch. Užívateľovi poskytuje nástroje pre merania v reálnom čase, prácu s matematikou, grafikou, prenos dát a podobne. To všetko v graficky príjemnom prostredí. Všestrannosť programu je zaistená otvorenou architektúrou a veľkým množstvom knižníc, tzv. toolboxov, ktorými tento výpočtový systém disponuje. MATLAB v sebe integruje:

1. matematické výpočty,
2. vývoj algoritmov,
3. modelovanie a simulácie,
4. analýzu dát a ich vizualizáciu,
5. vedecké a inžinierske grafy,
6. vývoj aplikácií a programovanie.

MATLAB má vlastný programovací jazyk a umožňuje nielen kvalitné spracovanie dát, ale aj ich zobrazenie. Základom filozofie v MATLABe je práca s maticami, čo výrazne zjednodušuje prácu oproti iným programovací jazykom ako sú C, C++, Fortran alebo Java. Navyše je s týmito jazykmi plne kompatibilný.

Jednoduchá rozšíriteľnosť je najoceňovanejšou vlastnosťou MATLABu, ktorá umožňuje doplňovať systém o užívateľom napísané funkcie (M-súbory) i o celé aplikácie.

MATLAB sa skladá z týchto základných častí:

- **Výpočtové jadro**, pracuje najmä s operátormi pre matice reálnych a komplexných čísel. MATLAB ale umožňuje aj bežné matematické operácie ako násobenie, delenie, sčítanie a odčítanie. Okrem dátových typov jednoduchších ako tradičné

matice podporuje MATLAB taktiež typy zložitejšie, ako sú napr. viacrozmerné polia, polia buniek, v ktorých každý prvok môže byť iného typu.

- **Grafický subsystém**, ktorý ponúka široké spektrum možností prezentácie výsledkov. Je možné vykresľovať dvojrozmerné a trojrozmerné grafy, histogramy, koláčové grafy a ďalšie.
- **Otvorená architektúra**, umožňuje užívateľovi vytvárať vlastné funkcie, ktoré budú priamo zodpovedať požiadavkám nim vytváranej aplikácie.
- **Pracovné nástroje**, ako prehliadač adresárov a súborov, prehliadač pracovného priestoru, okno histórie príkazov, interaktívny spúšťač aplikácií, editor, debugger, profiler, hypertextová pomoc a príkazové okno sú do prostredia úplne integrované.
- **Toolboxy** sú špecializované knižnice MATLABu, ktoré zaisťujú jeho všestranné použitie v rôznych vedných a technických odboroch. Knižnice obsahujú dopredu spracované funkcie určené pre daný vedný odbor.

1.6 M-súbory

Súbory, ktoré obsahujú kód jazyka MATLAB nazývame **M-súbory**. Tento názov je odvodený z názvu súborov, ktoré musia mať koncovku ***.m**. Pri tvorbe M-súborov môžeme využiť akýkoľvek textový editor, ktorý je v systéme nainštalovaný alebo editor, ktorý má MATLAB v sebe zabudovaný.

Poznáme dva druhy M-súborov, ktoré sa odlišujú nasledovnými vlastnosťami:

1. ***M-súbor typu skript***[4]
   - nemá vstupné ani výstupné parametre,
   - používa dáta z pracovného priestoru MATLABu,
   - vhodný pre automatické spracovanie postupnosti príkazov ktoré sa opakujú viackrát.
2. ***M-súbor typu funkcia***
   - má jeden alebo viac vstupných parametrov a vracia jeden alebo viac výstupných parametrov,
   - vnútorné premenné sú lokálne vzhľadom na funkciu,
   - vhodný pre rozšírenie jazyka MATLAB pre konkrétne aplikácie, ktoré užívateľ vyvíja.

[4] Skript – pochádza z anglického script a predstavuje postupnosť príkazov, ktoré zapisujeme do prázdneho M-súboru tým istým spôsobom ako do príkazového riadka.

1.6.1 *Skript súbory*

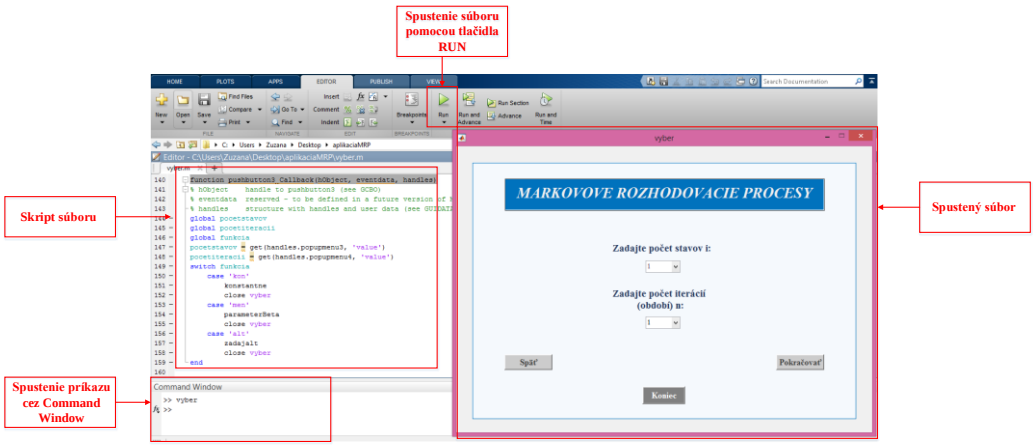
Z vyššie uvedeného vyplýva, že skript súbory sa využívajú na *zautomatizovanie vykonávania príkazov*, napríklad výpočtov uložených v súboroch, ktoré by sme z príkazového riadka museli vykonávať opakovane (napríklad pri cykloch)

„Skripty teda pracujú na základe existujúcich dát uložených v pracovnom priestore (*Workspace*) alebo môžeme v nich vytvárať nové dáta, s ktorými potom budeme pracovať ďalej. Akékoľvek premenné, ktoré v nich použijeme, môžeme využiť aj na ďalšie spracovanie.“

Pre skript súbory je charakteristické, že popri príkazoch si môžeme do nich súčasne zapisovať aj svoje vlastné **komentáre**, poznámky. Umožňuje nám to použitie znaku %. Znak % v súbore naznačuje, že akékoľvek výroky, príkazy, nasledujúce po tomto znaku v rovnakom riadku majú byť pri výpočtoch ignorované. Takto pridané komentáre nám pomáhajú pri ľahšej orientácii v súbore.

Spustenie skriptu dáva MATLABu príkaz na vykonanie v ňom zapísanej postupnosti príkazov. Uskutoční sa tak, že do príkazového riadku napíšeme len meno súboru, pod ktorým sme ho uložili a stačíme klávesnicu *Enter*. Môžeme ho taktiež spustiť priamo zo súboru stlačením príkazov *Run* na paneli nástrojov (Obrázok 2) alebo jednoducho pomocou klávesy F5.

Obrázok 2: Spustenie skript súboru



Zdroj: *Vlastné spracovanie*

1.6.2 *Vytváranie vlastných funkcií*

Druhý typ m-súborov slúži na *zadefinovanie funkcií*, môžeme si vytvárať nové funkcie, ktoré budú mať rovnaký status ako už existujúce funkcie. Riadok, v ktorom nové

funkcie definujeme, definičný riadok, dáva MATLABu informáciu, že funkcionálny súbor obsahuje funkciu a špecifikuje jej parametre.

Pri definovaní nových „matlabovských“ funkcií (Obrázok 3) je potrebné dodržať nasledujúce kroky:

1. Určiť ***meno funkcie***, ktoré musí byť identické s názvom súboru a uistiť sa, že v MATLABe ešte nebolo použité.

2. Prvý riadok novej funkcie musí byť v tvare:

$$\boldsymbol{function}\ [zoznam\ výstupných\ údajov] =$$
$$= \boldsymbol{meno\ funkcie}\ (zoznam\ vstupných\ údajov)$$

3. Stručne ***popísať funkciu***, jej účel a použitie. Pred týmito riadkami musí byť znak %, ktorý zaručí, že komentár nebude braný do úvahy, keď sa bude funkcia vyhodnocovať. Riadky komentára sa môžu objaviť kdekoľvek vo funkcionálnom súbore, t. j. môžeme ich dať aj na koniec riadka.
4. ***Napísať predpis, ktorý definuje novú funkciu***. Mal by byť dostatočne jasný, aby umožnil ďalšiemu používateľovi pochopiť jej použitie.
5. V jednom funkcionálnom súbore môže byť definovaná vždy ***len jedna funkcia***.

Obrázok 3: Definovanie vlastnej funkcie

priemer.m

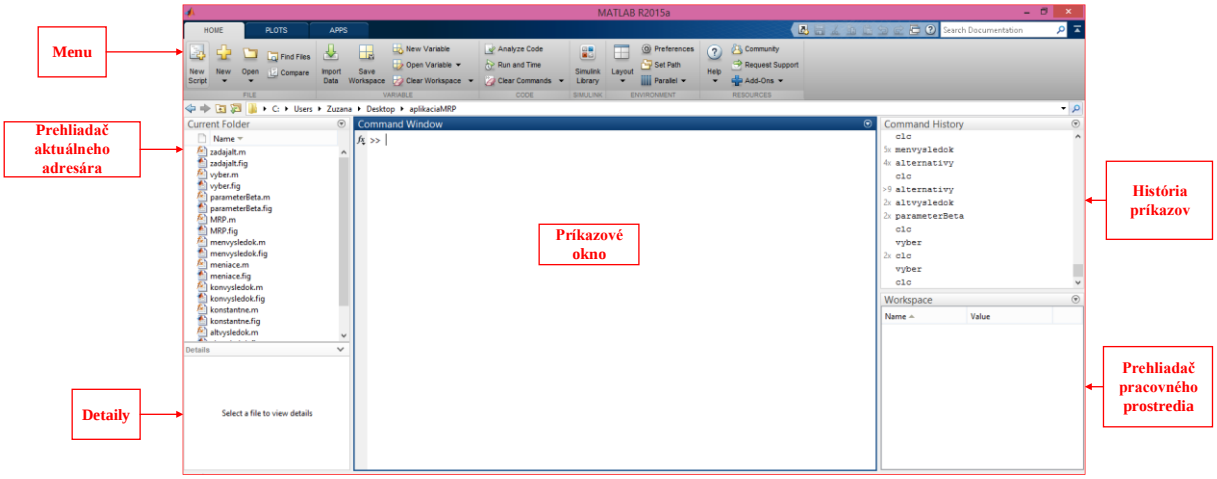
```
function y = priemer(x)
% PRIEMER stredna hodnota vektora.
% PRIEMER(X), kde X je vektor vrati strednu hodnotu elementov vektora
% Ak vstup nie je vektor, vrati chybu.
[m,n] = size(x);
if (~((m == 1) | (n == 1)) | (m == 1 & n == 1))
error('Vstup musi byt vektor')
end
y = sum(x)/length(x); % Vypocet
```

Zdroj: *Vlastné spracovanie*

1.7 Pracovné prostredie MATLABu

MATLAB je programovací jazyk a zároveň aj pracovné prostredie. Po spustení systému sa objaví hlavné okno zložené z niekoľkých častí. Pracovné prostredie (Obrázok 4) obsahuje jednotlivé okná ako napríklad okno pre správu premenných, okno pre správu pracovného priestoru *Workspace,* okno s históriou príkazov. Najdôležitejšie z nich je okno príkazov *Command Window* s promptom >>, jeho zobrazenie znamená, že je pripravený prijať príkazy. Usporiadania okien môžeme zmeniť, resp. môžeme niektoré okná zavrieť.

*Obrázok 4:*Pracovné prostredie MATLABu



Zdroj: *Vlastné spracovanie*

Pracovné prostredie MATLABu sa skladá z nasledujúcich nástrojov:

1. **Command Window (Príkazové okno)**

   Programátor komunikuje s MATLABom pomocou Príkazového okna. Príkazový riadok sa hlási cez prompt >>, signalizuje, že MATLAB je pripravený prijímať príkazy od programátora. Ak napíšeme a odošleme príkaz cez *Enter*, tak je hneď vykonaný. Zakázanie výpisu odpovede na obrazovku je možné dosiahnuť zápisom bodkočiarky na koniec výrazu v príkazovom riadku. Na vymazanie príkazového okna môžeme použiť príkaz *clc* ale aj menu cez *Home* → *Clear Commands* → *Command Window*.
2. **Command History (História príkazov)** –zaznamenáva postupnosť príkazov, ktoré sme zadali v príkazovom okne.
3. **Current Folder (Prehliadač aktuálneho adresára)** – v tomto okne sa zobrazia súbory aktuálneho adresára, pri stlačení pravého tlačidla myši sa objaví ponuka pre prácu so súbormi. Kontextové menu súboru umožňuje štandardnú prácu so súbormi pracovného adresára (premenovanie, kópia, presun, zmazanie) a navyše tiež otvorenie M-súboru.
4. **Workspace (Prehliadač pracovného priestoru)**

   Pracovný priestor tvoria premenné, ktoré sme vygenerovali počas práce v MATLABe. Zobrazuje všetky dostupné premenné pracovného prostredia a umožňuje s nimi pracovať:

- vytvorenie novej premennej,
- zobrazenie hodnoty premennej (dvojklikom),
- načítanie *.mat súboru,
- uloženie všetkých premenných do *.mat súboru na disk alebo následne vytvoriť graf,
- vymazanie premennej,
- editovanie premennej.

1.8 Riadenie toku programu

Vo svojej základnej forme aj v MATLABe fungujú podmienené príkazy rovnako ako vo väčšine počítačových jazykov. Aj najjednoduchší z algoritmov na testovanie nejakej podmienky si vyžaduje podmienené uskutočnenie programových blokov. Mnoho algoritmov tiež vyžaduje opakované zhodnotenie tých istých formúl alebo aplikáciu rovnakých formúl na veľký počet dát.

