UNIVERZA V NOVI GORICI
POSLOVNO-TEHNIŠKA FAKULTETA

REŠEVANJE OPTIMIZACIJSKIH PROBLEMOV S PROGRAMSKIM PAKETOM SCICOSLAB

DIPLOMSKO DELO

Jana Miklavič

Mentor: prof. dr. Juš Kocijan

Nova Gorica, 2012

NASLOV

Reševanje optimizacijskih problemov s programskim paketom ScicosLab

IZVLEČEK

Programski paket ScicosLab je odprtokodni program, ki se uporablja za numerično reševanje znanstvenih problemov in ga lahko brezplačno dobimo na svetovnem spletu. Program vsebuje tudi funkcije za reševanje optimizacijskih problemov. Namen diplomskega dela je prikazati zmožnosti programskega paketa ScicosLab za reševanje optimizacijskih problemov in uporabniku programskega paketa ScicosLab olajšati reševanje osnovnih optimizacijskih problemov.

Prvi del diplomskega dela vsebuje osnove teorije optimizacije, nato sledi matematična razlaga nekaterih najbolj znanih optimizacijskih metod. V drugem delu je najprej kratek opis programskega paketa ScicosLab, ki mu sledi predstavitev funkcij programa, ki so namenjene reševanju optimizacijskih problemov. Funkcije so razdeljene v tri skupine. V prvi skupini je funkcija, ki jo uporabljamo za linearno programiranje, v drugi je funkcija za kvadratno programiranje in v tretji so predstavljene funkcije nelinearnega programiranja. Na koncu so predstavljeni še rešeni primeri za vsako obliko programiranja.

KLJUČNE BESEDE

optimizacija, kriterijska funkcija, ScicosLab, linearno programiranje, nelinearno programiranje, kvadratno programiranje, omejitev

TITLE

Solving optimization problems using ScicosLab software package

ABSTRACT

ScicosLab is an open source program that can be downloaded from the internet free of charge. It is used for scientific computation and contains functions for optimization. The aim of this thesis is the demonstration of ScicosLab package utility for solving optimization problems and to facilitate basic optimization problem solving for ScicosLab user.

The first part of the thesis describes the theoretical part, followed by mathematical explanation of some frequently used optimization techniques. A brief description of ScicosLab software package is in the second part, followed by the presentation of program functions, for solving optimization problems. Functions are divided into three groups. The function used for linear programming is in the first group. The function for quadratic programming is in the second group and the third group contains functions and features of non-linear programming. Illustrative examples for each of optimization-function groups are presented at the end.

KEYWORDS

optimization, cost function, ScicosLab, linear programming, quadratic programming, non-linear programming, constraints

KAZALO

KAZALO SLIK

KAZALO TABEL

1 UVOD

Optimizacija pomeni iskanje neke ekstremne vrednosti oziroma, kako neko stvar glede na dane pogoje prilagoditi ali spremeniti tako, da dobimo čim boljši rezultat ali učinek. Pojem optimizacija se uporablja v različnih panogah, npr. v tehniki, transportu, ekonomiji, računalništvu itd.

Značilnost vsakega optimizacijskega problema je, da ima določeno število spremenljivk, za katere iščemo optimalne vrednosti. Pri iskanju optimalne rešitve smo omejeni z omejitvami, ki se nanašajo na spremenljivke in skupaj definirajo dopustno rešitev. Obstaja več metod reševanja optimizacijskih problemov, odvisnih od njihovih značilnosti (Srebrnič, 2011).

Pripomoček, ki lahko pomaga pri optimizaciji nekega problema, je programski paket ScicosLab, ki se uporablja za reševanje numeričnih problemov s področja tehnike, naravoslovja, ekonomije in drugih področij, ki so zanimiva za gospodarske inženirje.

Namen diplomske naloge je narediti sistematični prikaz, kako s programskim paketom rešujemo numerične optimizacijske probleme.

Cilj diplomske naloge je napisati priročnik, ki bo uporabniku programskega paketa ScicosLab v pomoč pri reševanju nalog s področja optimizacije.

V prvem delu je predstavljen splošni pregled optimizacije, kakšno vlogo ima v tehniki in pri poslovnih procesih, kje vse se uporablja, kako jo izvajamo in kako je formuliran optimizacijski problem. V drugem delu je predstavljen teoretični del, opisane so matematične metode optimiranja. Tretji del vsebuje opis metod oziroma funkcij, ki jih vsebuje programsko orodje ScicosLab, ter primere uporabnih nalog.

2 OPTIMIZACIJA

Beseda optimizacija ima več pomenov. V matematiki je optimizacija veda, ki se ukvarja z iskanjem maksimuma in minimuma funkcije znotraj nekih meja. V računalništvu je optimizacija proces izpolnitve sistema na način, ki povečuje hitrost izvajanja ali zmanjšuje porabo pomnilnika. Z vidika ekonomije optimiranje predstavlja proces, kako zmanjšati stroške obratovanja, materiala ali kako doseči maksimalni dobiček. S pojmom optimizacija se srečujemo tudi v vsakdanjem življenju, kako priti najhitreje iz enega kraja v drugega, kako najbolje naložiti prihranke in podobno.

V leksikonih pod pojmom optimizacija najdemo naslednjo definicijo: »Optimizacija je proces iskanja najboljše rešitve nekega problema oziroma optimalnost je stanje, ki je glede na dane možnosti najugodnejše, najboljše« (Leksikon, 1998, str. 743).

Proces optimizacije je v splošnem sestavljen iz naslednjih faz:

1. formuliranje,
2. optimiranje in
3. preizkus rezultata.

2.1 Vloga optimizacije v tehniki

Optimizacijo v širšem pomenu uporabljamo pri reševanju katerega koli inženirskega problema. Tipična področja so:

- optimalno načrtovanje in obratovanje proizvodnje,
- obvladovanje časov v čakalnih vrstah proizvodnje,
- projektiranje sistema za doseganje maksimalne učinkovitosti.

S prehodom na nove tehnologije in z raznimi programskimi paketi se je to področje precej razvilo, tako da pridemo dokaj hitro do točnejših rezultatov oz. rešitev.

Proces optimiranja (Kravanja in Novak Pintarič, 2006, str. 5) je ena od temeljnih tehnik za načrtovanje novih, boljših in cenejših sistemov ali za izboljšanje učinkovitosti delovanja obstoječih sistemov. Optimizacija je uporabna v vseh vejah tehnike. Štiri glavna področja (Kravanja in Novak Pintarič, 2006, str. 8) so:

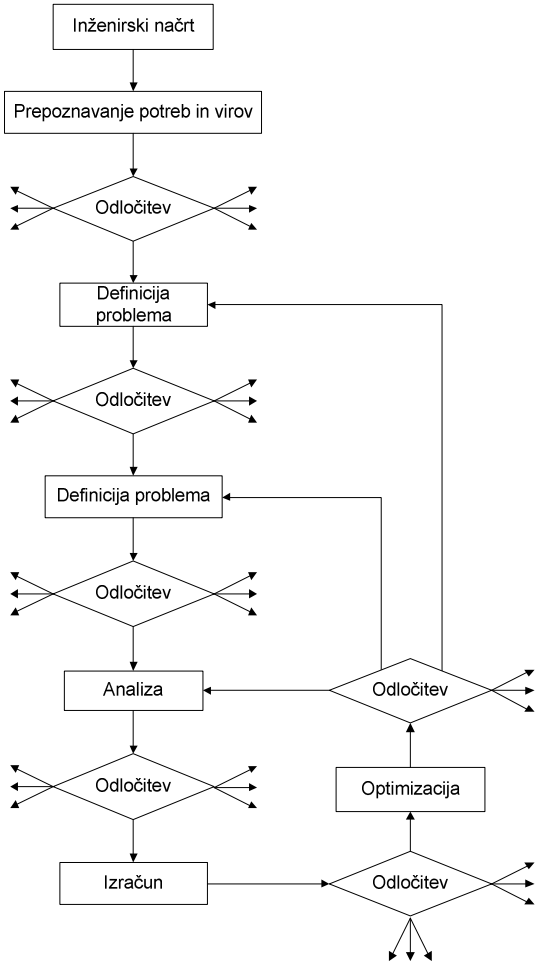
i) načrtovanje komponent ali celotnega sistema,

ii) načrtovanje in analiza obstoječih postopkov,

iii) inženirska analiza in redukcija podatkov in

iv) vodenje oz. regulacija dinamičnih sistemov.

Vedeti moramo, da je uporaba optimizacijskih metod pri projektiranju sistema samo en korak v celotnem procesu doseganja optimalnega projektiranja oziroma en korak pri doseganju učinkovitosti delovanja sistema. Celoten sistem načrtovanja (Ravindran in drugi, 2006) je prikazan na sliki 1.



Slika 1: Načrt procesa načrtovanja

Optimizacija se izvaja po korakih, ki jih lahko razdelimo v naslednje faze:

- analiza problema,

- identifikacija problema,
- opredelitev rešitve in cilja,
- testiranje rešitev in
- analiza rezultatov.

Ravindran in drugi (2006) navajajo, da je proces sestavljen iz ponavljajočega se cikla, ki definira strukturo sistema. Končen optimalni načrt je mogoče dobiti oz. doseči le po reševanju nizov optimizacijskih problemov, katerih vsak rezultat služi kot »generator« za proizvajanje novih idej za nadaljnjo strukturo sistema.

Sistemu, ki v celoti izpolnjuje načrtovalske zahteve, pravimo optimalni sistem. Te zahteve so v večini idealizirane in jih z realnim sistemom ne moremo doseči. Vsak sistem, ki se v okviru dovoljenih odstopanj najbolj približa zahtevam načrtovalca, imenujemo optimalni sistem. Ker vnaprej ne more vedeti, ali bo sistem izpolnjeval zahteve, najprej oblikuje začetnega. Na podlagi analiz se prikažejo resnične lastnosti sistema, ki se v večini razlikujejo od zahtevanih. Na podlagi razlik sistem spreminjamo toliko časa, dokler ta ni optimalen. Sistematični postopek spreminjanja sistema, ki pripelje do optimalnega, imenujemo optimizacija sistema (Matko in Bratkovič, 1993).

Optimizacija je proces, ki je sestavljen iz več med seboj povezanih korakov. Pred začetkom je treba korake dobro definirati. Pomembno je, da cilje in izboljšave določimo natančno, da bomo lahko ob zaključku optimiranja primerjali rezultate s cilji, ki smo si jih zastavili.

2.2 Optimizacija poslovnih procesov

Pri poslovnih procesih (Osojnik, 2009) se pogosto srečujemo z reševanjem nalog, ki imajo mnogo možnih rešitev. Med vsemi danimi možnostmi želimo poiskati le tisto, pri kateri je pri danih pogojih dosežen najboljši poslovni učinek. Temeljno načelo gospodarjenja je, da z danimi sredstvi skušamo doseči maksimalni rezultat (dobiček), ki ga želimo dobiti z minimalno porabo sredstev.

Poglejmo si nekaj primerov:

- Iščemo najoptimalnejši proizvodni plan določenih izdelkov, ki se izdelujejo iz omejenih količin surovin, ki so na zalogi. Pri prodaji izdelkov želimo ustvariti maksimalni dobiček, če poznamo dobiček za posamezni izdelek.
- Želimo nabaviti določene sestavine, ki jih mešamo med seboj, in tako nastajajo novi produkti, za katere poznamo stroške nabave. Izdelati želimo optimalni plan nabave posameznih in posledično plan izdelave produktov, s katerimi bi dosegli minimalne stroške nabave in bi zadovoljili kupčeve potrebe.
- Transportni problemi obravnavajo prevoze določene količine izdelkov od proizvajalcev (prodajalcev) do kupcev. Če imamo znane stroške prevoza izdelka za posameznega kupca, želimo doseči minimalne stroške skupnih prevozov. Ob tem je potrebno upoštevati omejitve, kot so omejitev razpoložljivih izdelkov v proizvodnji (skladišču), in izpolniti vse zahteve kupcev po določeni količini izdelkov.

Pri optimizaciji zalog je potrebno upoštevati gibanje zalog v odvisnosti od gibanja prodaje in obratno. To pomeni, da je potrebno določiti tisti nivo zalog, ki še zagotavlja optimalno pokrivanje proizvodnje in prodaje.

V mnogih podjetjih predstavljajo stroški in storitve več kot 70 % prodajne cene proizvoda in storitev. Zato je nadzor nad cenami ključnega pomena za doseganje poslovne uspešnosti. Za uspešen nadzor je potrebno:

- opraviti pregled in analizo vseh stroškov v podjetju,
- vzpostaviti ustrezno vodenje materialnega poslovanja, ki vključuje ustrezno strukturo zajemanja podatkov v informacijskem sistemu in
- pri iskanju optimalne ravni stroškov je možnih več strategij, odvisno predvsem od stanja v dejavnosti, v kateri podjetje posluje.

2.3 Formuliranje problema

2.3.1 Definiranje mej sistema

Preden pričnemo obravnavati nek optimizacijski problem, je pomembno, da jasno določimo omejenost oz. meje sistema (Kravanja in Novak Pintarič, 2006). Sistem

tako ločimo od okolice z jasno določenimi mejami. Tako izključimo vse interakcije med sistemom in okolico. Na ta način je definiranje mej prvi korak procesa k približevanju realnega sistema.

Pri podrobni analizi sistema se pri mnogih primerih izkaže, da so začetne mejne vrednosti preveč omejene. Obravnavanemu sistemu razširimo meje, ki omogočajo, da v dani sistem vključimo tudi druge podsisteme, ki močno vplivajo na delovanje proučevanega sistema. Tako ločimo dve vrsti sistemskih mej (Kravanja in Novak Pintarič, 2006):

- ozke meje in
- široke meje.

Sistem z ozkimi mejami je zelo poenostavljen in zožen. Meje so zelo omejene. Sistem je enostaven, zato ga je lažje formulirati in lažje optimirati.

Sistem s širokimi mejami je kompleksnejši, vsebuje več podsistemov in zato daje optimiranje celovitejši in boljši rezultat.

