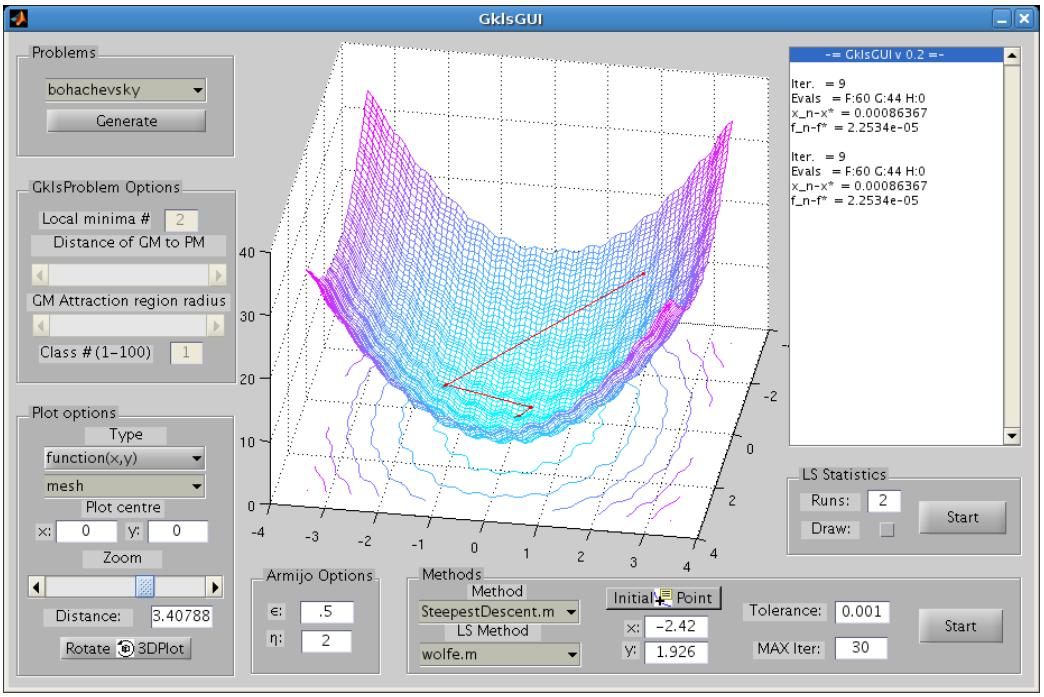
Aggiornamenti e nuove funzionalità del software GklsGUI

Enrico Boldrini



GklsGUI

Anno Accademico 2008-2009
Metodi Numerici per l'Ottimizzazione
Prof. Luigi Brugnano

Autore: Enrico Boldrini boldra@gmail.com
Titolo: Aggiornamenti e nuove funzionalità del software GklsGUI,
Anno Accademico 2008-2009, Università di Firenze,
Metodi Numerici per l'Ottimizzazione, Prof. Luigi Brugnano.

Indice

Capitolo 1

Introduzione

Oggetto di questo documento è la descrizione degli aggiornamenti e delle nuove funzionalità implementate in *GklsGUI*.
GklsGUI, sviluppato inizialmente da Valerio Angelini[1], è un software grafico interattivo per *MATLAB*, creato per testare e valutare algoritmi di ricerca di punti di minimo. In particolare dalla sua prima versione è possibile testare gli algoritmi su problemi di tipo *GKLS*, generati direttamente dalla libreria sviluppata dagli autori in linguaggio C[4] e sul problema di Rosenbrock.
Le nuove funzionalità includono:

- possibilità di scegliere il metodo per la minimizzazione unidimensionale. Sono stati implementati 8 metodi.
- implementazione di 12 nuovi problemi su cui poter testare i metodi.
- migliorie all'interfaccia, tra cui zoom e centratura del grafico.
- possibilità di generare statistiche su misure degli algoritmi, da esportare in un foglio di calcolo o in un documento LaTeX

Gli aggiornamenti comprendono modifiche per la compatibilità con *MATLAB 7.7* e *MATLAB 7.8* e compatibilità con compilatore C ANSI standard.
La prima sezione, è dedicata all'interfaccia grafica e funge da guida per il programma. Sono poi presentati gli algoritmi unidimensionali e i problemi che sono stati introdotti a partire da questa versione, corredati da alcune statistiche ottenute direttamente dal software.

Capitolo 2

Guida all'interfaccia grafica

Per avviare il programma è sufficiente lanciare MATLAB dalla cartella di GklsGUI ed eseguire lo script `run`. Se si vuole utilizzare la modalità compatibile con *MATLAB 7.7* occorre invece eseguire lo script `run_7_7`.
Il programma si presenterà con l'interfaccia grafica visibile in figura 2.1. Possiamo suddividere l'interfaccia nelle seguenti aree (con riferimento alla figura):

1. generazione del problema
2. opzioni del problema Gkls
3. opzioni del grafico
4. grafico
5. opzioni del metodo di Armijo
6. parametri dei metodi di risoluzione
7. pannello delle statistiche LS
8. pannello dei risultati

2.1 Generazione del problema

Si può scegliere il problema da analizzare scegliendolo dal menù a tendina; premere il pulsante `Generate` per generarlo.
Per aggiungere un nuovo problema alla lista è sufficiente creare una cartella dal nome `@*PROBLEM*Problem` nella root directory, dove `*PROBLEM*` è il nome del nuovo problema ed implementare le funzioni MATLAB che calcolano il valore, il gradiente e l'Hessiana del nuovo problema.

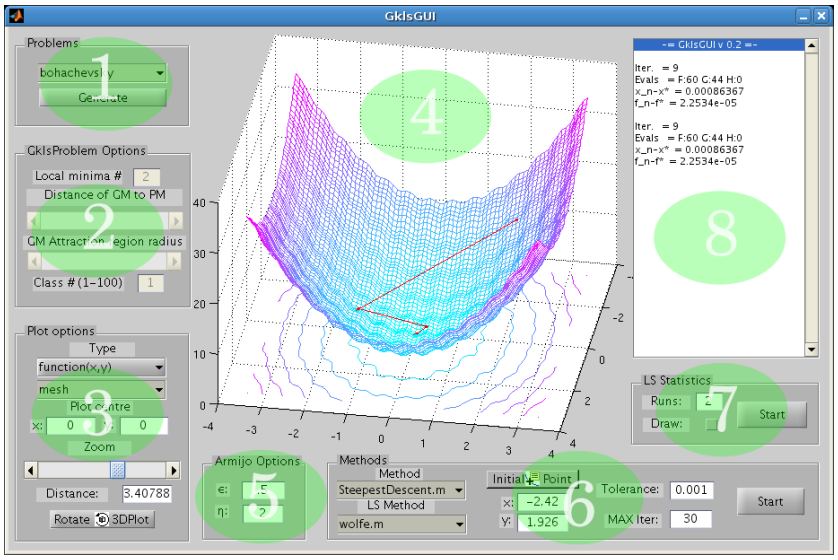


Figura 2.1: Interfaccia grafica

2.2 Opzioni del problema Gkls

Questo pannello contiene opzioni relative soltanto al problema Gkls, tra cui:

- numero di minimi locali
- distanza del minimo globale dai minimi locali
- raggio d'attrazione del minimo globale
- classe del problema Gkls

Il pannello diventa abilitato quando il problema Gkls è selezionato.

2.3 Opzioni del grafico

Fra le opzioni del grafico ci sono:

- oggetto del grafico: funzione, derivata parziale in x, derivata parziale in y, derivata xy, derivata xx, derivata yy (si veda figura 2.2)
- tipo di grafico: mesh, contour, contour 3d, surface
- centro del grafico

- zoom/distanza: indica la distanza (-∞) dal centro del grafico visualizzata
- rotazione: premere il pulsante per attivare la modalità rotazione 3d. Successivamente usare il mouse direttamente sul grafico

2.4 Grafico

Il grafico permette la visualizzazione di una parte d'interesse del problema e delle *traiettorie* effettuate dai metodi nella ricerca del minimo (ottenute unendo in una spezzata i punti consecutivi di ciascuna ricerca), permettendo così di effettuare un confronto visivo dei metodi utilizzati.

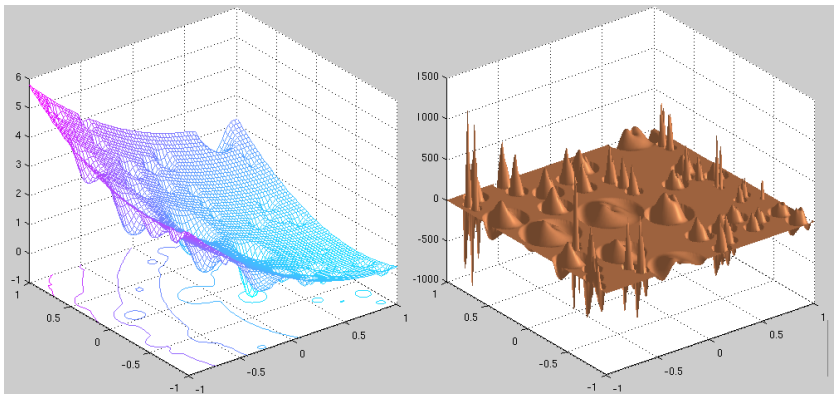


Figura 2.2: Visualizzazione di un problema Gkls con 50 minimi e della sua derivata seconda in x.

2.5 Opzioni del metodo di Armijo

Tramite questo pannello è possibile configurare globalmente i parametri del metodo line search di Armijo (ϵ e η).

2.6 Parametri dei metodi di risoluzione

Questo importante pannello permette di scegliere il metodo multidimensionale e line search da utilizzare nella ricerca e le relative opzioni. Sono presenti i seguenti componenti:

- tendina dei metodi multidimensionali: per scegliere il metodo multidimensionale da utilizzare

- tendina dei metodi unidimensionali: questa tendina è attiva nel caso di metodi multidimensionali che contemplano la minimizzazione unidimensionale (line search) e permette di scegliere il metodo da utilizzare. Sono mostrati solo i metodi line search compatibili con il metodo multidimensionale selezionato (ad esempio non sono mostrati i metodi LS che utilizzano l'Hessiana, es. Newton, se il metodo multidimensionale non sfrutta questa informazione, es. metodo del gradiente).

- punto iniziale: è posssibile inserire manualmente le coordinate del punto d'inizio della ricerca oppure premere il pulsante e selezionare dal grafico utilizzando il mouse.

- tolleranza: questo campo serve ad indicare la tolleranza da utilizzare, il valore immesso sarà utilizzato solo da quei metodi che hanno un criterio d'arresto basato sulla tolleranza.

- massimo numero di iterazioni: questo campo serve ad indicare il numero massimo d'iterazioni consentite ed è utilizzato sia dai metodi multidimensionali che da quelli unidimensionali.

- pulsante d'avvio: per avviare la ricerca, una volta che sono stati settati gli altri parametri. I punti individuati dall'algoritmo saranno mostrati sul grafico una volta terminata l'esecuzione, mentre i risultati saranno mostrati nel pannello dei risultati.

2.7 Pannello delle statistiche LS

Questo pannello serve per generare le statistiche sui metodi line search. È possibile indicare il numero di run su cui fare le statistiche e la possibilità di visualizzare graficamente i run. Al termine della generazione saranno visualizzati i risultati sul pannello risultati e generati 2 file da visualizzare con un foglio di calcolo oppure LaTeX.
Per una descrizione approfondita delle statistiche effettuate si rimanda al capitolo 4.

