ΤΜΗΜΑ ΜΑΘΗΜΑΤΙΚΩΝ
Μ.Δ.Ε. ΜΑΘΗΜΑΤΙΚΑ ΚΑΙ ΣΥΓΧΡΟΝΕΣ ΕΦΑΡΜΟΓΕΣ
ΜΑΘΗΜΑ: ΑΡΙΘΜΗΤΙΚΗ ΒΕΛΤΙΣΤΟΠΟΙΗΣΗ

Αριθμητική Βελτιστοποίηση χωρίς περιορισμούς με την SciPy

ΚΩΝΣΤΑΝΤΙΝΟΣ ΘΑΝΟΣ

ΜΙΧΑΗΛ ΛΙΑΡΜΑΚΟΠΟΥΛΟΣ

Επιβλέπουσα:

Αν. Καθ. Θεοδούλα Γράψα

26 Φεβρουαρίου 2017

Πάτρα

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή στην Βελτιστοποίηση

1.1 Είδη βελτιστοποίησης

Τόσο στα Μαθηματικά, όσο και στην επιστήμη των Υπολογιστών, ένα πρόβλημα βελτιστοποίησης είναι ένα πρόβλημα που έχει ως στόχο την εύρεση της «βέλτιστης» λύσης απ' όλες τις δυνατές λύσεις. Για παράδειγμα τέτοια προβλήματα είναι το μέγιστο κέρδος σε μια επιχείρηση, η ελαχιστοποίηση των απωλειών και του ρίσκου.

Στην συγκεκριμένη εργασία θα ασχοληθούμε με προβλήματα ελαχιστοποίησης, γι' αυτό το λόγο θα χρησιμοποιήσουμε και τις ανάλογες μεθόδους των οποίων θα συγκρίνουμε τα αποτελέσματα με τη γλώσσα $Python$ χρησιμοποιώντας τη βιβλιοθήκη SciPy.

Οι μέθοδοι που επιλύουν προβλήματα ελαχιστοποίησης διακρίνονται σε δυο ειδών μεθόδους:

- Μέθοδοι ελαχιστοποίησης με περιορισμούς (**constrained minimization**)
- Μέθοδοι ελαχιστοποίησης χωρίς περιορισμούς (**unconstrained minimization**)

Εμείς θα ασχοληθούμε με τις δεύτερες μεθόδους.

Ένα βασικό ερώτημα που καλούμαστε να απαντήσουμε είναι τι αποτελεί λύση ενός προβλήματος ελαχιστοποίησης από πλευράς Μαθηματικών, το οποίο το εξετάζουμε στην συνέχεια.

Λύση ενός προβλήματος ελαχιστοποίησης είναι η εύρεση ενός ολικού ελαχιστοποιητή **(global minimizer)**, έστω x^* , τέτοιο ώστε να ισχύει :

$$f(x^*) \leq f(x), \text{ για κάθε x στο πεδίο ορισμού της f(x)} \tag{1.1}$$

Πιο αυστηρά το πρόβλημά μας μπορεί να γραφτεί ως :

$$\min_{\mathbf{x}\in A} f(\mathbf{x}), \quad f : \mathbb{R}^n \to \mathbb{R} \tag{1.2}$$

με **x** το διάνυσμα των n-ανεξάρτητων μεταβλητών της μορφής $\mathbf{x} = [x_1, x_2, ..., x_n]^T$. Το σύνολο A είναι υποσύνολο του $\mathbb{R}^n$ και καλείται σύνολο περιορισμών (**constraint set**).

Υπάρχουν προβλήματα βελτιστοποίησης που απαιτούν τη μεγιστοποίηση της αντικειμενικής συνάρτησης. Αυτά τα προβλήματα μπορούν να περιγραφούν με τις παρακάτω ισοδύναμες εκφράσεις :

$$maximizing\ f \ \ \text{ή} \ \ minimizing\ (-f) \tag{1.3}$$

Το παραπάνω πρόβλημα αποτελεί τη γενική μορφή των προβλημάτων βελτιστοποίησης με περιορισμους (**constrained optimization problems**), ενώ αν θεωρήσουμε ότι $A = \mathbb{R}^n$ τότε κάνουμε αναφορά για προβλήματα βελτιστοποίησης χωρίς περιορισμούς (**unconstrained optimization problems**).

Τέλος οι επαναληπτικές μέθοδοι που χρησιμοποιούνται για προβλήματα βελτιστοποίησης διακρίνονται σε δύο κατηγορίες:

- Μέθοδοι αναζήτησης ευθείας (Line Search Methods)
- Μέθοδοι έμπιστης περιοχής (Trust Region Methods)

Εμείς θα ασχοληθούμε κυρίως με την πρώτη κατηγορία εξετάζοντας πάντα και διάφορες προϋποθέσεις. Οι μέθοδοι που υπάρχουν στην παραπάνω κατηγορία διακρίνονται σε **ακριβείς** και **μη-ακριβείς**

1.2 Συνθήκες για τοπικά ελάχιστα

Ορισμός 1.2.1. Έστω μια πραγματική συνάρτηση $f : \mathbb{R}^n \to \mathbb{R}$, ορισμένη σε ένα σύνολο $A \subset \mathbb{R}^n$. Ένα σημείο $x^* \in A$ καλείται τοπικό ελάχιστο (τοπικός ελαχιστοποιητής) της συνάρτησης στο A εάν υπάρχει $\epsilon > 0$ τέτοιο ώστε :

$$f(x^*) \leq f(x),\ \forall x \in A - \{x^*\} \ \ \text{και} \ \ ||x - x^*|| < \epsilon$$

Το σημείο $x^* \in A$ είναι ολικό ελάχιστο (ολικός ελαχιστοποιητής) της f στο A όταν $f(x^*) \leq f(x),\ \forall x \in A - \{x^*\}$

Θα εξετάσουμε κάποιες συνθήκες προκειμένου ένα σημείο (διάνυσμα) $\mathbf{x}^*$ να είναι τοπικό ελάχιστο. Για μια συνάρτηση $f : \mathbb{R}^n \to \mathbb{R}$ με ανεξάρτητη μεταβλητή ένα n-διάστατο διάνυσμα **x**, γνωρίζουμε ότι η πρώτης τάξης παράγωγοι της δίνονται από την κλίση (gradient) της f δηλαδή :

$$\nabla f(x) = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \cdot \\ \cdot \\ \cdot \\ \frac{\partial f}{\partial x_n} \end{pmatrix} \tag{1.4}$$

ενώ η δεύτερης τάξης παράγωγος αντιστοιχεί στον Εσσιανό πίνακα :

$$H(x) = \nabla^2 f(x) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_1 \partial x_2} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ . & . & \cdots & . \\ \frac{\partial^2 f}{\partial x_1 \partial x_n} & \frac{\partial^2 f}{\partial x_2 \partial x_n} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix} \tag{1.5}$$

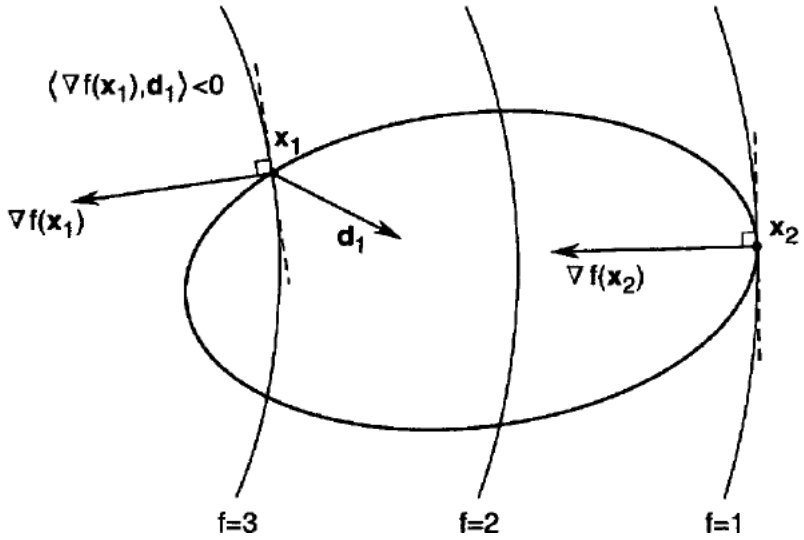
Στις μεθόδους Line Search θα μας απασχολήσουν τα ακόλουθα μεγέθη : Μια **κατεύθυνση** $d^{(k)}$, ένα **βήμα** $a^{(k)}$ και η **αρχική τιμή** $x^{(0)}$. Συνδυάζοντάς τα δημιουργείται ο επαναληπτικός τύπος αυτής της μεθόδου αναζήτησης:

$$x^{(k+1)} = x^{(k)} + a^{(k)} d^{(k)} \tag{1.6}$$

Η επιτυχία της συγκεκριμένης μεθόδου βασίζεται στην κατάλληλη επιλογή του βήματος αλλά και της κατεύθυνσης και οπότε στην συνέχεια διατυπώνουμε μερικά σημαντικά θεωρήματα.

Θεώρημα 1.2.1. ***Αναγκαία Συνθήκη Πρώτης Τάξης (FONC).*** *Έστω A ένα υποσύνολο του $\mathbb{R}^n$ και $f \in C^1$ μια συνάρτηση πραγματικών μεταβλητών στο A. Εάν x^* είναι ένα τοπικό ελάχιστο της f στο A, τότε για κάθε κατεύθυνση $d^{(k)}$ στο x^*, έχουμε :*

$$\langle \boldsymbol{d}, \nabla f(x^*) \rangle \geq 0 \tag{1.7}$$



Σχήμα 1.1: Γεωμετρική Ερμηνεία της αναγκαίας συνθήκης πρώτης τάξεως FONC. [1].

Στο παραπάνω σχήμα η x_1 δεν ικανοποιεί την ανισότητα 1.7 (συνθήκη FONC) ενώ η x_2 την ικανοποιεί. Αυτό συμβαίνει γιατί στο σημείο x_1 το εσωτερικό γινόμενο $\langle \nabla f(x_1), \mathbf{d}_1 \rangle$ ισούται με $||\nabla f(x_1)|| \cdot ||\mathbf{d}_1|| cos(\theta_1)$. Αλλά είναι εμφανές στο σχήμα πως το συνημίτονο

της γωνίας μεταξύ του $\nabla f(x_1)$ και του διανύσματος κατεύθυνσης $\mathbf{d}_1$ είναι αρνητικό. Και αυτό γιατί η γωνία είναι μεγαλύτερη από 90.

Στην περίπτωση του x_2 η γωνία μεταξύ του $\nabla f(x_2)$ και του διανύσματος κατεύθυνσης $\mathbf{d}_2$ είναι 90 και οπότε το συνημίτονό της μηδενίζεται. Άρα η συνθήκη FONC (1.7) ικανοποιείται.

Θεώρημα 1.2.2. ***Αναγκαία Συνθήκη Δεύτερης Τάξης (SONC).*** *Έστω A υποσύνολο του $\mathbb{R}^n$, μια συνάρτηση $f \in C^2$ στο A, x^* ένα τοπικό ελάχιστο της f στο A και d η κατεύθυνση στο x^*. Εάν $d^T \nabla f(x^*) = 0$, τότε :*

$$d^T H(x^*) d \geq 0 \tag{1.8}$$

Ακολουθεί ένα παράδειγμα όπου βλέπουμε πώς μπορούν να αξιοποιηθούν οι παραπάνω συνθήκες FONC, SONC.

Παράδειγμα 1.2.1. Έστω η συνάρτηση $f : \mathbb{R}^2 \to \mathbb{R}$ με $f(x_1, x_2) = x_1^2 - x_2^2$. Θα εξετάσουμε ποιες από τις δύο προηγούμενες συνθήκες $(FONC, SONC)$ ικανοποιούνται.

Λύση

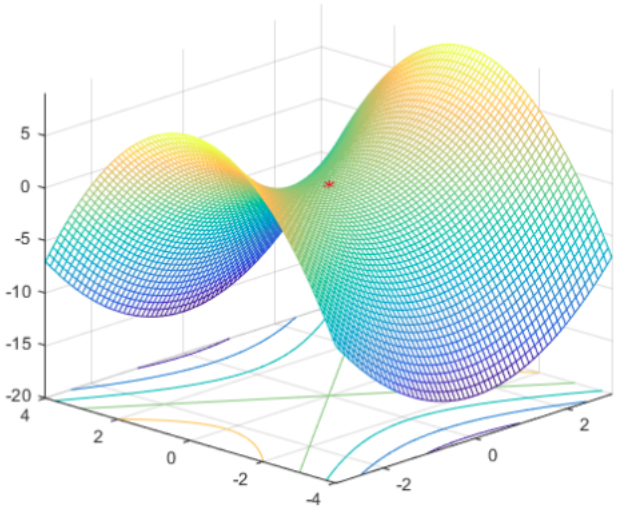
Βρίσκουμε αρχικά ότι $\nabla f(x_1, x_2) = \begin{pmatrix} 2x_1 \\ -2x_2 \end{pmatrix}$ Η συνθήκη FONC απαιτεί να ισχύει $\nabla f(x) = 0$. Έτσι στη συγκεκριμένη περίπτωση θα πρέπει $x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ Σχετικά με τη δεύτερη συνθήκη SONC αρχικά υπολογίζουμε τον εσσιανό πίνακα, Έτσι βρίσκουμε :

$$H(x) = \begin{pmatrix} 2 & 0 \\ 0 & -2 \end{pmatrix}$$

Παρατηρούμε ότι ο Εσσιανός είναι αόριστος επομένως για κάποια $d_1, d_2 \in \mathbb{R}^2$ θα ισχύουν:

$$d_1^T H(x) d_1 > 0 \quad \text{και} \quad d_2^T H(x) d_2 < 0$$

Για παράδειγμα έστω τα $d_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ και $d_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ Επομένως το σημείο $x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ ικανοποιεί την συνθήκη FONC αλλά όχι τη συνθήκη SONC, άρα δεν αποτελεί ελάχιστο για τη συνάρτηση f. Μια γεωμετρική εικόνα παίρνουμε και από το ακόλουθο γράφημα :



Σχήμα 1.2: Το γράφημα της συνάρτησης $f(x_1, x_2) = x_1^2 - x_2^2$. Το σημείο $(0, 0)^T$ ικανοποιεί τη FONC αλλά όχι τη SONC.

Λόγω του παραπάνω προβλήματος διατυπώνουμε την επαρκή συνθήκη δεύτερης τάξης όπως δίνεται από το επόμενο θεώρημα.

Θεώρημα 1.2.3. ***Επαρκής Συνθήκη δεύτερης τάξης (SOSC).*** *Έστω μια συνάρτηση $f \in C^2$ ορισμένη σε μια περιοχή όπου το x^* είναι εσωτερικό σημείο.*

Έστω ακόμα ότι :

1. $\nabla f(x^*) = 0$
2. $H(x^*) = 0$

*Τότε το x^* είναι **αυστηρά** τοπικό ελάχιστο της f*

1.3 Μέθοδοι μονοδιάστατης βελτιστοποίησης

Μονοδιάστατη μέθοδος Newton

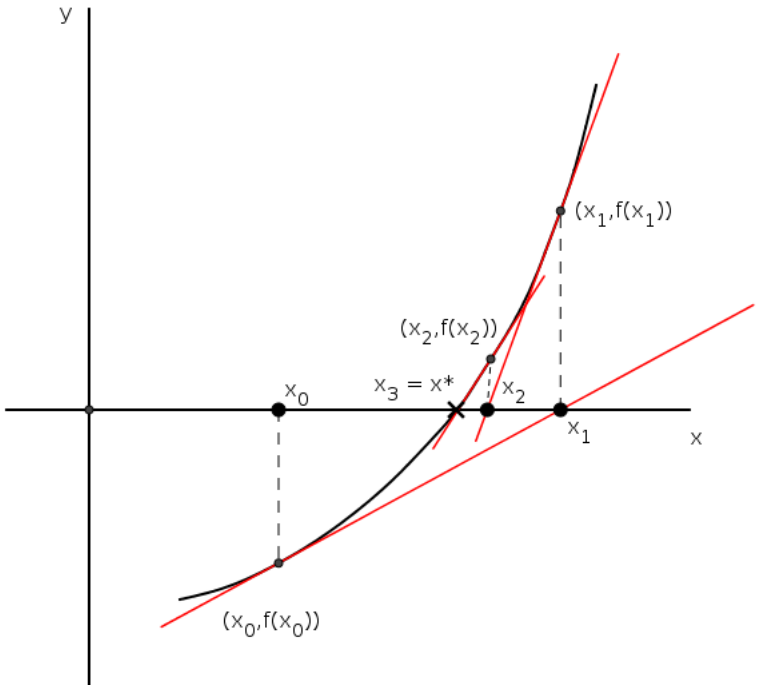
Η μέθοδος Newton [1] ή καλύτερα η μέθοδος των Newton-Raphson είναι η πιο ”δημοφιλής” μέθοδος για τη λύση μη γραμμικών εξισώσεων, διότι είναι απλή στην εφαρμογή της και επί πλέον, για απλές ρίζες είναι τετραγωνικής σύγκλισης (άρα αρκετά γρήγορη σε σχέση με άλλες μεθόδους).