2.3.2 Določanje kriterija

Drugi pomemben korak pri obravnavanju sistema je določiti kriterij na osnovi katerega ocenjujemo učinke ali namen sistema tako, da dobimo najboljši načrt in obratovalne pogoje sistema. Velikokrat je izbran ekonomski kriterij. Pri tem je pomembno upoštevati način opredelitve takega kriterija. Ekonomski kriteriji so: skupna vrednost kapitala, letni stroški, letni neto dobiček, vrnitev investicije, neto sedanja vrednost, razmerje med stroški in dobičkom itd.

Pri drugih primerih upoštevamo tehnološke dejavnike, kot so: minimalni čas proizvodnje, maksimalna dinamika proizvodnje, minimalna poraba energije, maksimalni navor, maksimalna teža itd.

Optimiranje izvajamo:

- po enem kriteriju ali
- redkeje po več kriterijih.

Pogosto se pri obravnavi sistema osredotočimo le na en kriterij, s katerim bomo dosegli optimum. Z optimizacijskimi metodami ni enostavno najti rešitev, da bi npr.

neko podjetje istočasno delovalo z minimalnimi stroški in dosegalo maksimalno proizvodnjo z minimalnim energetskim izkoriščanjem.

V primeru, ko optimiramo sistem po več kriterijih, si pogosto izberemo primarni in sekundarni kriterij. Primarni kriterij se uporablja kot merilo uspešnosti in ga opišemo z namensko funkcijo, medtem ko sekundarni kriterij uporabljamo kot dodatne omejitve oz. pogoje.

V primeru enakovrednih kriterijev pa govorimo o večkriterijski optimizaciji (Ravindran in drugi, 2006), ki je posebna veja optimizacije.

2.3.3 Neodvisne spremenljivke

Tretji pomemben element pri formulaciji problema je izbira neodvisne spremenljivke. Ta mora ustrezati značilnostim, ki so primerne za označevanje možnega načrtovanja »kandidata«, ali ustrezati pogoju sistema.

Pri izbiri neodvisne spremenljivke (Ravindran in drugi, 2006) obstaja več dejavnikov, ki jih je treba upoštevati.

- Razlikovati moramo med spremenljivkami, katerih vrednosti se spreminjajo, in tistimi, katerih vrednosti so fiksne oz. stalne. Stalne vrednosti določajo zunanji dejavniki, ki se nahajajo zunaj meja izbranega obravnavanega sistema.
- Poleg tega je pomembno ločevati tudi sistemske parametre. Nekatere obravnavamo kot stalne, ti imajo stalno vrednost, druge kot spremenljive, katerih vrednosti se spreminjajo. Na njihovo vrednost vplivajo nekontrolirani zunanji dejavniki.
- Pri formulaciji modela je pomembno vključiti vse pomembne spremenljivke, ki vplivajo na delovanje sistema ali na opredelitev načrta.
- Četrti vidik pri izbiri spremenljivke je raven podrobnosti, ki vplivajo na sistem. Čeprav je pomembno obravnavati vse ključne neodvisne spremenljivke, je enako pomembno, da ne zakriva problema, ki vključuje večje število »finih« pomembnosti, ki so podrejenega pomena.

Dobro pravilo pri izbiri neodvisne spremenljivke je, da vključimo samo tiste, ki pomembno vplivajo na merilo uspešnosti sestavljenega sistema.

2.3.4 Model sistema

Ko so izbrani kriteriji za zmogljivost in neodvisne spremenljivke, sestavimo še model, ki opisuje zvezo med spremenljivkami in kako neodvisne spremenljivke vplivajo na kriterij optimiranja.

Moderni postopki načrtovanja vodenja temeljijo na modelu procesa. Model procesa je možno dobiti na dva načina:

- s teoretično in
- z eksperimentalno analizo.

Teoretična analiza temelji na fizikalnih zakonih, eksperimentalna na meritvah.

Osnovno načelo optimizacije je, da neodvisno spremenljivko sistema v praksi nastavimo na določene vrednosti in opazujemo, kako se s spreminjanjem teh vrednosti izboljšujejo oz. spreminjajo kriteriji. Tak eksperimentalni pristop je drag, dolgotrajen in tvegan. V praksi se zato večinoma izvaja poenostavljeno matematično prikazovanje dejanskega sistema, ki ga imenujemo model. Tako se pri večini modelov uporablja teoretični pristop, saj je ta v primerjavi z eksperimentalnim cenejši in hitrejši.

Model je sestavljen (Kravanja in Novak Pintarič, 2006) iz:

a) enačb, ki:
   - opisujejo masne in energetske bilance,
   - načrtovalske enačbe,
   - fizikalne lastnosti,
   - fizikalne pojave v sistemu, ali

b) neenačb, ki:
   - opisujejo območja dovoljenih obratovalnih pogojev,
   - proizvodne specifikacije (npr. povpraševanje, kvaliteta proizvodov itd.) in
   - logične pogoje.

Model je tako sestavljen iz več elementov, ki opisujejo, kako so med seboj povezane spremenljivke sistema, ki jih moramo upoštevati pri preračunavanju oziroma napovedovanju uspešnosti tehničnega sistema.

»Numerične metode so temeljno orodje za reševanje matematičnih modelov, ki jih formuliramo pri reševanju tehniških problemov. Pri numeričnih metodah vedno računamo s števili. Podatki so števila in rezultati so tabele, ki jih pogosto predstavimo še grafično. Tipični postopek reševanja tehniškega problema ima naslednje faze:

1. Modeliranje. Postavimo matematični model naloge, to je integral, sistem enačb ali sistem diferencialnih enačb.
2. Izberimo numerično metodo in parametre.
3. Napišemo program v izbranem programskem jeziku in uporabimo čim več funkcij, ki so na voljo.
4. Izvedemo izračun.
5. Interpretiramo numerične rezultate kot rešitev tehničnega problema.

Najpomembnejši je prvi korak. Majhna sprememba modela lahko pomembno vpliva na njegovo uporabnost. V drugem koraku moramo izbrati učinkovito numerično metodo, zato moramo to najprej spoznati« (Petrišič, 2011, str. 93).

3 TEORIJA IN METODE

Pri formuliranju definiramo model, namensko oz. kriterijsko funkcijo, meje sistema oz. problema in določimo spremenljivke. Če pogledamo iz matematičnega vidika, je model sestavljen iz:

i. spremenljivk,
ii. kriterijske funkcije in
iii. omejitev.

Splošni matematični model problema zapišemo tako:

Poišči $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$, ki minimizira $f(\mathbf{x})$ z naslednjima mejnima pogojema

$$g_j(x) \leq 0, \quad j = 1,2,\dots,m$$
$$h_j(x) = 0, \quad j = 1,2,\dots,p$$

pri tem je :

$\mathbf{x}$ – n-dimenzionalni vektor,
$f(\boldsymbol{x})$ – kriterijska funkcija,
$g_j(x)$ – mejni pogoj neenakosti,
$h_j(x)$ – mejni pogoj enakosti,
n – število spremenljivk,
m – število mejnih pogojev za g in
p – število mejnih pogojev za h.

Prikazan matematični model je omejen z mejnimi pogoji. V primeru, ko problem ne vsebuje mejnih pogojev, imamo opravka z neomejenim oz. s problemom brez mejnih pogojev.

Pri problemih brez omejitev, pri katerih je funkcija dvakrat zvezno odvedljiva, iščemo rešitve z iskanjem stacionarnih točk, v katerih je gradient kriterijske funkcije enak nič. Za določitev tipa stacionarne točke uporabimo Hessovo matriko. Če je Hessova matrika pozitivno definitna, je točka lokalni minimum, če je negativno definitna imamo v točki lokalni minimum.

V naslednjih podpoglavjih bomo opisali osnovne omejitvene optimizacijske metode. Osnovna in ena preprostejših omejenih optimizacijskih metod je zagotovo linearno programiranje, ki se uporablja za iskanje ekstremne vrednosti kriterijske funkcije.

3.1 Linearno programiranje

Osnovna značilnost linearnega programiranja (Nemanič, 2009) je linearna povezava med spremenljivkami.

Z metodami in postopki linearnega programiranja (v nadaljevanju LP) rešujemo probleme tako, da maksimiramo ali minimiziramo kriterijsko funkcijo. Ekstremno vrednost kriterijske funkcije iščemo v pogojih, ki jih opisuje sistem linearnih enačb in/ali neenačb. Neznanke, ki jih želimo poiskati, predstavljajo konkretno vrednost količine, naj bo to dolžina, cena, število izdelkov, material, čas itd. Lastnosti teh neznank je, da so pozitivne.

Praktični problemi LP so običajno zelo obsežni, zaradi česar pri numeričnem reševanju zahtevajo toliko računanja, da jih lahko rešujemo le z računalnikom.

Razpoložljiva programska oprema za matematično programiranje nudi uporabniku poleg osnovnih rešitev v numerični ali grafični obliki tudi dodatne analize rezultatov optimizacije, ki znatno olajšajo sprejemanje čim boljše odločitve v pravem času.

Najprej moramo formulirati problem na osnovi podatkov, z ustreznimi enačbami. Zelo pozorni moramo biti, da matematična formulacija kar najbolje ustreza danemu problemu, jasno definiramo cilje in kriterij optimiranja. Dobra formulacija problema (Osojnik, 2009) brez dvoma predstavlja vsaj pol poti do končne rešitve.

Matematična oblika modela LP, pri kateri iščemo maksimalno vrednost kriterijske funkcije, je naslednja:

Določiti moramo vrednost nenegativne komponente vektorja $\mathbf{x} = (x_1, x_2, \dots, x_n)$, ki bo dala maksimalno vrednost ciljne funkcije:

$$\max f(\mathbf{x}) = \mathbf{c}^{\mathrm{T}} \cdot \mathbf{x} \qquad (1)$$

Pri tem moramo upoštevati mejne pogoje:

$$\mathbf{A} \cdot \mathbf{x} \leq \mathbf{b}, \tag{2}$$

$$\mathbf{x} \geq \mathbf{0}.$$

Pri tem so:

$\mathbf{A} = \begin{bmatrix} a_{11} & a_{11} & \dots & a_{1n} \\ a_{21} & a_{11} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}$ – matrika znanih koeficientov vrste (m, n),

$\mathbf{c}^{\mathrm{T}} = \begin{bmatrix} c_1 & c_2 & \dots & c_n \end{bmatrix}$ – vektor, transponent matrike $\mathbf{C}$, koeficienti ciljne funkcije,

$\mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_n \end{bmatrix}$ – vektor prostih členov, ki označujejo mejne pogoje,

$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix}$ – vektor neznanih vrednosti spremenljivk $\mathbf{x}$,

m in n sta dimenziji vektorja.

Tak je zapis v matrični obliki. V skalarni obliki je problem LP zapisan na naslednji način:

$$\text{maksimiramo } f(x_1, x_2, \dots, x_n) = c_1x_1 + c_2x_2 + \dots + c_nx_n\ , \tag{3}$$

ki je omejena z:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ \dots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \ \dots + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{m1}x_1 + a_{m2}x_2 + \ \dots + a_{mn}x_n &= b_m \end{aligned} \tag{4}$$

c_j, b_j, in a_{ij} $(i = 1,2, \dots, m;\ j = 1,2, \dots, n)$ označujejo koeficiente in x_j je odločitvena spremenljivka.

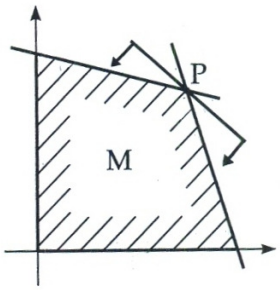
Glavne značilnosti LP v standardni obliki so:

1. kriterijska funkcija tipa minimum,
2. vse omejitve so v obliki linearnih enačb in
3. vse spremenljivke so nenegativne.

3.1.1 Grafična metoda reševanja LP

Metodo uporabljamo pri reševanju problemov z dvema ali s tremi spremenljivkami, se pravi v dveh ali v treh dimenzijah. Omejitve so tako lahko premice in ravnine, prav tako je lahko kriterijska funkcija premica ali ravnina.

»Množico dopustnih rešitev definira polje ali prostor, določen z oglišči presečišč funkcij omejitev. Omejitve določajo mnogokotnik v ravnini ali polieder v prostoru neodvisnih spremenljivk. Dopustne rešitve linearnega problema so tiste, ki zadovoljujejo vse omejitve. Optimalna rešitev v območju vseh dopustnih rešitev je tista rešitev, ki leži na meji območja in pri kateri kriterijska funkcija dosega maksimum ali minimum« (Srebrnič, 2011, str. 10).



Slika 2: Dopustno območje **M**

Na omejenem nepraznem dopustnem območju M (slika 2) kriterijska funkcija zavzame največjo in najmanjšo vrednost v najmanj enem oglišču. Če območje M ni omejeno, se lahko zgodi, da kriterijska funkcija ne zavzame najmanjše ali največje vrednosti.

Grafična metoda je enostavna, ima pa eno pomanjkljivost. V kolikor ima problem več kot dve spremenljivki, ni več primerna. Za reševanje problemov z več spremenljivkami uporabljajmo metodo simpleks.

3.1.2 Metoda simpleks

Metoda simpleks (Nemanič, 2009, str. 10-11) je osnovna metoda za reševanje LP. Pri reševanju problemov z metodo simpleks lahko uporabljamo poljubno število spremenljivk. Danes je z računalnikom možno reševati tudi zelo kompleksne probleme.

Ta metoda je ponavljajoča in po določenem številu ponovitev privede do optimalne rešitve. Problem rešujemo po korakih, od ene do druge rešitve tako, da vrednost kriterijske funkcije vse bolj približamo optimalni oz. ekstremni vrednosti.

Za dani problem postavimo matematični model. Pri uporabi te metode moramo mejne pogoje, ki so podani v obliki neenačb, spremeniti v enačbe. Ker je enačbe lažje obravnavati kot neenačbe, vpeljemo dopolnilne spremenljivke, ki neenačbe pretvorijo v enačbe. Dopolnilne spremenljivke ne vplivajo na optimalno rešitev. Tako so koeficienti dopolnilnih spremenljivk v kriterijski funkciji enaki nič. Pri tem velja pravilo, da kolikor je neenačb, toliko je dopolnilnih spremenljivk. Po uvedbi dopolnilnih spremenljivk osnovna rešitev ne vsebuje enotske matrike. Da bi to dosegli, je potrebno uvesti t. i. pomožne spremenljivke. Glede na to, da v optimalni rešitvi ne sme biti pomožne spremenljivke, dodamo koeficient kriterijske funkcije, ki ima poljubno vrednost. Osnovni korak do optimalne rešitve je načrtovanje prve možne osnovne rešitve. V tej rešitvi se nahajajo samo dopolnilne spremenljivke (enotska matrika), vse strukturne spremenljivke imajo vrednost nič. Rezultati simpleks metode se običajno prikažejo v tabeli, kar poenostavi izračun in olajša analizo dobljenih rezultatov.