2.8 Pannello dei risultati

Il pannello dei risultati mostra informazioni sull'ultima ricerca effettuata utili all'analisi del metodo, come:

- il numero di iterazioni effettuate

- il numero di valutazioni della funzione, del gradiente e dell'Hessiana
- la distanza dal punto di minimo trovato e da quello effettivo
- la distanza dal valore di minimo trovato e dal minimo effettivo

Capitolo 3

Metodi unidimensionali per l'ottimizzazione

I metodi di minimizzazione unidimensionale (o line search) sono usati da un'ampia classe di metodi iterativi di ricerca del minimo a più dimensioni per trovare il minimo lungo la direzione ammissibile individuata dall'algoritmo multidimensionale.
Sia $\underline{x}_n$ l'approssimazione corrente, $\underline{d}_k \in \Re^n$ direzione di ricerca ammissibile per $\underline{x}_k$; si risolve il problema $min_{\alpha \geq 0} f(\underline{x}_k + \alpha \underline{d}_k) = \alpha_k$ individuando così la nuova approssimazione $\underline{x}_{k+1} = \underline{x}_k + \alpha_k \underline{d}_k$

Gli algoritmi che sono stati implementati si possono suddividere in 3 gruppi:

- Metodi per funzioni unimodali
  - Fibonacci
  - Sezione aurea
- Metodi curve fitting
  - Newton (con criteri d'arresto: armijo, goldstein, wolfe, tolAlpha, tolG)
  - Secante (con criteri d'arresto: armijo, goldstein, wolfe, tolAlpha, tolG)
  - Fit cubico (con criteri d'arresto: armijo, goldstein, wolfe, tolAlpha, tolG)
  - Fit quadratico (con criteri d'arresto: armijo, goldstein, wolfe, tolAlpha, tolG)

- Metodi inesatti
  - Armijo
  - Goldstein
  - Wolfe

Di seguito saranno esaminati i metodi in dettaglio. Una sezione è poi dedicata alla tecnica del Three Point Pattern usata per ottenere la convergenza dei metodi di curve fit.

3.1 Metodi per funzioni unimodali

L'unico requisito richiesto dai metodi di questa sezione è che la funzione da minimizzare sia unimodale, cioè abbia un singolo punto di minimo relativo.

3.1.1 Fibonacci

Il metodo di Fibonacci è un metodo molto popolare ed analiticamente elegante. Il metodo detemina il minimo di una funzione f unimodale all'interno di un intervallo c_1, c_2. Il minimo di f viene determinato (almeno approssimativamente) misurando il valore di f in corrispondenza di un determinato numero N di punti.
Il metodo di Fibonacci indica come fissare gli N punti allo scopo di determinare la più piccola regione d'incertezza in cui cade il minimo. Così, dopo aver selezionato gli N punti $x_1, x_2, \ldots, x_N$ tali che:

$$a \leq x_1 < x_2 \ldots < x_{N-1} < x_N \leq b$$

la regione d'incertezza sarà l'intervallo $[x_{k-1}, x_{k+1}]$, dove x_k è il minore degli N punti, in più si definisce $x_0 = a, x_{N+1} = b$ per consistenza. Il minimo giacerà in questo intervallo.
Sia d_k l'ampiezza dell'intervallo di confidenza al k-esimo passo e $d_1 = b - a$. Consideriamo il caso $N = 2$. Valuto la funzione nei punti x_1 ed x_2 disponendoli molto vicini al punto medio e tali che $x_1 < x_2$. Sono possibili due casi:

1. $f(x_1) > f(x_2) \rightarrow$ l'intervallo di confidenza è $[x_1, b]$, inoltre $d_2 = \frac{b-a}{2}$.
2. altrimenti l'intervallo di confidenza è $[a, x_2]$ e si ha sempre $d_2 = \frac{b-a}{2}$ dato che sono di poco sfasati.

Nel caso $N = 2$ ho quindi $x_1 = a + l_1$ ed $x_2 = b - l_1$.
Consideriamo adesso il caso $N = 3$. Dopo aver disposto i primi due punti x_1 ed x_2 in maniera simmetrica, ricordando che d_1 è l'intervallo di confidenza iniziale, avrò i due casi:

1. se $f(x_1) > f(x_2) \rightarrow [x_1, b]$ sarà il nuovo intervallo di confidenza
2. altrimenti sarà $[a, x_2]$

In entrambi casi si avrà il nuovo intervallo di confidenza $d_2 = d_1 - l_1$, (anche se per il momento non conosciamo l_1). Supponiamo che $d_2 = x_2 - a$, si deve fare in modo di trovarsi nella stessa condizione di prima ($N = 1$), quindi x_1 deve distare da x_2 di l_1. Si potrà in questo modo fare la valutazione nel punto x_3 quasi coincidente con x_1.
Nel caso generale si avrà per il nuovo intervallo di confidenza:

$$d_{i+1} = d_i - l_i = l_i + l_{i+1}$$

e quindi:

$$l_{i+1} = d_{i+1} - l_i = d_i - 2l_i$$

Quindi: $d_{i+2} = d_{i+1} - l_{i+1} = l_i + l_{i+1} - l_{i+1} = l_i$
Dunque abbiamo: $d_{i+1} = d_i - l_i = d_i - d_{i+2}$ e possiamo scrivere la relazione:

$$\begin{cases} d_i = d_{i+1} + d_{i+2} & , i = 1, \ldots, N-1 \\ d_1 = b - a \end{cases}$$

Si deve però anche avere: $l_N = 0 \rightarrow d_{N+1} = d_N$. Inoltre, da $d_{i+2} = l_i \rightarrow l_1 = d_3$, quindi se conosciamo la successione dei d_i sappiamo dove fissare i punti. Definiamo la successione $F_i = \frac{d_{N+1-i}}{d_{N+1}}$. Ovviamente $F_0 = 1$ e $F_1 = \frac{d_N}{d_{N+1}} = 1$.
Quindi $F_i = F_{i-1} + F_{i-2}$, per $i = 2, \ldots, N$.
Siamo arrivati alla successione di Fibonacci:

$$\begin{cases} F_i = F_{i-1} + F_{i-2} \\ F0 = F1 = 1 \end{cases}$$

Vediamo quanto vale $\frac{d_k}{d_1}$. Vale:

$$F_i = \frac{d_{N+1-i}}{d_{N+1}} \rightarrow d_{N+1-i} = d_{N+1} F_i$$

e vogliamo che $k = N + 1 - i \rightarrow i = N + 1 - k$.
Quindi:

$$\frac{d_k}{d_1} = \frac{d_{N+1} F_{N+1-k}}{d_{N+1} F_N} = \frac{F_{N+1-k}}{F_N}$$

Dunque $d_k = d_1 \frac{F_{N+1-k}}{F_N}$. Poiché vale:

$$\begin{cases} d_{i+1} = d_i - l_i \\ l_{i+1} = d_{i+1} - l_i \end{cases} \quad i = 1, \ldots, N-1.$$

con $d_1 = b - a$ e $l_1 = d_3$. A questo punto ci basta conoscere d_3 per calcolare gli altri termini della successione, in maniera iterativa:

$$l_1 = d_3 = d_1 \frac{F_{N-2}}{F_N} = (b-a)\frac{F_{N-2}}{F_N}$$

Supponiamo a questo punto di aver fissato x_1 ed x_2 (mediante la conoscenza di l_1) e ci accingiamo a fissare x_3. Se $f(x_2) > f(x_1) \rightarrow [a, b] \equiv [a, x_2]$. E il punto successivo sarà:

$$\begin{cases} x_3 = a + b - x_1 \\ b = x_2 \end{cases}$$

Infatti $b - x_i = x_2 - x_1 = l_{i+1} \rightarrow x_3 = a + l_{i+1}$ come deve essere.

3.1.2 Sezione Aurea

Con il metodo Fibonacci è fissato il numero N di iterazioni del metodo. Il metodo della sezione aurea nasce con l'intento di modificare il metodo di Fibonacci in un metodo iterativo puro. Dobbiamo capire cosa succede a $\frac{F_{N-2}}{F_N}$ quando $N \rightarrow +\infty$.
Abbiamo quindi:

$$\begin{cases} F_{i+2} = F_{i+1} + F_i \\ F_0 = F_1 = 1 \end{cases}$$

Questa è un'equazione alle differenze lineare del secondo ordine, a coefficienti costanti. Cerchiamo soluzioni del tipo: $F_i = z^i, z \in C \rightarrow z^{i+2} - z^{i+1} - z^i = 0$ e il polinomio caratteristico associato è: $p(z) = z^2 - z - 1 = 0 \rightarrow z_{1,2} = \frac{1 \pm \sqrt{5}}{2}$ e $\frac{1+\sqrt{5}}{2}$ è il rapporto aureo. Le soluzioni sono linearmente indipendenti. Quindi: $F_i = c_1 z_1^i + c_2 z_2^i$ e ponendo:

$$\begin{aligned} F0 &= c_1 + c_2 = 1 \\ F_1 &= c_1 z_1 + c_2 z_2 = 1 \end{aligned}$$

si ottiene:

$$\begin{pmatrix} 1 & 1 \\ z_1 & z_2 \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

Abbiamo che $z_1 = 1.618\ldots > |z_2| = 0.618\ldots$, dunque $F_i = z_1^i(c_1 + c_2(\frac{z_2}{z_1})^1)$, ma se $i >> 1$ si ha che $f_i \approx z_i^i c_1$.

Quindi $\frac{F_{N-2}}{F_N} \approx \frac{c_1 z_1^{N-2}}{c_1 z_1^N} = \frac{1}{z_1^2} = (\frac{\sqrt{5}-1}{2})^2 \approx 0.618^2$.
Quindi possiamo porre $l_1 = (b-a)\frac{(\sqrt{5}-1)^2}{4}$ ed avremo un metodo iterativo nella forma pura. Nell'implementazine sarà anche necessario predisporre un criterio d'arresto del tipo `while(b-a)>tol`: in tal modo la ricerca si arresterà quando l'interallo di confidenza avrà raggiunto l'accuratezza desiderata. Il metodo della sezione aurea è un metodo a convergenza lineare e la costante asintotica dell'errore è il reciproco della sezione aurea.

3.2 Metodi curve fit

In una larga quantità di casi si può assumere che la funzione da minimizzare sia abbastanza regolare: si possono quindi usare tecniche che sfruttano questa regolarità. È il caso delle tecniche di curve fit che consistono solitamente nell'interpolare con una curva la funzione da minimizzare, sulla base dei valori misurati precedentemente e in taluni casi anche del valore delle derivate. La terminazione della ricerca è determinata dal soddisfacimento di un criterio d'arresto. Nella prossima sezione saranno esaminati criteri d'arresto molto usati (Armijo, Goldstein e Wolfe). Sono stati usati inoltre altri due criteri, il primo controlla la grandezza dell'ultimo alpha trovato e il secondo è un controllo sulla norma del gradiente.

3.2.1 Newton

Supponiamo di dover minimizzare una funzione f di una singola variabile x e seupponiamo che in corrispondenza del punto x_k sia possibile valutare $f(x_k), f'(x_k), f''(x_k)$. Allora è possibile costruire (mediante sviluppo in serie di Taylor) una funzione quadratica q che coincide in x_k con f fino alla derivata seconda, ovvero:

$$q(x) = f(x_k) + f'(x_k)(x - x_k) + 1/2 f''(x_k)(x - x_k)^2$$

Si può calcolare l'approsimazione x_{k+1} del punto di minimo di f cercando il punto dove la derivata di q si annulla. Dunque imponendo

$$0 = q'(x_{k+1}) = f'(x_k) + f''(x_k)(x_{k+1} - x_k)$$

si trova

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$$

Questo procedimento può quindi essere ripetuto al punto x_{k+1} e così via.
Il metodo di Newton può essere visto come un metodo per trovare le soluzioni

di equazioni della forma:

$$g(x) = 0$$

Applicato alla minimizzazione poniamo $g(x) \equiv f'(x)$ ottenendo la forma:

$$x_{k+1} = x_k - \frac{g(x_k)}{g'(x_k)}$$

Questo metodo converge con un ordine di almeno 2 per punti sufficientemente vicini alla soluzione.