Ξεκινάει από ένα τυχαίο σημείο του πεδίου ορισμού που καλό είναι να βρίσκεται πλησίον της ρίζας και δίνεται από τον επαναληπτικό τύπο :

$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})} \tag{1.9}$$

[1] https://en.wikipedia.org/wiki/Newton's_method

όπου k αντιστοιχεί στον αριθμό της εκάστοτε επανάληψης.



Σχήμα 1.3: Γεωμετρική Ερμηνεία της μονοδιάστατης μεθόδου Newton

Τετραγωνική σύγκλιση της μεθόδου

Έστω ότι η συνάρτηση f είναι δύο φορές συνεχώς παραγωγίσιμη. Από τη μέθοδο Newton ισχύει :

$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})}$$

Αν x^* είναι η ρίζα της f τότε από τη μέθοδο Newton πρυκύπτει :

$$g(x) = x - \frac{f(x)}{f'(x)} \text{ και } g(x^*) = x^*, \ g'(x) = \frac{f(x)f'(x)}{[f'(x)]^2}$$

Για να προσεγγίσουμε τη συνάρτηση g θα χρησιμοποιήσουμε ένα πεπερασμένο πλήθος όρων της σειράς Taylor[2].

Αναπτύσουμε τη $g(x)$ κατά Taylor κοντά στη ρίζα x^*. Άρα :

$$g(x) = g(x^*) + (x - x^*)g'(x^*) + (x - x^*)^2 \frac{g''(\xi)}{2!}, \ \xi \in (x, x^*)$$

Είδαμε ότι $g(x^*) = x^*$ και επιπλέον παρατηρούμε ότι : $g'(x^*) = 0$ ενώ $g''(x^*) \neq 0$.

[2] https://en.wikipedia.org/wiki/Taylor_series

Συνεπώς :

$$\begin{aligned} g(x) &= x^* + \frac{1}{2}(x - x^*)^2 g''(\xi) \\ g(x) - x^* &= \frac{1}{2}(x - x^*)^2 g''(\xi) \\ x^{(k+1)} - x^* &= \frac{1}{2}(x^{(k)} - x^*)^2 g''(\xi) \\ |\varepsilon^{(k+1)}| &\leq c \cdot |\varepsilon^k|^2 \end{aligned}$$

Άρα η σύγκλιση είναι τετραγωνική αφού το σφάλμα στη νέα επανάληψη είναι ανάλογο του τετραγώνου του σφάλματος της προηγούμενης.

Μέθοδος της τέμνουσας (Secant Method)

Η μέθοδος Newton με χρήση παραγώγων 2ης τάξης παίρνει τη μορφή :

$$x^{(k+1)} = x^{(k)} - \frac{f'(x^{(k)})}{f''(x^{(k)})}$$

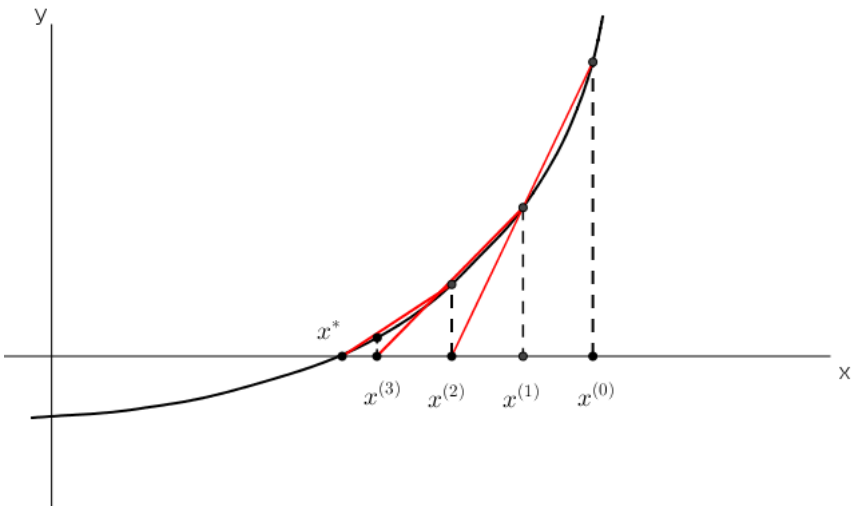
Σε περίπτωση όπου δεν είναι δυνατός ο υπολογισμός της 2ης παραγώγου, τότε δοκιμάζουμε να την προσεγγίσουμε μέσω της παραγώγου πρώτης τάξης. Δηλαδή προσεγγίζουμε την $f''(x^{(k)})$ μέσω του τύπου :

$$\frac{f'(x^{(k)}) - f'(x^{(k-1)})}{x^{(k)} - x^{(k-1)}}$$

Έτσι μέσω της παραπάνω προσέγγισης λαμβάνουμε τον επαναληπτικό τύπο της μεθόδου της τέμνουσας :

$$\begin{aligned} x^{(k+1)} &= x^{(k)} - \frac{x^{(k)} - x^{(k-1)}}{f'(x^{(k)}) - f'(x^{(k-1)})} f'(x^{(k)}) \\ x^{(k+1)} &= \frac{f'(x^{(k)})x^{(k-1)} - f'(x^{(k-1)})x^{(k)}}{f'(x^k) - f'(x^{k-1})} \end{aligned}$$

Παρατηρούμε ότι η μέθοδος της τέμνουσας απαιτεί δύο αρχικά σημεία για να ξεκινήσει, έστω τα $x^{(-1)}$ και $x^{(0)}$. Όπως και στη μέθοδο Newton, έτσι και στη μέθοδο της τέμνουσας δεν συμπεριλαμβάνεται τιμές της $f(x^{(k)})$, αντίθετα προσπαθούμε να μηδενίσουμε την παράγωγο.



Σχήμα 1.4: Γεωμετρική Ερμηνεία της μεθόδου της τέμνουσας

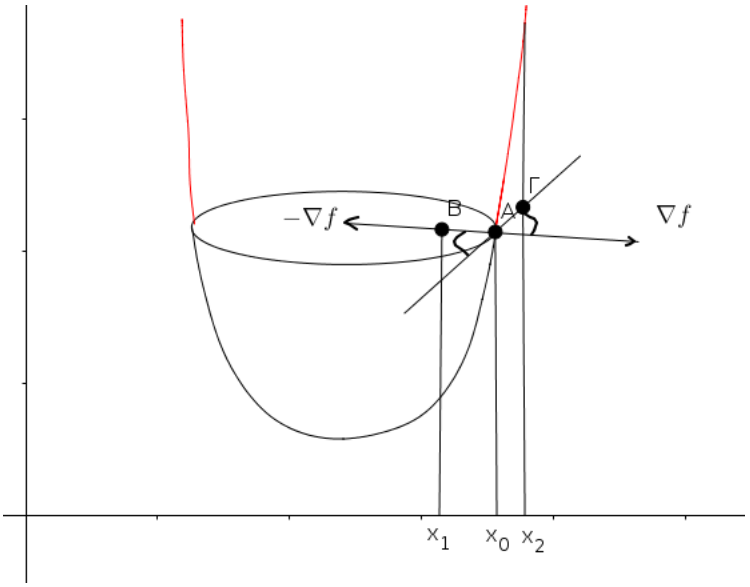
Κεφάλαιο 2

Μέθοδοι Πολυδιάστατης Βελτιστοποίησης Χωρίς Περιορισμούς

2.1 Μέθοδοι Κλίσης

2.1.1 Εισαγωγικά για τις μεθόδους κλίσης

Το gradient της f στο x_0, $\nabla f(x_0)$ εάν δεν είναι 0 τότε είναι κάθετο στο διάνυσμα της εφαπτομένης στο σημείο αυτό. Έτσι το gradient κινείται στην κατεύθυνση εκείνη τέτοια ώστε η συνάρτηση να μεγιστοποιεί την άυξηση ή τη μείωση στην κατεύθυνση του gradient παρά σε οποιαδήποτε άλλη κατεύθυνση.



Σχήμα 2.1: Γράφημα κατεύθυνσης των $\nabla f(x)$ και $-\nabla f(x)$

Για τα επόμενα θα χρειαστούμε το ακόλουθο σημαντικό θεώρημα από την γραμμική άλγεβρα, οπότε το διατυπώνουμε ακολούθως:

Θεώρημα 2.1.1 (Ανισότητα Cauchy-Schwarz)**.** *Για κάθε διάνυσμα* $\boldsymbol{x,y} \in \mathbb{R}^n$ *ισχύει η ανι-*

σότητα:

$$|\langle \boldsymbol{x},\boldsymbol{y}\rangle| \leq ||\boldsymbol{x}||||\boldsymbol{y}|| \tag{2.1}$$

Επίσης, η ισότητα ισχύει αν και μόνο αν το $\boldsymbol{x}$ *είναι γραμμικός συνδυασμός του* $\boldsymbol{y}$*, δηλαδή αν* $\boldsymbol{x} = \alpha \boldsymbol{y}$*, για κάποιο* $\alpha \in \mathbb{R}$*.*

Από την ανισότητα **Cauchy-Schwarz** (2.1) παίρνουμε:

$$\langle \nabla f(x), \mathbf{d}\rangle = ||\nabla f(x)|| \cdot ||\mathbf{d}|| \leq ||\nabla f(x)||, \text{ για } ||\mathbf{d}|| = 1$$

Δηλαδή για κάθε κατεύθυνση **d** με μοναδιαίο μέτρο, ο ρυθμός της μέγιστης αύξησης της f στο x, $\langle \nabla f(x), \mathbf{d}\rangle$ έχει μέγιστη τιμή $||\nabla f(x)||$.

Οπότε αν θέσουμε ως κατεύθυνση $\mathbf{d} = \nabla f(x)/||\nabla f(x)||$ παίρνουμε:

$$\langle \nabla f(x), \frac{\nabla f(x)}{||\nabla f(x)||}\rangle = ||\nabla f(x)|| \cdot ||\mathbf{d}|| = ||\nabla f(x)||$$

Επομένως από τα παραπάνω καταλήγουμε ότι η κατεύθυνση στην οποία δείχνει η κλίση της $f(x)$, $\nabla f(x)$, είναι η **κατεύθυνση μέγιστης αύξησης** της f στο x. Άρα η κατεύθυνση του $-\nabla f(x)$ είναι η **κατεύθυνση μέγιστης μείωσης** της f και η οποία είναι η πλέον κατάλληλη για αναζήτηση των global minimizers.

Στη συνέχεια αν θεωρήσουμε ένα αρχικό σημείο x_0 τότε ο προηγούμενος τύπος των Line search μεθόδων:

$$x^{(k+1)} = x^{(k)} + a^{(k)} d^{(k)}$$

για την πρώτη επανάληψη και για $d^{(k)} = -\nabla f(x_k)$ θα γίνει :

$$x^{(1)} = x^{(0)} - a^{(0)} \nabla f(x^{(0)})$$

Έτσι μέσω του αλγορίθμου αυτού καταλήγουμε στον τύπο : $x^{(k+1)} = x^{(k)} - a^{(k)} \nabla f(x^{(k)})$ Γενικότερα αναφερόμαστε στον παραπάνω τύπο ως ο **αλγόριθμος gradient descent**.

Παρατήρηση. Αν συνεχίσουμε την παραπάνω διαδικασία της πρώτης επανάληψης $x_1 = x_0 - a_0 \nabla f(x_0)$, τότε από το Θεώρημα Taylor [1] παίρνουμε :

$$f(x^{(0)} - a^{(0)} \nabla f(x^{(0)})) = f(x^{(0)}) - a^{(0)} ||\nabla f(x^{(0)})|| + o(a)$$

έτσι για $\nabla f(x_0) \neq 0$ και σχετικά μικρό βήμα α, παίρνουμε :

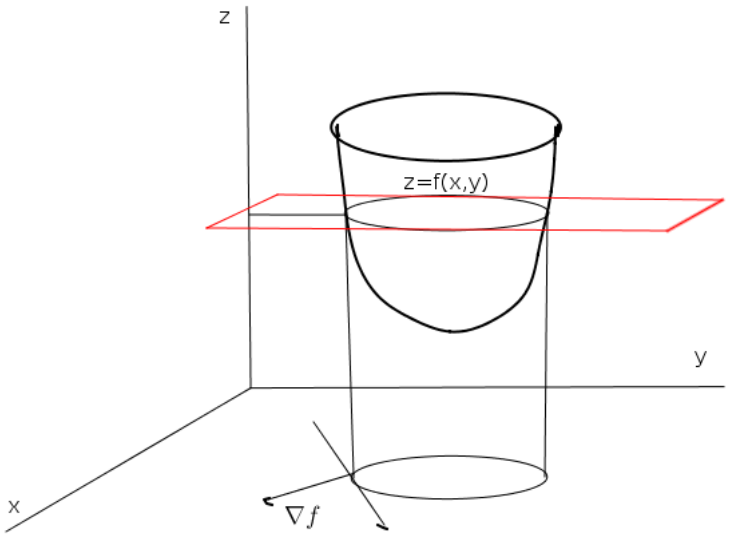
$$f(x^{(0)} - a^{(0)} \nabla f(x^{(0)})) < f(x^{(0)})$$

Επομένως το $x^{(0)} - a^{(0)} \nabla f(x^{(0)})$ αποτελεί βελτίωση του x_0 αν ψάχνουμε για ολικό ελαχιστοποιητή.

[1] https://en.wikipedia.org/wiki/Taylor%27s_theorem

2.1.2 Ισοϋψείς Καμπύλες

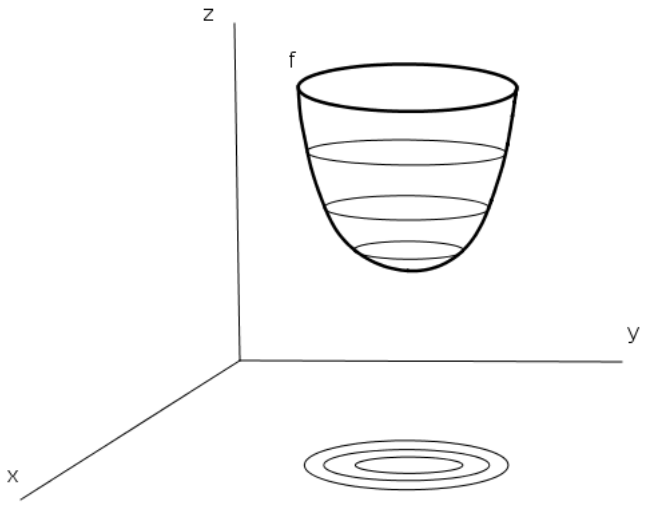
Πρώτα απ' όλα να σημειωθεί ότι στο χώρο η συναρτησιακή τιμή είναι ένα επίπεδο που "κόβει" την καμπύλη και είναι παράλληλο με το $z = 0$. Ακουλουθεί και ενδεικτικό σχήμα:



Σχήμα 2.2: Απεικόνιση της συναρτησιακής τιμής στο χώρο

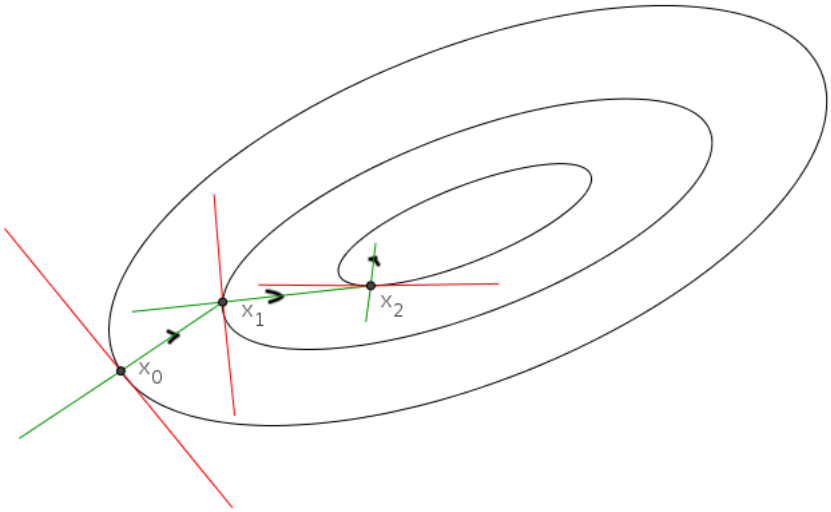
Ορισμός 2.1.1. Ισοϋψής καμπύλη καλείται ο γεωμετρικός τόπος των σημείων που έχουν την ίδια συναρτησιακή τιμή. Δηλαδή σε μαθηματική μορφή μια ισοϋψής καμπύλη είναι το σύνολο $C_1 = \{\forall x \forall y | f(x, y) = c\}$.

