EKONOMICKÁ UNIVERZITA V BRATISLAVE
FAKULTA HOSPODÁRSKEJ INFORMATIKY

Evidenčné číslo: 103006/I/2022/36124048423528196

SIMPLEXOVÁ METÓDA VYUŽITÍM JAZYKA C++

Diplomová práca

2022 **Bc. Matej Drha**

EKONOMICKÁ UNIVERZITA V BRATISLAVE
FAKULTA HOSPODÁRSKEJ INFORMATIKY

SIMPLEXOVÁ METÓDA VYUŽITÍM JAZYKA C++

Diplomová práca

Študijný program: Informačný manažment
Študijný odbor: Ekonómia a manažment
Školiace pracovisko: Katedra matematiky a aktuárstva FHI
Vedúci záverečnej práce: PaedDr. Zsolt Simonka, PhD.

Bratislava 2022 **Bc. Matej Drha**

Ekonomická univerzita v Bratislave
Fakulta hospodárskej informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Matej Drha
Študijný program: informačný manažment (Jednoodborové štúdium, inžiniersky II. st., denná forma)
Študijný odbor: ekonómia a manažment
Typ záverečnej práce: Inžinierska záverečná práca
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Simplexová metóda využitím jazyka C++

Anotácia: Práca bude venovaná využitiu programovacieho jazyka C++ v lineárnej algebre. Súčasťou práce bude vytvorenie programu, ktorý zostavuje a počíta simplexnú tabuľku elementárnej zmeny bázy n-rozmerného lineárneho priestoru s výstupmi použiteľnými ako vo vektorovom, tak aj maticovom počte, pri riešení systémov m lineárnych rovníc s n neznámymi. Programmá slúžiť jednak ako učebná pomôcka pri štúdiu lineárnej algebry, a tiežako efektívna pomôcka v kontrolnej fáze vyučovacieho procesu. V tejto súvislosti požiadavky kladené na jeho tvorbu sú jednoduchosť ovládania, názornosť a dobrá čitateľnosť výstupov, ako aj zabezpečenie jeho funkčnosti pre prípad nevhodných vstupov. Nakoľko simplexová metóda (Simplexový algoritmus) je iteračný výpočtový postup použiteľný napr. pri riešení optimalizačných úloh v operačnom výskume, resp. pre nájdenie optimálneho riešenia úlohy lineárneho programovania (ak také riešenie vôbec existuje), takéto demonštrovanie možností praktického využitia vytvoreného programu bude súčasťou záverečnej práce.

Vedúci: PaedDr. Zsolt Simonka, PhD.
Oponent: Ing. Michal Závodný
Katedra: KMA FHI - Katedra matematiky a aktuárstva FHI
Vedúci katedry: doc. Ing. Michal Páleš, PhD.

Dátum zadania: 02.11.2020

Dátum schválenia: 04.11.2020

doc. Ing. Michal Páleš, PhD.
vedúci katedry

POĎAKOVANIE

Touto cestou by som chcel poďakovať môjmu vedúcemu diplomovej práce PaedDr. Zsoltovi Simonkovi, PhD. za pomoc a cenné rady počas celého vývoja diplomovej práce. Ďalej by som chcel poďakovať mojim rodičom za ich podporu, trpezlivosť a umožnenie štúdia na vysokej škole. V neposlednom rade by som chcel poďakovať celej mojej rodine, priateľke a kamarátom.

ABSTRAKT

DRHA, Matej: *Simplexová metóda využitím jazyka C++*. – Ekonomická univerzita v Bratislave. Fakulta hospodárskej informatiky; Katedra matematiky a aktuárstva FHI. – Vedúci záverečnej práce PaedDr. Zsolt Simonka, PhD. – Bratislava: FHI EU, 2022, 49 s.

Záverečná práca je vypracovaná na tému využitia jazyka C++ na riešenie úloh lineárnej algebry. Cieľom záverečnej práce bolo vytvoriť vytvorenie programu, ktorý zostavuje a počíta simplexnú tabuľku elementárnej zmeny bázy n-rozmerného lineárneho priestoru s výstupmi použiteľnými ako vo vektorovom, tak aj maticovom počte, pri riešení systémov m lineárnych rovníc s n neznámymi. Jednotlivé časti záverečnej práce boli zamerané na popísanie problematiky úloh lineárnej algebry, na stanovenie cieľov, ktoré chceme pomocou práce dosiahnuť, ako aj na podrobný opis programu vytvoreného v rámci praktickej časti diplomovej práce. Výsledkom riešenia danej problematiky je okrem písaného dokumentu aj funkčný aplikácia, ktorá je intuitívna, názorná a dobre čitateľná. Aplikácia používa na počítanie príkladov lineárnej algebry a lineárneho programovania simplexovú metódu.

Kľúčové slová:
lineárna algebra, lineárne programovanie, simplexov algoritmus, elementárna zmena bázy

ABSTRAKT

DRHA, Matej: *Simplex method using C++*. – University of Economics in Bratislava. Faculty of Economic Informatics; Department of Mathematics and Actuarial Science. – Thesis supervisor PaedDr. Zsolt Simonka, PhD. – Bratislava: FHI EU, 2022, 49 s.

The diploma thesis is elaborated on the topic of the use of C++ language to solve tasks of linear algebra. The work's aim was to create a program that generates and calculates a simplex table of elementary change of basis of n-dimensional linear space with outputs usable in vector and matrix calculus, in solving systems of m linear equations with n variables. Individual parts of the diploma thesis were focused on describing the problems of linear algebra, specifying the goals to be achieved through the work, as well as on a detailed description of the program created within the practical part of the thesis. The result of solving the given problem is, in addition to the written document, also a functional application that is intuitive, illustrative, and easy to use. The application uses the simplex method to calculate tasks of linear algebra and linear programming.

Keywords:
linear algebra, linear programming, simplex algorithm, change of basis

Obsah

Úvod

Keďže je lineárna algebra ústredným prvkom takmer všetkých oblastí matematiky, v súčasnosti výučba lineárnej algebry tvorí spolu s matematickou analýzou dva základné piliere univerzitného matematického vzdelania. Základné pojmy, ako matica, vektorový priestor, báza, lineárne zobrazenie a pod., sú abstraktné objekty definované určitými podmienkami alebo axiómami.

V súčasnosti veľmi obľúbená vedná disciplína, operačný výskum, sa zameriava na riešenie problémov manažmentu podnikov pomocou matematických modelov a metód, súčasťou ktorých je lineárna algebra. Simplexová metóda je jedným z algoritmov lineárneho programovania, ktorého cieľom je nájsť optimálne riešenie pri určitých ohraničujúcich podmienkach. Tento algoritmus sa často používa na optimalizačné úlohy výrobných aj podnikových procesoch a taktiež dokáže pomôcť jednotlivcovi pri rozhodovaní.

Lineárna algebra aj operačný výskum sú často predmetmi univerzitného vzdelávania a študenti sa s nimi stretávajú nielen na matematicky zameraných fakultách. Výpočet mnohých úloh lineárnej algebry vyžaduje dlhší netriviálny postup a často rozdelený do viacerých iterácií. Pre overenie riešenia úlohy boli vyvinuté viaceré aplikácie, ktoré často vysvetľujú aj elementárne kroky, a slúžia tak aj na lepšie pochopenie algoritmov. Aj keď takéto aplikácie dokážu pomôcť pri vzdelávacom procese, nie sú často používané na univerzitách, napríklad z dôvodu odlišných elementárnych krokov riešenia úlohy.

Účelom tejto práce je vytvorenie didaktickej pomôcky pre študentov, ktorá používa metódu elementárnej zmeny bázy na riešenie úloh lineárnej algebry a simplexov algoritmus na nájdenie optimálneho riešenia. Na vývoj tohto programu použijeme jazyk C++.

V prvej kapitole sa zoznámime s lineárnou algebrou a jej algoritmami, predovšetkým elementárnou zmenou bázy a simplexovým algoritmom, preskúmame existujúce aplikácie na riešenie príkladov lineárnej algebry a na záver sa ešte zoznámime s knižnicami programovacieho jazyka C++. Druhá kapitola bude opisovať hlavné aj čiastkové ciele. V tretej kapitole sa opíšeme použité metódy a metodiky pri navrhovaní aplikácie. Štvrtá kapitola bude zameraná na implementáciu a prezentáciu aplikácie. V poslednej kapitole porovnáme našu aplikáciu s podobnými existujúcimi aplikáciami a skúsime sa zamyslieť nad prípadnými rozšíreniami aplikácie do budúcnosti.

1 Súčasný stav riešenej problematiky doma a v zahraničí

Prvá časť tejto kapitoly je venovaná lineárnej algebre, metóde elementárnej zmeny bázy, lineárnemu programovaniu a simlexovému algoritmu. Hlavným cieľom opisu metód lineárnej algebry je poukázať na spôsob, ako je v súčasnosti lineárna algebra vyučovaná na fakultách. Táto kapitola má čitateľa oboznámiť s riešenou problematikou práce a priblížiť mu jednotlivé algoritmy, ich prerekvizity, elementárne kroky aj kritické miesta, kvôli lepšiemu pochopeniu ďalších procesov vo vývoji aplikácie na riešenie úloh lineárnej algebry. V druhej časti sa pokúsime čitateľovi priblížiť podobné existujúce riešenia – aplikácie na riešenie úloh lineárnej algebry. Posledná časť je venovaná využitiu programovacieho jazyka C++ na riešenie úloh lineárnej algebry.

1.1 Lineárna algebra

Lineárna algebra je odvetvie matematiky týkajúce sa lineárnych rovníc, lineárnych máp a ich reprezentácií vo vektorových priestoroch a prostredníctvom matíc. Je ústredným prvkom takmer všetkých oblastí matematiky, napríklad je základom moderných prezentácií geometrie, vrátane definovania základných objektov, ako sú čiary, roviny a rotácie. Lineárna algebra sa tiež používa vo väčšine vied a oblastí inžinierstva, pretože umožňuje modelovanie mnohých prírodných javov a efektívne výpočty s takýmito modelmi. Okrem geometrie je základom aj pre ostatné odvetvia matematiky ako matematická analýza, funkcionálna analýza alebo lineárne programovanie.

V minulosti sa lineárna algebra prezentovala prostredníctvom systémov lineárnych rovníc a matíc. V modernej matematike sa vo všeobecnosti uprednostňuje prezentácia prostredníctvom vektorových priestorov, pretože je syntetickejšia, všeobecnejšia (neobmedzuje sa len na prípad konečných rozmerov) a koncepčne jednoduchšia, hoci abstraktnejšia.[1]

Konečný súbor lineárnych rovníc v konečnej množine premenných sa nazýva systém lineárnych rovníc alebo lineárny systém. Systémy lineárnych rovníc tvoria základnú časť lineárnej algebry. Historicky bola na riešenie takýchto systémov vyvinutá lineárna algebra a teória matíc. V modernej prezentácii lineárnej algebry prostredníctvom

vektorových priestorov a matíc možno mnohé problémy interpretovať z hľadiska lineárnych systémov.[2]

V tejto práci sa budeme zaoberať prioritne simplexovej metóde, ktorá používa metódu elementárnej zmeny bázy n-rozmerného lineárneho priestoru. Preto sa v ďalšej kapitole zameriame výlučne na metódu EZB.

1.2 Elementárna zmena bázy lineárneho priestoru

Použitím metódy EZB lineárne nezávislého priestoru L_n(priestor, ktorý obsahuje lineárne nezávislé vektory) možno riešiť niekoľko úloh, medzi ktoré patrí určenie bázy a dimenzie priestoru L_n a jeho podpriestoru $L(A)$. Ďalej môžeme určiť závislosť a nezávislosť systému vektorov, jeho hodnosť, lineárnu kombináciu vektorov a zistiť či daný vektor je alebo nie je lineárnou kombináciou vektorov systému, ale aj určiť, či daný vektor kompatibilný alebo inkompatibilný s L_n. Túto metódu môžeme použiť aj pri operáciách s maticami: na výpočet hodnosti matice, rozklad matice na súčin matíc, výpočet inverznej matice alebo aj výpočet determinantu matice.[3]

1.2.1 Predpoklady použitia algoritmu

Systém vektorov $\{\bar{a}_1,\ \bar{a}_2, \dots, \bar{a}_n\}$sa nazýva báza lineárneho priestoru L ak vektory $\bar{a}_1,\ \bar{a}_2, \dots, \bar{a}_n \in L,\ n \in N$, sú lineárne nezávislé a každý vektor $\bar{x} \in L$ sa dá vyjadriť ako lineárna kombinácia vektorov $\bar{a}_1,\ \bar{a}_2, \dots, \bar{a}_n$. Z tejto definície vyplýva, že $\{\bar{a}_1,\ \bar{a}_2, \dots, \bar{a}_n\}$ je generujúcim systémom alebo množinou generátorov lineárneho priestoru L. Zároveň platí, že dimenzia lineárneho priestoru L, v ktorom existuje n-prvková báza, kde $n \in N$ sa rovná n $(\dim L = n)$ a takýto lineárny priestor sa nazýva konečnorozmerný lineárny priestor (L_n). Na jednoznačné vyjadrenie vektora $\bar{b}$ $(\bar{b} \in L_n)$ ako lineárnej kombinácie vektorov bázy $B = \{\bar{a}_1,\ \bar{a}_2, \dots, \bar{a}_n\}$je potrebné nájsť jeho súradnice v báze B priestoru L_n. Znamená to nájsť také koeficienty $b_1, b_2, \dots, b_n \in R$, aby platilo:

$$\bar{b} = b_1 \cdot \bar{a}_1 + b_2 \cdot \bar{a}_2 + \cdots + b_n \cdot \bar{a}_n.$$

Ak je daná báza $B_0 = \{\bar{a}_1,\ \bar{a}_2, \dots, \bar{a}_k, \dots, \bar{a}_n\}$ lineárneho priestoru L_n, môžeme nájsť inú bázu B_1 začlenením jedného nenulového vektora $\bar{b}$:

$$\bar{b} = (b_1, b_2, \dots, b_k, \dots, b_n) \in L_n\ (b_k \neq 0)$$

za iný vektor bázy B_0. Nová báza B_1 má potom tvar:

$$B_1 = \{\bar{a}_1,\ \bar{a}_2, \dots, \bar{a}_{k-1}, \bar{b},\ \bar{a}_{k+1}, \dots, \bar{a}_n\}.$$

Číslo b_k sa nazýva vedúci prvok elementárnej zmeny bázy priestoru L_n. Ak ľubovoľný vektor $\bar{x} \in L_n$ má v báze B_0 súradnice $\bar{x} = (x_1, x_2, \dots, x_k, \dots, x_n)$, potom bude mať v báze B_1 lineárneho priestoru súradnice

$$\bar{x} = (x_1 - \delta \cdot b_1, x_2 - \delta \cdot b_2, \dots, x_{k-1} - \delta \cdot b_{k-1}, \delta, x_{k+1} - \delta \cdot b_{k+1}, \dots, x_n - \delta \cdot b_n),$$

kde $\delta = \frac{x_k}{b_k},\ b_k \neq 0$. Pre jednoduchší zápis používame tabuľku EZB_0 priestoru L_n (viď. nasledujúci obrázok).