3.2.2 Secante

Il metodo di Newton si basa sull'interpolazione della funzione da minimizzare con una funzione quadratica sulla base dell'informazione ricavata su un unico punto; usando più punti è richiesta minore informazione su ciascuno. Quindi, usando $f(x_k), f'(x_k), f'(x_{k-1})$ è possibile interpolare con una quadratica che ha gli stessi valori:

$$q(x) = f(x_k) + f'(x_k)(x - x_k) + \frac{f'(x_{k-1} - f'(x_k))}{x_{k-1} - x_k} \frac{(x - x_k)^2}{2}$$

L'approssimazione x_{k+1} può quindi essere determinata cercando il punto dove la deriata di q si annulla, trovando:

$$x_{k+1} = x_k - f'(x_k) \frac{x_{k-1} - x_k}{f'(x_{k-1}) - f'(x_k)}$$

Il nuovo metodo può anche essere considerato un'approssimazione del metodo di Newton dove la derivata seconda è rimpiazzata dalla differenza di due derivate prime. Questo metodo può essere visto a sua volta come un metodo per risolvere $f'(x) \equiv g(x) = 0$ e prendere la forma seguente:

$$x_{k+1} = x_k - g(x_k) \frac{x_k - x_{k-1}}{g(x_k - g(x_{k-1}))}$$

L'ordine di convergenza del metodo della secante è la sezione aurea, $p = \frac{1+\sqrt{5}}{2} \approx 1.618$.
È stata implementata anche una versione con TPP per facilitare la convergenza.

3.2.3 Fit cubico

Dati i punti x_{k-1} e x_k ed i valori $f(x_{k-1})$, $f'(x_{x-1})$, $f(x_k)$, $f'(x_k)$, è possibile trovare un'equazione cubica che interpoli la funzione nei punti dati. L'approssimazione successiva x_{k+1} può quindi essere determinata come il punto di minimo relativo della cubica. Questo porta al metodo del fit cubico:

$$x_{x+1} = x_k - (x_k - x_{k-1})\frac{f'(x_k) + u_2 - u_1}{f'(x_k) - f'(x_{k-1}) + 2u_2}$$

dove

$$u_1 = f'(x_{k-1}) + f'(x_k) - 3\frac{f(x_{k-1}) - f(x_k)}{x_{k-1} - x_k}$$

$$u_2 = \sqrt{u_1^2 - f'(x_{k-1}f')x_k}$$

Si può mostrare che l'ordine di convergenza del fit cubico è 2.
Per garantire la convergenza questo metodo è stato implementato in abbinamento con il TPP.

3.2.4 Fit quadratico

Si tratta di un metodo molto usato, consiste nell'interpolare con una quadratica utilizzando tre punti. Questo metodo ha il vantaggio di non richiedere informazioni sulla derivata. Dati x_1, x_2, x_3 e i corrispondenti valori $f(x_1) = f_1$,$f(x_2) = f_2$, $f(x_3) = f_3$ si costruisce la quadratica che passa da questi 3 punti:

$$q(x) = \sum_{i=1}^{3} f_i \frac{\prod_{j \neq i}(x - x_j)}{\prod_{j \neq i}(x_i - x_j)}$$

e si determina il nuovo punto x_4 come punto dove la derivata di q si annulla. Quindi abbiamo:

$$x_4 = \frac{1}{2}\frac{b_{23}f_1 + b_{31}f_2 + b_{12}f_3}{a_{23}f_1 + a_{31}f_2 + a_{12}f_3}$$

dove $a_{ij} = x_i - x_j, b_{ij} = x_i^2 - x_j^2$
L'ordine di convergenza di questo metodo è 1.3.
Per garantire la convergenza questo metodo è stato implementato in abbinamento con il TPP.

3.2.5 Three Points Pattern

I metodi di curve fit appena descritti hanno il difetto, se applicati come sopra esposto, di non garantire niente sulla convergenza se non ci si trova in

un intorno sufficientemente vicino alla soluzione: sarà anzi probabile che la procedura diverga o vaghi senza convergere.
È importante dunque impiegare tecniche che aumentino l'efficacia dei metodi descritti favorendone la convergenza, il Three Points Pattern (TPP) è una di queste. Assumiamo che la funzione f da minimizzare sia strettamente unimodale e abbia almeno derivate parziali seconde continue. Iniziamo la procedura di ricerca cercando lungo la direzione ammissibile finché non si trovino 3 punti x_1, x_2, x_3 con $x_1 < x_2 < x_3$ tali che $f(x_1) \geq f(x_2) \leq f(x_3)$. In altre parole, il valore del punto di mezzo è minore di quelli esterni. Questa sequenza di valori può essere determinata in vari modi [5]. La ragione principale per usare una sequenza di questo tipo è che il minimo di una quadratica che passi per i 3 punti avrà un minimo nell'intervallo $[x_1, x_3]$. Vediamo in dettaglio come si può applicare il TPP al fit quadratico.
Il punto x_4 è calcolato tramite fit quadratico e $f(x_4)$ viene valutato. Assumendo $x_2 < x_4 < x_3$ e ricordando l'unimodalità di f si hanno solo 2 possibilità:

1. $f(x_4) \leq f(x_2)$
2. $f(x_2) < f(x_4) \leq f(x_3)$

In entrambi i casi un nuovo TPP, $\overline{x}_1, \overline{x}_2, \overline{x}_3$, comprendente x_4 e 2 dei punti precedenti, può essere determinato: Nel caso (1) sarà:

$$(\overline{x}_1, \overline{x}_2, \overline{x}_3) = (x_2, x_4, x_3)$$

mentre nel caso (2) sarà:

$$(\overline{x}_1, \overline{x}_2, \overline{x}_3) = (x_1, x_2, x_4)$$

Il nuovo pattern sarà usato per trovare il nuovo fit quadratico e così via. Questa procedura porta a convergenza globale e può essere adattata per funzionare anche con gli altri metodi. In particolare i metodi del fit quadratico, cubico, e delle secanti sono stati implementati usando la tecnica del TPP.

3.3 Metodi inesatti

Nella pratica non si riuscirà nella maggior parte dei casi a trovare il punto di minimo esatto. Spesso si preferisce sacrificare la precisione del metodo line search per conservare un tempo globale basso. Generalmente l'inesattezza nasce dal terminare prima della convergenza la procedura iterativa di ricerca di uno dei metodi descritti. Ci sono diversi criteri d'arresto, tra cui il criterio di Armijo, Goldstein e e.
Da questi criteri nasce l'implementazione di corrispondenti semplici metodi di ricerca.

3.3.1 Armijo

Il criterio di Armijo cerca di garantire che il valore di α selezionato non sia né troppo grande né troppo piccolo. Definiamo la funzione:

$$\phi(\alpha) = f(x_k + \alpha d_k)$$

Il criterio di Armijo è implementato tenendo conto della funzione $\phi(0) + \epsilon\phi'(0)\alpha$ per ϵ fissato, $0 < \epsilon < 1$. Un valore di α è considerato essere non troppo grande se vale:

$$\phi(\alpha) \leq \phi(0) + \epsilon\phi'(0)\alpha$$

Per assicurarsi che α non sia troppo piccolo, un valore $\eta > 1$ è selezionato e α sarà considerato non troppo piccolo se vale:

$$\phi(\eta\alpha) > \phi(0) + \epsilon\phi'(0)\eta\alpha$$

Il metodo di Armijo viene quindi così definito. Si comincia con un valore di α arbitrario (si è usato 0.1). Se il primo test è soddisfatto, allora si aumenta moltiplicando ripetutamente α per η (si sono usati $\eta = 2$ e $\epsilon = .5$), fino a non soddisfare il test, e il penultimo valore è selezionato. Se invece il valore di α originale non soddisfacesse il primo test, allora viene ripetutamente diviso per η fino a che il valore di α risultante soddisfi il test. È stato comunque inserito un pannello nell'interfaccia grafica per poter modificare i parametri dei metodi inesatti.

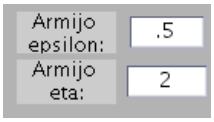


Figura 3.1: Pannello per la modifica dei parametri dei metodi inesatti.

3.3.2 Goldstein

Il test di Goldstein è un altro criterio d'arresto usato in pratica. Come nel criterio di Armijo, un valore di α è considerato non troppo grande se soddisfa:

$$\phi(\alpha) \leq \phi(0) + \epsilon\phi'(0)\alpha$$

con $0 < \epsilon < 1/2$. Un valore di α è considerato non troppo piccolo invece se vale:

$$\phi(\alpha) > \phi(0) + (1 - \epsilon)\phi'(0)\alpha$$

È stato ricavato ed implementato un *metodo di Goldstein* nello spirito del metodo di Armijo facendo anche uso del metodo di bisezione.

3.3.3 Wolfe

Il criterio di Wolfe fa uso anche dell'informazione sulla derivata ed è una variazione del criterio di Goldstein. In questo caso ϵ è selezionato in modo che $0 < \epsilon < 1/2$ ed è richiesto per α di soddisfare la condizione:

$$\phi(\alpha) \leq \phi(0) + \epsilon\phi'(0)\alpha$$

oltre che la condizione:

$$\phi'(\alpha) > (1 - \epsilon)\phi'(0)$$

Un vantaggio di questo ultimo test è l'essere invariante a fattori di scala, a differenza di Goldstein.

Anche in questo caso è stato implementato un semplice metodo di ricerca prendendo spunto dal criterio d'arresto.

chapter

Capitolo 4

Problemi per l'ottimizzazione non vincolata

I nuovi problemi di *GklsGUI* sono stati implementati scegliendoli fra problemi noti della letteratura sulla minimizzazione non vincolata. Sono state consultate raccolte online di problemi di questo tipo, ad esempio la raccolta curata da Abdel-Rahman Hedar[6] e la raccolta per il toolbox di algoritmi genetici di *MATLAB* curata da Hartmut Pohlheim [7].
Sono stati fatti inoltre degli esperimenti per ottenere statistiche di performance dei vari metodi line search implementati, anche in relazione ai nuovi problemi.

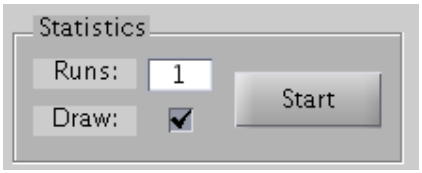


Figura 4.1: Pannello per generare le statistiche.

È stato infatti aggiunto il pannello *LS Statistics* al software. Per generare le statistiche la procedura è la seguente:

1. Si sceglie un problema dal pannello *problems* e si genera, eventualmente settando le opzioni.

2. Si sceglie la zona d'interesse, scegliendo il centro (*Plot centre*) e lo zoom/distanza ($\max |x_i - x0_i|; i = 1, 2, \ldots, n$) dal centro.

3. Si sceglie il metodo (multidimensionale) da utilizzare per la minimizzazione.

4. Eventualmente si settano le opzioni per il metodo unidimensionale.

5. Dal pannello *LS statistics* si indica il numero di run e si spunta eventualmente la casella *draw* per visualizzare graficamente l'esperimento. Si preme poi il pulsante *start* del pannello *LS statistics*.

L'esperimento è diviso nel numero prescelto di run. Per ogni run viene selezionato un punto iniziale casualmente nell'area di interesse. Quindi per ogni metodo LS presente nel sistema viene avviata una ricerca del minimo. Per ogni run si accumulano i seguenti valori: (*iterations*) numero di iterazioni effettuate, (*EvalF*) numero di valutazioni della funzione, (*EvalG*) numero di valutazioni del gradiente, (*EvalH*) numero di valutazioni dell'Hessiana, $(x_n - x^*)$ distanza raggiunta dal minimo noto (in certi problemi è la distanza dal minimo locale più vicino, in altri la distanza dal minimo globale), $(f_n - f^*)$ distanza raggiunta dal valore del minimo noto. Viene quindi effettuata la media sui run.