Στο παρακάτω σχήμα βλέπουμε ουσιαστικά τρεις συναρτησιακές τιμές οι οποίες απεικονίζονται στο χώρο αλλά και στο επίπεδο. Αν παρατηρήσουμε τις καμπύλες στο επίπεδο, τότε καθώς κινούμαστε προς το εσωτερικό δηλαδη τη μικρότερη (ως προς το εμβαδό) καμπύλη, κατευθυνόμαστε σε μικρότερη συναρτησιακή τιμή. Άρα ακολουθούμε την κατεύθυνση του $-\nabla f(x, y)$.



Σχήμα 2.3: Απεικόνιση στο χώρο και το επίπεδο xy

Εάν μεγενθύνουμε κατάλληλα το προηγούμενο σχήμα και συγκεριμένα στο επίπεδο που απεικονίζονται οι τρεις καμπύλες τότε θα είχαμε μια εικόνα σαν την παρακάτω



Στο παραπάνω σχήμα παρατηρούμε τα ακόλουθα άξια αναφοράς:

- Οι κόκκινες γραμμές αντιστοιχούν στις εφαπτομένες της εκάστοτε καμπύλης στα σημεία x_0, x_1 και x_2 καθώς κινούμαστε σε μικρότερη καμπύλη αντίστοιχα.
- Οι πράσινες γραμμές αντιστοιχούν στις κατευθύνσεις των ∇f (από μικρότερες προς μεγαλύτερες καμπύλες) και $-\nabla f$ (από μεγαλύτερες προς μικρότερες καμπύλες), γι' αυτό και έχουν διεύθυνση κάθετη στις εφαπτομένες.

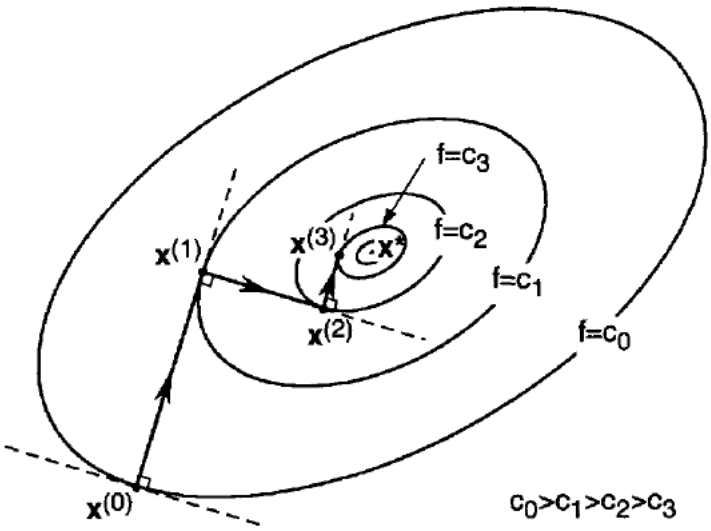
Συμπέρασμα : Βλέπουμε ότι καθώς κινούμαστε σε μικρότερη καμπύλη, δηλαδή από το x_0 στο x_1 και από αυτό στο x_2 μειώνεται αντίστοιχα και η συναρτησιακή τιμή, όπως φαίνεται καλύτερα και στο παραπάνω σχήμα.

2.1.3 Μέθοδος Απότομης Καθόδου (Steepest Descent)

Η μέθοδος Steepest Descent αποτελεί έναν αλγόριθμος κλίσης όπου το βήμα a_k επιλέγεται κατάλληλα ώστε να έχουμε τη μέγιστη δυνατή μείωση της συναρτησιακής τιμής. Δηλαδή το βήμα επιλέγεται τ.ω. να ελαχιστοποιείται η συνάρτηση :

$$\phi(a) = f(x^{(k)} - a\nabla f(x^{(k)}))$$

Τελικά σε αυτή τη μέθοδο σε κάθε βήμα ξεκινώντας από ένα αρχικό σημείο (διάνυσμα για μεγαλύτερες διαστάσεις) $\mathbf{x}^{(0)}$ γίνεται ένα βήμα line search στην κατεύθυνση του $-\nabla f(x^{(k)})$ μέχρι την εύρεση του ζητούμενου ελαχιστοποιητή $x^{(k+1)}$. Ακολουθεί και ενδεικτικό σχήμα :



Θεώρημα 2.1.2. *Εάν $\{x^{(k)}\}_{k=0}^{\infty}$ είναι μια ακολουθία απότομης καθόδου για μια συνάρτηση $f : \mathbb{R}^n \to \mathbb{R}$, τότε για κάθε k το διάνυσμα $x^{(k+1)} - x^{(k)}$ θα είναι ορθογώνιο ως προς το διάνυσμα $x^{(k+2)} - x^{(k+1)}$.*

Απόδειξη:

Από τον επαναληπτικό τύπο για τη μέθοδο steepest descent προκύπτει άμεσα ότι :

$$\langle x^{(k+1)} - x^{(k)}, x^{(k+2)} - x^{(k+1)} \rangle = a^{(k)} a^{(k+1)} \langle \nabla f(x^{(k)}), \nabla f(x^{(k+1)}) \rangle$$

Επομένως αρκεί να δείξουμε ότι $\langle \nabla f(x^{(k)}), \nabla f(x^{(k+1)}) \rangle = 0$

Το βήμα $a^{(k)}$ είναι τέτοιο ώστε να ελαχιστοποιεί τη συνάρτηση

$$\phi_k(a) = f(x^{(k)} - a^k \nabla f(x^{(k)}))$$

Έτσι με χρήση της συνθήκης **FONC** θα πάρουμε :

$$\begin{aligned} 0 &= \phi_k'(a^{(k)}) \\ &= \frac{d\phi_k}{da}(a^{(k)}) \\ &= \nabla f(x^{(k)} - a^{(k)}\nabla f(x^{(k)}))^T(-\nabla f(x^{(k)})) \\ &= -\langle \nabla f(x^{(k+1)}), \nabla f(x^{(k)}) \rangle \end{aligned} \tag{2.2}$$

Από το παραπάνω θεώρημα προκύπτει το ακόλουθο πόρισμα.

Πόρισμα 2.1.1. *Εάν* $\{x^{(k)}\}_{k=0}^{\infty}$ *είναι μια ακολουθία steepest descent για μια συνάρτηση* $f : \mathbb{R}^n \to \mathbb{R}$ *και* $\nabla f(x^{(k)}) \neq 0$ *τότε* $f(x^{(k+1)}) < f(x^{(k)})$.

Τα κριτήρια τερματισμού μιας ακολουθίας απότομης καθόδου είναι:

Κριτήρια τερματισμού

- $\nabla f(x^{(k+1)}) = 0$
- $|f(x^{(k+1)}) - f(x^{(k)})| < \varepsilon$, όπου $\varepsilon > 0$
- $||x^{(k+1)} - x^{(k)}|| < \varepsilon$
- $\frac{|f(x^{(k+1)}) - f(x^{(k)})|}{|f(x^{(k)})|} < \varepsilon$
- $\frac{||x^{(k+1)} - x^{(k)}||}{||x^{(k)}||} < \varepsilon$

Συμπεριφορά της steepest descent για τετραγωνικές συναρτήσεις

Έστω η συνάρτηση της μορφής $f(x) = \frac{1}{2}x^TQx - b^Tx$, όπου $Q \in \mathbb{R}^{n\times n}$ είναι ένας θετικά ορισμένος πίνακας, $b \in \mathbb{R}^n$ και $x \in \mathbb{R}^n$. Το ολικό ελάχιστο της f μπορεί να βρεθεί θέτοντας το gradient της f ίσο με το μηδέν. Δηλαδή :

$$\nabla f(x) = Qx - b = 0$$

Ο εσσιανός πίνακας της f είναι ο $H(x) = Q = Q^T > 0$. Επομένως ο αλγόριθμος steepest descent για μια τετραγωνική συνάρτηση της παραπάνω μορφής θα δίνεται από το ακόλουθο επαναληπτικό σχήμα.

$$x^{(k+1)} = x^{(k)} - a^k\nabla f(x^{(k)}) \tag{2.3}$$

Έστω τώρα ότι $\nabla f(x^{(k)}) \neq 0$. Εφαρμόζοντας τη συνθήκη **FONC** στη συνάρτηση $\phi_k(a) = f(x^{(k)} - a^k\nabla f(x^{(k)}))$ θα πάρουμε

$$\phi_k'(a) = (x^{(k)} - a^k\nabla f(x^{(k)}))^T(-\nabla f(x^{(k)})) - b^T(-\nabla f(x^{(k)}))$$

Επομένως $\phi_k'(a) = 0$ αν $(x^{(k)} - a^k\nabla f(x^{(k)}))^T(-\nabla f(x^{(k)})) = b^T(-\nabla f(x^{(k)}))$ Τελικά καταλήγουμε για το βήμα ότι δίνεται από τον τύπο :

$$a^{(k)} = \frac{\nabla f(x^{(k)})^T\nabla f(x^{(k)})}{\nabla f(x^{(k)})^TQ\nabla f(x^{(k)})}$$

2.2 Μέθοδος πολυδιάστατης Newton

2.3 Μέθοδοι Quasi Newton

Γύρω στο 1950, ένας φυσικός ονόματι W.C. Davidon, εργαζότανε πάνω σε μεγάλα προβλήματα υπολογισμών βελτιστοποίησης στο Argonne National Laboratory χρησιμοποιώντας τη μέθοδο $coordinate-descent$. Ο αλγόριθμος τον οποίο ανέπτυξε ήταν ο πρώτος $Quasi-Newton$ αλγόριθμος. Γρήγορα αποδείχθηκε από τους Fletcher και Powell, ότι ο νέος αυτός αλγόριθμος ήταν πιο γρήγορος και πιο αξιόπιστος από οποιοδήποτε άλλη μέθοδο που υπήρχε εκείνη τη στιγμή.

Οι μέθοδοι Quasi-Newton όπως και οι μέθοδοι steepest descent, απαιτούν μόνο τον υπολογισμό του gradient της αντικειμενικής συνάρτησης σε κάθε επανάληψη. Ελέγχοντας τις αλλαγές του gradient σε κάθε επανάληψη κατασκευάζουν ένα μοντέλο της συνάρτησης που είναι αρκετά καλό για "υπεργραμμική" σύγκλιση. Η βελτίωση σε σχέση με τις μεθόδους Steepest descent είναι δραματική, ειδικά σε δύσκολα προβλήματα. Τέλος δεν απαιτείται ο υπολογισμός παραγώγων δεύτερης τάξης και για αυτό το λόγο αρκετές φορές οι Quasi Newton μέθοδοι είναι πιο αποτελεσματικοί από τη μέθοδο Newton[7].

2.3.1 BFGS & DFP

Η πιο γνωστή Quasi-Newton μέθδος είναι η **BFGS** η οποία πήρε το όνομα της από τους δημιουργούς της, Broyden, Fletcher, Goldfarb, και Shanno. Έστω ότι το μοντέλο της αντικειμενικής συνάρτησης στην τρέχουσα επανάληψη $x^{(k)}$ είναι :

$$m^{(k)}(d) = f_k + \nabla f_k^T d + \frac{1}{2} d^T B_k d \tag{2.4}$$

και ο επαναληπτικός τύπος δίνεται αρχικά ως εξής :

$$x^{(k+1)} = x^{(k)} + a^{(k)} d^{(k)} \tag{2.5}$$

όπου $a^{(k)}$ είναι το κατάλληλο βήμα που ικανοποιεί τις συνθήκες Wolfe[2] και συγκεκριμένα η συνθήκη $f(x^{(k+1)}) < f(x^{(k)})$.

Επιπλέον η κατεύθυνση $d^{(k)}$ είναι κατεύθυνση μείωσης και δίνεται από τον τύπο :

$$d^{(k)} = -B_k^{-1} \nabla f_k \tag{2.6}$$

και B_k είναι ένας θετικά ορισμένος πίνακας που αποτελεί προσέγγιση του Εσσιανού πίνακα H_k.

Αντί του υπολογισμού του B_k σε κάθε επανάληψη, ο Davidon πρότεινε την ανανέωση του πίνακα λαμβάνοντας υπόψιν την καμπυλότητα όπως αυτή έχει μετρηθεί στο πιο

[2] https://en.wikipedia.org/wiki/Wolfe_conditions

πρόσφατο βήμα. Έστω ότι έχουμε κατασκευάσει τη νέα επανάληψη $x^{(k+1)}$ και θέλουμε να κατασκευάσουμε ένα νέο μοντέλο με τετραγωνική λειτουργία :

$$m^{(k+1)}(d) = f_{k+1} + \nabla f_{k+1}^T d + \frac{1}{2} d^T B_{k+1} d \tag{2.7}$$

Μια βασική απαίτηση είναι ότι το gradient του νέου μοντέλου θα πρέπει να ταυτίζεται με το gradient της αντικειμενικής συνάρτησης στις 2 τελευταίες επαναλήψεις $x^{(k)}$ και $x^{(k+1)}$. Έτσι θα έχουμε

$$\nabla m^{(k+1)}(-a^{(k)} d^{(k)}) = \nabla f_{k+1} - a^{(k)} B_{k+1} d^{(k)} = \nabla f_k \tag{2.8}$$

ή ισοδύναμα

$$B_{k+1} a^{(k)} d^{(k)} = \nabla f_{k+1} - \nabla f_k \tag{2.9}$$

Εάν ορίσουμε τα διανύσματα

$$s^{(k)} = x^{(k+1)} - x^{(k)} \text{ και } y_k = \nabla f_{k+1} - \nabla f_k \tag{2.10}$$

τότε η 2.9 γίνεται

$$B_{k+1} s^{(k)} = y_k \tag{2.11}$$

στην οποία αναφερόμαστε ως εξίσωση τέμνουσας ($secant\ equation$).

Η εξίσωση της τέμνουσας (secant) απαιτεί ο θετικά ορισμένος πίνακας B_{k+1} να απεικονίζει το $s^{(k)}$ στο y_k. Για να ισχύει αυτό θα πρέπει να ικανοποιείται η συνθήκη καμπυλότητας

$$s^{(k)T} y_k > 0 \tag{2.12}$$

όταν η συνθήκη αυτή ικανοποιείται, η εξίσωση 2.11 έχει πάντα μια λύση B_{k+1}.

Για να καθορίσουμε πλήρως το B_{k+1} εισάγουμε τη συνθήκη που λέει "μεταξύ όλων των συμμετρικών πινάκων που ικανοποιούν την εξίσωση της τενουσας, ο B_{k+1} είναι ο πιο κοντινός του B_k." Δηλαδή επιλύουμε το πρόβλημα :

$$\min_B ||B - B_k|| \tag{2.13}$$

$$B = B^T,\ Bs^k = y_k \tag{2.14}$$

όπου τα s^k, y_k ικανοποιούν τη συνθήκη καμπυλότητας και ο B_k είναι συμμετρικός και θετικά ορισμένος. Στη σχέση 2.13 μπορούν να χρησιμοποιούν πολλές νόρμες αλλά εμείς θα προτιμήσουμε τη σταθμισμένη νόρμα Frobenius[3] που ορίζεται ως εξής :

$$||A||_W \equiv ||W^{\frac{1}{2}} A W^{\frac{1}{2}}||_F$$

$$||C||_F^2 = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij}^2$$

[3] https://en.wikipedia.org/wiki/Low-rank_approximation#Weighted_low-rank_approximation_problems

Το βάρος W επιλέγεται ως ο πίνακας που ικανοποιεί τη σχέση $Wy_k = s^{(k)}$.

Συγκεκριμένα, μπορούμε να επιλέξουμε ως πίνακα-βάρος τον $W = \overline{G}_k^{-1}$ όπου με $\overline{G}_k$ ορίζεται [7] η μέση τιμή του εσσιανού πίνακα:

$$\overline{G}_k = [\int_0^1 \nabla^2 f(x^{(k)} + \tau a^{(k)} d^{(k)}) d\tau] \tag{2.15}$$

$$y_k = \overline{G}_k a^{(k)} d^{(k)} = \overline{G}_k s^{(k)} \tag{2.16}$$

Με τον σταθμισμένο πίνακα και την παραπάνω νόρμα η λύση της 2.13 είναι

$$B_{k+1} = (I - \gamma_k y_k s^{(k)T}) B_k (I - \gamma_k s^{(k)} y_k^T) + \gamma_k y_k y_k^T \quad (\textbf{DFP}) \tag{2.17}$$

όπου

$$\gamma_k = \frac{1}{y_k^T s^{(k)}} \tag{2.18}$$

Ο τύπος 2.17 καλείται **τύπος ανανέωσης DFP** (DFP updating formula) από τα ονόματα των Davidon, Fletcher και Powell. Αν θεωρήσουμε τον αντίστροφο του πίνακα B_k ως $H_k = B_k^{-1}$ τότε παίρνουμε τον τύπο

$$H_{k+1} = H_k - \frac{H_k y_k y_k^T H_k}{y_k^T H_k y_k} + \frac{s^{(k)} s^{(k)T}}{y_k^T s^{(k)}} \tag{2.19}$$

Όπως πριν έτσι και τώρα θέλουμε να ισχύουν κάποιες συνθήκες για τον H_{k+1}. Οπότε η μοναδική λύση του πίνακα αυτού θα δίνεται από τον τύπο

$$H_{k+1} = (I - \rho_k s^{(k)} y_k^T) H_k (I - \gamma_k y_k s^{(k)T}) + \gamma_k s^{(k)} s^{(k)T} \quad (\textbf{BFGS}) \tag{2.20}$$

όπου

$$\rho_k = \frac{1}{y_k^T s^{(k)}} \tag{2.21}$$

2.4 Μέθοδοι Conjugate Direction

2.4.1 Εισαγωγικά

Οι μέθοδοι $conjugate\ direction$ βρίσκονται στο ενδιάμεσο των μεθόδων Newton και Steepest Descent. Κάποιες βασικές ιδιότητες των μεθόδων $conjugate\ direction$ είναι :

- Επιλύουν τετραγωνικές συναρτήσεις n-μεταβλητών σε n-βήματα.
- Η εφαρμογή τους δεν απαιτεί τη χρήση υπολογισμών εσσιανού πίνακα.
- Δεν απαιτούν αντιστροφή πινάκων και δεν απαιτείται χώρος για αποθήκευση ενός $n \times n$ μητρώου.

