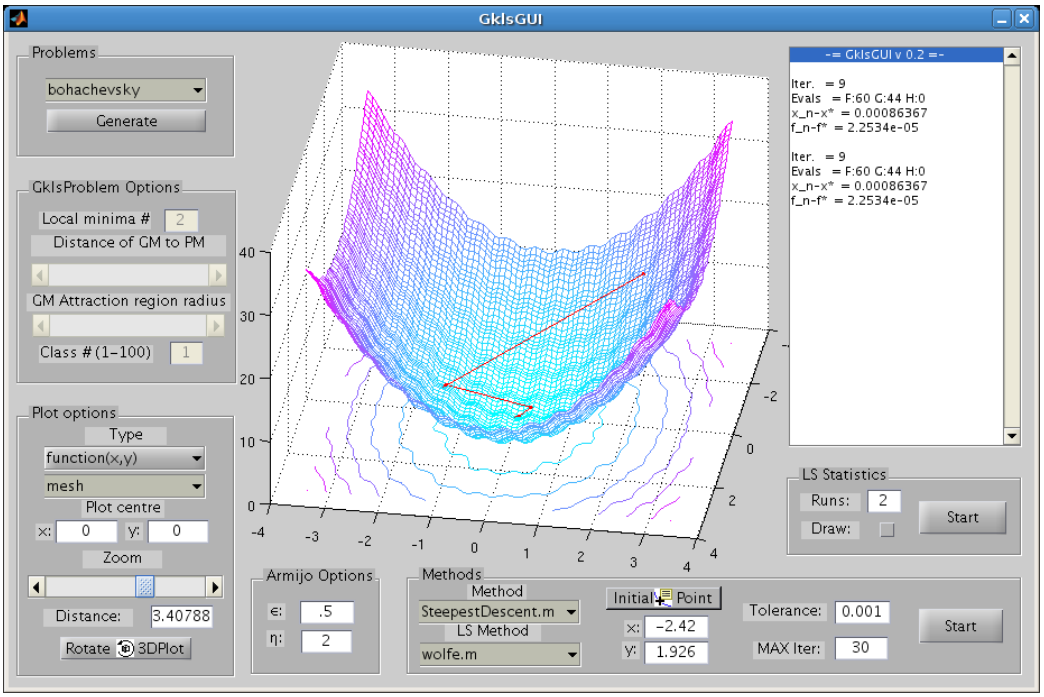
Aggiornamenti e nuove funzionalità del software GklsGUI

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 lavoro è stato l'aggiornamento e l'implementazione di nuove funzionalità di *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 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 7.8* e compatibilità con compilatore C ANSI standard.

Capitolo 2

Metodi unidimensionali per l'ottimizzazione

I metodi di minimizzazione unidimensionale (o line search) sono usati dai 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.

2.1 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.

2.1.1 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.

2.1.2 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.

2.2 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. E' 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.

2.2.1 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.

2.2.2 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, $p = \frac{1+\sqrt{5}}{2} \approx 1.618$.
E'stata implementata anche una versione con TPP per facilitare la convergenza.

2.2.3 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.

2.2.4 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.

2.2.5 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.
E'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.

2.3 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.

2.3.1 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. E' stato comunque inserito un pannello nell'interfaccia grafica per poter modificare i parametri dei metodi inesatti.

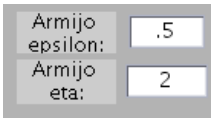


Figura 2.1: Pannello per la modifica dei parametri dei metodi inesatti.

2.3.2 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:

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

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

2.3.3 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.

Capitolo 3

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.

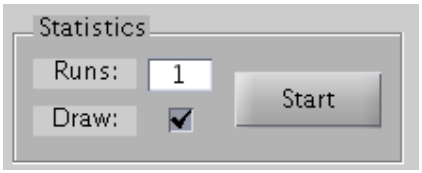


Figura 3.1: Pannello per generare le statistiche.

E' 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.

3.1 Gkls

Per una descrizione dei problemi di tipo Gkls si rimanda a [4].
I risultati degli esperimenti effettuati su questo problema sono in tabella 3.1. E'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 mostrano come al variare del metodo di line search si abbia comunque convergenza, in media da un minimo di 2.84 (Newton) passi a un massimo di 7.28 (Armijo). Fra i metodi più economici in termini di numero di valutazioni appare il metodo delle secanti ed il metodo di Newton (se disposti a calcolare l'Hessiana).

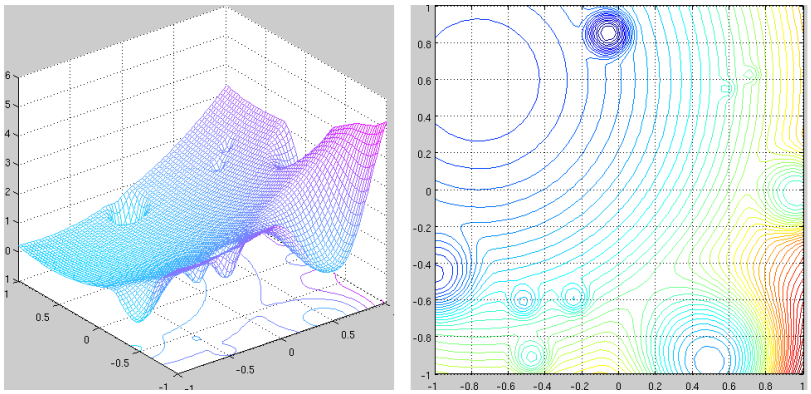


Figura 3.2: Problema di Gkls con 10 minimi locali e classe 1 su cui sono stati effettuati gli esperimenti.

3.2 Rosenbrock

La funzione di Rosenbrock ha un minimo locale nel punto $1, 1$. E' 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: $x^* = (1, 1), f(x^*) = 0$

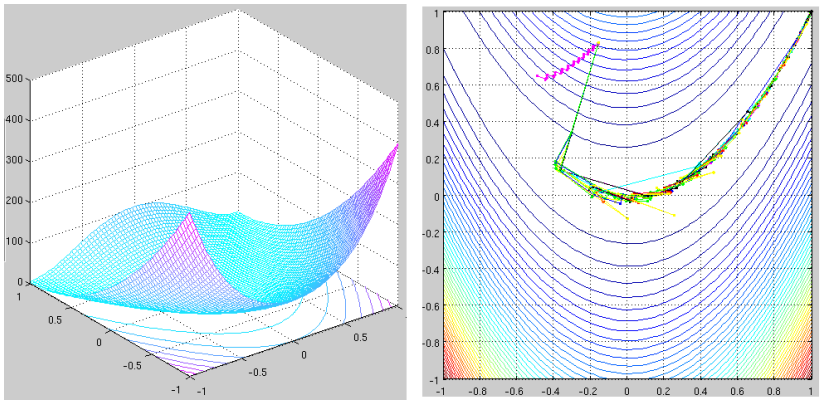


Figura 3.3: Problema di Rosenbrock. Vari metodi LS a confronto.

In tabella 3.2 i risultati degli esperimenti effettuati su questo problema. La ricerca è stata effettuata selezionando punti d'inizio casuali nell'intervallo

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

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, newton-armijo, newton-wolfe, 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 si trovano invece il metodo delle secanti con TPP-tolG e il cubic fit con criterio d'arresto tolleranza su alfa.