Al termine dell'esperimento vengono generati due file, di nome:

- `statistics_problema-metodo.csv` in formato csv, da importare con un foglio di calcolo come *Microsoft Excel* o *Openoffice.org Calc*.
- `statistics_problema-metodo.tex` in formato tex, da importare in un documento LaTeX.

4.1 Gkls

Per una descrizione dei problemi di tipo Gkls si rimanda a [4].
I risultati degli esperimenti effettuati su questo problema sono in tabella 4.1. È stato generato un problema di classe 1 con 10 minimi locali. La ricerca è stata effettuata selezionando punti d'inizio casuali nell'intervallo $-1 < x < 1, -1 < y < 1$.

I risultati in tabella 4.1 mostrano come al variare del metodo di line search si abbia comunque convergenza, in media da un minimo di 2.8 passi (Sezione Aurea e Fibonacci) a un massimo di 6.98 (Armijo). Fra i metodi più economici in termini di numero di valutazioni appare il metodo delle secanti ed il fit cubico.

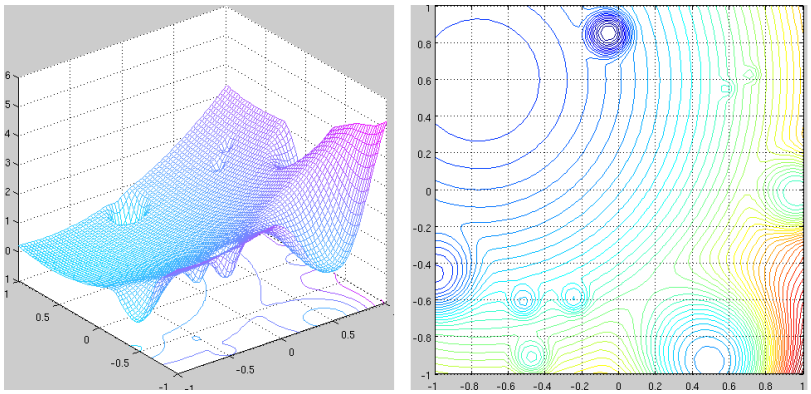


Figura 4.2: Problema di Gkls con 10 minimi locali e classe 1 su cui sono stati effettuati gli esperimenti.

4.2 Rosenbrock

La funzione di Rosenbrock ha un minimo locale nel punto $1, 1$. È un famoso banco di prova per i metodi di minimizzazione per la sua forma particolare a valle o banana. Gli algoritmi trovano facilmente il fondo della valle, ma hanno in genere difficoltà a seguire il fondo fino a raggiungere il minimo globale.

$$f(x, y) = r = 100 * (y - x^2)^2 + (1 - x)^2$$

Minimo locale: $x^* = (1, 1), f(x^*) = 0$

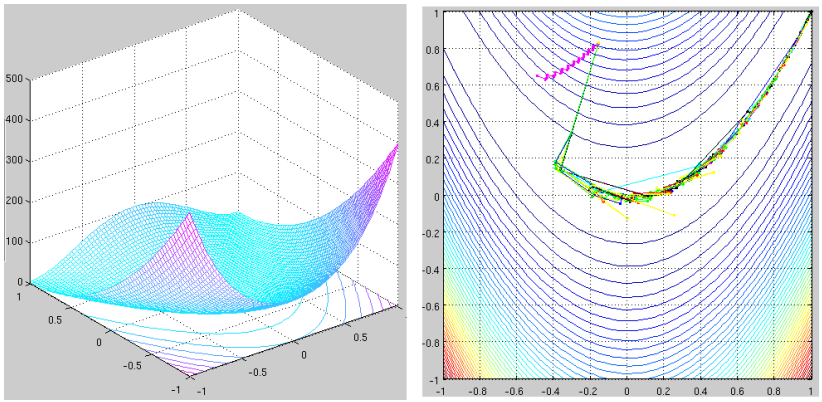


Figura 4.3: Problema di Rosenbrock. Vari metodi LS a confronto.

In tabella 4.2 i risultati degli esperimenti effettuati su questo problema. La ricerca è stata effettuata selezionando punti d'inizio casuali nell'intervallo

Tabella 4.1: Mean results for problem gkls (method DFP), centered on x:0 y:0 distance:1 number of runs:50

Method	Iterations	EvalF	EvalG	EvalH	$x_n - x^*$	$f_n - f^*$
armijo	6.98	34.8	6.98	0	0.00023274	8.3877e-08
aurea	2.8	38.48	2.8	0	0.00014893	4.2569e-08
cubicfitTPP-armijo	3.18	60.56	31.32	0	3.2461e-06	7.0504e-10
cubicfitTPP-goldstein	2.78	62.36	59.74	0	5.6963e-06	1.0477e-09
cubicfitTPP-tolAlpha	2.78	23.58	22.74	0	5.6962e-06	1.048e-09
cubicfitTPP-tolG	3.1	18.24	15.2	0	9.1239e-06	1.8612e-09
cubicfitTPP-wolfe	3.16	33.9	30.66	0	5.3148e-06	1.6879e-09
fibonacci	2.8	38.48	2.8	0	0.00014893	4.2569e-08
goldstein	3.96	40.16	3.96	0	3.5576e-05	1.3185e-08
quadfitTPP-armijo	3.48	56.7	3.48	0	1.1085e-05	2.9946e-09
quadfitTPP-goldstein	3.04	73.32	3.04	0	1.0478e-05	3.3819e-09
quadfitTPP-tolAlpha	3.04	20.92	3.04	0	1.0473e-05	3.3804e-09
quadfitTPP-tolG	3.04	42.22	37.22	0	1.5081e-05	4.704e-09
quadfitTPP-wolfe	3.16	36.4	26.86	0	1.3516e-05	4.272e-09
secant	3.4	2.4	8.78	0	2.6189e-06	2.9851e-10
secantTPP-armijo	3.34	68.76	34.3	0	8.3915e-06	1.7936e-09
secantTPP-goldstein	2.98	48.36	43.44	0	5.5947e-06	9.5774e-10
secantTPP-tolAlpha	3.02	31.38	28.28	0	5.4789e-06	9.4952e-10
secantTPP-tolG	2.98	39.54	36.6	0	6.3179e-06	1.068e-09
secantTPP-wolfe	4.2	45.06	37.12	0	1.2451e-05	4.0904e-09
wolfe	5.32	51.76	40.9	0	6.2038e-05	2.521e-08

$-1 < x < 1, -1 < y < 1$.

Si può vedere come l'utilizzo di molti metodi LS faccia aumentare di molto il numero di passi necessari per ottenere la convergenza su questo problema (es. armijo, goldstein, secanti, secanti con TPP-wolfe, wolfe). Per questi metodi il numero medio di passi sfiora il massimo numero di iterazioni disponibili. Fra gli algoritmi che raggiungono la convergenza in un piccolo numero di passi spicca il metodo delle secanti con criterio d'arresto tolleranza su alfa.

4.3 Beale

La funzione di Beale ha un minimo globale nel punto $3, 0.5$, oltre ad altri minimi locali.

Tabella 4.2: Mean results for problem rosenbrock (method DFP), centered on x:0 y:0 distance:1 number of runs:50

Method	Iterations	EvalF	EvalG	EvalH	$x_n - x^*$	$f_n - f^*$
armijo	26.34	178.3	26.34	0	0.012924	0.00057829
aurea	18.98	391.68	18.98	0	0.00019889	3.647e-08
cubicfitTPP-armijo	22.78	479.74	229	0	0.0034111	0.00010508
cubicfitTPP-goldstein	17.38	593.66	541.54	0	0.00041652	9.6139e-08
cubicfitTPP-tolAlpha	17.38	222.38	186.64	0	0.00041729	9.6243e-08
cubicfitTPP-tolG	18.18	154.3	97.32	0	0.0006883	1.465e-07
cubicfitTPP-wolfe	22.64	295.22	229.78	0	0.003473	0.00010508
fibonacci	18.92	390.08	18.92	0	0.00020505	3.6449e-08
goldstein	27.12	341.52	27.12	0	0.050984	0.005468
quadfitTPP-armijo	24.42	705.1	24.42	0	0.033623	0.0094816
quadfitTPP-goldstein	20.14	708.38	20.14	0	0.00016297	2.3246e-08
quadfitTPP-tolAlpha	20.14	294.32	20.14	0	0.00016296	2.3246e-08
quadfitTPP-tolG	20.04	656.16	580.76	0	0.00029685	7.505e-08
quadfitTPP-wolfe	22.78	435.46	300.42	0	0.0047016	9.1012e-05
secant	30	29	95.7	0	1.817	3.5409
secantTPP-armijo	24.78	475.54	209.82	0	0.0057025	0.00016705
secantTPP-goldstein	16.26	531.74	466.34	0	0.00045203	1.0528e-07
secantTPP-tolAlpha	16.14	281.74	231.88	0	0.00054216	1.5296e-07
secantTPP-tolG	16.26	481	430.86	0	0.00045203	1.0528e-07
secantTPP-wolfe	29.88	325.44	156.3	0	0.64859	18.877
wolfe	29.88	365.58	264.4	0	0.059485	0.024415

$$f(x,y) = (1.5 - x(1-y))^2 + (2.25 - x(1-y^2))^2 + (2.625 - x(1-y^3))^2$$

Minimo locale: $x^* = (3, 0.5), f(x^*) = 0$

Nell'esperimento i cui risultati sono riportati in tabella 4.3 si effettua una ricerca del minimo selezionando punti di partenza casuali nell'intervallo $-4.5 < x < 4.5, -4.5 < y < 4.5$. La distanza a cui si fa riferimento in tabella è dal minimo globale. Come si vede questo valore è sempre molto maggiore della tolleranza ($tol = 0.001$), infatti il minimo è ≈ 4.7 per il metodo delle secanti con criterio di arresto Wolfe. Questo indica che i metodi fanno fatica a convergere o convergono verso gli altri punti di minimo.

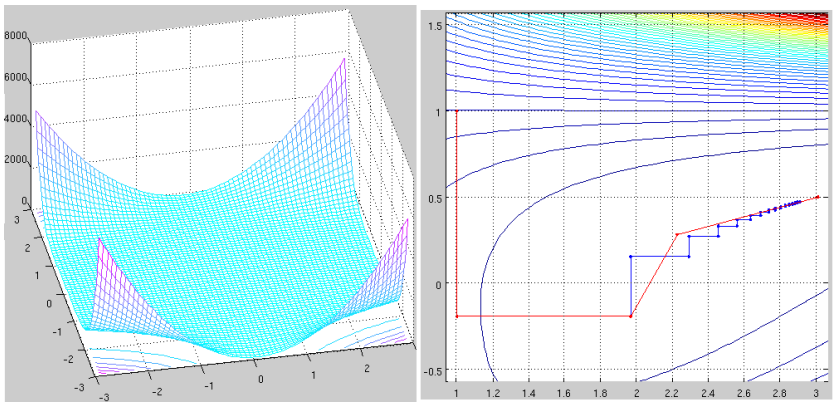


Figura 4.4: Problema di Beale: metodi del Gradiente (in blu) e DFP (in rosso) a confronto. Il primo non riesce a convergere nel massimo numero di iterazioni, il secondo arriva alla convergenza in 7 iterazioni.

4.4 Bohachevsky

La funzione presenta un andamento quadratico generale caratterizzato da vari minimi locali, derivanti dai termini del coseno, in cui gli algoritmi possono venire attratti anziché finire nel minimo globale posto nell'origine.

$$f(x,y) = x^2 + 2y^2 - 0.3\cos(3\pi x) - 0.4\cos(4\pi y) + 0.7$$

Minimo globale: $x^* = (0,0), f(x^*) = 0$

In tabella 4.4 i risultati degli esperimenti effettuati su questo problema. La ricerca del minimo è effettuata selezionando punti di partenza casuali nell'intervallo $-4 < x < 4, -4 < y < 4$. I metodi in media si arrestano distanti dal minimo globale (come si può vedere dalla colonna distanze dal minimo globale). Il metodo di fibonacci sembra il più affidabile per raggiungere il minimo globale di questo problema.

