



**KOMPUTASI FUNGSI NILAI DISTORSI DENGAN  
PEMROGRAMAN GEOMETRIK**

**SKRIPSI**

Asal:	Hadiah	Klass
Terima Tgl :	Pembelian 18 JUL 2007	S16
No. Induk :		SAF
KLASIR / PENYALIN :	<i>fa</i>	h
Oleh		e.1

**Muhammad Safak**  
031810101078

**JURUSAN MATEMATIKA  
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM  
UNIVERSITAS JEMBER  
2007**



**KOMPUTASI FUNGSI NILAI DISTORSI DENGAN  
PEMROGRAMAN GEOMETRIK**

**SKRIPSI**

diajukan guna melengkapi tugas akhir dan memenuhi salah satu syarat  
untuk menyelesaikan Program Studi Matematika (S1)  
dan mencapai gelar sarjana Sains

Oleh

**Muhammad Safak**

**031810101078**

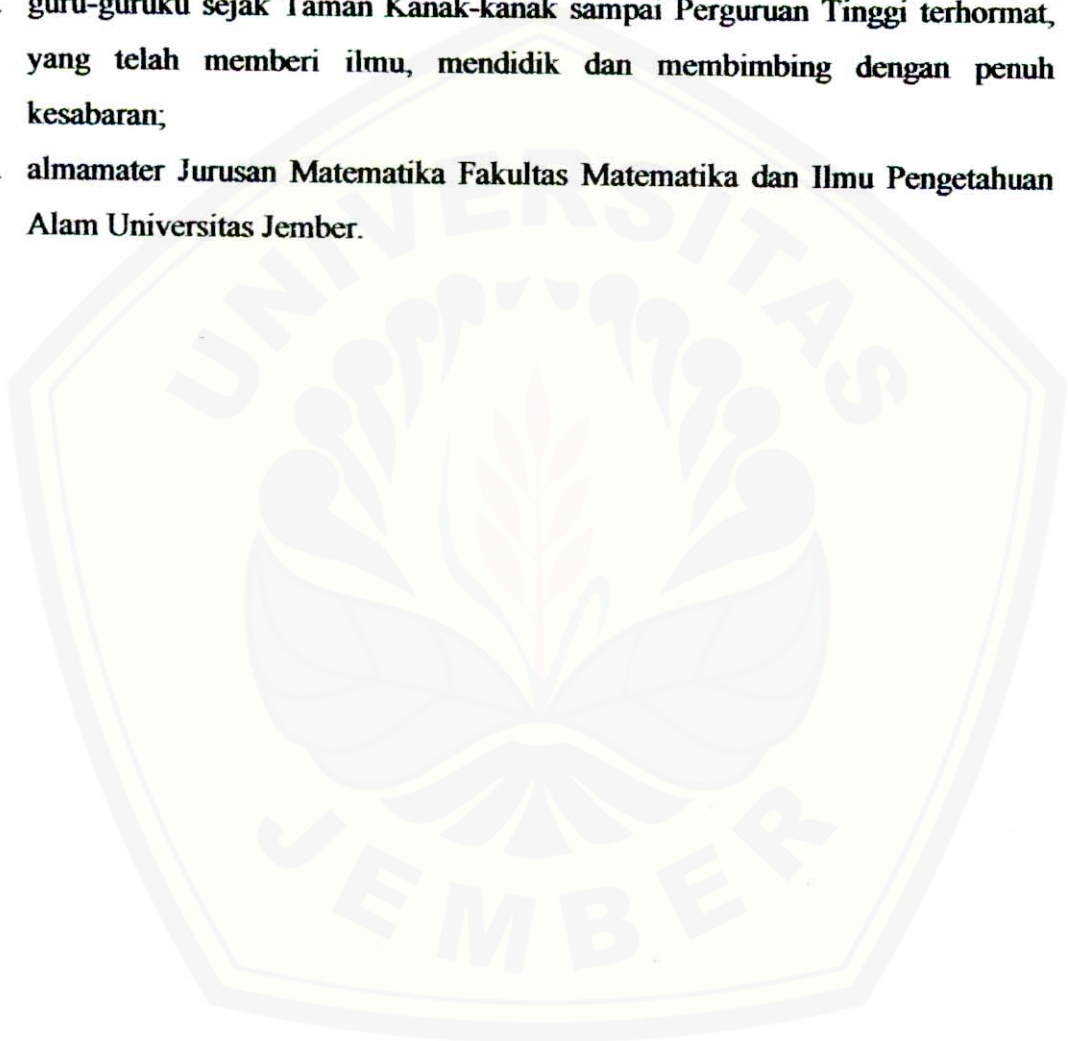
**JURUSAN MATEMATIKA  
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM  
UNIVERSITAS JEMBER**

**2007**

**PERSEMBAHAN**

Skripsi ini saya persembahkan untuk :

1. Bapak Supardi dan Ibu Suparmi terhormat, tersayang dan tercinta, yang telah mendoakan dan memberi segala kasih sayang serta pengorbanan selama ini;
2. guru-guruku sejak Taman Kanak-kanak sampai Perguruan Tinggi terhormat, yang telah memberi ilmu, mendidik dan membimbing dengan penuh kesabaran;
3. almamater Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.



**MOTTO**

Allah akan meninggikan orang-orang yang beriman di antara kamu dan orang-orang yang diberi ilmu pengetahuan beberapa derajat.

*(Terjemahan Surat Al-Mujadalah Ayat 11)*

Memang penting untuk menjadi orang penting  
tapi lebih baik menjadi orang baik.

*(Ebet Kadarusman)*

Selalu berusaha menjadi nomor 1  
meski nomor 1 tidak pernah datang padaku

*(Muhammad Safak)*

**PERNYATAAN**

Saya yang bertanda tangan di bawah ini:

nama : Muhammad Safak

NIM : 031810101078

menyatakan dengan sesungguhnya bahwa skripsi yang berjudul: *Komputasi Fungsi Nilai Distorsi Dengan Pemrograman Geometrik* adalah benar-benar hasil karya sendiri, kecuali jika dalam pengutipan substansi disebutkan sumbernya, dan belum pernah diajukan pada institusi mana pun, serta bukan karya jiplakan. Saya bertanggung jawab atas keabsahan dan kebenaran isinya sesuai dengan sikap ilmiah yang harus dijunjung tinggi.

Demikian pernyataan ini saya buat dengan sebenarnya, tanpa adanya tekanan dan paksaan dari pihak mana pun serta bersedia mendapat sanksi akademik jika ternyata di kemudian hari pernyataan ini tidak benar.

Jember, Juni 2007

Yang menyatakan,



Muhammad Safak

NIM 031810101078

**SKRIPSI**

**KOMPUTASI FUNGSI NILAI DISTORSI DENGAN  
PEMROGRAMAN GEOMETRIK**

Oleh

**Muhammad Safak**

**NIM 031810101078**

**Pembimbing**

**Dosen Pembimbing Utama : Agustina Pradjaningsih, S.Si., M.Si.**

**Dosen Pembimbing Anggota : Ahmad Kamsyakawuni, S.Si.**

**PENGESAHAN**

Skripsi berjudul *Komputasi Fungsi Nilai Distorsi Dengan Pemrograman Geometrik* telah diuji dan disahkan oleh Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember pada:

Hari : **SENIN**

Tanggal : **09 JUL 2007**

Tempat : Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

Tim Penguji:

Ketua,



Agustina Pradjaningsih, S.Si., M.Si.

NIP. 132 257 933

Sekretaris,



Ahmad Kamsyakawuni, S.Si.

NIP. 132 206 038

Anggota Tim Penguji

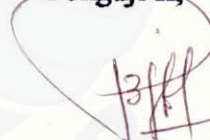
Penguji I,



Drs. Budi Lestari, PGD.Sc., M.Si.

NIP. 131 945 800

Penguji II,



Bagus Juliyanto, S.Si.

NIP. 132 304 782

Mengesahkan

Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam,



  
Ir. Sumadi, MS.

NIP. 130 368 784

## RINGKASAN

**Komputasi Fungsi Nilai Distorsi Dengan Pemrograman Geometrik;**  
Muhammad Safak, 031810101078; 2007; 26 halaman; Jurusan Matematika  
Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

Fungsi nilai distorsi merupakan salah satu dari konsep dasar teori informasi. Dalam mencari nilai optimasi dari fungsi nilai distorsi secara analitik dibutuhkan perhitungan yang rumit dan kompleks. Sehingga tujuan dari penelitian ini adalah mendapatkan nilai optimum dari fungsi nilai distorsi secara komputasi dengan mengubah fungsi nilai distorsi ke dalam bentuk konveks pemrograman geometrik terlebih dahulu, dengan menggunakan bantuan program komputer.

Penelitian diawali dengan proses instalasi jalur ggplab ke dalam MATLAB. Selanjutnya adalah membangun fungsi objektif dan fungsi kendala fungsi nilai distorsi pemrograman geometrik bentuk standar dengan input peluang kemunculan setiap huruf-huruf, simbol-simbol dan spasi ( $p$ ), kendala distorsi yang disebabkan adanya gangguan di dalam pengiriman ( $D$ ) dan distorsi dari kemunculan setiap huruf-huruf, simbol-simbol dan spasi ( $d$ ). Langkah selanjutnya adalah menghitung nilai optimum fungsi nilai distorsi dengan mencari nilai variabel optimum fungsi nilai distorsi pemrograman geometrik bentuk konveks terlebih dahulu. Hasil dari nilai variabel optimum fungsi nilai distorsi pemrograman geometrik bentuk konveks ditransformasi menjadi variabel optimum fungsi nilai distorsi pemrograman geometrik bentuk standar, sehingga nilai optimum fungsi nilai distorsi dapat ditentukan dengan memasukkan variabel optimum pemrograman geometrik bentuk standar ke dalam fungsi objektif fungsi nilai distorsi pemrograman geometrik bentuk standar. Untuk mengetahui nilai optimum fungsi nilai distorsi yang didapat adalah benar, yang harus dilakukan adalah mengetahui hasil pengiterasian dari nilai *duality gap*.



Hasil yang diperoleh dari penelitian ini bahwa yang mempunyai pengaruh terbesar terhadap perubahan nilai optimum fungsi nilai distorsi adalah input  $D$ , dengan  $0 \leq D \leq 1,9805$  memberikan nilai optimum fungsi nilai distorsi yang berbeda-beda dan mendekati 0 seiring dengan naiknya nilai  $D$ , sedang  $D > 1,9805$  memberikan nilai distorsi sebesar 0.



## PRAKATA

Puji syukur kehadirat Allah SWT atas segala rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan skripsi yang berjudul *Komputasi Fungsi Nilai Distorsi Dengan Pemrograman Geometrik*. Skripsi ini disusun untuk memenuhi salah satu syarat untuk menyelesaikan pendidikan strata satu (S1) pada Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

Penyusunan skripsi ini tidak lepas dari bantuan berbagai pihak, oleh karena itu penulis ingin menyampaikan ucapan terima kasih kepada:

1. Ibu Agustina Pradjaningsih, S.Si., M.Si., dan Bapak Ahmad Kamsyakawuni, S.Si., selaku Dosen Pembimbing yang telah meluangkan waktu, pikiran, dan perhatian dalam penulisan skripsi ini;
2. Bapak Drs. Budi Lestari, PGD.Sc., M.Si., dan Bapak Bagus Juliyanto, S.Si, selaku dosen penguji yang telah memberikan segala masukan buat terselesainya skripsi ini;
3. Bapak Mohammad Fatekurohman, S.Si., M.Si., selaku Dosen Pembimbing Akademik yang telah membimbing selama menjadi mahasiswa;
4. Bapak, Ibu, serta Kakak-kakaku sekeluarga yang telah memberikan dorongan dan doanya demi terselesaikannya skripsi ini;
5. Lia, Nyas, Titin, Azwar, Edy, Siro, Eni, Nora, Nyam, Dewi, semua rekan seangkatan, kakak dan adik angkatan yang telah memberi dorongan semangat;
6. Insan, Jumadi, Ricky, Masrur, Pak Topan sekeluarga, Zainul, Joko, Didin serta semua penghuni BT Club yang telah memberi warna dalam hidupku;
7. semua pihak yang tidak dapat disebutkan satu per satu.