Tabela 1: Tabela simpleksov

	x_1	$\cdots$	x_{n-m}	
x_{n-m+1}	$a_{1,1}$	$\cdots$	$a_{1,n-m}$	b_1
$\vdots$	$\vdots$		$\vdots$	$\vdots$
x_n	$a_{m,1}$	$\cdots$	$a_{m,n-m}$	b_m
	c_1	$\cdots$	c_{n-m}	$-c_0$

V naslednjem koraku moramo izbrati drugo osnovno rešitev, kar pomeni, da v bazo vektorskega prostora uvedemo eno od strukturnih spremenljivk. Pri izbiri spremenljivke za vhod v bazo ima glavno vlogo zadnja vrsta v tabeli, $z_j - c_j$, ki jo imenujemo simpleks kriterij. Ta kriterij je indikator rasti kriterijske funkcije. Simpleks kriterij se računa kot razlika seštevkov vsote koeficientov iz stolpca neke j-te spremenljivke, s koeficienti spremenljivk v bazi in vrednosti koeficientov ciljne funkcije te iste spremenljivke. Negativni predznak simpleks kriterija označuje velikost izgube kriterijske funkcije, ki nastaja zaradi tega, ker ustrezna nebazna spremenljivka ni v bazi.

K izboljšanju vrednosti kriterijske funkcije bo največ pripomogla tista spremenljivka, ki ima maksimalno vrednost simpleks kriterija z negativnim predznakom v primeru maksimuma. Na splošno lahko to napišemo z naslednjim izrazom:

$$z_j - c_j = max\{|(z_j - c_j < 0)|\}(j = 1,2, \dots n) \tag{5}$$

Kriterij za izhod spremenljivke iz baze je določen z naslednjim izrazom:

$$max(x_j) = min\left(\frac{b_i}{a_{ij}}\right) > 0 \tag{6}$$

Negativni koeficienti $\left(\frac{b_i}{a_{ij}}\right)$ se zanemarijo, saj je že na začetku definiran pogoj nenegativnosti.

Sedaj preidemo na izvedbo druge osnovne rešitve. Pri transformaciji podatkov simpleks tabele obstajajo trije načini:

i. Transformacija podatkov iz stolpca, ki se nanaša na spremenljivko, ki vstopa v bazo, se izvede tako, da polje na presečišču značilne vrstice in stolpca dobi vrednost 1, vsa ostala polja v stolpcu pa dobijo vrednost 0. Razlog temu je, da vsaki bazni spremenljivki pripada enotski vektor.
ii. Transformacija podatkov v značilni vrstici se izvede tako, da se vrednost polja te vrstice v novi simpleks tabeli dobi z deljenjem vrednosti polja značilne vrstice z vrednostmi glavnega elementa.
iii. Vse preostale vrednosti v tabeli se dobijo z naslednjim izrazom:

$$\left(v_{ij}\right)_{k+1} = \left(v_{ij}\right)_k - \left(\frac{v_{jr} \cdot v_{is}}{v_{rs}}\right)_k \quad (7)$$

$\left(v_{ij}\right)_{k+1}$ – vrednost poljubnega mesta v $k+1$ tabeli

$\left(v_{ij}\right)_k$ – vrednost istega polja kot v prejšnji tabeli

v_{jr} – vrednost v značilni vrstici, ki pripada stolpcu j

v_{is} – vrednost v značilnem stolpcu, ki pripada vrstici i

v_{rs} – vrednost glavnega elementa

Pogoj za zaključek simpleks metode je $\left(z_j - c_j\right) \geq 0$, $(j = 1,2, \dots, n)$.

Vsakemu problemu linearne optimizacije lahko priredimo natanko določen drug problem linearne optimizacije, ki ga imenujemo dualni problem. Če se pri primarnem modelu išče maksimum kriterijske funkcije, pri dualnem problemu iščemo minimum. Število mejnih pogojev pri primarnem modelu je neko število spremenljivk pri dualnem modelu, število spremenljivk pri primarnem modelu je enako številu mejnih pogojev pri dualnem problemu. Prehod na dualni problem se uporablja, kadar:

- iščemo minimum kriterijske funkcije in
- ko je število mejnih pogojev m večje od števila spremenljivk n.

V tabeli 1 sta prikazana primarni in dualni problem LP.

Tabela 2: Primer primarnega in dualnega modela

Primarni model	Kriterijska funkcija: $f(x) = c_1 \cdot x_1 + c_2 \cdot x_2 + \cdots + c_n \cdot x_n \rightarrow max$
	Mejni pogoji: $a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \leq b_1$ $a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \leq b_2$ …… $a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \leq b_m$
Dualni model	Kriterijska funkcija: $f(y) = b_1 \cdot y_1 + b_2 \cdot y_2 + \cdots + b_n \cdot y_n \rightarrow min$
	Mejni pogoji: $b_{11}y_1 + b_{12}y_2 + \cdots + b_{1n}y_n \leq c_1$ $b_{21}y_1 + b_{22}y_2 + \cdots + b_{2n}y_n \leq c_2$ …… $b_{m1}y_1 + b_{m2}y_2 + \cdots + b_{mn}y_n \leq c_m$

3.2 Kvadratno programiranje

Kvadratno programiranje (Bronštejn in drugi, 2009) uporabljamo takrat, ko v problemu nastopa kriterijska funkcija kvadratne oblike. Omejitve kriterijske funkcije so v linearni obliki. Problem kvadratnega programiranja je v vektorski obliki definiran na naslednji način:

$$\min f(x) = \mathbf{c}^{\mathrm{T}}\mathbf{x} + \mathbf{x}^{\mathrm{T}}\mathbf{Q}\mathbf{x}, \in \mathbb{R}^n \tag{8}$$

omejen z :

$$\mathbf{A}\mathbf{x} \leq \mathbf{b} \tag{9}$$

$$\mathbf{x} > \mathbf{0} \tag{10}$$

$\mathbf{A}$ – matrika velikosti mxn z omejenimi koeficienti

$\mathbf{b}$ – stolpični vektor, velikosti $mx1$

$\mathbf{c}$ – vrstični vektor, velikosti $1xn$

$\mathbf{Q}$ – simetrična matrika, velikosti nxn

$\mathbf{x}$ – spremenljivka, vektor velikosti $nx1$

Dopustno območje (množica vektorjev, ki zadoščajo omejitvam) M opišemo še na dva načina:

$$\text{i) } \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0} \tag{11}$$

$$\text{ii) } \mathbf{A}\mathbf{x} \leq \mathbf{b} \tag{12}$$

Ko v kriterijski funkciji nastopata dve spremenljivki $\mathbf{x} = [x_1, x_2]$, lahko funkcijo zapišemo v naslednji matrični obliki:

$$f(x_1, x_2) = [x_1 \quad x_2] \cdot \begin{bmatrix} q_{1,1} & q_{1,2} \\ q_{2,1} & q_{2,2} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + [c_1 \quad c_2] \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{13}$$

V primeru, ko je $f(x)$ strogo konveksna za vse možne točke, ima problem enkraten lokalni minimum, ki je obenem tudi globalni minimum. Zadosten pogoj, da je funkcija strogo konveksna je, da je matrika $\mathbf{Q}$ pozitivno definitna. Če je $\mathbf{Q} = 0$, potem problem postane linearen.

Metoda Lagrangeovih multipleksorjev (Bronštejn in drugi, 2009)

Pri tej metodi uporabimo Lagrangeovo funkcijo in Kuhn-Tuckerjev pogoj. Lagrangeova funkcija problema je :

$$L(\mathbf{x},\mathbf{u}) = \mathbf{x}^{\mathrm{T}}\mathbf{Q}\mathbf{x} + \mathbf{c}^{\mathrm{T}}\mathbf{x} + \mathbf{u}^{\mathrm{T}}(\mathbf{A}\mathbf{x} - \mathbf{b}). \tag{14}$$

Huhan-Tuckerjevi pogoji:

$$\mathbf{v} = \frac{\partial L}{\partial \mathbf{x}} = \mathbf{c} + 2\mathbf{Q}\mathbf{x} + \mathbf{A}^{\mathrm{T}}\mathbf{u} \quad \text{in} \quad \mathbf{y} = \frac{\partial L}{\partial \mathbf{u}} = -\mathbf{A}\mathbf{x} + \mathbf{b}. \tag{15}$$

Dopustna območja kvadratnega problema, ki ima definirane Kuhan-Tuckerjeve pogoje, so:

	Primer I	Primer II	Primer III	
a)	$\mathbf{A}\mathbf{x} + \mathbf{y} = \mathbf{b}$	$\mathbf{A}\mathbf{x} = \mathbf{b}$	$\mathbf{A}\mathbf{x} + \mathbf{y} = \mathbf{b}$	(16)
b)	$2\mathbf{C}\mathbf{x} - \mathbf{v} + \mathbf{A}^{\mathrm{T}}\mathbf{u} = -\mathbf{d}$	$2\mathbf{C}\mathbf{x} - \mathbf{v} + \mathbf{A}^{\mathrm{T}}\mathbf{u} = -\mathbf{d}$	$2\mathbf{C}\mathbf{x} + \mathbf{A}^{\mathrm{T}}\mathbf{u} = -\mathbf{d}$	(17)
c)	$\mathbf{x} \geq \mathbf{0}, \mathbf{v} \geq \mathbf{0}, \mathbf{y} \geq \mathbf{0}, \mathbf{u} \geq \mathbf{0}$	$\mathbf{x} \geq \mathbf{0}, \mathbf{v} \geq \mathbf{0}$	$\mathbf{y} \geq \mathbf{0}, \mathbf{u} \geq \mathbf{0}$	(18)
d)	$\mathbf{x}^{\mathbf{T}}\mathbf{v} + \mathbf{y}^{\mathbf{T}}\mathbf{u} = 0$	$\mathbf{x}^{\mathbf{T}}\mathbf{v} = 0$	$\mathbf{y}^{\mathbf{T}}\mathbf{u} = 0$	(19)

Za reševanje kvadratne optimizacije uporabljamo Wolfejevo metodo, ki ima naslednjo obliko:

$$f(\mathrm{x}) = \mathbf{x}^{\mathrm{T}}\mathbf{Q}\mathbf{x} + \mathbf{c}^{\mathrm{T}}\mathbf{x} = \min \text{ in omejitev } \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \text{ in } \boldsymbol{x} \geq \mathbf{0}. \tag{20}$$

in Hildreth-d'Esopovo metodo, ki jo uporabljamo pri problemu oblike:

$$f(\mathrm{x}) = \mathbf{x}^{\mathrm{T}}\mathbf{Q}\mathbf{x} + \mathbf{c}^{\mathrm{T}}\mathbf{x} = \min \text{ in omejitvijo } \mathbf{A}\mathbf{x} \leq \boldsymbol{b}. \tag{21}$$

3.3 Nelinearno programiranje

Nelinearno programiranje (Nemanič, 2009) se ukvarja z optimiranjem nelinearnih funkcij z linearnimi ali nelinearnimi mejnimi pogoji. Problemi s področij npr. ekonomije in tehnike imajo večje število možnih rešitev. Pri tem je naloga optimiranja najti najbolj ugodno možno rešitev za dani problem med večjim številom potencialnih rešitev s stališča koristnosti ali izvedbe. Optimiranje lahko izvajamo z numeričnimi in matematičnimi algoritmi.

Na splošno metode optimiranja delimo na analitične, numerične, grafične in eksperimentalne.

- Pri analitični metodi uporabljamo tehnike diferencialnega in variacijskega računanja. S temi metodami iščemo ekstrem funkcije $f(x)$. Najti moramo tako vrednost x, da bo odvod funkcije $f(x)$ enak nič. Ko uporabljamo analitično metodo, mora biti problem, za katerega iščemo optimum, opisan z matematičnimi členi v taki obliki, da lahko s funkcijami in spremenljivkami računamo po znanih pravilih. Za komplicirane nelinearne probleme analitična metoda ni primerna.
- Numerično metodo uporabljamo v primeru, ko odpove analitična. Informacije, ki jih dobimo iz predhodnih korakov, uporabljamo za pridobivanje boljših rešitev v naslednjem koraku ponovitve. Ta metoda je prilagodljiva pri reševanju praktičnih problemov.
- Pri grafični metodi iščemo funkcije ene ali več spremenljivk, pri katerih iščemo maksimum ali minimum. Iz grafa narisane funkcije preberemo ekstrem.
- Ekstrem funkcije problema pri eksperimentalni metodi dobimo z uporabo matematičnega opisa problema. Rezultat eksperimenta se uporablja za sprejetje odločitve o tem, kam bi bilo potrebno postaviti naslednji eksperiment z željo izboljšati rezultate prejšnjega eksperimenta.

Osnovna oblika matematičnega modela nelinearnega programiranja je formulirana na naslednji način: Določiti moramo maksimum ali minimum kriterijske funkcije

$$f(x) = f(x_1, x_2, \dots, x_n) \tag{22}$$

pri pogojih, definiranih s sistemi enačb in/ali neenačb.

$$g_i(x) = g_i(x_1, x_2, \dots, x_n) \leq/\geq 0, i = 1,2, \dots, n \tag{23}$$

Funkciji $f(x)$ in $g_i(x)$ imata lahko poljubno obliko, saj so pogoji nenegativnosti upoštevani v mejnih pogojih.

Pri reševanju problemov poskušamo najti vrednost n-dimenzionalega vektorja oziroma točke x v n-dimenzionalnem evklidskem prostoru. Cilj reševanja je, da se s tako definirano množico možnih rešitev določi točka x, za katero kriterijska funkcija

zavzame ekstremno vrednost. Tako dobimo točko optimuma pod pogojem, da točka zadovolji vnaprej definirane mejne pogoje danega problema.