Obrázok č. 1 - tabuľka elementárnej zmeny bázy B_0 priestoru L_n

	Báza	Súradnice vektorov	
		$\bar{b}$	$\bar{x}$
B_0	$\bar{a}_1$	b_1	x_1
	$\bar{a}_2$	b_2	x_2
	$\vdots$	$\vdots$	$\vdots$
	$\bar{b} \rightarrow \bar{a}_k$	$b_k \neq 0$	x_k
	$\vdots$	$\vdots$	$\vdots$
	$\bar{a}_n$	b_n	x_n
B_1	$\bar{a}_1$	0	$x_1 - \delta \cdot b_1 = x'_1$
	$\bar{a}_2$	0	$x_2 - \delta \cdot b_2 = x'_2$
	$\vdots$	$\vdots$	$\vdots$
	$\bar{a}_{k-1}$	0	$x_{k-1} - \delta \cdot b_{k-1} = x'_{k-1}$
	$\bar{b}$	1	$\delta = x_k / b_k$
	$\bar{a}_{k+1}$	0	$x_{k+1} - \delta \cdot b_{k+1} = x'_{k+1}$
	$\vdots$	$\vdots$	$\vdots$
	$\bar{a}_n$	0	$x_n - \delta \cdot b_n = x'_n$

Tabuľka sa skladá zo záhlavia rozdeleného na dve časti a z dvoch horizontálnych polí. Ľavá časť záhlavia sa označuje „báza“, pričom pravá časť reprezentuje „súradnice vektorov“, ktorú rozdelíme na dve časti a označíme $\bar{b}$ a $\bar{x}$.[3]

1.2.2 Postup algoritmu EZB

1. Do prvého stĺpca prvej časti poľa tabuľky označenej „báza“ zapíšeme systém vektorov danej bázy B_0 priestoru L_n a do ďalších dvoch stĺpcov poľa tabuľky zapíšeme dané súradnice vektorov $\bar{b}$ a $\bar{x}$ v báze B_0 priestoru L_n.

2. Ak sa má začleniť vektor $\bar{b}$ do bázy B_0 priestoru L_n, zvolíme si niektorú jej nenulovú súradnicu ($b_k \neq 0$). Výhodné je si zvoliť číslo $b_k \pm 1$ alebo také, ktoré je celočíselným deliteľom celého riadka. Tým je určený vedúci prvok (pivot), ktorý si v tabuľke označíme rámčekom. V tabuľke potom môžeme ľahko definovať vedúci riadok a vedúci stĺpec (viď. Obrázok č. 1). Vedúci prvok určuje začlenenie vektora $\bar{b}$ do bázy B_0 priestoru L_n (označíme $\bar{b} \rightarrow \bar{a}_k$), teda EZB$_0$ na bázu B_1 (označíme $B_0 \rightarrow B_1$). Systém vektorov bázy B_1 zapíšeme do prvého ľavého stĺpca druhej časti tabuľky.

3. Zmenené súradnice $\bar{b}, \bar{x} \in L_n$ v báze B_1 po uskutočnení EZB priestoru L_n zapíšeme do stĺpcov poľa dolnej časti tabuľky v nasledujúcom poradí:

- Zapíšeme súradnice bázického vektora $\bar{b} = (0,0, \dots ,0,1,0, \dots ,0)_{B_1}$, kde číslo 1 určuje jeho k-tu súradnicu v báze $\boldsymbol{B_1}$, pretože:

$$\bar{b} = 0 \cdot \bar{a}_1 + 0 \cdot \bar{a}_2 + \cdots + 1 \cdot \bar{b} + \cdots + 0 \cdot \bar{a}_n$$

- Potom určíme δ ako podiel súradnice x_k z vedúceho riadku a vedúceho prvku $\boldsymbol{b_k}$. $\delta = \frac{x_k}{b_k} = x'_k$
- Ostatné zmenené súradnice x'_i zodpovedajúce súradnici x_i, okrem $x'_k = \delta$, v báze $\boldsymbol{B_1}$ priestoru $\boldsymbol{L_n}$ určíme podľa algoritmu:

$$x'_i = x_i - \delta \cdot b_i, \qquad i = 1, 2, \dots, n; i \neq k$$

ako rozdiel i-tej súradnice x_i vektora $\bar{b}$ vo vedúcom stĺpci EZB$_0$ zapísaného v 1. časti poľa tabuľky.

Pri výpočte súradníc vektorov $\bar{b}, \bar{x} \in L_n$ sa na kontrolu výpočtu používa tzv. súčtový vektor označený ako Σ, ktorý umiestnime do pravého stĺpca tabuľky. Jeho i-ta súradnica je

daná súčtom i-tých súradníc vektorov $\bar{b}$, $\bar{x}$ v báze B_0 priestoru L_n. Súradnice Σ v báze B_1 priestoru L_n určíme rovnakým spôsobom ako súradnice vektora $\bar{x}$ v báze B_1 po EZB$_0$ priestoru L_n a zapíšeme do pravého stĺpca druhej časti poľa tabuľky. Takto zmenená i-ta súradnica súčtového vektora Σ sa musí rovnať súčtu i-tých súradníc $x_i' + b_i'$ vektorov $\bar{b}$ a $\bar{x}$ v báze B_1 priestoru L_n po EZB$_0$ priestoru L_n.[3]

1.2.3 Riešenie úloh lineárnej algebry metódou EZB

1.2.3.1 Riešenie úloh systému vektorov použitím EZB

Pomocou elementárnej zmeny bázy priestoru L_n môžeme určiť, či je systém vektorov $A = \{\bar{a}_1, \bar{a}_2, \dots, \bar{a}_n\} \in L_n$ lineárne závislý alebo lineárne nezávislý, akú má hodnosť, a či vektor $\bar{b}$ je lineárnou kombináciou systému vektorov $A \in L_n$. Úlohu riešime metódou EZB$_0$, kde východisková báza je $B_0 = \{\bar{e}_1, \bar{e}_2, \dots, e_n\}$. Následne zapíšeme súradnice vektorov systému A spolu s vektorom $\bar{b}$ do stĺpcov poľa tabuľky EZB$_0$ a vhodnou voľbou vedúcich prvkov postupne uskutočníme EZB$_0$ priestoru L_n.

Z výslednej tabuľky vieme vyčítať koľko vektorov môžeme začleniť do bázy B_0 priestoru L_n (zvyšné vektory sú ich lineárnou kombináciou). Lineárnu závislosť vektorov zistíme podľa toho, či je aspoň jeden z vektorov $\bar{a}_1, \bar{a}_2, \dots, \bar{a}_n \in L_n$ lineámou kombináciou začlenených (bázických) vektorov systému $A \in L_n$. Maximálny počet vektorov, ktoré môžeme začleniť do bázy je rovný hodnosti systému vektorov $h\{\bar{a}_1, \bar{a}_2, \dots, \bar{a}_n\}$. Z poslednej časti tabuľky zistíme súradnice vektora $\bar{b}$ v báze B. $\bar{b}$ je lineárnou kombináciou tých bázických vektorov, pri ktorých súradnica vektora $\bar{b}$ nie je nulová. Správnosť tohto tvrdenia overíme dosadením súradníc vektorov v báze B_0 do vyjadrenia lineárnej kombinácie; musí platiť rovnosť.[3]

1.2.3.2 Výpočet hodnosti matice použitím EZB

Hodnosťou matice $h(\boldsymbol{A}_{m\times n})$ rozumieme jej stĺpcovú, resp. riadkovú hodnosť a je definovaná vzťahom $h(\boldsymbol{A}) = h_r(\boldsymbol{A}) = h_s(\boldsymbol{A})$, kde h_r, resp. h_s je číslo, ktoré udáva maximálny počet lineárne nezávislých riadkových, resp. stĺpcových vektorov matice A. Hodnosti matice A určíme použitím EZB priestoru L_n, kde z poslednej časti tabuľky zistíme maximálny počet začlenených (lineárne nezávislých) stĺpcových vektorov matice A do bázy B_0 priestoru L_n.[3]

1.2.3.3 Rozklad matice na súčin matíc použitím EZB

Ak sa dá matica $\boldsymbol{A} = [a_{ij}]_{m\times n}$ vyjadriť ako súčin dvoch matíc

$$\boldsymbol{B} = [b_{ij}]_{m\times p} \text{ a } \boldsymbol{C} = [c_{ij}]_{pxn} \text{ v tvare } \boldsymbol{A} = \boldsymbol{B}\cdot\boldsymbol{C},$$

hovoríme, že matica $\boldsymbol{A}$ je rozložená na súčin matíc $\boldsymbol{B}$ a $\boldsymbol{C}$. Z toho vyplýva, že každú maticu A_{mxn} môžeme rozložiť na súčin dvoch matíc: $\boldsymbol{A} = \boldsymbol{E}_m\cdot\boldsymbol{A} = \boldsymbol{A}\cdot\boldsymbol{E}_n$, kde $\boldsymbol{E}_n$, resp. $\boldsymbol{E}_m$ sú jednotkové matice. Pomocou metódy EZB môžeme určiť bázický rozklad matice $\boldsymbol{A}$, za predpokladu, že prvých $h_s(\boldsymbol{A})$ stĺpcových vektorov matice $\boldsymbol{A}$ je lineárne nezávislých. Potom platí: $\boldsymbol{A} = \boldsymbol{B}\cdot\boldsymbol{C} = \boldsymbol{B}\cdot(E_{h_s}|\boldsymbol{D})$, kde $\boldsymbol{B}$ je matica typu $m \times h_s$, vytvorená z prvých h_s lineárne nezávislých stĺpcových vektorov matice $\boldsymbol{A}$ uvedených v tom istom poradí ako v matici $\boldsymbol{A}$ a $\boldsymbol{C}$ je bloková matica typu $h_s \times n$ tvaru $(E_{h_s}|\boldsymbol{D})$, kde E_{h_s} je jednotková matica stupňa h_s a matica $\boldsymbol{D}$ je matica vytvorená zo stĺpcových vektorov, ktorých zložky sú súradnice stĺpcových vektorov $\bar{a}_{h_s+1}, \bar{a}_{h_s+2}, \ldots, \bar{a}_n$ matice $\boldsymbol{A}$ v báze stĺpcového priestoru. O správnosti výpočtu sa môžeme presvedčiť vynásobením matíc $\boldsymbol{B}$ a $\boldsymbol{C}$.[3]

1.2.3.4 Výpočet inverznej matice použitím EZB

Štvorcová matica $\boldsymbol{A}$ n-tého stupňa je invertibilná práve vtedy, ak je regulárna, t.j. ak jej hodnosť sa rovná n, zároveň k nej existuje jedna inverzná matica $\boldsymbol{A}^{-1}$. Pre inverznú maticu platí vzťah: $\boldsymbol{A}\cdot\boldsymbol{A}^{-1} = \boldsymbol{A}^{-1}\cdot\boldsymbol{A} = \boldsymbol{E}$, pričom $\boldsymbol{E}$ je jednotková matica stupňa n. Metódou EZB hľadáme inverznú maticu tak, že do tabuľky EZB_0 zapíšeme stĺpcové vektory matice $(\boldsymbol{A}|\boldsymbol{E})$ a postupne uskutočníme elementárne zmeny bázy. Z poslednej časti tabuľky zistíme hodnosť matice $\boldsymbol{A}$, resp. či je matica regulárna a či k nej existuje inverzná matica. Súradnice vektorov $\bar{e}_1, \bar{e}_2, \ldots, \bar{e}_n$ v báze $\bar{a}_1, \bar{a}_2, \ldots, \bar{a}_n$ určujú zložky stĺpcových vektorov inverznej matice $\boldsymbol{A}^{-1}$. O správnosti výpočtu sa presvedčíme dosadením do rovnosti $\boldsymbol{A}\cdot\boldsymbol{A}^{-1} = \boldsymbol{A}^{-1}\cdot\boldsymbol{A} = \boldsymbol{E}$.[3]

1.2.3.5 Výpočet determinantu použitím EZB

Ak uskutočníme na stĺpcových vektoroch matice $\boldsymbol{A} = [a_{ij}]_{n\times n}, n \geq 2$ takých n elementárnych zmien bázy, že postupne vektory jednotkovej bázy nahrádzame stĺpcovými vektormi matice $\boldsymbol{A}$, až kým dostaneme bázu $B = \{\bar{a}_{k_1}, \bar{a}_{k_2}, \ldots, \bar{a}_{k_n}\}$ tvorenú iba stĺpcovými vektormi matice $\boldsymbol{A}$, potom sa determinant matice $\boldsymbol{A}$ rovná súčinu vedúcich prvkov

elementárnych zmien báz a čísla$(-1)^{I(k_1,k_2,\dots,k_n)}$, kde $I(k_1,k_2,\dots,k_n)$ je počet inverzií v permutácii $(k_1,k_2,\dots,k_n)$. Pokiaľ matica nie je regulárna, tak determinant je rovný 0.[3]

1.2.3.6 Riešenie systému lineárnych rovníc metódou EZB

Systém *m* lineárnych rovníc s *n* neznámymi je riešiteľný (má riešenie), ak má aspoň jedno riešenie. Frobeinova veta hovorí, že systém *m* lineárnych rovníc s *n* neznámymi $A \cdot \bar{x} = \bar{b}$ je riešiteľný práve vtedy, ak hodnosť matice systému A sa rovná hodnosti rozšírenej matice systému $\bar{A}$, t. j. ak $h(A) = h(\bar{A})$. Frobeinova veta hovorí len o existencii riešenia systému lineárnych rovníc a nie o tom, koľko riešení má. Pre počet riešení platí pravidlo, že ak $h(A) = h(\bar{A}) = n$, tak systém rovníc má jediné riešenie a ak $h(A) = h(\bar{A}) < n$, tak existuje nekonečne veľa riešení závislých od $n - h(A)$ parametrov (voľných neznámych). Hodnosti matíc A a $\bar{A}$ určíme použitím EZB priestoru L_n, kde z poslednej časti tabuľky zistíme maximálny počet začlenených (lineárne nezávislých) stĺpových vektorov matice A do bázy B_0 priestoru L_n.

Ak je systém m lineárnych rovníc s n neznámymi $\boldsymbol{A} \cdot \bar{x} = \bar{b}$ riešiteľný, má hodnosť matice $h(\boldsymbol{A}) = h_s$ a prvých h_s stĺpcových vektorov matice systému je lineárne nezávislých, potom všeobecné riešenie systému lineárnych rovníc je dané vzťahom $\bar{x}_h = \bar{d} - \boldsymbol{D} \cdot \bar{x}_s$ a množina všetkých riešení $M = \{\bar{x} \in L_n : \boldsymbol{A} \cdot \bar{x} = \bar{b}\}$ systému rovníc je daná vzťahom:

$$\bar{x} = \begin{pmatrix} \bar{x}_h \\ \bar{x}_s \end{pmatrix} = \begin{pmatrix} \bar{d} \\ 0 \end{pmatrix} + \begin{pmatrix} -\boldsymbol{D} \\ \boldsymbol{E_s} \end{pmatrix} \cdot \bar{x}_s,$$

kde $\bar{x}_h$ je stĺpcový vektor bázických neznámych, $\bar{x}_s$ je stĺpový vektor nebázických premenných, $\bar{d}$ je stĺpcový vektor, ktorého zložky sú súradnice vektora $\bar{b}$ v báze $S(\boldsymbol{A})$, **D** je matica, ktorej prvky sú súradnice $n - h$ nebázických premenných stĺpcových vektorov matice $\boldsymbol{A}$ v báze $S(\boldsymbol{A})$ a $\boldsymbol{E_s}$ je jednotková matica stupňa s $(s = n - h)$. Partikulárne riešenie získame dosadením ľubovoľných reálnych čísiel za nebázické premenné.

Pri riešení systému lineárnych rovníc použitím metódy elementárnej zmeny bázy B_0 priestoru L_n je postup nasledovný:

Zapíšeme stĺpcové vektory rozšírenej matice $(\bar{a}_1, \bar{a}_2, \dots, \bar{a}_n | \bar{b})$ do tabuľky EZB_0 a považujeme ich zložky za súradnice v jednotkovej báze B_0 priestoru L_n.

1. Postupne uskutočňujeme EZB_0 priestoru L_n. Z poslednej časti tabuľky EZB_0 nájdeme $h(\bar{A})$, $h(A)$ ako $dim S(\boldsymbol{A}) = h(A)$, a rozhodneme o riešiteľnosti a o počte riešení systému lineárnych rovníc na základe vyššie uvedených podmienok.