Penulis juga telah menerima segala kritik dan saran dari semua pihak demi kesempurnaan skripsi ini. Akhirnya penulis berharap, semoga skripsi ini dapat bermanfaat.

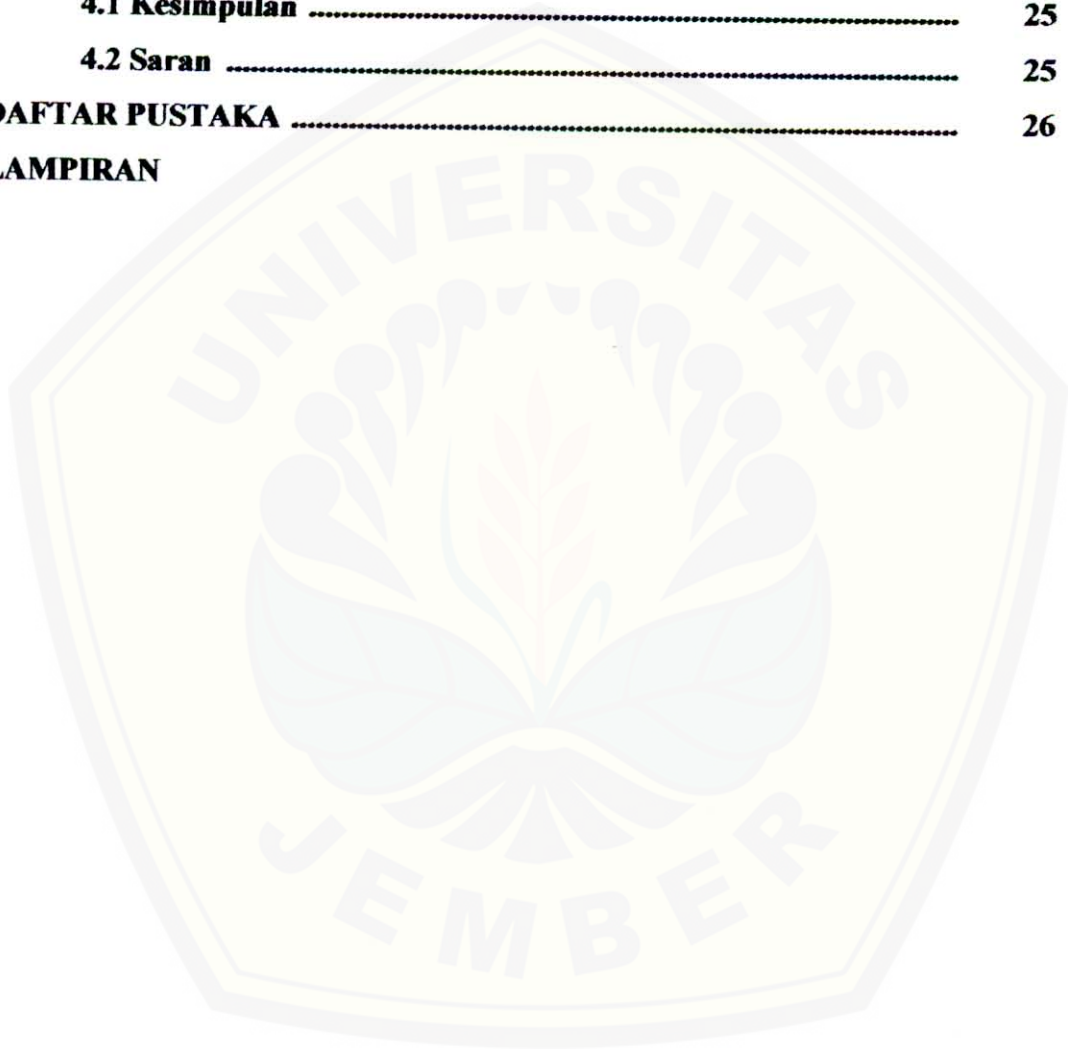
Jember, Juni 2007

Penulis

DAFTAR ISI

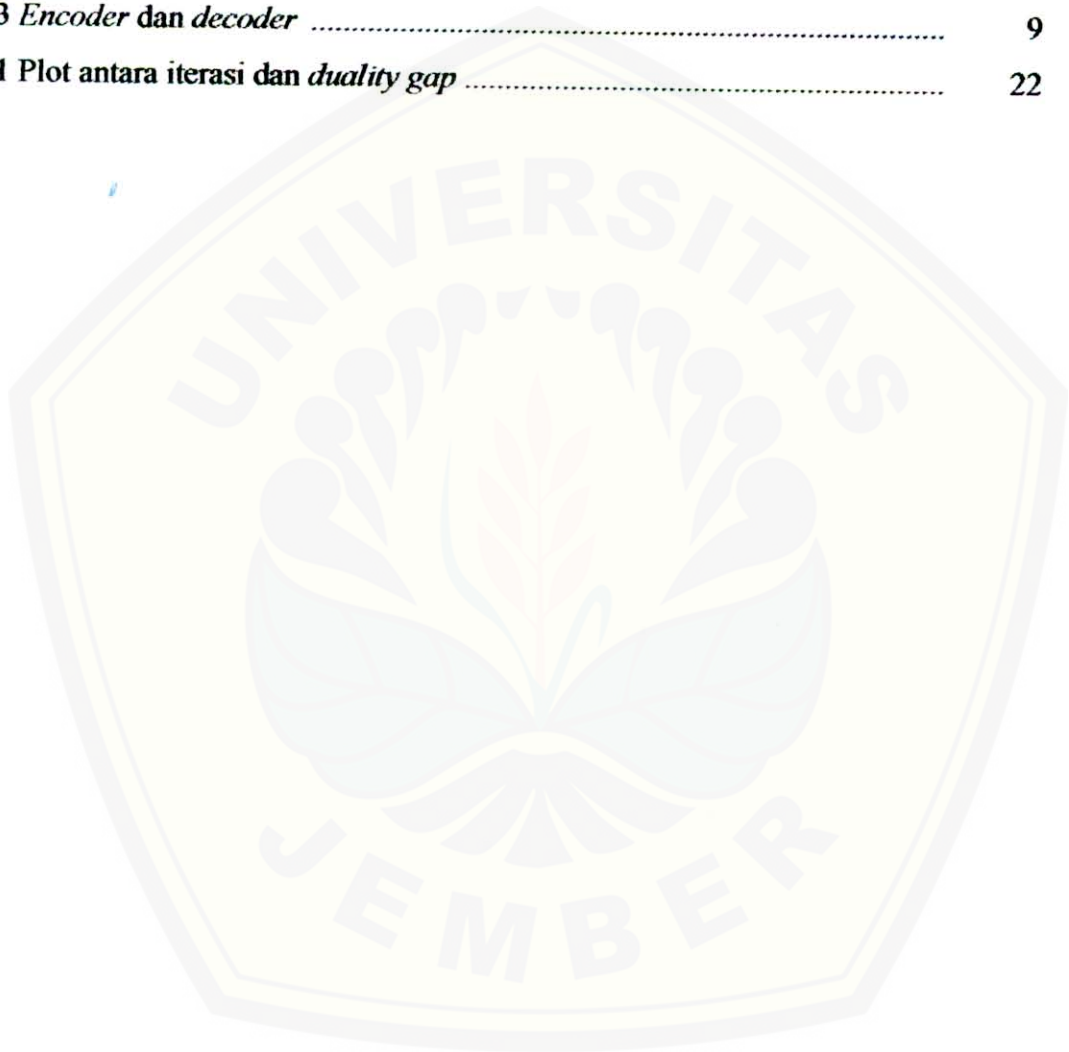
	Halaman
<b>HALAMAN JUDUL</b> .....	i
<b>HALAMAN PERSEMBAHAN</b> .....	ii
<b>HALAMAN MOTTO</b> .....	iii
<b>HALAMAN PERNYATAAN</b> .....	iv
<b>HALAMAN PEMBIMBINGAN</b> .....	v
<b>HALAMAN PENGESAHAN</b> .....	vi
<b>RINGKASAN</b> .....	vii
<b>PRAKATA</b> .....	ix
<b>DAFTAR ISI</b> .....	x
<b>DAFTAR GAMBAR</b> .....	xii
<b>BAB 1. PENDAHULUAN</b> .....	1
<b>1.1 Latar Belakang</b> .....	2
<b>1.2 Rumusan Permasalahan</b> .....	2
<b>1.3 Tujuan Penelitian</b> .....	2
<b>1.4 Manfaat Penelitian</b> .....	2
<b>BAB 2. TINJAUAN PUSTAKA</b> .....	3
<b>2.1 Himpunan Konveks dan Fungsi Konveks</b> .....	3
<b>2.2 Bentuk Umum Monomial dan Posinomial</b> .....	5
<b>2.3 Bentuk Umum Pemrograman Geometrik</b> .....	6
<b>2.4 Lagrange Dual Nilai Distorsi</b> .....	8
2.4.1 Masalah Fungsi Nilai Distorsi .....	8
2.4.2 Fungsi Nilai Distorsi Pemrograman Geometrik .....	10
<b>2.5 Algoritma dan Pemrograman</b> .....	11
<b>BAB 3. HASIL DAN PEMBAHASAN</b> .....	13
<b>3.1 Hasil</b> .....	13

3.1.1	Prosedur Membangun Fungsi Objektif dan Fungsi Kendala .....	13
3.1.2	Prosedur Menghitung Nilai Fungsi Distorsi .....	16
3.2	Pembahasan .....	20
<b>BAB 4.</b>	<b>KESIMPULAN DAN SARAN .....</b>	<b>25</b>
4.1	Kesimpulan .....	25
4.2	Saran .....	25
<b>DAFTAR PUSTAKA .....</b>		<b>26</b>
<b>LAMPIRAN</b>		



**DAFTAR GAMBAR**

	<b>Halaman</b>
2.1 Himpunan konveks dan konkaf .....	4
2.2 Fungsi konveks dan konkaf .....	5
2.3 <i>Encoder</i> dan <i>decoder</i> .....	9
3.1 Plot antara iterasi dan <i>duality gap</i> .....	22





## BAB 1. PENDAHULUAN

### 1.1 Latar Belakang

Pemrograman non linier merupakan alat analisis suatu masalah yang mempunyai variabel-variabel bersifat terukur (deterministik) dan setiap variabel-variabelnya mempunyai hubungan non linier satu dengan lainnya. Pada pemrograman non linier, fungsi tujuan dan fungsi kendala berbentuk non linier. Sampai saat ini pemrograman non linier terus berkembang, banyak ahli yang berusaha untuk menemukan berbagai teknik pemrograman non linier. Adapun salah satu kelas pemrograman non linier yang berkembang hingga saat ini adalah pemrograman geometrik.

Pemrograman geometrik pertama kali diperkenalkan pada tahun 1961 oleh Clarence Zener dan Richard Duffin. Pemrograman geometrik banyak digunakan untuk menyelesaikan berbagai macam permasalahan. Salah satunya adalah masalah sistem komunikasi. Masalah yang dibahas dalam sistem komunikasi antara lain arsitektur jaringan komunikasi, algoritma *processing* sinyal, teori informasi, optimasi sistem antrean dan beberapa alokasi jaringan (Chiang, tanpa tahun).

Dalam skripsi ini akan dibahas mengenai salah satu dari konsep dasar teori informasi yaitu fungsi nilai distorsi, lebih khusus lagi akan dibahas mengenai komputasi dari fungsi nilai distorsi. Thomas dan Cover (1991) menyatakan bahwa fungsi nilai distorsi bisa disebut juga fungsi informasi nilai distorsi. Fungsi informasi nilai distorsi untuk sumber  $X$  dengan fungsi distorsi  $d(x, \hat{x})$  didefinisikan sebagai  $R(D) = \min_{P \in \{d(x, \hat{x}) \leq D\}} I(X; \hat{X})$ .