Για μια τετραγωνική συνάρτηση n μεταβλητών $f(x) = \frac{1}{2}x^T Qx - x^T b$ με $x \in \mathbb{R}^n$ και $Q = Q^T > 0$, η καλύτερη κατεύθυνση αναζήτησης είναι η κατεύθυνση $Q - conjugate$.

Ορισμός 2.4.1. Δύο κατευθύνσεις $d^{(1)}, d^{(2)}$ λέμε ότι είναι **Q-conjugate** όταν :

$$d^{(1)T} Q d^{(2)} = 0$$

Ορισμός 2.4.2. Έστω Q ένας πραγματικός συμμετρικός $n \times n$ πίνακας. Οι κατευθύνσεις $d^{(0)}, d^{(1)}, ..., d^{(n)}$ είναι **Q-conjugate** αν για κάθε $i \neq j$, έχουμε :

$$d^{(i)T} Q d^{(j)} = 0$$

Πόρισμα 2.4.1. *Έστω Q ένας θετικά ορισμένος και συμμετρικός $n \times n$ πίνακας. Αν οι κατευθύνσεις $d^{(0)}, d^{(1)}, ..., d^{(k)}, k \leq n-1$ είναι μη-μηδενικές και **Q-conjugate** τότε είναι και γραμμικώς ανεξάρτητες.*

Στη συνέχεια θα παρουσιάσουμε το βασικό αλγόριθμο για μεθόδους Conjugate direction. Έστω η τετραγωνική συνάρτηση

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T Q\mathbf{x} - \mathbf{x}^T b,\ Q = Q^T > 0$$

Επειδή Q>0, η συνάρτηση f θα έχει ολικό ελάχιστο το οποίο μπορεί να βρεθεί μέσω της επίλυσης του συστήματος $Qx = b$

Δοθέντος ενός αρχικού σημείου $\mathbf{x}^{(0)}$ και των $d^{(0)}, d^{(1)}, ..., d^{(n-1)}$ Q-conjugate κατευθύνσεων, για $k \geq 0$ ορίζουμε τα ακόλουθα μεγέθη :

$$\mathbf{g}^{(k)} = \nabla f(x^{(k)}) = \mathbf{Q}\mathbf{x}^{(k)} - b, \tag{2.22}$$

$$\alpha^{(k)} = -\frac{\mathbf{g}^{(k)T}\mathbf{d}^{(k)}}{\mathbf{d}^{(k)T}\mathbf{Q}\mathbf{d}^{(k)}}, \tag{2.23}$$

$$x^{(k+1)} = x^{(k)} + \alpha^{(k)}\mathbf{d}^{(k)} \tag{2.24}$$

Θεώρημα 2.4.1. *Για κάθε αρχικό σημείο $x^{(0)}$, ο βασικός αλγόριθμος conjugate direction συγκλίνει στη μοναδική λύση του συστήματος $\boldsymbol{x}^* = \boldsymbol{x}^{(n)}$ σε n βήματα.*

Παράδειγμα 2.4.1. Θα ψάξουμε να βρούμε το ολικό ελάχιστο της συνάρτησης

$$f(x_1, x_2) = \frac{1}{2}\mathbf{x}^T \begin{bmatrix} 4 & 2 \\ 2 & 2 \end{bmatrix} \mathbf{x} - \mathbf{x}^T \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

με τη μέθοδο conjugate direction και για αρχικό σημείο $\mathbf{x}^{(0)} = (0,0)^T$ και κατευθύνσεις $d^{(0)} = [1,0]^T$ και $d^{(1)} = [-\frac{3}{8}, \frac{3}{4}]^T$.

Λύση

Αρχικά βρίσκουμε :

$$\mathbf{g}^{(0)} = -b = [1, -1]^T$$

Επομένως

$$\alpha^{(0)} = -\frac{\mathbf{g}^{(0)T}\mathbf{d}^{(0)}}{\mathbf{d}^{(0)T}\mathbf{Q}\mathbf{d}^{(0)}} = -\frac{[1,-1][1,0]^T}{[1,0]\begin{bmatrix}4 & 2\\ 2 & 2\end{bmatrix}[1,0]^T} = -\frac{1}{4}$$

Έτσι για την πρώτη επανάληψη θα πάρουμε :

$$x^{(1)} = x^{(0)} + a^{(0)}d^{(0)} = \begin{bmatrix}0\\0\end{bmatrix} - \frac{1}{4}\begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}-\frac{1}{4}\\0\end{bmatrix}$$

Συνεχίζοντας τη διαδικασία για τη 2η επανάληψη θα πρέπει πρώτα να υπολογίσουμε

$$g^{(1)} = Qx^{(1)} - b = \begin{bmatrix}4 & 2\\ 2 & 2\end{bmatrix}[-\frac{1}{4}, 0]^T - [-1, 1]^T = [0, -\frac{3}{2}]^T$$

και

$$\alpha^{(1)} = -\frac{\mathbf{g}^{(1)T}\mathbf{d}^{(1)}}{\mathbf{d}^{(1)T}\mathbf{Q}\mathbf{d}^{(1)}} = -\frac{[0,-\frac{3}{2}][-\frac{3}{8},\frac{3}{4}]^T}{[-\frac{3}{8},\frac{3}{4}]\begin{bmatrix}4 & 2\\ 2 & 2\end{bmatrix}[-\frac{3}{8},\frac{3}{4}]^T} = 2$$

Επομένως :

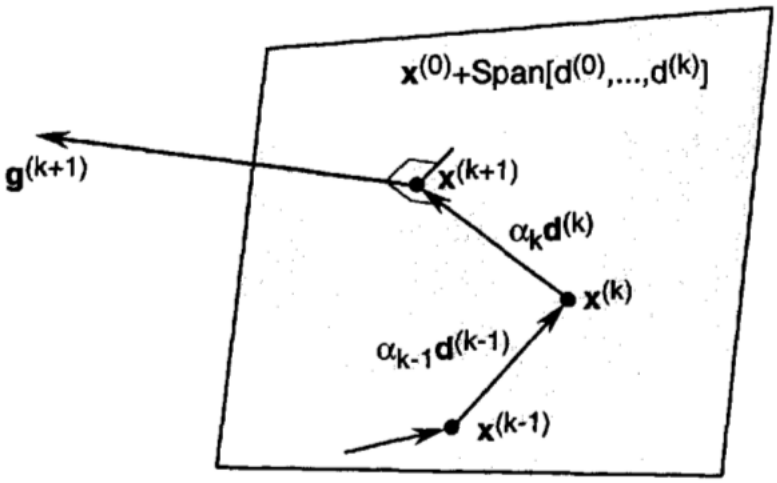
$$x^{(2)} = x^{(1)} + a^{(1)}d^{(1)} = [-\frac{1}{4}, 0]^T + 2[-\frac{3}{8}, \frac{3}{4}]^T = [-1, \frac{3}{2}]^T$$

Τελικά καταλήξαμε στη λύση που είναι η $x^* = x^{(2)}$

Πόρισμα 2.4.2. *Στον αλγόριθμο conjugate direction, αποδεικνύεται ότι*

$$\boldsymbol{g}^{(k+1)T}\boldsymbol{d}^{(i)} = 0, \;\; \forall k, 0 \leq k \leq n-1, \;\; 0 \leq i \leq k$$

Από το προηγούμενο πόρισμα προκύπτει ότι το διάνυσμα $\mathbf{g}^{(k+1)}$ είναι ορθογώνιο με οποιοδήποτε διάνυσμα από τα $d^{(0)}, d^{(1)}, ..., d^{(k)}$. Ακολουθεί και ενδεικτικό σχήμα :



Σχήμα 2.4: Πόρισμα 2.2 - Καθετότητα διανυσμάτων

2.4.2 Conjugate Gradient

Ο αλγόριθμος $conjugate\ gradient$ δεν χρησιμοποιεί ήδη υπολογισμένες κατευθύνσεις αντιθέτως τις υπολογίζει καθώς "τρέχει".

Σε κάθε στάδιο του αλγορίθμου η εκάστοτε κατεύθυνση υπολογίζεται ως ένας γραμμικός συνδιασμός της προηγούμενης κατεύθυνσης και του τρέχον gradient, έτσι ώστε όλες οι κατευθύνσεις να είναι $Q - conjugate$ 2.4.1.

Ομοίως με πριν θα παρουσιάσουμε τα βήματα του παραπάνω αλγορίθμου :

Αρχικά θεωρούμε την τετραγωνική συνάρτηση

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T Q\mathbf{x} - \mathbf{x}^T b,\ Q = Q^T > 0$$

Η πρώτη κατεύθυνση αναζήτης που θα ψάξουμε να βρούμε για αρχικό σημείο $\mathbf{x}^{(0)}$ θα είναι στην κατεύθυνση **steepest descent**. Επόμενως θα έχουμε :

$$\mathbf{d}^{(0)} = -\mathbf{g}^{(0)} = -\nabla f(\mathbf{x}^{(0)}) \tag{2.25}$$

Άρα για την πρώτη επανάληψη λαμβάνουμε το διάνυσμα :

$$\mathbf{x}^{(1)} = \mathbf{x}^{(0)} + a^{(0)}\mathbf{d}^{(0)} \tag{2.26}$$

$$a^{(0)} = -\frac{\mathbf{g}^{(0)T}\mathbf{d}^{(0)}}{\mathbf{d}^{(0)T}Q\mathbf{d}^{(0)}} \tag{2.27}$$

Έπειτα συνεχίζουμε τον αλγόριθμο στην κατέυθυνση $\mathbf{d}^{(1)}$ που θα είναι $Q - conjugate$ στην $\mathbf{d}^{(0)}$. Επομένως επιλέγουμε την κατεύθυνση $\mathbf{d}^{(1)}$ ως ένα γραμμικό συνδυασμό των $\mathbf{d}^{(0)}$ και $\mathbf{g}^{(1)}$. Επομένως όταν φτάσουμε στο (k+1)-βήμα θα έχουμε τον τύπο :

$$\mathbf{d}^{(k+1)} = -\mathbf{g}^{(k+1)} + \beta_k\mathbf{d}^{(k)}. \tag{2.28}$$

όπου οι συντελεστές β_k δίνονται από τον ακόλουθο τύπο

$$\beta_k = \frac{\mathbf{g}^{(k+1)T}Q\mathbf{d}^{(k)}}{\mathbf{d}^{(k)T}Q\mathbf{d}^{(k)}} \tag{2.29}$$

Πόρισμα 2.4.3. *Οι κατευθύνσεις* $\boldsymbol{d}^{(0)}$,$\boldsymbol{d}^{(1)}$,...,$\boldsymbol{d}^{(n-1)}$ *στον* $conjugate\ gradient$ *αλγόριθμο είναι* $Q - conjugate$.

Παράδειγμα 2.4.2. Έστω ότι θέλουμε να βρούμε το ολικό ελάχιστο με χρήση του αλγορίθμου conjugate gradient , της συνάρτησης

$$f(x_1, x_2, x_3) = \frac{3}{2}x_1^2 + 2x_2^2 + \frac{3}{2}x_3^2 + x_1x_3 + 2x_2x_3 - 3x_1 - x_3,$$

με αρχικό σημείο $\mathbf{x}^{(0)} = [0, 0, 0]^T$.

Λύση

Αρχικά θα παραστίσουμε την f στη μορφή $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T Q\mathbf{x} - \mathbf{x}^T b$, όπου :

$$Q = \begin{bmatrix} 3 & 0 & 1 \\ 0 & 4 & 2 \\ 1 & 2 & 3 \end{bmatrix}, \quad b = \begin{bmatrix} 3 \\ 0 \\ 1 \end{bmatrix}$$

Στη συνέχεια υπολογίζουμε το gradient της συνάρτησης

$$\mathbf{g}(\mathbf{x}) = \nabla f(\mathbf{x}) = \begin{bmatrix} 3x_1 + x_3 - 3 \\ 4x_2 + 2x_3 \\ x_1 + 2x_2 + 3x_3 - 1 \end{bmatrix}$$

Επομένως βρίσκοντας τα απαραίτητα μεγέθη για τον υπολογισμό της πρώτης επανάληψης, θα έχουμε :

$$\begin{aligned} \mathbf{g}^{(0)} &= [-3, 0, -1]^T, \\ \mathbf{d}^{(0)} &= -\mathbf{g}^{(0)}, \\ a^{(0)} &= -\frac{\mathbf{g}^{(0)T}\mathbf{d}^{(0)}}{\mathbf{d}^{(0)T} Q \mathbf{d}^{(0)}} = \frac{5}{18}, \\ \mathbf{x}^{(1)} &= [0.8333, 0, 0.2778]^T \end{aligned}$$

Στη συνέχεια του αλγορίθμου υπολογίζουμε διαδοχικά :

$$\begin{aligned} \mathbf{g}^{(1)} = \nabla f(\mathbf{x}^{(1)}) &= [-0.2222, 0.5556, 0.6667]^T, \\ \beta_0 &= \frac{\mathbf{g}^{(1)T} Q \mathbf{d}^{(0)}}{\mathbf{d}^{(0)T} Q \mathbf{d}^{(0)}} = 0.08025 \end{aligned}$$

Η νέα κατεύθυνση θα είναι

$$\begin{aligned} \mathbf{d}^{(1)} = -\mathbf{g}^{(1)} + \beta_0 \mathbf{d}^{(0)} &= [0.4630, -0.5556, -0.5864]^T, \\ a^{(1)} &= -\frac{\mathbf{g}^{(1)T}\mathbf{d}^{(1)}}{\mathbf{d}^{(1)T} Q \mathbf{d}^{(1)}} = 0.2187 \end{aligned}$$

Έτσι βρίσκουμε στη δεύτερη επανάληψη το διάνυσμα :

$$\mathbf{x}^{(2)} = \mathbf{x}^{(1)} + a^{(1)}\mathbf{d}^{(1)} = [0.9346, -0.1215, 0.1495]^T$$

Η διαδικασία του αλγορίθμου συνεχίζεται με ανάλογο τρόπο.

2.5 Μέθοδοι Ολικής Βελτιστοποίησης

2.5.1 Nelder-Mead

Ορισμός 2.5.1. Στη γεωμετρία ο όρος **Simplex** αποτελεί γενίκευση της έννοιας του τριγώνου ή του τετραέδρου για αυθέραιτες διαστάσεις. Γενικότερα ένα **k-Simplex** αποτελεί

ένα n-διάστατο γεωμετρικό πολύτοπο[4] που είναι η κυρτή θήκη n+1 κορυφών. Πιο τυπικά έστω n+1 σημεία-κορυφές $x_0, x_1, ..., x_n \in \mathbb{R}^n$ ανα δύο γραμμικώς ανεξάρτητα. Τότε το Simplex που ορίζεται από αυτά τα σημεία είναι το σύνολο :

$$C = \{\theta_0 x_0 + \theta_1 x_1 + ... + \theta_n x_n | \sum_{i=0}^{n} \theta_i = 1,\ \theta_i \geq 0\ \forall i\} \tag{2.30}$$

Η μέθοδος **Nelder Mead** [5] προτάθηκε από τους John Nelder και Roger Mead το (1965). Είναι μια αριθμητική μέθοδος που χρησιμοποιείται για εύρεση του ελάχιστου ή μέγιστου μιας αντικειμενικής συνάρτησης. Εφαρμόζεται κυρίως σε μη γραμμικά προβλήματα βελτιστοποίησης για τα οποία πιθανόν δε γνωρίζουμε τις παραγώγους.