3.3 Beale

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

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

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

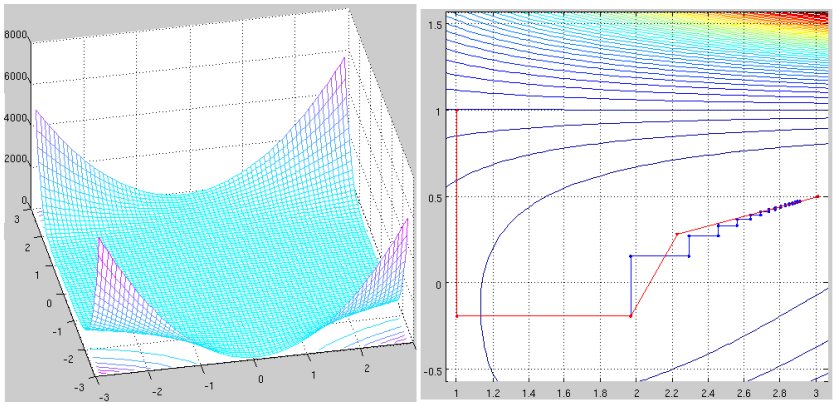


Figura 3.4: Problema di Beale: metodi del Gradiente (in blu) e DFP (in rosso) a confronto. Il primo non riesce a convergere nel massimo numero di iterazioni, il secondo arriva alla convergenza in 7 iterazioni.

Nell'esperimento i cui risultati sono riportati in tabella 3.3 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 ≈ 2.7 per il metodo delle secanti. Questo indica che i metodi fanno fatica a convergere o convergono verso gli altri punti di minimo.

3.4 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 3.4 i risultati degli esperimenti effettuati su questo problema. La ricerca del minimo è effettuata selezionando punti di partenza casuali nell'intervallo $-4 < x < 4, -4 < y < 4$. Per questo problema si evidenziano problemi di convergenza per il metodo di Newton. Anche gli altri metodi comunque in media si arrestano distanti dal minimo globale (come si può vedere dalla colonna distanze dal minimo globale). Il metodo quadratic fit, nelle sue varianti per criterio d'arresto, sembra il più affidabile per raggiungere il minimo globale di questo problema.

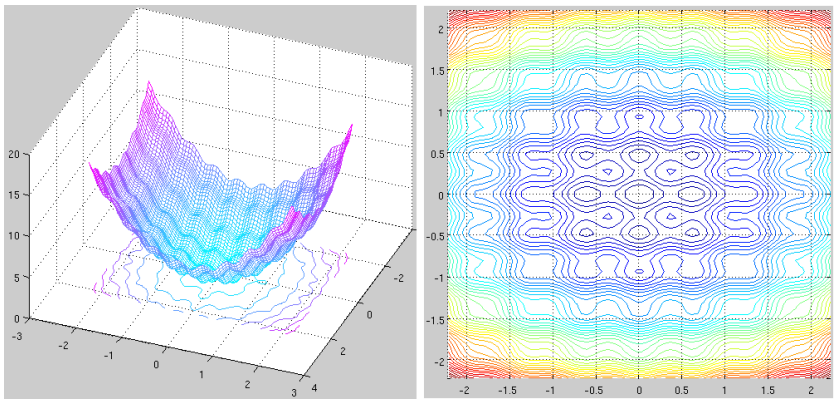


Figura 3.5: La funzione di Bohachevsky

3.5 Booth

E' 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 3.5 i risultati degli esperimenti effettuati su questo problema. La ricerca del minimo è effettuata selezionando punti di partenza casuali a

distanza-∞4 da $(1, 3)$.
Si riesce ad arrivare a convergenza con tutti i metodi LS, solo il metodo delle secanti con TPP e criterio d'arresto di Wolfe sembra avere problemi. Il metodo delle secanti invece è particolarmente efficace.

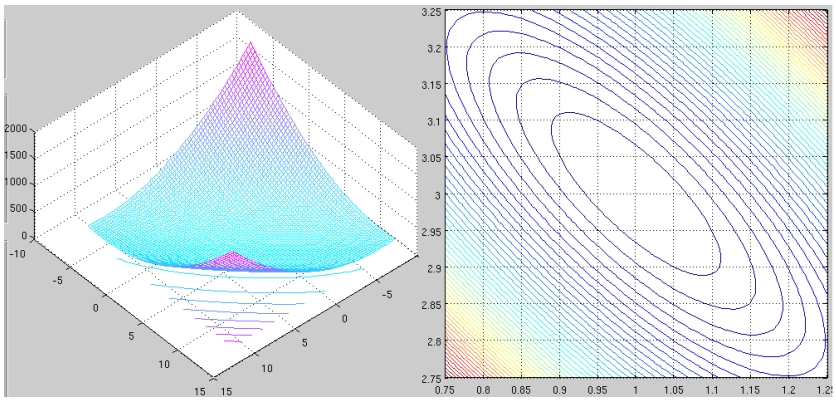


Figura 3.6: Problema di Booth.

3.6 Easom

E' 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 3.6 i risultati degli esperimenti effettuati su questo problema. 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) convergono verso il minimo asintotico.

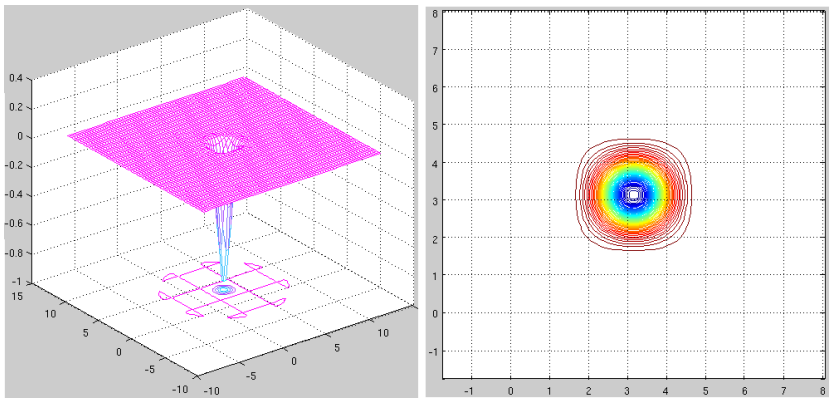


Figura 3.7: La funzione di Easom

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

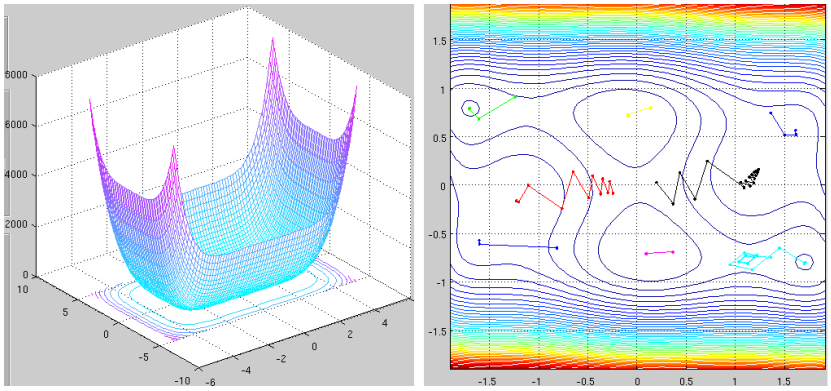


Figura 3.8: Funzione Hump: Convergenza del metodo di Levenberg-Marquardt a differenti minimi locali a seconda del punto iniziale.