Dalam penghitungan nilai optimum dari  $R(D)$  secara analitik dibutuhkan perhitungan yang rumit dan kompleks sehingga penulis tertarik untuk menghitung

nilai optimum menggunakan bantuan komputer dengan terlebih dahulu mengubah  $R(D)$  ke dalam bentuk konveks pemrograman geometrik.

### **1.2 Rumusan Permasalahan**

Berdasarkan pada latar belakang, permasalahan yang akan dibahas adalah bagaimana mendapatkan nilai optimum dari  $R(D)$  secara komputasi dengan terlebih dahulu mengubah  $R(D)$  ke dalam bentuk konveks pemrograman geometrik dengan menggunakan bantuan program komputer.

### **1.3 Tujuan Penelitian**

Tujuan dari penelitian ini adalah untuk mendapatkan nilai optimum dari  $R(D)$  secara komputasi dengan menggunakan bantuan program komputer.

### **1.4 Manfaat Penelitian**

Manfaat dari penelitian ini selain menambah pengetahuan tentang masalah teori informasi juga bermanfaat bagi peneliti lain untuk lebih mendalami sistem komunikasi khususnya masalah teori informasi karena fungsi nilai distorsi merupakan salah satu konsep dasar dari teori informasi.



## BAB 2. TINJAUAN PUSTAKA

Masalah fungsi nilai distorsi merupakan aplikasi dari pemrograman geometrik. Beberapa konsep yang diperlukan dalam masalah optimasi fungsi nilai distorsi dengan menggunakan pemrograman geometrik disajikan berikut ini.

### 2.1 Himpunan Konveks dan Fungsi Konveks

Sebuah titik  $X \in \mathfrak{R}^n$  dapat dituliskan  $X = (x_1, x_2, \dots, x_n)^T$  dengan  $x_i \in \mathfrak{R}$ ,  $i = 1, 2, \dots, n$ . Titik  $X \in \mathfrak{R}^n$  dikatakan kombinasi konveks dari  $X_1, X_2, \dots, X_m \in \mathfrak{R}^n$  apabila terdapat  $\lambda_j \geq 0$ ,  $j = 1, 2, \dots, m$ ,  $m \geq n$ , dan  $\sum_{j=1}^m \lambda_j = 1$  sehingga

$$X = \sum_{j=1}^m \lambda_j X_j = \lambda_1 X_1 + \lambda_2 X_2 + \dots + \lambda_m X_m \quad (2.1)$$

Jika diketahui titik  $X = (x_1, x_2, \dots, x_n)^T$  dan  $Y = (y_1, y_2, \dots, y_n)^T$  di  $\mathfrak{R}^n$  maka segmen garis antara kedua titik tersebut didefinisikan sebagai himpunan semua titik kombinasi konveks dari  $X$  dan  $Y$ , atau dengan kata lain himpunan semua titik  $Z \in \mathfrak{R}^n$  dengan koordinat

$$z_i = \lambda x_i + (1 - \lambda) y_i, \quad i = 1, 2, \dots, n \quad \text{dan} \quad 0 \leq \lambda \leq 1 \quad (2.2)$$

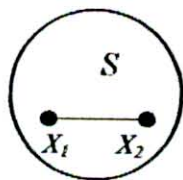
Jika sebarang 2 titik  $X_1$  dan  $X_2$  berada pada himpunan titik di  $\mathfrak{R}^n$  maka semua titik pada segmen garis antara kedua titik tersebut berada dalam himpunan titik di  $\mathfrak{R}^n$ , dengan demikian himpunan titik di  $\mathfrak{R}^n$  adalah konveks. Misalkan titik  $X_1, X_2 \in \mathfrak{R}^n$  dan  $S$  menunjukkan himpunan konveks, secara matematis dapat didefinisikan sebagai berikut :

Jika titik  $X_1, X_2 \in S$ , maka titik  $X \in S$  memenuhi

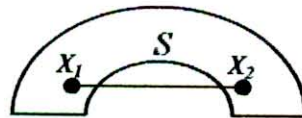
$$X = \alpha X_1 + (1 - \alpha) X_2, \quad 0 \leq \alpha \leq 1 \quad (2.3)$$



Berikut ini contoh himpunan konveks dan himpunan konkaf.



a. Himpunan konveks



b. Himpunan konkaf

Gambar 2.1 Himpunan konveks dan konkaf

### Teorema 2.1

Interseksi dari beberapa himpunan konveks juga himpunan konveks.

Suatu fungsi  $f(X)$  dikatakan konveks apabila untuk setiap dua titik yang berbeda  $X_1 = (x_{11}, x_{12}, \dots, x_{1n})^T$  dan  $X_2 = (x_{21}, x_{22}, \dots, x_{2n})^T$  dan untuk semua  $0 \leq \lambda \leq 1$  memenuhi

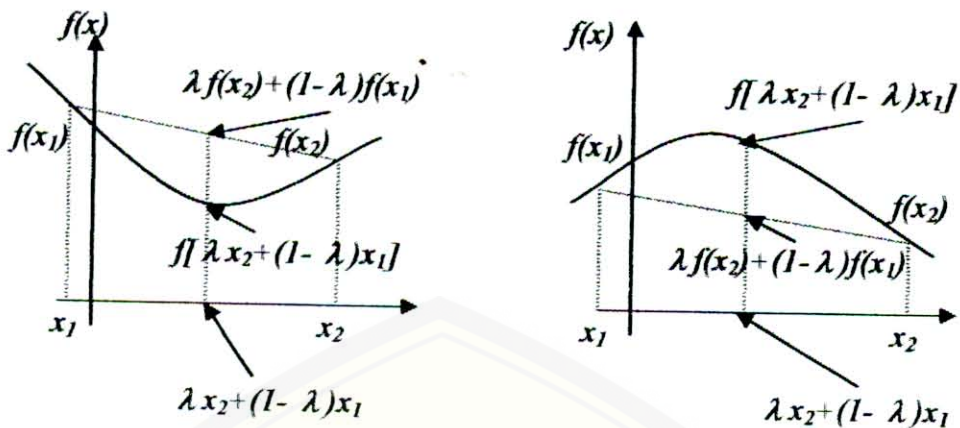
$$f(\lambda X_2 + (1-\lambda)X_1) \leq \lambda f(X_2) + (1-\lambda)f(X_1) \quad (2.4)$$

yaitu segmen garis antara 2 titik berada diatas atau pada grafik  $f(X)$ . Begitu juga  $f(X)$  dikatakan konkaf apabila untuk setiap dua titik yang berbeda  $X_1 = (x_{11}, x_{12}, \dots, x_{1n})^T$  dan  $X_2 = (x_{21}, x_{22}, \dots, x_{2n})^T$  dan untuk semua  $0 \leq \lambda \leq 1$  memenuhi

$$f(\lambda X_2 + (1-\lambda)X_1) \geq \lambda f(X_2) + (1-\lambda)f(X_1) \quad (2.5)$$

yaitu segmen garis antara 2 titik berada dibawah atau pada grafik  $f(X)$ .

Jika ditinjau dari grafiknya maka fungsi konveks melengkung ke atas dan fungsi konkaf melengkung ke bawah (Gambar 2.2). Negatif fungsi konveks adalah fungsi konkaf dan sebaliknya. Penjumlahan dari fungsi konveks adalah fungsi konveks dan penjumlahan dari fungsi konkaf adalah fungsi konkaf.



a. Fungsi konveks

b. Fungsi konkaf

Gambar 2.2 Fungsi konveks dan konkaf

Jika tanda " $\leq$ " pada persamaan (2.4) diganti tanda " $<$ " maka  $f(X)$  disebut fungsi konveks ketat (*strictly convex*) dan jika tanda " $\geq$ " pada persamaan (2.5) diganti tanda " $>$ " maka  $f(X)$  disebut fungsi konkaf ketat (*strictly concave*).

### Teorema 2.2

Suatu fungsi  $f(X)$  adalah konveks apabila untuk dua titik  $X_1$  dan  $X_2$  memenuhi

$$f(X_2) \geq f(X_1) + \nabla f^T(X_1)(X_2 - X_1) \quad (2.6)$$

dengan  $\nabla f^T = \frac{\partial f^T}{\partial X}$

Apabila  $f(X)$  fungsi konkaf maka tanda " $\geq$ " pada pertidaksamaan (2.6) menjadi " $\leq$ " (Boyd, S & Vandenberg, L. 2004).

### 2.2 Bentuk Umum Monomial dan Posinomial

Misal  $x_1, x_2, \dots, x_n$  merupakan bilangan real positif dan  $X = (x_1, x_2, \dots, x_n)$  adalah himpunan titik bilangan real positif dengan komponen-komponen  $x_i$ ,  $i = 1, 2, \dots, n$ . Sehingga himpunan titik bilangan real positif tersebut akan menghasilkan sebuah fungsi  $f(X)$  berbentuk:

$$f(X) = cx_1^{a_1} x_2^{a_2} \dots x_n^{a_n} \quad (2.7)$$

dengan  $c > 0$  dan  $a_i \in \mathfrak{R}$ ,  $i = 1, 2, \dots, n$  yang disebut dengan fungsi monomial, atau secara umum disebut monomial. Konstanta  $c$  tersebut menyatakan koefisien sedangkan konstanta  $a_1, a_2, \dots, a_n$  menyatakan konstanta eksponensial.

Posinomial merupakan jumlah dari satu atau lebih monomial. Sebuah fungsi posinomial dapat didefinisikan sebagai:

$$F(X) = \sum_{k=1}^K f_k(X) = f_1(X) + f_2(X) + \dots + f_K(X) \quad (2.8)$$

dengan  $f_k(X) = c_k \prod_{i=1}^n x_i^{a_{ik}} = c_k x_1^{a_{1k}} x_2^{a_{2k}} \dots x_n^{a_{nk}}$ ;  $c_k > 0$ ;  $a_{ik} \in \mathfrak{R}$ ;  $x_i > 0$ ;  $1 \leq i \leq n$ ;

dan  $1 \leq k \leq K$ . Fungsi  $F(X)$  mengambil sebuah bentuk polinomial dan karena semua  $c_k > 0$  serta variabelnya bernilai positif, maka  $F(X)$  diberi nama posinomial (Rakhmawati, 2006).

Sebagai contoh,  $2x_1^{-x} x_2^{0.5} + 3x_1 x_3^{100}$  adalah fungsi posinomial dan  $x_1 x_2^{-1}$  adalah fungsi monomial sekaligus fungsi posinomial.

### 2.3 Bentuk Umum Pemrograman Geometrik

Pemrograman geometrik merupakan kelas optimasi yang dikembangkan untuk menyelesaikan salah satu masalah pemrograman non linier. Ada dua bentuk persamaan pemrograman geometrik yaitu bentuk standar dan bentuk konveks. Pemrograman ini digunakan untuk mencari nilai optimum sebuah fungsi tujuan yang berbentuk posinomial. Jika ada kendala maka kendala tersebut dapat berbentuk posinomial ataupun monomial.

Bentuk standar pemrograman geometrik atau bisa juga disebut bentuk posinomial dapat diformulasikan sebagai berikut :

minimumkan  $F_0(X)$ ;

dengan kendala  $F_i(X) \leq 1, \quad i = 1, 2, \dots, m$ ;

$$f_l(X) = 1, \quad l = 1, 2, \dots, M; \quad (2.9)$$

dimana  $F_i(X) = \sum_{k=1}^{K_i} c_{ik} x_1^{a_{ik}} x_2^{a_{ik}} \dots x_n^{a_{ik}}$  adalah fungsi posinomial dengan  $i = 1, 2, \dots, m$ ;  $f_l(X) = c_l x_1^{a_l} x_2^{a_l} \dots x_n^{a_l}$  adalah fungsi monomial dengan  $l = 1, 2, \dots, M$  dan  $x_i$  dengan  $i = 1, 2, \dots, n$  adalah variabel optimasi. Kendala persamaan monomial dapat diekspresikan sebagai dua kendala pertidaksamaan monomial yaitu  $f_l(X) \geq 1$  dan  $f_l(X) \leq 1$ . Berikut adalah contoh pemrograman geometrik dalam bentuk standar.

minimumkan  $xy + xz$ ;

dengan kendala  $\frac{0,8\sqrt{yz}}{x^2} \leq 1$ ;

$$\frac{0,5}{\sqrt{xy}} \leq 1 \text{ dan } \frac{1}{x} \leq 1;$$

dengan variabel optimum  $x, y$ , dan  $z$ .