2. Z poslednej časti tabuľky nájdeme súčasne všeobecné riešenie systému rovníc, množinu riešení, partikulárne a bázické riešenie.[3]

1.3 Lineárne programovanie

Lineárne programovanie je jedna z najrozpracovanejších oblastí vednej disciplíny operačný výskum, ktorý je zameraný na riešenie problémov manažmentu podnikov pomocou matematických modelov a metód lineárnej algebry. Pod pojmom matematický model ekonomickej reality si môžeme predstaviť zjednodušené formalizované zobrazenie ekonomických zákonitostí pomocou matematických vzťahov. Cieľom lineárneho programovania je nájsť najlepšie riešenie pri určitých ohraničujúcich podmienkach.

Úlohou lineárneho programovania je hľadanie extrémov lineárnych funkcií, pričom ohraničujúce podmienky (rovnice, nerovnice) sú takisto lineárne. Ďalej sa budeme zaoberať iba jednokriteriálnymi úlohami, resp. úlohami skalárnej optimalizácie, t. j. budeme uvažovať iba jednu účelovú funkciu. Úloha lineárneho programovania je definovaná jej účelovými funkciami, ohraničujúcimi podmienkami a špeciálnymi podmienkami pre premenné (viď. obrázok nižšie).

Obrázok č. 2- všeobecná formulácia úlohy matematického programovania

1.	***Stanovené ciele / účelové funkcie*** max, resp. min	$f_1(x_1, x_2, \dots x_n)$ $f_2(x_1, x_2, \dots x_n)$ ... $f_k(x_1, x_2, \dots x_n)$
2.	***Štruktúrne (ohraničujúce) podmienky***	$g_1(x_1, x_2, \dots x_n) \geq 0$ $g_2(x_1, x_2, \dots x_n) \geq 0$ ... $g_m(x_1, x_2, \dots x_n) \geq 0$
3.	***Špeciálne podmienky pre premenné (podmienky nezápornosti, dodatočné podmienky)***	$x_j \geq 0$
		$x_j \in D_j$

Z obrázku môžeme definovať jednotlivé premenné: n je počet rozhodovacích premenných, m je počet ohraničujúcich podmienok, k je počet účelových funkcií (v našom prípade vždy 1), x_j sú rozhodovacie premenné ($j = 1 \dots n$), f_s sú reálne funkcie premenných $x_1 \dots x_n$, ktoré opisujú ciele optimalizácie ($s = 1 \dots k$), g_i sú reálne funkcie premenných $x_1 \dots x_n$, ktoré opisujú ohraničujúce podmienky ($i = 1 \dots m$), D_j sú množiny tých hodnôt reálnych čísel, ktoré môžu nadobúdať premenné x_j ($j = 1 \dots n$). [4]

1.4 Simplexová metóda

Simplexov algoritmus je všeobecná metóda na hľadanie optimálneho riešenia úlohy lineárneho programovania, pričom neskúma všetky prípustné riešenia, ale iba bázické riešenia. Počet bázických riešení predstavuje maximálny počet možností, ktorými možno vybrať bázu z n premenných, teda počet kombinácií m-tej triedy z n prvkov:

$$\binom{n}{m} = \frac{n!}{m!(n-m)!}.$$

Simplexov algoritmus je iteračná procedúra, ktorá po určitom konečnom počte krokov umožňuje určiť optimálne riešenie alebo určiť, že optimálne riešenie neexistuje. V každej iterácií možno získať prípustné bázické riešenie, ktoré nie je horšie ako predchádzajúce, a následne sa testuje, či je optimálne. Ak prípustné bázické riešenie neexistuje, procedúra končí.[4]

1.4.1 Predpoklady použitia algoritmu

Pred aplikáciou primárneho simplexového algoritmu musí byť úloha lineárneho programovania upravená na taký tvar, aby všetky ohraničenia (okrem podmienok nezápornosti premenných) boli v tvare „=“, pričom koeficienty pravých strán musia byť nezáporné a matica koeficientov sústavy ohraničení $\boldsymbol{A}$ musí obsahovať jednotkovú submaticu $\boldsymbol{I}$ rozmeru $m \times m$, ktorá tvorí východiskovú bázu. Aby sme mohli použiť primárny simplexov algoritmus, musí existovať východiskové prípustné bázické riešenie, t. j. musí byť splnená podmienka primárnej prípustnosti.[4]

1.4.2 Postup simplexového algoritmu

Pokiaľ sme určili východiskové prípustné bázické riešenie, je potrebné otestovať jeho optimálnosť, pričom môžu nastať dve situácie: preskúmavané riešenie je optimálne

a algoritmus sa končí, alebo riešenie nie je optimálne a vo výpočte pokračujeme. Kritérium optimálnosti vzhľadom na typ extremalizácie účelovej funkcie môžeme odvodiť takto:

Ak vektor x je bázickým prípustným riešením s ohľadom na bázu $\boldsymbol{B}$ a ak platí:

- v prípade maximalizačnej účelovej funkcie: $c_j - z_j = c_j - c_B^T \boldsymbol{B}^{-1} \boldsymbol{A}_j \leq 0,\ j \in \boldsymbol{I}_N$
- v prípade minimalizačnej účelovej funkcie: $c_j - z_j = c_j - c_B^T \boldsymbol{B}^{-1} \boldsymbol{A}_j \geq 0,\ j \in \boldsymbol{I}_N$

potom je vektor x optimálnym riešením úlohy lineárneho programovania.

V prípade, že vo výpočte pokračujeme, je potrebné zvoliť vstupujúcu a vystupujúcu premennú z bázy. Vstupujúca premenná ($x_s, s \in \boldsymbol{I}_n$) je tá, pre ktorú je najviac porušené kritérium optimálnosti, t. j.:

v prípade maximalizačnej úlohy: $c_s - z_s = \max \{c_j - z_j | c_j - z_j > 0\},\ j \in \boldsymbol{I}_N$

v prípade minimalizačnej úlohy: $c_s - z_s = \min \{c_j - z_j | c_j - z_j < 0\},\ j \in \boldsymbol{I}_N$

Identifikovaním vstupujúcej premennej zároveň získavame vedúci stĺpec. Pri identifikácií vystupujúcej premennej môžu nastať dve situácie:
Všetky prvky $x_{is} \leq 0,\ i \in \boldsymbol{I}_B$, t. j. že účelová funkcia je na množine prípustných riešení neohraničená a jej hodnota môže rásť alebo klesať k $\pm\infty$. Ak je aspoň jedna premenná $x_{is} > 0$, vystupujúcu premennú, resp. vedúci stĺpec určíme zo vzťahu:

$$\min \{\frac{x_{i0}}{x_{is}} | x_{is} > 0, x_{i0} \in \boldsymbol{B}^{-1} b\}.$$

Vedúcim prvkom sa stáva premenná x_{is}, na priesečníku vedúceho riadka a vedúceho stĺpca.

Ďalším krokom je prechod k novému bázickému prípustnému riešeniu použitím EZB. Proces sa opakuje, pokiaľ nezískame optimálne riešenie, alebo nedokážeme, že optimálne riešenie neexistuje. Nasledujúci príklad zobrazuje použitie simplexového algoritmu v ideálnom prípade.[4]

Príklad:

$$\max f(x) = x_1 + 2x_2$$
$$2x_1 + 5x_2 \leq 10$$
$$4x_1 + x_2 \leq 4$$

Riešenie:

$$\max f(x) = x_1 + 2x_2$$
$$2x_1 + 5x_2 + s_1 = 10$$
$$4x_1 + x_2 + s_2 = 4$$

Optimálne riešenie zistíme zo simplexovej tabuľky (viď. tabuľka nižšie). Účelová funkcia dosahuje svoje maximum pre hodnoty x_1 a x_2 nasledovne:

$$\max f(x) = \frac{5}{9} + 2 \cdot \frac{16}{9} = \mathbf{\frac{37}{9}}$$

Tabuľka č. 1- Riešenie úlohy lineárneho programovania simplexovou metódou

ouX_B	c_j	1	2	0	0	b
	c_B	x_1	x_2	s_1	s_2	
s_1	0	2	5	1	0	10
s_2	0	4	1	0	1	4
$c_j - z_j$		1	2	0	0	0
x_2	2	2/5	1	1/5	0	2
s2	0	18/5	0	-1/5	1	2
$c_j - z_j$		1/5	0	-2/5	0	4
x_2	2	0	1	2/9	-1/9	16/9
x_1	1	1	0	-1/18	5/18	5/9
$c_j - z_j$		0	0	-7/18	-1/18	37/9

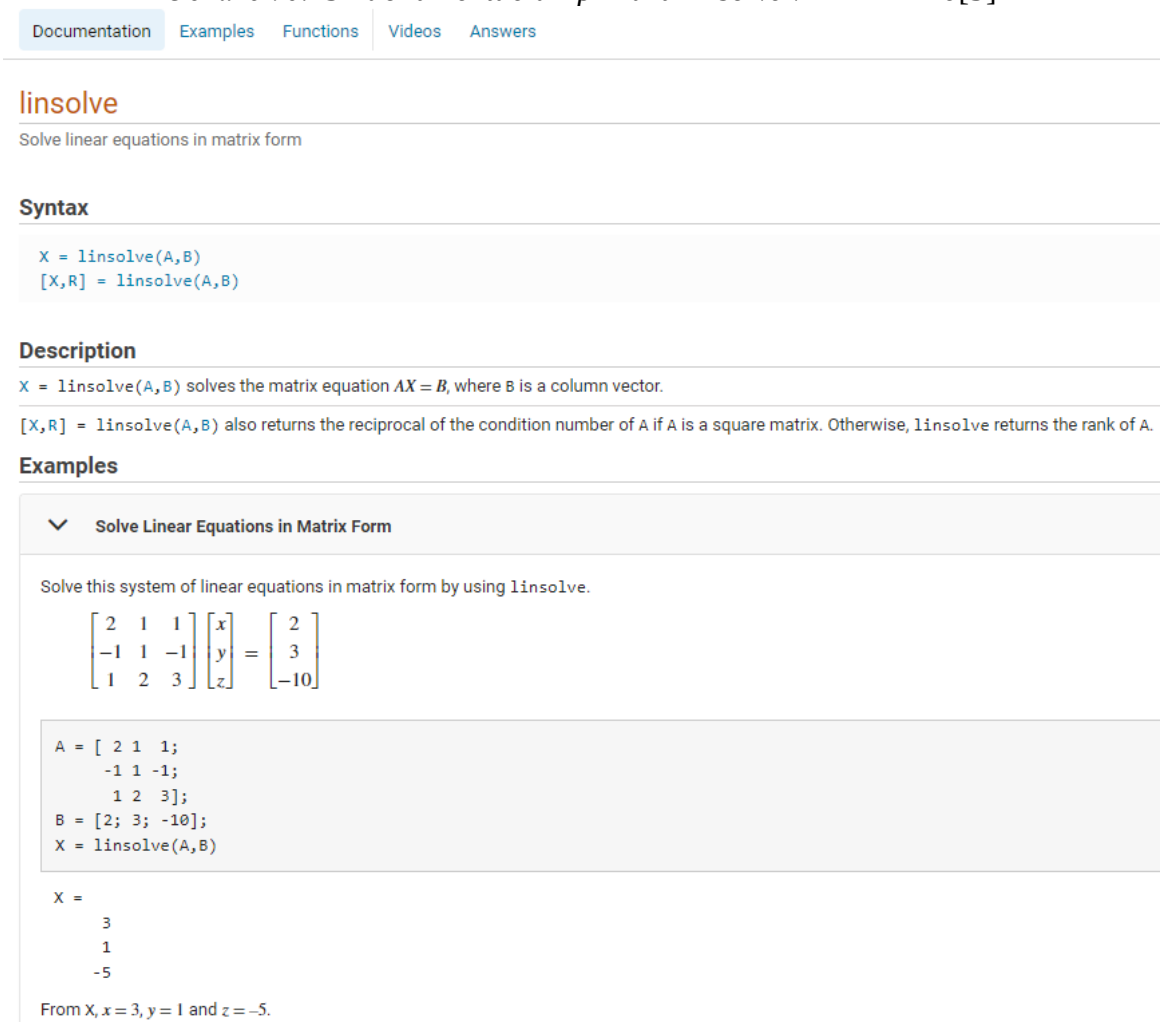
1.5 Existujúce aplikácie na riešenie úloh lineárnej algebry

1.5.1 MATLAB

MATLAB je programovacia a numerická výpočtová platforma, ktorú používajú milióny inžinierov a vedcov na analýzu údajov, vývoj algoritmov a vytváranie modelov. Kombinuje desktopové prostredie vyladené pre iteračnú analýzu a návrhové procesy s programovacím jazykom, ktorý pracuje s dátami vo forme vektorov a matíc. Obsahuje Live Editor na vytváranie skriptov, ktoré kombinujú kód, výstup a formátovaný text v spustiteľnom „poznámkovom bloku“ (notebook).[5]

Tým že je MATLAB veľmi populárna a rozšírená platforma, ktorá ponúka naozaj širokú škálu algoritmov a modelov na takmer všetky matematické a výpočtové problémy, stal sa prirodzene voľbou číslo jedna pre mnoho študentov a vedcov. Ponúka naozaj obsahovo kvalitnú dokumentáciu, možnosť vývoja vlastných výpočtových programov s využitím algoritmov MATLAB-u, dátovú analýzu, vizualizácie dát a mnoho ďalšieho. Nasledujúci obrázok uvádza dokumentáciu k príkazu *linsolve*, ktorý sa používa na riešenie sústavy lineárnych rovníc.

Obrázok č. 3 - dokumentácia k príkazu linsolve v MATLAB-e[5]



Documentation Examples Functions Videos Answers

linsolve

Solve linear equations in matrix form

Syntax

```
X = linsolve(A,B)
[X,R] = linsolve(A,B)
```

Description

`X = linsolve(A,B)` solves the matrix equation $AX = B$, where `B` is a column vector.

`[X,R] = linsolve(A,B)` also returns the reciprocal of the condition number of `A` if `A` is a square matrix. Otherwise, `linsolve` returns the rank of `A`.

Examples

Solve Linear Equations in Matrix Form

Solve this system of linear equations in matrix form by using `linsolve`.

$$\begin{bmatrix} 2 & 1 & 1 \\ -1 & 1 & -1 \\ 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \\ -10 \end{bmatrix}$$

```
A = [ 2 1  1;
     -1 1 -1;
      1 2  3];
B = [2; 3; -10];
X = linsolve(A,B)
```

```
X =
     3
     1
    -5
```

From `X`, $x = 3$, $y = 1$ and $z = -5$.

Ako môžeme vidieť príklad v časti „Examples“, na jeho riešenie je potrebné definovať matice a zavolať funkciu *linsolve,* ktorá vráti koeficienty premenných. Hlavnou nevýhodou takéhoto spôsobu riešenia úlohy je obmedzenie v grafickom zobrazení. Používateľ nevidí maticu v prehľadnej tabuľke, chýbajú aj elementárne kroky riešenia príkladu, ako aj ich vysvetlenie a zdôvodnenie. Používanie platformy Live Editor je pre začiatočníkov trošku chaotické a nezrozumiteľné, potrebná je aspoň základná znalosť vysokoúrovňového programovacieho jazyka (napr. Python).