MATLAB ponúka osem príkazov na riadenie toku programu:

- ***if – else – elseif*** vyhodnotí skupinu výrazov na základe logickej podmienky,
- ***switch – case – otherwise*** vyhodnotí rôzne skupiny výrazov v závislosti na hodnote výrazu v podmienke,
- ***while*** vyhodnocuje opakovane skupinu výrazov pokiaľ je splnená logická podmienka,
- ***for*** vykoná dopredu stanovený počet opakovaní vyhodnocovania skupiny výrazov,
- ***continue*** preskočí vykonávanie prebiehajúcej iterácie slučky *for/while* a pokračuje nasledujúcou iteráciou,
- ***break*** ukončuje vykonávanie slučky *for/while*,
- ***try – catch*** mení tok programu, ak sa počas vyhodnocovania vyskytne chyba,
- ***return*** spôsobí odovzdanie riadenia volajúcej funkcii.

Tabuľka 7: Príkazy na riadenie toku programu jazyka MATLAB

Formát zápisu	**Sémantika**
if <podmienka> príkazy ... [elseif <iná podmienka> príkazy ...] [else príkazy] end	Ak je splnená logická podmienka, vykonajú sa príkazy v tomto bloku. *Podmienka* môže byť v MATLABe ľubovoľný výraz, ktorý nadobúda nulovú alebo nenulovú hodnotu. Nulová hodnota znamená, že podmienka nebola splnená. V prípade, že výsledkom výrazu v podmienke je pole, je podmienka nepravdivá len ak sú všetky prvky tohto poľa nulové. Ak je vetva *if* nepravdivá a existuje vetva *elseif* alebo *else*, pokračuje sa vo vyhodnocovaní tejto vetvy. Vetvy *elseif* môžu byť vnorené v ľubovoľnom počte, vetva *else* môže byť len jedna.
switch <výraz> case <h> príkazy ... [otherwise príkazy ...] end	Vykonávaný blok príkazov sa volí podľa hodnoty výrazu. *Výraz* sa najprv vyhodnotí a potom sa vykoná ten blok príkazov, ktorý má rovnakú hodnotu v príkaze *case*. V prípade, že hodnota výrazu nie je obsiahnutá v žiadnej vetve *case*, tak sa vykonajú príkazy vetvy *otherwise* ak táto existuje.
for <prem>=<zoznam> príkazy ... end	Premenná *prem* bude postupne nadobúdať hodnoty zo zoznamu hodnôt. Telo cyklu sa bude opakovať kým sa bude dať zo zoznamu priradiť premennej ďalšia hodnota. Ak je *zoznam* matica, bude premenná postupne nadobúdať hodnoty jednotlivých stĺpcov. Hodnota premennej je prístupná vnútri cyklu, nemala by sa však vnútri cyklu meniť. Počet opakovaní cyklu je dopredu daný počtom hodnôt v zozname.
while <podmienka> príkazy ... end	Na rozdiel od cyklu for, tu nie je stanovený počet opakovaní. Ak je *podmienka* splnená, vykoná sa ďalšia iterácia cyklu.

Zdroj: *Vlastné spracovanie*

2 Cieľ práce

Hlavným cieľom diplomovej práce je implementácia Markovových rozhodovacích procesov v programovej aplikácii pomocou zvoleného programu. Prítomnosť výpočtovej techniky je v súčasnosti neodmysliteľnou súčasťou pri riešení rôznych úloh alebo riešení nejakej problematiky. Cieľom je vytvoriť aplikáciu, ktorá bude riešiť úlohy MRP a v budúcnosti ponúkne možnosť využitia na študijne účely pri riešení optimalizačných úloh. Práca prezentuje jednoduchý spôsob vytvorenia aplikácie za pomoci nástroja GUI integrovaného v programe MATLAB. Účelom je navrhnúť používateľsky jednoduchú a atraktívnu aplikáciu, ktorá bude sprístupnená aj užívateľom, ktorý nemajú nainštalovaný systém MATLAB. Dosiahnutiu hlavného cieľa predchádzalo splnenie niekoľkých čiastkových cieľov, ktoré boli vymedzené pred samotnou implementáciou aplikácie.

Hlavný cieľ: Vytvorenie programovej aplikácie na základe modelov MRP.

Čiastkové ciele:

1. čiastkový cieľ: štúdium problematiky MRP
2. čiastkový cieľ: výber vhodného prostredia pre implementáciu aplikácie
    - prehľad dostupných riešení,
    - výber s pomedzi jednotlivých programových prostriedkov,
3. čiastkový cieľ: návrh aplikácie
    - rozmyslieť si, ako bude aplikácia vyzerať,
    - aké ovládacie prvky použijeme,
    - návrh grafického rozhrania,
    - realizácia návrhu v MATLAB nástroji GUI,
4. čiastkový cieľ: implementácia aplikácie
    - úprava zdrojového kódu,
    - naprogramovanie jednotlivých ovládacích prvkov pomocou Callback funkcií,
    - otestovanie aplikácie,
    - vytvorenie *.exe súboru.

3 Metodika práce a metódy skúmania

3.1 Nástroj GUIDE

V moderných operačných systémoch sú takmer všetky aplikácie vytvárané ako grafické užívateľské rozhranie. Sú to teda aplikácie, ktoré obsahujú grafické ovládacie prvky a sprostredkovávajú väzbu medzi užívateľom a programom. Tieto aplikácie sa často označujú skratkou GUI z anglického (Graphical User Interface).

Pri tvorbe GUI v MATLABe je možné použiť editor GUIDE (Graphical User Interface Development Environment). GUIDE je prostredie pre vytváranie grafických aplikácií pomocou grafických objektov, ktoré poskytuje GUI.

GUIDE umožňuje jednoduché rozmiestnenie grafických prvkov, ktoré doplní automaticky generovaným zdrojovým kódom. Riešenie je teda časovo menej náročné. Nevýhodou však je, že automaticky generovaný zdrojový kód nie je optimálny. Na rozdiel od aplikácie, ktorú si užívateľ vytvorí sám, obsahuje nepotrebné časti zdrojového kódu.

3.1.1 Spustenie GUIDE

Editor je možné spustiť zápisom príkazu *guide* v okne *Command Window* alebo pomocou *File →New → Graphical User Interface*. Po jeho spustení sa zobrazí okno *GUIDE Quick Start* (Príloha A). GUIDE ponúka štyri možnosti vytvorenia grafického užívateľského rozhrania, ako aj otvorenie už existujúceho GUI (*Open Existing GUI).*

Užívateľ má na výber:

- Prázdnu aplikáciu (*Blank GUI),*
- Aplikáciu s ovládacími prvkami Uicontrol *(GUI with Uicontrols),*
- Aplikáciu s objektom Axes a menu *(GUI with Axes and Menu),*
- Vytvorenie dotazu pre chod aplikácie *(Modal Question Dialog).*

Pre naše potreby pri tvorbe aplikácie sme zvolili prázdnu aplikáciu, teda možnosť *Blank GUI.* Okrem typu aplikácie si v tomto okne taktiež môžeme zvoliť miesto uloženia aplikácie *Save new figure as*. Po tejto voľbe sa spustí prostredie pre vývoj GUI. Vzhľad vytvoreného GUI ukladá GUIDE do súboru s príponou **.fig* a jeho zdrojový kód do súboru **.m*.

Vzhľad sprievodcu (Príloha B) možno rozdeliť do troch skupín. Na ľavej strane sa nachádza tzv. *Component Pallet,* na ktorej sú objekty *Uicontrol*. Pracovnú plochu, tvorí štvorcová sieť – *Layout Area*, ktorá umožňuje približne určiť pozíciu jednotlivých objektov, ktoré na pracovnú plochu umiestnime. Layout Area ponúka intuitívnu prácu pri návrhu

užívateľskej obrazovky jednoduchým pridávaním komponentov na pracovnú plochu, čo dáva celkom jasnú predstavu o podobe výslednej aplikácie. Menu GUIDE editoru možno rozdeliť do dvoch skupín štandardné (*File, Edit, View,...*) a špecifické (*New Figure, Open Figure, Run Figure,...*).

3.1.2 Nastavenie GUIDE

Ak chceme upraviť vzhľad pracovnej plochy (Layout Area), na ktorú umiestňujeme jednotlivé objekty, zmeny vykonáme v menu *Tools → Grid and Rulers* (Obrázok 3). Tu je vhodné zaškrtnúť položku *Show rulers*, ktorá slúži k zobrazeniu pravítka. Ďalej je dobré zmeniť veľkosť štvorcovej siete (*Grid size(in pixels)).* Prednastavená veľkosť štvorca je 50 pixelov(Príloha C).

V menu *File → Preferences* si môžeme prispôsobiť prostredie GUIDE vlastným potrebám. Samostatná vetva *GUIDE* sa nachádza v stromovej štruktúre preferencií a slúži na zjednodušenie práce v prostredí a to vo forme rozšíreného zobrazenia ovládacích prvkov. Pri zaškrtnutí položky *Show names in component palette* sa zobrazia názvy objektov v *Component palette* (Príloha D).

3.1.3 Component palette

Po upravení prostredia vlastným požiadavkám môžeme pristúpiť k návrhu vlastnej aplikácie. Pridávanie objektov na pracovnú plochu je možné vykonať jednoduchým kliknutím na lištu objektov a ťahom umiestniť objekt na požadované miesto, tzv. Drag&Drop. Objekty GUI plnia rôzne funkcie, majú odlišné vlastnosti a používajú sa na rôzne udalosti. Hlavnú skupinu tvoria komponenty slúžiace pre vkladanie údajov samotným užívateľom. Do tejto skupiny patria:

1. **Push Button (tlačidlo)** – tento prvok využívame najčastejšie pre zahájenie nejakej akcie. Môžeme ho využiť napríklad pre štart aplikácie alebo vykonanie výpočtu, nakreslenie grafu a pod.
2. **Slider (posuvník)** – môže slúžiť napríklad ku zmene hodnôt v určitom rozsahu. Programátor nadefinuje maximálnu a minimálnu hodnotu, čím určí rozsah, v ktorom bude veličina menená. Ďalej tu môžeme zaistiť, aby veličina nadobúdala iba celočíselné hodnoty.
3. **Radio Button (zaškrtávacie pole)** – toto pole môže slúžiť napríklad k výberu medzi akciami, ktoré bude aplikácia vykonávať.

4. **Check Box (zaškrtávacie pole)** – má obdobnú funkciu ako *Radio Button*. Môžeme ho využiť pre zapnutie či vypnutie mriežky grafu, viditeľnosť či neviditeľnosť objektu alebo nastavenie ďalších podmienok aplikácie, kde sa rozhodujeme medzi dvomi akciami.
5. **Edit Text (prepisovateľný text)** – tento komponent môžeme použiť pre zadávanie vstupných hodnôt užívateľom. Napríklad ak budeme chcieť vykonať nejaký výpočet, kde si užívateľ sám zadá vstupné hodnoty, využijeme k tomu práve *Edit Text.*
6. **Static Text (popisy)** – slúži najčastejšie k popisu ďalších prvkov. *Static Text* používame najčastejšie k lepšej prehľadnosti aplikácie a zaisteniu lepšej orientácie pre užívateľa.
7. **Pop-up Menu (ponuka)** – používame, ak chceme vyberať z určitej skupiny prvkov podobného druhu. Podmienkou je, že užívateľ môže vyberať iba z prvkov, ktoré sú v menu k dispozícii.
8. **Listbox (ponuka)** – má rovnakú funkciu ako *Pop-up Menu*. Rozdiel je iba v spôsobe zobrazenia dát. Pri *Pop-up Menu* vyberáme prvky po kliknutí na šípku, ktorá zobrazí ich ponuku, pričom medzi prvkami v *Listboxe* listujeme.
9. **Toggle Button (prepínač)** – je tlačidlo, ktoré slúži k prepínaniu medzi hodnotami. Môžeme ho použiť napríklad k prepínaniu medzi stupňami a radiánmi.
10. **Table (tabuľka)** – tento komponent slúži k zadávaniu väčšieho množstva dát. Napríklad pri práci s maticami či štatistickými dátami.
11. **Axes (súradnicový systém)** – slúži k vymedzeniu priestoru, v ktorom sa bude zobrazovať graf.
12. **Panel** – komponent, ktorý slúži najmä k zoskupovaniu objektov napríklad ovládacích prvkov. Slúži k zaisteniu lepšej prehľadnosti aplikácie.
13. **Button Group (skupina komponentov)** – má podobnú funkciu ako komponent *Panel*. Je špecifický svojim charakterom, pretože sa doň vkladajú komponenty, ktorých akcie sa navzájom ovplyvňujú a súvisia spolu (skupina checkboxov, alebo toggle buttonov).
14. **ActiveX Control**

3.1.4 Menu editora GUIDE

Editor GUIDE (Príloha E) obsahuje niekoľko ikon, ktoré sú typické iba pre tento editor. V nasledujúcom texte preberieme a vysvetlíme niektoré z týchto funkcií funkcie.

Pre grafické spracovanie je veľmi užitočná funkcia ***Align Objects***  (Príloha F), ktorá umožňuje nastavovať ako vertikálne tak horizontálne zarovnanie jednotlivých prvkov. Dovoľuje nastaviť zarovnanie a odsadenie medzi objektmi navzájom.

Pri vytváraní aplikácií skladáme jednotlivé objekty na pracovnú plochu GUIDE. Ak chceme zobraziť zoznam všetkých použitých objektov a ich hierarchiu, môžeme použiť *Object Browser*  (Príloha F).