Pemrograman geometrik bentuk standar bukan masalah optimasi konveks karena posinomialnya bukan merupakan fungsi konveks. Namun demikian, dengan memisalkan  $y_i = \log(x_i)$ ,  $b_{ik} = \log(c_{ik})$  dan  $b_l = \log(c_l)$  dan mensubstitusikannya ke dalam persamaan (2.9) didapatkan

minimumkan  $\sum_{k=1}^{K_i} \exp(a_{ik}^T y + b_{ik})$ ;

dengan kendala  $\sum_{k=1}^{K_i} \exp(a_{ik}^T y + b_{ik}) \leq 1, \quad i = 0, 1, \dots, m$ ;

$$a_l^T y + b_l = 0, \quad l = 1, 2, \dots, M;$$

dengan variabel optimum  $y$  dan  $a_{ik} = [a_{ik}^{(1)}, a_{ik}^{(2)}, \dots, a_{ik}^{(n)}]$  (2.10)

Sehingga bentuk konveks pemrograman geometrik dari bentuk standar pemrograman geometrik masalah (2.9) adalah

$$\text{minimumkan } p_0(y) = \log \sum_{k=1}^{K_0} \exp(a_{0k}^T y + b_{0k})$$

$$\text{dengan kendala } p_i(y) = \log \sum_{k=1}^{K_i} \exp(a_{ik}^T y + b_{ik}) \leq 0, \quad i = 0, 1, \dots, m$$

$$q_l(y) = a_l^T y + b_l = 0, \quad l = 1, 2, \dots, M$$

$$\text{dengan variabel optimum } y \quad (2.11)$$

Berikut ini diberikan sebuah contoh bentuk konveks pemrograman geometrik yang dibangun dari contoh bentuk standar pemrograman geometrik dengan memisalkan  $\tilde{x} = \log(x)$ ,  $\tilde{y} = \log(y)$  dan  $\tilde{z} = \log(z)$  sehingga diperoleh (Chiang dan Boyd, 2004).

$$\text{minimumkan } \log(\exp(\tilde{x} + \tilde{y}) + \exp(\tilde{x} + \tilde{z}))$$

$$\text{dengan kendala } 0,5\tilde{y} + 0,5\tilde{z} - 2\tilde{x} + \log(0,8) \leq 0$$

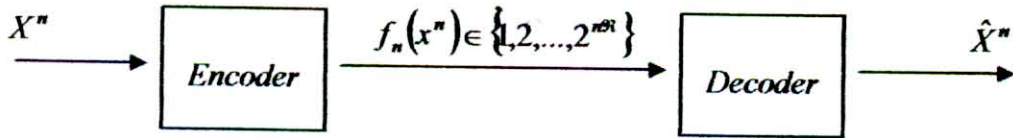
$$0,5\tilde{x} + \tilde{y} + \log(0,5) \leq 0 \text{ dan } -\tilde{x} \leq 0$$

$$\text{dengan variabel optimum } \tilde{x}, \tilde{y}, \tilde{z}.$$

## 2.4 Lagrange Dual Nilai Distorsi

### 2.4.1 Masalah Fungsi Nilai Distorsi

Diasumsikan sebuah sumber yang menghasilkan barisan variabel random  $X_1, X_2, \dots, X_n \sim p$ , dengan  $X_i$  adalah sebuah sumber alphabet  $\mathcal{X}$  yang diskrit dengan  $N$  alphabet simbol dan  $p \in \mathfrak{R}^{1 \times N}$  adalah distribusi sumber. Encoder menggambarkan barisan sumber  $X^n$  dengan  $f_n(x^n) \in \{1, 2, \dots, 2^{nR}\}$ , dan  $x^n \in X^n$ . Decoder membangun kembali  $X^n$  dengan estimasi  $\hat{X}^n = g_n(f_n(X^n))$  dalam membangun kembali alphabet  $\hat{X}$  yang berhingga, sebagai ilustrasi dapat dilihat Gambar 2.3 berikut ini.



Gambar 2.3 Encoder dan decoder

**Definisi 2.1**

Fungsi distorsi adalah pemetaan dari himpunan pasangan terurut *alphabet* sumber dan penghasil *alphabet* terhadap himpunan bilangan real non negatif

$$d : x \times \hat{x} \rightarrow \mathfrak{R}^+ \tag{2.12}$$

**Definisi 2.2**

Distorsi antara barisan  $x^n$  dan  $\hat{x}^n$  didefinisikan sebagai berikut

$$d(x^n, \hat{x}^n) = \frac{1}{n} \sum_{i=1}^n d(x_i, \hat{x}_i) \tag{2.13}$$

Fungsi nilai distorsi  $R(D)$  dari sumber diskrit dapat dievaluasi sebagai minimum informasi timbal balik  $I(X; \hat{X})$  diantara sumber dan membangun kembali dibawah kendala penyimpangan. Jadi

$$R(D) = \min_{P \in \mathcal{P}(X, \hat{X})} I(X; \hat{X}) \tag{2.14}$$

dengan  $P_{ij} = \text{Prob}\{\hat{X} = j | X = i\}$   $i = 1, 2, \dots, N$  dan  $j = 1, 2, \dots, M$  atau bisa diformulasikan dalam bentuk:

$$\text{minimumkan } \sum_{i=1}^N \sum_{j=1}^M p_i P_{ij} \log \frac{P_{ij}}{\sum_k P_{kj} p_k} \tag{2.15}$$

$$\text{dengan kendala } \sum_{i=1}^N \sum_{j=1}^M p_i P_{ij} d_{ij} \leq D$$

$$\sum_{j=1}^M P_{ij} = 1, \quad i = 0, 1, \dots, N$$

$$P_{ij} \geq 0, \quad i = 1, 2, \dots, N \text{ dan } j = 1, 2, \dots, M$$

dimana variabel-variabelnya adalah rekonstruksi probabilitas  $P_{ij}$ ; dengan parameter konstan adalah distribusi sumber  $p$ ; ukuran distorsi  $d_{ij} = d(X = i, \hat{X} = j)$  dan kendala distorsi  $D$ .

#### 2.4.2 Fungsi Nilai Distorsi Pemrograman Geometrik

##### Teorema 2.3

Lagrange dual dari masalah nilai distorsi atau persamaan (2.15) dapat ditulis kembali dalam bentuk pemrograman geometrik konveks sebagai berikut :

$$\text{maksimumkan } p\alpha^T - \gamma D \quad (2.16)$$

$$\text{dengan kendala } \log \sum_{i=1}^N e^{(\log p_i + \alpha_i - \gamma_j)} \leq 0, \quad j = 1, 2, \dots, M, \quad \gamma \geq 0$$

dimana  $\alpha, p \in \mathfrak{R}^{1 \times N}$ ,  $\gamma$  adalah variabel optimasi,  $d_{ij}$ , dan  $D$  adalah kendala distorsi.

Versi yang sama dari masalah lagrange dual dapat juga ditulis dalam bentuk pemrograman geometrik standar sebagai berikut :

$$\text{maksimumkan } w^{-D} \prod_{i=1}^N z_i^{p_i} \quad (2.17)$$

$$\text{dengan kendala } \sum_{i=1}^N p_i z_i w^{-d_{ij}} \leq 1, \quad j = 1, 2, \dots, M$$

$$w \geq 1, \quad z_i \geq 0, \quad i = 1, 2, \dots, N$$

dimana  $z$  dan  $w$  adalah variabel optimasi sedang  $p \in \mathfrak{R}^{1 \times N}$ ,  $d_{ij}$ , dan  $D$  adalah kendal distorsi.

Lagrange dual antara masalah pada persamaan (2.15) dan masalah pada persamaan (2.16) menghasilkan pernyataan berikut ini.

1. *Weak duality*. Penyelesaian layak  $(\alpha, \gamma)$  dari lagrange dual masalah pada persamaan (2.16) menghasilkan batas bawah dari fungsi nilai distorsi

$$p\alpha - \gamma D \leq R(D)$$

2. *Strong duality*. Nilai optimal dari lagrange dual masalah pada persamaan (2.16) adalah  $R(D)$  (Chiang dan Boyd, 2004).

## 2.5 Algoritma dan Pemrograman

Algoritma adalah urutan langkah instruksi logis yang disusun secara sistematis untuk menyelesaikan suatu masalah (Munir, 1999). Rancangan yang baik untuk suatu algoritma adalah menguraikan prosedur dalam beberapa sub prosedur, dari sub prosedur ini diuraikan lagi menjadi sub-sub prosedur dan seterusnya. Metode ini disebut rancangan algoritma terstruktur yang memberikan kemudahan pemahaman logika algoritma.

Pemrograman komputer adalah suatu instruksi untuk dilaksanakan komputer agar hasil tertentu dapat diperoleh. Jadi pemrograman komputer merupakan perwujudan dan implementasi dari algoritma yang ditulis dalam bahasa pemrograman tertentu sehingga dapat dilaksanakan komputer.

Adapun langkah-langkah untuk menyelesaikan permasalahan yang telah dijelaskan adalah :

1. membuat instalasi jalur ggplab ke dalam MATLAB;
2. membangun fungsi nilai distorsi pemrograman geometrik bentuk standar, dengan langkah-langkah sebagai berikut:
  1. menginput  $p$ ,  $D$  dan  $d$ ;
  2. mendefinisikan variabel optimum fungsi nilai distorsi pemrograman geometrik bentuk standar.
3. membangun fungsi objektif dan fungsi kendala.



3. mendapatkan nilai optimum dari fungsi nilai distorsi, dengan langkah-langkah sebagai berikut:
  1. membuat matriks konstanta eksponensial dan matriks logaritma koefisien dari fungsi objektif dan fungsi kendala fungsi nilai distorsi pemrograman geometrik bentuk standar;
  2. mendapatkan nilai dari variabel optimum fungsi nilai distorsi pemrograman geometrik bentuk konveks;
  3. mengubah nilai variabel optimum fungsi nilai distorsi pemrograman geometrik bentuk konveks menjadi variabel optimum fungsi nilai distorsi pemrograman geometrik bentuk standar;
  4. memasukkan nilai variabel optimum fungsi nilai distorsi pemrograman geometrik bentuk standar kedalam fungsi objektif fungsi nilai distorsi pemrograman geometrik bentuk standar;



### BAB 3. HASIL DAN PEMBAHASAN

#### 3.1 Hasil

Untuk mencari nilai optimum dari  $R(D)$  secara komputasi dengan terlebih dahulu mengubah  $R(D)$  ke dalam bentuk konveks pemrograman geometrik menggunakan bantuan MATLAB, diperlukan instalasi jalur ggplab kedalam program MATLAB. Ggplab adalah toolbox dalam MATLAB yang khusus digunakan untuk menyelesaikan masalah pemrograman geometrik. Skrip ggplab dapat dilihat di halaman lampiran. Adapun perintah untuk instalasi jalur ggplab adalah sebagai berikut.

```
>> addpath/home/username/ggplab
```

#### 3.1.1 Prosedur Membangun Fungsi Nilai Distorsi Pemrograman Geometrik Bentuk Standar

Dalam membangun fungsi objektif dan fungsi kendala diperlukan beberapa input sebagai berikut:

1. Variabel  $p$  yaitu variabel yang mewakili sebuah matriks berordo  $1 \times N$ , dimana elemen-elemennya adalah sebuah peluang dari kemunculan setiap huruf-huruf, simbol-simbol atau spasi dalam sebuah pesan yang ditulis. Variabel  $p$  dinyatakan dalam bentuk:

$$p = [p_1 \ p_2 \ p_3 \ \dots \ p_N] \text{ dengan } \sum_{i=1}^N p_i = 1$$

2. Variabel  $D$  yaitu variabel yang mewakili nilai kendala distorsi sebuah pesan yang diakibatkan oleh adanya gangguan di dalam pengiriman. Variabel ini berupa sebuah bilangan real non negatif atau  $D \in (\mathbb{R}^+, 0)$

3. Variabel  $d$  yaitu variabel yang mewakili matriks berordo  $N \times M$ , dengan elemen-elemennya adalah sebuah nilai distorsi dari setiap huruf-huruf, simbol-simbol atau spasi dalam sebuah pesan dengan  $d_{ij} \in (\mathbb{R}^+, 0)$   $i = 1, 2, \dots, N$  dan  $j = 1, 2, \dots, M$ . Variabel  $d$  dapat ditulis dalam bentuk:

$$d = \begin{bmatrix} d_{11} & d_{12} & \cdot & \cdot & \cdot & d_{1M} \\ d_{21} & d_{22} & \cdot & \cdot & \cdot & d_{2M} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ d_{N1} & d_{N2} & \cdot & \cdot & \cdot & d_{NM} \end{bmatrix}$$

Dalam membangun fungsi objektif dan fungsi kendala tidak hanya diperlukan input-input seperti diatas tetapi diperlukan juga beberapa variabel optimasi untuk membangun fungsi-fungsi tersebut. Oleh sebab itu setelah memasukkan input-input tersebut, terlebih dahulu harus didefinisikan variabel-variabel optimum fungsi nilai distorsi.