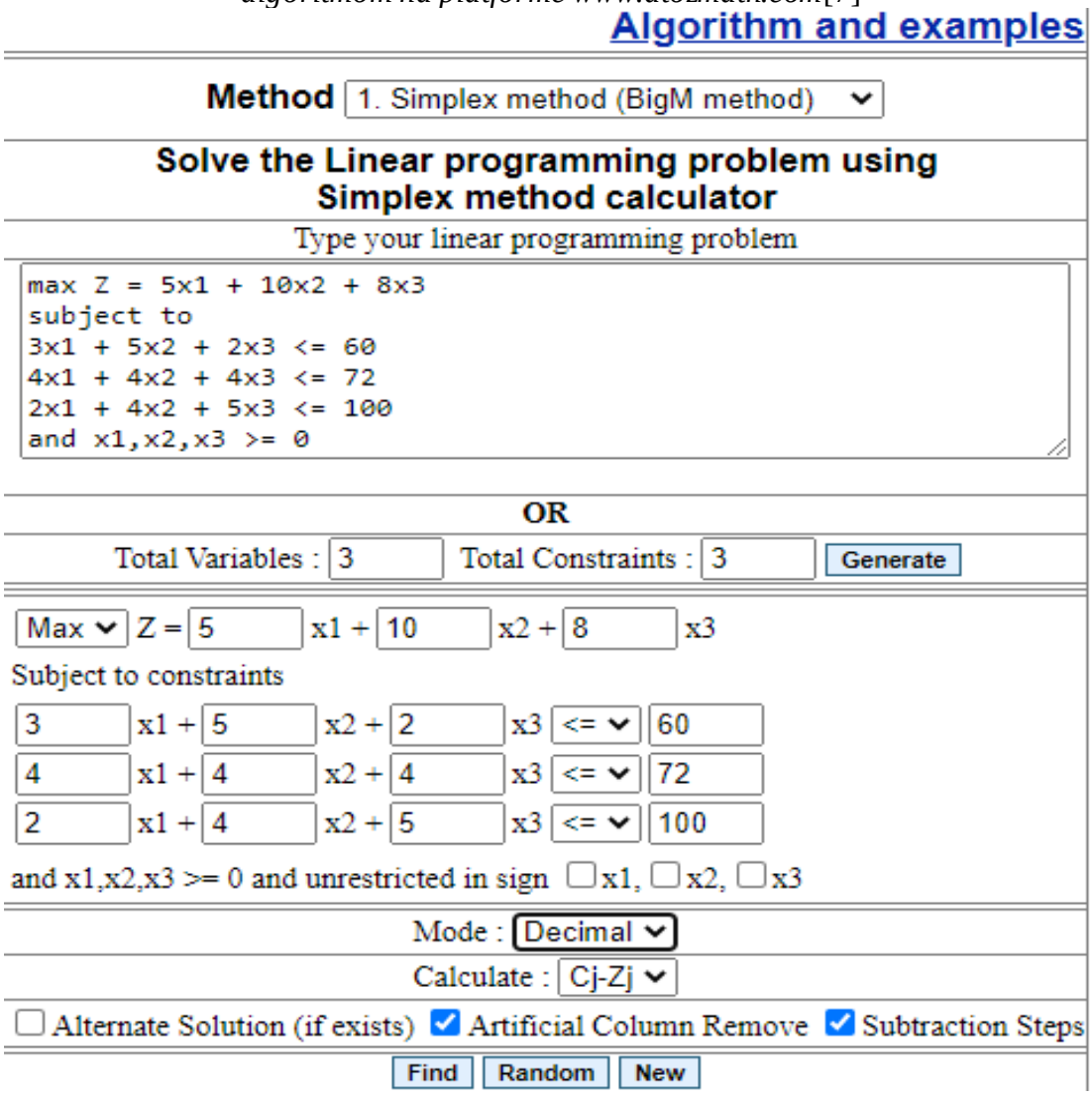
Celkovo je MATLAB veľmi obľúbeným a silným nástrojom na riešenie takmer všetkých typov matematických úloh, avšak je potrebné naučiť sa jeho koncepciu príkazov, poprípade si naprogramovať vlastné knižnice algoritmov. Je tak vhodný nástroj pre vedcov, výskumníkov a vývojárov, poprípade na rýchle overenie správnosti výsledku. Pre začiatočníkov, ktorí si potrebujú osvojiť elementárne kroky algoritmu nebude veľmi vhodný. Navyše používanie MATLAB-u nie je zadarmo.

1.5.2 AtoZmath

AtoZmath je webová stránka, ktorá obsahuje mnoho online „kalkulačiek“. Rieši matematické úlohy krok za krokom podobne ako vo vyučovacom procese v školách. Kalkulačky sú použiteľné na prácu s algebrou, maticou a vektorom, poskytuje riešenie numerickými a štatistickými metódami, taktiež rieši úlohy operačného výskumu, slovné úlohy, geometrické úlohy a ďalšie. Všetky príklady sú riešené krok za krokom spolu s vysvetlením. Mnoho kalkulačiek nie je dostupných na žiadnych iných webových stránkach.[6]

Aj keď je používateľské rozhranie webovej aplikácie na prvý dojem veľmi zastaralé, ponúka veľmi veľa užitočných funkcionalít. Ku každej kalkulačke je veľmi podrobná dokumentácia, rovnako tak aj k typu a postupu riešenej úlohy. Jednotlivé kroky sú jasne oddelené. Na chudobné grafické rozhranie s pár reklamami na stranách sa dá zvyknúť, a v konečnom dôsledku je orientácia na stránke a v jednotlivých algoritmoch prehľadná a jednoduchá. Ako jedna z mála poskytuje riešenie optimalizačných úloh simplexovou metódou spolu s vysvetlením jednotlivých krokov. Ako príklad môžeme uviesť kalkulačku na riešenie úlohy lineárneho programovania simplexovou metódou (obrázok nižšie).

Obrázok č. 4 - kalkulačka na riešenie úlohy lineárneho programovania simplexovým algoritmom na platforme www.atozmath.com[7]



Ako nevýhodu môžeme spomenúť chudobné používateľské rozhranie s reklamami. Pokiaľ algoritmus prechádza viacerými iteráciami, treba po každom kroku kliknúť pre zobrazenie ďalších iterácií. Celkovo platforma AtoZmath poskytuje veľmi veľa kalkulačiek na riešenie širokého spektra matematických problémov spolu s podrobnými vysvetleniami jednotlivých krokov a postupov, ktorú však brzdí chudobné používateľské rozhranie.

1.5.3 Matrix calculator

Maticová kalkulačka je jednoduchá prehľadná webová aplikácia, ktorá poskytuje riešenie úloh primárne s maticami. Taktiež ponúka riešenie sústavy lineárnych rovníc a výpočet determinantu. Na riešenie operácií z maticami používa základné algoritmy ako

Gaussova eliminačná metóda, Gauss-Montane metóda alebo Cramerovo pravidlo. Pri každom type príkladu sa stretávame so stručným, ale jasným popisom úlohy a rovnako tak je to aj pri jednotlivých elementárnych krokoch riešenia. Stránka navyše poskytuje používateľsky príjemné prostredie a zadávanie vstupov, rovnako ako zobrazenie výsledkov, je jasne čitateľné a jednoducho pochopiteľné. Ako príklad uvádzame možnosti operácií s maticou (obrázok nižšie).

Obrázok č. 5 - Operácie s maticami webovej aplikácie Matrix calculator [8]

Matrix calculator je zameraný vyslovene na základné operácie s maticami, preto tu nenájdeme riešenia pre úlohy operačného výskumu alebo lineárneho programovania. Na stránke sa nachádza aj záložka „Potrebná teória“, ktorá je, žiaľ, len odkazom na stránku Wikipédie.

1.5.4 Simplexová metoda – online kalkulátor

Simplexova metoda je veľmi jednoduchá webová aplikácia na riešenie optimalizačných úloh lineárneho programovania. Bola naprogramovaná v jazyku PHP a jej zdrojový kód je dostupný na GitHube. Poskytuje iba riešenie maximalizačnej úlohy lineárneho programovania, ktoré je však veľmi dobre odkrokované. Na začiatku používateľ zadá počet premenných účelovej funkcie a v ďalšom kroku dopĺňa jednotlivé hodnoty premenných do predpripravenej formy. Po stlačení tlačidla „Spočítej“ je vygenerované riešenie v jednotlivých krokoch. Algoritmus prechádza viacerými iteráciami a každá simplexová tabuľka je prehľadne zobrazená spolu s vedúcim riadkom a stĺpcom. Jedná sa

o veľmi jednoduché riešenie, ale za to je veľmi dobre odkrokované, prehľadné a z grafického hľadiska na vyššej úrovni ako pri predchádzajúcich aplikáciách.

Obrázok č. 6 - Zadávanie vstupu vo webovej aplikácií Simplexová metoda[9]

1.6 Použitie programovacieho jazyka C++ na riešenie úloh lineárnej algebry

V tejto podkapitole sa zoznámime s dvomi nástrojmi (knižnicami), ktoré rozširujú použitie jazyka C++ aj do oblasti lineárnej algebry a operačného výskumu, zjednodušujú vývoj aplikácií takéhoto typu a výrazne dokážu urýchliť a optimalizovať fungovanie algoritmu a aplikácie celkovo. Tieto knižnice, žiaľ, nespolupracujú s najnovším nástrojom MS Visual Studio 2022, preto sme sa pri vývoji aplikácie zamerali na riešenie úloh bez týchto knižníc.

1.6.1 Gurobi

Gurobi Optimatization je profesionálny optimalizačný nástroj. Balík programov obsahuje solver, aj modelovacie prostredie. Optimalizačný program je široko implementovateľný do celej rady softvérov a priestorov vrátane C++. Gurobi je použiteľný

na všetkých softvérových platformách: Windows, Linux, MacOS. Okrem úloh lineárneho programovania je softvér schopný riešiť celú radu ďalších úloh matematického programovania - ako celočíselné programovanie a niektoré úlohy nelineárneho programovania. Okrem štandardných metód (primárny a duálny simplexový algoritmus) ponúka niekoľko pokročilejších algoritmov.[10]

1.6.2 Armadillo

Armadillo je vysoko kvalitná knižnica lineárnej algebry pre jazyk C++, ktorej cieľom je dosiahnuť dobrú rovnováhu medzi rýchlosťou a jednoduchosťou použitia. Poskytuje efektívne triedy pre vektory, matice a kocky a ponúka syntax a funkcie na vysokej úrovni podobné MATLAB-u. Ponúka rôzne rozklady matíc (eigen, SVD, QR, atď.) s integráciou LAPACK alebo niektorou z jeho vysokovýkonných náhrad (napr. MKL alebo OpenBLAS). Medzi prednosti patrí aj automatické použitie OpenMP multi-threading (paralelizáciu) na urýchlenie výpočtovo nákladných operácií. Celkovo sa jedná o sofistikovaný nástroj užitočný pre vývoj algoritmov priamo v C++ alebo rýchlu konverziu výskumného kódu do produkčného prostredia. Môže byť použitý na strojové učenie, rozpoznávanie vzorov, počítačové videnie, spracovanie signálov, bioinformatiku, štatistiku, financie atď. [11]

2 Cieľ práce

Cieľom tejto práce je vytvorenie programu v programovacom jazyku C++ , ktorý zostavuje a počíta simplexnú tabuľku elementárnej zmeny bázy n-rozmerného lineámeho priestoru. Interaktívna aplikácia bude slúžiť jednak ako učebná pomôcka pri štúdiu lineárnej algebry, a tiež ako efektívna pomôcka v kontrolnej fáze vyučovacieho procesu. Hlavné požiadavky aplikácie sú jednoduchosť ovládania, názornosť a dobrá čitateľnosť výstupov, ako aj zabezpečenie funkčnosti pre prípad nevhodných vstupov. Taktiež sa budeme snažiť demonštrovať funkčnosť simplexového algoritmu na nájdenie optimálneho riešenia úloh lineárneho programovania.

Ďalší cieľom práce je oboznámiť čitateľa s lineárnou algebrou, resp. jej hlavnou metódou elementárna zmena bázy. Prostredníctvom interaktívnej aplikácie by sme chceli dosiahnuť, aby používateľ lepšie porozumel jednotlivým krokom elementárnej zmeny bázy.

3 Metodika práce a metódy skúmania

Podrobná analýza metódy elementárnej zmeny bázy nám pomohla detailne pochopiť jej funkčnosť, čo nám dokáže pomôcť pri návrhu a vývoji algoritmov aplikácie v jazyku C++. Preštudovaním univerzitných publikácií zameraných na vyučovací proces sme si bližšie osvojili význam lineárnej algebry v bežnom živote, čo nám pomohlo určiť zameranie aplikácie pre širšie praktické použitie.

Analyzovaním existujúcich aplikácií na riešenie úloh lineárnej algebry sme spoznali ich výhody a odhalili slabšie miesta, ktorým by sme sa chceli v našej aplikácii vyvarovať. Väčšina aplikácií používala rôzne postupy na riešenie úloh toho istého typu, bolo tak ťažšie zosúladiť výpočtový proces aplikácie a výpočtovým procesom na univerzite. Tento nedostatok sa pokúsime eliminovať a navrhneme aplikáciu, ktorej postup riešenia úlohy lineárnej algebry sa najviac približuje postupu vyučovaného na Ekonomickej Univerzite v Bratislave.

V ďalšej časti sa budeme venovať návrhu samotnej aplikácie, jej jednotlivých prípadov použitia, návrhu grafického rozhrania, používateľských obrazoviek, jednotlivých módov a funkcií aplikácie, ako aj „príručiek“ k algoritmom, ktoré budú súčasťou aplikácie.

3.1 Návrh aplikácie

Táto kapitola obsahuje postup návrhu samotnej aplikácie. Aplikácia bude vyvíjaná v programovacom jazyku C++ a bude spustiteľná ako samostatná desktopová aplikácia pre operačný systém Windows. Nebude k nej potreba vykonať inštaláciu ďalších doplnkov ani pripojenie na internet. Aplikácia bude mať nastaviteľnú veľkosť hlavnej obrazovky tak, aby sa dala použiť s čo možno najvyšším počtom rôznych zariadení s operačným systémom Windows a zároveň, aby stále ponúkala plnohodnotné používateľské rozhranie.

3.1.1 Požiadavky na aplikáciu

V tejto kapitole bližšie opisujeme jednotlivé funkcionálne a grafické požiadavky aplikácie. Zameriame sa predovšetkým na konkrétne funkčné prvky z pohľadu používateľa aplikácie ako aj na celkový vzhľad.

3.1.1.1 Výpočet úloh lineárnej algebry

Základnou a hlavnou požiadavkou aplikácie je korektný výpočet príkladov lineárnej algebry a lineárneho programovania. Oblasti typov príkladov môžeme rozdeliť do štyroch častí: vektory, matice, systém lineárnych rovníc a úlohy lineárneho programovania. Všetky typy príkladov budú riešené metódou elementárnej zmeny bázy, ktorej funkčnosť musí byť názorne demonštrovaná.

Pri vektoroch sa budeme bližšie zaoberať určovaniu závislosti a nezávislosti systému vektorov, jeho hodnosti, lineárnej kombinácií vektorov a budeme môcť zistiť, či daný vektor je alebo nie je lineárnou kombináciou vektorov systému. Medzi operácie s maticami zaradíme výpočet hodnosti matice, rozklad matice na súčin matíc, výpočet inverznej matice a aj výpočet jej determinantu. Pri systéme lineárnych rovníc budeme uvažovať len nájdenie riešenia, resp. riešení.

Z lineárneho programovania vyberieme iba primárny simplexový algoritmus na nájdenie extrému účelovej funkcie pri ohraničujúcich podmienkach, keďže lineárne programovanie nie je priamo témou tejto práce. Dôležitou myšlienkou je poukázať na využitie metódy elementárnej zmeny bázy aj v lineárnom programovaní, resp. v operačnom výskume.

3.1.1.2 Interaktívnosť aplikácie

Pre posilnenie edukatívneho charakteru aplikácie je nutné, aby bola aplikácia interaktívna. Počnúc uvítacím, resp. informačným oknom pri spustení aplikácie až po možnosť voľby vedúcich prvkov pri elementárnej zmene bázy. Hlavnou podmienkou výpočtu je aj korektné spracovanie vstupov, teda aplikácia musí nevhodné vstupy odfiltrovať, resp. upozorniť používateľa na chybu pri ich zadávaní. Používateľ bude mať možnosť kedykoľvek výpočet prerušiť, spustiť znova, zmeniť vstupy alebo úlohu zmazať (viac v kapitole Prípady použitia).

3.1.1.3 Jednoduchosť ovládania

S interaktívnosťou je spojená aj jednoduchosť ovládania. Pri celkom rozsiahlom obsahu hlavnej obrazovky (tlačidlá, zobrazené vstupy, výstupy, zadanie...) je dôležité upriamiť pozornosť používateľa na to najpodstatnejšie: v našom prípade tabuľka elementárnej zmeny bázy. Preto je vhodné hlavné okno rozdeliť do štyroch logických

celkov: vstupy, zadanie úlohy, postup riešenia (tabuľka elementárnej zmeny bázy) a výstupy. Ovládanie prvky (tlačidlá) by mali byť umiestnené nerušivo vo vrchnej časti.