Vlastnosti jednotlivých objektov aplikácie ako aj údaje o všetkých objektoch sú zahrnuté v tzv. *Property Inspectore* (Príloha G), ktorý spustíme buď dvojklikom na vybraný objekt alebo kliknutím na ikonu  . V okne, ktoré sa následne zobrazí, sú všetky vlastnosti objektu. Každému objektu sa dajú nastavovať parametre, ako názov objektu, jeho typ, rozmery, farbu a pod.

Posledná ikona *Run Figure*  slúži ku spusteniu nami vytvorenej aplikácie. Ak sme v okne *GUIDE Quick Start* nezaškrtli voľbu *Save on start up as*, MATLAB nás teraz vyzve k uloženiu našej práce. Aplikácia je uložená v dvoch súboroch s rovnakým názvom s príponou *.fig a *.m.

***.fig** – tu je uložená grafická podoba aplikácie. Obsahuje kompletný popis grafickej časti GUI a jej súčastí ako napríklad tlačidlá, menu, panely a pod. V tomto momente končí fáza grafického návrhu projektu a začína programátorská fáza.

***.m** – je súbor, ktorý sa vygeneruje automaticky pri prvom spustení. V súbore sú časti procedúr a akcií, ktoré sa vytvárajú automaticky v závislosti na akciách, ktoré sme vykonali v GUIDE editore. Vznikajú tu aj prázdne Callback funkcie[5] určené pre obsluhu jednotlivých objektov. Tu si zdrojový kód doplníme sami tak, aby aplikácia robila to, čo od nej požadujeme.

3.2 Postup pri tvorbe GUI aplikácie

Ešte predtým, ako sa pustíme do vlastného návrhu aplikácie pomocou GUIDE editoru, je potrebné si všetko dobre premyslieť. Predovšetkým pri zložitejších aplikáciách nie je vhodné začať ihneď navrhovať bez hlbšieho preštudovania problému. Postup pri navrhovaní GUI aplikácie si môžeme rozdeliť do niekoľkých etáp.

[5] Callback je funkcia, ktorá sa spustí, ak užívateľ vykoná určitú akciu, napr. otvorí zatvorí model, spustí simuláciu, vytvorí tabuľku, ...

1. Rozmyslieť si, ako bude aplikácia vyzerať, ktoré veličiny bude treba ovládať a aké ovládacie prvky k tomu použijeme. Ako budú jednotlivé ovládacie prvky rozmiestnené, aby aplikácia bola užívateľsky prívetivá a prehľadná. Vhodný je aj náčrt rozloženia prvkov.
2. Do *Layout Area* umiestnime jednotlivé objekty, upravíme ich veľkosť, vzhľad a zarovnanie. Ovládacie prvky je vhodné zoskupovať do blokov a priradiť im rovnaké rozmery. Pokiaľ teda napríklad použijeme niekoľko tlačidiel je vhodné im nastaviť rovnakú výšku a šírku.
3. Akonáhle máme pripravený návrh, uložíme aplikáciu, teda vygenerujeme dva súbory s príponou *.fig a *.m a spustíme ju. Tu môžeme vidieť ako aplikácia v skutočnosti vyzerá a pokiaľ sme s jej vzhľadom spokojní, môžeme sa pustiť do úpravy zdrojového kódu.
4. Otvoríme si súbor s príponou *.m, do ktorého doplníme príslušné Callback funkcie.
5. Vytvorenú aplikáciu otestujeme pomocou Debugger a ak je všetko v poriadku, teda pokiaľ aplikácia robí to, čo sa od nej požaduje, sme hotoví.

3.3 Programovanie GUIDE

Push Button (tlačidlo):

Programovanie každého komponentu v GUI sa začína jeho vložením na pracovnú plochu (Príloha H). V Editore sa po uložení projektu automaticky generuje kód v .m súbore. M-súbor obsahuje štandardný kód, ktorý reprezentuje samotný projekt ale aj funkciu reprezentujúcu práve priradený komponent. Názov tejto funkcie je definovaný v Property Inspectore pod parametrom *Tag*, taktiež ho môžeme vidieť v ľavom dolnom rohu okna GUI. Tag je dôležitou vlastnosťou každého použitého objektu, preto treba tento názov voliť pozorne a venovať mu patričnú pozornosť. V našom prípade necháváme názvy prednastavené, to znamená, že v prípade objektu *Push Button* má vlastnosť Tag názov napríklad pushbutton1. Takisto môžeme v Property Inspectore zmeniť názov tlačidla, ktorý sa zobrazuje v aplikácii pri behu programu. Tentoraz pôjde o zmenu parametra *String*.

Na obrázku 5 môžete vidieť ukážku priradenia akcie pre zvolené tlačidlo jednoduchým zápisom príslušného príkazu do jeho funkcie resp. Callbacku. Daný kód po stlačení tlačidla zobrazí okno s názvom „MRP“ a zároveň sa zatvorí okno „vyber“.

Obrázok 5: Callback objektu Push Button s vlastným kódom

```
% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
MRP
close vyber
```

Zdroj: *Vlastné spracovanie*

Pop-up Menu

Pop-up Menu slúži k výberu z niekoľkých známych položiek. V tomto menu sa po rozkliknutí šípky objaví ponuka, z ktorej si užívateľ vyberie práve jednu z možností. Jednotlivé položky sú v menu definované pomocou vlastnosti *String* (Príloha I). Každej položke je potom priradený index. Hodnoty indexov sú uchovávané pomocou *value*, ktorá určuje zvolenú ponuku užívateľom. Prvá položka má teda hodnotu 1, ďalšia 2 atď.

Hodnoty indexov sú uchovávané pomocou *value*, ktorá určuje práve zvolenú hodnotu (Obrázok 6). Prvá položka má teda hodnotu 1, ďalšia 2 atď.

Obrázok 6: Načítanie hodnôt z Pop-up menu

```
% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global pocetstavov
global pocetiteracii
global funkcia
pocetstavov = get(handles.popupmenu3, 'value')
pocetiteracii = get(handles.popupmenu4, 'value')
switch funkcia
    case 'kon'
        konstantne
        close vyber
    case 'men'
        parameterBeta
        close vyber
end
```

Zdroj: *Vlastné spracovanie*

Edit text

Jedným z najdôležitejších komponentov v rámci GUI uicontrol je Edit text, ktorý slúži na priame zadávanie parametrov užívateľom. Údaje vložené do tohto objektu sa ukladajú do parametra *String*. Tieto údaje sú vkladané ako textové reťazce, aj v tom prípade ak sú

vložené čísla alebo špecifické znaky (napr. [], ; atď.). Ak chceme vykonať napríklad nejakú matematickú operáciu so zadanými údajmi od používateľa, je potrebné ich najskôr previesť na číselné znaky a až potom s nimi pracovať.

Obrázok 7: Načítanie hodnoty z poľa edit text

```
function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global beta
beta = get (handles.edit1, 'String')
```

Zdroj: *Vlastné spracovanie*

4 Výsledky práce a diskusia

Táto časť poskytuje určitý manuál k vytvoreniu aplikácie v MATLABe. Užívateľ vďaka nej dôjde k riešeniu úloh Markovových rozhodovacích procesov, ktoré by pri ručnom počítaní boli veľmi zdĺhavé. Grafický tvar riešenia je zhodný s tvarom výsledkov uvádzaných vo väčšine učebníc zaoberajúcich sa Markovovými procesmi.

4.1 Základné údaje o aplikácii

Aplikácia je vytvorená v prostredí MATLAB, pomocou nástroja GUIDE sú vytvorené všetky grafické úpravy a výpočty sú definované v zdrojovom kóde. Aplikácia je tvorená pomocou niekoľkých GUI okien, medzi ktorými sa užívateľ pohybuje pomocou tlačidiel. Medzi zadaním vstupných parametrov a záverečným výsledkom sú vykonávané kontroly vstupných údajov, vďaka čomu je užívateľ upozornený na prípadné nesprávne zadanie jednotlivých parametrov. Všetky výpočty sú realizované na základe zmienených vzorcov v prvej kapitole diplomovej práce.

Užívateľ môže vytvorenú aplikáciu použiť k riešeniu úloh:

1. Konštantného ocenenia prechodov medzi stavmi,
2. Meniaceho sa ocenenia prechodov medzi stavmi,
3. Rozhodovacieho procesu s alternatívami.

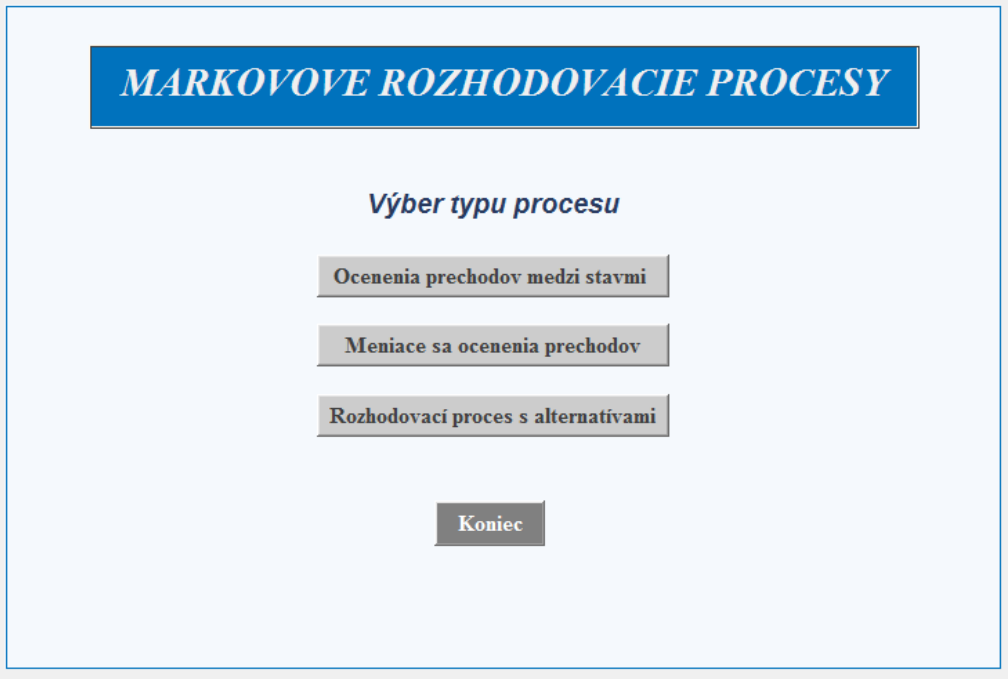
Pre riešenie tohto typu úloh sme vytvorili určité obmedzenia, avšak myslíme si, že pre študijné účely sú takto nastavené obmedzenia postačujúce. Je možné počítať úlohy až do počtu **30-tich stavov** v systému. Pre tieto stavy dokáže program vypočítať až **50 iterácií.** Počet iterácií bude v zobrazených výsledkoch číslovaný od čísla 1 nakoľko sa v matlabe nie je možné číslovať hlavičku od 0 iterácie (museli by sme hľadať riešenie v inom programovacom jazyku), preto bude vo výsledkoch o 1 iteráciu viac ako bolo zadané v okne pre zadávanie počtu iterácií. Pre rozhodovací proces s alternatívami môžeme pri každom stave vypočítať až **10 možností prechodu**. Nakoľko sú tieto výpočty vykonávané dynamicky, môžeme tieto obmedzenia v prípade nutnosti zväčšiť.

4.2 Popis aplikácie

Pri spustení aplikácie sú pomocou nástroja GUIDE vytvorené základné vizuálne úpravy, tak aby prostredie samotného programu bolo prívetivé pre užívateľa a boli zakázané prípadné nechcené úpravy aplikácie.

V úvodnom okne (Obrázok 8) si užívateľ vyberá typ, úlohy ktorá sa má riešiť. Voľba je vykonaná stlačením tlačidla s názvom úlohy. Tlačidlo **Koniec** umožňuje bezpečné ukončenie aplikácie.

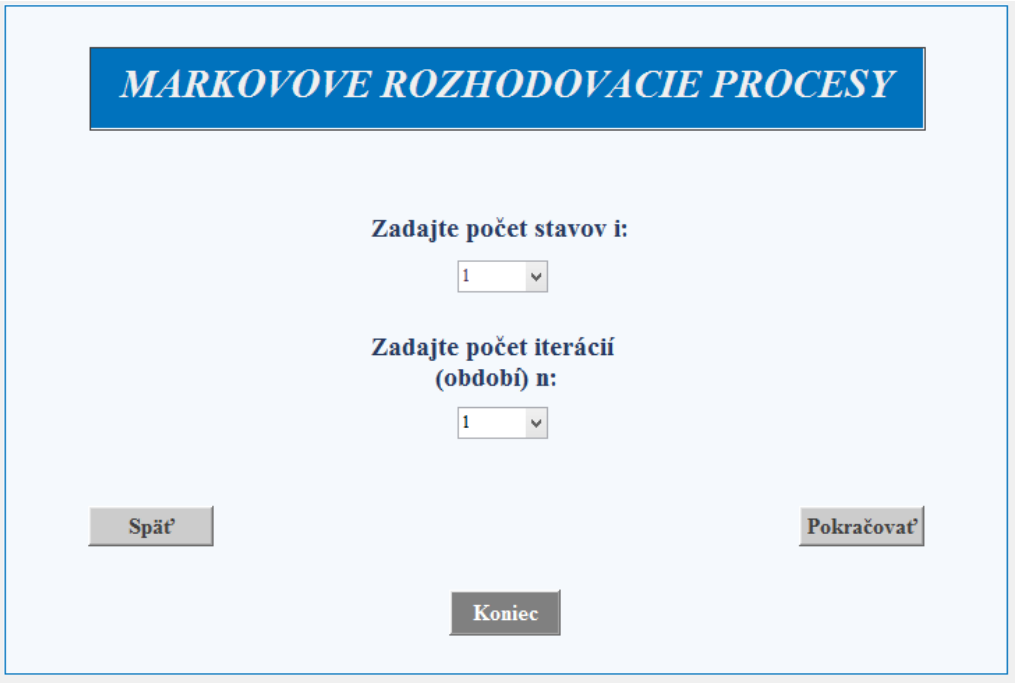
Obrázok 8: Úvodné okno aplikácie



Zdroj: *Vlastné spracovanie*

Po kliknutí, na ktorýkoľvek typ úlohy sa zobrazí okno, v ktorom je užívateľ povinný zadať počet stavov v systéme a počet iterácií výpočtu (Obrázok 9). Oba tieto parametre sú základom pre výpočet všetkých troch typov úloh, ktoré sa dajú riešiť pomocou tohto programu. Údaje sú vyberané z poľa zo zoznamom a vďaka tomu je nemožné zadať nesprávnu hodnotu parametra.