Variabel-variabel optimum fungsi nilai distorsi didefinisikan dengan menggunakan prosedur gpvar. Prosedur gpvar mendefinisikan variabel-variabel optimum fungsi nilai distorsi sebagai matriks tunggal ataupun matriks baris. Untuk masalah fungsi nilai distorsi pemrograman geometrik bentuk standar dengan parameter masukan untuk prosedur gpvar adalah  $w$  dan  $z$ . Dalam prosedur ini  $w$  adalah parameter optimum yang didefinisikan sebagai matriks tunggal dan  $z$  adalah parameter optimum yang didefinisikan sebagai matriks baris. Penulisan skrip MATLABnya adalah sebagai berikut.

```
>>gpvar w z(N)
```

Selanjutnya variabel-variabel optimum fungsi nilai distorsi digunakan untuk membangun fungsi objektif dan fungsi kendala pertidaksamaan. Dalam subbab 2.3 telah dijelaskan bahwa pemrograman geometrik bentuk standar

mempunyai fungsi objektif berbentuk posinomial dan jika ada kendala maka kendalanya bisa berbentuk posinomial ataupun monomial. Untuk fungsi nilai distorsi, fungsi objektifnya berbentuk monomial dengan kendala pertidaksamaan berbentuk posinomial.

Untuk membangun fungsi objektif dan fungsi kendala fungsi nilai distorsi pemrograman geometrik bentuk standar ke dalam MATLAB, digunakan perintah-perintah yang sudah ada di dalam MATLAB. Untuk penulisan skrip MATLABnya adalah sebagai berikut:

fungsi objektif :

```
>> objekif=(w^(-D))* (prod(z.^(p')))
```

fungsi kendala :

```
>> for i=1:N
    for j=1:M
        kendala=sum(p(i)*z(i)*(w.^(-d(i,j))))
    end
end
```

Sehingga penulisan skrip MATLAB untuk prosedur membangun fungsi objektif dan fungsi kendala fungsi nilai distorsi pemrograman geometrik bentuk standar dengan terlebih dahulu memasukkan input-inputnya adalah sebagai berikut:

```
>> [N,M]=size(d)
>> gpvar w z(N)
>> objektif= obj=(w^(-D))* (prod(z.^(p'))
>> for i=1:N
    for j=1:M
        kendala=sum(p(i)*z(i)*(w.^(-d(i,j))))
    end
end
```

Sehingga dihasilkan fungsi objektif dan fungsi kendala fungsi nilai distorsi pemrograman geometrik bentuk standar sebagai berikut:

1. Fungsi objektif :  $w^{-D} z_1^{p_1} z_2^{p_2} z_3^{p_3} \dots z_N^{p_N}$
2. Fungsi kendala :  $p_1 w^{-d_{11}} z_1 + p_1 w^{-d_{12}} z_1 + p_1 w^{-d_{13}} z_1 + \dots + p_1 w^{-d_{1M}} z_1 +$   
 $p_2 w^{-d_{21}} z_2 + p_2 w^{-d_{22}} z_2 + p_2 w^{-d_{23}} z_2 + \dots + p_2 w^{-d_{2M}} z_2 +$   
 $p_3 w^{-d_{31}} z_3 + p_3 w^{-d_{32}} z_3 + p_3 w^{-d_{33}} z_3 + \dots + p_3 w^{-d_{3M}} z_3 +$   
 $\dots +$   
 $p_N w^{-d_{N1}} z_n + p_N w^{-d_{N2}} z_N + p_N w^{-d_{N3}} z_N + \dots + p_N w^{-d_{NM}} z_N \leq 1$

dengan fungsi objektif adalah sebuah fungsi monomial dan fungsi kendala adalah fungsi posinomial.

### 3.1.2 Prosedur Menghitung Nilai Optimum Fungsi Nilai Distorsi

Setelah diperoleh fungsi objektif dan fungsi kendala fungsi nilai distorsi pemrograman geometrik bentuk standar, langkah yang harus dilakukan adalah memasukkan input yang digunakan untuk menghitung variabel-variabel optimum fungsi nilai distorsi pemrograman geometrik bentuk konveks. Variabel optimum harus dihitung menggunakan bentuk konveks karena dapat dipastikan bahwa minimum lokal yang dihasilkan adalah minimum global. Adapun input-input yang diperlukan adalah sebagai berikut.

1. Matriks yang elemen-elemennya berisi konstanta eksponensial dari fungsi objektif dan fungsi kendala fungsi nilai distorsi pemrograman geometrik bentuk standar.
  - a.  $A_0$  adalah matriks berordo  $1 \times (1 + N)$  yang semua elemennya berupa nilai konstanta eksponensial fungsi objektif fungsi nilai distorsi pemrograman geometrik bentuk standar. Atau dinyatakan dalam bentuk:

$$\mathbf{A0} = [-D \quad p_1 \quad p_2 \quad p_3 \quad \dots \quad p_N]$$

- b.  $\mathbf{A1}$  adalah matriks berordo  $(N \times M) \times (1 + N)$  yang semua elemennya berupa nilai konstanta eksponensial fungsi kendala fungsi nilai distorsi pemrograman geometrik bentuk standar. Atau dinyatakan dalam bentuk:

$$\mathbf{A1} = \begin{bmatrix} -d_{11} & 1 & 0 & \dots & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ -d_{1M} & 1 & 0 & \dots & 0 \\ -d_{21} & 0 & 1 & \dots & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ -d_{2M} & 0 & 1 & \dots & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ -d_{N1} & 0 & 0 & \dots & 1 \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ -d_{NM} & 0 & 0 & \dots & 1 \end{bmatrix}$$

2. Matriks yang elemen-elemennya berisi logaritma dari koefisien fungsi objektif dan fungsi kendala fungsi nilai distorsi pemrograman geometrik bentuk standar.
- a.  $\mathbf{B0}$  adalah matriks yang semua elemennya berupa nilai logaritma dari koefisien fungsi objektif fungsi nilai distorsi pemrograman geometrik bentuk standar dengan ordo  $1 \times 1$  yang dinyatakan dalam bentuk:

$$\mathbf{B0} = [\log(1)]$$

- b. **B1** adalah matriks beordo  $(N \times M) \times 1$  yang semua elemennya berupa nilai logaritma dari koefisien fungsi kendala fungsi nilai distorsi pemrograman geometrik bentuk yang dinyatakan dalam bentuk:

$$\mathbf{B1} = \begin{bmatrix} \log(p_1) \\ \cdot \\ \cdot \\ \cdot \\ \log(p_1) \\ \log(p_2) \\ \cdot \\ \cdot \\ \cdot \\ \log(p_2) \\ \cdot \\ \cdot \\ \cdot \\ \log(p_N) \\ \cdot \\ \cdot \\ \cdot \\ \log(p_N) \end{bmatrix}$$

Sebanyak  $M$

3. **A** merupakan matriks berordo  $(1 + (N \times M)) \times (1 + N)$  yang elemennya adalah semua elemen dari matriks **A0** dan semua elemen matriks **A1**, atau dalam skrip MATLAB ditulis sebagai berikut:

```
>> A=[A0;A1]
```

4. **B** merupakan matriks yang semua elemennya adalah semua elemen dari matriks **B0** dan matriks **B1** sehingga ordo matriks ini adalah  $(1 + (N \times M)) \times 1$ , atau dalam skrip MATLAB ditulis sebagai berikut:

```
>> B=[B0;B1]
```

5. *szs* merupakan matriks yang berelemen banyaknya baris dari matriks **A0** dan matriks **A1**, atau dalam skrip MATLAB ditulis sebagai berikut:

```
>> szs=[size(A0,1);size(A1,1)]
```

Langkah selanjutnya yang dilakukan adalah menghitung variabel-variabel optimum fungsi nilai distorsi dengan menggunakan perintah MATLAB sebagai berikut:

```
>> [x,status]=gpcvx(A,B,szs)
```

dengan memanggil perintah seperti diatas maka diperoleh sebuah output *x*, status dan beberapa informasi yang diperlukan untuk mendukung kebenaran dari nilai variabel optimasi yang didapat. Variabel *x* adalah variabel optimum fungsi nilai distorsi pemrograman geometrik bentuk konveks. Dengan variabel optimum pemrograman geometrik bentuk standar adalah eksponensial dari variabel optimum pemrograman geometrik bentuk konveks. Status menjelaskan bahwa perintah mencari nilai variabel-variabel optimum fungsi nilai distorsi pemrograman geometrik bentuk konveks dapat diselesaikan atau tidak.

Setelah diperoleh nilai dari variabel-variabel optimum fungsi nilai distorsi pemrograman geometrik bentuk konveks, langkah selanjutnya mendapatkan nilai fungsi nilai distorsi dengan cara memasukkan variabel-variabel optimum pemrograman geometrik bentuk standar ke dalam fungsi kendala fungsi nilai distorsi pemrograman geometrik bentuk standar. Untuk mengetahui fungsi nilai distorsi yang didapat adalah benar, langkah yang dilakukan adalah mengetahui nilai dari *duality gap*.

*Duality gap* adalah perbedaan antara nilai optimal primal objektif dan nilai optimal dual objektif. *Duality gap* selalu bernilai non negatif, dikarenakan sifat dari *weak duality* fungsi nilai distorsi ( $R(D) - (p\alpha - \gamma D) \geq 0$ ). Untuk mengetahui



fungsi nilai distorsi yang diperoleh benar atau mendekati kebenaran, nilai *duality gap* yang diperoleh harus mendekati atau sama dengan nol, pernyataan ini diperkuat dengan sifat dari *strong duality* fungsi nilai distorsi.

### 3.2 Pembahasan

Misal terdapat sebuah saluran diskrit dengan informasi-informasi sebagai berikut. Peluang dari setiap huruf-huruf, simbol-simbol dan spasi dalam sebuah pesan dinyatakan dalam matriks  $p = [0,2 \ 0,3 \ 0,1 \ 0,15 \ 0,25]$ , distorsi dari sebuah pesan adalah 10, dan distorsi dari setiap huruf-huruf, simbol-simbol dan

spasi dalam sebuah pesan dinyatakan dalam matrik berikut ini  $d = \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 0 & 1 \\ 1 & 0 \\ 1 & 0 \end{bmatrix}$ .

Akan dihitung fungsi nilai distorsi dari sebuah pesan.