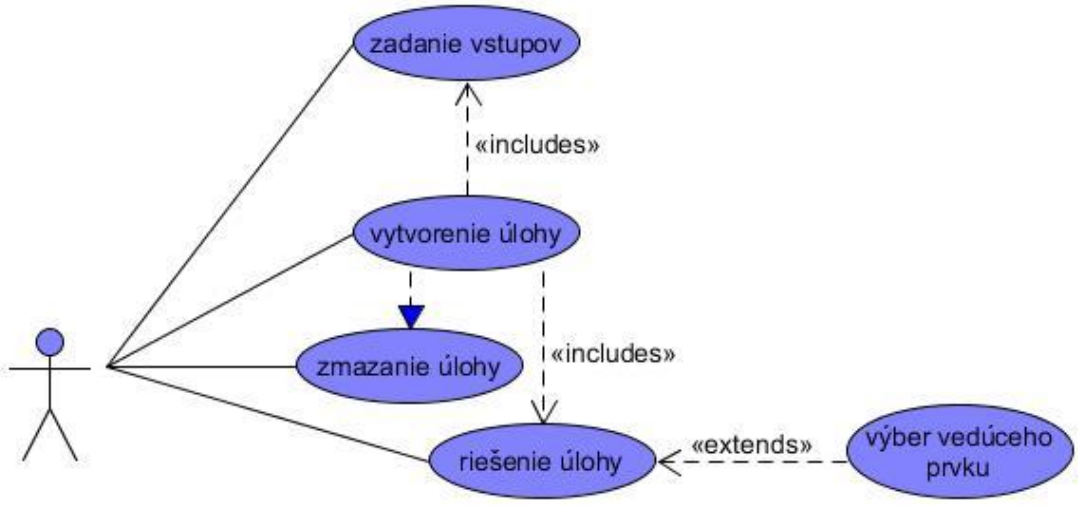
Základom je upútať pozornosť používateľa uvítacím oknom, v ktorom sú zobrazené aj základné inštrukcie k používaniu aplikácie. Pri implementácií budeme klásť dôraz na to, aby bol dizajn hlavného okna ihneď po spustení aplikácie čo najčistejší a s používateľského hľadiska intuitívny. Jednotlivé formuláre na zadávanie vstupov musia byť jednoducho pochopiteľné. Pri zadávaní väčšieho počtu vstupov je vhodné pre urýchlenie procesu umožniť použitie klávesových skratiek. Výber vedúceho prvku z tabuľky by mal byť pre jednoduchosť umožnený na kliknutie myšou.

Farebná neutrálnosť prvkov, ktoré chceme skryť, v kombinácií s farebne výraznejšími prvkami (hlavné ovládacie prvky) prinesie svieži, moderný a používateľsky príjemný vzhľad. Samozrejmosťou budú pop-up okná pre najdôležitejšie upozornenia, resp. varovania a „pomocníky“ na pomoc pri ovládaní aplikácie.

3.1.2 Prípady použitia

V tejto podkapitole si bližšie popíšeme jednotlivé prípady použitia, ktoré boli identifikované pri formulácií požiadaviek aplikácie. Nasledujúci obrázok zobrazuje všetky prípady použitia.

Obrázok č. 7 - prípady použitia aplikácie



3.1.2.1 Vytvorenie úlohy

Základnou podmienkou pre riešenie úlohy je jej korektné vytvorenie. Používateľovi sa po kliknutí na tlačidlo zobrazí formulár, ktorý musí korektne vyplniť. V prípade, že sú vstupy nesprávne, resp. chýbajú, aplikácia zobrazí pop-up okno a upozorní používateľa. Ak sú všetky vstupy v správnom formáte, nová úloha je vytvorená.

3.1.2.2 Zadanie vstupov

Zadanie vstupov je jednou z elementárnych činností pri vytváraní novej úlohy. Používateľ bude môcť zadať iba povolené hodnoty. Medzi jednotlivými políčkami sa bude pre urýchlenie procesu zadávania vstupov premiestňovať tabulátorom alebo inou vhodnou metódou. Vo formulári bude takisto možnosť na zmazanie všetkých zadaných vstupov.

3.1.2.3 Zmazanie úlohy

Používateľ bude môcť riešenú úlohu zmazať, či už priebežne, počas alebo po skončení riešenia. Vstupy úlohy sa budú dať aj upraviť, ale v konečnom dôsledku je to len kombinácia odstránenia a vytvorenia novej úlohy.

3.1.2.4 Riešenie úlohy

Po vytvorení úlohy je možné ju riešiť. Najprv aplikácia overí, či je možné prejsť na ďalšiu iteráciu riešenia a v prípade, že áno, je nutné vybrať vedúci prvok. V opačnom prípade výpočet končí.

3.1.2.5 Výber vedúceho prvku

Výber vedúceho prvku je nevyhnutný pokiaľ chceme vykonať ďalšiu iteráciu riešenia úlohy. Vedúci prvok vyberie používateľ z tabuľky elementárnej zmeny bázy dvojklikom myšou na bunku podľa príslušných pravidiel, resp. odporúčaní. Pokiaľ zvolí používateľ vedúci prvok s hodnotou 0, zobrazí sa pop-up okno s upozornením, že vedúci prvok nemôže byť 0.

3.1.3 Postup riešenia úloh lineárnej algebry

Po vytvorení novej úlohy sa do príslušných okien zapíšu parametre úlohy, úlohy pre nájdenie riešenia. Zo vstupov sa vytvorí tabuľka elementárnej zmeny bázy, ktorá je

najprv vykreslená na obrazovku a následne sa, v závislosti od typu úlohy, spustí kontrola riešiteľnosti.

Pokiaľ je úloha ďalej neriešiteľná, tabuľka elementárnej zmeny bázy sa zablokuje a nebude možné zvoliť vedúci prvok. Následne sa vypíše „riešenie“, čo je, v tomto prípade, informácia o neriešiteľnosti zadanej úlohy. V opačnom prípade je úloha riešiteľná a používateľ môže zvoliť vedúci prvok dvojklikom v tabuľke elementárnej zmeny bázy (viac v kapitole Výber vedúceho prvku).

Po výbere vedúceho prvku sa uskutoční elementárna zmena bázy a nová časť bázickej tabuľky sa pripojí ku pôvodnej tabuľke. Opäť je vykonaná kontrola ďalšej riešiteľnosti, resp. kontrola, či už sme našli riešenie.

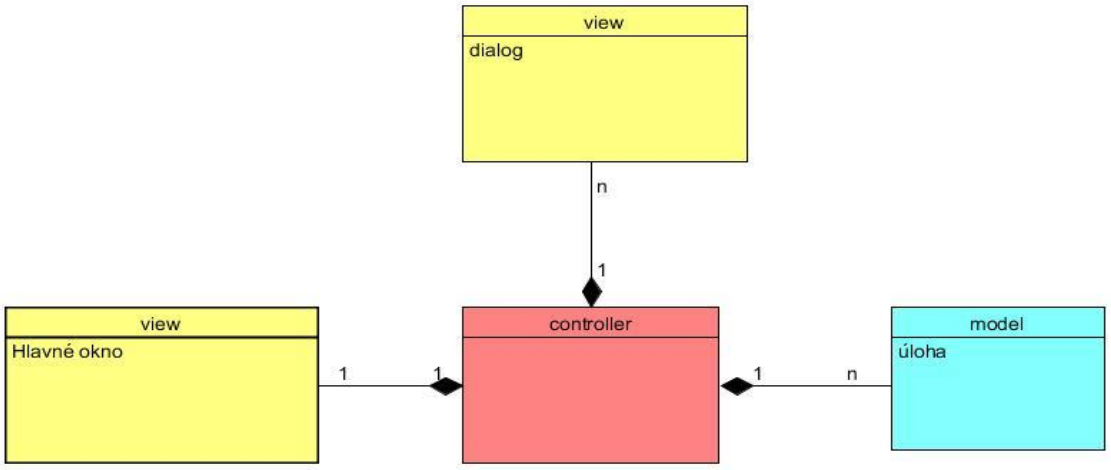
Pokiaľ sme vykonali maximálny možný počet elementárnych zmien bázy, vypíše sa riešenie, resp. záver úlohy podľa bodov (a), b) , ...) zo zadania. Používateľ už nemôže ďalej zvoliť vedúci prvok.

Počas celého cyklu má používateľ možnosť úlohu zmazať, upraviť jej vstupné parametre alebo začať odznovu s pôvodnými parametrami.

3.1.4 Základný model aplikácie

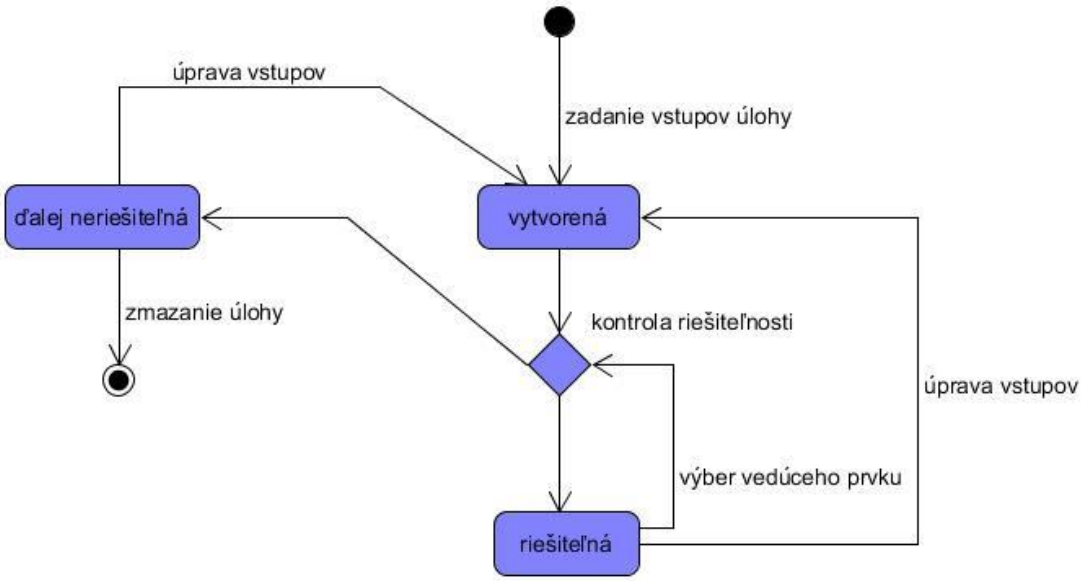
Nasledujúci obrázok zobrazuje jednoduchý model aplikácie formou diagramu. Model mvc (model view controller) pozostáva v tomto prípade z používateľských okien a dialógov(view), tried metód a funkcií (controller) a samotných dát (model). Z hlavného používateľského okna je možné napríklad zobraziť (vytvoriť) dialógové okno pre vytvorenie novej úlohy. Z používateľského rozhrania pristupujeme k dátam (dáta riešenej úlohy) pomocou metód.

Obrázok č. 8 - základný model aplikácie (mvc)



Z návrhu aplikácie vyplýva, že naším hlavným objektom bude úloha. Úloha bude, v ideálnom prípade, na začiatku vytvorená, potom riešená a nakoniec buď upravená alebo zmazaná. Úlohy môžeme ďalej kategorizovať podľa ich typu: vektory, matice, systém lineárnych rovníc alebo úloha lineárneho programovania. Nasledujúci obrázok zobrazuje stavový diagram objektu úloha, resp. jej životný cyklus.

Obrázok č. 9 - Stavový diagram objektu „úloha“



4 Výsledky práce

V tejto kapitole sa bližšie zoznámime s výslednou aplikáciou a overíme, či spĺňa požiadavky definované v návrhu. Prvá časť je venovaná implementácií aplikácie. Popisujeme tu použité postupy implementácie a najmä postup vývoja hlavnej funkcionality: výpočet úloh použitím elementárnej zmeny bázy. V druhej časti sa zoznámime s používateľským rozhraním a následne opíšeme jednotlivé funkcie aplikácie ako postupy prípadov použitia. Táto kapitola zároveň slúži aj ako návod na používanie.

4.1 Implementácia aplikácie

Na implementáciu aplikácie sme zvolili programovací jazyk C++. Aplikáciu sme pomocou nástroja Microsoft Visual Studio Community 2022. Tento nástroj sme si vybrali, pretože poskytuje prehľadné implementačné prostredie a spoľahlivý „debugger“. Výhodou je možnosť otvorenia a editácie kódov v rôznych programovacích jazykoch. V tomto projekte sme použili verziu programu 17.1.3.

Celú implementáciu sme realizovali na základe návrhu a v tejto časti sa budeme explicitne odkazovať na časti návrhu.

4.1.1 Výpočet úloh lineárnej algebry a lineárneho programovania

Výpočet úloh lineárnej algebry a lineárneho programovania predstavuje hlavnú funkcionalitu vytvorenej aplikácie. V nasledujúcich podkapitolách si detailne popíšeme ako prebiehajú elementárne aj komplexné výpočty spolu s ukážkami zdrojového kódu.

4.1.1.1 Vytvorenie novej bázickej tabuľky

```
double** bazickaMatica = 0;
bazickaMatica = new double* [mt->pocetRiadkov];
for (int h = 0; h < mt->pocetRiadkov; h++){
   bazickaMatica[h] = new double[mt->pocetStlpcov + 1];
}
```

Matica `bazickaMatica` predstavuje východiskovú bázu B_0. Jeden stĺpec navyše je určený pre kontrolnú sumu. Následne získame vstupné hodnoty z dialógového okna, ktoré slúži na zadanie parametrov novej úlohy. Hodnoty zapíšeme aj do bázickej matice, aj do tabuľky elementárnej zmeny bázy:

```
for (int j = 0; j < mt->pocetRiadkov; j++) {
   double suma = 0;
   for (int i = 0; i < mt->pocetStlpcov; i++) {
      suma += mt->matrix[j][i];
      ezbTable[i+1, j]->Value = round_up(mt->matrix[j][i], 2);
      bazickaMatica[j][i] = mt->matrix[j][i];
   }
   ezbTable[PS + 1, j]->Value = round_up(suma, 2);
   bazickaMatica[j][ mt->pocetStlpcov] = suma;
}
```

V ďalšom kroku voláme funkciu, ktorá určí, či je úloha ďalej riešiteľná:

ezb->checkMatrix(mt->pocetStlpcov);

4.1.1.2 Určenie riešiteľnosti úlohy lineárnej algebry

Úloha lineárnej algebry nie je riešiteľná v prípade, že už sú v báze začlenené všetky stĺpcové vektory, ak nie je voľné miesto v báze na začlenenie ďalších stĺpcových vektorov alebo ak máme v báze príliš veľa nulových riadkov (nie je voľné miesto v báze na začlenenie ďalších stĺpcových vektorov):

```
if (height < width - vB - 1) {
   if (countZaclenenych == height) {
      return 1;
   }
}
else {
   if (countZaclenenych == width - vB - 1) {
      return 1;
   }
}

if (countNulovych >= height - countZaclenenych) {
   return 0;
}
```

V opačnom prípade je úloha lineárnej algebry riešiteľná a používateľ môže zvoliť vedúci prvok.