3.3.1 Funkcija ene spremenljivke

Najbolj osnovna in poznana metoda za iskanje lokalnega minimuma in maksimuma je metoda z uporabo odvodov ali Newtonova metoda.

Velja izrek: Naj bo funkcija $f(n+1)$-krat zvezno odvedljiva na neki okolici točke x_0 in naj bo

$$f'(x_0) = f''(x_0) = \ ... = f^{(n-1)}(x_0) = 0 \quad \text{ter} \qquad f^{(n)}(x_0) \neq 0, \tag{24}$$

potem velja:

i) če je n sodo število in $f^{(n)}(x_0) < 0$, potem ima f v x_0 lokalni maksimum,

ii) če je n sodo število in $f^{(n)}(x_0) > 0$, potem ima f v x_0 lokalni minimum,

iii) v primeru, da je n liho število, potem f v x_0 nima lokalnega ekstrema.

Pomembna je naslednja trditev:
Naj bo f odvedljiva v neki okolici točke x_0 in naj bo $f'(x_0) = 0$. Tedaj velja:

a) če je $f''(x_0) < 0$, potem ima f v x_0 lokalni maksimum,

b) če je $f''(x_0) > 0$, potem ima f v x_0 lokalni minimum.

Med drugim lahko uporabljamo še naslednje metode:

- polinomsko aproksimacijo,
- uniformno iskanje in
- metodo razvejanja.

3.3.2 Funkcija več spremenljivk

a) Ekstremi funkcije dveh spremenljivk

Funkcija f ima v točki (x_0, y_0) lokalni maksimum (minimum), če obstaja taka okolica točke (x_0, y_0), da za vsako točko (x, y) iz te okolice velja, da je:

$$f(x,y) \leq f(x_0, y_0) \text{ za maksimum oziroma} \tag{25}$$

$$f(x,y) \geq f(x_0, y_0) \text{ za minimum.} \tag{26}$$

Potreben pogoj za nastop ekstrema je: $\frac{\partial f}{\partial x}(x_0, y_0) = 0$ in $\frac{\partial f}{\partial y}(x_0, y_0) = 0$. Da je točka (x_0, y_0) res ekstrem, ugotovimo z odvodi drugega reda, ki ga označimo z A, B in C.

$$A = \frac{\partial^2 f}{\partial x^2}(x_0, y_0) \tag{27}$$

$$B = \frac{\partial^2 f}{\partial x \partial y}(x_0, y_0) \tag{28}$$

$$C = \frac{\partial^2 f}{\partial y^2}(x_0, y_0) \tag{29}$$

Kvadratna oblika: $J(x_0, y_0) = AC - B^2 = \begin{vmatrix} A & B \\ B & C \end{vmatrix}$. (30)

Naj bo $\frac{\partial f}{\partial x} = 0$ in $\frac{\partial f}{\partial y} = 0$. Če je $AC - B^2 > 0$, potem ima funkcija f v točki (x_0, y_0) ekstrem, in sicer:

a) če je $A > 0$ ima v tej točki minimum,
b) če je $A < 0$ ima v tej točki maksimum.

Če je $AC - B^2 < 0$, potem v točki (x_0, y_0) ni ekstrema, temveč sedlo.
Če je $AC - B^2 = 0$ iz vrednosti parcialnih odvodov drugega reda, ne moremo nič sklepati.

b) Ekstremi funkcije več spremenljivk

Metoda eliminacije

Tudi pri iskanju ekstrema funkcije več spremenljivk uporabljamo metode z uporabo odvodov. Med njimi je najbolj znana metoda eliminacije. Ta metoda pripada skupini analitičnih metod reševanja nalog nelinearnega programiranja (v nadaljevanju NP).

Z analitičnimi postopki določimo stacionarne točke kriterijske funkcije $f(x)$. Določitev stacionarnih točk je vezano na reševanje sistema enačb:

$$\frac{\partial F}{\partial x_j} = 0 \qquad j = 1, 2, \dots, n \tag{31}$$

Značilnosti stacionarne točke proučujemo s parcialnimi odvodi druge stopnje oz. z determinanto D_j, ki ima naslednjo obliko:

$$D_j = \begin{vmatrix} Fx_1,x_1 & Fx_1,x_2 & \dots & Fx_1,x_j \\ Fx_2,x_1 & Fx_2,x_2 & \dots & Fx_2,x_j \\ \dots & \dots & \dots & \dots \\ Fx_j,x_1 & Fx_j,x_2 & \dots & Fx_j,x_j \end{vmatrix}, \quad (32)$$

kjer so:

$$Fx_1,x_1 = \frac{\partial^2 F(x)}{\partial x_1^2} \quad (33)$$

$$Fx_1,x_2 = \frac{\partial^2 F(x)}{\partial x_1 \partial x_2} \quad (34)$$

$$\dots$$

$$Fx_i,x_j = \frac{\partial^2 F(x)}{\partial x_i \partial x_j} \quad (35)$$

$$D_1 = Fx_1,x_1$$

$$D_2 = \begin{vmatrix} Fx_1,x_1 & Fx_1,x_2 \\ Fx_2,x_1 & Fx_2,x_2 \end{vmatrix}$$

$$D_3 = \begin{vmatrix} Fx_1,x_1 & Fx_1,x_2 & Fx_1,x_3 \\ Fx_2,x_1 & Fx_2,x_2 & Fx_2,x_3 \\ Fx_3,x_1 & Fx_3,x_2 & Fx_3,x_3 \end{vmatrix}$$

Zato da v stacionarni točki obstaja minimum, mora veljati, da so vse determinante $D_j > 0$.

Da dosežemo maksimum, morata veljati naslednja pogoja:

- $D_j > 0$ za $j = 1, 3, 5, \dots, 2n+1$,
- $D_j < 0$ za $j = 2, 4, 6, \dots, 2n$.

Ta metoda se uporablja pri reševanju enostavnih problemov. Pri tem moramo paziti, da so mejni pogoji linearne oblike. Bistvo metode je, da posamezno, eno po eno, neznanko eliminiramo, vse dokler obstaja možnost eliminacije.

Pri reševanju problemov nelinearnega programiranja, pri katerih ni podanih omejitev, lahko uporabljamo metode (Bronštejn in drugi 2009):

- najstrmejšega spusta,
- Quasi-Newtonovo (metoda za iskanje ničel: $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$),
- konjugiranih gradientov,
- Davidon-Fletcher Powellovo (DFP),

- Levenberg-Marquardt.

To so metode, pri katerih uporabljamo gradient funkcije.

Druge metode, ki jih uporabljamo pri NP in pri katerih ne uporabljamo gradienta, so (Bronštejn in drugi 2009):

- metoda vloženih intervalov (sloni na primerjavi funkcijskih vrednosti),
- omejenega gradienta,
- dopustnih smeri,
- kazni in ovir,
- konveksnih kombinacij,
- Monte Carlo,
- Nelder-Meadova (nelinearna simpleks metoda),
- Hooke-Jeevesova,
- hibridna Powelova
- idr.

4 SCICOSLAB

ScicosLab je prosto dostopen programski paket za numerično reševanje in analiziranje podatkov. Razvili so ga v Franciji, na institutu INRIA[1], in je prosto odprto programsko okolje. Deluje na različnih operacijskih sistemih, kot so Windows, Linux, Unix itd. Vsebuje izvorno kodo, sprotno pomoč in uporabniški priročnik v angleškem jeziku (ScicosLab, 2011).

Program se uporablja za reševanje matematičnih zapisov in struktur z vsemi znanimi oblikami števil, kot so kompleksna števila, matrike, polinomi, racionalne, eksponentne in logaritemske funkcije. Analiziramo lahko linearne in nelinearne dinamične sisteme. Z objekti lahko prikazujejo modele dinamičnih sistemov. Uporabljamo ga kot pripomoček za numerično optimizacijo.

Funkcije prepoznava kot podatkovne objekte, z njimi računa ali jih pretvori v druge podatkovne objekte. Funkcijo, ki je definirana znotraj ScicosLab-a, lahko uporabljamo kot vhodni ali izhodni argument druge funkcije. Podpira znakovne nize podatkovnega tipa, kar v posebnih pogojih omogoča sprotno tvorjenje funkcij (Makarovič, 2005).

Visoka raven programskega jezika dovoljuje dostop do naprednih podatkovnih struktur. Program omogoča grafični prikaz funkcij, premic, vektorjev, polinomov, racionalnih funkcij, diagramov, animacij. Funkcije lahko prikažemo v dvodimenzionalnem in tridimenzionalnem prostoru. Grafe lahko izvozimo v različne formate, kot so .gif, .bmp, .jpg, .pdf ipd. Odprtih imamo lahko več grafičnih oken hkrati, vendar je lahko v danem trenutku aktivno le eno okno (Jereb in drugi, 2010).

Program ScicosLab vsebuje modul Scicos (Bajc, 2007). Uporablja se za modeliranje in simuliranje (hibridnih) dinamičnih sistemov. S Scicos-om lahko modeliramo in grafično sestavimo bločne simulacijske sheme za zelo kompleksne dinamične sisteme. Sistemi so prikazani v povezovalnih blokih in podsistemih ali nadblokih.

Pri združevanju znanosti in industrijskih potreb je lahko programski paket ScicosLab pri reševanju raznih problemov v veliko pomoč.

[1]angleško: National Institute for Research in Computer Science and Control, francosko: Institut national de recherche en informatique et en automatique.

4.1 Zgodovina – razvoj ScicosLab-a

ScicosLab je bil osnovan na podlagi programskega paketa Scilab in programskega modula Scicos. Scicos je bil razvit kot del znanstvenega programa programskega paketa Scilab. Ta paket je razvila skupina raziskovalcev, ki so navdih za program dobili pri programu Matlab. Hoteli so izdelati paket, ki ne bi vključeval le tolmača in osnovnih matematičnih operacij, ampak tudi omogočal simulacije in postopke za optimizacijo.

Leta 1994 je izšla prva različica Scilab programa. Sproti so ga popravljali, posodabljali ter izdali priročnik za uporabo v angleškem in francoskem jeziku.

Hitrejši razvoj je dosegel po letu 2000. Razvit je bil nov simulator v programskem jeziku C, razširjena je bila nova podatkovna vrstica, posodobljena je bila tudi knjižnica z grafičnimi postopki.

Idejo za orodje Scicos so dobili pri razširitvi jezika sinhronega signala s stalno časovno dinamiko. Prva verzija je bila napisana v Scilab jeziku, vendar ni imela grafične podpore. Kasneje so kodirali simulator v Fortranu in del predvajalnika v C in tako je leta 1994 program vseboval tudi grafični urejevalnik, ki so ga postopoma nadgrajevali in razvijali.

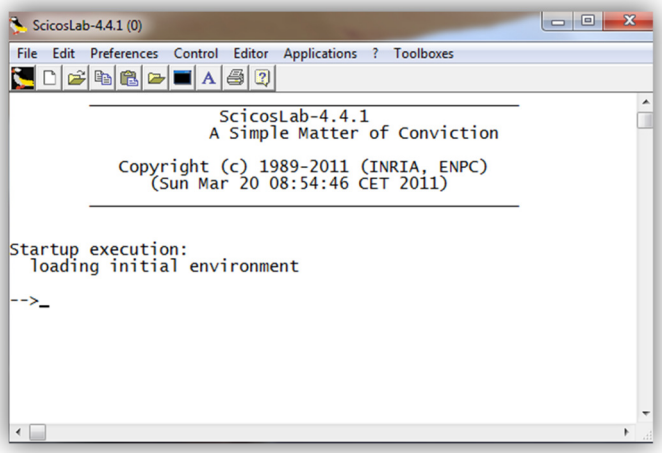
Zaradi težav, stabilnosti in učinkovitosti delovanja s Scilab-om 5, Scicos razvijajo le še za ScicosLab. Program ScicosLab so ločili od razvoja novih različic z namenom, da bi med njimi ohranili skladnost. Najnovejša različica ScicosLab-a je na voljo na strani http://www.scicoslab.org/, verzija 4.4.1.

4.2 Prenos in namestitev programa

Z uradne spletne strani *www.scicoslab.org* program brezplačno prenesemo in namestimo na računalnik, sledimo navodilom, ki se prikazujejo, ter pustimo, da računalnik opravi namestitev.

Slika 3: Zagon programa ScicosLab-a

Program zaženemo z dvakratnim klikom na ikono (slika 3). Ob zagonu se prikaže osnovno okno z menijsko in orodno vrstico (slika 4).



Slika 4: Zagon programa (menijska in orodna vrstica)

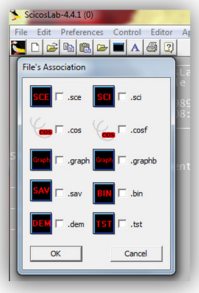
Vrstico z menijem sestavlja osem razdelkov.

i. Razdelek *Datoteka (File)* vsebuje ukaze, s katerimi odpremo novo okno programa *(NewScicosLab)*. Omogoča zagon datoteke *(Execute),* odprtje že shranjene datoteke *(Open)*, naložitev shranjene spremenljivke oz. programa *(Load)*, shranitev spremenljivke ali datoteke *(Save)* spreminjanje delovne mape *(Change Directory),* izpis trenutne delovne mape *(Get Current Directory)*, predogled oz. nastavitve tiskanja *(Print Setup)*, tiskanje zapisanega programa *(Print)* ter izhod iz programa *(Exit)*.

ii. Razdelek *Uredi (Edit)* vsebuje ukaze izberi vse *(Select All)*, kopiranje *(Copy)*, lepljenje podatkov, ki se nahajajo v drugih dokumentih/datotekah *(Paste)*, izprazni odložišče *(Empty Clipboard)* in zgodovino *(History).* Zgodovina nam omogoča naslednje ukaze:
   - kurzor prikaže zadnji-/-sledeč ukaz,

- kurzor se postavi en znak naprej-/-nazaj,
- kurzor se postavi na začetek-/-konec vrstice,
- izbriši prejšnji-/-trenutni znak,
- izbriši zadnjo vrstico, konec vrstice ali celotno vrstico in
- ponovno prikaži vrstico *(Redraw the Line)*.

iii. Razdelek *Možnosti (Preference)* ponuja izbiro jezika *(Language)*, ki ga lahko uporablja program, francoščino ali angleščino, izbir barve *(Colors)* za tekst in ozadje ali izbiro prevzete barvne sheme programa *(Default System Colors)*, ukaz za prikaz orodne vrstice *(Toolbar)*, ukaz za združevanje datotek *(Files Association),(slika 5)*, ukaz za izbiro vrste pisave *(Choose Font).* V primeru, ko imamo v okencu podatke, ki jih ne potrebujemo več, jih enostavno izbrišemo z ukazom izbriši zgodovino ukazov *(Clear History),* izbriši ukazno okno *(Clear Command Window)* in ukazni vmesnik *(Console)*.



