3. IL METODO DI TUNNELING

L'analisi numerica ha da tempo reso disponibili una varietà di metodi di tipo iterativo per la soluzione di problemi di minimo. In questo campo si sono dimostrati particolarmente efficienti da un punto di vista computazionale i metodi di tipo locale, che consentono di ottenere rapidamente una buona approssimazione del minimo per problemi convessi. Data la loro economicità tali metodi hanno suscitato nel corso degli anni un largo interesse per quanto concerne la loro applicazione alla soluzione di problemi inversi caratterizzati da funzione oggetto particolarmente costose da un punto di vista computazionale. Nel ambito dell'*History Matching*, una volta sviluppate tecniche di calcolo analitico delle derivate della funzione di risposta del sistema rispetto ai parametri, tale interesse si è tradotto nello sviluppo di metodologie e software basati sull'implementazione di algoritmi locali, considerando in particolare metodi, quali il metodo di Levenberg – Marquardt, particolarmente adatti alla soluzione di problemi ai minimi quadrati non lineari.

Il successo di un tale approccio, basato appunto su metodi di minimizzazione di tipo locale, non deve però far dimenticare il limite intrinseco quando la funzione investigata non è convessa. In tal caso il minimo individuato per una fissata parametrizzazione è generalmente un semplice minimo locale, spesso associato ad un valore della funzione oggetto tale da non poter considerare soddisfacente la qualità dell'accordo. D'altra parte i metodi globali, più ambiziosi in quanto diretti all'individuazione del minimo globale della funzione oggetto, sono generalmente troppo costosi da un punto di vista computazionale.

Col desiderio di investigare possibili alternative economiche al semplice utilizzo di metodi locali, è stato considerato un particolare algoritmo di minimizzazione, detto di *Tunneling*, con caratteristiche, come vedremo, ibride. Tale metodo è stato sviluppato per risolvere problemi del tipo:

$$\min_{x_l \le x \le x_u} f(x) \qquad [3.1]$$

ove la funzione $f(x)$ è una funzione sufficientemente continua, non necessariamente ai

minimi quadrati, e x_l, x_u rappresentano i vincoli di minimo e di massimo imposti nello spazio delle variabili.

Il metodo che descriviamo in questo capitolo presenta caratteristiche di convergenza globali verso il minimo caratterizzato dal minor valore della funzione obiettivo nella spazio delle variabili. Tuttavia, il raggiungimento di tale obiettivo comporta un costo computazionale molto elevato, quindi risulta più importante sfruttare le caratteristiche locali ibride dell'algoritmo piuttosto che le sue proprietà di convergenza verso il minimo globale.

3.1. L'Algoritmo di Tunneling

Il metodo di *Tunneling* fu introdotto per la prima volta nell'anno 1977 da A.V. Levy e A. Montalvo [17]. L'idea di base di questo algoritmo consiste di due fasi fondamentali: nella prima fase, si applica un metodo di minimizzazione qualunque per trovare un primo minimo locale. Nella seconda fase si definisce una funzione ausiliaria, detta appunto funzione di *Tunneling*, che presenta una singolarità nel punto di minimo appena calcolato: questo accorgimento permette di allontanarsi dal bacino di attrazione del minimo locale e consente di esplorare lo spazio delle variabili per localizzare un punto in un'altra valle che diventerà un nuovo punto di partenza per il metodo di minimizzazione locale. Matematicamente, una volta trovato il primo minimo x^*, in qualche modo si cerca un punto x^0 in un'altra valle tale che:

$$f\left(x^0\right) - f\left(x^*\right) \leq 0 \qquad [3.2]$$

L'algoritmo di *Tunneling* può essere interpretato in modo molto semplice ed esaustivo considerando la minimizzazione di una funzione di una variabile reale, in modo da poter facilmente visualizzare graficamente il processo, nonché la filosofia che regge questo metodo. Le due fasi dell'algoritmo possono essere riassunte nella figura 3-1, in cui la prima fase di minimizzazione è seguita dalla fase di *Tunneling*, alla ricerca di un punto in un'altra valle caratterizzato da un valore inferiore della funzione oggetto.

Una volta terminata la fase di minimizzazione, è necessario allontanare la

funzione di *Tunneling* dalla zona di attrazione del minimo locale: ciò è possibile grazie all'introduzione di un polo artificiale.