4.5 Booth

È una funzione convessa con un minimo in $(1,3)$.

$$f(x,y) = (x + 2y - 7)^2 + (2x + y - 5)^2$$

Minimo globale: $x^* = (1,3), f(x^*) = 0$

In tabella 4.5 i risultati degli esperimenti effettuati su questo problema. La ricerca del minimo è effettuata selezionando punti di partenza casuali a distanza-∞4 da $(1,3)$.

Tabella 4.3: Mean results for problem beale (method DFP), centered on x:0 y:0 distance:4.51 number of runs:50

Method	Iterations	EvalF	EvalG	EvalH	$x_n - x^*$	$f_n - f^*$
armijo	26.68	213.86	26.68	0	6.8258	0.82066
aurea	19.38	464.3	19.38	0	7.286	0.4194
cubicfitTPP-armijo	19.62	431.58	192.44	0	7.7728	0.56812
cubicfitTPP-goldstein	18.62	679.64	582.46	0	12.488	0.52868
cubicfitTPP-tolAlpha	21.58	348.12	253.04	0	10.767	0.57618
cubicfitTPP-tolG	20.14	193.2	107.28	0	7.0584	0.5508
cubicfitTPP-wolfe	19.62	286.56	200.88	0	8.0863	0.55579
fibonacci	19.38	464.44	19.38	0	7.2898	0.41941
goldstein	23.82	364.88	23.82	0	8.3686	0.85051
quadfitTPP-armijo	20.42	496.26	20.42	0	5.3584	0.68834
quadfitTPP-goldstein	18.64	692.48	18.64	0	8.0989	0.35212
quadfitTPP-tolAlpha	20.9	422.42	20.9	0	9.2625	0.37913
quadfitTPP-tolG	18.6	655.04	545.92	0	8.099	0.35212
quadfitTPP-wolfe	20.26	318.58	176.76	0	6.0373	0.85937
secant	20.92	19.92	85.98	0	14.132	2730
secantTPP-armijo	23.16	488.4	201.66	0	6.344	0.80204
secantTPP-goldstein	21.1	751.7	617.9	0	12.796	0.86432
secantTPP-tolAlpha	20.66	485	363.1	0	10.588	0.86239
secantTPP-tolG	21.1	697.78	584.08	0	12.796	0.86432
secantTPP-wolfe	28.48	435.28	94.86	0	4.7577	9848.9
wolfe	26.46	361.12	228.12	0	9.9526	0.94311

Si riesce ad arrivare a convergenza con tutti i metodi LS, solo il metodo di Newton e il metodo delle secanti con TPP e criterio d'arresto di Wolfe sembrano avere problemi.

4.6 Easom

È una funzione unimodale con un minimo pronunciato situato al centro di una vasta superficie leggermente decrescente asintoticamente.

$$f(x,y) = -\prod_{i=1}^{n} \cos(x_i) \exp(-\sum_{i=1}^{n} (x_i - \pi)^2)$$

Minimo globale: $x^* = (\pi, \pi), f(x^*) = -1$

In tabella 4.6 i risultati degli esperimenti effettuati su questo problema.

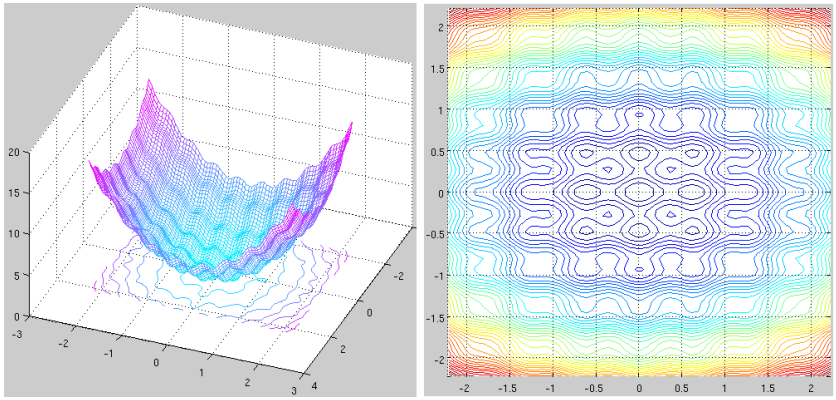


Figura 4.5: La funzione di Bohachevsky

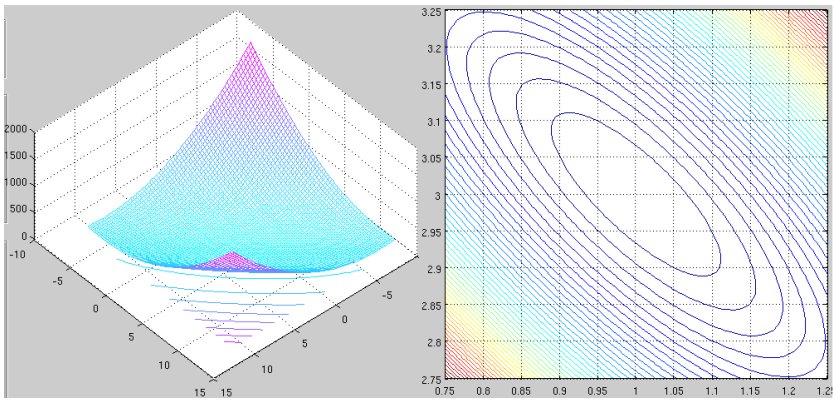


Figura 4.6: Problema di Booth.

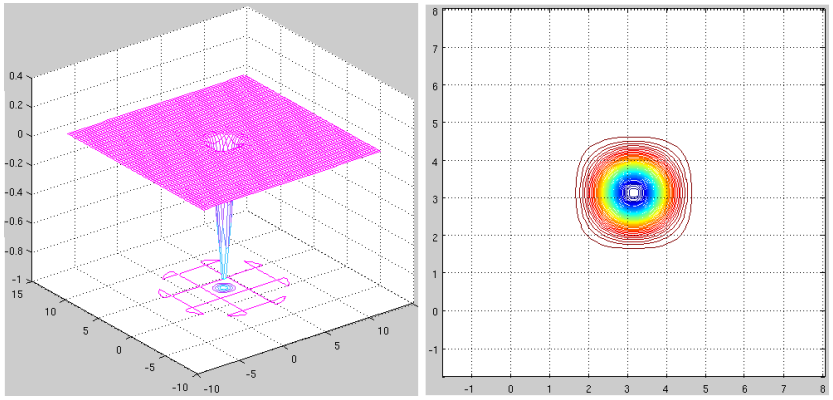


Figura 4.7: La funzione di Easom

Tabella 4.4: Mean results for problem bohachevsky (method SteepestDescent), centered on x:0 y:0 distance:4 number of runs:50

Method	Iterations	EvalF	EvalG	EvalH	$x_n - x^*$	$f_n - f^*$
armijo	8.62	33.94	8.62	0	0.84632	1.2751
aurea	8.66	173.24	8.66	0	0.44174	0.49915
cubicfitTPP-armijo	7.94	240.18	115	0	0.50458	0.59214
cubicfitTPP-goldstein	7.42	236.3	212.86	0	0.51826	0.52714
cubicfitTPP-tolAlpha	7.38	116.08	99.24	0	0.51826	0.52714
cubicfitTPP-tolG	7.36	108.76	85.7	0	0.52136	0.52578
cubicfitTPP-wolfe	8.04	135.78	111.14	0	0.58268	0.69611
fibonacci	8.94	179.12	8.94	0	0.44184	0.49889
goldstein	8.44	74	8.44	0	0.92983	1.4835
quadfitTPP-armijo	7.96	300.12	7.96	0	0.60161	0.67238
quadfitTPP-goldstein	7.62	250.74	7.62	0	0.49073	0.49374
quadfitTPP-tolAlpha	7.62	66.64	7.62	0	0.49073	0.49374
quadfitTPP-tolG	7.62	239.4	208.16	0	0.49073	0.49374
quadfitTPP-wolfe	7.78	173.56	128.38	0	0.4849	0.50367
secant	16.4	15.4	62.1	0	551.68	2.579e+07
secantTPP-armijo	8.96	341.6	162.12	0	0.61421	0.72326
secantTPP-goldstein	8.52	265.3	232.52	0	0.59915	0.70708
secantTPP-tolAlpha	8.5	211.18	186.1	0	0.61377	0.71736
secantTPP-tolG	8.52	250.72	225.46	0	0.59915	0.70708
secantTPP-wolfe	15.8	251.46	177.56	0	0.72709	0.96171
wolfe	7.64	50.94	40.96	0	0.93075	1.645

La ricerca del minimo è effettuata selezionando punti di partenza casuali a distanza-∞4 da (π, π), minimo globale.
Come si vede dalla colonna distanza dal minimo globale, si hanno problemi di convergenza per questo problema a prescindere dal metodo LS utilizzato. La maggior parte degli algoritmi si ferma alla prima/seconda iterazione a causa del raggiungimento della condizione d'arresto sul gradiente, pur essendo lontani dal minimo globale. Alcuni invece (Fibonacci e sezione aurea), segnati con un asterisco, convergono verso il minimo asintotico.

4.7 Hump

La cosiddetta funzione a gobba, o dorso di cammello (per via dei due minimi globali gemelli). La funzione possiede anche altri minimi locali.

$$f(x,y) = 1.0316285 + 4x^2 - 2.1x^4 + x^6/3 + xy - 4y^2 + 4y^4$$

Minimo globale: $x^* = (0.0898, -0.7126), (0.0898, -0.7126), f(x^*) = 0$

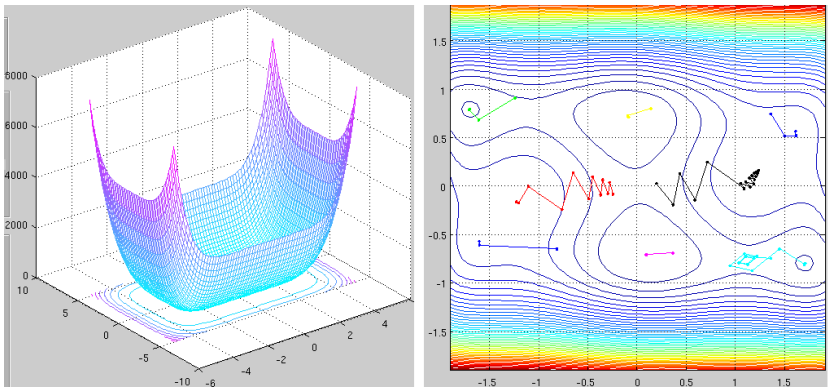


Figura 4.8: Funzione Hump: Convergenza del metodo di Levenberg-Marquardt a differenti minimi locali a seconda del punto iniziale.

In tabella 4.7 i risultati degli esperimenti effettuati su questo problema. La ricerca del minimo è effettuata selezionando punti di partenza casuali a distanza-∞2 da $(0,0)$. Le distanze riportate in tabella sono rispetto ai due minimi globali.

4.8 Matyas

$$f(x,y) = 0.26(x^2 + y^2) - 0.48xy;$$

Minimo globale: $x^* = (0,0), f(x^*0) =$

In tabella 4.8 i risultati degli esperimenti effettuati su questo problema. La ricerca del minimo è effettuata selezionando punti di partenza casuali a distanza-∞4 da $(0,0)$.

La convergenza avviene sempre al variare del metodo LS utilizzato eccetto il caso della secante (senza TPP) che diverge per qualche run e i metodi secante con criterio d'arresto Wolfe ed il metodo di Armijo che possono incorrere in problemi. I metodi di curve fit con TPP e criterio d'arresto di Armijo sembrano particolarmente efficaci in questo caso.