Slika 5: Združevanje datotek

iv. Razdelek *Nadzor (Control)* vsebuje tri orodja. (*Resume)* omogoča izvajanje programa po premoru, (*Abort*) preneha z izvajanjem trenutnega programa in prekinitev *(Interrupt)* z izvajanjem trenutnega programa.

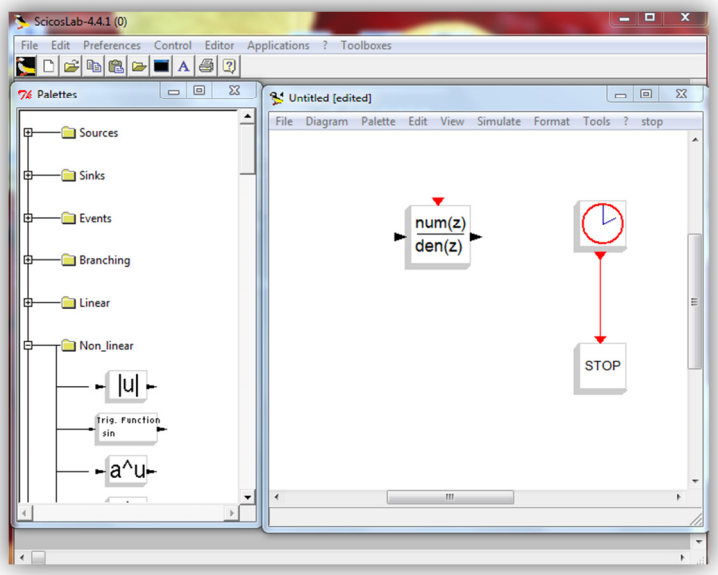
v. Ko (dvakrat) kliknemo na razdelek urejevalnik *(Editor),* se odpre novo okno za pisanje programa, ki ga nato izvozimo v osnovno programsko okno.

vi. Razdelek *Aplikacije (Application)* vsebuje orodje Scicos (slika 6), ki ga uporabljamo za risanje blokovnih shem, orodje za urejanje grafa

(*EditGraph*), orodje m2sci, s katerim pretvarjamo datoteke iz programa Matlab v Scilab, in brskalnik spremenljivk *(Browser Variables)*.

vii. Razdelek *?* vsebuje pomoč pri uporabi programa *(ScicosLab Help)* in konfiguriranje *(Configure)*. *Scilab Demos* nudi možnost ogleda že izdelanih primerov programa, *Web Links* omogoča dostop do spletne strani *www.scicoslab.org*.

viii. Razdelek *Orodja (Toolboxes)* vsebuje orodji cs5 in Coselica, ki vsebujeta osnovne bloke Modelica za Scicos in Scicoslab.

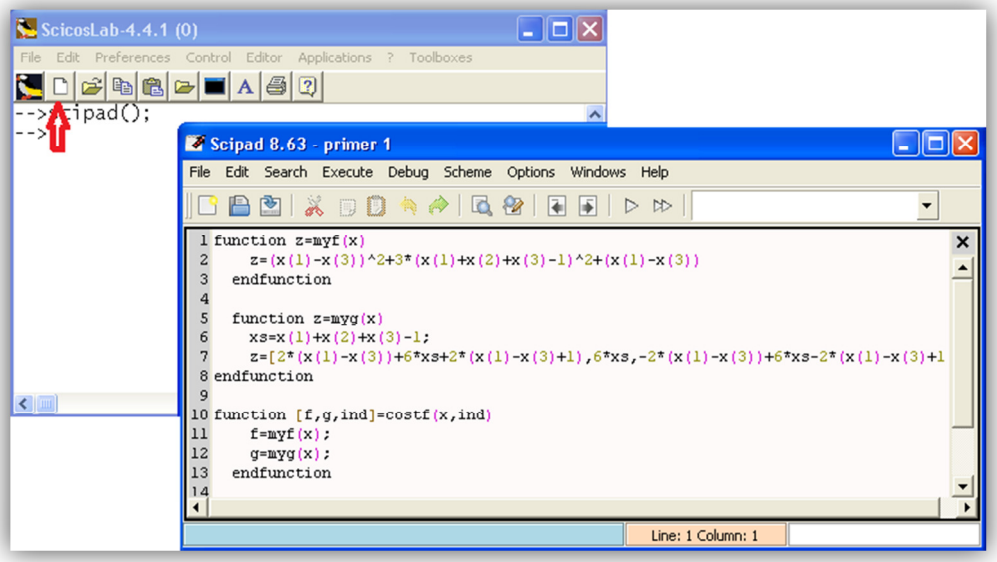


Slika 6: Shema orodja Scicos

Pri uporabi programskega paketa ScicosLab si lahko pomagamo s priročnikom (Makarovič, 2005), kjer so opisani osnovni napotki, kako zapišemo funkcijo, kako računamo z njimi in jih grafično prikažemo. Za uporabo orodja Scicos nam je v pomoč priročnik (Bajc, 2007).

4.2.1 Scipad

Ker program vsebuje urejevalnik Scipad, je priporočljivo, da zapise vpisujemo v ta urejevalnik. Urejevalnik Scipad ima to lastnost, da nam shranjuje zapisane ukaze in komentarje. Do urejevalnika pridemo tako, da kliknemo na orodno vrstico *Editor* ali da kliknemo na ikono v orodni vrstici, kot je prikazano na sliki 7. Odpre se urejevalnik, v katerega vpisujemo zapise in opombe. Datoteko shranimo v poljubno mapo, ki jo lahko uporabljamo tudi kasneje. Shranjene zapise lahko popravljamo in jih nato prenesemo v ukazno okno.



Slika 7: Pisanje programa v urejevalniku

4.3 Optimizacija v ScicosLabu

Programski paket Scicoslab vsebuje poglavje, ki se nanaša na optimizacijo. Izbira funkcije vpliva tudi na izbor parametrov. Predno začnemo z reševanjem problemov oziroma nalog moramo poznati značilnosti, ki določajo, katero funkcijo ali ukaz uporabimo. Funkcije za optimiranje lahko razdelimo v tri skupine:

1. funkcija za linearno programiranje:
   - `linpro`,
2. funkciji za kvadratno programiranje:
   - `quapro` in pomožna
     - `qld`,

3. funkcije za nelinearno programiranje:
    - `optim`,
    - `datafit`,
    - `fsolve`,
    - `leastsq`,
    - `lsqrsolve` in pomožne funkcije:
        - `derivative`,
        - `NDcost`,
        - `numdiff`.

Ko rešujemo optimizacijski problem, najprej preverimo, ali ima potrebna funkcija omejitev. To nas zanima predvsem za neodvisno spremenljivko x. Če za x obstaja omejitev, imamo opravka z omejeno funkcijo. Programsko orodje ScicosLab prepozna tri oblike omejenosti:

a) mejne omejitve: x se nahaja na določenem intervalu,
b) linearna enakost, oblike: $\mathbf{b}^{\mathrm{T}}\mathbf{x} = 0$ in
c) linearno neenakost, oblike: $\mathbf{b}^{\mathrm{T}}\mathbf{x} \leq 0$,

$\mathbf{b}$ in $\mathbf{x}$ sta stolpični matriki velikosti $(mx1)$.

Splošna omejitev ima obliko $g(x) = 0$. Programski paket ScicosLab za tako popolno splošno obliko omejitve še nima funkcije za optimizacijo. V primeru, ko imamo problem, oblike:

$$minf(x) = 0 \text{ in} \tag{36}$$

$$g(x) = 0\ , \tag{37}$$

pri čemer velja, da sta g in f odvedljivi. Da dobimo rešitev x morajo veljati naslednji pogoji:

$$\nabla f(x) + \lambda \nabla g(x) = 0 \text{ in} \tag{38}$$

$$g(x) = 0\ , \tag{39}$$

kjer je $g(x)$ Jacobijeva matrika funkcije g in λ je spremenljivka funkcije g, poznana kot Lagrangeov multipleksor. Pri takem problemu lahko uporabimo funkcijo `fsolve`.

Druga stvar, na katero moramo biti pozorni, je, ali je funkcija f odvedljiva in ali poznamo njen gradient. Če je funkcija f odvedljiva, se pogosto uporablja iterativno metodo. Znotraj iteracijske metode se v vsaki iteraciji oziroma ponovitvi določi smer iskanja in velikost premika v tej smeri. Pri tem se pojavi vprašanje, koliko ponovitev naj izvedemo. Izvedemo toliko ponovitev, dokler se vrednost funkcija f skoraj ne spremeni, in dokler je gradient funkcije f majhen ali ko je bilo upoštevano predpisano število ponovitev. Pri nekaterih iteracijskih metodah, pri katerih običajno iščemo minimum, najdemo lokalni minimum, ki pa ni nujno tudi globalni minimum (Campell in drugi, 2006).

Tudi nelinearne enačbe oblike:

$$\mathbf{f}(x) = 0 \tag{40}$$

lahko rešimo s ponovitveno metodo. Glavno vlogo pri tem ima Jacobijeva matrika $\mathbf{J}(x)$. Če ima funkcija m vhodov in x n vhodov, dobimo matriko:

$$\mathbf{J}(x) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix},$$

v matriki i-te vrstice je gradient i-te komponente funkcije f_i.

V primeru, da je $m = n$ in $\mathbf{J}(x)$ obrnljiva matrika, uporabimo Newtonovo metodo.

Pri neomejeni optimizaciji uporabljamo funkcijo `optim`. Ta algoritem uporabniku omogoča nadzirati optimizacijski proces. Funkcija sicer dopušča meje optimalnih parametrov, obenem pa mora poznati gradient iskane funkcije. Če gradienta ne poznamo ali je ta preveč zapleten, potem funkcija, `optim` računa samo s funkcijo, ki ji želimo poiskati minimum. Pri tem je v pomoč funkcija `NDcost`. Funkcija uporablja pri računanju odvoda numerično vrednost, ki je približek pravi vrednosti. Začetna vrednost `NDcost` ima obliko stolpičnega vektorja, pri tem dobimo numerično oceno gradienta, ki se nahaja v okolici ničle.

Za reševanje nelinearnih enačb z omejitvami uporabljamo funkcijo `fsolve`. Obstajajo primeri, pri katerih ne moremo rešiti enačbe $\mathbf{f}(x) = \mathbf{0}$. Najti želimo tak x, da dobimo vrednost $\mathbf{f}(x)$, ki je čim bližje 0. Če želimo dobiti x, ki minimizira želeno funkcijo, uporabimo metodo najmanjših kvadratov:

$$\|\mathbf{f}(x)\|^2 = \mathbf{f}(x)^{\mathrm{T}}\mathbf{f}(x) = \sum_{i=1}^{m} f_i(x)^2 \ .$$

To velja v primeru, ko imamo več enačb kot neznank. Funkciji, s katerima računamo po metodi najmanjših kvadratov, sta `leastsq` in `lsqrsolve`.

`Leastsq` je podobna funkciji `optim` s to razliko, da je poleg funkcije f, dana še Jacobijeva matrika, $\mathbf{J} = (\mathrm{x})$. Algoritem nato računa kriterijsko funkcijo f in njen gradient. Razlika je le ta, da so parametri podani v drugačnem vrstnem redu kot pri `optim`.

Včasih pri reševanju problemov, ki jih rešujemo z različnimi metodami, dobimo različne rešitve, kljub temu da smo pri vseh uporabili enake začetne vrednosti.

Za optimizacijo prileganja funkcijskih parametrov skrbi funkcija `datafit`. Za dano funkcijo $G(p, z)$ najde `datafit` najboljši vektor s parametri p, ki se najbolj prilegajo $G(p, z_1) = 0$.

Osnovni linearni problem je minimizirati funkcijo $\mathbf{p}^{\mathrm{T}}\mathbf{x}$, z linearnimi omejitvami

$$\mathbf{C_1 x} \leq \mathbf{b}_1, \tag{41}$$

$$\mathbf{c}_{\mathrm{i}} \leq \mathbf{x} \leq \mathbf{c}_{\mathrm{s}}, \tag{42}$$

$$\mathbf{C}_2\mathbf{x} = \mathbf{b}_2. \tag{43}$$

Linearne omejitve so lahko v obliki enačbe ali neenačbe, ki so lahko odprte ali zaprte. Funkcija, ki v ScicosLab-u rešuje te vrte problemov, je `linpro`.

Za probleme kvadratnega programiranja uporabljamo funkcijo `quapro`, ki rešuje probleme oblike: $\frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{Q}\mathbf{x} + \mathbf{p}^{\mathrm{T}}\mathbf{x}$. Ker `quapro` ne zahteva, da je $\mathbf{Q}$ pozitivno definitna, lahko je tudi enaka nič, rešujemo tudi probleme linearnega programiranja. Ko je matrika $\mathbf{Q}$ simetrična, je sklic za `quapro` identičen funkciji `linpro`.

Funkcija `NDcost` omogoča, da rešimo nekatere probleme, ne da bi določili gradient funkcije. Nekatere funkcije samodejno ocenijo diferencial ali izračunajo Jacobijevo matriko, ko se za to pojavi zahteva. To lahko izvedemo s funkcijama `numdiff` in `derivative`.

Funkcija `numdiff` izračuna številčno oceno Jacobijeve matrike. Pri izračunu funkcija uporablja metodo končnih razlik. Če imamo opravka z diferenciali, ki so omejeni na prvi in drugi red, uporabljamo funkcijo `derivative`.