Dari masalah diatas diketahui :

$$p = [0,2 \ 0,3 \ 0,1 \ 0,15 \ 0,25], D = 10 \text{ dan } d = \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 0 & 1 \\ 1 & 0 \\ 1 & 0 \end{bmatrix}$$

skrip program MATLAB untuk mencari fungsi objektif dan fungsi kendala fungsi nilai distorsi pemrograman geometrik bentuk standar dari masalah diatas sebagai berikut

```
>> p=[0.2 0.3 0.1 0.15 0.25];
>> D=10;
>> d=[1 1;1 1;0 1;1 0;1 0];
>> [N,M]=size(d);
>> gpvar w z (N);
```

```

>> obj=(w^(-D))* (prod(z.^(p')) )
>> for i=1:N
    for j=1:M
        kendala=sum(p(i)*z(i)*(w.^(-d(i,j))))
    end
end
end

```

Dari potongan program diatas diperoleh fungsi objektif dan fungsi kendala fungsi nilai distorsi pemrograman geometrik bentuk standar dari masalah diatas sebagai berikut

1. fungsi objektif :  $w^{-10} z_1^{0,2} z_2^{0,3} z_3^{0,1} z_4^{0,15} z_5^{0,25}$
2. fungsi kendala :  $0,2w^{-1}z_1 + 0,2w^{-1}z_1 + 0,3w^{-1}z_2 + 0,3w^{-1}z_2 + 0,1z_3 + 0,1w^{-1}z_3 + 0,15w^{-1}z_4 + 0,15z_4 + 0,25w^{-1}z_5 + 0,25z_5 \leq 1$

dengan fungsi objektif berupa fungsi monomial dan fungsi kendala berupa fungsi posinomial.

Berikut merupakan skrip program MATLAB untuk mencari nilai variabel-variabel optimum fungsi nilai distorsi pemrograman geometrik bentuk konveks.

```

>>A0=[-10 0.2 0.3 0.1 0.15 0.25];
>>A1=[-1 1 0 0 0 0; -1 1 0 0 0 0; -1 0 1 0 0 0; -1 0 1 0 0
    0; 0 0 0 1 0 0; -1 0 0 1 0 0; -1 0 0 0 1 0; 0 0 0 0 1
    0; -1 0 0 0 0 1; 0 0 0 0 0 1];
>>b0=[log(1)];
>>b1=[log(0.2);log(0.2);log(0.3);log(0.3);log(0.1);log(0.1);
    log(0.15);log(0.15);log(0.25);log(0.25)];
>>A=[A0;A1];
>>b=[b0;b1];
>>szs=[size(A0,1); size(A1,1)];
>>[x,status]=gpcvx(A,b,szs)

```

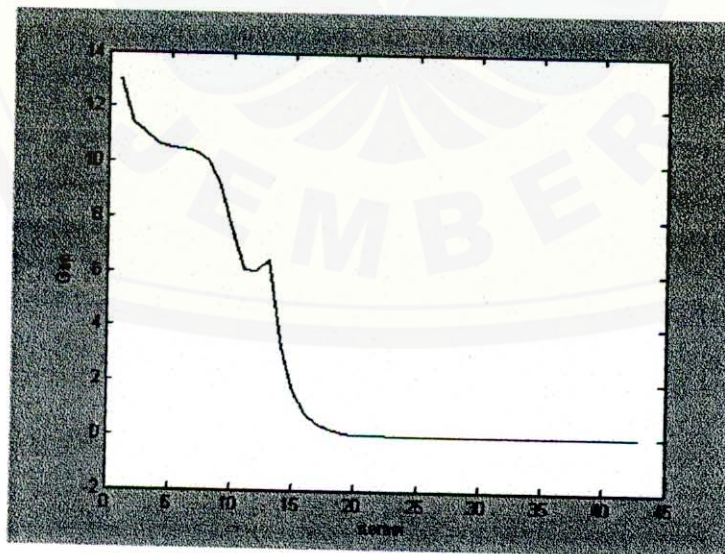
Dari program diatas diperoleh nilai variabel-variabel optimum fungsi nilai distorsi sebagai berikut

1.  $\gamma = 250$
2.  $\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = \alpha_5 = -250$

dengan status dapat diselesaikan, hasil dari program dapat dilihat di halaman lampiran sehingga dari persamaan  $w = \exp(\gamma)$  dan  $z = \exp(\alpha)$  diperoleh nilai variabel-variabel optimum nilai distorsi pemrograman geometrik bentuk standar sebagai berikut.

1.  $w = 3,7465e + 108$
2.  $z_1 = z_2 = z_3 = z_4 = z_5 = 2,6692e - 109$

dan hasil yang diperoleh juga sudah memenuhi syarat  $w \geq 1$  dan  $z_i \geq 0$ . Selanjutnya variabel-variabel optimum nilai distorsi pemrograman geometrik bentuk standar dimasukkan kedalam fungsi objektif sehingga diperoleh fungsi nilai distorsi sebesar 0. Hasil ini juga diperkuat dengan kondisi nilai pengiterasian *duality gap* yang mendekati nilai 0. Berikut adalah grafik hubungan antara iterasi dengan *duality gap*.



Gambar 3.1 Plot antara iterasi dan gap duality

Berikut juga disajikan beberapa contoh yang memberikan gambaran mengenai pengaruh input-input  $p$ ,  $D$  dan  $d$  terhadap besarnya nilai optimum fungsi nilai distorsi. Dan dari tabel tersebut diperoleh bahwa pengaruh terbesar dari input yang dapat mengubah nilai optimum fungsi nilai distorsi adalah input  $D$ . Juga dapat dilihat untuk nilai  $0 \leq D \leq 1,9805$ , input  $D$  memberikan nilai optimum fungsi nilai distorsi yang berbeda-beda dengan nilai optimum fungsi nilai distorsi yang menuju ke angka 0. sedangkan untuk nilai  $D > 1,9805$ , input  $D$  memberikan nilai optimum fungsi nilai distorsi sebesar 0.

1. Tabel dengan  $p = [0,2 \quad 0,3 \quad 0,1 \quad 0,15 \quad 0,25]$  dan  $d = \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 0 & 1 \\ 1 & 0 \\ 1 & 0 \end{bmatrix}$

Tabel 3.1 Hubungan antara berbagai nilai  $D$  terhadap fungsi nilai distorsi

No.	$D$	Fungsi nilai distorsi $[R(D)]$	No.	$D$	Fungsi nilai distorsi $[R(D)]$
1	0	2,6692e-109	6	1,9806	0
2	0,0001	2.6033e-109	7	1,981	0
3	1,0009	5,6891e-218	8	2	0
4	1,9	1,3693e-315	9	7	0
5	1,9805	4,9407e-324	10	100	0

2. Tabel dengan  $p = [0,4 \quad 0,25 \quad 0,35]$  dan  $d = \begin{bmatrix} 3 & 2 & 1 \\ 5 & 4 & 8 \\ 12 & 9 & 7 \end{bmatrix}$

Tabel 3.2 Hubungan antara berbagai nilai  $D$  terhadap fungsi nilai distorsi

No.	$D$	Fungsi nilai distorsi $[R(D)]$	No.	$D$	Fungsi nilai distorsi $[R(D)]$
1	0	2.6692e-109	6	1,9806	0
2	0,0001	2.6033e-109	7	1,981	0
3	1,0009	5.6891e-218	8	2	0
4	1,9	1,3693e-315	9	7	0
5	1,9805	4,9407e-324	10	100	0

3. Tabel dengan  $p = [0.1 \ 0.1 \ 0.1 \ 0.25 \ 0.1 \ 0.2 \ 0.15]$  dan  $d =$

$$\begin{bmatrix} 6 & 1 \\ 32 & 7 \\ 1 & 6 \\ 8 & 1 \\ 3 & 9 \\ 11 & 0 \\ 2 & 5 \end{bmatrix}$$

Tabel 3.3 Hubungan antara berbagai nilai  $D$  terhadap fungsi nilai distorsi

No	$D$	Fungsi nilai distorsi $[R(D)]$	No.	$D$	Fungsi nilai distorsi $[R(D)]$
1	0	2.6692e-109	6	1,9806	0
2	0,0001	2.6033e-109	7	1,981	0
3	1,0009	5.6891e-218	8	2	0
4	1,9	1,3693e-315	9	7	0
5	1,9805	4,9407e-324	10	100	0

**DAFTAR PUSTAKA**

- Almir, M. K., Kwangmoo, S., Kim, dan Boyd, S. 2006. *GGPLAB: A Simple Matlab Toolbox For Geometric Programming Version 1.00*.  
<http://www.stanford.edu/~boyd/ggplap>
- Boyd, S dan Vandenberg, L. 2004. *Convex Optimization*. USA: Cambridge University.
- Chiang, M. Tanpa Tahun. *Geometric Programming for Communication Systems*. USA: Princeton University.
- Chiang, M. dan Boyd, S. 2004. *Geometric Programming Duals of Channel Capacity and Rate Distortion*.  
<http://www.princeton.edu/~chiangm/gpdual.pdf>
- Cover, T. M dan Thomas, J. A. 1991. *Elements of Information Theory*. New York: John Willey & Sons, Inc.
- Munir, R. 1999. *Algoritma dan Pemrograman Komputer Bahasa Pascal dan C*. Bandung: CV. Informatika.
- Rakhmawati, A. 2006. *Optimasi Kode pada Gaussian Channel dengan Menggunakan Pemrograman Geometrik*. Skripsi. Jember: Universitas Jember. (Tidak Dipublikasikan).

**LAMPIRAN A****Bentuk M-File *ggplab*****M-File *gpvar***

```

% GPVARS Displays all the GP variables defined in the workspace.
%% GPVARS will list all the GP variables present in the workspace,
% together with their size, bytes size, and class.
%gpvars_names = []; gpvars_whos = whos;
for gpvars_k = 1:length(gpvars_whos)
    if strcmp(gpvars_whos(gpvars_k).class, 'gpvar')
        gpvars_names = [gpvars_names gpvars_whos(gpvars_k).name ' '];
    end
end
disp(' ');
disp('Available GP variables are:');
disp(' ');
if isempty(gpvars_names)
    disp(' None. ');
else
    eval(['whos ' gpvars_names])
end
disp(' ');
clear gpvars_names gpvars_whos gpvars_k;

```

**M-File *gp constraint***

```

function constr = gpconstraint(varargin)
% GPCONSTRAINT is a GP constraint class constructor.
%% Inequality operators between GP objects, such as positive
% scalars, GP variables, monomials, posynomials, and generalized
% posynomials (when valid) return GPCONSTRAINT objects.
% These GPCONSTRAINT objects can be assigned to variables, for example,
%     constr1 = x*y <= z;
%     constr2 = x^5 == 1;
% are two valid GP constraints.
%% A set of constraint is represented as an array of GP constraints, i.e.,
%     constr_set = [constr1 constr2];
%% You can also define GP constraints using:
%     constr = gpconstraint('constraint string')
%if nargin == 0 % create an empty constraint
    constr.lhs = [];
    constr.type = [];
    constr.rhs = [];
    constr.gpvars = {};
    constr = class(constr, 'gpconstraint');
return;

```

```

end

if nargin == 1 % one argument
    % invoke copy constructor if the new object is gpconstraint
    if isa(varargin(1), 'gpconstraint')
        constr = varargin(1);
        return;
    end
end

if nargin == 3
    % standard argument list
    constr.lhs      = varargin(1);
    constr.type     = varargin(2);
    constr.rhs      = monomial(varargin(3));
    if( isnumeric(constr.lhs) )
        constr.lhs  = monomial(constr.lhs);
        constr.gpvars = constr.rhs.gpvars;
    elseif( isa(constr.lhs, 'gpvar') )
        constr.lhs  = monomial(constr.lhs);
        constr.gpvars = union(constr.lhs.gpvars, constr.rhs.gpvars );
    else
        constr.gpvars = union(constr.lhs.gpvars, constr.rhs.gpvars);
    end
    constr = class(constr, 'gpconstraint');
    return;
end

M-File gpconstrain
function constr = gpconstraint(varargin)
% GPCONSTRAINT is a GP constraint class constructor.
%% Inequality operators between GP objects, such as positive
% scalars, GP variables, monomials, posynomials, and generalized
% posynomials (when valid) return GPCONSTRAINT objects.
% These GPCONSTRAINT objects can be assigned to variables, for example,
%   constr1 = x*y <= z;
%   constr2 = x^5 == 1;
% are two valid GP constraints.
%% A set of constraint is represented as an array of GP constraints, i.e.,
%   constr_set = [constr1 constr2];
%% You can also define GP constraints using:
%   constr = gpconstraint('constraint string')
%if nargin == 0 % create an empty constraint
    constr.lhs      = [];
    constr.type     = [];
    constr.rhs      = [];