In tabella 3.7 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.

3.8 Matyas

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

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

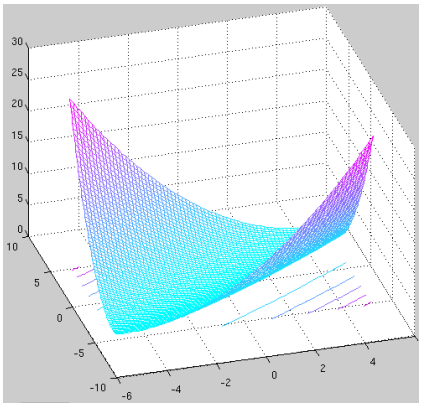


Figura 3.9: La funzione di Matyas

In tabella 3.8 i risultati degli esperimenti effettuati su questo problema. La ricerca del minimo è effettuata selezionando punti di partenza casuali a distanza-∞4 da $(0, 0)$.

Questo problema appare piuttosto facile. La convergenza avviene sempre al variare del metodo LS utilizzato eccetto il caso della secante (senza TPP) che può divergere. Il metodo cubic fit con TPP e criterio d'arresto di Wolfe sembra particolarmente efficace in questo caso.

3.9 Michalewicz

E' 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: $n = 2, f(x^*) = -1.8013$

In tabella 3.9 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*.

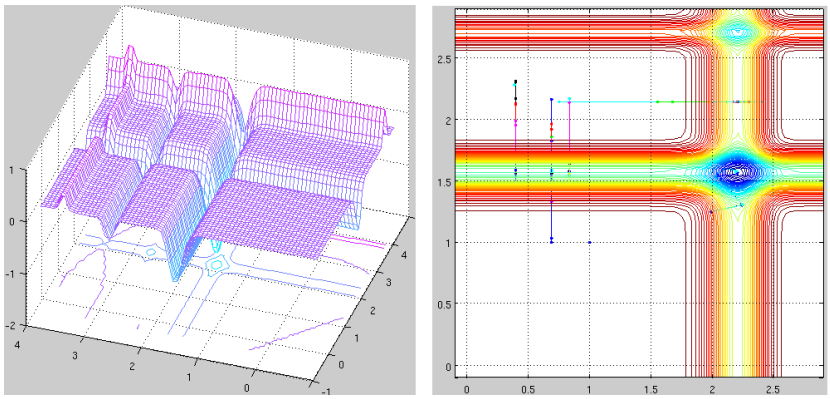


Figura 3.10: La funzione di Michalewicz. I metodi hanno difficoltà a convergere, come si vede dalla figura a destra.

3.10 Michalewicz modificata

E'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: $x^* = (0, 0), f(x^*) = -1.8013$

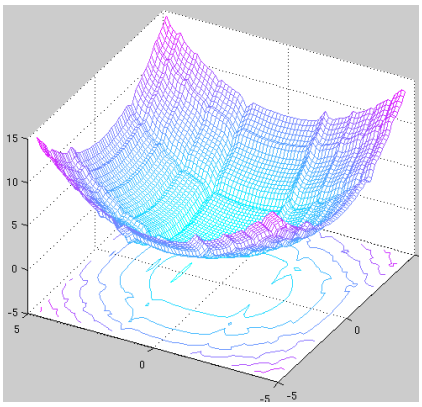


Figura 3.11: La funzione di Michalewicz modificata

In tabella 3.10 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.

3.11 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.

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

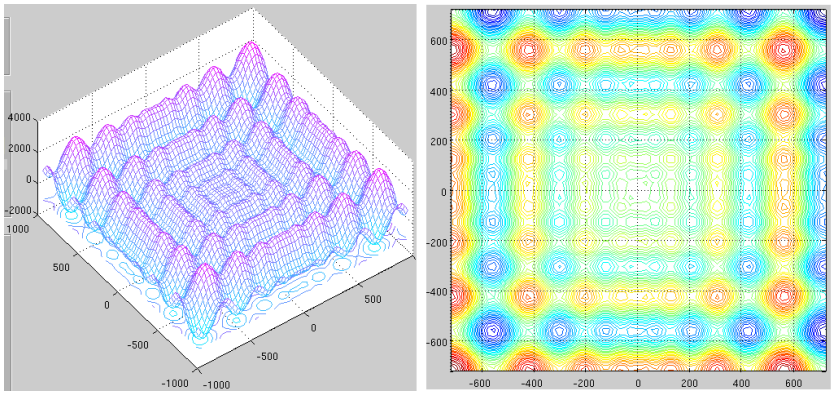


Figura 3.12: La funzione di Schwefel: si notano i numerosi minimi via via migliori allontanandosi dall'origine

In tabella 3.11 i risultati degli esperimenti effettuati su questo problema. La ricerca del minimo è effettuata selezionando punti di partenza casuali a distanza-∞100 da $(0, 0)$.

3.12 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.

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

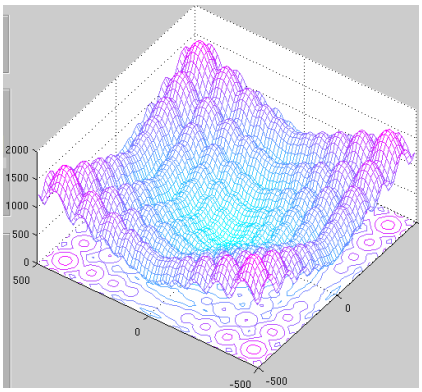


Figura 3.13: Funzione di Schwefel modificata

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

In tabella 3.12 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.

3.13 Sphere

E' 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: $x^* = (0, 0), f(x^*) = 0$

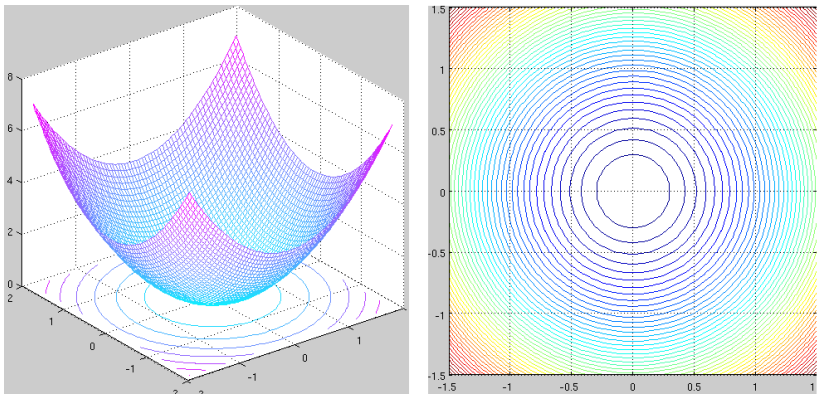


Figura 3.14: Funzione sfera

In tabella 3.13 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 Wolfe convergono in pochi passi.