Recimo, da ima funkcija (44) vektor x, dimenzije n, katere vrednost predstavljajo vektorji, dimenzije m. Točka a je potem določena s Taylorjevo vrsto:

$$\boldsymbol{f}(x) = \begin{bmatrix} f_1(x) \\ \vdots \\ f_m(x) \end{bmatrix} = \boldsymbol{f}(a) + \mathbf{J}(a)(x-a) + \begin{bmatrix} (x-a)'H_1(a)(x-a) \\ \vdots \\ (x-a)'H_m(a)(x-a) \end{bmatrix} + \cdots \qquad (44)$$

Tako je prvi odvod Jacobijeve matrike, velikosti mxn, drugi odvod je tako sedaj matrika m, velikosti nxn .

4.4 Neomejena optimizacija

Poglejmo si splošen primer optimizacije z uporabo funkcije `optim`, katere opis je dopolnjen v poglavju 4.7.

Primer: Minimizirati želimo sledečo funkcijo:

$$f(x,y,z) = (x-z)^2 + 3(x+y+z-1)^2 + (x-z+1)^2$$

z začetno vrednostjo $x_0 = (0,0,0)$.

V ScicosLab-u funkcijo $y = f(x)$ definiramo z ukazom `deff`, ali s `function – endfunction`. Razlika med ukazoma je ta, da z `deff` sproti definiramo funkcijo, ki jo definira uporabnik v samem programu, s funkcijo `function – endfunction` definiramo vgnezdene funkcije.

V zapisu `[f,xopt]=optim(costf,x0)`:
`x0 –` predstavlja začetno vrednost spremenljivke,
`xopt` – določa vrednost spremenljivk, kjer funkcija f doseže optimum,
`costf` – izračuna vrednost gradienta funkcije, ki ji želim poiskati minimum.

Da dobimo ekstrem funkcije, moramo pri uporabi ukaza `optim` navesti gradient funkcije.

Gradient funkcije f je: $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}\right]$

$$= \begin{bmatrix} 2(x-z)+6(x+y+z-1)+2(x-z+1), \\ 6(x+y+z-1), \\ -2(x-z+1)+6(x+y+z-1) \end{bmatrix}$$

V urejevalnik programa zapišemo funkcijo in njen gradient. Osnovno funkcijo smo zapisali kot $z = myf(x)$ in gradient $z = myg(x)$. Spremenljivke zapišemo z izrazi: $x = x(1)$, $y = x(2)$, $z = x(3)$. Ker ima gradient funkcije v vseh treh parcialnih odvodih enak člen, $(x + y + z - 1)$, ga zapišemo kot novo spremenljivko xs.

```
function z=myf(x)
z=(x(1)-x(3))^2+3*(x(1)+x(2)+x(3)-1)^2+(x(1)-x(3))
endfunction

function z=myg(x)
xs=x(1)+x(2)+x(3)-1;
z=[2*(x(1)-x(3))+6*xs+2*(x(1)-x(3)+1),6*xs,-2*(x(1)-
x(3))+6*xs-2*(x(1)-x(3)+1)]
endfunction
```

Sedaj zapišemo še ukaz, s katerim bomo izračunali zgornji primer.

```
function [f,g,ind]=costf(x,ind)
f=myf(x);
g=myg(x);
endfunction

x0=[0 0 0];
[f,xopt]=optim(costf,x0)
```

Izpiše se rezultat:

```
xopt =
      0.0833333    0.3333333    0.5833333

f =
   0.5
```

Pri bolj zapletenih primerih moramo določiti omejitev, koliko ponovitev naj program izvede. S tem nadzorujemo čas računanja.

4.5 Linearno programiranje

Pri reševanju nalog linearnega programiranja uporabljamo ukaz `linpro`. V primeru, da želimo poiskati rešitev sistema linearnih enačb brez kakršnih koli omejitev, lahko uporabimo ukaz: `linsolve`.

Linpro

`linpro` je funkcija, ki rešuje linearne probleme oblike: $min\ \mathrm{c}^{\mathrm{T}}\mathbf{x}$, za katero velja:

$$\mathbf{Ax} \leq \mathbf{b}$$

$\mathbf{A}_{eq}\mathbf{x} = \mathbf{b}_{eq}$ in $\mathbf{c}_i \leq \mathbf{x} \leq \mathbf{c}_s$, kjer $\mathbf{c}_i$ predstavlja spodnjo mejo in $\mathbf{c}_s$ zgornjo mejo. Funkcijo lahko zapišemo z naslednjimi ukazi:

a) `[x,lagr,f]=linpro(c,A,b [,x0])`
b) `[x,lagr,f]=linpro(c,A,b,ci,cs [,x0])`
c) `[x,lagr,f]=linpro(c,A,b,ci,cs,me [,x0])`
d) `[x,lagr,f]=linpro(c,A,b,ci,cs,me,x0 [,imp])`

`c` – vektor koeficientov linearne kriterijske funkcije z dimenzijo n,
`A` – matrika z $(m + p)$ vrstic in n stolpcev,
`b` – stolpični vektor z $(m + p)$ vrstic,
`ci` in `cs` – stolpična vektorja, ki predstavljata spodnjo in zgornjo mejo spremenljivk,
`lagr` – vektor Lagrangovega multipleksorja,
`f` – optimalna vrednost funkcije,
`x` – vrednost vektorja (spremenljivk), s katero minimiziramo kriterijsko funkcijo,
`x0` – vektor začetnih vrednosti optimiranih spremenljivk,
`imp` – celoštevilčna vrednost, ki določa vrednost (obseg) informacij,
`me` – predstavlja število omejitvenih enačb.

Kateri ukaz bomo uporabili, je odvisno od same strukture problema, oziroma kako so definirane omejitve. Prvi ukaz (a) uporabimo le v primeru, ko so omejitve dane v obliki neenačb. Drugi ukaz (b) uporabljamo, ko sta spodnja in zgornja meja dani v obliki neenačb. Ko so omejitve dane v obliki enačb in neenačb, uporabljamo tretji ukaz (c). Ukaz (d) uporabimo takrat, ko so omejitve opisane v obliki enakosti in

neenakosti. Poleg določene spodnje in zgornje meje imamo dano tudi začetno vrednost funkcije `x0`.

Če nimamo dane začetne točke `x0`, program sam izračuna možno začetno točko. Ta točka je teme območja možne točke, če velja $x0 =$ 'v'. Če v zapis zapišemo `x0='g'`, program prav tako izračuna začetno točko, ki pa ni nujno, da je teme območja. Ta zapis je priporočljivo uporabiti takrat, ko je kvadratna oblika (matrika) pozitivno definitna, in ko obstajajo omejitve funkcije, ali ko so meje spremenljivk velike, oziroma so to »varne« meje in niso pomembne pri reševanju optimuma.

Primer: Izračunajmo minimum funkcije $f(x) = \frac{x_1}{4} + \frac{x_2}{3}$, za katero velja:

$$-5x_1 - x_2 \leq -5,$$
$$-2x_1 + 5x_2 \leq -10,$$
$$x_1 \geq 0, x_2 \geq 0.$$

Dani problem v program zapišemo na sledeč način:

```
c=[1/4;1/3];
ci=[0;0];
cs=[%inf;%inf];
A=[-5,-1;-2,-5];
b=[-5;-10];
[x,larg,f]=linpro(c,A,b,ci,cs,0);
```

Izpiše se rezultat:

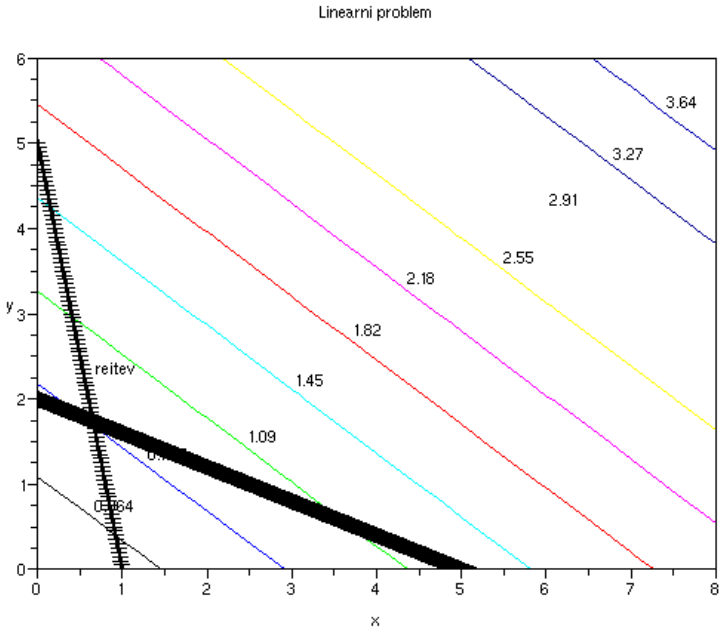
```
x =
   0.6521739
   1.7391304
```

Pri linearnem programiranju lahko rešitev prikažemo tudi grafično. Pri tem je treba biti pozoren, da ima funkcija le dve neodvisni spremenljivki, in sicer (x, y).

Poglejmo si še grafično rešitev zgoraj navedenega primera (slika 8).

```
deff('[w]=f(x,y)','w=c(1)*x+c(2)*y')
xx=[0:1:8];yy=[0:1:6];zz=feval(xx,yy,f);
contour(xx,yy,zz,10)
deff('[y1]=f1(x)','y1=-5*x+5')
deff('[y2]=f2(x)','y2=(10-2*x)/5')
xxx=[0:0.01:8];yy1=f1(xxx);yy2=f2(xxx);
contour(xx,yy,zz,10)
```

```
plot2d([xxx'xxx'],[yy1'  yy2'],[-1,-1],'011','',[0  0  8
6])
xstring(x(1)+0.05,x(2)+0.5,'rešitev')
xtitle('Linearni problem','x','y')
```



Slika 8: Grafična rešitev primera funkcije `linpro`

4.6 Kvadratno programiranje

Funkcija, ki jo uporabljamo za reševanje problemov kvadratnega programiranja, je `quapro`.

Quapro

Funkcija `quapro` rešuje probleme, pri katerih je kriterijska funkcija kvadratne oblike in je omejena z linearnimi omejitvami. Značilnost funkcije je, da išče minimum funkcije $f(x) = \frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{Q}\mathbf{x} + \mathbf{p}^{\mathrm{T}}\mathbf{x}$, z omejitvami:

a) $\mathbf{Cx} \leq \mathbf{b}$,

b) $\mathbf{Cx} \leq \mathbf{b}$, pri katerem velja: $\mathbf{c}_{\mathrm{i}} \leq \mathbf{x} \leq \mathbf{c}_{\mathrm{s}}$,

c) $\mathrm{C(j,:)x} = \mathrm{b(j)}, \qquad \mathrm{j} = 1, \ldots, \mathrm{me}$

$\mathrm{C(j,:)x} \leq \mathrm{b(j)}, \qquad \mathrm{j} = \mathrm{me}+1, \ldots, \mathrm{me}+\mathrm{md}$, za x velja: $c_i \leq x \leq c_s$.

Funkcijo zapišemo na štiri načine. Najpreprostejša inačica ukaza je:

`[x,larg, f]=quapro(Q,c,A,b[,x0])`.

Poleg tega ukaza se pogosto uporablja še naslednjo inačico:

- `[x,larg,f]=quapro(Q,c,A,b,ci,cs,me,x0[,imp])`

`Q` – kvadratna, simetrična matrika ($\mathbf{Q} = \mathbf{Q}^{\mathrm{T}}$), dimenzije ($n$ x n),

`c` – vektor, dimenzije (n), če ni omejitev je `c=[]`,

`A` – matrika, dimenzije ($me + md$) x n,

`b` – vektor, dimenzije ($me + md$),

`me` – število omejitvenih enakosti,

`x0` – začetna vrednost ali niz znakov,

`imp` – parameter za opozorila.

Primer: Izračunaj minimum funkcije: $f(x_1, x_2) = x_1^2 - x_1x_2 + \frac{3}{2}x_2^2 + 2x_1 + 4x_2$,

ki je omejena z:

$$-2x_1 - 3x_2 \leq -5,$$
$$-5x_1 - x_2 \leq -1,$$
$$x_1 \geq 0, x_2 \geq 0.$$

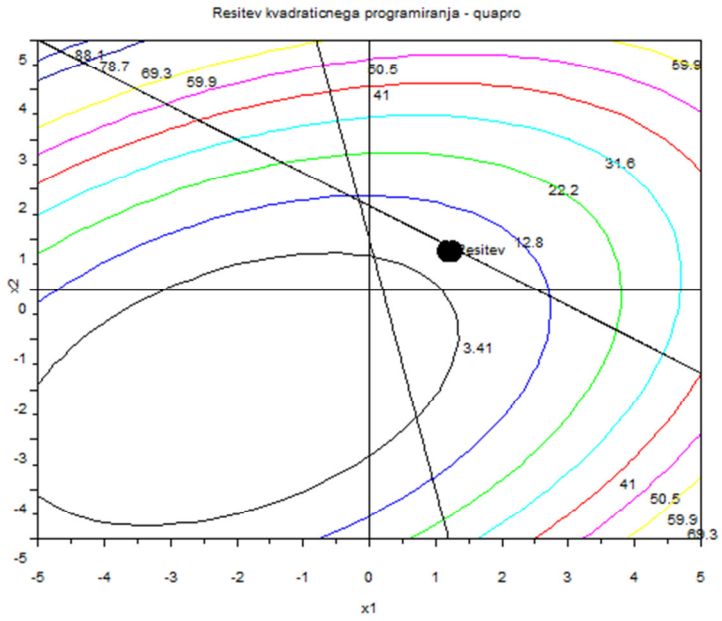
Funkcijo in omejitve zapišemo v matrični obliki:

$$\mathbf{Q} = \begin{bmatrix} 2 & -1 \\ -1 & 3 \end{bmatrix}, \mathbf{c} = \begin{bmatrix} 2 \\ 4 \end{bmatrix}, \mathbf{A} = \begin{bmatrix} -2 & -3 \\ -5 & -1 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} -5 \\ -1 \end{bmatrix}, \mathbf{c}_i = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \mathbf{c}_s = \begin{bmatrix} \infty \\ \infty \end{bmatrix}.$$

```
Q=[2,-1;-1,3];
c=[2;4];
A=[-2,-3;-5,-1];
b=[-5;-1];
ci=[0;0];
cs=[%inf;%inf];
[x,larg,f]=quapro(Q,c,C,b,ci,cs);
```

Rezultat in grafična rešitev problema (slika 9):

```
x0 =
     1.2142857
     0.8571429
```



Slika 9: Grafična rešitev primera za funkcijo `quapro`

Iz grafa lahko domnevamo, da je absolutni minimum funkcije nekje v tretjem kvadrantu, $x_1 < 0$ in $x_2 < 0$. Možno področje se nahaja v prvem kvadrantu. Optimalna rešitev je označena s točko »rešitev« v grafu.