4.1.1.3 Určenie riešiteľnosti úlohy lineárneho programovania

Ako prvé zisťujeme, či je množina prípustných riešení ohraničená tak, že sa snažíme dopredu nájsť vhodný vedúci prvok (vždy musíme určiť práve jeden). Ak na základe teoretických východísk nedokážeme určiť vedúci prvok, úloha nie je ďalej riešiteľná. Pre rozsiahlosť príslušného zdrojového kódu ho tu nebudeme uvádzať. Ďalšou podmienkou

je podmienka primárnej prípustnosti, ktorú určíme jednoducho kontrolou posledného stĺpca v matici simplexovej tabuľky:

```
for (int i = 0; i < lpt->pocetOhraniceni; i++)
   if (m[i][lpt->pocetOhraniceni + lpt->pocetPremennych] < 0)
      nepripustne = true;
```

Posledným krokom je určiť, či aktuálne riešenie je optimálnym riešením. Na základe typu extremalizačnej funkcie kontrolujeme riadok $c_j - z_j$:

```
if (lpNewTaskD->getMaxMin()) {
   for(int i=1; i <= lpt->pocetPremennych + lpt->pocetOhraniceni; i++) {
      if (System::Convert::ToDouble(ezbTable[i, ezbTable->RowCount - 1]-
>Value) > 0)
         optimalne = false;
   }
}
```

Pokiaľ riešenie optimálne nie je a úloha lineárneho programovania je ďalej riešiteľná, postupujeme voľbou vedúceho prvku.

4.1.1.4 Voľba vedúceho prvku

Ak sa po jednoduchom kliknutí na bunku v tabuľke bunka označí (modrou farbou), tento prvok je možné zvoliť ako vedúci. Potvrdenie voľby vedúceho prvku uskutočníme dvojklikom na bunku tabuľky. V tabuľke elementárnej zmeny bázy je možné kliknúť len na hodnoty bázickej matice okrem stĺpca vektora $\bar{b}$:

ezbTable->CurrentCell->OwningColumn->Name == "b\u20D7"

Navyše, nie je možné začleniť vektory, ktoré už boli začlenené do bázy. V prípade riešenia úlohy s inverznou maticou, nie je možné kliknúť na zložky doplnkových stĺpcov. Ak riešime úlohu lineárneho programovania, tak vhodný vedúci prvok je vybraný programovo dopredu a používateľ nemôže zvoliť iný. Po úspešnom zvolení vedúceho prvku pokračujeme elementárnou zmenou bázy.

4.1.1.5 Elementárna zmena bázy

Ide o jednoduchú procedúru, kedy sa prepočítaním aktuálnej bázy za pomoci vedúceho prvku vytvorí nová báza. Táto procedúra je pre všetky riešené úlohy rovnaká:

```
for (int j = 0; j < width; j++) {
   for (int i = 0; i < height; i++) {
      if (j == pivotXColumn) {
         if (i == pivotYRow) {
            newMatrix[i][j] = 1;
         }
         else {
            newMatrix[i][j] = 0;
         }
      }
      else {
         if (i == pivotYRow) {
            newMatrix[i][j] = oldMatrix[i][j] / pivot;
         }
         else {
            double sigma = oldMatrix[pivotYRow][j] / pivot;
            newMatrix[i][j] = oldMatrix[i][j] - (sigma *
oldMatrix[i][pivotXColumn]);
         }
      }
   }
}
```

Na konci funkcia vráti novú bázickú maticu: `return newMatrix;`

4.1.1.6 Určenie výsledku úlohy a formulácia záveru

Ak sme určili, že úloha nie je ďalej riešiteľná, je potrebné určiť výsledok. Opierajúc sa o teoretické znalosti lineárnej algebry, resp. lineárneho programovania, postupne z tabuľky elementárnej zmeny bázy určujeme výsledky korešpondujúce so zadaním a formulujeme závery, ktoré zobrazíme používateľovi na obrazovke. Ako príklad uvedieme časť kódu, ktorá určuje, či je vektor $\bar{b}$ lineárnou kombináciou bázických vektorov:

```
for (int i = 0; i < pocetSuradnic; i++) {
   int count = 0;
   for (int j = 1; j <= pocetVektorov - vectorB; j++)
      if (m[i][j - 1] != 0) count++;

   if (count == 0 && m[i][pocetVektorov] != 0) {
      output += "c) Vektor b\u20D7 nie je lineárnou kombináciou vektorov
bázy { " + zaclenene + " }, pretože zložka b" + subscript((i +
1).ToString()) + " \u2260 0.\r\n";
      return output;
   }
}
```

4.1.1.7 Ďalšie dôležité funkcie

Medzi dôležitú funkciu patrí zaokrúhľovanie. Všetky výpočty sa realizujú s nezaokrúhlenými hodnotami, ale pri výpise sa používa zaokrúhlený tvar na dve desatinné miesta.

```
double round_up(double value, int decimal_places) {
   const double multiplier = std::pow(10.0, decimal_places);
   return std::ceil(value * multiplier) / multiplier;
}
```

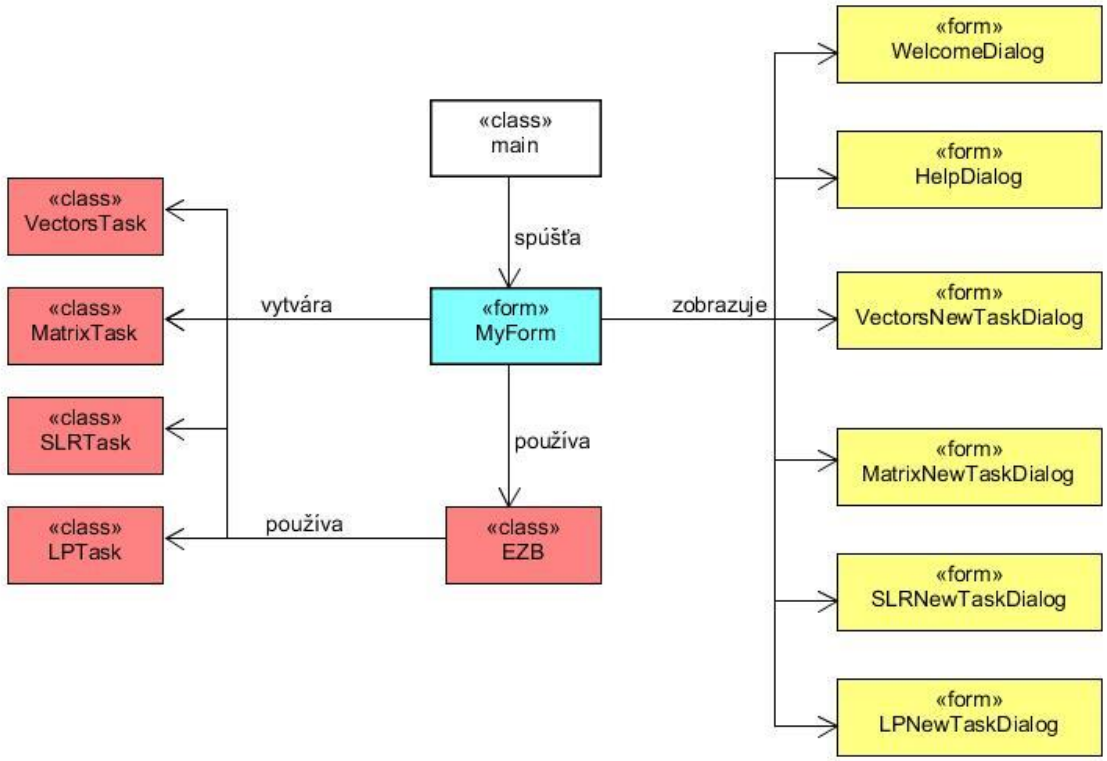
Zaujímavou funkciou je aj prepis inverznej matice z poslednej časti tabuľky do výstupu.

```
for (int j = 0; j < pocetStlpcov; j++) {
   for (int i = 0; i < pocetRiadkov; i++) {
      if (m[i][j] == 1) {
         for (int k = 0; k < pocetStlpcov * 2; k++) {
            if (k >= pocetStlpcov) {
               matrixB[count][k - pocetStlpcov] = round_up(m[i][k], 2);
            }
         }
         count++;
      }
   }
}
```

4.1.2 Komponenty aplikácie

V tejto časti sa zoznámime s jednotlivými súčasťami aplikácie a opíšeme si niektoré triedy. Nasledujúci obrázok reprezentuje diagram tried aplikácie. Červenou farbou sú označené triedy typu „class“ , žltou farbou sú označené dialógové okná a trieda zafarbená na modro predstavuje hlavné používateľské okno, ktoré je spúšťané z triedy *main*.

Obrázok č. 10 - diagram tried aplikácie



Trieda *MyForm* predstavuje hlavné okno používateľského rozhrania. V prípade vyžiadania vytvorenia novej úlohy, zobrazí jedno z dialógových okien. Triedu *EZB* používa ako „controller“ na riešenie elementárnej zmeny bázy a na určovanie riešiteľnosti úlohy. Dialógové okná, ako napríklad *LPNewTaskDialog*, ponúkajú možnosť zadať vstupné parametre novej úlohy. Triedy typu „class“, napr. *LPTask*, udržiavajú dáta riešenej úlohy. *HelpDialog* a *WelcomeDialog* slúžia ako doplnkové okná v aplikácii.

4.1.3 Testovanie aplikácie

Testovanie aplikácie sme rozdelili na dve hlavné časti: testovanie správnosti výpočtu a testovanie grafického (používateľského) rozhrania. Pri testovaní správnej funkčnosti algoritmov sme postupovali rovnakým logickým postupom ako sme opísali v kapitole *Výpočet úloh lineárnej algebry a lineárneho programovania*. Najprv sa otestovala každá elementárna funkcia a až následne sa elementárne funkcie spojili do jedného algoritmu. Takýmto spôsobom sme pri testovaní ušetrili množstvo času. Na testovanie správnej funkčnosti algoritmov sme používali predovšetkým vstupy z knihy *Lineárna algebra pre ekonómov*[3], odkiaľ sme zároveň čerpali inšpiráciu pre výpisy riešení. Keď aplikácia fungovala na jednoduchých vstupoch, prišlo na rad testovanie s nevhodnými vstupmi. Hlavné problémy sa týkali počítania a zápisu čísiel s veľkou hodnotou za desatinnou čiarkou. Problém z veľkej časti odstránilo vyššie spomenuté zaokrúhľovanie.

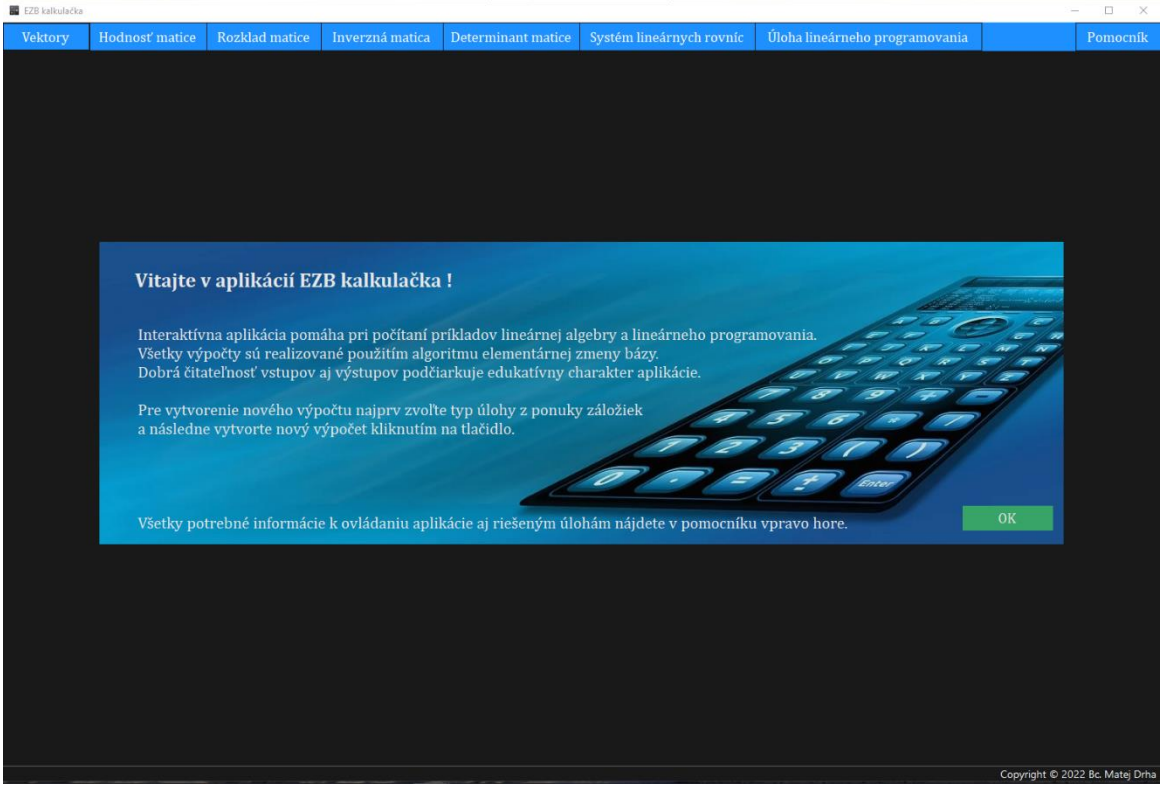
Testovanie funkčnosti používateľského rozhrania bolo taktiež závislé od parametrov riešenej úlohy. Ako príklad opäť uvedieme problémy s desatinnými číslami. Počas celého vývoja aplikácie sme neustále testovali používateľské rozhranie, pretože postupným začleňovaním nových komponentov, ako aj výpočtových rozšírení, sa objavovali nové problémy.

Po finálnej úprave aplikácie sme ju odprezentovali niekoľkým študentom a zároveň nechali aj otestovať. V poslednej verzii sa nám nepodarilo nájsť žiadne výpočtové ani grafické chyby, ktoré by vykazovali chybovosť aplikácie.

4.2 Požívateľské rozhranie

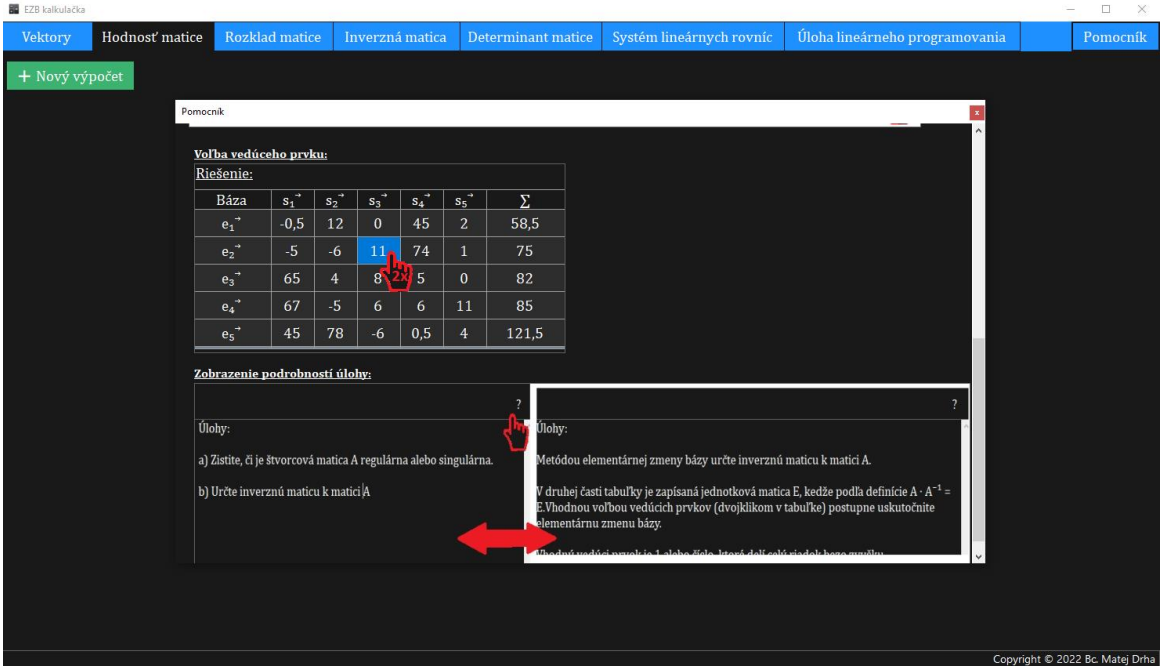
Po spustení aplikácie je zobrazená hlavná obrazovka, ktorá je prekrytá uvítacím oknom. Uvítacie okno má slúžiť aj ako informácia pre nových používateľov, že všetko, čo potrebujú, nájdu v pomocníku.