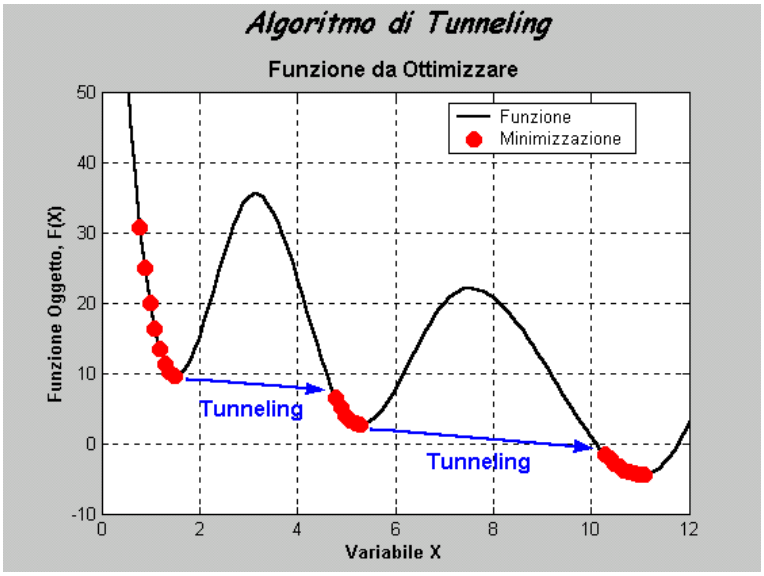

Figura 3-1. Schema dell'Algoritmo di Tunneling

L'introduzione di tale polo artificiale è ottenuta basandosi sulla definizione della funzione di *Tunneling* Classica:

$$T\left(x,x^*,\lambda^*\right)\overset{\text{def}}{=}\frac{f(x)-f\left(x^*\right)}{\left\|x-x^*\right\|^{2\lambda^*}} \qquad [3.3]$$

Dove il parametro $\lambda^*>0$ rappresenta la forza del polo artificiale e x^* è il minimo locale calcolato durante la fase di minimizzazione. Con il simbolo $\|\mathfrak{D}\|$ indicheremo d'ora in avanti sempre la norma euclidea del generico vettore $\mathfrak{D}$ nello spazio delle variabili. Graficamente, la definizione precedente si può interpretare osservando la figura 3-2.

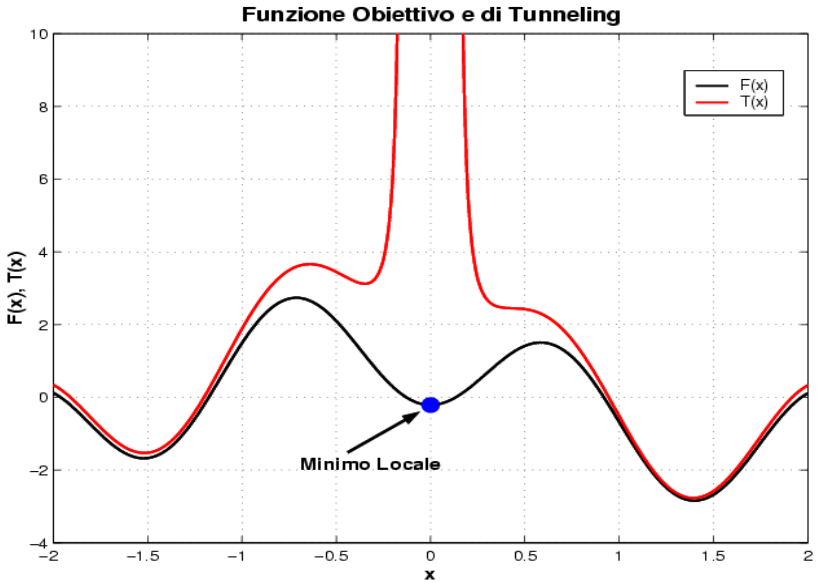

Figura 3-2. Collocamento del Polo Artificiale Nel Metodo di Tunneling

Il metodo di *Tunneling* può dunque essere riassunto considerando una prima fase di minimizzazione, in cui si calcola un minimo locale della funzione usando un generico algoritmo locale. Nella fase di *Tunneling* che segue, si genera un polo artificiale applicando la definizione [3.3] al problema traslato $f(x)-f(x^*)$. Quindi si cerca un nuovo punto x^0 tale per cui:

$$T(x^0, x^*, \lambda^*) \leq 0 \qquad [3.4]$$

Se la [3.4] è verificata, ne consegue che:

$$f(x^0) < f(x^*) \qquad [3.5]$$

Cioè è stato localizzato un punto x^0 in una valle differente caratterizzato da un valore della funzione obiettivo inferiore a quello relativo al minimo locale precedentemente calcolato. A questo punto, l'algoritmo è nuovamente inizializzato mediante una nuova fase di minimizzazione. Dato un punto di partenza x_0 nello spazio delle variabili, le due fasi sono ripetute alternativamente fino a convergenza; in uno

schema algoritmico si ha:

1. Minimizzazione Locale: $\rightarrow x_1 \quad , \quad \nabla f(x_1) = 0$

2. *Tunneling*: $\rightarrow x_1^* \quad , \quad f(x_1^*) \leq f(x_1)$

3. Minimizzazione Locale: $\rightarrow x_2 \quad , \quad \nabla f(x_2) = 0$

Etc…

L'idea, quindi, è quella di creare dei tunnel tra una valle e l'altra e generare una sequenza di minimi locali caratterizzati da un valore della funzione obiettivo sempre decrescente, cioè:

$$f(x_1) \geq f(x_2) \geq \ldots \geq f(x_G) \qquad [3.6]$$

Dove x_G è il minimo globale della funzione. Un'importante caratteristica del metodo è che è in grado di ignorare tutti i minimi locali dotati di un valore della funzione oggetto maggiore di quelli già calcolati.

3.1.1. IL METODO DI TUNNELING ESPONENZIALE

Uno dei principali problemi che si incontrano utilizzando la funzione ausiliaria di *Tunneling* Classica, definita dalla [3.3], deriva dalla precisione con la quale il minimo locale è calcolato. Se il metodo di minimizzazione non riesce a localizzare il punto stazionario con sufficiente precisione (ad esempio in caso di funzioni che presentano estese valli piatte), il numeratore della funzione di *Tunneling* non è positivo in tutto l'intorno del minimo x^*. In questo caso, il polo artificiale non può essere creato utilizzando la funzione di *Tunneling* Classica; questo effetto può essere analizzato grazie alla figura 3-3 e alla funzione obiettivo, presentata in figura, definita dall'espressione seguente:

$$F(x) = (x-a)\sum_{i=2}^{4}\{\sin[i(x-a)] + \cos[i(x-a)]\} \qquad [3.7]$$

In cui a è un parametro scalare fissato.

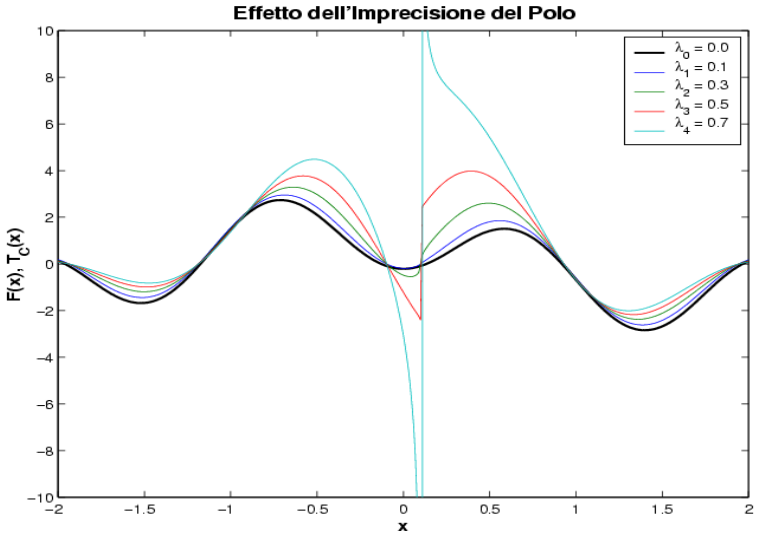

Figura 3-3. Tunneling Classico: Effetto dell'Imprecisione sul Polo

Come si nota analizzando la figura 3-3, il grafico della funzione di *Tunneling* Classica è completamente errato nell'intorno del minimo locale, dato che non mostra alcun punto di singolarità per tutti i valori di λ tranne $\lambda = 0.7$. Lo scopo dell'introduzione del polo artificiale è di annichilire il bacino di attrazione del minimo, ma nel metodo Classico le imprecisioni di calcolo rendono inutile questo stratagemma.