Obrázok 9: Zadávanie počtu stavov a iterácií



Zdroj: *Vlastné spracovanie*

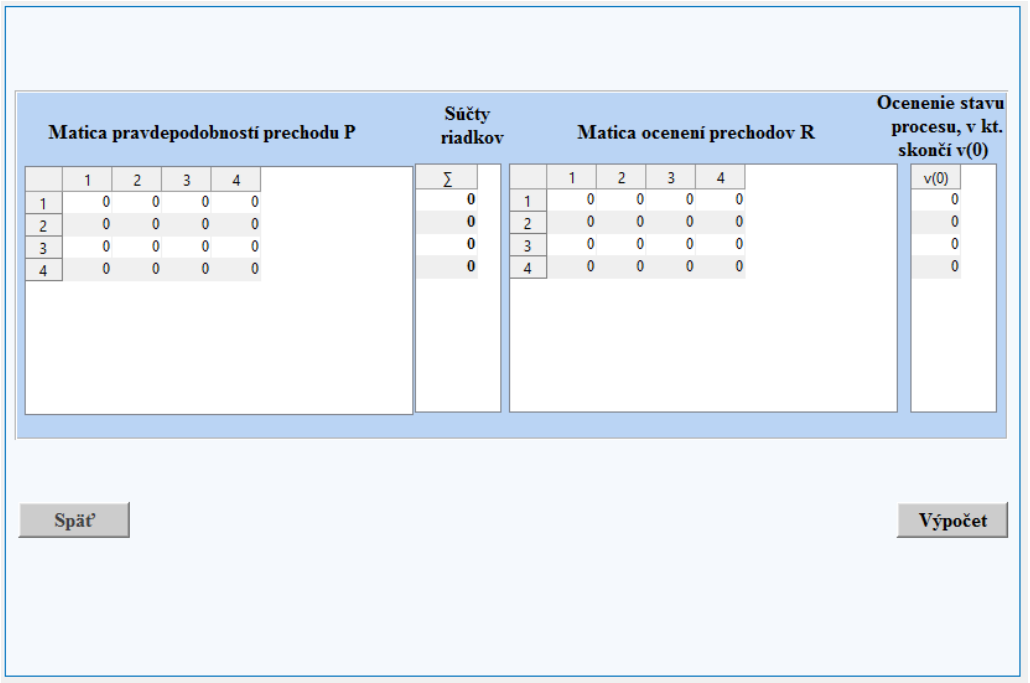
Pod týmito poľami zoznamov sú v spodnej časti okna umiestnené tri tlačidlá. Tlačidlo **Späť**, vráti užívateľa na úvodné okno, tlačidlo **Pokračovať** prepne na ďalší list aplikácie a tlačidlo **Koniec**, pomocou ktorého môžeme bezpečne ukončiť aplikáciu bez nechcených zmien.

V tomto bode, sa aplikácia rozdeľuje do jednotlivých okien k zadaniu konkrétnych údajov pre zvolený typ úlohy. Popíšeme si ich teda postupne, tak ako sú zoradené na úvodnom liste.

4.2.1 Ocenenia prechodov medzi stavmi

Okrem počtu stavov v systéme a počtu iterácií výpočtu, ktoré sú zadávané na druhom okne aplikácie, je pre tento typ úloh potrebné zadať **maticu pravdepodobností prechodov P, maticu ocenenia prechodov R** a **ocenenie stavu procesu v ktorom skončí v(0).** Tieto hodnoty sú zadávané do vopred definovaných matíc, resp. vektorov, ako môžeme vidieť na obrázku 20. Pre zobrazenie výsledku užívateľ po zadaní hodnôt zvolí tlačidlo **Výpočet.** Pokiaľ by sa chcel užívateľ v priebehu zadávania kedykoľvek zmeniť počet stavov alebo iterácií, je mu to umožnené pomocou tlačidla **Späť**.

Obrázok 10: Zadávanie hodnôt do matíc

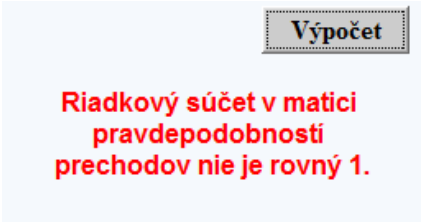


Zdroj: *Vlastné spracovanie*

Nakoľko každý riadok matice pravdepodobností prechodov predstavuje úplnú pravdepodobnosť, musí byť jej súčet rovný 1 a samozrejme hodnoty musia byť v intervale <0;1>. Pre kontrolu sú pre každý riadok matice pravdepodobností prechodu vpravo umiestnené jej riadkové súčty. Užívateľ tak vidí, či sú súčty zadaných hodnôt naozaj rovné 1.

Pokiaľ tomu tak nie je a užívateľ aj napriek tomu zvolí tlačidlo vypočítaj zobrazí sa mu varovanie, že niektorý z riadkových súčtov nie je rovný 1 (Obrázok 11), a aplikácia mu neumožní výpočet.

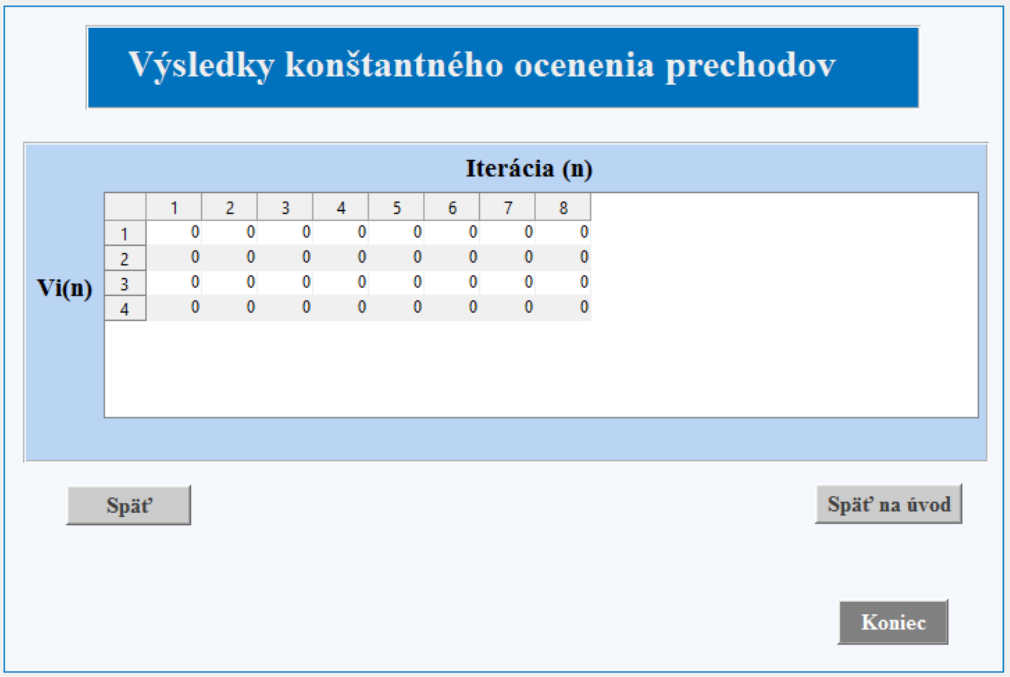
*Obrázok 11:*Upozornenie o nesprávnom riadkovom súčte v matici pravdepodobností prechodov



Zdroj: *Vlastné spracovanie*

Výsledky úlohy sú užívateľovi zobrazené (Obrázok 12) po kliknutí na tlačidlo vypočítaj, len v prípade, keď sú všetky zadané hodnoty správne. Z okna výsledkov tejto úlohy sa môže užívateľ vrátiť o krok späť pomocou tlačidla **Späť** alebo na úvodné okno aplikácie pomocou tlačidla **Späť na úvod.** Tlačidlo **Koniec** slúži na ukončenie aplikácie. Výraz $v_i^{(n)}$ označuje strednú hodnotu celkového očakávaného výnosu procesu, ktorý je na začiatku sledovania v i-tom stave a uskutoční n prechodov.

Obrázok 12: Zobrazenie výsledkov konštantného ocenenia prechodov



Zdroj: *Vlastné spracovanie*

4.2.2 Meniace sa ocenenia prechodov

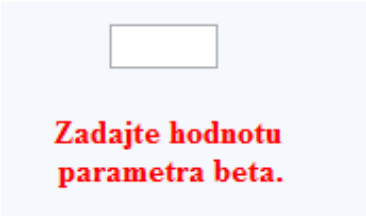
Druhé tlačidlo na úvodnom okne slúži ako názov napovedá, k výpočtu úloh meniacich s ocenení prechodov. Tento typ úloh, ako už bolo spomínané v teoretickej časti, sa líšia od predchádzajúcej úlohy zadaním parametra β. Jeho hodnota sa zadáva (Obrázok 13) ešte pred zadaním matice pravdepodobností prechodov, matice ocenení a ocenení stavu, v ktorom proces skončí. Hodnota parametra β musí byť z intervalu <0;1>, pre hodnotu 1 ide vlastne o úlohu konštantného ocenenia prechodov.

Obrázok 13: Zadanie hodnoty parametra β

Zdroj: *Vlastné spracovanie*

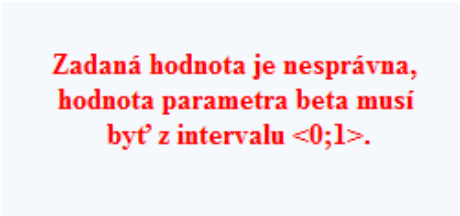
Tlačidlo **Späť**, slúži k návratu na zadávanie počtu stavov a iterácií výpočtu. Pre ďalšie pokračovanie užívateľ klikne na tlačidlo **Pokračovať**, zároveň je tu vykonaná kontrola parametra β, ak nie je vyplnené pole pre zadanie parametra β zobrazí sa upozornenie(Obrázok 14), takisto ak je zadaná chybná hodnota je zabránené ďalšiemu pokračovaniu a je zobrazené varovanie (Obrázok 15) .

Obrázok 14: Nezadaná hodnota parametra β



Zdroj: *Vlastné spracovanie*

Obrázok 15: Zadaná nesprávna hodnota parametra β



Zdroj: *Vlastné spracovanie*

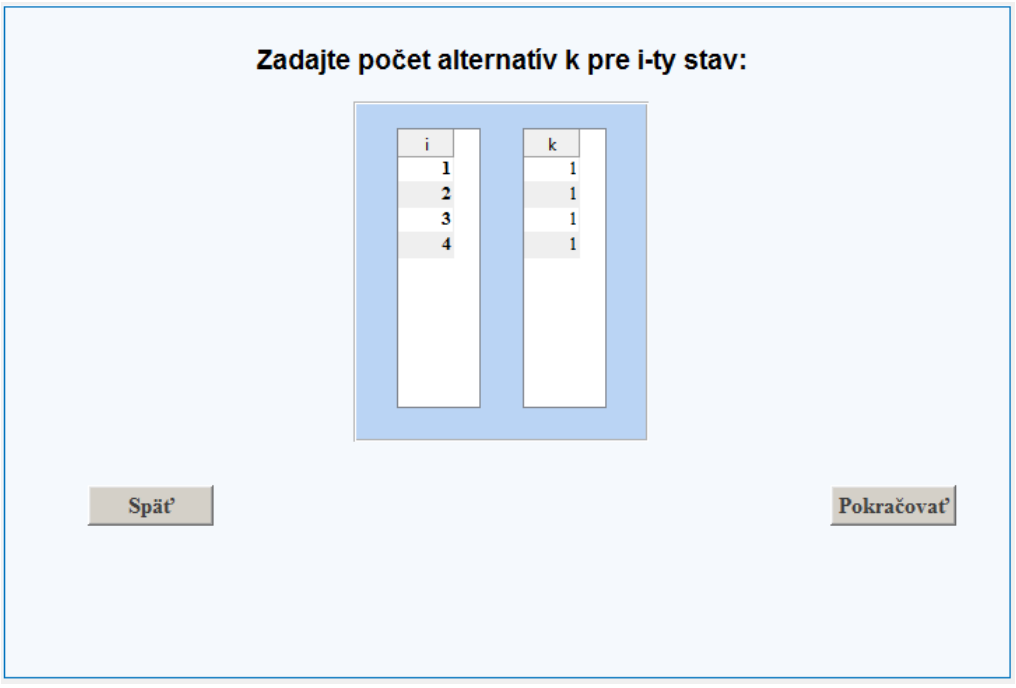
Pre zadanie hodnôt matice pravdepodobností prechodov, ocenení prechodov a ocenenie stavu procesu v ktorom skončí je použité okno zhodné s oknom použitým pri úlohe konštantného ocenenia prechodov popísané vyššie spolu so všetkými funkciami. Takisto aj

okno s výsledkami úlohy je zhodné s predchádzajúcim. Rozdiel oproti predošlej úlohe je iba v spôsobe výpočtu, kde sa hodnoty prenásobia parametrom β.