```



```

    constr.gpvars = {};
    constr = class(constr, 'gpconstraint');
    return;
end
if nargin == 1 % one argument
    % invoke copy constructor if the new object is gpconstraint
    if isa(varargin{1}, 'gpconstraint')
        constr = varargin{1};
        return;
    end
end
if nargin == 3
    % standard argument list
    constr.lhs      = varargin{1};
    constr.type     = varargin{2};
    constr.rhs      = monomial(varargin{3});
    if( isnumeric(constr.lhs) )
        constr.lhs      = monomial(constr.lhs);
        constr.gpvars = constr.rhs.gpvars;
    elseif( isa(constr.lhs, 'gpvar') )
        constr.lhs      = monomial(constr.lhs);
        constr.gpvars = union(constr.lhs.gpvars, constr.rhs.gpvars );
    else
        constr.gpvars = union(constr.lhs.gpvars, constr.rhs.gpvars);
    end
    constr = class(constr, 'gpconstraint');
    return;
end
M-File gpsolve
function [obj_value, solution, status] = gpsolve(varargin)
% GPSOLVE Solves a geometric programming optimization problem.
%% GPSOLVE calls the internal primal-dual interior point solver
% in order to solve the specified GP problem.
% GPSOLVE calling sequence is:
%% [obj_value, solution, status] = gpsolve(obj, constr_array, flag)
%% where inputs are:
% obj          - objective function for the GP problem
% constr_array - array of problem constraints
% flag         - 'min' or 'max' (optional, default is 'min')
%% and outputs are:
% obj_value - the optimal objective value (a number)
% solution  - a cell array of GP variable names and their optimal values
% status    - the problem status flag
%% The status can be 'Solved' (if the optimization was successful),

```

```

% 'Infeasible' (if the problem was determined to be infeasible), and
% 'Failed' (if the optimization was not successful).
%% The inputs can also be empty arrays. If the objective is an empty
% array or a constant, then GPSOLVE solves a feasibility problem.
% If the constraint array is empty, then we have an unconstrained GP.
%% Internally GPSOLVE creates a GP problem object (gpproblem) and
% calls its solve method.
%if nargin == 2
    obj = varargin{1};
    constr = varargin{2};
    flag = 'min';
elseif nargin == 3
    obj = varargin{1};
    constr = varargin{2};
    flag = varargin{3};
else
    error('Wrong number of input arguments.')
end
gp_problem_obj = gpproblem(obj, constr, flag);
result_obj = solve(gp_problem_obj);
obj_value = result_obj.obj_value;
solution = result_obj.solution;
status = result_obj.status;

function [x,status,lambda,nu,mu] = gpcvx(A,b,szs,varargin)
% [x,status,lambda,nu,mu] = gpcvx(A,b,szs,G,h,l,u,quiet)
% solves the geometric program in convex form
% minimize lse(y0)
% subject to lse(yi) <= 0, i=1,...,m,
% Ai*x+bi = yi, i=0,...,m,
% G*x+h = 0,
% li <= xi <= ui, i=1,...,n
% where lse is defined as lse(y) = log sum_i exp yi,
% and the dual problem,
% maximize b0'*nu0 + ... + bm'*num + h'*mu + lambda1'*1 - lambdau'*u +
% entr(nu0) + lambda1*entr(nu1/lambda1) +
% ,..., + lambda m*entr(num/lambda m)
% subject to nu i >= 0, i=0,...,m,
% lambda i >= 0, i=1,...,m,
% 1'*nu0 = 1
% 1'*nu i = lambda i, i=1,...,m,
% A0'*nu0 + ... + Am'*num + G'*mu + lambda u - lambda d = 0,
% where entr is defined as entr(y) = -sum_i yi*log(yi).
%% Calling sequences:

```

```

%% [x,status,lambda,nu,mu] = gpcvx(A,b,szs)
% [x,status,lambda,nu,mu] = gpcvx(A,b,szs,G,h)
% [x,status,lambda,nu,mu] = gpcvx(A,b,szs,G,h,l,u)
% [x,status,lambda,nu,mu] = gpcvx(A,b,szs,G,h,l,u,quiet)
%% Examples:
%% [x,status,lambda,nu,mu] = gpcvx(A,b,szs,G,h)
% [x,status,lambda,nu,mu] = gpcvx(A,b,szs,[],[],[],[],quite)
%% Input arguments:
%% - A: (sum_i n_i) x n matrix; A = [A0' A1' ... Am' ]'
% - b: (sum_i n_i) vector; b = [b0' b1' ... bm' ]'
% - szs: dimensions of Ai and bi; szs = [n0 n1 ... nm]'
% where Ai is (ni x n) and bi is (ni x 1)
% - G: p x n matrix
% - h: p-vector
% - l: n-vector; lower bound for x
% - u: n-vector; upper bound for x
% - quiet: boolean variable; suppress all the print messages if true
% Output arguments:
% - x: n-vector; primal optimal point
% - nu: (sum_i n_i) vector; nu = [nu0' nu1' ... num' ]'
% dual variables for constraints Ai*x + bi = yi
% - mu: (sum_i l_i) vector; mu = [mu0' mu1' ... mul' ]'
% dual variables for constraints G*x + h = 0
% - lambda: m-vector, dual variables for inequality constraints
% - status: string; 'Infeasible', 'Solved', or 'Failed'
%
% gpcvx sets up phase 1 and phase 2 and calls the real solver, gppd2.m.
%-----
% INITIALIZATION
%-----
% PARAMETERS
LOWER_BOUND = -250;
UPPER_BOUND = +250;
% DIMENSIONS
N = size(A,1); % # of terms in the obj. and inequalities
n = size(A,2); % # of variables (x1,...,xn)
m = length(szs)-1; % # of inequalities
n0 = szs(1); % # of terms in the objective
% VARIABLE ARGUMENT HANDLING
defaults = {[],[],LOWER_BOUND*ones(n,1),UPPER_BOUND*ones(n,1),false};
givenArgs = ~cellfun('isempty',varargin);
defaults(givenArgs) = varargin(givenArgs);
[G,h,l,u,quiet] = deal(defaults{:});
% MATLAB LIMIT OF LOWER/UPPER BOUNDS

```

```

l(l<LOWER_BOUND) = LOWER_BOUND;
u(u>UPPER_BOUND) = UPPER_BOUND;
if (isempty(G))
    G = zeros(0,n);
    h = zeros(0,1);
end
p = size(G,1);      % # of (terms in) the equality constraints
% E is a matrix s.t. [1'*y0 1'*y1 ... 1'*ym]' = E*y
inds1 = cumsum(szs);
indsf = inds1-szs+1;
lx = zeros(N,1);
lx(indsf) = 1;
E = sparse(cumsum(lx), [1:N], ones(N,1));
%-----
%                               PHASE I
%-----
% solves the feasibility problem
%% minimize    s
% subject to   lse(yi) <= s,    i=1,...,m,
%              Ai*x+bi = yi,    i=0,...,m,
%              G*x+h = 0,
%              li <= xi <= ui,  i=1,...,n
%% where lse is defined as lse(y) = log sum_i exp yi,
%% For phase I
% 1) change objective function to s
% 2) change constraints from fi(x) <= 0 to fi(x) <= s
% 3) add bound constraints; li <= xi <= ui
%% Hence, we set up a new objective and constraints,
%    i.e., A,b,G,h and szs for Phase I optimization.
%% Change the size vector, szs.
%% Change A and b
%      s    xi
% A1 = [ 1 | 0 0 0      b1 = [ 0      <- (a1) new objective
%      +-----+-----
%      -1 |
%      -1 |A(ineq)      b      <- (a2) new inequalities
%      -1 |
%      +-----+-----
%      -1 |-1 0 0      11
%      -1 | 0-1 0      12      <- (a4) 1 <= xi
%      -1 | 0 0-1      13
%      +-----+-----
%      -1 | 1 0 0      -u1
%      -1 | 0 1 0      -u2      <- (a6) xi <= u

```

```

%      -1 | 0 0 1 ];      -u3 ];
% FORM SZS
szs1  = [1; szs(2:end); ones(2*n,1) ];
% FORM INITIAL X
if (p == 0)
    xinit = zeros(n,1);
else
    xinit = G'*((G*G')\h);
end
% FORM INITIAL S
% sinit = max{fi,0} since fi <= si and 0 <= si
y = A*xinit+b;
[f,expyy] = lse(E,y);
finit = f(2:m+1);
linit = -xinit+1;
uinit = +xinit-u;
sinit = max([0; finit; linit; uinit]) + 1; % + 1 is for margin.
% FORM A AND B
A1 = [+1           , sparse(1,n);...
      -ones(N-n0,1) , A(n0+1:N,:);...
      -ones(n,1)   , -speye(n)  ;...
      -ones(n,1)   , +speye(n)  ];
b1 = [ 0; b((n0+1):N); 1; -u ];
% FORM G AND H
G1 = [spalloc(size(G,1),1,0), G];
h1 = [h];
% CALL THE INTERNAL GP SOLVER
[x,status,lambda,nu,mu] = gppd2(A1,b1,szs1,[sinit;xinit],G1,h1,true,quiet);
% EXTRACT X FROM [S; X]
x0 = x(2:n+1);
y = A*x0+b;
[f,expyy] = lse(E,y);
flm      = f(2:m+1);
% FEASIBILITY CHECK OF PHASE I SOLUTION
if (status <= 0 || max([flm; -Inf]) >= 0)
    status = 'Infeasible';
    if (~quiet) disp(status); end
    return
end
clear A1 b1 G1 h1 x01 szs1;
%-----
%
%              PHASE II
%-----
% solves the geometric program in convex form

```

```

%% minimize    lse(y0)
% subject to  lse(yi) <= 0,    i=1,...,m, .
%             Ai*x+bi = yi,    i=0,...,m,
%             G*x+h = 0,
%             li <= xi <= ui, i=1,...,n
%% where lse is defined as lse(y) = log sum_i exp yi,
%% Change A and b to add the bound of x into the inequality constraints.
%
%   A2 = [ A          b2 = [ b
%         -----      ----
%         -1 0 0          11
%         0-1 0          12    <- li <= xi
%         0 0-1          13
%         -----      ----
%         1 0 0          -u1    <- xi <= ui
%         0 1 0          -u2
%         0 0 1 ];      -u3 ];
szs2 = [ szs ; ones(2*n,1) ];
A2    = [ sparse(A); -speye(n); speye(n) ];
b2    = [ b; l; -u ];
% CALL THE INTERNAL GP SOLVER
[x,status,lambda,nu,mu] = gppd2(A2,b2,szs2,x0,G,h,false,quiet);
if (status <= 0)
    status = 'Failed';
    if (~quiet) disp(status); end
    return
else
    status = 'Solved';
    if (~quiet) disp(status); end
    return
end

function [x,status,la,nu,mu] = gppd2(A,b,szs,x0,G,h,phase1,quiet)
% [x,nu,mu,la,status] = gppd2(A,b,szs,x0,G,h,phase1,quiet)
% solves the geometric program in convex form with a starting point.
% minimize    lse(y0)
% subject to  lse(yi) <= 0,    i=1,...,m,
%             Ai*x+bi = yi,    i=0,...,m,
%             G*x+h = 0,
% where lse is defined as lse(y) = log sum_i exp yi,
% and the dual problem,
% maximize    b0'*nu0 + ... + bm'*num + h'*mu +
%             entr(nu0) + lal*entr(nul/lal) +
%             ,..., + lam*entr(num/lam)