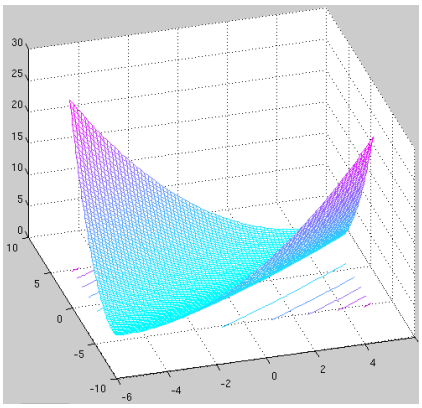


Figura 4.9: La funzione di Matyas

4.9 Michalewicz

È una funzione multimodale con vari punti di minimo locale, il parametro m che indica la *ripidità* della funzione è stato settato a 20 nell'implementazione.

$$f(x1, \ldots, x_n) = -\sum_{i=1}^{n} \sin(x_i) \sin\left(ix_i^2/\pi\right)^m$$

Minimo globale: $n = 2, f(x^*) = -1.8013$

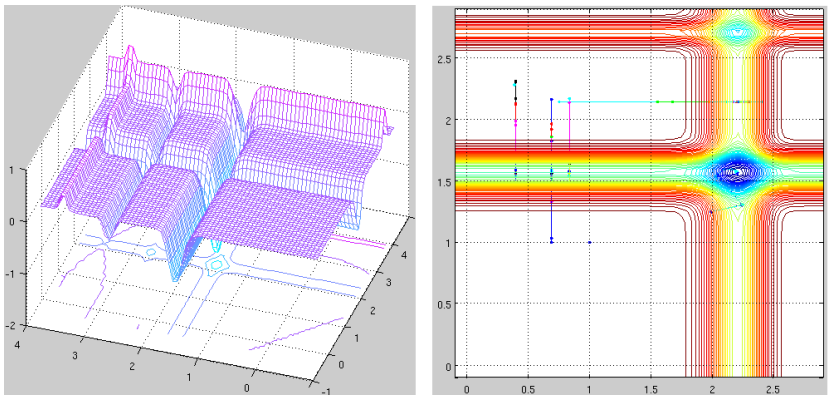


Figura 4.10: La funzione di Michalewicz. I metodi hanno difficoltà a convergere, come si vede dalla figura a destra.

In tabella 4.9 i risultati degli esperimenti effettuati su questo problema. La ricerca del minimo è effettuata selezionando punti di partenza casuali a distanza-∞1.5 da $(1.4, 1.4)$. La distanza riportata è quella dal minimo

in $(2.2029, 1.5708)$. Gli algoritmi hanno problemi a raggiungere un minimo locale, tendono a rimanere intrappolati nelle *valli* e sugli *altipiani*.

4.10 Michalewicz modificata

È ottenuta aggiungendo un termine quadratico alla funzione di Micheaelewicz originale, in modo da ottenere un minimo globale nell'origine, pur mantenendo i minimi locali tipici della funzione originaria.

$$f(x1, \ldots, x_n) = \sum_{i=1}^{n} (x_i^2/\pi) - \sum_{i=1}^{n} \sin x_i \sin i x_i^2/\pi^2 0;$$

Minimo globale: $x^* = (0, 0), f(x^*) = -1.8013$

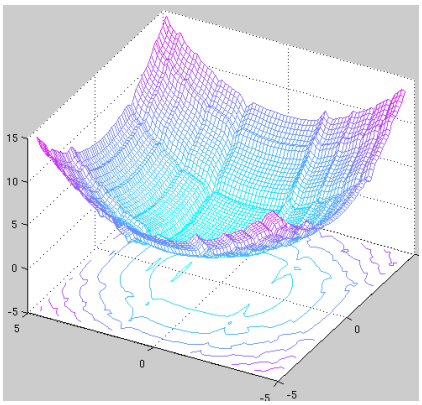


Figura 4.11: La funzione di Michalewicz modificata

In tabella 4.10 i risultati degli esperimenti effettuati su questo problema. La ricerca del minimo è effettuata selezionando punti di partenza casuali a distanza-∞5 da $(0, 0)$. I metodi tendono a rimanere intrappolati nei minimi locali.

4.11 Schwefel

La funzione di Schwefel è caratterizzata da svariati minimi e massimi locali, che allontanandosi dall'origine si fanno via via più importanti, facendo potenzialmente dirigere gli algoritmi nella direzione sbagliata, cioè verso il centro.

$$f(x1, \ldots, x_n) = 418.9829 * n - \sum_{i=1}^{n} x_i \sin \sqrt{|x_i|}$$

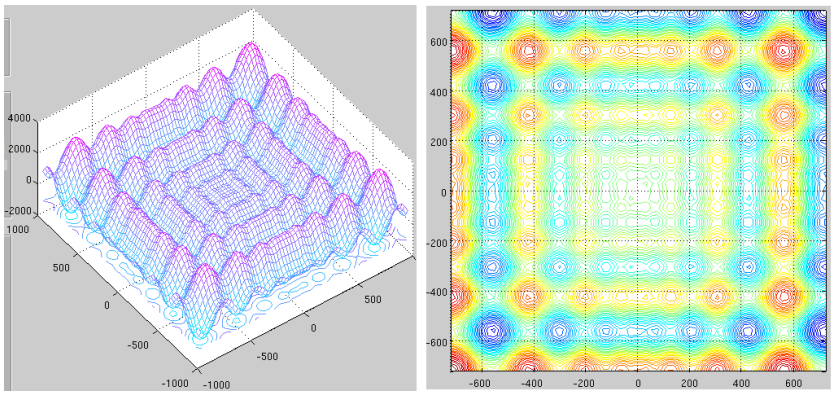


Figura 4.12: La funzione di Schwefel: si notano i numerosi minimi via via migliori allontanandosi dall'origine

In tabella 4.11 i risultati degli esperimenti effettuati su questo problema. La ricerca del minimo è effettuata selezionando punti di partenza casuali a distanza-∞100 da $(0, 0)$.

4.12 Schwefel modificata

Questa funzione è stata ottenuta dalla funzione di Schwefel in modo da ottenere un minimo globale posto nell'origine e al contempo mantenere le gibbosità proprie della funzione originale.

$$f(x1, \ldots, x_n) = \sum_{i=1}^{n} |x_i| + |x_i \sin \sqrt{|x_i|}|$$

Minimo globale: $x^* = (0, 0), f(x^*) = 0$

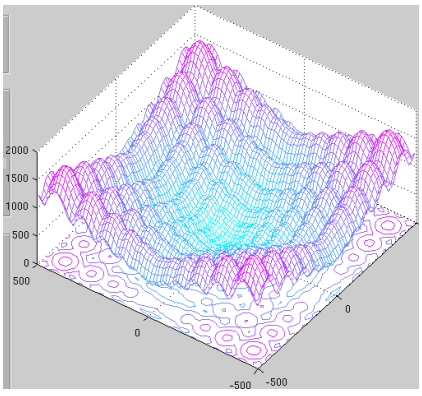


Figura 4.13: Funzione di Schwefel modificata

In tabella 4.12 i risultati degli esperimenti effettuati su questo problema. La ricerca del minimo è effettuata selezionando punti di partenza casuali a distanza-∞100 da $(0, 0)$. I metodi convergono verso i numerosi minimi locali.

4.13 Sphere

È una semplice funzione di test molto usata, continua, convessa e unimodale.

$$f(x1, \ldots, x_n) = \sum_{i=1}^{n} x_i^2$$

Minimo globale: $x^* = (0, 0), f(x^*) = 0$

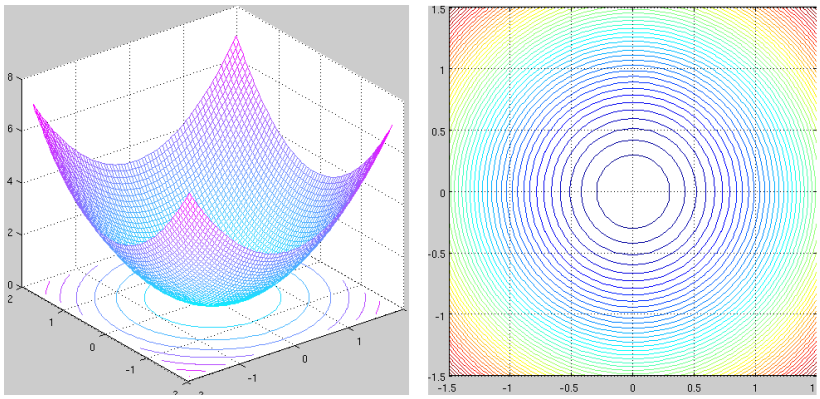


Figura 4.14: Funzione sfera

In tabella 4.13 i risultati degli esperimenti effettuati su questo problema. La ricerca del minimo è effettuata selezionando punti di partenza casuali a distanza-∞100 da $(0, 0)$. Tutti i metodi eccetto il metodo delle secanti con criterio d'arresto di Wolfe convergono con grande precisione in pochi passi.

4.14 Sphere modificata

Questa funzione è una variazione della funzione sfera precedente, ottenuta inserendo un coefficiente in modo da far variare la curvatura e far peggiorare il condizionamento del problema.

$$f(x, y) = 50x^2 + y^2$$

Minimo globale: $x^* = (0, 0), f(x^*) = 0$

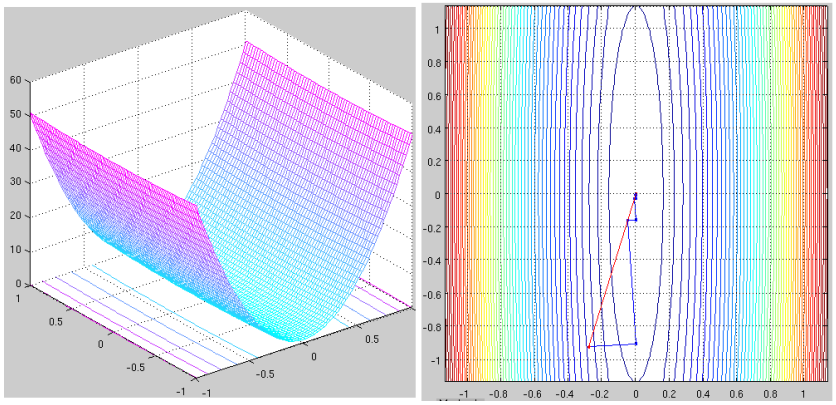


Figura 4.15: Funzione sfera modificata: il metodo del Gradiente (in blu) impiega 12 iterazioni per raggiungere l'intervallo di confidenza, mentre il metodo di Newton (in rosso) lo raggiunge in 2 iterazioni. Entrambi usano il metodo di Newton per la minimizzazione unidimensionale.

In tabella 4.14 i risultati degli esperimenti effettuati su questo problema. La ricerca del minimo è effettuata selezionando punti di partenza casuali a distanza-∞100 da $(0, 0)$. In questo caso le iterazioni necessarie per la convergenza aumentano rispetto alla sfera regolare, impedendo ad alcuni metodi la convergenza entro il massimo numero di iterazioni.