4.7 Nelinearno programiranje

Programski paket ScicosLab vsebuje funkcije, ki računajo minimalno vrednost x nelinearnim funkcijam $f(x)$, katerih spremenljivke so omejene s spodnjo in z zgornjo mejo. Funkcije, ki jih uporabljamo pri nelinearnem programiranju, so `optim`, `datafit`, `fsolve`, `lsqrsolve` in `leastsq`. Poleg omenjenih funkcij uporabljamo še pomožne funkcije: `derivative`, `NDcost` in `numdiff`.

Datafit

`datafit` uporabljamo, ko želimo za nek model poiskati optimalne koeficiente vnaprej določenih poljubnih enačb z definiranimi omejitvami. Funkcija `datafit` najde na dano funkcijo $\mathbf{G} = (\mathrm{p}, \mathrm{z})$ vektor, ki vsebuje parameter p. Vektor p poiščemo z minimiziranjem funkcije

$$\sum_{i=1}^{n} \mathbf{G}(p, z_i)^T \mathbf{W} \mathbf{G}(p, z_i).$$

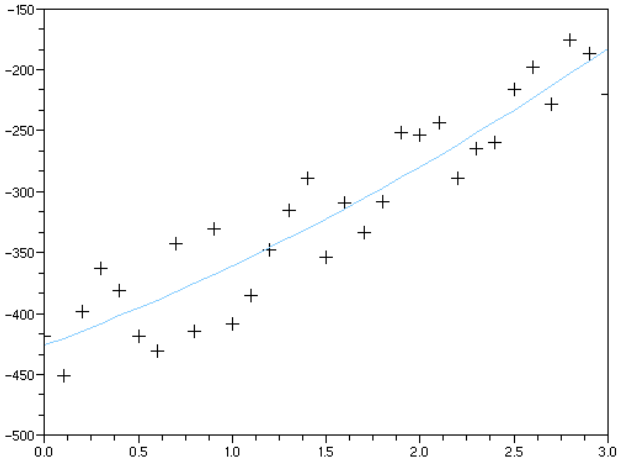
Spodaj je prikazan primer uporabe funkcije `datafit` ter grafični prikaz rešitve (slika 10).

Primer: Dobiti želimo optimalne koeficiente dane funkcije $f(x) = a(x - b + cx)$

```
function y=FF(x,p),y=p(1)*(x-p(2))+p(3)*x.*x,endfunction
X=[];Y=[];
pg=[34;12;14]          /generiranje podatkov s parametrom
for x=0:.1:3, Y=[Y,FF(x,pg)+100*(rand()-.5)];X=[X,x];end
Z=[Y;X];

                                        // kriterijska funkcija
function e=G(p,z)
y=z(1),x=z(2);
e=y-FF(x,p)
endfunction

                                              //rešitev problema
p0=[3;5;10]
[p,err]=datafit(G,Z,p0);

                                              // graf funkcije
[p,err]=datafit(G,Z,p0);
scf(1);clf()
plot2d(X,Y,-1)
plot2d(X,FF(X,p),12)
```

Rešitev:

```
pg =
    34.
    12.
    14.

p0 =
    3.
    5.
    10.
```



Slika 10: Grafični prikaz rešitve pri uporabi funkcije `datafit`, kjer so prikazani (+) in optimalna funkcija (polna črta)

Fsolve

Za iskanje ničle sistema, ki je sestavljen z n nelinearnimi funkcijami, z n spremenljivkami in omejitvami, uporabimo ScicosLab-ovo funkcijo `fsolve`, ki za algoritem uporablja Powel hibridno metodo, ki je omenjena v poglavju 3.3.2. Za pomoč lahko uporabimo še Jacobijevo matriko.

Primer: Za ilustracijo poglejmo, kako s funkcijo `fsolve` minimiziramo naslednjo funkcijo: $f(x,y,z) = (x-z)^2 + 3(x+y+z-1)^2 + (x-z+y)^2$, ki je omejena z $g(x,y,z) = 1 - x - 3y - z^2 = 0$.

V program zapišemo naslednji zapis:

```
function z=myf(x)
z=(x(1)-x(3))^2+3*(x(1)+x(2)+x(3)-1)^2+(x(1)-x(3)+1)^2
endfunction
function z=fct(x)
w1=[2*(x(1)-x(3))+6*(x(1)+x(2)+x(3)-1)+2*(x(1)-
x(3)+1),6*(x(1)+x(2)+x(3)-1),-2*(x(1)-
x(3))+6*(x(1)+x(2)+x(3)-1)-2*(x(1)-x(3)+1)]      //prvi
                                  odvod funkcije po x, y, z
w2=[-1 -3 -2*x(3)]    //funkcijo ∇g zapišemo kot matriko
z=[w1'+x(4)*w2';1-x(1)-3*x(2)-x(3)^2]
endfunction
x0=[0 0 0 0];
[x,v]=fsolve(x0,fct);
```

in dobimo rezultat:

```
v =
1.0D-16 *
0.0016752    0.0050255    0.0023359  - 1.6653345

x =
0.1972244    0.1055513    0.6972244  - 1.675D-19
```

Komponenta `x(4)` predstavlja spremenljivko λ enačbe (38).

Optim

Funkcijo `optim` uporabljamo za nelinearno optimizacijo pri problemih, ki so omejeni ali neomejeni. Algoritem funkcije išče optimalno rešitev s Quasi-Newtonovo metodo. Zapis za metodo Quasi-Newton se glasi tako: `[f,xopt]=optim(costf,x0,'qn')`

Namesto zgoraj omenjene metode, lahko pri iskanju optimalne rešitve algoritem funkcije `optim` računa po metodi konjugiranih gradientov '`gc`' ali po metodi neodvedljivosti (ang. non-differentiable) '`nd`', ki se uporablja v primeru, ko obstaja taka točka kriterijske funkcije, v kateri ne moremo določiti odvoda. Tako pri zapisu ukaza funkcije `optim` zapišemo argument '`qn`', '`gc`' ali '`nd`', ki določa, po kateri metodi naj algoritem izračuna optimalno vrednost.

Če želimo, da nam algoritem pri določitvi vrednosti spremenljivk `xopt`, kjer kriterijska funkcija doseže optimum, izračuna še gradient funkcije, ukaz zapišemo na naslednji način: `[f,xopt,gopt]=optim(costf,x0)`.

Derivative

Naloga pomožne funkcije `derivative` je aproksimacija odvoda funkciji f. Prvi in drugi odvod poskušamo približati funkciji ($\mathbb{R}^n \to \mathbb{R}^m$) pri točki x. Enako metodo uporabimo pri izračunu Jacobijeve matrike. Funkcija Jacobijevo matriko izračuna s približevanjem smernega odvoda komponent funkcije f v smeri stolpca matrike **Q.** Drugi odvod funkcije izračuna glede na sestavo prvega odvoda. Če je dana matrika **H** v privzeti obliki Taylorjeve vrste funkcije f, je pogoj drugega odvoda dan v obliki:

$$F(x + dx) = F(x) + \mathbf{J}(x)dx + 1/2\,\mathbf{H}(x)(dx.*.dx) + \ldots$$

V splošnem velja, da je numerično približevanje odvoda nestabilen proces. Korak približevanja h mora biti majhen, da dobimo čim manjšo napako. Toda če bomo izbrali premajhen h, bo numerična napaka prevelika. Temu se izognemo tako, da ne spreminjamo privzete vrednosti koraka.

Ukaz za `derivative` zapišemo na naslednji način: `[g,H]=derivative(f,x)`, `g` predstavlja gradient funkcije f in `H` Hessejevo matriko funkcije f.

NDcost

`NDcost` se uporablja kot »zunanja« funkcija `optim` pri minimiziranju problema, pri katerem je izračunani gradient preveč zapleten. Funkcija z uporabo končnih razlik izračuna gradient funkcije.

Primer: Iščemo minimum funkcije $f(x,y) = 100(y-x^2)^2 + (1-x)^2$.

```
function f=rosenbrock(x,varargin)
p=varargin(1)
f=1+sum( p*(x(2:$)-x(1:$-1)^2)^2 + (1-x(2:$))^2)
endfunction
x0=[1;2;3;4];
[f,xopt,gopt]=optim(list(NDcost,rosenbrock,200),x0)
```

Rešitev:

```
gopt =
    0.0000002
    0.0000002
  - 1.415D-08
  - 5.108D-08

xopt =
    1.
    1.
    1.
    1.0000000

f =
    1.
```

Numdiff

`numdiff` izračuna numerično oceno velikosti gradienta. Dana funkcija $fun(x)$ iz $\mathbb{R}^n$ v $\mathbb{R}^m$ izračuna matriko: $\mathbf{g}_{ij} = \left[\frac{df_i}{dx_j}\right]$, ki uporablja metodo končnih razlik.

Za `g=numdiff(f,x)` velja, da: a) če $f: \mathbb{R}^n \to \mathbb{R}$, potem je `g` gradient pri `x` in

b) če $f: \mathbb{R}^n \to \mathbb{R}^m$, potem je `g` Jacobijeva matrika, velikosti mxn.

Primer: Primerjava numerične in analitične vrednosti odvoda funkcije

$$f(x) = (x + a)^b + c$$

```
function
f=myfun(x,y,z,t)
f=(x+y)^t+z
endfunction

y=3;z=4;t=2;
g=numdiff(list(myfun,3,4,2),1)

x=1;
exact=t*(x+y)^(t-1)
```

Rezultat:

```
g =
    7.9999999

exact =
    8.
```

Med nelinearno programiranje spada tudi metoda najmanjših kvadratov. Funkciji, ki ju uporabljamo pri tej metodi, sta `leastsq` in `lsqrsolve`.

Leastsq

Funkcija rešuje nelinearne probleme z minimizacijo najmanjšega kvadrata napake. *fun* je funkcija iz $\mathbb{R}^n$ v $\mathbb{R}^m$, s katero skušamo minimizirati funkcijo $f(x) = \|fun\|^2 = \sum_{1=1}^{n} fun_i^2 x$, ki predstavlja vsoto vseh kvadratov komponent funkcije *fun*. Mejne vrednosti vplivajo predvsem na x. Ukaz za funkcijo `leastsq` zapišemo: `[fopt,[xopt,[grdopt]]]= leastsq(fun,x0).`

`fopt` – vrednost funkcije $f(x) = \|fun\|^2$, pri `xopt`,

`xopt` – najboljša vrednost, ki jo doseže x pri minimiziranju $\|fun\|^2$,

`grdopt` – gradient funkcije pri `xopt`,

`fun` – funkcija ScicosLab-a, ki definira funkcijo iz $\mathbb{R}^n$ v $\mathbb{R}^m$.

`x0` – vektor realnih števil.

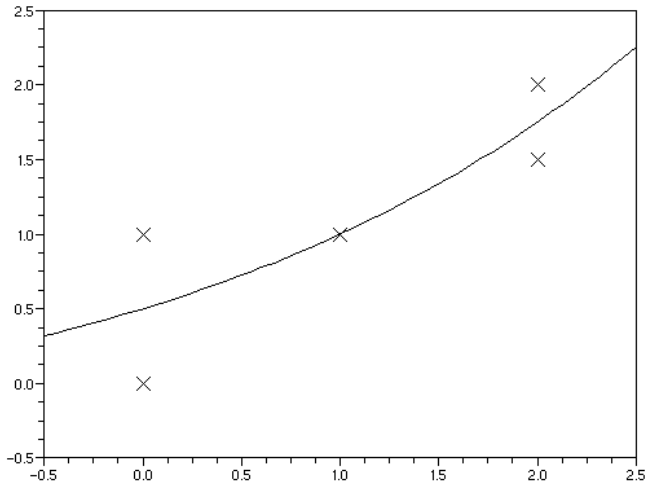
Primer: Dane imamo točke $\{(0,0),(0,1),(1,1),(2,1.5),(2,2)\}$. Poiskati želimo take parametre a, b in c, ki se najbolj prilegajo oziroma ujemajo s funkcijo $f = ae^{bx} + c$, po kriteriju najmanjših kvadratov.

Neznane parametre zapišemo v obliki $[p_1, p_2, p_3] = (a, b, c)$ in točke (x_i, y_i), nato rešimo pet enačb oblike: $y_i - p_1 e^{p_2 x_i} - p_3 = 0$, po kriteriju najmanjših kvadratov.

```
DAT=[0 0;0 1;1 1;2 1.5;2 2];
function z=fun(p)
DAT=[0 0;0 1;1 1;2 1.5;2 2]
z=DAT(:,2)-p(1)*exp(p(2)*DAT(:,1))-p(3)
endfunction
p0=[0,0,0];                              //začetna vrednost
[ff,p]=leastsq(fun,p0)

// drugi del: graf
xbasc();
plot2d(DAT(:,1),DAT(:,2),-3,rect=[-0.5,-0.5,2.5,2.5])
t=linspace(-0.5,2.5,50);
et=p(1)*exp(p(2)*t)+p(3);
plot2d(t,et,rect=[-0.5,-0.5,2.5,2.5])
```

Rešitev:

```
p =
    0.9999993    0.4054653  - 0.4999993

ff =
    0.625
```



Slika 11: Točke in krivulja, ki jo najde funkcija `leastsq`

Lsqrsolve

Funkcija `lsqrsolve` poišče minimum vsote kvadratov m nelinearne funkcije z n spremenljivkami, z algoritmom Levenberg-Marquard. Uporabnik določi kriterijsko funkcijo, sprogramirano kot ScicosLab funkcija. Jacobijeva matrika se potem takem izračuna s približevanjem odvoda.