Obrázok č. 11 - Hlavná obrazovka spolu s uvítacím oknom



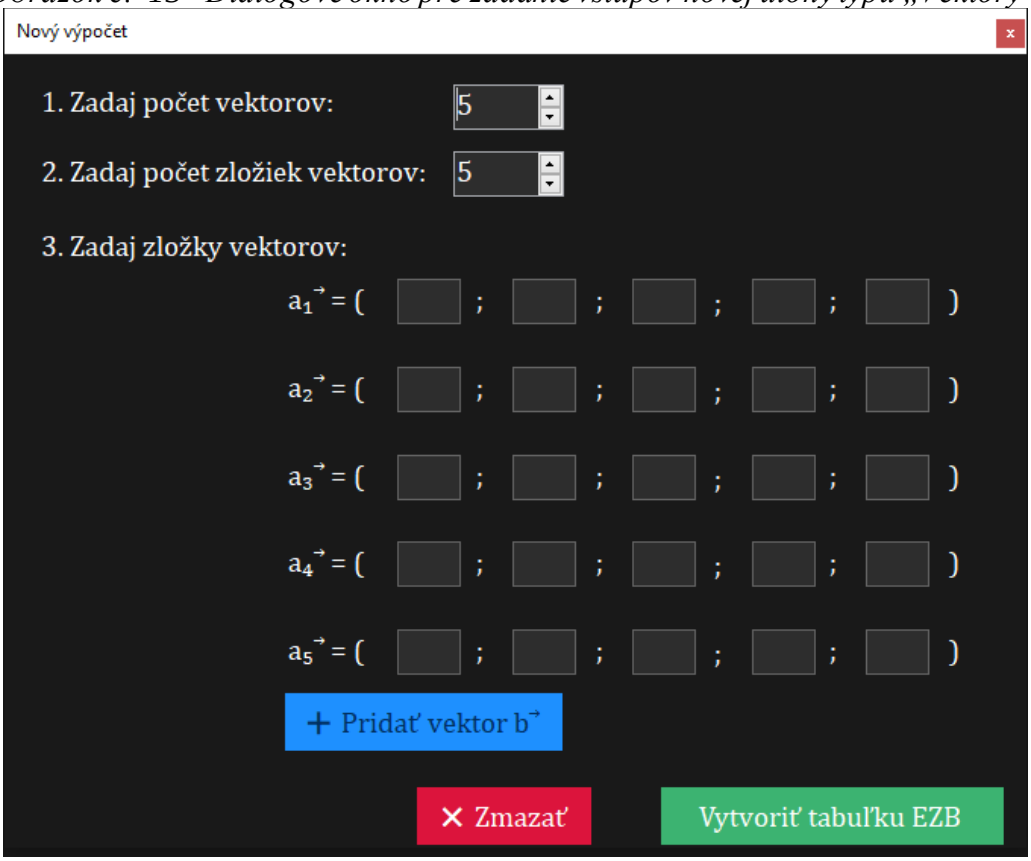
Po voľbe typu úlohy sa zobrazí možnosť vytvoriť nový výpočet (novú úlohu). Kliknutím na tlačidlo *Pomocník* sa zobrazí okno pomocníka.

Obrázok č. 12 - - Hlavná obrazovka spolu s obrazovkou pomocníka

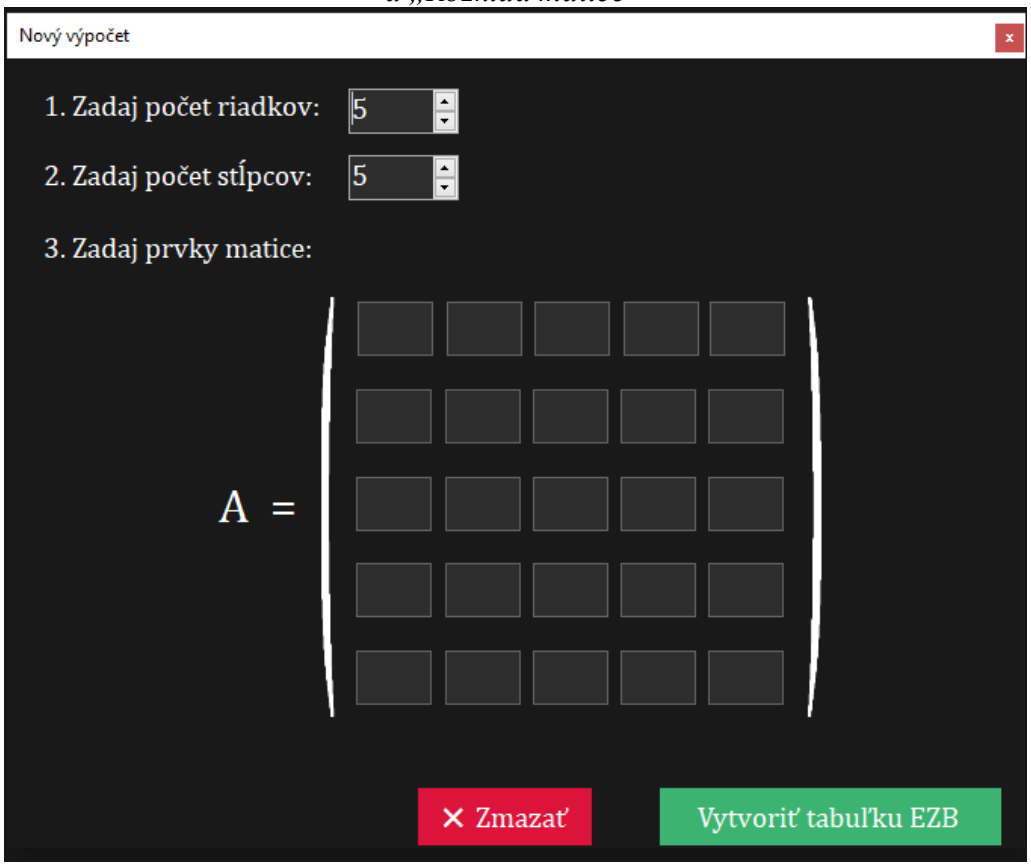


Kliknutím na tlačidlo + *Nový výpočet* sa zobrazí jedno z piatich dialógových okien, kde je možnosť vyplniť vstupné parametre úlohy.

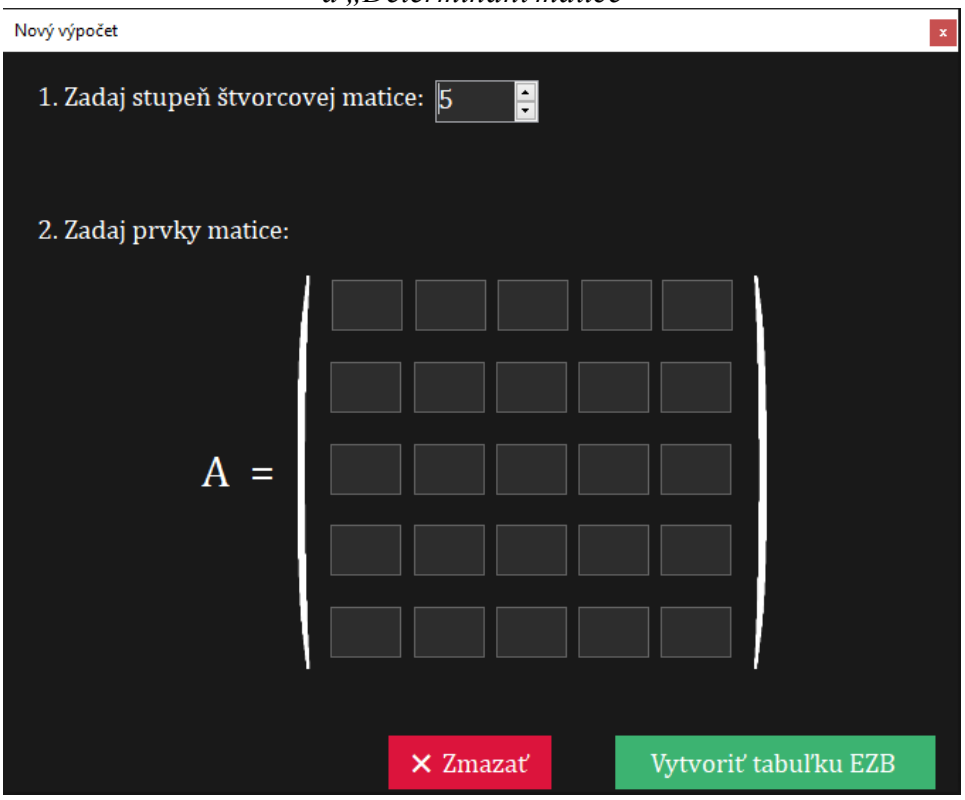
Obrázok č. 13 - Dialógové okno pre zadanie vstupov novej úlohy typu „Vektory“



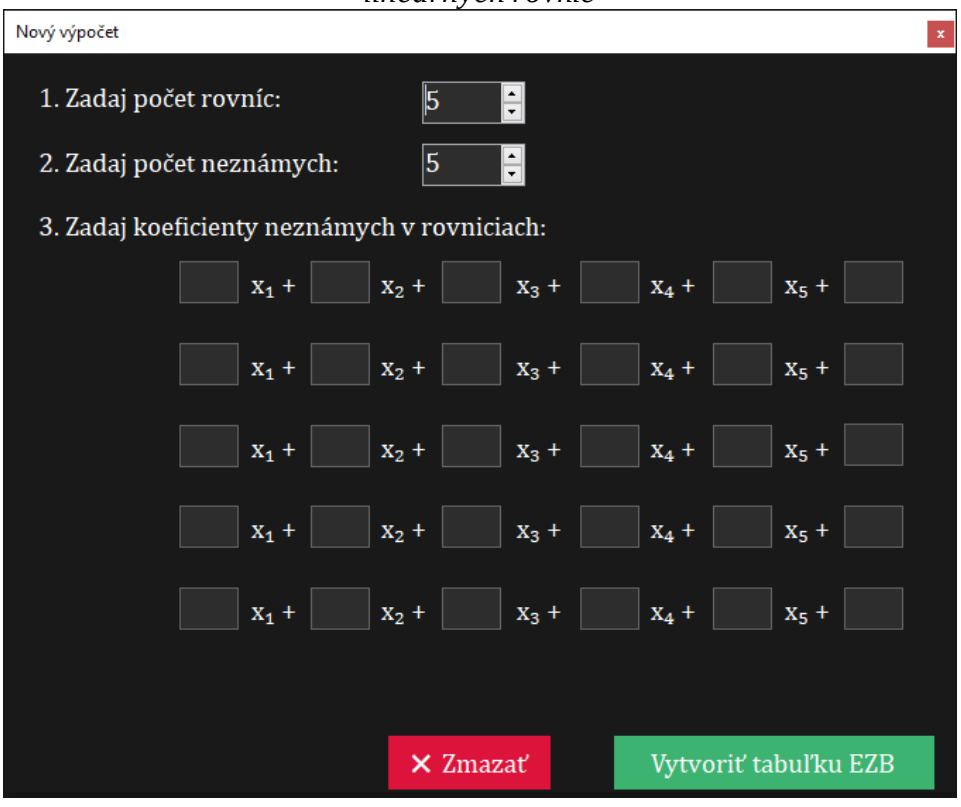
Obrázok č. 14 - Dialógové okno pre zadanie vstupov novej úlohy typu „Hodnosť matice“ a „Rozklad matice“



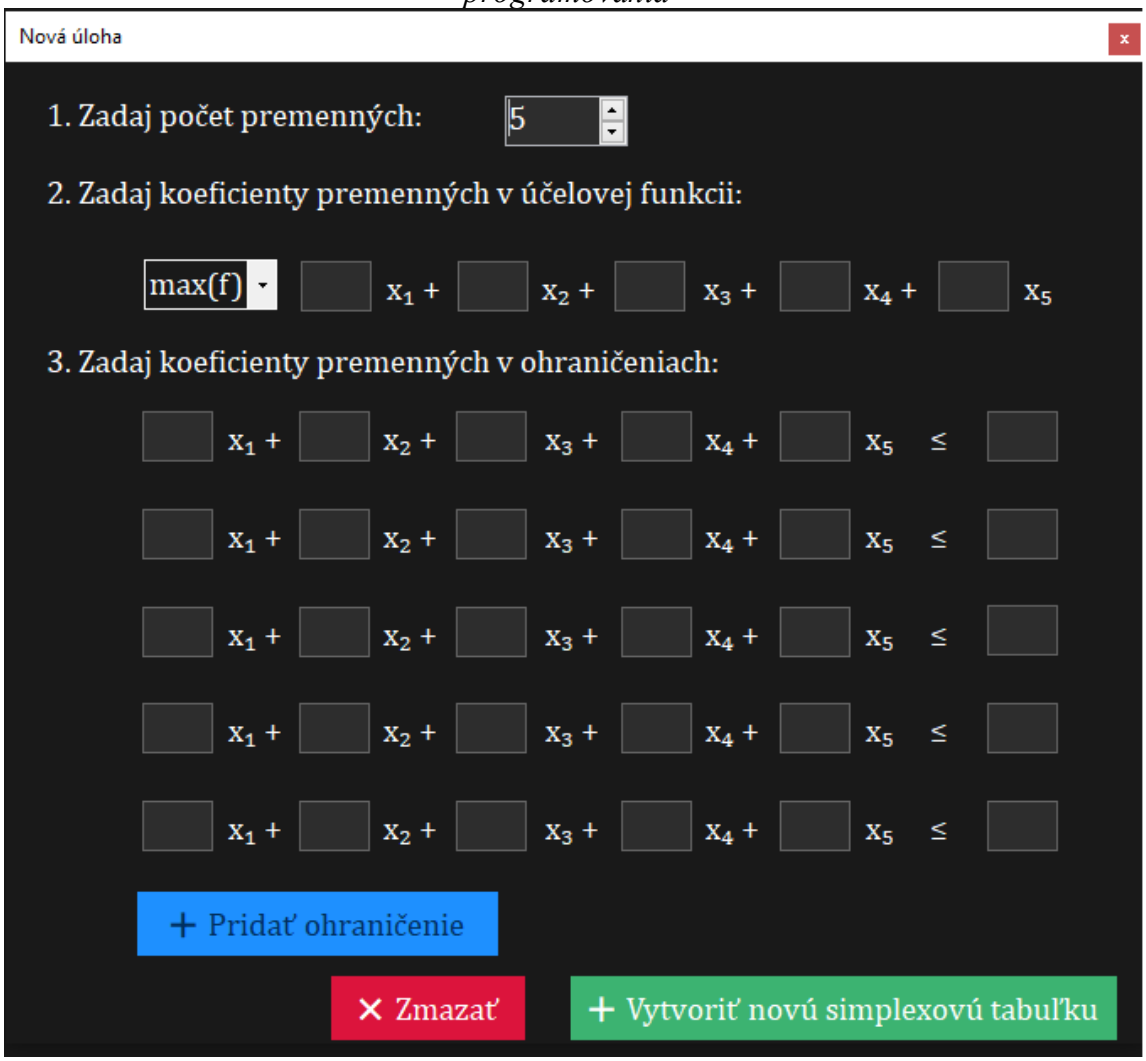
Obrázok č. 15 - Dialógové okno pre zadanie vstupov novej úlohy typu „Inverzná matica“ a „Determinant matice“



Obrázok č. 16 - Dialógové okno pre zadanie vstupov novej úlohy typu „Systém lineárnych rovníc