Tabella 4.5: Mean results for problem booth (method LevMarq), centered on x:1 y:3 distance:4 number of runs:50

Method	Iterations	EvalF	EvalG	EvalH	$x_n - x^*$	$f_n - f^*$
armijo	7.72	47.04	7.72	7.72	7.7886e-05	2.0599e-08
aurea	2.76	45.42	2.76	2.76	2.7524e-05	8.2662e-09
cubicfitTPP-armijo	2.04	37.26	17.3	2.04	8.5909e-07	8.9756e-11
cubicfitTPP-goldstein	2	39.7	34	2	7.9351e-16	1.9248e-30
cubicfitTPP-tolAlpha	2	15.1	10.4	2	7.9351e-16	1.9248e-30
cubicfitTPP-tolG	2	10.68	4.98	2	8.5909e-07	8.9756e-11
cubicfitTPP-wolfe	2.04	23.04	17.3	2.04	8.5909e-07	8.9756e-11
fibonacci	2.76	45.42	2.76	2.76	2.7524e-05	8.2664e-09
goldstein	2.98	33.66	2.98	2.98	3.2743e-06	2.5204e-10
newton	2	1	32	32	6.4292e-16	8.993e-31
newtonTPP-armijo	2.04	38.5	17.92	18.96	8.5909e-07	8.9756e-11
newtonTPP-goldstein	2	39.7	34	35	6.1669e-16	9.9396e-31
newtonTPP-tolAlpha	2	32.22	27.52	28.52	6.1669e-16	9.9396e-31
newtonTPP-tolG	2	9.68	4.98	5.98	8.5909e-07	8.9756e-11
newtonTPP-wolfe	22.28	201.32	64.84	86.12	0.99677	21.3
quadfitTPP-armijo	2.04	21.26	2.04	2.04	8.5909e-07	8.9756e-11
quadfitTPP-goldstein	2	40.7	2	2	6.4571e-16	1.0571e-30
quadfitTPP-tolAlpha	2	14.42	2	2	6.4571e-16	1.0571e-30
quadfitTPP-tolG	2	10.68	3.98	2	8.5909e-07	8.9756e-11
quadfitTPP-wolfe	2	15.4	6.7	2	3.2523e-15	2.1194e-28
secant	2	1	3.98	2	8.5909e-07	8.9756e-11
secantTPP-armijo	2.58	20.86	8.3	2.58	4.1273e-06	1.085e-09
secantTPP-goldstein	2	15.04	8.34	2	6.7148e-16	1.2306e-30
secantTPP-tolAlpha	2	13.5	7.8	2	6.7148e-16	1.2306e-30
secantTPP-tolG	2	13.32	7.62	2	6.9201e-16	1.3726e-30
secantTPP-wolfe	23.46	228.5	67.14	23.46	0.99679	21.3
wolfe	3.5	30.42	21.42	3.5	2.0703e-05	6.4412e-09

Tabella 4.6: Mean results for problem easom (method LevMarq), centered on x:3.1415 y:3.1415 distance:4 number of runs:50. The asterisk at the end of the method name indicates that the method did not converged for some run.

Method	Iterations	EvalF	EvalG	EvalH	$x_n - x^*$	$f_n - f^*$
armijo	2.74	12.1	2.74	2.74	2.5493	0.70001
aurea*	1.7143	15.6429	1.7143	1.7143	2.5937	0.66667
cubicfitTPP-armijo	2.06	273.86	15.52	2.06	2.558	0.70002
cubicfitTPP-goldstein	1.96	278.52	32.68	1.96	2.558	0.70002
cubicfitTPP-tolAlpha	1.96	259.06	14.18	1.96	2.558	0.70002
cubicfitTPP-tolG	2	256.1	10.22	2	2.558	0.70002
cubicfitTPP-wolfe	2.02	259.6	13.7	2.02	2.558	0.70002
fibonacci*	1.7143	15.6429	1.7143	1.7143	2.5937	0.66667
goldstein	2.46	23.82	2.46	2.46	2.5665	0.7
newton	1.78	0.78	25.18	25.18	4.0774	1.0009
newtonTPP-armijo	2.06	195.46	17.28	18.34	2.741	0.7
newtonTPP-goldstein	1.94	195.92	32.02	32.96	2.741	0.7
newtonTPP-tolAlpha	1.94	184.84	21.88	22.82	2.741	0.7
newtonTPP-tolG	1.94	181.36	18.4	19.34	2.5604	0.70002
newtonTPP-wolfe	2.22	257.82	11.72	12.94	2.5748	0.70002
quadfitTPP-armijo	2.16	274.56	2.16	2.16	2.558	0.70002
quadfitTPP-goldstein	2.06	282.98	2.06	2.06	2.558	0.70002
quadfitTPP-tolAlpha	2.06	253.34	2.06	2.06	2.558	0.70002
quadfitTPP-tolG	2.06	268.24	21.24	2.06	2.558	0.70002
quadfitTPP-wolfe	2.1	261.8	12.52	2.1	2.558	0.70002
secant	2.46	1.46	5.14	2.46	3.0692	0.86038
secantTPP-armijo	2.3	289.36	22.74	2.3	2.558	0.70002
secantTPP-goldstein	2.22	285.74	38.42	2.22	2.558	0.70002
secantTPP-tolAlpha	2.22	272.86	26.76	2.22	2.558	0.70002
secantTPP-tolG	2.02	266.02	20.12	2.02	2.9687	0.78002
secantTPP-wolfe	2.34	262.02	14.46	2.34	2.558	0.70002
wolfe	2.7	23.88	18.06	2.7	2.6374	0.72

Tabella 4.7: Mean results for problem hump (method DFP), centered on x:0 y:0 distance:2 number of runs:50

Method	Iterations	EvalF	EvalG	EvalH	$x_n - x^*$	$f_n - f^*$
armijo	10.9	50.22	10.9	0	0.72896	0.77826
aurea	5.6	99.76	5.6	0	0.65514	0.55844
cubicfitTPP-armijo	6.34	134	66.1	0	0.60072	0.54905
cubicfitTPP-goldstein	5.58	162.32	152.14	0	0.69267	0.71394
cubicfitTPP-tolAlpha	5.6	67.24	61.62	0	0.69267	0.71394
cubicfitTPP-tolG	7.04	43.2	30.3	0	0.69267	0.71394
cubicfitTPP-wolfe	6.28	80.78	69.18	0	0.62281	0.54211
fibonacci	5.6	99.76	5.6	0	0.65514	0.55844
goldstein	6.88	70.32	6.88	0	0.72673	0.77666
quadfitTPP-armijo	6.7	142.04	6.7	0	0.66842	0.65932
quadfitTPP-goldstein	6.6	206.04	6.6	0	0.70284	0.65323
quadfitTPP-tolAlpha	6.6	66.12	6.6	0	0.70284	0.65323
quadfitTPP-tolG	6.6	171.54	151.3	0	0.70284	0.65323
quadfitTPP-wolfe	6.74	97.62	66.36	0	0.66798	0.58664
secant	9.06	8.06	30.34	0	0.80647	1.0418
secantTPP-armijo	7.46	118.6	52.56	0	0.65028	0.57125
secantTPP-goldstein	6.04	171.24	155.12	0	0.68919	0.62115
secantTPP-tolAlpha	6.02	81.16	70.08	0	0.68919	0.62115
secantTPP-tolG	6.04	139.82	128.74	0	0.68919	0.62115
secantTPP-wolfe	11.5	122.78	68.82	0	0.5608	0.55854
wolfe	10.76	102.02	84.68	0	0.72674	0.77666

Tabella 4.8: Mean results for problem matyas (method SteepestDescent), centered on x:0 y:0 distance:4 number of runs:50. The asterisk at the end of the method name indicates that the method did not converged for some run.

Method	Iterations	EvalF	EvalG	EvalH	$x_n - x^*$	$f_n - f^*$
armijo	19.82	154.52	19.82	0	0.30154	0.0023056
aurea	9.16	217.62	9.16	0	0.0384	0.00012621
cubicfitTPP-armijo	5.88	114.12	54.78	0	4.6837e-06	8.1716e-12
cubicfitTPP-goldstein	9.16	293.46	270.28	0	0.037875	0.00012493
cubicfitTPP-tolAlpha	9.16	151.02	136	0	0.037875	0.00012493
cubicfitTPP-tolG	16.08	124.6	49.86	0	0.17036	0.00084954
cubicfitTPP-wolfe	5.88	70.1	54.78	0	4.6837e-06	8.1716e-12
fibonacci	9.16	217.62	9.16	0	0.0384	0.00012621
goldstein	9.24	164.2	9.24	0	0.044632	0.00013924
quadfitTPP-armijo	5.88	103.48	5.88	0	4.6837e-06	8.1716e-12
quadfitTPP-goldstein	9.16	301.62	9.16	0	0.037875	0.00012493
quadfitTPP-tolAlpha	9.16	117.32	9.16	0	0.037875	0.00012493
quadfitTPP-tolG	9.16	267.28	235.84	0	0.040008	0.00012569
quadfitTPP-wolfe	9.16	137.9	90.24	0	0.037875	0.00012493
secant*	5.9167	4.9167	20.8889	0	0.011347	7.3745e-06
secantTPP-armijo	5.88	81.68	36.12	0	4.6837e-06	8.1716e-12
secantTPP-goldstein	9.16	190.2	158.86	0	0.037875	0.00012493
secantTPP-tolAlpha	9.16	126.68	103.5	0	0.037875	0.00012493
secantTPP-tolG	9.16	157.98	134.8	0	0.038157	0.00012496
secantTPP-wolfe	28.5	263.56	83.5	0	0.70027	0.012722
wolfe	6.1	76.24	44.68	0	0.14111	0.0057686

Tabella 4.9: Mean results for problem michalewicz (method DFP), centered on x:1.4 y:1.4 distance:1.5 number of runs:50

Method	Iterations	EvalF	EvalG	EvalH	$x_n - x^*$	$f_n - f^*$
armijo	7.8	44.68	7.8	0	1.2485	0.8981
aurea	3.28	56.1	3.28	0	1.3157	0.92238
cubicfitTPP-armijo	4.64	112.3	54.88	0	1.2469	0.88635
cubicfitTPP-goldstein	3.8	103.06	93.4	0	1.2372	0.87461
cubicfitTPP-tolAlpha	3.82	60.86	54.08	0	1.2372	0.87461
cubicfitTPP-tolG	5.36	78.74	50.22	0	1.2485	0.95093
cubicfitTPP-wolfe	4.52	62.42	51.74	0	1.2469	0.88635
fibonacci	3.28	56.14	3.28	0	1.3157	0.92238
goldstein	5.88	68.36	5.88	0	1.2703	0.8981
quadfitTPP-armijo	4.54	101.28	4.54	0	1.3082	0.90379
quadfitTPP-goldstein	4.1	116.74	4.1	0	1.3065	0.90238
quadfitTPP-tolAlpha	4.1	31.06	4.1	0	1.3065	0.90238
quadfitTPP-tolG	4.02	85.42	72.58	0	1.315	0.92238
quadfitTPP-wolfe	4.32	63.12	43.72	0	1.3187	0.92238
secant	3.96	2.96	11.64	0	1.5579	1.6051
secantTPP-armijo	5.46	164.08	75.78	0	1.2826	0.95013
secantTPP-goldstein	4.96	142.52	123.76	0	1.2826	0.94665
secantTPP-tolAlpha	4.96	111.32	96.52	0	1.2826	0.94665
secantTPP-tolG	4.08	82.02	72.42	0	1.3844	1.2481
secantTPP-wolfe	6.16	116.44	74.68	0	1.255	1.0405
wolfe	7.04	67.3	50.78	0	1.2977	0.95015

Tabella 4.10: Mean results for problem michalewiczMod (method SteepestDescent), centered on x:0 y:0 distance:5 number of runs:50