3.14 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: $x^* = (0, 0), f(x^*) = 0$

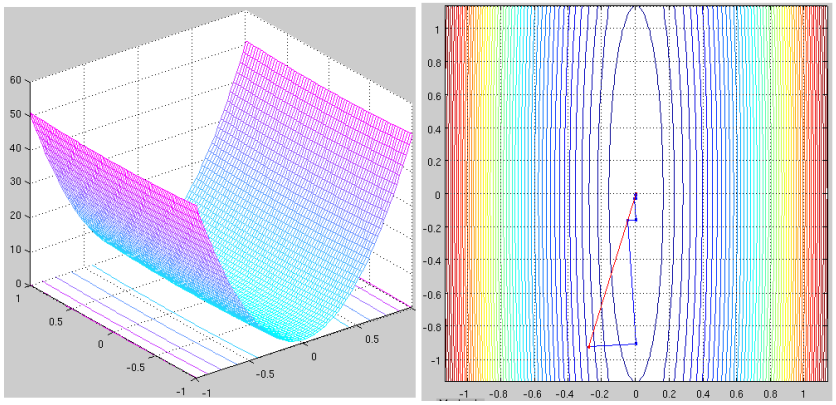


Figura 3.15: Funzione sfera modificata: il metodo del Gradiente (in blu) impiega 12 iterazioni per raggiungere l'intervallo di confidenza, mentre il metodo di Newton (in rosso) lo raggiunge in 2 iterazioni. Entrambi usano il metodo di Newton per la minimizzazione unidimensionale.

In tabella 3.14 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.

Tabella 3.1: Mean results for problem gkls (method DFP), centered on x:0 y:0 distance:1 number of runs:50

LS Method	Iter.	Ev.F	Ev.G	Ev.H	$x_n - x^*$	$f_n - f^*$
armijo	7.28	35.44	7.28	0	0.00020447	6.8508e-08
aurea	3.06	44.38	3.06	0	0.00010548	2.361e-08
cubicfitTPP-armijo	3.44	84.2	42.92	0	1.3196e-05	3.7469e-09
cubicfitTPP-goldstein	3.06	72.52	68.98	0	1.1834e-05	3.1866e-09
cubicfitTPP-tolAlpha	3.06	28.94	27.46	0	1.2339e-05	3.4675e-09
cubicfitTPP-tolG	3.56	21.34	16.98	0	1.5496e-05	4.2264e-09
cubicfitTPP-wolfe	3.54	46.68	42.2	0	1.3021e-05	3.7386e-09
fibonacci	3.06	44.38	3.06	0	0.00010548	2.361e-08
goldstein	4.3	43.46	4.3	0	1.9222e-05	5.2391e-09
newton-armijo	3.16	50.64	26.32	23.16	0.0082322	0.01776
newton-goldstein	2.84	58.88	58.04	55.2	0.0082341	0.01776
newton-tolAlpha	2.9	1.9	9.58	6.68	0.008233	0.01776
newton-tolG	2.84	1.84	58.04	55.2	0.0082341	0.01776
newton-wolfe	3.02	17.2	16.18	13.16	0.0082318	0.01776
quadfitTPP-armijo	4	76.72	4	0	1.4254e-05	4.4822e-09
quadfitTPP-goldstein	3.4	86.5	3.4	0	1.0059e-05	2.254e-09
quadfitTPP-tolAlpha	3.4	27.82	3.4	0	1.0059e-05	2.254e-09
quadfitTPP-tolG	3.4	55.12	48.82	0	1.5374e-05	4.3874e-09
quadfitTPP-wolfe	3.58	48.98	37	0	9.9196e-06	2.553e-09
secant	4.1	3.1	11.16	0	6.2084e-06	1.5248e-09
secantTPP-armijo	3.58	87.34	42.62	0	9.4066e-06	2.4906e-09
secantTPP-goldstein	3.22	60.02	52.34	0	9.5149e-06	2.7886e-09
secantTPP-tolAlpha	3.24	37.24	32.08	0	9.3904e-06	2.7803e-09
secantTPP-tolG	3.22	48.58	43.12	0	1.3405e-05	3.7467e-09
secantTPP-wolfe	4.3	57.22	47.18	0	1.5338e-05	4.2427e-09
wolfe	5.72	55.44	43.92	0	6.2721e-05	2.3529e-08

Tabella 3.2: Mean results for problem rosenbrock (method DFP), centered on x:0 y:0 distance:1 number of runs:50

LS Method	Iter.	Ev.F	Ev.G	Ev.H	$x_n - x^*$	$f_n - f^*$
armijo	26.74	180.48	26.74	0	0.0093465	0.00029029
aurea	16.86	349.7	16.86	0	0.00029645	5.6336e-08
cubicfitTPP-armijo	20.3	444.5	211.86	0	0.0094215	0.00037861
cubicfitTPP-goldstein	15.66	535.36	484.78	0	0.00055554	1.2356e-07
cubicfitTPP-tolAlpha	15.66	202.94	167	0	0.00053087	1.0884e-07
cubicfitTPP-tolG	16.24	135.6	81.8	0	0.00078942	1.7104e-07
cubicfitTPP-wolfe	20.06	263.88	204.96	0	0.0061897	0.00019899
fibonacci	16.86	349.78	16.86	0	0.00024155	4.1242e-08
goldstein	26.8	336.08	26.8	0	0.041082	0.003589
newton-armijo	24.88	166.64	84.32	59.44	0.0046413	0.00020073
newton-goldstein	16.82	506.24	491.42	474.6	0.0004911	8.7598e-08
newton-tolAlpha	17.38	16.38	88.38	71	0.00043023	9.4957e-08
newton-tolG	16.82	15.82	491.42	474.6	0.0004911	8.7598e-08
newton-wolfe	25.26	104.74	81.48	56.22	0.010591	0.00062759
quadfitTPP-armijo	23.46	684.06	23.46	0	0.032182	0.011256
quadfitTPP-goldstein	18.92	664.92	18.92	0	0.0001955	3.3141e-08
quadfitTPP-tolAlpha	18.92	280.62	18.92	0	0.0001955	3.3142e-08
quadfitTPP-tolG	18.8	613.46	541.14	0	0.00036551	8.4741e-08
quadfitTPP-wolfe	21.46	417.34	291.44	0	0.002045	9.088e-06
secant	28.42	27.42	89.44	0	0.884	0.65492
secantTPP-armijo	24.3	467.8	205.04	0	0.0025958	1.3314e-05
secantTPP-goldstein	15.78	518.92	454.36	0	0.00048689	1.0374e-07
secantTPP-tolAlpha	16.02	281.76	231.36	0	0.0045295	0.00021957
secantTPP-tolG	15.78	464.18	414.4	0	0.00048689	1.0374e-07
secantTPP-wolfe	28.22	311	138.02	0	0.90141	23.108
wolfe	28.44	345.9	250.4	0	0.046116	0.024528

Tabella 3.3: Mean results for problem beale (method DFP), centered on x:0 y:0 distance:4.5 number of runs:50