Un modo per aggirare questo problema è introdurre una nuova funzione ausiliaria; questa funzione è detta di *Tunneling* Esponenziale [11], secondo la definizione seguente:

$$T_E(x, x^*, \lambda^*) \overset{\text{def}}{=} [f(x) - f(x^*)] \cdot \exp\left(\frac{\lambda^*}{\|x - x^*\|^2}\right) \qquad [3.8]$$

La nuova funzione ausiliaria sfrutta le caratteristiche dell'esponenziale: è infatti possibile dimostrare che, utilizzando il metodo di *Tunneling* Esponenziale, la funzione crea un polo artificiale indipendentemente dalla precisione con la quale il minimo locale è stato calcolato e indipendentemente dalla forza del polo λ. La figura 3-4 mostra questo risultato utilizzando la funzione obiettivo [3.7].

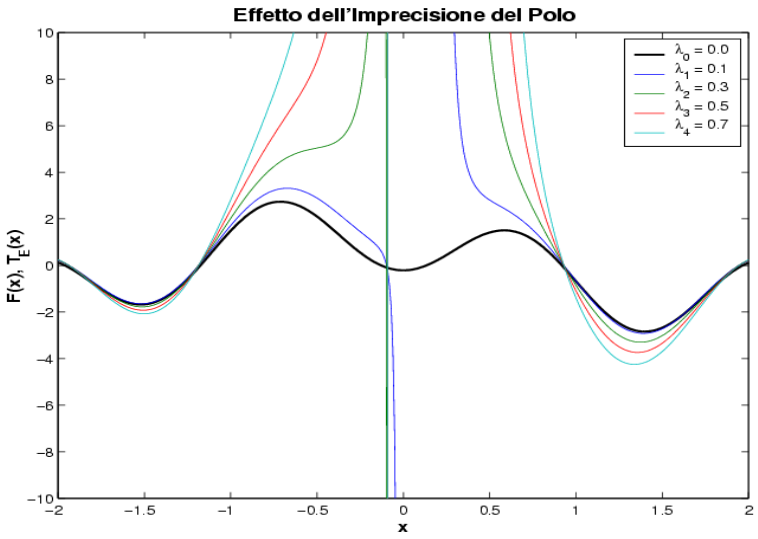

Figura 3-4. Tunneling Esponenziale: Effetto dell'Imprecisione sul Polo

D'ora in poi ci riferiremo alle funzioni di *Tunneling* Classica ed Esponenziale utilizzando i simboli T_C e T_E rispettivamente.

3.1.2. POLI MOBILI

Analizzando la definizione delle funzioni di *Tunneling*, si può notare che esistono dei punti per cui la condizione $\vec{\nabla} T(x) = 0$ è verificata (condizione di punto stazionario per la funzione di *Tunneling*) ma per i quali il valore della funzione oggetto originale è più elevato rispetto all'ultimo minimo calcolato. In questo caso, la minimizzazione della funzione di *Tunneling* si arresta, poiché è stata raggiunta una condizione necessaria di

punto stazionario. Tuttavia, la funzione ausiliaria $T(x)$ è ancora strettamente positiva, e di conseguenza la fase di *Tunneling* non può ancora essere terminata (cfr. [3.4]). Si è allora in presenza di un punto critico per $T(x)$.

Per evitare questo problema si crea un nuovo polo, detto polo mobile a causa del fatto che può essere attivato o disattivato a seconda della vicinanza della funzione di *Tunneling* ai punti critici. Detto x_m il punto critico per la funzione di *Tunneling*, i poli mobili possono essere creati modificando la funzione $T_C(x, x^*, \lambda^*)$ o $T_E(x, x^*, \lambda^*)$, e più precisamente, nel caso della funzione di *Tunneling* Classica:

$$T_C(x, x^*, \lambda^*, x_m, \lambda_m) = \frac{f(x) - f(x^*)}{\|x - x_m\|^{2\lambda_m} \|x - x^*\|^{2\lambda^*}} \qquad [3.9]$$

Mentre utilizzando la funzione di *Tunneling* Esponenziale:

$$T_E(x, x^*, \lambda^*, x_m, \lambda_m) = [f(x) - f(x^*)] \cdot \exp\left(\frac{\lambda_m}{\|x - x_m\|^2}\right) \cdot \exp\left(\frac{\lambda^*}{\|x - x^*\|^2}\right) \qquad [3.10]$$

Dove λ_m rappresenta la forza del polo mobile.