Method	Iterations	EvalF	EvalG	EvalH	$x_n - x^*$	$f_n - f^*$
armijo	10.58	51.28	10.58	0	1.4917	1.3537
aurea	6.48	131.34	6.48	0	0.33261	0.071893
cubicfitTPP-armijo	7.12	159.7	78.22	0	0.46931	0.18933
cubicfitTPP-goldstein	6.16	160.5	146.96	0	0.36017	0.083177
cubicfitTPP-tolAlpha	6.16	88.76	80.38	0	0.32899	0.078778
cubicfitTPP-tolG	6.18	83.32	69	0	0.34049	0.080522
cubicfitTPP-wolfe	7.94	99.86	80.12	0	0.52564	0.21968
fibonacci	6.48	131.08	6.48	0	0.33145	0.072994
goldstein	9.28	97.44	9.28	0	0.92683	0.80696
quadfitTPP-armijo	8	180.34	8	0	0.29271	0.059657
quadfitTPP-goldstein	6.42	173.34	6.42	0	0.35937	0.079649
quadfitTPP-tolAlpha	6.42	45.78	6.42	0	0.35937	0.079649
quadfitTPP-tolG	6.42	124.16	104.02	0	0.35941	0.079649
quadfitTPP-wolfe	6.68	113.5	84.02	0	0.36013	0.078962
secant	7.74	6.74	23.08	0	33.877	6463.2
secantTPP-armijo	8.5	195.78	91.88	0	0.39797	0.15015
secantTPP-goldstein	7.14	158.04	135	0	0.50288	0.14418
secantTPP-tolAlpha	7.14	119.66	102.8	0	0.50288	0.14418
secantTPP-tolG	7.14	147.38	130.08	0	0.50291	0.14418
secantTPP-wolfe	10.76	173.52	134.7	0	1.0194	0.54338
wolfe	8.02	65.44	46.78	0	1.0496	1.0212

Tabella 4.11: Mean results for problem schwefel (method SteepestDescent), centered on x:0 y:0 distance:100 number of runs:50

Method	Iterations	EvalF	EvalG	EvalH	$x_n - x^*$	$f_n - f^*$
armijo	7.36	58.04	7.36	0	90.513	725.86
aurea	6.04	145.84	6.04	0	306.03	897.54
cubicfitTPP-armijo	6.52	142.94	65.72	0	1213.3	2190.9
cubicfitTPP-goldstein	6.04	189.38	167.32	0	1378.3	2387.9
cubicfitTPP-tolAlpha	5.96	98.74	81.7	0	465.54	1100.9
cubicfitTPP-tolG	7.52	63.16	36.46	0	1212.1	2192.7
cubicfitTPP-wolfe	6.34	85.54	63	0	277.39	902.72
fibonacci	6.02	145.52	6.02	0	306.03	897.54
goldstein	6.5	118.54	6.5	0	89.927	726.26
quadfitTPP-armijo	6.36	170.72	6.36	0	205.97	768.07
quadfitTPP-goldstein	6.22	201.56	6.22	0	205.91	768.48
quadfitTPP-tolAlpha	6.24	83.34	6.24	0	205.91	768.48
quadfitTPP-tolG	8.38	140.94	104.92	0	205.91	768.48
quadfitTPP-wolfe	6.38	107.28	67.74	0	206.18	767.67
secant	8.32	7.32	26.3	0	893.13	1818.8
secantTPP-armijo	6.84	204.96	91.66	0	2983.1	4209.3
secantTPP-goldstein	6.7	210.82	177.3	0	2987.3	4205.4
secantTPP-tolAlpha	6.66	128.24	100.9	0	974.78	1636.5
secantTPP-tolG	9.84	160.12	122.74	0	2987.3	4205.4
secantTPP-wolfe	30.44	339.52	99.76	0	227.48	719.92
wolfe	7.82	116.62	75.3	0	100.06	710.04

Tabella 4.12: Mean results for problem schwefelMod (method SteepestDescent), centered on x:0 y:0 distance:100 number of runs:50

Method	Iterations	EvalF	EvalG	EvalH	$x_n - x^*$	$f_n - f^*$
armijo	11.42	87.04	11.42	0	58.853	84.271
aurea	9.04	201.76	9.04	0	42.264	61.152
cubicfitTPP-armijo	12.06	555.72	261.36	0	43.442	57.244
cubicfitTPP-goldstein	10.54	366.06	315.82	0	43.049	56.469
cubicfitTPP-tolAlpha	10.78	332.2	292.36	0	42.022	55.605
cubicfitTPP-tolG	10.72	350.94	299.92	0	43.049	56.451
cubicfitTPP-wolfe	10.74	287.4	236.66	0	43.36	57.531
fibonacci	9.12	203.54	9.12	0	42.264	61.153
goldstein	14.66	175	14.66	0	59.524	82.88
quadfitTPP-armijo	11.52	433.4	11.52	0	42.656	57.235
quadfitTPP-goldstein	9.84	363.92	9.84	0	43.244	57.988
quadfitTPP-tolAlpha	9.86	147.52	9.86	0	43.245	57.987
quadfitTPP-tolG	9.84	355.08	283.88	0	43.244	57.988
quadfitTPP-wolfe	10.42	257.06	168.82	0	43.053	57.773
secant	7.4	6.4	18.38	0	506.82	1300.1
secantTPP-armijo	16.66	835.86	390.04	0	18.4	24.324
secantTPP-goldstein	16.4	571.62	477.26	0	16.296	20.697
secantTPP-tolAlpha	15.72	483.22	409.02	0	20.291	26.024
secantTPP-tolG	16.4	555.74	476.78	0	16.296	20.697
secantTPP-wolfe	28.86	401.7	176.38	0	62.674	113.97
wolfe	13.04	118.72	58.76	0	60.94	84.855

Tabella 4.13: Mean results for problem sphere (method SteepestDescent), centered on x:0 y:0 distance:100 number of runs:50

Method	Iterations	EvalF	EvalG	EvalH	$x_n - x^*$	$f_n - f^*$
armijo	8.14	42.84	8.14	0	0.00079739	7.5761e-07
aurea	4.12	77.72	4.12	0	0	0
cubicfitTPP-armijo	4	33.7	12.04	0	0	0
cubicfitTPP-goldstein	4	58.08	43.04	0	0	0
cubicfitTPP-tolAlpha	4	23.68	11.64	0	0	0
cubicfitTPP-tolG	4	23.16	8.12	0	0	0
cubicfitTPP-wolfe	4	27.08	12.04	0	0	0
fibonacci	4.12	77.72	4.12	0	0	0
goldstein	4.96	59.4	4.96	0	2.9562e-09	1.601e-16
quadfitTPP-armijo	4	29.52	4	0	0	0
quadfitTPP-goldstein	4	58.64	4	0	0	0
quadfitTPP-tolAlpha	4	23.64	4	0	0	0
quadfitTPP-tolG	4	23.04	7	0	0	0
quadfitTPP-wolfe	4	26.36	6.7	0	0	0
secant	4	3	8	0	7.1621e-11	1.2432e-20
secantTPP-armijo	4	29.72	8.96	0	7.9248e-15	2.3813e-28
secantTPP-goldstein	4	26.32	9.72	0	2.4265e-15	6.6517e-29
secantTPP-tolAlpha	4	23.24	9.64	0	2.4265e-15	6.6517e-29
secantTPP-tolG	4	23.56	9.64	0	3.5899e-15	8.496e-29
secantTPP-wolfe	31	540.86	91	0	75.854	6624.1
wolfe	4	10	8	0	0	0

Tabella 4.14: Mean results for problem sphereMod (method SteepestDescent), centered on x:0 y:0 distance:100 number of runs:50

Method	Iterations	EvalF	EvalG	EvalH	$x_n - x^*$	$f_n - f^*$
armijo	31	135.3	31	0	9.7774	121.88
aurea	28.54	601.26	28.54	0	6.3968	94.281
cubicfitTPP-armijo	12.46	396.24	173.24	0	0.66389	9.5133
cubicfitTPP-goldstein	10.92	395.62	328.36	0	0.64727	9.5096
cubicfitTPP-tolAlpha	10.92	196.24	138.9	0	0.64727	9.5096
cubicfitTPP-tolG	10.94	106.74	39.3	0	0.64671	9.5096
cubicfitTPP-wolfe	12.46	246.92	173.24	0	0.66389	9.5133
fibonacci	28.8	601.32	28.8	0	6.4178	94.494
goldstein	28.5	253.7	28.5	0	10.318	178.74
quadfitTPP-armijo	12.46	376.62	12.46	0	0.66389	9.5133
quadfitTPP-goldstein	10.92	405.54	10.92	0	0.64727	9.5096
quadfitTPP-tolAlpha	10.92	143.86	10.92	0	0.64727	9.5096
quadfitTPP-tolG	10.92	378.62	301.44	0	0.64727	9.5096
quadfitTPP-wolfe	10.92	223.16	126.14	0	0.64727	9.5096
secant	10.92	9.92	42.04	0	0.64728	9.5096
secantTPP-armijo	12.46	283.66	111.22	0	0.66389	9.5133
secantTPP-goldstein	10.92	251.66	174.48	0	0.64727	9.5096
secantTPP-tolAlpha	10.92	181.44	114.18	0	0.64727	9.5096
secantTPP-tolG	10.92	233.1	165.84	0	0.64728	9.5096
secantTPP-wolfe	31	653.72	91	0	78.886	1.7142e+05
wolfe	30.36	210.18	157.04	0	5.0328	75.372

Capitolo 5

Altre modifiche

5.1 Supporto e compatibilità

GklsGUI è adesso compatibile con *MATLAB 7.7* e *MATLAB 7.8*. Per lanciare la versione per *MATLAB 7.7* è sufficiente lanciare il comando `run_7_7` al posto di `run`.
Una modifica alla capitalizzazione dei nomi dei path è stata necessaria per rendere possibile la compatibilità con i sistemi Linux.
Inoltre su alcuni sistemi (ad esempio Linux 64 bit) *MATLAB* è in abbinamento con un compilatore C ANSI standard. Questo tipo di compilatore è più restrittivo dei compilatori C più diffusi: è stato così necessario apportare modifiche alla libreria *GKLS* e alla libreria di supporto *Gkls2MATLAB*.
In particolare sono stati modificati i file `gkls.c, gkls2matlab.c, gkls2matlab2.c` in questo modo:

- Rimpiazzati i commenti del tipo `//` con `/*`
- Rimpiazzato un nome di variabile: `strlen` con `strLength`

Capitolo 6

Conclusioni

Sono state apportate modifiche ed aggiunte al software GklsGUI: in particolare sono stati implementati metodi per la ricerca unidimensionale e problemi su cui testare la minimizzazione, oltre a modifiche all'interfaccia grafica e per la compatibilità.
Sono stati presentati gli algoritmi e i nuovi problemi. Sono stati poi presentati i risultati di alcuni esperimenti effettuati utilizzando la nuova funzione per generare statistiche automatiche, con la quale è possibile comparare gli algoritmi al variare del metodo line search utilizzato e del problema ed esportare i risultati in un foglio di calcolo o in un documento LaTeX.

Bibliografia

[1] V. Angelini, *GklsGUI, Un insieme di tool didattici per l'ottimizzazione*, Tesi di laurea Università di Firenze, 2004-2005.

[2] L. Brugnano, C. Magherini, and A. Sestini, *Calcolo numerico*, Master Università e Professioni, Firenze, 2005.

[3] David G. Luenberger.*Linear and Nonlinear Programming*, Addison Wesley, 1989.

[4] Marco Gaviano, Dimitri E. Kvasov, Daniela Lera, and Yaroslav D. Sergeyev, *Software for generation of classes of test functions with known local and global minima for global optimization*, ACM Transactions on Mathematical Software 24 (2003), no. 4, 469–480.

[5] M. S. Bazaraa, Hanif D. Sherali, C. M. Shetty, *Nonlinear programming: theory and algorithms*, John Wiley and Sons Inc, 2006, New Jersey, 361.

[6] A. Hedar, *Test Functions for Unconstrained Global Optimization*, pagina web(2005), disponibile all'indirizzo `http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO_files/Page364.htm`.

[7] H. Pohlheim, *GEATbx Examples: Examples of Objective Functions*, pagina web(2005), disponibile all'indirizzo `www.geatbx.com/download/GEATbx_ObjFunExpl_v37.pdf`