LS Method	Iter.	Ev.F	Ev.G	Ev.H	$x_n - x^*$	$f_n - f^*$
armijo	26.02	210.6	26.02	0	5.8699	1.3767
aurea	19.62	453.34	19.62	0	6.7104	0.77574
cubicfitTPP-armijo	21.62	461.08	202.58	0	10.356	1.0482
cubicfitTPP-goldstein	21.06	774.4	662.98	0	15.775	1.0079
cubicfitTPP-tolAlpha	22.34	358.62	260.76	0	12.376	0.75437
cubicfitTPP-tolG	22.62	214.66	114.84	0	9.4594	1.0405
cubicfitTPP-wolfe	21.58	300.22	203.78	0	10.145	1.0483
fibonacci	19.62	453.36	19.62	0	6.6779	0.77647
goldstein	24	364.7	24	0	8.5706	1.4297
newton-armijo	21.12	221.88	111.94	90.82	4.1185	2.364
newton-goldstein	17.92	541.44	525.52	507.6	14.046	2.051
newton-tolAlpha	20.38	19.38	129.96	109.58	9.9006	40.884
newton-tolG	17.92	16.92	525.52	507.6	14.046	2.051
newton-wolfe	22.52	110.82	90.3	67.78	3.9799	2.2012
quadfitTPP-armijo	20.22	476.86	20.22	0	5.0704	0.55192
quadfitTPP-goldstein	18.24	673.46	18.24	0	7.3504	0.6304
quadfitTPP-tolAlpha	20.06	390.96	20.06	0	7.5957	0.84705
quadfitTPP-tolG	18.22	637.38	533.88	0	7.3505	0.6304
quadfitTPP-wolfe	19.92	339.52	199.54	0	4.9276	0.69762
secant	20.3	19.3	85.82	0	2.718	7.7573
secantTPP-armijo	20.32	491.02	207.04	0	4.1225	0.48599
secantTPP-goldstein	19.82	706.66	579.42	0	11.103	0.79934
secantTPP-tolAlpha	20.6	470.04	359.32	0	10.239	0.84401
secantTPP-tolG	19.82	657.78	549.36	0	11.103	0.79934
secantTPP-wolfe	28.9	453.6	97.68	0	3.9194	7732.7
wolfe	25.28	338.54	218.94	0	6.2316	1.4785

Tabella 3.4: Mean results for problem bohachevsky (method SteepestDescent), centered on x:0 y:0 distance:4 number of runs:50

LS Method	Iter.	Ev.F	Ev.G	Ev.H	$x_n - x^*$	$f_n - f^*$
armijo	7.74	29.9	7.74	0	0.97737	1.7205
aurea	9.3	182.54	9.3	0	0.48178	0.5947
cubicfitTPP-armijo	7.38	221.58	106.86	0	0.52433	0.65415
cubicfitTPP-goldstein	7.2	227.9	205.6	0	0.51984	0.64885
cubicfitTPP-tolAlpha	7.2	112.18	96.08	0	0.51984	0.64885
cubicfitTPP-tolG	7.2	98.82	76.42	0	0.51988	0.64885
cubicfitTPP-wolfe	7.54	126.82	104.04	0	0.49379	0.62165
fibonacci	9.22	181.72	9.22	0	0.46934	0.58654
goldstein	7.94	67.62	7.94	0	0.93233	1.4984
newton-armijo	9.96	440.28	221.14	211.18	7.0601e+07	2.7996e+17
newton-goldstein	10.06	289.92	281.86	271.8	265.93	1.8641e+06
newton-tolAlpha	11.38	10.38	240.68	229.3	1.6553e+09	1.4943e+20
newton-tolG	10.06	9.06	280.66	270.6	265.93	1.8641e+06
newton-wolfe	10.72	223.52	214.8	204.08	7.0601e+07	2.7996e+17
quadfitTPP-armijo	7.56	293.3	7.56	0	0.51017	0.59325
quadfitTPP-goldstein	7.32	239.36	7.32	0	0.47143	0.52946
quadfitTPP-tolAlpha	7.32	62.86	7.32	0	0.47143	0.52946
quadfitTPP-tolG	7.32	227.74	197.92	0	0.47143	0.52946
quadfitTPP-wolfe	7.36	167	124.18	0	0.50182	0.59943
secant	18.02	17.02	64.96	0	1.8392	20.241
secantTPP-armijo	8.2	335.5	160.42	0	0.59883	0.76056
secantTPP-goldstein	7.7	236.42	207.84	0	0.57576	0.71079
secantTPP-tolAlpha	7.7	189.36	167.48	0	0.57575	0.71079
secantTPP-tolG	7.7	226.54	204.66	0	0.57576	0.71079
secantTPP-wolfe	14.02	225.04	159.4	0	0.64127	0.83712
wolfe	7.3	47.02	38.32	0	0.92593	1.652

Tabella 3.5: Mean results for problem booth (method LevMarq), centered on x:1 y:3 distance:4 number of runs:50

LS Method	Iter.	Ev.F	Ev.G	Ev.H	$x_n - x^*$	$f_n - f^*$
armijo	7.9	48.3	7.9	7.9	5.0495e-05	1.1897e-08
aurea	2.7	44	2.7	2.7	2.592e-05	6.05e-09
cubicfitTPP-armijo	2.04	44.5	20.88	2.04	1.3126e-12	7.6894e-22
cubicfitTPP-goldstein	2	39.78	34	2	7.6426e-16	1.6093e-30
cubicfitTPP-tolAlpha	2	14.72	9.94	2	7.6426e-16	1.6093e-30
cubicfitTPP-tolG	2	10.78	5	2	9.7503e-16	4.2598e-30
cubicfitTPP-wolfe	2.04	26.7	20.88	2.04	1.3126e-12	7.6894e-22
fibonacci	2.7	44	2.7	2.7	2.592e-05	6.0501e-09
goldstein	3	34	3	3	1.7482e-06	2.2813e-11
newton-armijo	2	39	20.5	20.5	7.962e-16	1.6093e-30
newton-goldstein	2	32	32	32	7.8207e-16	1.4515e-30
newton-tolAlpha	2	1	4	4	7.8207e-16	1.4515e-30
newton-tolG	2	1	32	32	7.8207e-16	1.4515e-30
newton-wolfe	2	20.5	20.5	20.5	7.962e-16	1.6093e-30
quadfitTPP-armijo	2.04	34.02	2.04	2.04	1.3153e-12	7.6894e-22
quadfitTPP-goldstein	2	40.78	2	2	7.7795e-16	1.5777e-30
quadfitTPP-tolAlpha	2	16.12	2	2	7.7795e-16	1.5777e-30
quadfitTPP-tolG	2	10.78	4	2	5.736e-15	1.5763e-27
quadfitTPP-wolfe	2	21.8	13.02	2	3.5603e-15	9.9911e-28
secant	2	1	4	2	2.4784e-14	1.1801e-26
secantTPP-armijo	2.04	17.46	6.84	2.04	1.3126e-12	7.6894e-22
secantTPP-goldstein	2	15.28	8.5	2	8.1741e-16	1.4988e-30
secantTPP-tolAlpha	2	13.2	7.42	2	8.1741e-16	1.4988e-30
secantTPP-tolG	2	12.96	7.18	2	8.5307e-16	1.6251e-30
secantTPP-wolfe	22.96	230.3	66.46	22.96	1.1195	30.57
wolfe	3.48	28.56	19.64	3.48	1.4649e-05	4.9979e-09