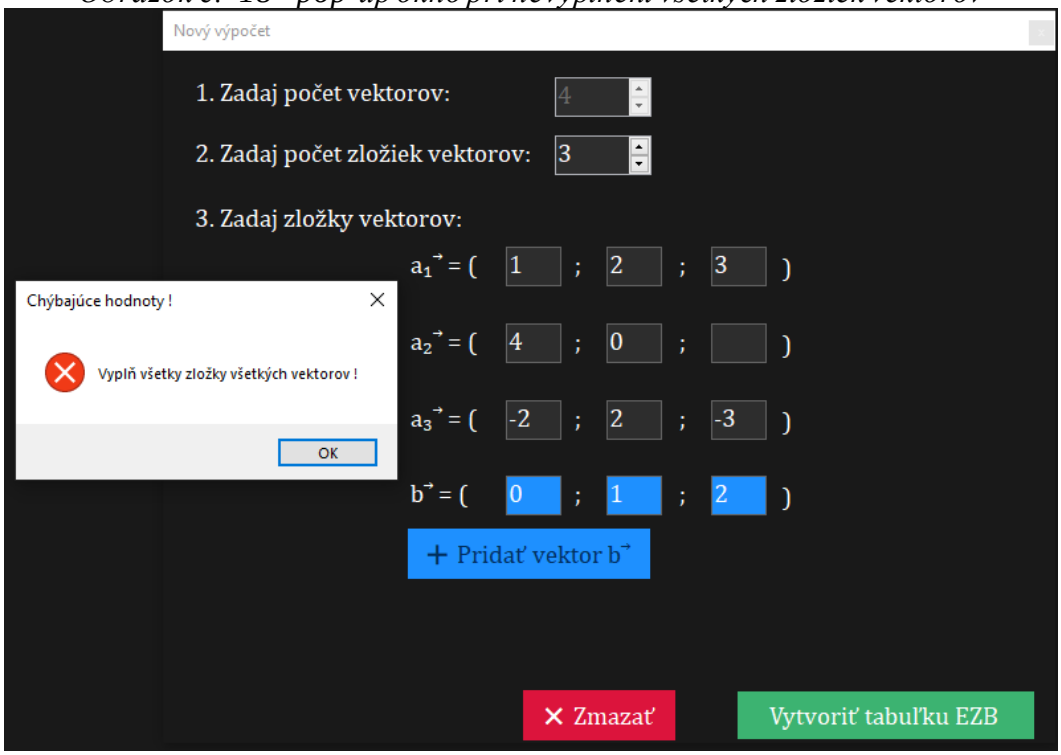
Obrázok č. 17 - Dialógové okno pre zadanie vstupov novej úlohy typu „Úloha lineárneho programovania“



4.3 Možnosti používania aplikácie

Medzi všeobecné možnosti používania patrí zadávanie vstupov novej úlohy, kde sú iba malé odlišnosti naprieč typmi úloh. Pri každom vypĺňaní vstupných parametrov platí, že všetky zobrazené políčka musia byť vyplnené, v opačnom prípade sa zobrazí varovanie a nebude možné vytvoriť novú úlohu (viď obrázok nižšie). Pre typ úlohy „Vektory“ je možné zadať vektor $\bar{b}$.

Obrázok č. 18 - pop-up okno pri nevyplnení všetkých zložiek vektorov



Pre typy úloh „Inverzná matica“ a „Determinant matice“ je základným predpokladom riešiteľnosti úlohy štvorcová matica, preto používateľ vie zadať iba jeden rozmer (viď obrázok vyššie). Pri type úlohy „Úloha lineárneho programovania“ musí používateľ zvoliť typ extremalizácie účelovej funkcie, pričom je v základe nastavený na *max(h)*.

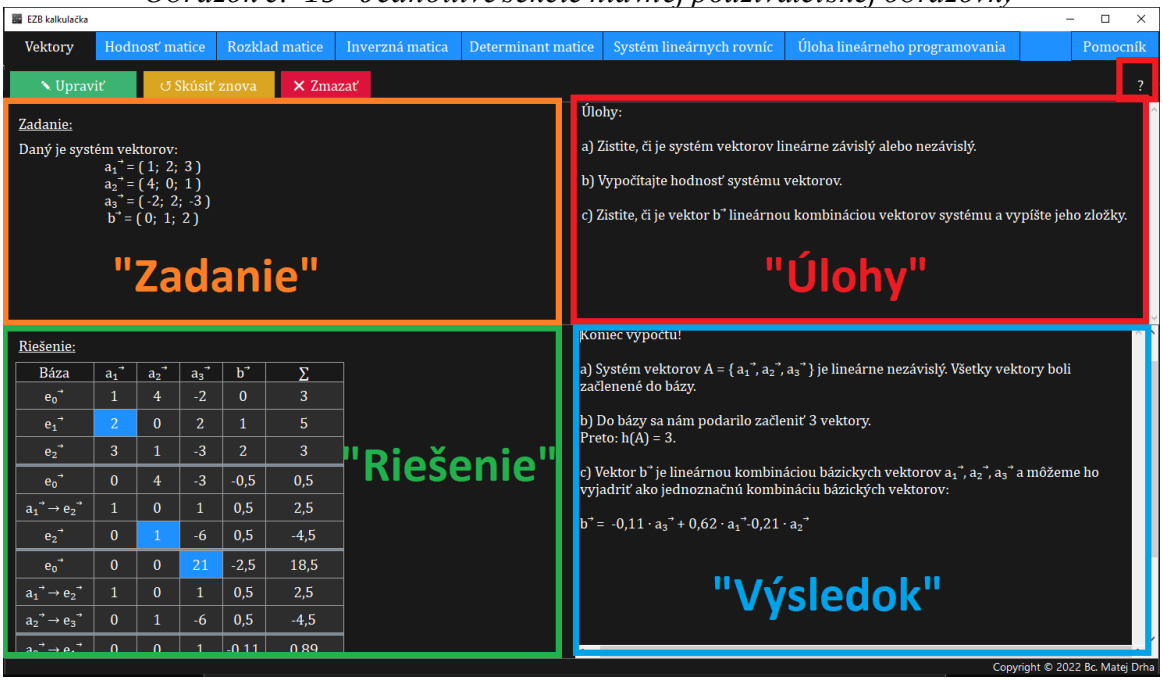
Rovnaké príslušné dialógové okná sa zobrazia aj po kliknutí na tlačidlo *„Skúsiť znova“*(tlačidlo v hlavnej obrazovke), ale už s pred-vyplnenými hodnotami z posledného vytvorenia novej úlohy. Tlačidlá v spodnej časti dialógových okien svojím textom jasne zdôrazňujú svoju funkciu.

Po kliknutí na tlačidlo „Vytvoriť tabuľku EZB“ sa dialógové okno zatvorí a používateľovi sa zobrazí hlavné okno s už zobrazenými súčasťami obrazovky. Súčasťami obrazovky rozumieme sekcie, na ktoré je obrazovka rozdelená a každá sekcia zobrazuje používateľovi informácie o zadanej úlohe. Tlačidlá v hornej časti svojím textom jasne definujú svoju funkciu.

Vrchná časť je rozdelená na „Zadanie“, v ktorom sú uvedené hodnoty vstupov úlohy a „Úlohy“, ktoré špecifikujú úlohy pre používateľa, resp. pre používanie aplikácie v ďalších krokoch. Stlačením otáznika v pravom hornom rohu sa zobrazia úlohy v rozšírenom formáte; majú pomôcť používateľovi pochopiť riešenú úlohu.

Spodná časť obrazovky je rozdelená na „Riešenie“, v ktorom sa s pomocou používateľa uskutočňuje elementárna zmena bázy. Pre ďalšiu iteráciu elementárnej zmeny bázy musí používateľ (dvojklikom v tabuľke) vybrať vedúci prvok, na základe jeho teoretických znalostí lineárnej algebry alebo „pomôcky“ v aplikácii. Pravá časť „Výsledok“ zobrazuje (priebežné) výsledky úlohy v textovom, resp. grafickom formáte.

Obrázok č. 19 - Jednotlivé sekcie hlavnej používateľskej obrazovky



Báza	a1→	a2→	a3→	b→	Σ
e0→	1	4	-2	0	3
e1→	2	0	2	1	5
e2→	3	1	-3	2	3
e0→	0	4	-3	-0,5	0,5
a1→ → e2→	1	0	1	0,5	2,5
e2→	0	1	-6	0,5	-4,5
e0→	0	0	21	-2,5	18,5
a1→ → e2→	1	0	1	0,5	2,5
a2→ → e3→	0	1	-6	0,5	-4,5
a3→ → e1→	0	0	1	-0,11	0,89

4.4 Využitie aplikácie

Ako už z názvu aplikácie vyplýva, aplikácia je použiteľná pri riešení viacerých typov úloh lineárnej algebry a jedného typu úlohy lineárneho programovania, resp. v takých prípadoch, kde vieme výsledok úlohy určiť pomocou elementárnej zmeny bázy. V podkapitolách nižšie je presne uvedené, čo je cieľom úlohy toho ktorého typu úlohy.

4.4.1 Úlohy a ich špecifikácie

4.4.1.1 Úlohy pre „Vektory“

a) Zistite, či je systém vektorov lineárne závislý alebo nezávislý
b) Vypočítajte hodnosť systému vektorov
c) Zistite, či je vektor $\bar{b}$ lineárnou kombináciou vektorov systému a vypíšte jeho zložky

4.4.1.2 Úlohy pre „Hodnosť matice“

a) Vypočítajte hodnosť matice A.

4.4.1.3 Úlohy pre „Rozklad matice“

a) Zistite, či je možné určiť základný bázický rozklad matice A.
b) Určte základný bázický rozklad matice A.

4.4.1.4 Úlohy pre „Inverzná matica“

a) Zistite, či je štvorcová matica A regulárna alebo singulárna.
b) Určte inverznú maticu k matici A.

4.4.1.5 Úlohy pre „Determinant matice“

a) Zistite, či sa dá pre maticu A vypočítať determinant.
b) Vypočítajte determinant štvorcovej matice A.

4.4.1.6 Úlohy pre „Systém lineárnych rovníc“

a) Zistite, či systém lineárnych rovníc má riešenie.
b) Určte všeobecné riešenie, resp. množinu riešení a zapíšte ho v parametrickom tvare.

4.4.1.7 Úlohy pre „Úloha lineárneho programovania“

a) Zapíšte ohraničujúce podmienky vo forme rovnosti.
b) Určte optimálne riešenie, ak existuje.

5 Diskusia

Vytvorená aplikácia je určená na počítanie niekoľkých typov úloh lineárnej algebry a lineárneho programovania. V porovnaní s inými aplikáciami (opísaných v kapitole 1) je naša aplikácia desktopová, čiže sa dá použiť iba po stiahnutí z externého zdroja. K našej aplikácií netreba byť pripojený na internet a dá sa použiť na všetkých počítačoch s operačným systémom Windows. Po stiahnutí je ihneď spustiteľná. Medzi hlavné výhody patrí moderné grafické rozhranie, jednoduchá formulácia úloh a prehľadné výpisy. Zároveň je pravdepodobne jedinou aplikáciou, ktorá je aspoň čiastočne interaktívna, čo prispieva k edukačnému charakteru aplikácie. Aplikácia nie je veľmi zložitá, čo môže byť aj nevýhoda.

Medzi ďalšie nevýhody patrí, že je spustiteľná len po stiahnutí, t.j. že nie je prístupná „kedykoľvek a kdekoľvek“. Hlavný nedostatok vidíme v počte riešiteľných úloh, keďže ostatné aplikácie ponúkajú oveľa širšie portfólio algoritmov.

Vzhľadom na pridelený čas na implementáciu, ponúka aplikácia celkom dostačujúci počet funkcionalít. Aplikácia bola otestovaná v rámci možností a nevykazuje žiadne známky chybovosti. Aplikácia je uložená ako „open source“ na portáli GitHub, je plne prístupná komukoľvek a pripravená na ďalšie rozšírenie, najmä v oblasti lineárneho programovania.

Záver

Táto práca bola venovaná programovaciemu jazyku C++ v lineárnej algebre, najmä na počítanie elementárnej zmeny bázy. Našou úlohou bolo vytvoriť aplikáciu, ktorá dokáže riešiť úlohy lineárnej algebry, poprípade aj úlohy lineárneho programovania.

V úvode práce sme sa zoznámili so súčasným stavom riešenej problematiky, a to najmä s postupmi, ako je lineárna algebra vyučovaná na Ekonomickej Univerzite v Bratislave. Preštudovaním knihy *Lineárna algebra pre ekonómov*[3] sme si osvojili základnú terminológiu, ale aj jednotlivé možnosti použitia metódy elementárnej zmeny bázy a jej predpoklady a dôsledky.

Podobné desktopové aplikácie sa nám, žiaľ, nepodarilo nájsť, na druhej strane sme však analyzovali tie, ktoré sú dostupné online. Jednalo sa predovšetkým o aplikácie s veľmi širokým spektrom použitia a bolo vopred jasné, že podobnú aplikáciu nie je v našich silách naprogramovať za obdobie riešenia tejto práce. Spoločnou črtou väčšiny skúmaných aplikácií bolo biedne grafické rozhranie s chaotickým rozložením komponentov. Usúdili sme, že bude lepšie vyvinúť jednoduchšiu, ale zato prehľadnú aplikáciu s prívetivým používateľským rozhraním. Zo skúmaných knižníc pre C++ sa nám nepodarilo vložiť do projektu ani jednu.

Za pomerne krátky čas sa nám podarilo navrhnúť aplikáciu, ktorá by spĺňala hlavné požiadavky. Definovali sme potrebné požiadavky, ako aj prípady použitia aplikácie. Počas vývoja aplikácie sme všetky jej súčasti podrobne testovali tak, aby šanca, že by aplikácia vykazovala chybovosť, bola minimálna. Aplikácia je v súčasnosti plne funkčná, spoľahlivo počíta mnoho úloh lineárnej algebry aj základná úlohu lineárneho programovania. Na všetky výpočty používa elementárnu zmenu bázy. Spĺňa všetky požiadavky zo zadania, je interaktívna, graficky na vysokej úrovni a jej používanie je intuitívne a jednoduché. Zároveň je aplikácia uložená ako „open source“ na portáli GitHub, je plne prístupná komukoľvek a pripravená na ďalšie rozšírenie.

Zoznam použitej literatúry

[1] "Linear Algebra" [Elektronický zdroj]. [06.12.2021].Dostupné na: https://en.wikibooks.org/wiki/Linear_Algebra.

[2] HRÚZOVÁ,Eva "NĚKTERÉ APLIKACE LINEÁRNÍ ALGEBRY" [Elektronický zdroj]. 2018. 50.[6.12.2021]. Dostupné na: https://dspace5.zcu.cz/bitstream/11025/32023/1/Hruzova Eva - Bakalarska prace.pdf

[3] SAKÁLOVÁ,Katarína-SIMONKA,Zsolt-STREŠŇÁKOVÁ,Anna,*Lineárna algebra pre ekonómov*. Letra Edu. 2020. 238. 978-80-89962-73-0

[4] CHOCHOLATÁ,Michaela *Lineárne programovanie pre manažérov*. Bratislava: Vydavateľstvo EKONÓM. 2013.234. 978-80-255-3562-5

[5] "MATLAB - MathWorks - MATLAB & Simulink." [Elektronický zdroj]. [16.12.2021]. Dostupné na: https://www.mathworks.com/products/matlab.html.

[6] "About Us." [Elektronický zdroj].[17.12.2021].Dostupné na: https://atozmath.com/Help/Aboutus.asp

[7] "Simplex method calculator." [Elektronický zdroj]. Dostupné na: https://cbom.atozmath.com/CBOM/Simplex.aspx?q=sm. [Accessed: 17-Dec-2021].

[8] "Maticová kalkulačka." [Elektronický zdroj]. [17.12.2021].Dostupné na: https://matrixcalc.org/sk/

[9] "Simplexová metoda – online kalkulátor." [Elektronický zdroj].[17.12.2021] Dostupné na: http://simplex.tode.cz/cs.

[10] "Gurobi Optimizer - Gurobi." [Elektronický zdroj].[18.12.2021].Dostupné na: https://www.gurobi.com/products/gurobi-optimizer/

[11] "Armadillo: C++ library for linear algebra & scientific computing." [Elektronický zdroj]. [18.12.2021].Dostupné na: http://arma.sourceforge.net/