Tako minimiziramo `sum(fct(x,m).^2)`, kjer je `fct` funkcija iz $\mathbb{R}^n$ v $\mathbb{R}^m$.
`lsqrsolve` zapišemo na naslednji način:
`[x[,v[,info]]]=lsqrsolve(x0,fct,m[,stop[diag]]).`

`fct` – zaporedje zapisa `v=fct(x,m)` pri danem `x` in `m`,
`fjac` – zunanja funkcija, ki nam da `v=d(fct)/dx(x)`, ki ga v program zapišemo `J=fjac(x,m)` pri danem `x` in `m`.

Druge funkcije

Poseben primer so optimizacijski problemi v obliki neenakosti linearnih matrik (LMI), za katere uporabljamo funkcijo `lmisolver` in posebno orodje `lmitool`. `lmisolver` rešuje probleme s področja LMI, pri katerih x_i ni matrika, temveč seznam matrik. Za določanje medsebojnih LMI problemov uporabimo `lmitool`.

Poleg naštetega omogoča program ScicosLab tudi semidefinitno programiranje, ki je področje konveksne optimizacije in zajema optimizacijo kriterijske funkcije, pri kateri so spremenljivke definirane v obliki pozitivno semidefinitnih matrik. Za reševanje tega problema uporabimo funkcijo `semidef`.

4.8 Primeri

a) **Primer linearnega programiranja** (Nemanič, 2009)

Poiskati moramo optimalno sestavo zmesi, ki mora vsebovati najmanj 30 % komponente A. V skladišču se nahajajo štiri vrste surovin, ki vsebujejo 20 %, 40 %, 15 % in 25 % komponente A. Posebna zahteva je, da zmes ne sme vsebovati več kot 30 % surovine S_1 in ne več kot 50 % surovine S_2 in S_3 skupaj. Nabavne cene za tono surovine so: 3000 de za S_1, 2500 de za S_2, 2800 de za S_3 in 1600 de za S_4. Določiti moramo količine surovin, ki jih lahko nabavimo z minimalnimi stroški.

Določimo kriterijsko funkcijo:

$$minf(x) = 3000x_1 + 2500x_2 + 2800x_3 + 1600x_4$$

z mejnimi pogoji:

$$0{,}2x_1 + 0{,}4x_2 + 0{,}15x_3 + 0{,}25x_4 \geq 0,$$
$$x_1 + x_2 + x_3 + x_4 = 1,$$
$$x_1 \leq 0{,}3\ ,$$
$$x_2 + x_3 \leq 0{,}5 \text{ in } x_1, x_2, x_3, x_4 \geq 0$$

ScicosLab koda s komentarji:

```
c=[3000;2500;2800;1600];        //koeficienti funkcije f(x)
ci=[0;0;0;0];                                //spodnja meja
cs=[%inf;%inf;%inf;%inf];                    //zgornja meja
C1=[1,1,1,1];                    //levi del pogojne enačbe
b1=[1];                         //desni del pogojne enačbe
C2=[-0.2,-0.4,-0.15,-0.25;1,0,0,0;0,1,1,1,0];     //levi del
                                                pogojnih neenačb
b2=[-0.3;0.3;0.5]              //desni del pogojnih neenačb
C=[C1;C2];
b=[b1;b2];
me=1;                  //predstavlja število pogojnih enačb
x0='v';                                  // teme možne točka
[x,lagr,f]=linpro(c,C,b,ci,cs,me,x0);
```

```
x             //izpis vrednost vektorja, ki minimizira f(x)
lagr           //izpis vektorja Lagrangovega multipleksorja
f                          //izpis optimalna vrednost f(x)
```

Rešitev:

```
x =
  - 2.328D-17
    0.3333333
    7.214D-18
    0.6666667

lagr =
  - 1700.
    0.
  - 1800.
    0.
  - 100.
    6000.
    0.
    0.

f =
    1900.
```

Za minimalne stroške nabave, 1900 denarnih enot, bomo potrebovali 0,333 tone surovine S_2 in 0,667 tone surovine S_4.

b) Primer kvadratnega programiranja

Na vrtu bi radi imeti bazen, ki ga želimo zgraditi z minimalnimi stroški. Stroški so določeni s funkcijo $f(x) = (x_1 - 4)^2 + (2x_2 - 4)^2 - 16$. Glede na velikost vrta imamo omejitve pri velikosti bazena. Maksimalna dolžina bazena je lahko 3 m, dolžina in širina bazena pa ne smeta biti večji od 5 m. Koliko znašajo dimenzije bazena, da bodo stroški izgradnje čim manjši?

Se pravi, da iščemo minimum funkcije:

$$f(x) = (x_1 - 4)^2 + (2x_2 - 4)^2 - 16\ ,$$

ki je omejena s pogoji:

$$x_1 + x_2 \leq 5,$$
$$x_1 \leq 3,$$

in za x_1 (dolžina) in x_2 (širina) velja: $x_1, x_2 \geq 0$.

Funkcije in omejitve zapišemo v matrični obliki.

Matriko $\mathbf{Q} = \begin{bmatrix} 2 & 0 \\ 0 & 8 \end{bmatrix}$ dobimo iz gradienta funkcije $f(x)$, $\mathbf{c} = \begin{bmatrix} -8 \\ -16 \end{bmatrix}$ predstavlja linearni del iskane funkcije $f(x)$. Levo stran neenačb smo zapisali z matriko $\mathbf{A} = \begin{bmatrix} 2 & 0 \\ 0 & 8 \end{bmatrix}$, desno stran pa z $\mathbf{b} = \begin{bmatrix} 5 \\ 3 \end{bmatrix}$. Spodnja meja je označena s c_i in s c_s zgornja meja.

ScicosLab koda s komentarji:

```
Q=[2,0;0,8];                    //matrika Q, gradient f(x)
c=[-8;-16];                  //matrika c, linearni del f(x)
A=[1,1;1,0];                //matrika A, leva stran neenačb
b=[5;3];                   //matrika b, desna stran neenačb
ci=[0;0];                                  //spodnja meja
cs=[%inf;%inf];                            //zgornja meja
[x,lagr,f]=quapro(Q,c,A,b,ci,cs)
```

Rešitev:

```
f =                            //optimalna vrednost f(x)
  - 31.

lagr =
    0.
    0.
    0.
    2.

x =                     //izračunana optimalna rešitev
    3.                                //dolžina bazena
    2.                                 //širina bazena
```

Funkcija $f(x)$ bo imela najmanjšo vrednost takrat, ko sta $x_1 = 3$ in $x_2 = 2$.

c) Primer nelinearnega programiranja (Nemanič, 2009)

Podjetje načrtuje, da bo v treh mesecih izdelalo 980 enot proizvoda A. Stroški proizvodnje so vsak mesec različni in znašajo:

1. mesec: $f_1(x_1) = 0{,}02x_1^2 + 0{,}8x_1$
2. mesec: $f_2(x_2) = 0{,}04x_2^2 + 60$
3. mesec: $f_3(x_3) = 0{,}03x_3^2 + 0{,}06x_3 - 10$

x_1, x_2, x_3 – število enot proizvoda A, narejenih v enem mesecu.

Poiskati moramo tak program proizvodnje, se pravi število x_1, x_2, x_3, da bodo stroški trimesečne proizvodnje minimalni.

Iščemo funkcijo:

$$minf(x) = f_1(x_1) + f_2(x_2) + f_3(x_3)$$
$$minf(x_1, x_2, x_3) = 0{,}02x_1^2 + 0{,}8x_1 + 0{,}04x_2^2 + 60 + 0{,}03x_3^2 + 0{,}06x_3 - 10$$

z mejnimi pogoji:

$$g(x_1, x_2, x_3) = x_1 + x_2 + x_3 = 980 \text{ in } x_1, x_2, x_3 \geq 0.$$

Potrebni pogoj za izračun minimuma funkcije je:

$$\nabla f + \lambda \nabla g = 0 \tag{45}$$
$$g = 0 \tag{46}$$

∇f je gradient funkcije $f(x)$, ki jo v programu ScicosLab zapišemo z `w1`, in ∇g je gradient funkcije $g(x)$, ki je zapisan s funkcijo `w2`. S funkcijo `z` sta zapisani enačbi (45) in (46) v matrični obliki.

ScicosLab koda s komentarji:

```
function z=fct(x)
w1=[0.04*x(1)+0.8,0.08*x(2),0.06*x(3)+0.6]   //gradient f(x)
                            po komponentah x(1), x(2) in x(3)
w2=[1 1 1]                   //mejni pogoji v matrični obliki
z=[w1'+x(4)*w2';x(1)+x(2)+x(3)-980]    //zapis f(x) funkcije
                in mejnega pogoja v obliki enačb (45) in /46)
endfunction

x0=[0,0,0,0];                        //vektor začetne vrednosti
[x,v]=fsolve(x0,fct);       //sklic za iskanje ničle sistema
                                        nelinearne funkcije
x                                    //optimalna rešitev f(x)
```

Rešitev:

```
x =
   446.15385    233.07692    300.76923  - 18.646154
```

Da bodo stroški podjetja minimalni, mora podjetje v prvem mesecu izdelati 446 enot, v drugem 233 enot in v tretjem 301 enot proizvoda A.

5 ZAKLJUČEK

Optimizacijske probleme v splošnem formuliramo tako, da moramo določiti njihov ekstrem, ki je lahko minimum ali maksimum kriterijske funkcije neodvisnih spremenljivk. Pri tem si pomagamo z metodami neomejene ali omenjene optimizacije v obliki linearnega, nelinearnega in kvadratnega programiranja. Reševanje raznovrstnih optimizacijskih problemov omogoča odprtokodni programski paket ScicosLab, ki vsebuje funkcije, s katerimi rešimo problem s prej omenjenimi metodami.

Namen tega diplomskega dela je bil narediti sistematični prikaz, katere funkcije s področja optimizacije vsebuje programski paket ScicosLab, in razdelitev funkcij po skupinah glede na obliko kriterijske funkcije. V nalogi so predstavljene splošne značilnosti danih funkcij, njihove razlike in povezave med njimi. Uporabnikom so najprej predstavljene enostavnejše funkcije in na koncu zahtevnejše. Njihova uporaba je prikazana v splošnih in posebnih primerih. Priporočljivo je, da uporabnik preden začne reševati optimizacijske probleme, pozna vsaj osnovne značilnosti programskega orodja ScicosLab. Zahtevnejši uporabniki ScicosLab-a lahko na spletnih straneh najdejo dovolj dodatnega gradiva in literature.

Program ScicosLab je zelo prijazen do uporabnika. V kolikor se pojavijo težave pri zapisu sintakse, nas program hitro opomni in s sprotno pomočjo ScicosLab Help lahko hitro preverimo, kje smo naredili napako. Glede na to, da drugih programov, s katerimi rešujemo optimizacijske probleme, nisem preizkusila, ne morem oceniti, kaj so prednosti in slabosti tega programa.

Ker je področje uporabe programskega paketa dokaj zanimivo, bi želela, da bi bilo v literaturi, ki opisuje programski paket ScicosLab, predstavljenih več primerov iz prakse. Upam, da bo to diplomsko delo pomagalo k bolj razširjeni in lažji uporabi orodja ScicosLab za reševanje optimizacijskih problemov.

6 LITERATURA

Bajc, A. (2007). Uporaba programskega modula Scicos za gospodarskega inženirja, Diplomsko delo. (Poslovno-tehniška fakulteta, Univerza v Novi Gorici), Nova Gorica: [A. Bajc].

Bronštejn, I. N., Semendjajev, K. A., Musiol, G., Muhling, H. (2009). Matematični priročnik. Ljubljana: Tehniška založba slovenije.

Campbell, S. L., Chancelier, J-P., Nikoukhah, R. (2006). Modeling and Simulation in Scilab/Scicos. New York: Springer.

Jereb, B., Skok, D., Šafran, M., Škornik, M. (2010). Programi za logistike, Pridobljeno 10.11. 2011 s svetovnega spleta: www.labinf.uni-mb.si/4L/poglavja/

Kravanja, Z., Novak Pintarič, Z. (2006). Optimiranje procesov. Zbirno gradivo. (Fakulteta za kemijo in kemijsko tehnologijo, Univerze v Mariboru). Pridobljeno 12.6.2011 s svetovnega spleta:

Leksikon Cankarjeve založbe. (1988). Ljubljana: Cankarjeva založba.

Makarovič, M. (2005). Priročnik programskega paketa Scilab za uporabnika začetnika. Diplomsko delo. (Poslovno-tehniška fakulteta, Univerza v Novi Gorici), Nova Gorica: [M. Makarovič].

Matko, D., Bratkovič, F. (1993). Računalniško inženirstvo v vodenju sistemov. Ljubljana: Fakulteta za elektrotehniko in računalništvo.

Nemanič, J. (2009) Metode optimiranja proizvodnje. Diplomsko delo. (Fakulteta za strojništvo, Univerze v Mariboru), Maribor: [J. Nemanič].

Optimimizacija. Pridobljeno 12.06.2012 s svetovnega spleta: http://sl.wikipedia.org/wiki/Optimizacija

Osojnik, M. (2009). Primerjava analitičnih programskih orodij pri reševanju problemov v poslovnih procesih. Diplomsko delo. (Fakulteta za organizacijske vede, Univerze v Mariboru), Kranj: [M. Osojnik].

Petrišič, J. (2011). Uvod v Matlab. Ljubljana: Fakulteta za strojništvo.

Ravindran, A., Ragsdell, K. M., Reklatis, G. V. (2006). Engineering optimization: metods and applications. 2 izdaja. New Jersey: John Wiley & Sons

ScicosLab. Pridobljeno 10.10.2011 s svetovnega spleta: http://en/wikipedia/wiki/ScicosLab

ScicosLab. Pridobljeno 20.4.2011 s svetovnega spleta: www.scicoslab.org

Srebrnič V. (2011). Optimizacija oskrbe z zelenjavnimi živili z uporabo linearnega programiranja. Magistrsko delo. (Poslovno-tehniška fakulteta, Univerza v Novi Gorici), Nova Gorica: [V. Srebrnič].