Tabella 3.6: Mean results for problem easom (method LevMarq), centered on x:3.1415 y:3.1415 distance:4 number of runs:50

LS Method	Iter.	Ev.F	Ev.G	Ev.H	$x_n - x^*$	$f_n - f^*$
armijo	2.32	9.52	2.32	2.32	2.9596	0.80002
aurea	1.62	235.76	1.62	1.62	NaN	NaN
cubicfitTPP-armijo	2.36	855.68	30.02	2.36	2.9531	0.80003
cubicfitTPP-goldstein	2.28	842.52	43.24	2.28	2.9531	0.80003
cubicfitTPP-tolAlpha	2.28	810.56	12.56	2.28	2.9531	0.80003
cubicfitTPP-tolG	2.28	809.42	10.14	2.28	2.9379	0.80003
cubicfitTPP-wolfe	2.34	827.54	28.2	2.34	2.9531	0.80003
fibonacci	1.62	241.7	1.62	1.62	NaN	NaN
goldstein	2.14	19.62	2.14	2.14	2.98	0.80001
newton-armijo	1.86	31.36	16.68	16.68	3.5259	0.98056
newton-goldstein	1.76	24.32	24.56	24.56	3.8079	0.98054
newton-tolAlpha	1.76	0.76	8.54	8.54	3.8079	0.98054
newton-tolG	1.76	0.76	24.56	24.56	3.8079	0.98054
newton-wolfe	2.08	5.46	5.38	5.38	4.5703	0.98056
quadfitTPP-armijo	2.34	856.9	2.34	2.34	2.9537	0.80003
quadfitTPP-goldstein	2.34	845.9	2.34	2.34	2.9537	0.80003
quadfitTPP-tolAlpha	2.34	807.04	2.34	2.34	2.9537	0.80003
quadfitTPP-tolG	2.34	832	31.32	2.34	2.9379	0.80003
quadfitTPP-wolfe	2.32	829.6	26.3	2.32	2.9537	0.80003
secant	1.7	0.7	3.04	1.7	3.9697	0.9202
secantTPP-armijo	2.46	861.14	32.06	2.46	2.9553	0.80003
secantTPP-goldstein	2.4	845.22	44.4	2.4	2.9553	0.80003
secantTPP-tolAlpha	2.4	817.78	18.36	2.4	2.9553	0.80003
secantTPP-tolG	2.26	811.54	12.28	2.26	3.4355	0.86003
secantTPP-wolfe	2.46	809.96	9.02	2.46	2.9379	0.80003
wolfe	2.9	28.52	21.66	2.9	2.9937	0.80001

Tabella 3.7: Mean results for problem hump (method DFP), centered on x:0 y:0 distance:2 number of runs:50

LS Method	Iter.	Ev.F	Ev.G	Ev.H	$x_n - x^*$	$f_n - f^*$
armijo	12.02	57.12	12.02	0	0.69213	0.58
aurea	5.56	100.44	5.56	0	0.71106	0.35911
cubicfitTPP-armijo	6.4	163.04	78.88	0	0.67549	0.39835
cubicfitTPP-goldstein	5.36	157.2	144.88	0	0.74513	0.42183
cubicfitTPP-tolAlpha	5.36	62.84	54.88	0	0.74513	0.42183
cubicfitTPP-tolG	6.66	43.66	28.68	0	0.74512	0.42183
cubicfitTPP-wolfe	6.32	95.28	81.2	0	0.68049	0.38918
fibonacci	5.56	100.44	5.56	0	0.71106	0.35911
goldstein	6.92	69.48	6.92	0	0.71977	0.59108
newton-armijo	7.9	78.4	40.2	32.3	1.1564	1.6612
newton-goldstein	5	128	125	120	1.1553	1.5145
newton-tolAlpha	4.96	3.96	19.74	14.78	1.1434	1.5297
newton-tolG	5	4	125	120	1.1553	1.5145
newton-wolfe	8.42	26.7	20.28	11.86	1.1221	1.5863
quadfitTPP-armijo	7.1	165.94	7.1	0	0.68397	0.48197
quadfitTPP-goldstein	6.58	206.9	6.58	0	0.68963	0.53227
quadfitTPP-tolAlpha	6.58	65.96	6.58	0	0.68963	0.53227
quadfitTPP-tolG	6.58	172.64	150.88	0	0.68963	0.53227
quadfitTPP-wolfe	6.7	101.42	66.68	0	0.68789	0.48588
secant	8.48	7.48	28.84	0	0.7703	0.82676
secantTPP-armijo	8	156.12	69.74	0	NaN	NaN
secantTPP-goldstein	6.12	176.66	156.82	0	0.64244	0.54077
secantTPP-tolAlpha	6.12	89.26	74.52	0	0.64244	0.54077
secantTPP-tolG	6.12	146.14	131.42	0	0.64244	0.54077
secantTPP-wolfe	14.66	163.44	80.28	0	0.6009	0.62957
wolfe	11.9	113.7	94.22	0	0.81673	0.64005

Tabella 3.8: Mean results for problem matyas (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	18.12	139.9	18.12	0	0.28371	0.0019807
aurea	7.96	187.64	7.96	0	0.026859	6.4431e-05
cubicfitTPP-armijo	5.82	113.1	54.16	0	6.5063e-06	1.5262e-11
cubicfitTPP-goldstein	7.92	249.98	229.36	0	0.026934	6.6253e-05
cubicfitTPP-tolAlpha	7.92	119.84	106.14	0	0.026934	6.6253e-05
cubicfitTPP-tolG	14.92	116.7	45.62	0	0.16321	0.00072362
cubicfitTPP-wolfe	5.82	69.58	54.16	0	6.5063e-06	1.5262e-11
fibonacci	7.96	187.64	7.96	0	0.026859	6.4431e-05
goldstein	8.12	141.2	8.12	0	0.038067	8.5622e-05
newton-armijo	7.92	187.64	94.82	86.9	0.026934	6.6253e-05
newton-goldstein	7.92	221.44	215.52	207.6	0.026934	6.6253e-05
newton-tolAlpha	7.92	6.92	21.76	13.84	0.026934	6.6253e-05
newton-tolG	7.92	6.92	204.12	196.2	0.029039	6.6782e-05
newton-wolfe	7.92	100.74	94.82	86.9	0.026934	6.6253e-05
quadfitTPP-armijo	5.82	103.12	5.82	0	6.5063e-06	1.5262e-11
quadfitTPP-goldstein	7.92	256.9	7.92	0	0.026934	6.6253e-05
quadfitTPP-tolAlpha	7.92	102.38	7.92	0	0.026934	6.6253e-05
quadfitTPP-tolG	7.92	219.44	191.9	0	0.028934	6.6736e-05
quadfitTPP-wolfe	7.92	121.86	80.48	0	0.026934	6.6253e-05
secant	5.98	4.98	21.54	0	Inf	Inf
secantTPP-armijo	5.82	88.48	39.44	0	6.5063e-06	1.5262e-11
secantTPP-goldstein	7.92	162.4	134.86	0	0.026934	6.6253e-05
secantTPP-tolAlpha	7.92	109.9	89.28	0	0.026934	6.6253e-05
secantTPP-tolG	7.92	137.96	117.34	0	0.028635	6.6627e-05
secantTPP-wolfe	27.26	249.94	79.78	0	0.58423	0.0091893
wolfe	6.28	76.26	44.04	0	0.068855	0.0011459