Ο αλγόριθμος μπορεί να επεκταθεί σε προβλήματα ελαχιστοποίησης με περιορισμούς με την προσθήκη μιας συνάρτησης ποινής (**penalty function**). Η μέθοδος χρησιμοποιεί τη διατήρηση ενός Simplex S. Το Simplex που διατηρείται κατα τη διάρκεια του αλγορίθμου μπορεί να θεωρηθεί ως ένα πολύγωνο με n+1 κορυφές . Για n=2 θα έχουμε ένα τρίγωνο όπου ο αλγόριθμος Nelder-Mead μπορεί να γίνει εύκολα αντιληπτός.

Οι κορυφές $\{x_i\}_{i=1}^{n}$ είναι διατεταγμένες σύμφωνα με τις συναρτησιακές τιμές

$$f(x_1) \leq f(x_2) \leq ... \leq f(x_{n+1})$$

όπου x_1 αποτελεί την καλύτερη κορυφή (**best vertex**) και η x_{n+1} τη χειρότερη (**worst vertex**).

Εάν υπάρχουν κορυφές με την ίδια τιμή με τη x_1 τότε η καλύτερη κορυφή δεν είναι μοναδική. Το σκεπτικό του αλγορίθμου είναι η αντικατάσταση της χειρότερης κορυφής x_{n+1} με ένα νέο σημείο της μορφής

$$x(\mu) = (1 + \mu)\overline{x} - \mu x_n$$

όπου $\overline{x}$ είναι το κεντροειδές της κυρτής θήκης των σημείων x_0 έως x_{n-1} και ορίζεται ως

$$\overline{x} = \frac{1}{n} \sum_{i=i}^{n} x_i$$

Συγκεκριμένα ο αλγόριθμος Nelder-Mead για ένα Simplex με 3 κορυφές u, v, w και αντίστοιχες συναρτησιακές τιμές $f(u), f(v), f(w)$ ακολουθεί τα παρακάτω βήματα.

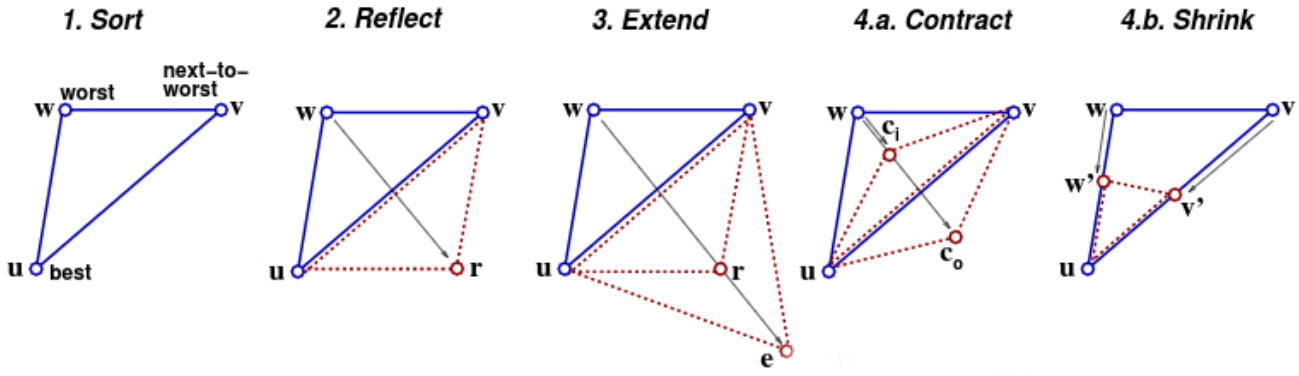
1. Διάταξη (**Sort**) των κορυφών u, v, w τ.ω. $f(u) < f(v) < f(w)$.

   Το σημείο u αποτελεί την "καλύτερη" κορυφή και το w τη "χειρότερη" κορυφή.

2. "Αντανάκλαση" (**Reflect**) της χειρότερης κορυφής w από το κεντροειδές $\overline{x}$ των σημείων που μένουν u,v και υπολογισμός του σημείου αντανάκλασης r όπως και της συναρτησιακής τιμής αυτού.

   Εάν $f(u) < f(r) < f(v)$ τότε αντικαθιστώ το w με το r και πηγαίνω στο βήμα 5.

[4] https://en.wikipedia.org/wiki/Polytope



Σχήμα 2.5: Τα βήματα του αλγορίμου Nelder-Mead (για n = 2) [2].

3. Εάν $f(r) < f(u)$, άρα έχω νέα καλύτερη κορυφή, τότε "επεκτείνω" (**Extend**) το r, πέρα από το μέσο όρο των u, v σε ένα σημείο e και υπολογίζω το $f(e)$.

   - Εάν ισχύει ότι $f(e) < f(r)$, τότε αντικαθιστώ τη χειρότερη κορυφή w με το σημείο e και πηγαίνω στο βήμα 5
   - Διαφορετικά αντικαθιστώ το w με το σημείο αντανάκλασης r και πηγαίνω στο βήμα 5.

4. Εάν οι ανισότητες των βημάτων 2 και 3 δεν ικανοποιούνται, τότε σίγουρα το r είναι χειρότερη κορυφή από το v δηλαδή $f(v) < f(r)$, και πιθανόν να υπάρχει μικρότερη συναρτησιακή τιμή μεταξύ των σημείων w και r.

   Έτσι κάνω σύμβαση (**Contract**) για το w με ένα σημείο c το οποίο βρίσκεται μεταξύ των w και r, υπολογίζοντας και το $f(w)$. Γενικά κάποιες καλές τιμές για το c είναι 1/4 και 3/4 της απόστασης από το w στο r. Τα σημεία αυτά καλούνται **εσωτερικό** και **εξωτερικό** σημείο σύμβασης, και συμβολίζονται αντίστοιχα c_i και c_o.

   - Εάν $min\{f(c_i), f(c_0)\} < f(v)$ τότε αντικαθιστώ το w με το καλύτερο από τα σημεία σύμβασης c_i και c_0.
   - Διαφορετικά γίνεται "συστολή" του Simplex στην καλύτερη κορυφή u και πηγαίνω στο βήμα 5.

5. Έλεγχος σύγκλισης.

   Το Simplex συγκλίνει εάν είναι όπως λέμε "επαρκώς" μικρό και αν συναρτησιακές τιμές βρίσκονται "επαρκώς" κοντά. Η επάρκεια καθορίζεται από τις ακρίβειες ε_x και ε_f αντίστοιχα. Ακολουθούν οι απαραίτητες συνθήκες για τη σύγκλιση.

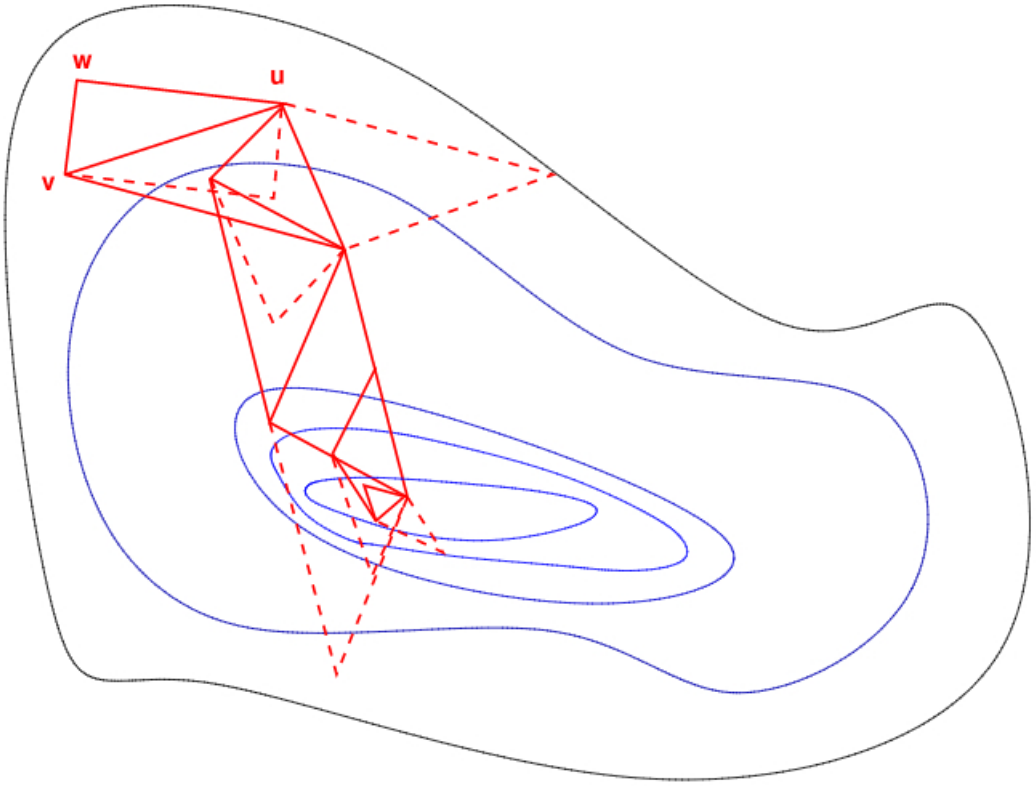
$$2\underset{n=2}{max}|\frac{[u,v]-[v,w]}{[u,v]+[v,w]}| < \varepsilon_x, \quad 2\underset{n\geq 2}{max}|\frac{\mathbf{S}(:,1:n)\text{–}\mathbf{S}(:,2:n+1)}{\mathbf{S}(:,1:n)+\mathbf{S}(:,2:n+1)}| < \varepsilon_x$$

και οι αντίστοιχες συνθήκες για τις συναρτησιακές τιμές :

$$|\frac{f(w)-f(u)}{f(u)+10^{-9}}|_{n=2} < \varepsilon_f, \quad |\frac{f(n+1)-f(1)}{f(1)+10^{-9}}|_{n\geq 2} < \varepsilon_f$$

Όπου $\mathbf{S}(:, 1:n)$ αντιστοιχεί στο Simplex που σχηματίζεται κατά την εκτέλεση του αλγορίθμου, από το πρώτο μέχρι το n-σημείο.

- Άν οι παραπάνω συνθήκες ισχύουν τότε έχουμε σύγκλιση και ο αλγόριθμος τερματίζει.
- Άν το πλήθος των επαναλήψεων έχει ξεπεράσει κάποιο όριο τότε ο αλγόριθμος τερματίζει.



Σχήμα 2.6: Η ακολουθία επτά βημάτων του αλγορίθμου Nelder-Mead. Οι διακεκομμένες γραμμές δείχνουν τις προσπάθειες βημάτων ανάκλασης [2].

2.5.2 Basin Hopping

Ο αλγόριθμος αυτός έρχεται από τον χώρο της φυσικοχημείας, όπου οι επιστήμονες μελετάνε την ευστάθεια μοριακών συστημάτων ως το ελάχιστο της ολικής τους ενέργειας. Όσο αυξάνεται η πολυπλοκότητα των μοριακών δομών όμως, τόσο πιο δύσκολο

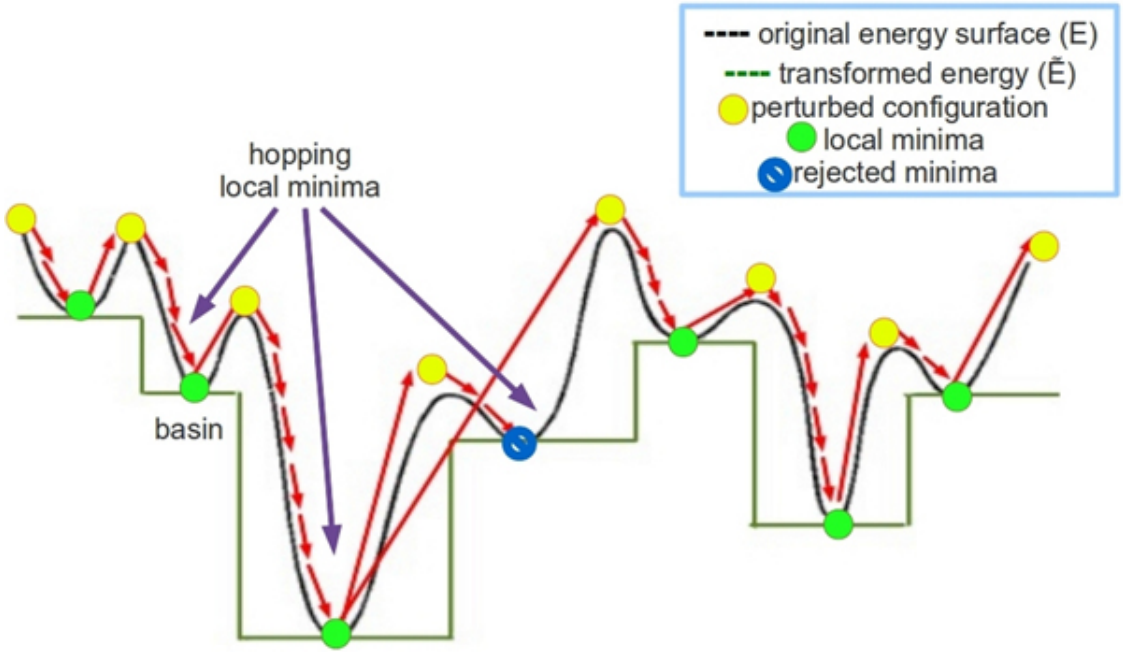
είναι να βρει κανείς ολικά ελάχιστα του ενεργιακού επίπεδου του κάθε συστύματος, μιας και στις χαμηλές ενέργειες (θερμοκρασίες) είναι πολύ πιθανό το σύστημα να ισορροπήσει σε κάποιο τοπικό ελάχιστο. Η ενεργειακή συνάρτηση είναι γνωστή στον χώρο της φυσικοχημείας και ως δυναμικό Lennard-Jones [5] και στην περίπτωσή μας που έχουμε πολυμοριακά συστήματα δίνεται από την ακόλουθη σχέση (Lennard-Jones cluster [4]):

$$E_n = 2\sum_{i=1}^{n}\sum_{j=1}^{n}(\frac{1}{r_{ij}^{12}} - \frac{1}{r_{ij}^{6}}) \tag{2.31}$$

, όπου n είναι ο αριθμός των σωματιδίων του συστήματός (cluster) υπό μελέτη, E_n η ενέργεια του $n^{\text{στού}}$ σωματιδίου, και r_{ij} η ευκλείδια απόσταση μεταξύ των σωματιδίων i και j.

Για αυτό τον λόγο οι Wales κ.α. [8] ανέπτυξαν την μέθοδο basin hopping που χρησιμοποιεί κάποια από τις μεθόδους εύρεσης τοπικών ελαχίστων που ήδη έχουμε αναφέρει για να βρούμε όλα τα τοπικά ελάχιστα της ενεργειακής συνάρτησης, και έπειτα με το κριτήριο Metropolis, η λύση μας μεταβαίνει στο επόμενο ενεργειακό επίπεδο ή παραμένει εκεί που είναι.

Στο σχήμα που ακολουθεί βλέπουμε με την μπλε ομαλή καμπύλη την ενεργειακή συνάρτηση E όπως φαίνεται, ενώ η μετασχηματισμένη ενέργεια $\tilde{E}$ φαίνεται με τα πράσινα κατώφλια (basins), που ορίζουν τα τοπικά ελάχιστα της αρχικής συνάρτησης E.