4.2.3 Rozhodovací proces s alternatívami

Tretím typom úloh, ktoré aplikácia dokáže počítať sú rozhodovacie procesy s alternatívami. Po kliknutí na ich názov v úvodnom okne a následnom zadaní počtu stavov systému a počtu iterácií, sa užívateľovi zobrazí tabuľka (Obrázok 16) pre zadanie počtu alternatív. V prvom stĺpci tabuľky sú čísla stavov i, ktorých počet závisí od počtu stavov zadaných v predchádzajúcom okne, tieto čísla sú automaticky generované aplikáciou. Do druhého stĺpca, užívateľ zadá počet alternatív pri jednotlivých stavoch procesu. Pre prehľadnosť je ich maximálny počet nastavený na 10. Prednastavená hodnota je 1, čo je taktiež minimálna hodnota, ktorá môže byť zadaná.

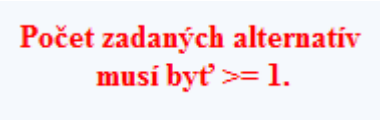
Obrázok 16: Zadanie počtu alternatív



Zdroj: *Vlastné spracovanie*

Aj tu je vykonaná kontrola zadaných hodnôt, po stlačení **Pokračovať**, ak nie sú všetky alternatívy zadané správne, nie je možné pokračovať v úlohe a je zobrazené varovanie (Obrázok 17).

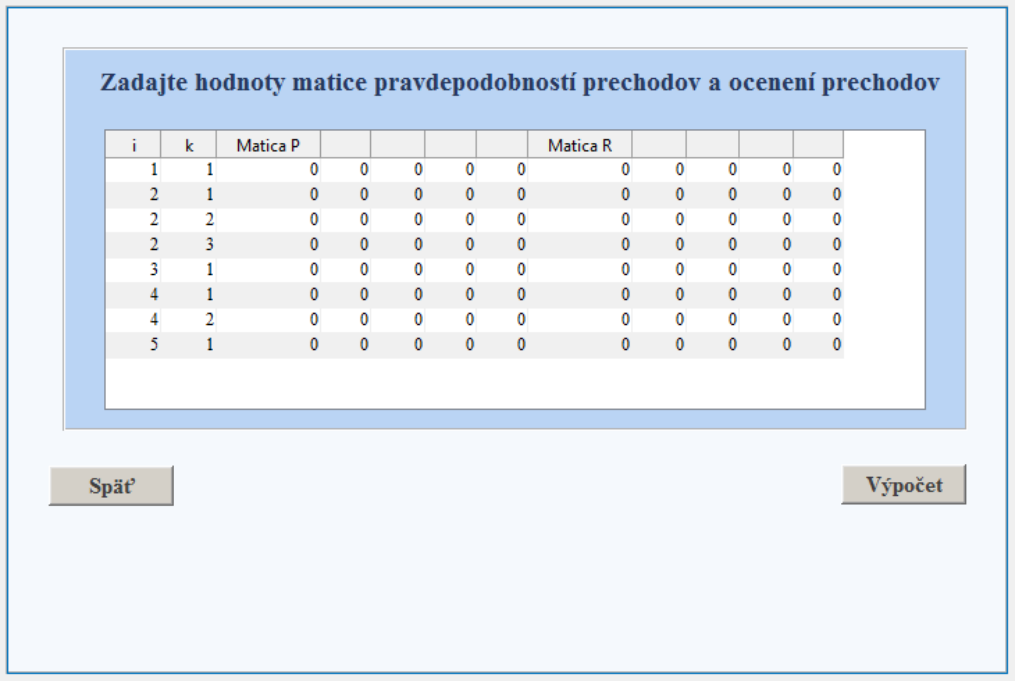
Obrázok 17: Nesprávne zadaný počet alternatív



Zdroj: *Vlastné spracovanie*

Tlačidlo **Späť**, vráti používateľa k možnosti zmeny počtu stavov a iterácií. Kliknutím na tlačidlo pokračovať sa užívateľ dostáva k oknu, kde zadáva pre jednotlivé stavy a ich príslušné alternatívy matice pravdepodobností prechodov a matice ocenení. V tomto okne (Obrázok) sa na rozdiel od predchádzajúcich úloh nenachádza stĺpec, v ktorom by boli riadkové súčty matíc pravdepodobností prechodov. Je tomu tak z dôvodu prehľadnosti, a taktiež z očakávaní, že si užívateľ najskôr vyskúša počítať úlohy bez alternatív, kde si nutnosť riadkového súčtu rovného 1 zafixuje.

Obrázok 18: Zadanie matíc pravdepodobností prechodov a ocenení prechodov



Zdroj: *Vlastné spracovanie*

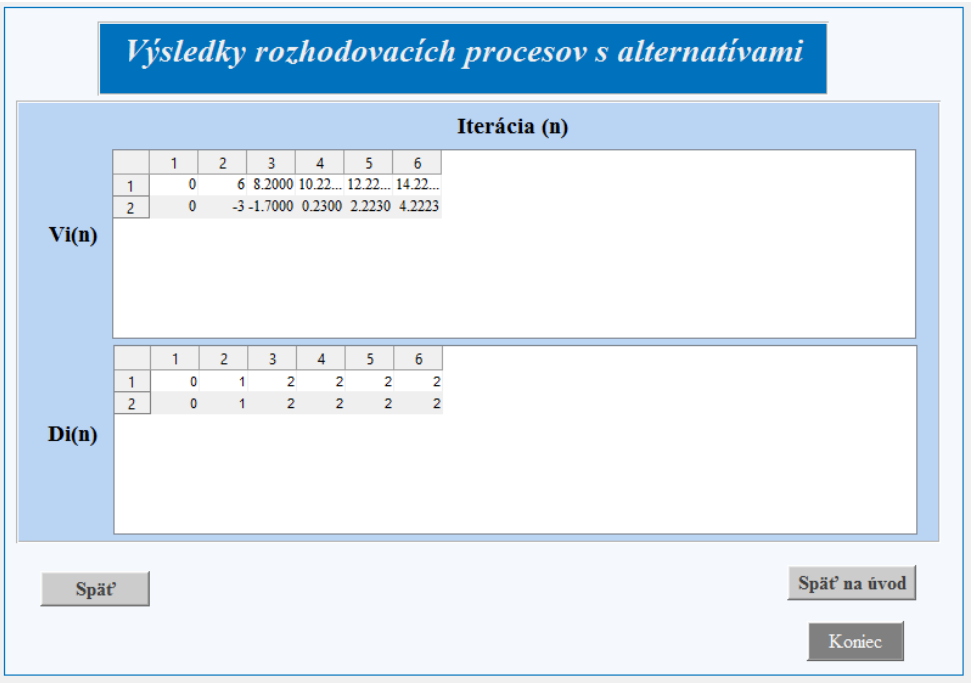
Užívateľ má možnosť zmeniť počet alternatív pomocou tlačidla **Späť**, ale samozrejme tým príde o už zadané hodnoty. Zadané hodnoty sú kontrolované pred vykonaním výpočtu po stlačení **Vypočítaj** a ak sú nájdené akékoľvek chybné hodnoty, je zabránené výpočtu

a užívateľovi je zobrazené príslušné varovanie zhodné s varovaniami v predchádzajúcich typoch úloh.

Výsledky tohto typu úloh sú od predošlých trochu odlišné (Obrázok 19). Najdôležitejším údajom výsledku je vektor d, ktorého hodnoty označujú, ktorá z jednotlivých alternatív, prinesie najväčší úžitok. Ak je ocenenie viacerých alternatív jedného stavu rovnakých, tak algoritmus vyberá ten posledný z nich.

Pre zmenu zadaných hodnôt je potrebné kliknúť na tlačidlo **Späť**. Pre riešenie nového typu úlohy, je pripravené tlačidlo **Späť na úvod**. Tlačidlom **Koniec** môžeme bezpečne ukončiť aplikáciu bez nechcených zmien.

Obrázok 19: Výsledky rozhodovacích procesov s alternatívami



Zdroj: *Vlastné spracovanie*

4.3 Popis častí zdrojového kódu

V tejto časti si rozoberieme zdrojový kód, bez ktorého by vyššie navrhnuté okná nespĺňali požadovanú funkcionalitu. Nebudeme popisovať celý kód ale len jeho časti, ktoré považujeme za podstatné a najviac používané pri tvorbe aplikácie. Taktiež nebudeme rozoberať časti kódu vygenerovaného automaticky programom MATLAB.

Na začiatok si uvedieme najviac používané tlačidlá, ktoré slúžia na prepínanie aplikácie medzi jednotlivými oknami. Ako príklad sme zvolili tlačidlo Pokračovať, na okne pre

zadávanie počtu stavov systému a počtu iterácií výpočtu (Obrázok 20). Na začiatku funkcie pre dané tlačidlo sme si vytvorili tri globálne premenné (**global**) s názvom *pocetstavov, pocetiteracii a funkcia*. Následne sme definovali jednotlivé premenné. Pre získanie hodnôt premenných použijeme funkciu **get**, ktorá pre premennú *pocetstavov* načíta hodnotu zadanú v poli pre zadanie počtu stavov v systéme a takisto pre premennú *pocetiteracii* so zoznamu poľa pre počet iterácií. Hodnoty indexov sú uchovávané pomocou **value.**

Obrázok 20: Callback funkcia tlačidla Pokračovať

```
139     % --- Executes on button press in pushbutton3.
140     function pushbutton3_Callback(hObject, eventdata, handles)
141     % hObject    handle to pushbutton3 (see GCBO)
142     % eventdata  reserved - to be defined in a future version of MATLA
143     % handles    structure with handles and user data (see GUIDATA)
144 -   global pocetstavov
145 -   global pocetiteracii
146 -   global funkcia
147 -   pocetstavov = get(handles.popupmenu3, 'value')
148 -   pocetiteracii = get(handles.popupmenu4, 'value')
149 -   switch funkcia
150 -       case 'kon'
151 -           konstantne
152 -           close vyber
153 -       case 'men'
154 -           parameterBeta
155 -           close vyber
156 -       case 'alt'
157 -           zadajalt
158 -           close vyber
159 -   end
```

Zdroj: *Vlastné spracovanie*

Príkaz **switch-case** slúži na vetvenie programu, ak chceme jeden výraz porovnať s niekoľkými hodnotami. V našom prípade porovnávame hodnotu premennej *funkcia*, s jednotlivými vetvami **case**. V prvej vetve case porovnávame prípad zvolenia úlohy konštantného ocenenia prechodov medzi stavmi v úvodnom okne s výrazom za príkazom switch. Ak hodnota zodpovedá výrazu sú vykonané príkazy pre otvorenie okna (*konstantne)* na zadávanie matíc a ukončenie súčasného okna (*vyber*) .

Pre tvorbu matíc sme využili zdrojový kód zobrazený nižšie (Obrázok 21). Vytvorili sme novú globálnu premennú *maticapravdepodobnosti* a *pocetstavov*, ktorá načíta hodnoty zadané v predchádzajúcom okne z poľa počtu stavov. Matica pravdepodobnosti bude mať predvolené nulové hodnoty (*zeros)* o rozmere počtu stavov systému. Funkcia **set** slúži na uchovanie dát pre ďalší výpočet.

Obrázok 21: Funkcia pre vytvorenie tabuľky

```
% --- Executes during object creation, after setting all properties.
function uitable1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to uitable1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
global maticapravdepodobnosti
global pocetstavov
maticapravdepodobnosti=zeros(pocetstavov);
set (hObject,'data',maticapravdepodobnosti)
```

Zdroj: *Vlastné spracovanie*

Ďalšou časťou kódu, ktorú si opíšeme je riadková kontrola súčtov v matici pravdepodobností prechodov, kde súčet v každom riadku musí byť rovný 1 (Obrázok 22).Táto kontrola sa vykoná po stlačení tlačidla Výpočet. Opäť sme si na začiatku vytvorili globálnu premennú s názvom *sucty* a taktiež aj lokálnu premennú *najdene* s hodnotou 0. Keďže potrebujeme preveriť hodnotu každého riadku v tabuľke súčtov použijeme cyklus **for.** Cyklus for používame, ak poznáme počet opakovaní, bude sa opakovať až kým sa neskontrolujú všetky riadky tabuľky súčtov. Príkaz **if** je vnorený v cykle for, ktorý vyhodnocuje podmienku nerovnosti riadkových súčtov 1. Ak sa v priebehu cyklu for nájde hodnota nerovnajúca sa 1 program vypíše upozornenie *„Riadkový súčet v matici pravdepodobností prechodov nie je rovný 1.“* Po tom čo je príkaz for ukončený program pokračuje za blokom for. V prípade, že sú všetky hodnoty súčtov rovné 1, program pokračuje vo výpočte výsledku.