```

```

% subject to  nui >= 0,          i=0,...,m,
%             lai >= 0,          i=1,...,m,
%             1'*nu0 = 1
%             1'*nui = lai, i=1,...,m,
%             A0'*nu0 + ... + Am'*num + G'*mu = 0,
% where entr is defined as  entr(y) = -sum_i yi*log(yi).
% x0 should satisfy the primal inequality constraints.
% Input arguments:
% - A:         (sum_i ni) x n matrix; A = [A0' A1' ... Am' ]'
% - b:         (sum_i ni) vector;   b = [b0' b1' ... bm' ]'
% - szs:       dimensions of Ai and bi; szs = [n0 n1 ... nm]'
%             where Ai is (ni x n) and bi is (ni x 1)
% - x0:       n-vector; MUST BE STRICTLY FEASIBLE FOR INEQUALITIES
% - G:        p x n matrix
% - h:        p-vector
% - phasel:   boolean variable; indicator for phase I and phase II
%             true -> Phase I, false -> Phase II
% - quiet:    boolean variable; suppress all the print messages if true
% Output arguments:
% - x:        n-vector; primal optimal point
% - nu:       (sum_i ni) vector; nu = [nu0' nu1' ... num']'
%            dual variables for constraints Ai*x + bi = yi
% - mu:       p vector; mu = [mu1 ... mup]'
%            dual variables for constraints G*x + h = 0
% - la:       m vector; la = [lambda1 ... lambdam]'
%            dual variables lambda; lai = sum(nui)
% - status:   scalar;
%             2      Function converged to a solution x.
%             1      Phase I success; x(1:szs(1)) <= 0.
%             -1     Number of iterations exceeded MAXITERS.
%             -2     Starting point is not strictly feasible.
%             -3     Newton step calculation failure.
%-----
%
%             INITIALIZATION
%-----
% PARAMETERS
ALPHA  = 0.01;    % backtracking linesearch parameter (0,0.5]
BETA   = 0.5;    % backtracking linesearch parameter (0,1)
MU     = 2;      % IPM parameter: t update
MAXITER = 500;  % IPM parameter: max iteration of IPM
EPS    = 1e-8;  % IPM parameter: tolerance of surrogate duality gap
EPSfeas = 1e-8; % IPM parameter: tolerance of feasibility
% DIMENSIONS
[N,n] = size(A); m = length(szs)-1; p = size(G,1); n0 = szs(1);

```

```

if (isempty(G)), G = zeros(0,n); h = zeros(0,1); end
warning off all;
% SPARSE ZERO MATRIX
Opxp = sparse(p,p);
% SUM MATRIX: E is a matrix s.t. [1'*y0 1'*y1 ... 1'*ym ]' = E*y
inds1 = cumsum(szs);
indsf = inds1-szs+1;
lx = zeros(N,1);
lx(indsf) = 1;
E = sparse(cumsum(lx), [1:N], ones(N,1));
x = x0;
% flm is a LHS vector of inequality constraints s.t [f1' ... fm']'
y = A*x+b;
[f,expyy] = lse(E,y);
flm = f(2:m+1);
% CHECK THE INITIAL CONDITIONS
if (max(flm) >= 0)
    if (~quiet) disp(['x0 is not strictly feasible.']); end
    la = []; nu = []; mu = [];
    status = -2;
    return;
end;
% INITIAL DUAL POINT
la = -1./flm; % positive value with duality gap 1.
nu = ones(N,1); % ANY value.
mu = ones(p,1); % ANY value.
step = Inf;
if (~quiet) disp(sprintf('\n%s %15s %11s %20s %18s \n',...
    'Iteration', 'primal obj.', 'gap', 'dual residual', 'previous step.)); end
%-----
%                MAIN LOOP
%-----
for iters = 1:MAXITER
    gap = -flm'*la;
    % UPDATE T
    % update t only when the current x is not too far from the central path.
    if (step > 0.5)
        t = m*mu/gap;
    end
    % CALCULATE RESIDUALS
    % gradfy = exp(y)./(E'*(E*exp(y)));
    gradfy = expyy./(E'*(E*expyy));
    resDual = A'*(gradfy.*(E'*[1;la])) + G'*mu;
    resPrim = G*x + h;

```



```

resCent = [-la.*flm-1/t];
residual = [resDual; resCent; resPrim];
if (~quiet) disp(sprintf('%4d %20.5e %16.5e %14.2e %16.2e',...
    iters,f(1),-flm'*la,norm(resDual),step)); end
% STOPPING CRITERION FOR PHASE I
if ((phases == true) & max(x(1:szs(1))) < 0)
    nu = gradfy.*(E'*[1;la]);
    status = 1;
    return;
end;
% STOPPING CRITERION FOR PHASE I & II
if ( (gap <= EPS) & ...
    (norm(resDual) <= EPSfeas) & (norm(resPrim) <= EPSfeas) )
    % this calculation of nu is correct only when reached to optimal
    nu = gradfy.*(E'*[1;la]);
    status = 2;
    return;
end;
% CALCULATE NEWTON STEP
diagL1 = sparse(1:m+1,1:m+1,[0;-la./flm]);
diagL2 = sparse(1:m+1,1:m+1,[1;la]);
diagG1 = sparse(1:N,1:N,gradfy);
diagG2 = sparse(1:N,1:N,gradfy.*(E'*[1;la]));
EGA = E*diagG1*A;
H1 = EGA.*(diagL1-diaG2)*EGA + A'*diagG2*A;
dz = -[ H1, G' ; ...
    G , Opxp ]...
    \ ...
    [EGA'*[1;-1./(t*flm)]+G'*mu ; resPrim ];
% PERTURB KKT WHEN (ALMOST) SINGULAR
% add small diagonal terms when the matrix is almost singular.
perturb = EPS;
while (any(isinf(dz)))
    dz = -([ H1, G'; G , Opxp ] + sparse(1:n+p,1:n+p,perturb))...
        \ ...
        [EGA'*[1;-1./(t*flm)]+G'*mu ; resPrim ];
    % increase the size of diagonal term if still singular
    perturb = perturb*10;
end
% ERROR CHECK FOR NEWTON STEP CALCULATION
if (any(isnan(dz)))
    nu = gradfy.*(E'*[1;la]);
    status = -3;
    return;
end

```

```

end
dx = dz(1:n); dy = A*dx; dmu = dz(n+[1:p]');
dla = -la./flm.*(E(2:m+1,:)*(gradfy.*dy))+resCent./flm;
% BACKTRACKING LINESEARCH
negIdx = (dla < 0);
if (any(negIdx))
    step = min( 1, 0.99*min(-la(negIdx)./dla(negIdx)) );
else
    step = 1;
end
while (1)
    newx = x + step*dx;
    newy = y + step*dy;
    newla = la + step*dla;
    newmu = mu + step*dmu;
    [newf,newexpyy] = lse(E,newy);
    newflm = newf(2:end);
    % UPDATE RESIDUAL
    % newGradfy = exp(newy)./(E'*(E*exp(newy)));
    newGradfy = newexpyy./(E'*(E*newexpyy));
    newResDual = A'*(newGradfy.*(E'*[1;newla])) + G'*newmu;
    newResPrim = G*newx + h;
    newResCent = [-newla.*newflm-1/t];
    newResidual = [newResDual; newResCent; newResPrim];
    if ( max(newflm) < 0 && ...
        norm(newResidual) <= (1-ALPHA*step)*norm(residual) )
        break;
    end
    step = BETA*step;
end;
% UPDATE PRIMAL AND DUAL VARIABLES
x = newx; mu = newmu; la = newla; y = A*x+b;
[f,expyy] = lse(E,y); flm = f(2:m+1);
end
if (iters >= MAXITER)
    if (~quiet) disp(['Maxiters exceeded.']); end
    nu = gradfy.*(E'*[1;la]);
    status = -1;
    return;
end

```

**LAMPIRAN B****Bentuk M-File program pembahasan**

```

p=[0.2 0.3 0.1 0.15 0.25];
D=10;
d=[1 1;1 1;0 1;1 0;1 0];
[N,M]=size(d);
gpvar w z(N)
obj=(w^(-D))*(prod(z.^(p')));
for i=1:N
    for j=1:M
        kendala=sum(p(i)*z(i)*(w.^(-d(i,j))));
    end
end
clear
A0=[-10 0.2 0.3 0.1 0.15 0.25];
A1=[-1 1 0 0 0 0; -1 1 0 0 0 0; -1 0 1 0 0 0; -1 0 1 0 0 0; 0 0 0 1 0 0; -1 0 0 1
0 0; -1 0 0 0 1 0; 0 0 0 0 1 0; -1 0 0 0 0 1; 0 0 0 0 0 1];
b0=[log(1)];
b1=[log(0.2); log(0.2); log(0.3); log(0.3); log(0.1); log(0.1); log(0.15);
log(0.15); log(0.25); log(0.25)];
A=[A0;A1];
b=[b0;b1];
szs=[size(A0,1);size(A1,1)];
[x,status]=gpcvx(A,b,szs)

```

Iteration	primal obj.	gap	dual residual	previous step.
1	1.69315e+000	1.30000e+001	8.89e-001	Inf
2	-8.61716e+001	1.01824e+001	8.61e-001	3.13e-002

Iteration	primal obj.	gap	dual residual	previous step.
1	-8.85843e+001	1.30000e+001	1.00e+001	Inf
2	-1.08543e+003	1.14014e+001	1.00e+001	1.54e-002
3	-1.94906e+003	1.10760e+001	1.00e+001	3.75e-004
4	-2.48903e+003	1.06746e+001	1.00e+001	7.38e-004
5	-2.60458e+003	1.04958e+001	9.98e+000	1.50e-003
6	-2.61486e+003	1.04721e+001	9.96e+000	1.68e-003
7	-2.63117e+003	1.03959e+001	9.93e+000	3.81e-003
8	-2.66243e+003	1.00452e+001	9.83e+000	1.02e-002
9	-2.69845e+003	9.26378e+000	9.63e+000	2.04e-002
10	-2.73275e+003	7.55066e+000	9.16e+000	4.88e-002
11	-2.74618e+003	6.04775e+000	8.10e+000	1.16e-001
12	-2.74697e+003	6.01036e+000	6.07e+000	2.50e-001
13	-2.74710e+003	6.42168e+000	7.36e-005	1.00e+000
14	-2.74852e+003	3.21109e+000	1.83e-004	1.00e+000
15	-2.74926e+003	1.60555e+000	9.19e-005	1.00e+000
16	-2.74963e+003	8.02760e-001	2.26e-005	1.00e+000
17	-2.74981e+003	4.01378e-001	5.35e-006	1.00e+000
18	-2.74991e+003	2.00689e-001	1.29e-006	1.00e+000
19	-2.74995e+003	1.00344e-001	3.15e-007	1.00e+000
20	-2.74998e+003	5.01722e-002	7.79e-008	1.00e+000
21	-2.74999e+003	2.50861e-002	1.94e-008	1.00e+000
22	-2.74999e+003	1.25431e-002	4.83e-009	1.00e+000
23	-2.75000e+003	6.27153e-003	1.21e-009	1.00e+000
24	-2.75000e+003	3.13576e-003	3.01e-010	1.00e+000
25	-2.75000e+003	1.56788e-003	7.52e-011	1.00e+000
26	-2.75000e+003	7.83941e-004	1.88e-011	1.00e+000
27	-2.75000e+003	3.91970e-004	4.70e-012	1.00e+000
28	-2.75000e+003	1.95985e-004	1.18e-012	1.00e+000
29	-2.75000e+003	9.79926e-005	2.94e-013	1.00e+000
30	-2.75000e+003	4.89963e-005	7.34e-014	1.00e+000
31	-2.75000e+003	2.44981e-005	1.84e-014	1.00e+000