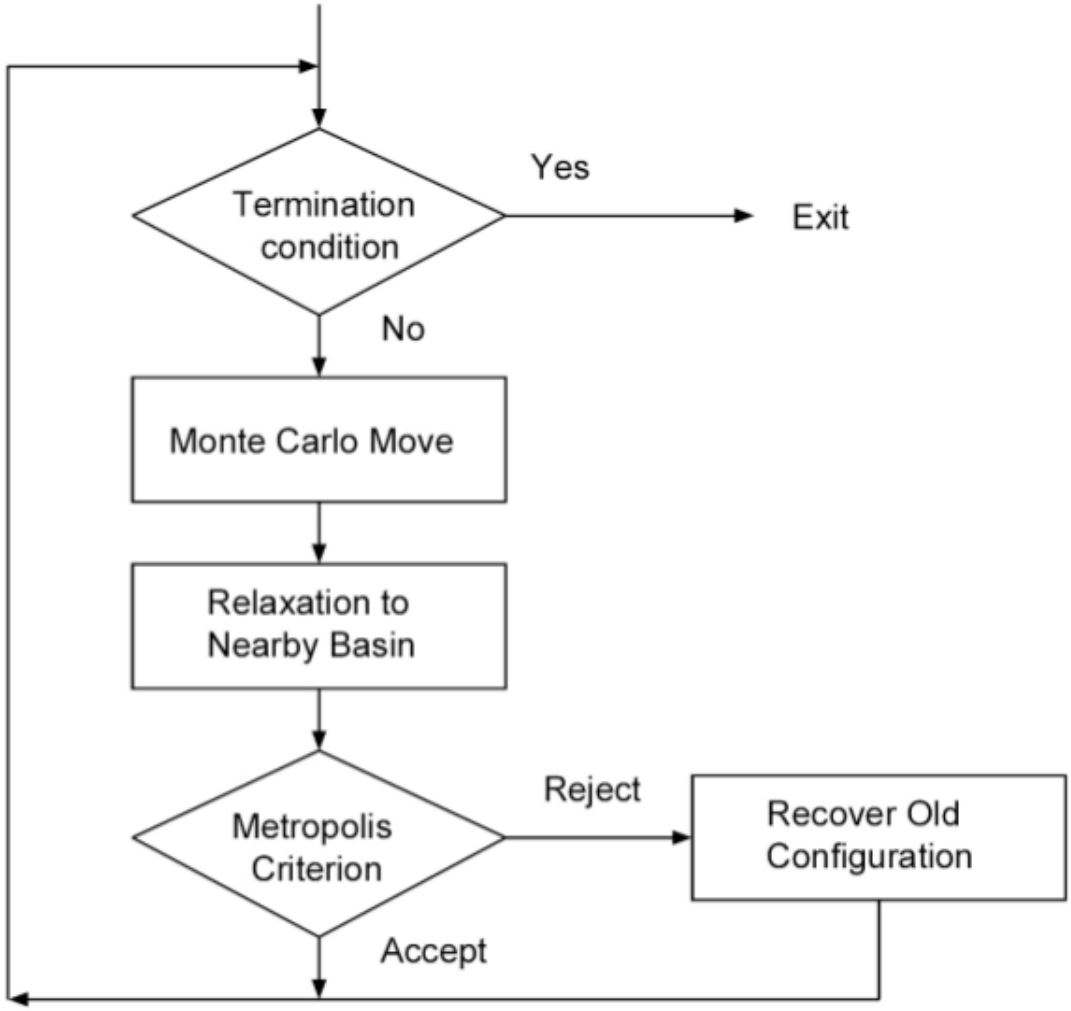
Σχήμα 2.7: Εικόνα της λειτουργίας του αλγορίθμου basin hopping [3]

Το κριτήριο μετάβασης από το ένα τοπικό ελάχιστο σε κάποιο άλλο, το δίνει το κριτή-

[5] https://en.wikipedia.org/wiki/Lennard-Jones_potential

ριο του Metropolis [9] που μετακινεί τυχαία την λύση/σωματίδιο i κατά $\Delta\mathbf{r}_i$ αν η μεταβολή της ολικής ενέργειας $\Delta E = E(\mathbf{r}_1, ..., \mathbf{r}_i + \Delta\mathbf{r}_i, ..., r_N) - E(\mathbf{r}_1, ..., \mathbf{r}_i, ..., \mathbf{r}_N)$ είναι αρνητική ή μηδέν. Δηλαδή αν πάει σε μικρότερο τοπικό ελάχιστο ή μείνει στο ίδιο. Αν η μεταβολή της ενέργειας είναι θετική, τότε το σωματίδιο έχει πιθανότητα $exp(-\Delta E/T)$ να αλλάξει θέση, με T να υποδηλώνει την θερμοκρασία.

Το λογικό διάγραμμα του αλγόριθμου basin hopping είναι το ακόλουθο, όπου ως κριτήριο τερματισμού δίνουμε τον μέγιστο αριθμό των μεταβάσεων που θέλουμε να κάνει η λύση της συναρτησης υπό βελτιστοποίηση.



Σχήμα 2.8: Το λογικό διάγραμμα του αλγορίθμου Basin Hopping (Iwamatsu et al [4])

Κεφάλαιο 3

Εισαγωγή στα Jupyter Notebooks

3.1 Οδηγίες εγκατάστασης

3.1.1 Στο λειτουργικό σύστημα MS Windows

Σε αυτό το λειτουργικό σύστημα αρκεί ο χρήστης να εγκαταστήσει το πακέτο της Python Anaconda, το οποίο εγκαθιστά όλα τα απαραίτητα πακέτα, την IPython, καθώς και τις βιβλιοθήκες NumPy, SciPy, Pandas, Matplotlib. Ό,τι δηλαδή χρειάζεται κανείς για να αναπαραγάγει τα αποτελέσματα αυτής της εργασίας.

Για να εγκαταστήσει κανείς το Anaconda θα χρειαστεί να πάει στον ιστότοπο της Anaconda και να επιλέξει την έκδοση 2.7 της Python (”Python 2.7 version”).

Έπειτα, αφότου κατεβάσει το πακέτο για υπολογιστή 64bit, το εγκαθιστά και από την Έναρξη/Start του Windows πηγαίνει στα προγράμματα, στον φάκελο Anaconda.

3.1.2 Στο λειτουργικό σύστημα GNU/Linux

Οι τρεις πιο διαδεδομένες «γεύσεις» (flavors) αυτού του λειτουργικού συστήματος είναι σύμφωνα με τον ιστότοπο Distrowatch [1] το Mint, το Ubuntu και το Debian.

Το Mint είναι παράγωγο GNU/Linux του Ubuntu. Ενώ το Ubuntu είναι παράγωγο του Debian. Συνεπώς και τα τρία χρησιμοποιούν το ίδιο πρόγραμμα εγκατάστασης πακέτων, το apt-get. Ο κώδικας αυτής της εργασίας έχει δοκιμαστεί και τρέχει τόσο σε Ubuntu όσο και σε Lubuntu, οπότε αναμένουμε πως θα τρέχει και σε Debian και Mint.

Η εγκατάσταση των Jupyter Notebooks λοιπόν αφορά και τα τρία (Mint, Ubuntu, Debian), αλλά προτείνουμε να διαλέξετε κάποιο από τα Ubuntu (Ubuntu, Lubuntu, Kubuntu). Οι διαδικασίες που ακολουθούν έχουν ως αυστηρό προαπαιτούμενο ο υπολογιστής μας να είναι συνδεδεμένος στο διαδίκτυο.

[1] `https://distrowatch.com/dwres.php?resource=major`

Εγκατάσταση της Python

Ανοίγουμε ένα τερματικό εντολών. Έπειτα γράφουμε:

```
sudo apt-get install python2.7 -y
```

Θα μας ζητήσει τον κωδικό του username, και αφότου τον δώσουμε θα εγκαταστήσει την python2.7 και όλα τα σχετικά.

Εγκατάσταση του pip

Εφόσον εγκαταστήθηκε η Python, ήδη υπάρχει και το pip, αλλά σε παλαιότερη έκδοση. Για να κατεβάσουμε την τελευταία έκδοση γράφουμε στο τερματικό:

```
pip install -U pip setuptools
```

Εγκατάσταση του virtual environment

Μια καλή τακτική όταν κανείς δουλεύει με τις βιβλιοθήκες της Python σε κάποιο project είναι να φτιάξει ένα virtual environment για το συγκεκριμένο project το οποίο δεν θα επικοινωνεί με τα υπόλοιπα πακέτα της Python που είναι ήδη εγκατεστημένα στον υπολογιστή.

Για να εγκαταστήσουμε το virtual environment, και να δημιουργήσουμε ένα φάκελο με τις βιβλιοθήκες μας για ένα project με όνομα NumericalOptimization και για να το ενεργοποιήσουμε, γράφουμε στο τερματικό μας:

```
pip install virtualenv
virtualenv NumericalOptimization
source NumericalOptimization/bin/activate
```

Τώρα η γραμμή εντολών μας θα πρέπει να φαίνεται κάπως έτσι:

```
(NumericalOptimization)milia@Newton$_
```

για έναν χρήστη με όνομα *milia* και με όνομα υπολογιστή *Newton*. Η παρένθεση *(Numerical Optimization)* μπροστά από το όνομα του χρήστη δείχνει πως χρησιμοποιούμε το περιβάλλον της Python ονόματι «NumericalOptimization».

Εγκατάσταση των Jupyter Notebooks, NumPy, SciPy, Matplotlib

Για να εγκαταστήσουμε τα παραπάνω θα πρέπει να ακολουθήσουμε τις ακόλουθες ενέργειες πρώτα:

```
1 sudo apt-get install build-essential python-dev
2 sudo apt-get install libblas-dev liblapack-dev libatlas-base-dev
    gfortran
```

Τώρα μπορούμε να εγκαταστήσουμε τα προγράμματά μας με το pip:

```
1 pip install scipy numpy matplotlib jupyter
```

Αναβάθμιση των προγραμμάτων που εγκαταστήσαμε με το pip

Δεν είναι σπάνιο φαινόμενο να θελήσουμε να ξαναπιάσουμε το συγκεκριμένο project μετά από μερικούς μήνες. Μέχρι τότε θα έχουν βγει νέες εκδόσεις των πακέτων που θα'χουμε ήδη κατεβάσει. Για να κατεβάσουμε τις νέες εκδόσεις όλες μαζί υπάρχει ένα εξαίρετο εργαλείο, το `pip-review` το οποίο και καλό είναι να εγκαταστήσουμε:

```
1 pip install pip-review
```

Έτσι, για την ανανέωση των προγραμμάτων της Python με τις τελευταίες εκδόσεις τους αρκεί να γράψουμε στο τερματικό:

```
1 pip-review --local --interactive
```

και να επιλέξουμε την επιλογή "All" [2].

3.2 Οδηγίες χρήσης

Ωραία, τώρα που εγκαταστήσαμε τα Jupyter Notebooks πώς τα ξεκινάμε; Το σημαντικό τους προσόν είναι ότι τρέχουν στον browser μας.

[2] `http://stackoverflow.com/questions/2720014/upgrading-all-packages-with-pip`

3.2.1 Στο MS Windows

Για να ανοίξουμε ένα Jupyter Notebook στο λειτουργικό σύστημα Windows 7 ακολουθούμε τις ακόλουθες ενέργειες:

1. Πηγαίνουμε στο κουμπί Start/Έναρξη
2. Πηγαίνουμε στο All Programms/Προγράμματα
3. Βρίσκουμε τον φάκελο Anaconda και τον πατάμε με αριστερό κλικ του ποντικιού
4. Επιλέγουμε την εφαρμογή Jupyter Notebook και την πατάμε με το αριστερό κλικ
5. Παρατηρούμε πως άνοιξε ένα νέο υποπαράθυρο (tab) στον browser μας μαζί και με ένα μικρό παράθυρο εντολών (command line).
6. Αγνοούμε το παράθυρο εντολών και πηγαίνουμε στο tab του browser που άνοιξε
7. Πατάμε New και επιλέγουμε Python 2, κάτω από τον τίτλο Notebooks.

Αν όλα έχουν πάει καλά, και τα Jupyter Notebooks εχουν ρυθμιστεί έτσι ώστε να ανοίγουν σε έναν φάκελο που έχετε δικαιώματα γραφής (write permissions) στον σκληρό, θα βλέπετε ένα κενό Jupyter Notebook στο συγκεκριμένο tab, με ένα κενό κελί όπου μπορείτε να γράψετε κάποιον κώδικα Python.

Αφότου γράψετε κάποια εντολή σε γλώσσα Python στο συγκεκριμένο κελί, για την εκτέλεσή της πατήστε Shift + Enter, οπως στην Mathematica. Τα Jupyter Notebooks ξεκίνησαν ως προέκταση των IPython Notebooks που εφευρέθηκαν από έναν ερευνητή που ήθελε να μεταφέρει την ευκολία των Mathematica Notebooks στην Python. Περισσότερες πληροφορίες για την ιστορία των Jupyter Notebooks και της IPython μπορείτε να βρείτε στην wikipedia[3].

3.2.2 Στο Ubuntu

Ανοίγουμε ένα τερματικό εντολών. Μπαίνουμε στο virtual environment που έχουμε φτιάξει όπου είναι εγκατεστημένο και το Jupyter Notebook:

```
1 source NumericalOptimization/bin/activate
```

Τώρα πάμε στο directory που θέλουμε να δουλέψουμε ή όπου υπάρχει το notebook που φτιάξαμε την προηγούμενη φορά. Έστω πως βρίσκεται στο home directory `/home/milia/`. Οπότε μπορούμε να γράψουμε:

[3] `https://en.wikipedia.org/wiki/IPython`

```
cd
jupyter notebook
```

και θα δούμε στο τερματικό

```
[I 21:45:24.587 NotebookApp] Writing notebook server cookie secret to
/run/user/1000/jupyter/notebook_cookie_secret
[I 21:45:28.190 NotebookApp] Serving notebooks from local directory:
/home/milia/
[I 21:45:28.191 NotebookApp] 0 active kernels
[I 21:45:28.191 NotebookApp] The Jupyter Notebook is running at:
http://localhost:8888/
[I 21:45:28.191 NotebookApp] Use Control-C to stop this server and
shut down all kernels (twice to skip confirmation).
```

Λογικά θα ανοίξει αυτόματα τον ορισμένο ως κύριο browser μας στην διεύθυνση `http://localhost:8888/`, αλλιώς μπορείτε να το αντιγράψετε και επικολλήσετε μόνοι σας στον browser που έχετε ανοιχτό.

3.3 Η βιβλιοθήκη SciPy

Η βιβλιοθήκη SciPy[4] ανήκει στην συλλογή πακέτων για επιστημονικούς υπολογισμούς στην γλώσσα Python, γνωστή και ως SciPy Stack. Εμείς θα ασχοληθούμε μόνο με την βιβλιοθήκη SciPy και θα αναφερθούμε και στην βιβλιοθήκη NumPy.

Από την SciPy μας ενδιαφέρει η συλλογή αλγορίθμων που αφορούν επίλυση προβλημάτων βελτιστοποίησης. Η κλάση αυτών των μεθόδων ονομάζεται optimize [5]. Όπως μπορεί να δει κανείς στο εγχειρίδιο χρήσης (tutorial) στον σχετικό σύνδεσμο, αυτή η κλάση περιέχει όλες τις μεθόδους τις οποίες εξετάζουμε σε αυτή την εργασία. Μια αναλυτική λίστα για τις συγκεκριμένες μεθόδους υπάρχει σε αυτόν τον σύνδεσμο.

Πιο συγκεκριμένα, από την κλάση `scipy.optimize` θα χρησιμοποιήσουμε τις μεθόδους `minimize` και `basinhopping`. Η πρώτη μέθοδος βρίσκει το ελάχιστο δεδομένης συνάρτησης ανάλογα με την μέθοδο που θα επιλέξουμε (Nelder-Mead, Powell, CG, Newton-CG, BFGS) ενώ η δεύτερη χρησιμοποιεί τον στοχαστικό αλγόριθμο Basin Hopping για την εύρεση του ελάχιστου μιας δεδομένης συνάρτησης.

Περισσότερες λεπτομέρειες για το πώς λειτουργεί η `minimize` και η `basinhopping`

[4] `https://www.scipy.org/about.html`

[5] `https://docs.scipy.org/doc/scipy-0.18.1/reference/tutorial/optimize.html`

μπορούν να βρεθούν στον κώδικά τους που υπάρχει διαθέσιμος στο διαδίκτυο[6] [7]. Στην παρούσα εργασία θα εξετάσουμε τις συγκεκριμένες μεθόδους όσον αφορά την απόδοσή τους σε συγκεκριμένα προβλήματα ως κλειστά κουτιά (black boxes).

Η NumPy στην οποία αναφερθήκαμε πιο πάνω και την οποία καλούμε στον κώδικά μας (`import numpy as np`) περιέχει όλες τις βασικές συναρτήσεις που θα χρειαστούμε για την υλοποίηση των προβλημάτων προς επίλυση.

[6] `https://docs.scipy.org/doc/scipy-0.18.1/reference/generated/scipy.optimize.minimize.html#scipy.optimize.minimize`

[7] `https://docs.scipy.org/doc/scipy-0.18.1/reference/generated/scipy.optimize.basinhopping.html#scipy.optimize.basinhopping`

Κεφάλαιο 4

Πειραματική διαδικασία

4.1 Παρουσίαση των προβλημάτων προς επίλυση

Από την εργασία των More et al [6] πήραμε τα ακόλουθα προβλήματα ώστε να ελέγξουμε την απόδοση των υλοποιήσεων των παραπάνω αλγορίθμων της βιβλιοθήκης SciPy.