Obrázok 22: Funkcia pre kontrolu riadkových súčtov

```
104    % --- Executes on button press in pushbutton1.
105    function pushbutton1_Callback(hObject, eventdata, handles)
106    % hObject    handle to pushbutton1 (see GCBO)
107    % eventdata  reserved - to be defined in a future version of MATLAB
108    % handles    structure with handles and user data (see GUIDATA)
109 -  global sucty
110 -  najdene = 0
111 -  for i = 1:length(sucty)
112 -      if (sucty(i)~=1)
113 -          set (handles.text2,'String','Riadkový súčet v matici pravdepodobností prechodov nie je rovný 1.')
114 -          najdene = 1
115 -      end
116 -  end
117 -  if (najdene~=1)
118 -      konvysledok
119 -      close konstantne
120 -  end
```

Zdroj: *Vlastné spracovanie*

Zdrojový kód slúžiaci k zobrazeniu výsledku výpočtu úlohy (Obrázok 23) začína globálnymi premennými, ktorých hodnoty boli definované v iných častiach kódu. Na

výpočet úlohy sme použili cyklus for s vnoreným cyklom for. Vonkajší cyklus for zabezpečuje opakovanie výpočtu v závislosti od počtu iterácií (stĺpce tabuľky) zadaných používateľom. Vnútorný cyklus for opakuje výpočet podľa počtu stavov systému v každej iterácii výpočtu.

Obrázok 23: Funkcia na výpočet úlohy

```
92    % --- Executes during object creation, after setting all properties.
93    function uitable1_CreateFcn(hObject, eventdata, handles)
94    % hObject    handle to uitable1 (see GCBO)
95    % eventdata  reserved - to be defined in a future version of MATLAB
96    % handles    empty - handles not created until after all CreateFcns called
97    global pocetiteracii
98    global pocetstavov
99    global maticapravdepodobnosti
100   global maticaoceneni
101   global vynos0
102   vysledok = vynos0;
103   for i=1:pocetiteracii
104       iteracia = [];
105       for j=1:pocetstavov
106           iteracia = [iteracia; maticapravdepodobnosti(j:j,1:pocetstavov)*maticaoceneni(1:pocetstavov,j:j)
107               + maticapravdepodobnosti(j:j,1:pocetstavov)* vysledok(1:pocetstavov,i)];
108       end
109        vysledok=[vysledok iteracia];
110   end
111   set (hObject, 'data', vysledok)
```

Zdroj: *Vlastné spracovanie*

4.4 Problém vodiča taxi služby

V nasledujúcom texte si ukážeme fungovanie aplikácie na jednoduchom príklade vodiča taxi služby. Príkladom sme sa inšpirovali z knihy [1], problém sme rozšírili o viac stavov a stratégií a modifikovali sme ho pre jednotlivé typy úloh.

Vodič taxi služby pracuje v mestskej časti Bratislava – Petržalka (Obrázok 24), ktorá sa skladá z 7 častí:

1. Pečňa
2. Dvory
3. Kapitulský dvor
4. Háje
5. Kopčany
6. Lúky
7. Janíkov dvor

V každej časti si môže zvoliť z niekoľkých stratégií poskytovania služby, k dispozícii má nasledovné stratégie:

1. Môže jazdiť po uliciach mestskej časti, kým si ho neprivolá nový cestujúci.
2. Zostať na mieste a čakať na rozhlasovú výzvu od dispečera s pokynmi.
3. Presunúť sa na parkovisko pre taxíky na železničnej stanici a čakať v rade na zákazníka.

Petržalka má 1 železničnú stanu v časti 6. Lúky. V časti 1. Pečňa nemá taxikár k dispozícii dispečing. Pre túto časť a zvolenú stratégiu máme k dispozícii pravdepodobnosť pre každú cieľovú oblasť, že cestujúci bude chcieť odvoz práve do danej cieľovej oblasti. Pravdepodobnosti prechodu (Tabuľka 8) a ocenení (Tabuľka 9) za každý prechod, ktoré obsahujú príjem taxikára, náklady spojené s vybranou stratégiou a náklady na cestu, teda sú závislé na zvolenej stratégii, ktorej výber spôsobí, že sa vodič taxi služby zameriava na odlišných zákazníkov. Našou úlohou je nájsť riadenie, t.j. pre každú časť mestskej časti Petržalka zvoliť stratégiu, aby maximalizovala celkový výnos vodiča.

Obrázok 24: Plán mestskej časti Bratislava - Petržalka

Zdroj: WIKIPEDIA.: *Bratislava-mestská časť Petržalka.* Dostupné na internete: < https://goo.gl/sVP32c >

Tabuľka 8: Matica pravdepodobností prechodov

i	k	$p_{ij}(k)$						
		1	2	3	4	5	6	7
1	1	0,2	0,15	0,15	0,3	0,05	0,1	0,05
2	1	0,35	0,1	0,05	0,25	0,05	0,05	0,15
	2	0,15	0,1	0,25	0,15	0,2	0,025	0,125
3	1	0,2	0,1	0,1	0,25	0,1	0,15	0,1
	2	0,1	0,15	0,3	0,15	0,1	0,1	0,1
4	1	0,3	0,05	0,05	0,4	0,025	0,15	0,025
	2	0,2	0,075	0,15	0,15	0,3	0,025	0,1
5	1	0,1	0,025	0,175	0,2	0,3	0,1	0,1
	2	0,05	0,05	0,175	0,2	0,25	0,15	0,125
6	1	0,05	0,05	0,075	0,45	0,15	0,2	0,025
	2	0,15	0,1	0,1	0,2	0,2	0,1	0,15
	3	0,05	0,05	0,05	0,3	0,25	0,15	0,15
7	1	0,15	0,15	0,15	0,2	0,05	0,1	0,2
	2	0,1	0,15	0,175	0,225	0,05	0,1	0,2

Zdroj: *Vlastné spracovanie*

Tabuľka 9: Matica ocenení prechodov

i	k	$r_{ij}(k)$						
		1	**2**	**3**	**4**	**5**	**6**	**7**
1	**1**	4	3	3	6	1	2	1
2	**1**	4	1	1	3	1	1	2
	2	2	2	4	2	3	0	2
3	**1**	3	1	1	3	1	2	1
	2	1	2	4	2	1	1	1
4	**1**	5	1	1	7	0	2	0
	2	3	1	2	2	4	0	1
5	**1**	1	0	2	2	3	1	1
	2	1	1	3	3	4	2	2
6	**1**	1	1	1	6	2	2	0
	2	3	2	2	4	4	2	3
	3	1	1	1	5	4	2	2
7	**1**	2	2	2	3	1	1	3
	2	1	2	2	3	1	1	2

Zdroj: *Vlastné spracovanie*

1. Ocenenie prechodov medzi stavmi

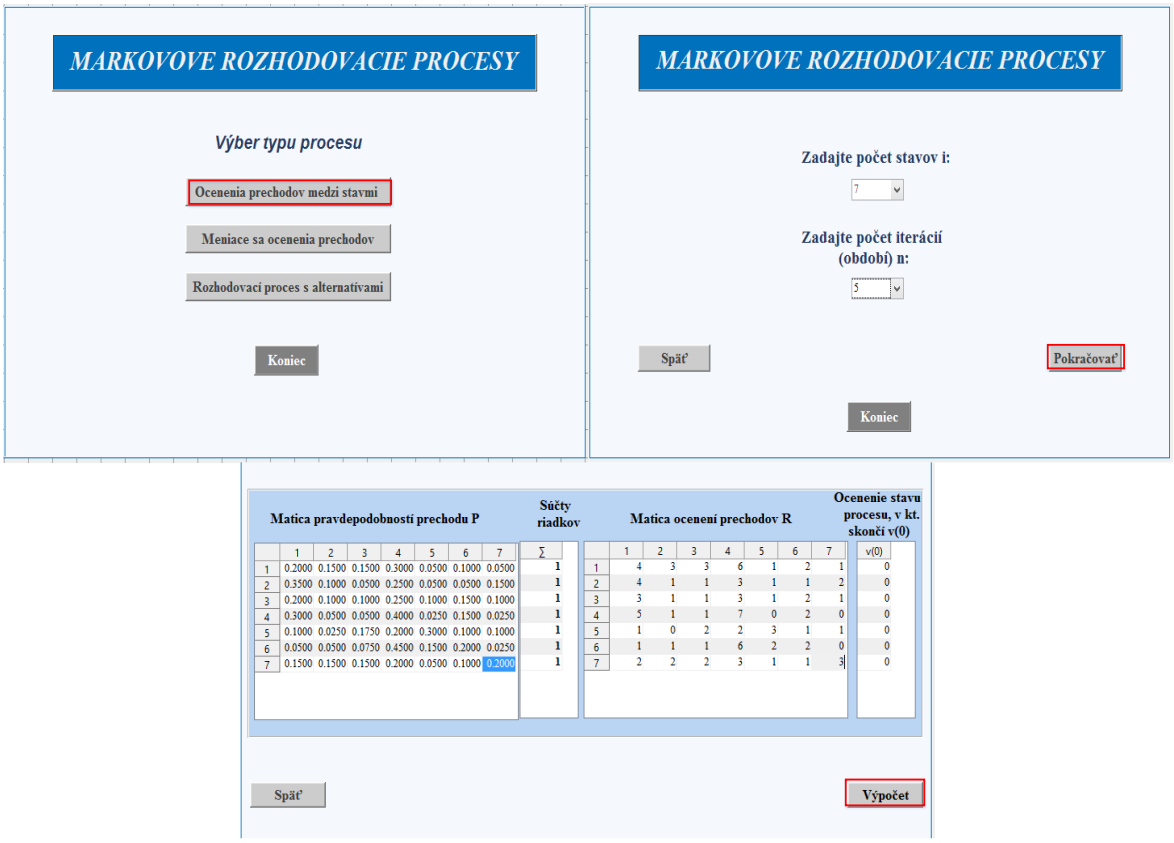
Najskôr si ilustrujeme situáciu, ktorá by nastala ak by vodič taxi služby po každom odvoze zákazníka jazdil po uliciach Petržalky, kým si ho neprivolá nový zákazník. Na základe vyššie uvedených údajov.

Počet stavov systému: 7

Počet iterácií : 5

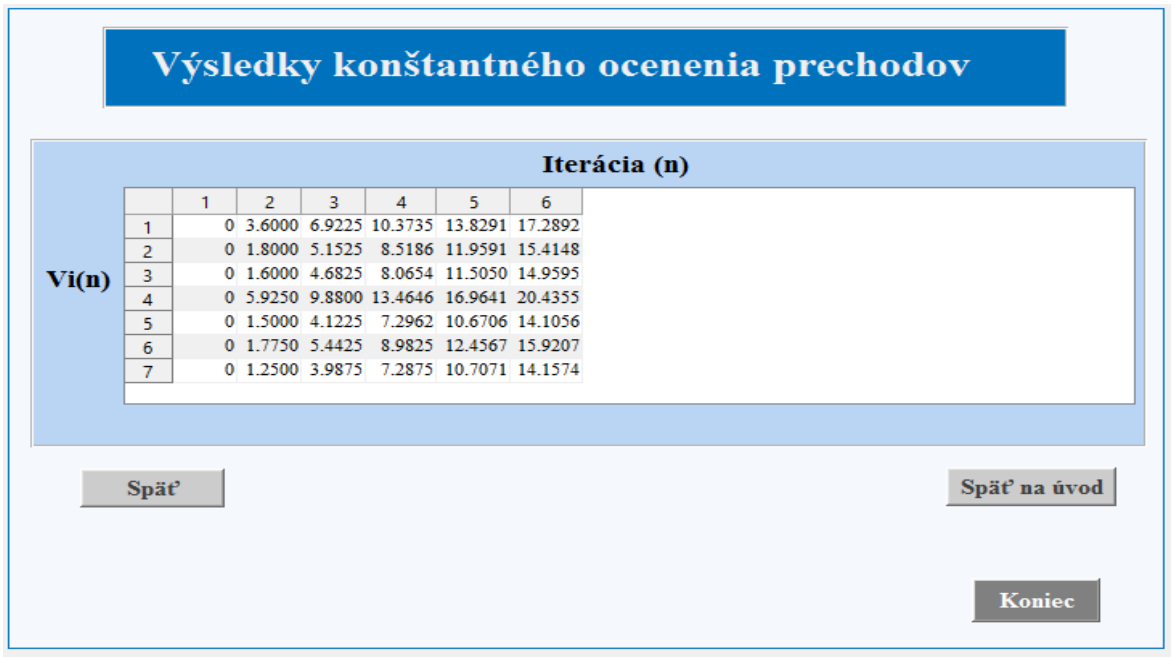
Teraz môžeme zadať uvedené hodnoty do aplikácie (Obrázok 25) a vypočítať očakávaný výnos taxikára za 5 období.