Tabella 3.9: 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	7.34	41.6	7.34	0	1.1705	0.95998
aurea	3.42	59.08	3.42	0	1.1601	0.9464
cubicfitTPP-armijo	4.68	118.56	57.68	0	1.1601	0.9464
cubicfitTPP-goldstein	3.5	90.24	83.5	0	1.1601	0.9464
cubicfitTPP-tolAlpha	3.5	54.3	50.06	0	1.1601	0.9464
cubicfitTPP-tolG	5.22	83.32	43.9	0	1.2088	1.0985
cubicfitTPP-wolfe	4.56	64.44	54.84	0	1.1601	0.9464
fibonacci	3.4	58.46	3.4	0	1.1601	0.9464
goldstein	5.56	65.04	5.56	0	1.1863	0.97814
newton-armijo	2.6	55.52	28.76	26.16	467.69	1.6812
newton-goldstein	2.2	38.4	38.2	36	467.71	1.6612
newton-tolAlpha	2.28	1.28	28.92	26.64	467.98	1.6412
newton-tolG	2.2	1.2	38.2	36	467.71	1.6612
newton-wolfe	3.04	9.52	8.48	5.44	1.6395	1.6809
quadfitTPP-armijo	4.36	89.44	4.36	0	1.1636	0.95604
quadfitTPP-goldstein	4	112.48	4	0	1.1602	0.9464
quadfitTPP-tolAlpha	4	29.76	4	0	1.1602	0.9464
quadfitTPP-tolG	4	84.5	72.02	0	1.1602	0.9464
quadfitTPP-wolfe	4.46	60.14	40.4	0	1.1599	0.9464
secant	3.56	2.56	10.14	0	1.3634	1.6851
secantTPP-armijo	5.1	153.1	69.98	0	1.1888	1.0765
secantTPP-goldstein	4.62	133.7	112.78	0	1.1888	1.0765
secantTPP-tolAlpha	4.62	105.98	88.68	0	1.1888	1.0765
secantTPP-tolG	3.92	73.24	64.48	0	1.2762	1.2675
secantTPP-wolfe	6.24	128.38	75.72	0	1.2117	1.1554
wolfe	6.5	60.24	45.52	0	1.2059	0.99417

Tabella 3.10: Mean results for problem michalewiczMod (method SteepestDescent), centered on x:0 y:0 distance:5 number of runs:50

LS Method	Iter.	Ev.F	Ev.G	Ev.H	$x_n - x^*$	$f_n - f^*$
armijo	11.2	56.38	11.2	0	1.3247	0.90165
aurea	6.54	134.98	6.54	0	0.27898	0.070671
cubicfitTPP-armijo	7.2	135.64	66.3	0	0.55126	0.41452
cubicfitTPP-goldstein	5.88	148.78	136.44	0	0.56527	0.43776
cubicfitTPP-tolAlpha	5.88	62.38	54.9	0	0.56527	0.43776
cubicfitTPP-tolG	5.66	51.32	39.22	0	0.5346	0.43338
cubicfitTPP-wolfe	7.52	81.9	64.58	0	0.56428	0.44783
fibonacci	6.54	134.98	6.54	0	0.27898	0.070671
goldstein	8.56	90.44	8.56	0	1.0809	1.096
newton-armijo	10.48	255.2	128.6	118.12	3.4759	49.169
newton-goldstein	10.72	297.24	288.52	277.8	4.907	65.957
newton-tolAlpha	10.16	9.16	37.56	27.4	3.1753	40.681
newton-tolG	10.72	9.72	274.72	264	4.907	65.957
newton-wolfe	11.54	68.88	59.34	47.8	3.3736	43.392
quadfitTPP-armijo	7.48	159.86	7.48	0	0.23543	0.094744
quadfitTPP-goldstein	7.12	196.32	7.12	0	0.26367	0.096649
quadfitTPP-tolAlpha	7.12	52.54	7.12	0	0.26367	0.096649
quadfitTPP-tolG	7.12	149.36	125.62	0	0.26371	0.096649
quadfitTPP-wolfe	7.22	126.28	92.8	0	0.29596	0.10009
secant	6.3	5.3	18.6	0	13.204	2510.4
secantTPP-armijo	7.68	168.5	78.98	0	0.39139	0.17601
secantTPP-goldstein	8.02	187.78	159.26	0	0.53686	0.22996
secantTPP-tolAlpha	8.02	151.38	129.92	0	0.53686	0.22996
secantTPP-tolG	8.02	176.04	153.9	0	0.53695	0.22996
secantTPP-wolfe	10.94	168.56	128.24	0	0.88755	0.52187
wolfe	7.64	60.38	41.96	0	0.90723	0.88887

Tabella 3.11: Mean results for problem schwefel (method SteepestDescent), centered on x:0 y:0 distance:100 number of runs:50

LS Method	Iter.	Ev.F	Ev.G	Ev.H
armijo	7.36	57.36	7.36	0
aurea	6	140.86	6	0
cubicfitTPP-armijo	6.12	139.44	65.3	0
cubicfitTPP-goldstein	5.84	179.52	160.72	0
cubicfitTPP-tolAlpha	5.82	98.2	84.26	0
cubicfitTPP-tolG	8.16	60.66	35.18	0
cubicfitTPP-wolfe	6.16	86.3	66.3	0
fibonacci	6	140.64	6	0
goldstein	6.62	120.42	6.62	0
newton-armijo	8.5	269.6	135.8	127.3
newton-goldstein	8.64	244.48	237.84	229.2
newton-tolAlpha	8.8	7.8	41.46	32.66
newton-tolG	8.36	7.36	202.76	194.4
newton-wolfe	7.28	39.7	34.42	27.14
quadfitTPP-armijo	6	148.76	6	0
quadfitTPP-goldstein	6	191.4	6	0
quadfitTPP-tolAlpha	6	78.42	6	0
quadfitTPP-tolG	6.76	121.14	92.76	0
quadfitTPP-wolfe	6.02	95.62	60.46	0
secant	8.06	7.06	23.96	0
secantTPP-armijo	6.72	197.42	88.68	0
secantTPP-goldstein	6.48	201.8	170.3	0
secantTPP-tolAlpha	6.56	122.9	96.88	0
secantTPP-tolG	9.5	157.72	122.04	0
secantTPP-wolfe	28.58	314.32	92.06	0
wolfe	7.26	104.58	67.2	0

Tabella 3.12: Mean results for problem schwefelMod (method SteepestDescent), centered on x:0 y:0 distance:100 number of runs:50