1. **Συνάρτηση Badly scaled του Powell**

$$f(\mathbf{x}) = (10^4 x_1 x_2 - 1)^2 + (e^{-x_1} + e^{-x_2} - 1.0001)^2 \tag{4.1}$$

όπου $\mathbf{x}_0 = (0, 1)^T$, $\mathbf{x}^* = (1.098 \times 10^{-5}, 9.106)^T$, $f(\mathbf{x}^*) = 0$.

```
f1a = lambda x: (10**4)*x[0]*x[1] - 1
f2a = lambda x: (m.exp(-x[0]) + m.exp(-x[1]) - 1.0001)
powell_func = lambda x: square(f1a(x)) + square(f2a(x))
powell_f = lambda x: powell_func([x[0],x[1]])

def powell_f2(x):
    f = (square((10**4)*x[0]*x[1] - 1) + square((m.exp(-x[0]) +
      m.exp(-x[1]) - 1.0001)))
    df = np.zeros(2)
    df[0] = 2*((10**4)*x[0]*x[1] - 1)*(10**4)*x[1] -2*m.exp(-x
    [0])
    df[1] = 2*((10**4)*x[0]*x[1] - 1)*(10**4)*x[0] -2*m.exp(-x
    [1])
    return f,df
```

Listing 4.1: ”Η συνάρτηση Powell σε κώδικα Python”

2. **Συνάρτηση Badly scaled του Brown**

$$f(\mathbf{x}) = (x_1 - 10^6)^2 + (x_2 - 2 \times 10^{-6} + (x_1 x_2 - 2))^2 \tag{4.2}$$

όπου $\mathbf{x}_0 = (1, 1^T)$, $\mathbf{x}^* = (10^6, 2 \times 10^6)^T$, $f(\mathbf{x}^*) = 0$.

```
f1b = lambda x: x[0] - 10**6
```

```
f2b = lambda x: x[1] - 2*10**(-6)
f3b = lambda x: x[0]*x[1] - 2

brown_func = lambda x: (square(f1b(x)) + square(f2b(x)) +
                square(f3b(x)))

def brown_func2(x):
    f = brown_func(x)
    df = np.zeros(2)
    df[0] = 2*(x[0] - 10**6) + 0 + 2*(x[0]*x[1] - 2)*x[1]
    df[1] = 0 + 2*(x[1] - 2*10**(-6)) + 2*(x[0]*x[1] - 2)*x[0]
    return f,df
```

Listing 4.2: "Η συνάρτηση Brown σε κώδικα Python"

3. **Συνάρτηση του Beale**

$$f(\mathbf{x}) = (1.5 - x_1(1 - x_2))^2 + (2.25 - x_1(1 - x_2^2))^2 + (2.625 - x_1(1 - x_2^3))^2 \quad (4.3)$$

όπου $\mathbf{x}_0 = (1, 1)^T$, $\mathbf{x}^* = (3, 0.5)^T$, $f(\mathbf{x}^*) = 0$.

```
const = [1.5, 2.25, 2.625]
f1c = lambda x: const[0] -x[0]*(1 - x[1])
f2c = lambda x: const[1] -x[0]*(1 - x[1]*x[1])
f3c = lambda x: const[2] -x[0]*(1 - x[1]*x[1]*x[1])

beal_func = lambda x: (square(f1c(x)) + square(f2c(x)) +
            square(f3c(x)))
def beal_func2(x):
    f = beal_func(x)
    df = np.zeros(2)
    df[0] = (2*(const[0] -x[0]*(1 - x[1]))*(-(1 - x[1]))
            +2*(const[1] -x[0]*(1 - x[1]*x[1]))*(-(1 - x[1]*x[1])
    )
            +(2*(const[2] -x[0]*(1 - x[1]*x[1]*x[1]))*
             (-(1 - x[1]*x[1]*x[1]))))

    df[1] = (2*(const[0] -x[0]*(1 - x[1]))*x[0]
            +2*(const[1] -x[0]*(1 - x[1]*x[1]))*(2*x[1])
            +(2*(const[2] -x[0]*(1 - x[1]*x[1]*x[1]))*
             (3*x[1]*x[1])))
    return f,df
```

Listing 4.3: "Η συνάρτηση Beale σε κώδικα Python"

4. **Συνάρτηση ελικοειδούς κοιλάδας (Helical valey)**

$$f(\mathbf{x}) = 100(x_3 - 10\theta(x_1, x_2))^2 + (\sqrt{x_1^2 + x_2^2} - 1)^2 + x_3^2 \quad (4.4)$$

$$\theta(x_1, x_2) = \begin{cases} (1/2\pi) \cdot arctan(x_2/x_1), \text{αν } x_1 > 0 \\ 0.5 + (1/2\pi) \cdot arctan(x_2/x_1), \text{αν } x_1 < 0 \end{cases} \quad (4.5)$$

```
def theta(x0,x1):
    if x0 > 0:
        return np.arctan(x1/x0)/PI
    if x0 < 0:
        return 0.5 + np.arctan(x1/x0)/PI

f1d = lambda x: 10*(x[2] - 10*theta(x[0],x[1]))
f2d = lambda x: 10*((x[0]**2 + x[1]**2)**(1/2) - 1.0)
f3d = lambda x: x[2]

def helic_func(x): return (square(f1d(x)) + square(f2d(x)) +
                square(f3d(x)))

def helic_func2(x):
    f = helic_func(x)
    df = np.zeros(3)
    df[0] = ( 200*(x[2] - 10*theta(x[0],x[1]))*(x[1]/(x[0]**2 +
          x[1]**2)) + 20*((x[0]**2 + x[1]**2)**(1/2)
              - 1.0)*(0.5*((x[0]**2 + x[1]**2)**(-1/2))*2*x[0])
              + 0 )
    df[1] = ( 200*(x[2] - 10*theta(x[0],x[1]))*(-x[1]/(x[0]**2 +
          x[1]**2)) + 20*((x[0]**2 + x[1]**2)**(1/2)
              - 1.0)*(0.5*((x[0]**2 + x[1]**2)**(-1/2))*2*x[1])
              + 0 )
    df[2] = ( 20*(10*(x[2] - 10*theta(x[0],x[1]))) + 0  + 1 )
    return f,df
```

Listing 4.4: ”Η συνάρτηση Helical valey σε κώδικα Python”

όπου $\mathbf{x}_0 = (-1, 0, 0)^T$, $\mathbf{x}^* = (1, 0, 0)^T$, $f(\mathbf{x}^*) = 0$.

5. **Συνάρτηση του Wood**

$$f(\mathbf{x}) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 + 90(x_4 - x_3^2)^2 + (1 - x_3)^2 + 10(x_2 + x_4 - 2)^2 + 0.1(x_2 - x_4)^2 \tag{4.6}$$

όπου $\mathbf{x}_0 = (-3, -1, -3, -1)^T$, $\mathbf{x}^* = (1, 1, 1, 1)^T$, $f(\mathbf{x}^*) = 0$.

```
f1e = lambda x: (10*(x[1] - x[0]*x[0]))
f2e = lambda x: (1 - x[0])
f3e = lambda x: np.sqrt(90)*(x[3] - x[2]*x[2])
f4e = lambda x: (1 - x[2])
f5e = lambda x: np.sqrt(10)*(x[1] + x[3] - 2)
f6e = lambda x: (1/np.sqrt(10))*(x[1] - x[3])

wood_func = lambda x: (square(f1e(x)) + square(f2e(x)) +
            square(f3e(x)) + square(f4e(x)) +
                        square(f5e(x)) + square(f6e(x)))
def wood_func2(x):
    f = wood_func(x)
    df=np.zeros(4)
    df[0] = 2*100*(x[1]-x[0]*x[0])*(-2*x[0]) - 2*(1-x[0])
    df[1] = 2*100*(x[1]-x[0]*x[0]) + 2*10*(x[1]+x[3]-2)
        + 0.1*2*(x[1]-x[3])
```

```
    df[2] = 2*90*(x[3]-x[2]*x[2])*(-2*x[2]) - 2*(1-x[2])
    df[3] = 2*90*(x[3]-x[2]*x[2]) + 2*10*(x[1]+x[3]-2)
        - 2*0.1*(x[1]-x[3])
    return f,df
```

Listing 4.5: ”Η συνάρτηση Wood σε κώδικα Python”

6. **Συνάρτηση του Rosenbrock**

$$f(\mathbf{x}) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 \tag{4.7}$$

όπου $\mathbf{x}_0 = (-1.2, 1)^T$, $\mathbf{x}^* = (1, 1)^T$, $f(\mathbf{x}^*) = 0$.

```
rosen = lambda x: scipy.optimize.rosen(x)
drosen = lambda x: scipy.optimize.rosen_der(x)

def rosen2(x):
    f = rosen(x)
    df = drosen(x)
    return f,df
```

Listing 4.6: ”Η συνάρτηση Rosenbrock σε κώδικα Python”

7. **Συνάρτηση των Freudestein και Roth**

$$\begin{aligned} f(\mathbf{x}) = &\{-13 + x_1 + [(5 - x_2)x_2 - 2]x_2\}^2 \\ &+ \{-29 + x_1 + [(1 + x_2)x_2 - 14]x_2\}^2 \end{aligned} \tag{4.8}$$

όπου $\mathbf{x}_0 = (0.5, -2)^T$, $\mathbf{x}^* = (5, 4)^T$, $f(\mathbf{x}^*) = 0$,
x'=(11.41..., -0.8968...), f(**x'**)= 48.9842...

```
f1e = lambda x: -13 + x[0] + ((5-x[1])*x[1]-2)*x[1]
f2e = lambda x: -29 + x[0] + ((x[1]+1)*x[1]-14)*x[1]

fr_func = lambda x: (square(f1e(x)) + square(f2e(x)))

def fr_func2(x):
    f = fr_func(x)
    df=np.zeros(2)
    df[0] = 2*(-13 + x[0] + ((5-x[1])*x[1]-2)*x[1]
         -29 + x[0] + ((x[1]+1)*x[1]-14)*x[1])
    df[1] = 2*(-13 + x[0] + ((5-x[1])*x[1]-2)*x[1])*
         (((5-x[1])*x[1]-2) + x[1]*(5-2*x[1])) +
           2*(-29 + x[0] + ((x[1]+1)*x[1]-14)*x[1])*
           (((x[1]+1)*x[1]-14)+x[1]*(1 + 2*x[1]))
    return f,df
```

Listing 4.7: ”Η συνάρτηση των Freudestein και Roth σε κώδικα Python”

4.2 Υλοποιήσεις υπολογισμών

4.2.1 Κλήση των βιβλιοθηκών

```
import math as m
from scipy.optimize import minimize, basinhopping
import scipy
import numpy as np
PI = np.pi

# Class for Basin-Hopping method.
class MyTakeStep(object):
    def __init__(self, stepsize=0.5):
        self.stepsize = stepsize
    def __call__(self, x):
        s = self.stepsize
        x[0] += np.random.uniform(-2.*s, 2.*s)
        x[1:] += np.random.uniform(-s, s, x[1:].shape)
        return x

# Function that calculates the square of a given x
def square(x): return x*x

# Defining two arrays with the names of the methods we'll use.
gradientMethods = ['Newton-CG', 'CG']
nogradientMethods = ['bfgs', 'Powell', 'nelder-mead']

# Defining the function that calls the Basin-Hopping method.
def basinalgo(func,x0):
    minimizer_kwargs = {"method":"L-BFGS-B", "jac":True}
    mytakestep = MyTakeStep()
    res = basinhopping(func, x0, minimizer_kwargs=minimizer_kwargs,
      niter=500, take_step=mytakestep)
    print 'Basin-Hopping',res
```

Listing 4.8: Η βασική εργαλειοθήκη

4.2.2 Κώδικες για την βελτιστοποίηση συναρτήσεων μέσω των μεθόδων Newton-CG, CG και Basin-Hopping

```
# Calling iteratively all the methods named in the array
# gradientMethods.
# gradientMethods = ['Newton-CG', 'CG']

for meth in gradientMethods:
    res = minimize(func, x0, method = meth, tol =1e-8,
        jac=True)
    print meth, res, '\n\n'

# Calling the Basin-Hopping method
basinalgo(func2, x0)
```

Listing 4.9: Κλήση των μεθόδων Newton-CG, CG και Basin-Hopping

4.2.3 Κώδικες για την βελτιστοποίηση συναρτήσεων μέσω των μεθόδων BFGS, Powell και Nelder-Mead

```
# nogradientMethods = ['bfgs', 'Powell', 'nelder-mead']
for meth in nogradientMethods:
    res = minimize(func, x0, method = meth, tol =1e-8,
      jac=None)
    print meth, res, '\n\n'
```

Listing 4.10: BFGS, Powell και Nelder-Mead

4.3 Αποτελέσματα υπολογισμών

Μέθοδος	$\mathbf{x}^*$	f($\mathbf{x}^*$)	Σημειώσεις	n iterations	f evaluations
BFGS	(1.623E-05, 6.147E+00)	8.500E-06	Desired error not necessarily achieved due to precision loss	36	332
Newton-CG	(2.342E-05, 4.269E+00)	1.925E-04	CG iterations didn't converge. Hessian is not positive definite	29	85
CG	(1.000E-04, 1.000E+00)	1.352E-01	Desired error not necessarily achieved due to precision loss	1	81
Powell	(1.228E-05, 8.140E+00)	3.350E-08	Maximum number of function evaluations has been exceeded	51	2041
Nelder-Mead	(1.177E-05, 8.495E+00)	8.594E-09	Maximum number of function evaluations has been exceeded	222	401
Basin Hopping	(1.199E-05, 8.336E+00)	1.639E-08	Abnormal Termination in LNSRCH	2	44
Αναμενόμενη τιμή	(1.098E-05, 9.106E+00)	0.000E+00	-	-	-

Πίνακας 4.1: Αποτελέσματα για την συνάρτηση Powell

Μέθοδος	$\mathbf{x}^*$	f($\mathbf{x}^*$)	Σημειώσεις	n iterations	f evaluations
BFGS	(1.000E+06, 1.993E-06)	5.402E-05	Desired error not necessarily achieved due to precision loss	15	320
Newton-CG	(5.000E+05, 4.000E-06)	2.500E+11	CG iterations didn't converge.The Hessian is not positive definite	0	3
CG	(1.000E+06, 2.000E-06)	1.972E-31	Optimization terminated successfully	15	43
Powell	(1.000E+06, 2.000E-06)	1.961E-23	Optimization terminated successfully	4	117
Nelder-Mead	(1.000E+06, 2.000E-06)	7.039E-18	Optimization terminated successfully	177	335
Basin Hopping	(1.000E+06, 2.000E-06)	2.061E-15	Success	13	25
Αναμενόμενη τιμή	(1.0E+06, 2.0E-6)	0.000E+00	-	-	-

Πίνακας 4.2: Αποτελέσματα για την συνάρτηση Brown

Μέθοδος	$\mathbf{x}^*$	f($\mathbf{x}^*$)	Σημειώσεις	n iterations	f evaluationsl
BFGS	(2.99999966, 0.49999991)	1.976E-14	Optimization terminated successfully	16	72
Newton-CG	(1. , 0.59489031)	7.140426805	Iterations didn't converge. The Hessian is not positive definite	1	2
CG	(2.93125818, 0.48407468)	8.717E-04	Desired error not necessarily achieved due to precision loss	6	75
Powell	(3.0E+00, 5.0E-01)	0.0E+00	Optimization terminated successfully	7	263
Nelder-Mead	(3.0E+00, 5.0E-01)	6.114-18	Optimization terminated successfully	84	162
Basin Hopping	(3., 0.5)	7.448E-18	Success	15	18
Αναμενόμενη τιμή	(3.0. 0.5)	0.000E+00	-	-	-

Πίνακας 4.3: Αποτελέσματα για την συνάρτηση Beale

Μέθοδος	$\mathbf{x}^*$	f($\mathbf{x}^*$)	Σημειώσεις	n iterations	f evaluations
BFGS	(3.474E+02, -8.198E-05, -7.492E-07)	5.617E-13	Desired error not necessarily achieved due to precision loss	69	807
Newton-CG	(-1. , 0. , 4.995)	2.495E+01	Optimization terminated successfully	2	3
CG	(-1., 0. ,5.001)	2.501E+01	Desired error not necessarily achieved due to precision loss	1	15
Powell	Αποτυγχάνει	Αποτυγχάνει	TypeError	-	-
Nelder-Mead	(-2.664E-15, 7.917E-03, 3.333E-04)	1.122E-05	Optimization terminated successfully	62	264
Basin Hopping	(-4.353E-05,4.252E-01, 1.4264E-05)	7.807E-06	Abnormal Termination in LNSRCH	15	84
Αναμενόμενη τιμή	(1, 0, 0)	0.000E+00	-	-	-

Πίνακας 4.4: Αποτελέσματα για την συνάρτηση Helical valey

Μέθοδος	$\mathbf{x}^*$	f($\mathbf{x}^*$)	Σημειώσεις	n iterations	f evaluations
BFGS	(0.99999552, 0.99999104)	2.005E-11	Optimization terminated successfully	34	164
Newton-CG	(1.00000002, 1.00000003)	2.820E-16	Optimization terminated successfully	88	108
CG	(-1.01703014, 1.07468158)	4.231E+00	Desired error not necessarily achieved due to precision loss	1	15
Powell	(1.0,1.0)	9.023E-28	Optimization terminated successfully	23	850
Nelder-Mead	(1.0,1.0)	1.099E-18	Optimization terminated successfully	117	219
Basin Hopping	(1.0,1.0)	1.160E-19	Success	22	28
Αναμενόμενη τιμή	(1, 1)	0.000E+00	-	-	-

Πίνακας 4.5: Αποτελέσματα για την συνάρτηση Rosenbrock

Μέθοδος	$\mathbf{x}^*$	f($\mathbf{x}^*$)	Σημειώσεις	n iterations	f evaluations
BFGS	(0.99999981, 0.99999964, 1.00000006, 1.00000014)	5.752E-13	Optimization terminated successfully	87	600
Newton-CG	(1.00000001, 1.00000001, 0.99999999, 0.99999999)	5.752E-13	Optimization terminated successfully	203	239
CG	(1.0,1.0,1.0,1.0)	1.845E-17	Optimization terminated successfully	79	168
Powell	(1.0,1.0,1.0,1.0)	3.260E-21	Optimization terminated successfully	23	1416
Nelder-Mead	(1.0,1.0,1.0,1.0)	1.526E-16	Optimization terminated successfully	387	655
Basin Hopping	(1.0,1.0,1.0,1.0)	1.197E-17	Success	23	26
Αναμενόμενη τιμή	(1, 1,1,1)	0.000E+00	-	-	-

Πίνακας 4.6: Αποτελέσματα για την συνάρτηση Wood

Μέθοδος	$\mathbf{x}^*$	f($\mathbf{x}^*$)	Σημειώσεις	n iterations	f evaluations
BFGS	(11.4127789 , -0.89680526)	48.984253	Desired error not necessarily achieved due to precision loss	10	204
Newton-CG	(11.41277905, -0.89680525)	48.984253	Optimization terminated successfully	11	14
CG	(7.94388211, -1.0549995)	54.677406	Desired error not necessarily achieved due to precision loss	4	29
Powell	(11.41277923, -0.89680524)	48.984253	Optimization terminated successfully	5	180
Nelder-Mead	(11.41277892, -0.89680526)	48.984253	Optimization terminated successfully	87	172
Basin Hopping	(11.41277894, -0.89680526)	48.984253	Optimization terminated successfully	17	18
Αναμενόμενη τιμή	(11.41..., -0.8968...)	48.9842	-	-	-

Πίνακας 4.7: Αποτελέσματα για την συνάρτηση Freudestein και Roth

Κεφάλαιο 5

Συμπεράσματα

5.1 Συγκρίσεις ανά πρόβλημα

Θα συγκρίνουμε τα αποτελέσματα όσον αφορά αν η βελτιστοποίηση ήταν επιτυχής ανά μέθοδο για κάθε πρόβλημα. Επίσης, μπορούμε να δούμε ποια μέθοδος ανά πρόβλημα είχε τις λιγότερες επαναλήψεις ως προς το x (n iterations) για να φτάσει στην επιθυμητή τιμή και έκανε και τους λιγότερους υπολογισμούς της τιμής της συνάρτησης σε κάθε νέο δοκιμαστικό x (f evaluations).