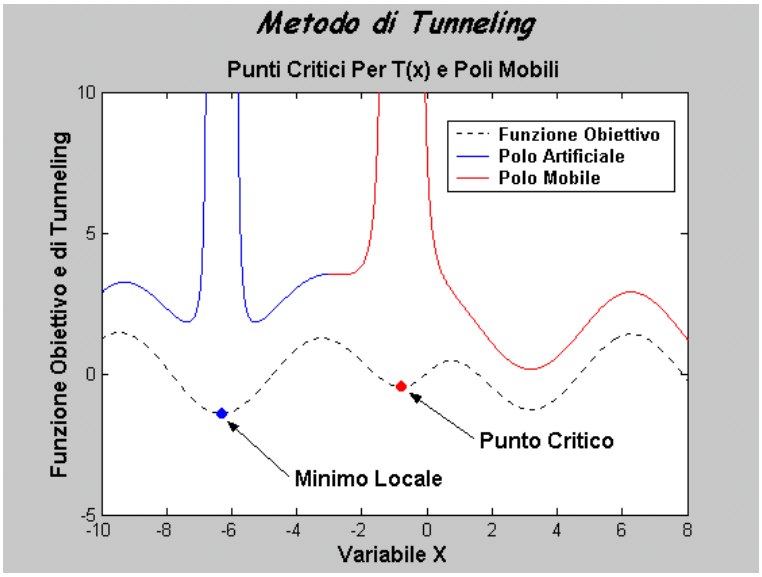

Figura 3-5. Punti Critici per la Funzione di Tunneling

Graficamente, la presenza del polo mobile è illustrata nella figura 3-5, in cui si evidenzia anche la presenza di un minimo locale e il relativo polo artificiale.

3.1.3. MINIMI LOCALI MULTIPLI

Originariamente, il metodo di *Tunneling* fu creato per risolvere problemi particolarmente ardui nel campo della chimica computazionale, quindi per ottimizzare le strutture molecolari basandosi sulla definizione di potenziale: il cosiddetto problema di Lennard – Jones. In questo campo, nonché per molte funzioni test tratte dalla letteratura matematica, si presenta un ostacolo: la possibile presenza di minimi (locali o globali) allo stesso livello, cioè caratterizzati dallo stesso valore della funzione oggetto (entro una certa tolleranza) ma differenti valori dei parametri.

Quindi, per evitare che il metodo venga nuovamente attirato nel bacino di attrazione di un minimo già calcolato, è necessario serbare memoria dei minimi locali o globali calcolati durante l'ottimizzazione.

Per ovviare a questo inconveniente, nelle funzioni di *Tunneling* Classica ed Esponenziale, rispettivamente, vengono introdotti i fattori:

$$\prod_{j=1}^{k} \frac{1}{\left\| x - x_j^* \right\|^{2\lambda_j^*}} \quad , \quad \prod_{j=1}^{k} \exp\left(\frac{\lambda_j^*}{\left\| x - x_j^* \right\|^2} \right) \qquad [3.11]$$

Dove $x_j^*,\ j = 1,...,k$ sono i punti di minimo locale allo stesso livello. L'espressione finale per la funzione di *Tunneling* Classica è dunque:

$$T_C\left(x, x^*, \lambda^*, x_m, \lambda_m\right) = \frac{f(x) - f\left(x^*\right)}{\left\| x - x_m \right\|^{2\lambda_m} \prod_{j=1}^{k} \left\| x - x_j^* \right\|^{2\lambda_j^*}} \qquad [3.12]$$

Mentre per la funzione di *Tunneling* Esponenziale si ha:

$$T_E\left(x, x^*, \lambda^*, x_m, \lambda_m\right) = \left[f(x) - f\left(x^*\right)\right] \cdot \exp\left(\frac{\lambda_m}{\left\|x - x_m\right\|^2}\right) \cdot \prod_{j=1}^{k} \exp\left(\frac{\lambda_j^*}{\left\|x - x_j^*\right\|^2}\right) \quad [3.13]$$

È interessante notare che, se x_G è un minimo globale per la funzione obiettivo, allora le funzioni di *Tunneling* sono positive nell'intero spazio delle variabili, vale a dire:

$$T_C\left(x, x_G, \lambda^*\right) \geq 0 \qquad \text{e} \qquad T_E\left(x, x_G, \lambda^*\right) \geq 0 \qquad \forall x \in \left[x_l, x_u\right] \qquad [3.14]$$

3.2. Algoritmo del Metodo di Tunneling

In questa sezione è descritto l'algoritmo generico per entrambe le funzioni di *Tunneling* [11]: per i dettagli di implementazione per ciascuno dei due metodi si rimanda alla sezione successiva. Dato quindi un punto iniziale x^0:

1. Eseguire una fase di minimizzazione che permetta di trovare un minimo locale $x^* \in \left[x_l, x_u\right]$ tale per cui $\vec{\nabla} f\left(x^*\right) = 0$ usando un algoritmo locale qualunque.