Obrázok 25: Zadávanie uvedených hodnôt



Zdroj: *Vlastné spracovanie*

Obrázok 26: Výsledky problému taxikára s konštantným ocenením prechodov

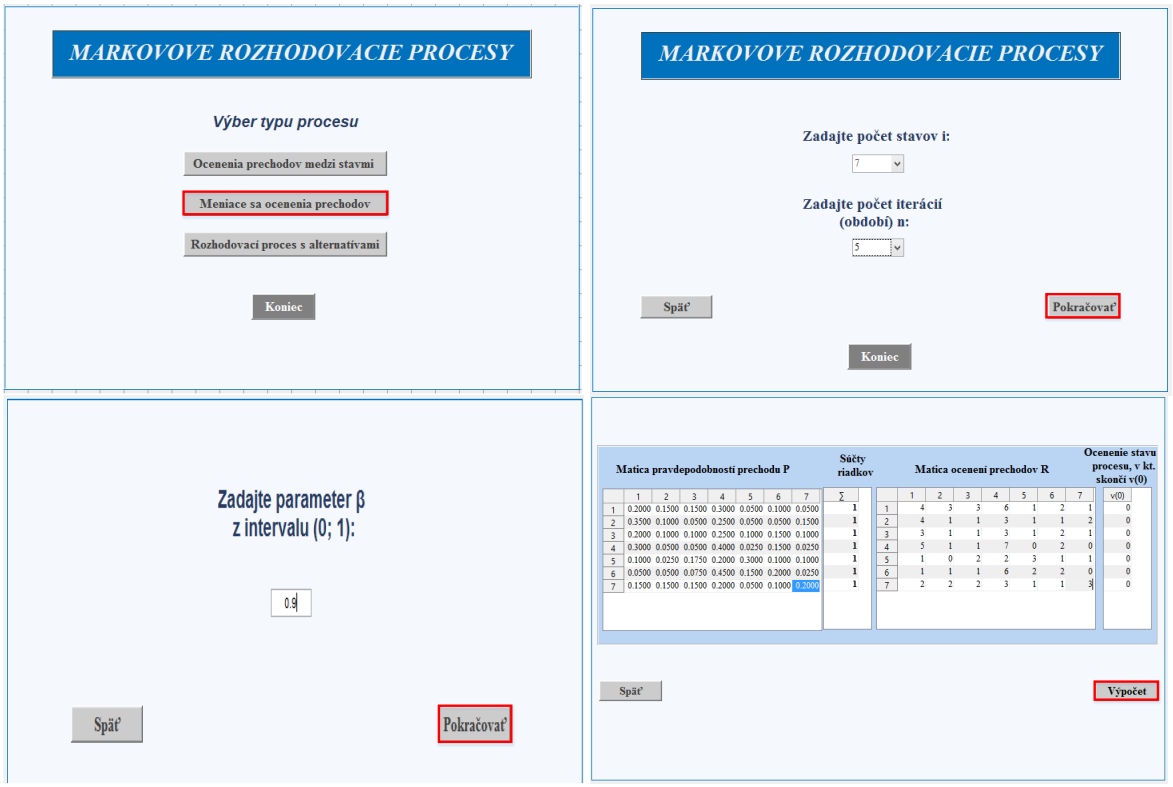


Zdroj: *Vlastné spracovanie*

2. Meniace sa ocenenie prechodov medzi stavmi

Pri výpočte meniacich sa ocenení prechodov medzi stavmi môžeme uvažovať prípad, taxikár sa po každom prechode vráti na parkovisko pre taxíky a čaká v rade, v tomto prípade uvažujeme diskontný faktor **β=0,9**.

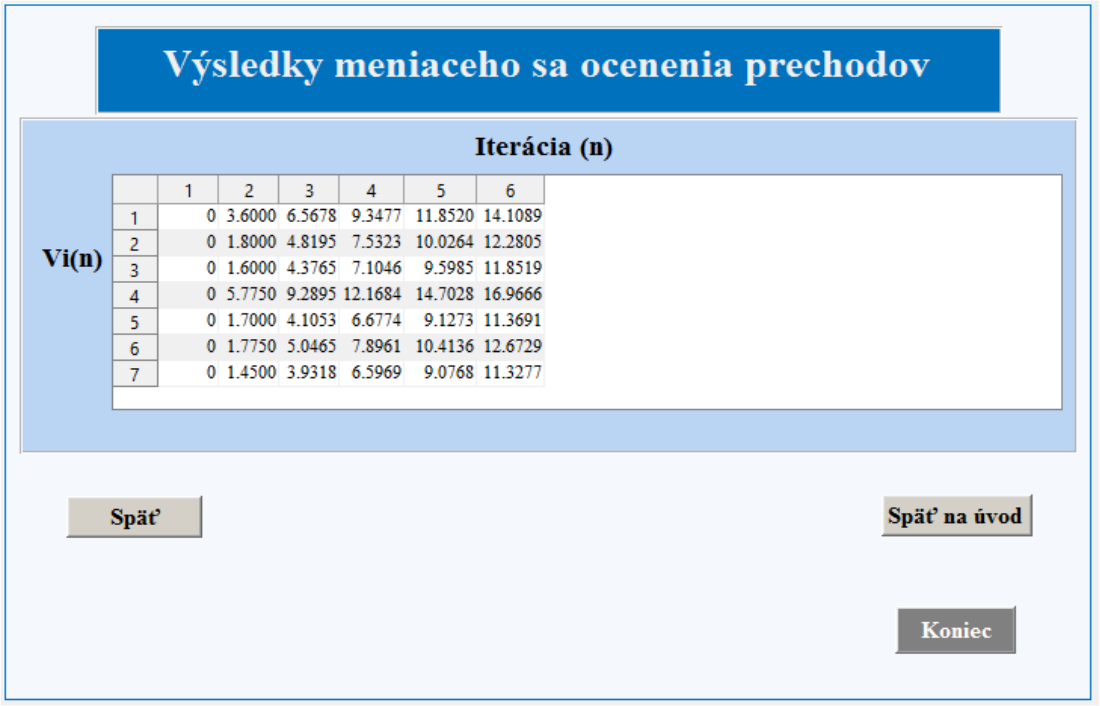
Obrázok 27: Zadanie hodnôt pre meniace sa ocenenia prechodov



Zdroj: *Vlastné spracovanie*

Z výsledkov prvých dvoch úloh (Obrázok 26, Obrázok28) môžeme vidieť, že výnosy taxikára s počtom opakovaní prechodov klesajú, čo môže byť spôsobené dlhým časom stráveným hľadaním nového zákazníka.

Obrázok 28: Výsledky problému taxikára s diskontovaným faktorom β



	1	2	3	4	5	6
1	0	3.6000	6.5678	9.3477	11.8520	14.1089
2	0	1.8000	4.8195	7.5323	10.0264	12.2805
3	0	1.6000	4.3765	7.1046	9.5985	11.8519
4	0	5.7750	9.2895	12.1684	14.7028	16.9666
5	0	1.7000	4.1053	6.6774	9.1273	11.3691
6	0	1.7750	5.0465	7.8961	10.4136	12.6729
7	0	1.4500	3.9318	6.5969	9.0768	11.3277

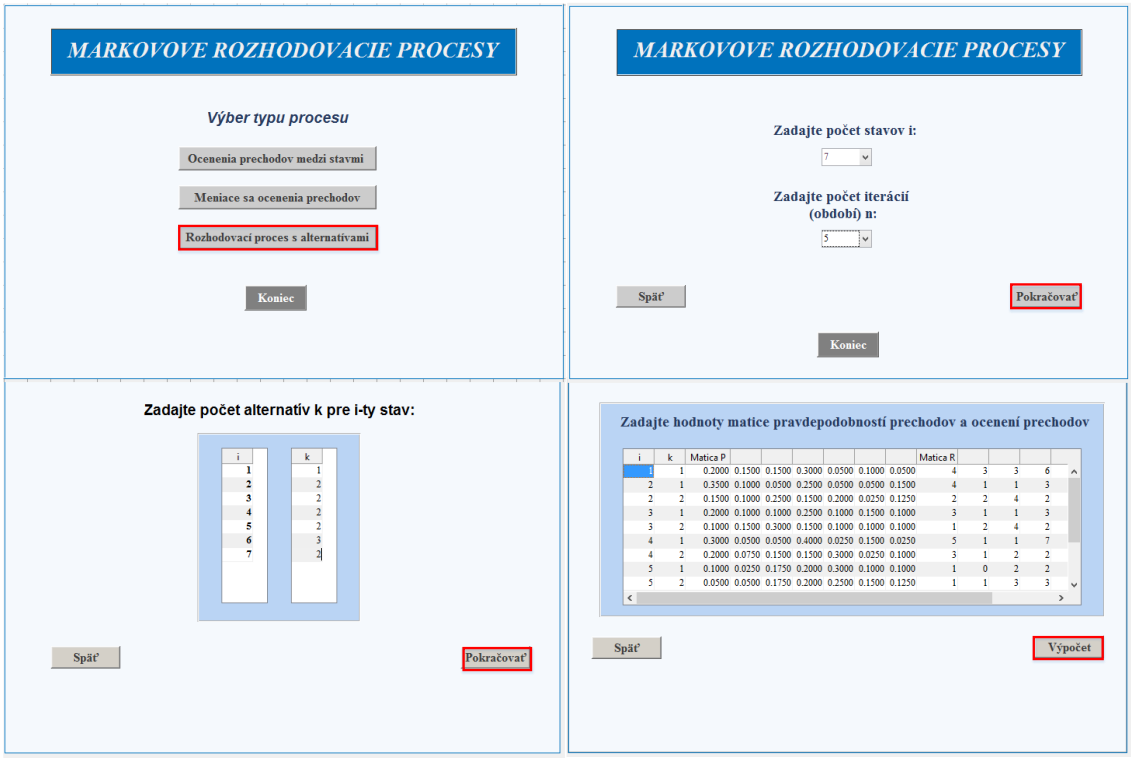
Zdroj: *Vlastné spracovanie*

3. Rozhodujúci proces s alternatívami

V poslednom type úloh počítame so všetkými stratégiami, ktoré si taxikár môže zvoliť a nájdeme optimálne správanie taxikára pre daný typ úlohy.

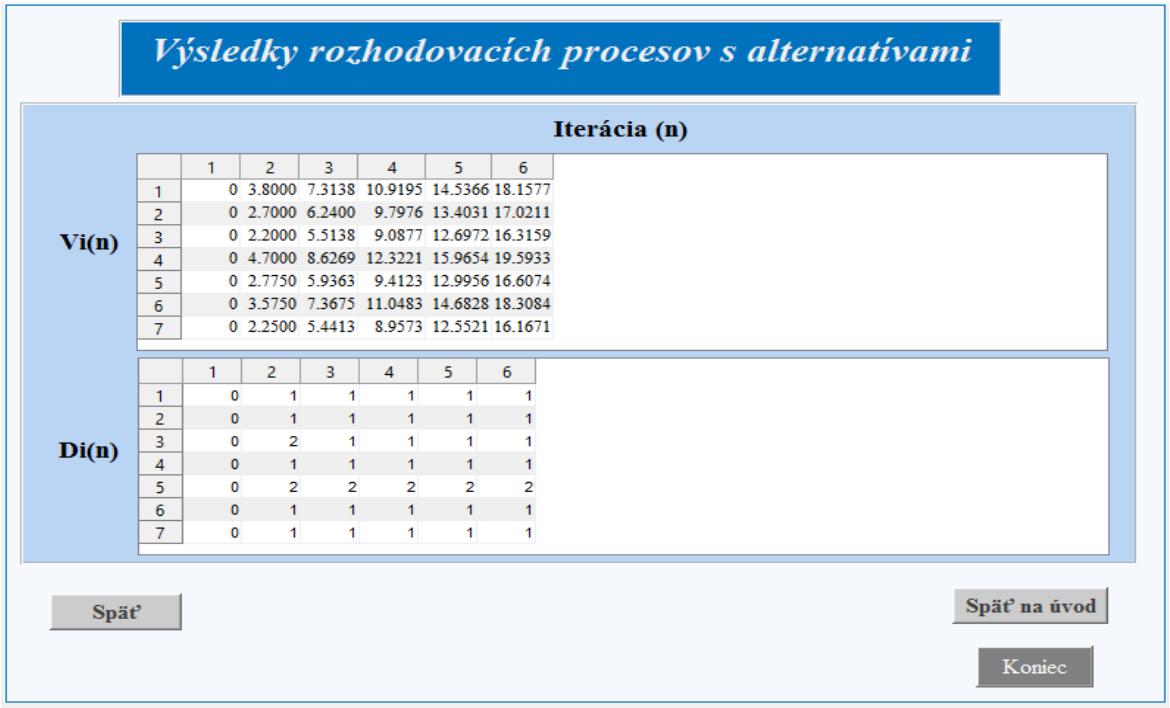
Z výsledkov (Obrázok 30) môžeme vidieť, že ak sa taxikár nachádza v časti 1. Pečňa a 2. Dvory, 4. Háje, 6. Lúky a 7. Janíkov dvor po celý čas výhodné jazdiť po okolí a čakať kým si ho neprivolá zákazník. S časťou tri je to podobné ale tu, pred nástupom posledného zákazníka je výhodnejšie počkať na mieste na výzvu dispečera kde sa tento zákazník nachádza. V časti 5. Kopčany by mal taxikár pri každom vysadení zákazníka čakať na výzvu dispečera.

Obrázok 29: Zadanie hodnôt pre rozhodovacie procesy s alternatívami



Zdroj: *Vlastné spracovanie*

Obrázok 30: Výsledky rozhodovacích procesov s alternatívami



Zdroj: *Vlastné spracovanie*

Záver

V záverečnej práci sme sa zaoberali Markovovými rozhodovacími procesmi. V úvode sme definovali základné pojmy pre pochopenie problematiky a taktiež sme nadobudnuté poznatky demonštrovali na jednoduchom príklade výrobcu cukroviniek. Popísali sme programový prostriedok, v ktorom bola vytvorená programová aplikácia pre riešenie úloh MRP. Oboznámili sme čitateľa s užívateľským prostredím a ozrejmili sme si základy potrebné pre programovanie v MATLABe.

Po definovaní cieľa práce, ktorým bola tvorba programovej aplikácie pre modelovanie Markovových rozhodovacích procesov sme v 3. kapitole popísali postup tvorby aplikácie pomocou nástroja GUIDE. Nakoľko sa jedná o veľmi obsiahlu problematiku, obmedzili sme sa iba na základné postupy. Možnosti vytvárania vlastných aplikácií sú však takmer nevyčerpateľné. Snažili sme sa priblížiť a popísať spôsob návrhu aplikácie v GUIDE čo možno najzrozumiteľnejšie. Práca opisuje prostredie a možnosti jeho prispôsobenia vlastným požiadavkám, jednotlivé objekty z palety komponentov, ich funkcie a možnosti použitia.

Priblížili sme spôsob vytvárania vlastnej aplikácie v programe MATLAB, vkladanie a programovanie komponentov a tiež ich vzájomnú závislosť. Tým pádom môže práca v budúcnosti slúžiť aj ako užitočný návod na tvorbu vlastných aplikácií s rozličným charakterom využitia a činnosti.