5.1.1 Το πρόβλημα της συνάρτησης Powell

Συγκρίνοντας τα αποτελέσματα του πίνακα 4.1 όλων των μεθόδων σε αυτό το πρόβλημα παρατηρούμε για αρχή πως οι μέθοδοι που έδωσαν τα χειρότερα αποτελέσματα στην ελαχιστοποίηση της συνάρτησης ήταν η Newton-CG και η CG. Η αναμενόμενη τιμή ήταν το μηδέν, και πιο κοντά σε αυτό, δηλαδή τις μικρότερες τιμές της συνάρτησης έφτασαν οι μέθοδοι Nelder-Mead, Basin Hopping, Powell και BFGS.

Όσον αφορά τις τιμές του διανύσματος x που έδιναν την ελάχιστη τιμή στην συνάρτηση, οι τρεις μέθοδοι που έφτασαν πιο κοντά στην θεωρητική (αναμενόμενη) τιμή ήταν η Nelder-Mead, η Basin Hopping και η Powell.

Όσον αφορά τον αριθμό των επαναλήψεων που χρειάστηκαν να εκτελέσουν οι τέσσερις τελευταίες μέθοδοι που αναφέραμε, τις λιγότερες επαναλήψεις (στην συνάρτηση που έβρισκε τα τοπικά ελάχιστα) εκτέλεσε η Basin Hopping. Τις λιγότερες επαναλήψεις (n itertions), πέρα από την Basin Hopping, έκανε η BFGS. Τον μικρότερο αριθμό επαναληπτικών υπολογισμών των τιμών της υπό ελαχιστοποίησης συνάρτησης έκανε πάλι η BFGS.

5.1.2 Το πρόβλημα της συνάρτησης Brown

Από τον πίνακα 4.2 βλέπουμε πως η μέθοδος κλίσης Newton-CG δεν έδωσε καλά αποτελέσματα. Πιο συγκεκριμένα, όχι μόνο δεν έδωσε καλά αποτελέσματα (δηλαδή εμφάνισε σφάλματα ακρίβειας), αλλά η μέθοδος δεν συνέκλινε διότι ο εσσιανός πίνακας δεν ήταν θετικά ορισμένος. Για αυτό τον λόγο στον τερματισμό της διαδικασίας η τιμή της συνάρτησης που πήραμε ήταν έντεκα τάξεις μεγέθους μεγαλύτερη από την αναμενόμενη (θεωρητική) τιμή. Αντίθετα, οι υπόλοιπες μέθοδοι έδωσαν αποτελέσματα κοντά στην θεωρητική τιμή, όσον αφορά τις τιμές της συνάρτησης και τα καλύτερα αποτελέσματα τα έδωσαν οι CG, Powell.

Όσον αφορά τις τιμές του διανύσματος x που έδινε την έλαχιστη τιμή στην συνάρτηση, οι μέθοδοι CG, Nelder-Mead, Basin Hopping και Powell έδωσαν τις ίδιες τιμές με ακρίβεια ακριβώς όση και η αναμενόμενη. Ενώ η BFGS ακολούθησε με μικρή διαφορά.

Όσον αφορά τον αριθμό επαναλήψεων (n iterations), εξαιρουμένης της υβριδικής Basin Hopping, από τις λοιπές μεθόδους που δεν εμφάνισαν κανένα πρόβλημα κατά την εκτέλεσή τους (BFGS, CG, Nelder-Mead, Powell), λιγότερες επαναλήψεις έκανε η Powell. Τον μικρότερο αριθμό επαναληπτικών υπολογισμών των τιμών της υπό ελαχιστοποίησης συνάρτησης έκανε η CG.

5.1.3 Το πρόβλημα της συνάρτησης Beale

Από τον πίνακα 4.3 βλέπουμε πως η μέθοδος Newton-CG δεν έδωσε καλά αποτελέσματα πάλι. Όπως και πριν η μέθοδος δεν συνέκλινε με αποτέλεσμα να δώσει στο τέλος μη μηδενικό αποτέλεσμα. Αντίθετα οι υπόλοιπες μέθοδοι έδωσαν καλά αποτελέσματα στις τιμές της συνάρτησης, με πρωταθλητές να είναι αυτή την φορά οι μέθοδοι Nelder-Mead και Basin Hopping. Αξίζει να σημειώσουμε πως η μέθοδος CG λόγω σφαλμάτων ακρίβειας δεν πέτυχε να τερματίσει με το επιθυμητό σφάλμα που είχαμε δώσει ως κριτήριο τερματισμού (1E-8).

Όσον αφορά τις τιμές του διανύσματος x που έδινε την ελάχιστη τιμή στην συνάρτηση, πιο κοντά στην αναμενόμεη τιμή ήταν οι μέθοδοι Basin Hopping, Nelder-Mead και Powell.

Σχετικά με τον αριθμό επαναλήψεων (n iterations), εξαιρουμένης της Basin Hopping, από τις μεθόδους που δεν εμφάνισαν κανένα πρόβλημα τις λιγότερες επαναλήψεις έκανε η Powell. Τον μικρότερο αριθμό επαναληπτικών υπολογισμών των τιμών της υπό ελαχιστοποίησης συνάρτησης έκανε η BFGS.

5.1.4 Το πρόβλημα της συνάρτησης Helical Valey

Από τον πίνακα 4.4 βλέπουμε πως οι μέθοδοι Newton-CG και CG δεν έδωσαν καλά αποτελέσματα. Επίσης, η Powell εμφάνισε σφάλμα TypeError και δεν κατάφερε να τερματίσει επιτυχώς. Από τις λοιπές τρεις μεθόδους τα καλύτερα αποτελέσματα στην τιμή της συνάρτησης έδωσαν με σειρά εγγύτητας στην θεωρητική τιμή η BFGS με διαφορά από τις άλλες, και έπειτα οι Basin Hopping και η Nelder-Mead.

Όσον αφορά τις τιμές του διανύσματος x που έδινε την ελάχιστη τιμή στην συνάρτηση, καμία από τις μεθόδους δεν πλησίασε την αναμενόμενη τιμή και για τις τρεις τιμές του διανύσματος x, ενώ η BFGS πλησίασε πιο κοντά από όλες στις δύο από τις τρεις.

Σχετικά με τον αριθμό επαναλήψεων (n iterations), εξαιρουμένης της Basin Hopping, από τις μεθόδους που δεν εμφάνισαν κανένα πρόβλημα τις λιγότερες επαναλήψεις έκανε η Nelder-Mead. Τον μικρότερο αριθμό επαναληπτικών υπολογισμών των τιμών της υπό ελαχιστοποίησης συνάρτησης έκανε η Nelder-Mead.

5.1.5 Το πρόβλημα της συνάρτησης Rosenbrock

Από τον πίνακα 4.5 βλέπουμε πως μόνο η CG απέκλινε αρκετά από την μηδενική τιμή του ελάχιστου της συνάρτησης. Από τις υπόλοιπες συναρτήσεις οι τρεις καλύτερες ήταν με σειρά προτεραιότητας η Powell, η Basin Hopping και η Newton-CG.

Όσον αφορά τις τιμές του διανύσματος x που έδινε την ελάχιστη τιμή στην συνάρτηση, πέρα από την CG που εμφάνισε σφάλμα ακρίβειας όλες έφτασαν την αναμενόμενη τιμή. Πιο συγκεκριμένα, οι Powell, Nelder-Mead και Basin Hopping είχαν ακριβώς τιμή ίση με την θεωρητική.

Σχετικά με τον αριθμό επαναλήψεων (n iterations), εξαιρουμένης της Basin Hopping, από τις μεθόδους που δεν εμφάνισαν κανένα πρόβλημα τις λιγότερες επαναλήψεις έκανε η Powell. Τον μικρότερο αριθμό επαναληπτικών υπολογισμών των τιμών της υπό ελαχιστοποίησης συνάρτησης έκανε η Newton-CG.

5.1.6 Το πρόβλημα της συνάρτησης Wood

Από τον πίνακα 4.6 βλέπουμε πως όλες οι μέθοδοι έλυσαν το συγκεκριμένο πρόβλημα βελτιστοποίησης επιτυχώς. Μεγαλύτερη ακρίβεια στον μηδενισμό της τιμής της συνάρτησης είχαν με σειρά προτεραιότητας οι Powell, Basin Hopping και CG.

Όσον αφορά τις τιμές του διανύσματος x που έδινε την ελάχιστη τιμή στην συνάρτηση, όλες έφτασαν πολύ κοντά στο αναμενόμενο διάνυσμα που ελαχιστοποιούσε την συνάρτηση. Πιο συγκεκριμένα, οι μέθοδοι CG, Powell, Nelder-Mead και Basin Hopping συνέκλιναν στην ακριβή τιμή.

Σχετικά με τον αριθμό επαναλήψεων (n iterations), εξαιρουμένης της Basin Hopping, τις λιγότερες επαναλήψεις έκανε η Powell. Τον μικρότερο αριθμό επαναληπτικών υπολογισμών των τιμών της υπό ελαχιστοποίησης συνάρτησης έκανε η CG.

5.1.7 Το πρόβλημα της συνάρτησης Freudenstein

Από τον πίνακα 4.7 βλέπουμε πως όλες οι μέθοδοι εκτός από την CG έλυσαν το πρόβλημα επιτυχώς. Η CG εμφάνισε σφάλματα ακρίβειας και για αυτό τον λόγο δεν προσέγγισε την θεωρητική τιμή τόσο πολύ όσο οι υπόλοιπες μέθοδοι. Οι υπόλοιπες μέθοδοι αντίθετα ισοβάθμησαν μιας και όλες μεγιστοποίησαν την συνάρτηση με όση ακρίβεια αναμέναμε.

Όσον αφορά τις τιμές του διανύσματος x που έδινε την ελάχιστη τιμή στην συνάρτηση, πέρα από την CG, όλες οι μέθοδοι έδωσαν την ίδια τιμή με ακρίβεια πέντε δεκαδικών ψηφίων, η οποία ήταν και η αναμενόμενη.

Σχετικά με τον αριθμό επαναλήψεων (n iterations), εξαιρουμένης της Basin Hopping, τις λιγότερες επαναλήψεις έκανε η CG και ακολούθησε η Powell. Τον μικρότερο αριθμό επαναληπτικών υπολογισμών των τιμών της υπό ελαχιστοποίησης συνάρτησης έκανε η Newton-CG.

5.2 Τελικά συμπεράσματα

Οι παραπάνω παρατηρήσεις φαίνονται καλύτερα στον ακόλουθο πίνακα:

Πρόβλημα	**f(x^*)**	x^*	**n iter**	**f eval**
Powell	N-M	N-M	BFGS	BFGS
Brown	CG	CG, N-M, BH, Powell	Powell	CG
Beale	N-M	BH	Powell	BFGS
Helical Valey	BFGS	BFGS	N-M	N-M
Rosenbrock	Powell	Powell,N-M,BH	Powell	Newton-CG
Wood	Powell	CG,Powell,N-M,BH	Powell	CG
Freudenstein	Όλες πλην CG	Όλες	CG	Newton-CG

Πίνακας 5.1: Αποτελέσματα σύγκρισης απόδοσης μεθόδων ανά πρόβλημα

Στον πίνακα 5.2 στην πρωτη στήλη έχουμε βάλει το όνομα του κάθε προβλήματος που εξετάσαμε. Η δεύτερη στήλη, κάτω από την τιμή της συνάρτησης f στον ελαχιστοποιητή της, x^*, έχει τα ονόματα των μεθόδων που έδωσαν αποτέλεσμα πιο κοντά στην αναμενόμενη (θεωρητική τιμή). Δηλαδή καταγράψαμε την μέθοδο που δίνει την μεγαλύτερη ακρίβεια στο συγκεκριμένο πρόβλημα. Στην τρίτη στήλη σημειώσαμε την μέθοδο (ή τις μεθόδους) που υπολόγισε (υπολόγισαν) τον ελαχιστοποιητή x^* με την καλύτερη ακρίβεια, συγκριτικά με την αναμενόμενη τιμή. Τέλος, η προτελευταία και τελευταία στήλη

αναφέρουν: την μέθοδο που χρειάστηκε τις λιγότερες επαναλήψεις (n iter) και αυτήν που έκανε τους λιγότερους υπολογισμούς της συναρτησιακής τιμής (f eval), αντίστοχα.

Από τα αποτελέσματα του πίνακα λοιπόν συμπεραίνουμε πως καλό θα ήταν να αποφεύγεται η χρήση μεθόδων που χρησιμοποιούν τον ιακωβιανό πίνακα, όπως την CG και την Newton-CG γιατί σhttps://www.overleaf.com/read/jnfghdqhxrsbε μερικά προβλήματα δεν συγκλίνουν ή εμφανίζουν σφάλματα ακρίβειας.

Αντίθετα, είναι προτιμότερο η χρήση μεθόδων που δεν χρησιμοποιούν πληροφορίες παραγώγου των συναρτήσεών μας, όπως οι Powell, BFGS και Nelder-Mead. Όσο για την μέθοδο Basin Hopping, αν και χρησιμοποιεί πληροφορίες παραγώγου των συναρτήσεών μας, επειδή είναι υβριδική, δηλαδή χρησιμοποιεί μια από τις μεθόδους που αναφέραμε για την εύρεση των τοπικών ελαχίστων και έπειτα αρχίζει η στοχαστική της διαδικασία, χρειάζεται περισσότερη μελέτη.

Αναφορές

[1] E. K. Chong and S. H. Zak, *An introduction to optimization*, vol. 76, John Wiley & Sons, 2013.

[2] H. P. Gavin, *The nelder-mead algorithm in two dimensions*, CEE 201L. Duke U, (2013).

[3] I. Hashmi and A. Shehu, *HopDock: a probabilistic search algorithm for decoy sampling in protein-protein docking*, Proteome Science, 11 (2013), p. S6.

[4] M. Iwamatsu and Y. Okabe, *Basin hopping with occasional jumping*, Chemical Physics Letters, 399 (2004), pp. 396–400.

[5] C. T. Kelley, *Iterative methods for optimization*, vol. 18, Siam, 1999.

[6] J. J. Moré, B. S. Garbow, and K. E. Hillstrom, *Testing unconstrained optimization software*, ACM Trans. Math. Softw., 7 (1981), pp. 17–41.

[7] J. Nocedal and S. Wright, *Numerical optimization*, Springer Science & Business Media, 2006.

[8] D. J. Wales and J. P. K. Doye, *Global optimization by basin-hopping and the lowest energy structures of lennard-jones clusters containing up to 110 atoms*, The Journal of Physical Chemistry A, 101 (1997), pp. 5111–5116.

[9] L. Wille, *Minimum-energy configurations of atomic clusters: new results obtained by simulated annealing*, Chemical Physics Letters, 133 (1987), pp. 405–410.