2. Eseguire una fase di *Tunneling*, cioè, in dettaglio:

   (a) Definire la funzione di *Tunneling* utilizzando la [3.3] o la [3.8] a seconda del metodo scelto.
   (b) Controllare se il metodo ha già trovato altri minimi locali allo stesso livello: se questo è il caso, calcolare i fattori [3.11] e ricalcolare la funzione ausiliaria di *Tunneling*.
   (c) Iniziare la fase di *Tunneling* partendo dal punto $\tilde{x}$, generato come segue:

   $$\tilde{x} = x^* + \varepsilon r$$

   Dove $\varepsilon << 1$ è un parametro che dipende dal tipo di funzione di *Tunneling* scelto (per i dettagli si rimanda alla sezione seguente) e $r \in \mathbb{R}^n$ è un vettore di componenti aleatorie tali per cui $-1 \leq r \leq 1$.
   (d) Calcolare la forza del polo λ^* necessaria ad annichilire il minimo locale; questo

valore è calcolato contemporaneamente alla direzione di discesa *d* basata sul gradiente della funzione obiettivo.

(e) Trovare un punto in una valle differente, ottenuto generando la sequenza:

$$x^{k+1} = x^k + \alpha^k d^k$$

dove α^k è il passo lungo la direzione di discesa del gradiente.

(f) Se un polo mobile è attivo, controllare se può essere disattivato (allontanamento dalla regione di attrazione di un punto critico per la funzione di *Tunneling*). Questo passo è descritto nella sezione seguente.

(g) Se la lunghezza del passo lungo la direzione di discesa α^k è prossima a zero, attivare un polo mobile; se ve n'è già uno attivo, incrementare la forza del polo mobile ponendo:

$$\lambda_m = \lambda_m + \Delta\lambda_m$$

e ricalcolare la funzione di *Tunneling* secondo la [3.12] o la [3.13].

(g) Se il numero di iterazioni nella fase di *Tunneling* è maggiore di un valore massimo *itermax*, iniziare una nuova fase di *Tunneling* generando un nuovo punto in una direzione aleatoria secondo il passo 2 – (b).

3.3. Dettagli di Implementazione

In questa sezione sono illustrati alcuni dettagli dell'implementazione del metodo di *Tunneling*, sia per la funzione di *Tunneling* Classica che per quella Esponenziale.

1. Nel passo 2 – (b) dell'algoritmo, il parametro ε deve essere valutato per generare un punto di partenza per la fase di *Tunneling*. Questo parametro è cruciale, dato che un piccolo valore di ε potrebbe non allontanare a sufficienza la funzione di *Tunneling* da una zona di attrazione di un punto critico, mentre con un valore elevato di ε si potrebbero mancare altri minimi locali presenti nelle vicinanze.

   Mediante un'analisi di sensitività, si è arrivati a concludere che, per la funzione di *Tunneling* Classica:

$$\varepsilon = \begin{cases} 0.1 & \text{se} \quad T_C(x) \geq 0.1 \\ 0.1 \cdot \left\| T_C(x) \right\| & \text{altrimenti} \end{cases} \qquad [3.15]$$

Mentre per la funzione di *Tunneling* Esponenziale:

$$\varepsilon = \frac{\lambda^*}{\ln(100)} \qquad [3.16]$$

Nel caso di problemi di *History Matching*, il fatto di calcolare un valore ottimale per il parametro ε non sarebbe strettamente necessario, dato che non si è alla ricerca di tutti i minimi locali della funzione ma solo di quelli caratterizzati da un piccolo valore della funzione obiettivo. Inoltre, è estremamente raro che esistano minimi locali o globali dotati dello stesso valore della funzione oggetto. Tuttavia, è stato detto che il metodo di *Tunneling* fu creato per risolvere problemi di chimica computazionale, per cui il fatto di trovare un gran numero di minimi locali e globali allo stesso livello può essere rilevante e a volte fondamentale.