Pri plnení hlavného cieľa diplomovej práce sme postupovali na základe vymedzených čiastkových cieľov. Začali sme s oboznámením sa programového prostredia MATLAB, nakoľko sme nemali predošlé praktické skúsenosti s používaním tohto programu. Počas písania práce sme sa naučili používať systém MATLAB a vytvárať aplikácie pomocou nástoja GUIDE. Pokračovali sme tvorbou návrhu užívateľského rozhrania a definovali sme ovládacie prvky, ktoré boli použité. Navrhnuté užívateľské prostredie sme v GUIDE premietli do grafickej podoby. Následne nám bol po uložení vygenerovaný zdrojový kód aplikácie, kde sme pomocou Callback funkcií naprogramovali jednotlivé ovládacie prvky. Všetko je doplnené o praktickú ukážku fungovania aplikácie na probléme vodiča taxi služby.

Celá aplikácia je navrhovaná tak, aby sme do nej mohli ľahko doprogramovať novú funkcionalitu. Nakoľko problematika Markovových rozhodovacích procesov je rozsiahla a obsahuje ešte veľa algoritmov, ktoré by sme mohli v práci zahrnúť. Aplikácia by mohla

byť rozšírená o implementáciu rozhodovacích procesoch s alternatívami počítajúcimi s diskontným faktorom, MRP s nekonečným časovým horizontom, ...

Tvorba tejto diplomovej práce nám umožnila naučiť sa pracovať so systémom MATLAB a prehĺbiť si znalosti o markovových rozhodovacích procesov. Verím, že táto práca bude prínosom pre študentov pri výpočte týchto typov úloh a taktiež pri tvorbe vlastných aplikácií v prostredí MATLAB.

Zoznam použitej literatúry

Knižné pramene

[1] HOWARD, R. 1971. *Dynamic Probabilistic Systems: Volume I:Markov Models, Volume II: Semimarkov and Decision Processes.* Canada. John Wiley & Sons, Inc, 1971. ISBN 0-471-41666-5.

[2] UNČOVSKÝ, L., ČEMINCKÁ K., 1992. *Stochastické procesy a modely.* Bratislava: Editačné stredisko vysoká škola ekonomická Bratislava, 1971. ISBN 80-225-0340-1.

[3] HILLER, F., LIEBERMAN G. 2015. *Introduction to Operation Research.* New York. 10. vydanie. McGraw-Hill Education, 2015. ISBN 978-0-07-352345-3.

[4] ROSS, S. M. 2014. *Introduction to probability models*. San Diego.11.vydanie Elsevier Inc. 2014. ISBN 978-0-12-407948-9.

Internetové zdroje:

[5] KOLOBOV, A. 2015.: *Probabilistic Planning with Markov Decision Processes.* 2015. Dostupné na internete: <https://homes.cs.washington.edu/~mausam/papers/tut12b.pdf>

[6] MATHWORKS. Internetová stránka spoločnosti MathWorks. [online] Dostupné na internete: <http://www.mathworks.com/>

[7] HUMUSOFT. Internetová stránka spoločnosti Humusoft. [online] Dostupné na internete: < http://www.humusoft.cz/matlab/ >

[8] UNDOCUMENTED MATLAB. Portál zaoberajúci sa prácou so systémom MATLAB. [online] Dostupné na internete: <http://undocumentedmatlab.com/>

[9] THE EDITORS OF ENCYCLOPÆDIA BRITANNICA. 2016.: *Andrey Andreyevich Markov - Russian mathematician*. 2016. Dostupné na internete: < http://www.britannica.com/biography/Andrey-Andreyevich-Markov>

[10] TAYLOR, J., 2012.: *Markov decision processes: Lecture notes for STP 425.* Arizona State University. Dostupné na internete: <https://math.la.asu.edu/~jtaylor/teaching/Fall2012/STP425/lectures/MDP.pdf>

[11] STACK OVERFLOW. Blog pre programátorov. [online] Dostupné na internete: <http://stackoverflow.com/questions/tagged/matlab-guide>

Prílohy

Príloha A - Spustenie editoru GUIDE

Príloha B - Vzhľad sprievodcu tvorbou GUI

Príloha C - Nástroj na uľahčenie usporiadania objektov

Príloha D - Nastavenie pracovného prostredia GUIDE

Príloha E - Menu GUIDE

Príloha F - Okná Align Objects a Object Browser

Príloha F - Nastavenie vlastností objektu

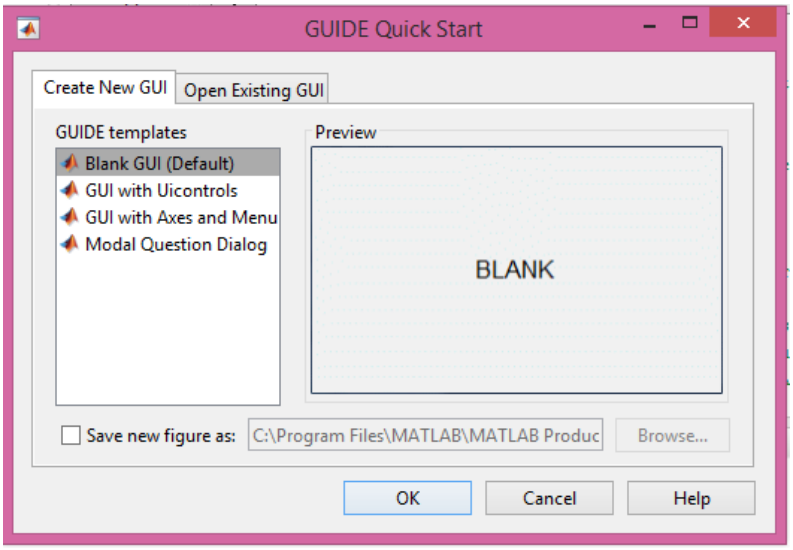
Príloha H - Pridanie Push Button na pracovnú plochu

Príloha I - Vkladanie položiek do String

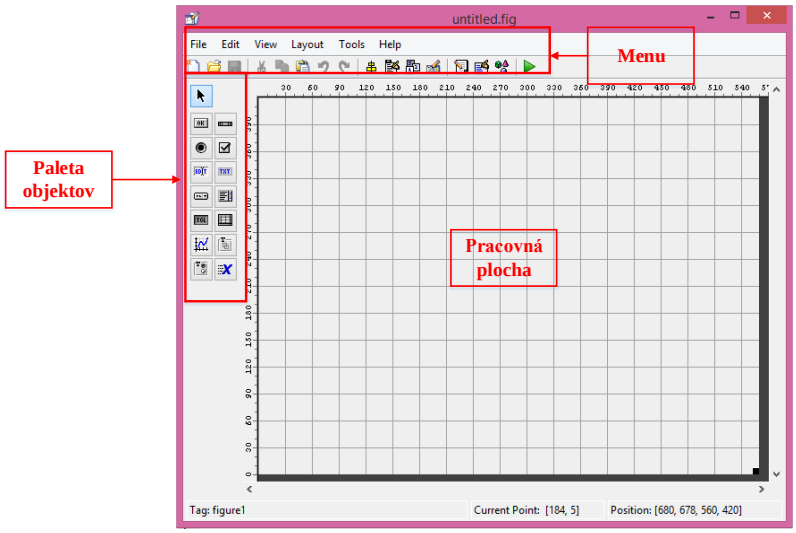
Príloha J – Priečinok so zdrojovým kódom

Príloha K – Priečinok s *.exe súborom aplikácie

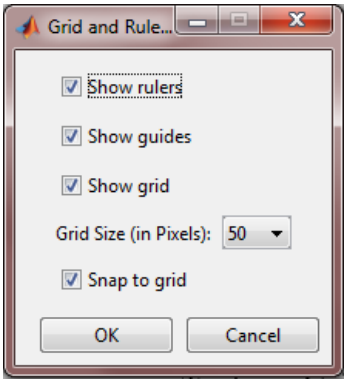
Príloha A - Spustenie editoru GUIDE



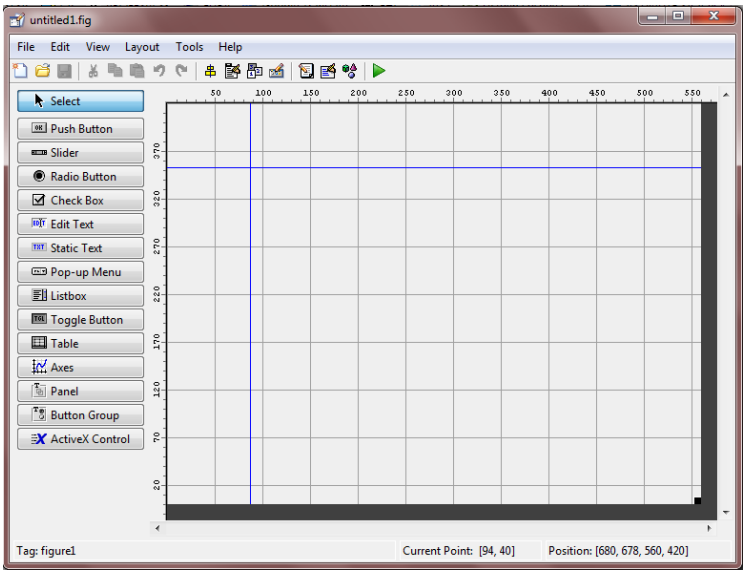
Príloha B - Vzhľad sprievodcu tvorbou GUI



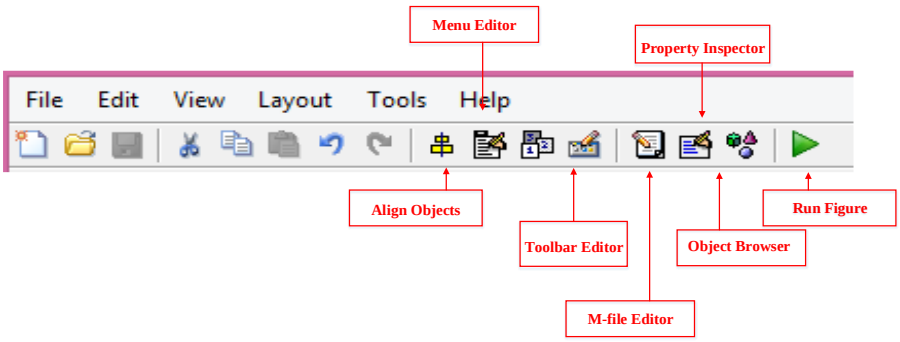
Príloha C - Nástroj na uľahčenie usporiadania objektov



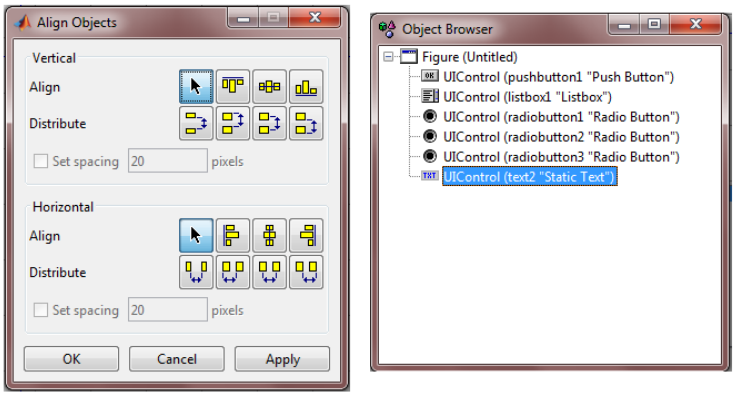
Príloha D - Nastavenie pracovného prostredia GUIDE



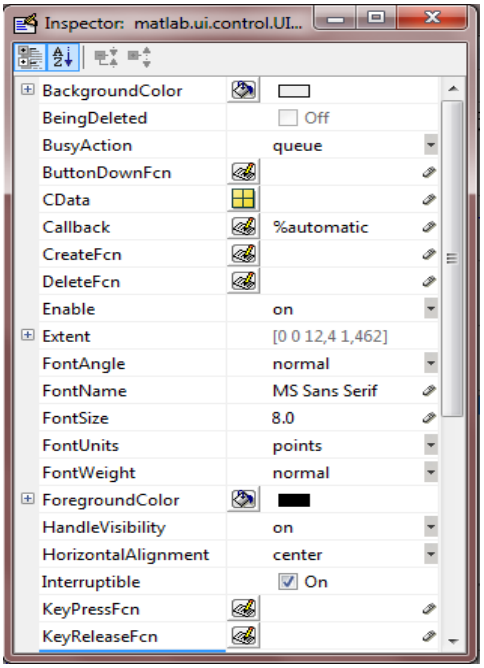
Príloha E - Menu GUIDE



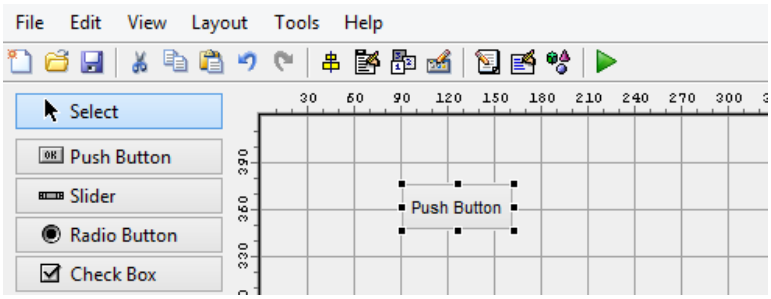
Príloha F - Okná Align Objects a Object Browser



Príloha F - Nastavenie vlastností objektu



Príloha H - Pridanie Push Button na pracovnú plochu



Príloha I - Vkladanie položiek do String