LS Method	Iter.	Ev.F	Ev.G	Ev.H	$x_n - x^*$	$f_n - f^*$
armijo	10.22	82.2	10.22	0	69.009	103.53
aurea	8.08	172.66	8.08	0	54.153	78.548
cubicfitTPP-armijo	12	530.44	249.88	0	48.909	67.283
cubicfitTPP-goldstein	10.96	397.14	329.68	0	48.553	65.348
cubicfitTPP-tolAlpha	11.5	354.86	311.24	0	48.356	65.15
cubicfitTPP-tolG	10.92	375.46	308.5	0	48.553	65.348
cubicfitTPP-wolfe	12.1	315.58	249.26	0	46.509	64.13
fibonacci	8.08	172.58	8.08	0	54.058	78.517
goldstein	14.26	159.62	14.26	0	68.594	98.257
newton-armijo	7.62	384	193	185.38	216.58	542.47
newton-goldstein	8.06	225.92	219.86	211.8	216.72	542.14
newton-tolAlpha	7.94	6.94	65.78	57.84	216.74	542.3
newton-tolG	7.52	6.52	169.52	162	216.72	542.14
newton-wolfe	8.44	135.66	129.22	120.78	220.97	534.52
quadfitTPP-armijo	11.78	425.1	11.78	0	52.867	73.074
quadfitTPP-goldstein	9.82	362.94	9.82	0	52.871	73.678
quadfitTPP-tolAlpha	9.74	138.8	9.74	0	51.795	71.985
quadfitTPP-tolG	9.82	354.12	283.24	0	52.871	73.678
quadfitTPP-wolfe	10.9	276.86	186.1	0	52.628	72.708
secant	9.54	8.54	24.98	0	540.56	1434.8
secantTPP-armijo	14.4	643.7	298.86	0	31.572	42.238
secantTPP-goldstein	15.36	530.28	446.16	0	22.186	28.81
secantTPP-tolAlpha	15.2	477.08	408.94	0	22.418	29.051
secantTPP-tolG	15.36	515.92	446.16	0	22.186	28.81
secantTPP-wolfe	25.8	344.54	163.96	0	59.644	96.987
wolfe	13.42	117.8	61	0	69.775	101.96

Tabella 3.13: Mean results for problem sphere (method SteepestDescent), centered on x:0 y:0 distance:100 number of runs:50

LS Method	Iter.	Ev.F	Ev.G	Ev.H	$x_n - x^*$	$f_n - f^*$
armijo	8.16	42.96	8.16	0	0.00079648	7.4695e-07
aurea	4.08	77.86	4.08	0	0	0
cubicfitTPP-armijo	4	33.72	11.96	0	0	0
cubicfitTPP-goldstein	4	58.86	43.68	0	0	0
cubicfitTPP-tolAlpha	4	24.44	12.26	0	0	0
cubicfitTPP-tolG	4	23.22	8.04	0	0	0
cubicfitTPP-wolfe	4	27.14	11.96	0	0	0
fibonacci	4.08	77.86	4.08	0	0	0
goldstein	5	60	5	0	4.5582e-10	2.3006e-19
newton-armijo	4	8	5	1	0	0
newton-goldstein	4	36	34	30	0	0
newton-tolAlpha	4	3	6	2	0	0
newton-tolG	4	3	34	30	0	0
newton-wolfe	4	7	5	1	0	0
quadfitTPP-armijo	4	29.86	4	0	0	0
quadfitTPP-goldstein	4	56.8	4	0	0	0
quadfitTPP-tolAlpha	4	23.02	4	0	0	0
quadfitTPP-tolG	4	23.18	7	0	0	0
quadfitTPP-wolfe	4	26.8	6.7	0	0	0
secant	4	3	8	0	6.9409e-11	1.0221e-20
secantTPP-armijo	4	30.7	9.42	0	8.0904e-15	2.4104e-28
secantTPP-goldstein	4	26.92	10.18	0	2.6241e-15	6.7984e-29
secantTPP-tolAlpha	4	23.88	10.14	0	2.6241e-15	6.7984e-29
secantTPP-tolG	4	23.98	10.08	0	3.3124e-15	8.0164e-29
secantTPP-wolfe	31	545.4	91	0	76.113	6428.5
wolfe	4	1048	20	0	NaN	NaN

Tabella 3.14: Mean results for problem sphereMod (method SteepestDescent), centered on x:0 y:0 distance:100 number of runs:50

LS Method	Iter.	Ev.F	Ev.G	Ev.H	$x_n - x^*$	$f_n - f^*$
armijo	31	137.12	31	0	7.974	102.72
aurea	30.34	629.3	30.34	0	4.7274	68.722
cubicfitTPP-armijo	11.38	363.62	159.62	0	0.41309	7.7759
cubicfitTPP-goldstein	9.36	334.24	276.88	0	0.38641	7.7545
cubicfitTPP-tolAlpha	9.36	167.08	118.08	0	0.38641	7.7545
cubicfitTPP-tolG	9.36	90.42	32.86	0	0.386	7.7545
cubicfitTPP-wolfe	11.38	225.76	159.62	0	0.41309	7.7759
fibonacci	30.2	622.28	30.2	0	4.6893	68.449
goldstein	26.84	240.56	26.84	0	7.9436	135.48
newton-armijo	9.36	189.52	95.76	86.4	0.38641	7.7545
newton-goldstein	9.36	267.52	260.16	250.8	0.38641	7.7545
newton-tolAlpha	9.36	8.36	26.08	16.72	0.38641	7.7545
newton-tolG	9.36	8.36	253.56	244.2	0.38645	7.7545
newton-wolfe	9.36	103.12	95.76	86.4	0.38641	7.7545
quadfitTPP-armijo	11.38	372.2	11.38	0	0.41309	7.7759
quadfitTPP-goldstein	9.36	342.6	9.36	0	0.38641	7.7545
quadfitTPP-tolAlpha	9.36	116.56	9.36	0	0.38641	7.7545
quadfitTPP-tolG	9.36	308.96	243.24	0	0.38641	7.7545
quadfitTPP-wolfe	9.36	200.92	118.48	0	0.38641	7.7545
secant	8.96	7.96	33.1	0	Inf	Inf
secantTPP-armijo	11.38	266.84	106.04	0	0.41309	7.7759
secantTPP-goldstein	9.36	213.3	147.58	0	0.38641	7.7545
secantTPP-tolAlpha	9.36	146.1	88.74	0	0.38641	7.7545
secantTPP-tolG	9.36	193.02	135.76	0	0.38643	7.7545
secantTPP-wolfe	31	667.8	91	0	74.961	1.7713e+05
wolfe	29.12	203.92	153.2	0	4.565	74.526

Capitolo 4

Altre modifiche

4.1 Zoom e centratura del grafico

Sono stati aggiunti due controlli per aumentare l'ingrandimento dell'immagine e per spostare il punto di origine del grafico. Questo permette di seguire le traiettorie di ricerca degli algoritmi con più precisione, oltre che a visualizzare particolari zone d'interesse del problema.

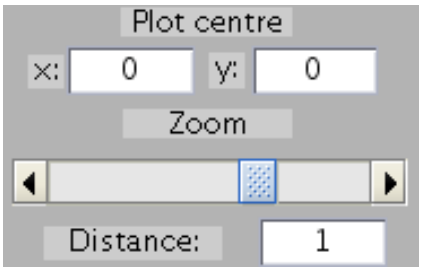


Figura 4.1: I controlli per zoom e centratura

4.2 Supporto e compatibilità

GklsGUI è adesso compatibile con *MATLAB 7.8*.
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 5

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`