2. Nel passo 2 – (e) dell'algoritmo, è necessario verificare se il polo mobile può essere disattivato. Perciò, la procedura da seguire è la seguente:

   a) Si generano due prodotti scalari: entrambi sono dati dal prodotto della derivata direzionale della funzione di *Tunneling* con la funzione stessa, con la differenza che nel primo il polo mobile è disattivato. Si valutano cioè i prodotti $d^T \vec{\nabla}_x T\left(x^k, x^*, \lambda^*, x_m, 0\right)$ e $d^T \vec{\nabla}_x T\left(x^k, x^*, \lambda^*, x_m, \lambda_m\right)$. Se questi hanno lo stesso segno, si pone $\lambda_m = 0$.

   b) Se $f\left(x^k\right) < 0.9 f\left(x_m\right)$ si pone $\lambda_m = 0$.

   c) Se il prodotto scalare tra la derivata direzionale con polo mobile disattivato e il vettore $\left(x - x_m\right)$ è positivo, si pone $\lambda_m = 0$.

3.4. Alcuni Esempi

Come tutti gli altri metodi di ottimizzazione esposti nella tesi, l'algoritmo di *Tunneling* è stato inizialmente testato avvalendosi di funzioni complesse, ma dotate di espressione analitica della funzione e dei gradienti, disponibili nella letteratura matematica riguardante l'ottimizzazione multidimensionale [12].

È qui possibile citare due esempi che rendano chiaro quali siano le prestazioni del metodo di *Tunneling*; entrambi sono generalmente classificati come *Hard Problem* [20], cioè sono un buon banco di prova per qualunque metodo di ottimizzazione si scelga di implementare. Il primo di essi è definito dalla [3.17], un esempio proposto da Levy e Gomez [16]:

$$f(x) = \frac{\pi}{n}\left\{k\sin\left(\pi x_1\right)^2 + \sum_{i=1}^{n-1}\left[\left(x_i - 1\right)^2\left(1 + k\sin\left(\pi x_{i+1}\right)^2\right)\right] + \left(x_n - 1\right)^2\right\}$$
$$-10 \le x_i \le 10 \qquad i = 1,\ldots,n \qquad [3.17]$$
$$f^* = f\left([1,1,\ldots,1]\right) = 0$$

La funzione presenta 10^n minimi locali (dove n è la dimensionalità del problema) e un solo minimo globale. La figura 3-6 mostra le iterazioni del metodo di *Tunneling*; in evidenza il punto di partenza, la soluzione e le curve di livello della funzione obiettivo.

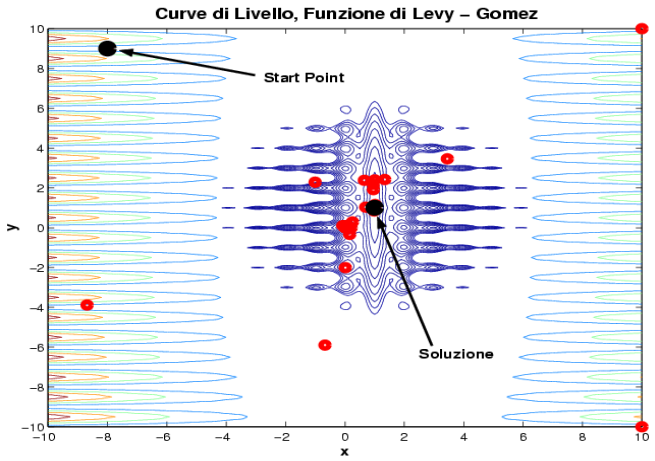

Figura 3-6. Tunneling: Funzione di Levy – Gomez

Il secondo esempio che prendiamo in considerazione è conosciuto come funzione di Shubert, ed è forse il test più diffuso nell'ambito dell'ottimizzazione matematica di funzioni. Il problema è definito dalla [3.18], e rappresentato in tre dimensioni nella figura 3-7.

$$f(x,y) = \left\{\sum_{i=1}^{5} i\cos\left[(i+1)x+i\right]\right\}\left\{\sum_{i=1}^{5} i\cos\left[(i+1)y+i\right]\right\}$$
$$-10 \le x, y \le 10 \qquad [3.18]$$
$$f^* = -187.36$$

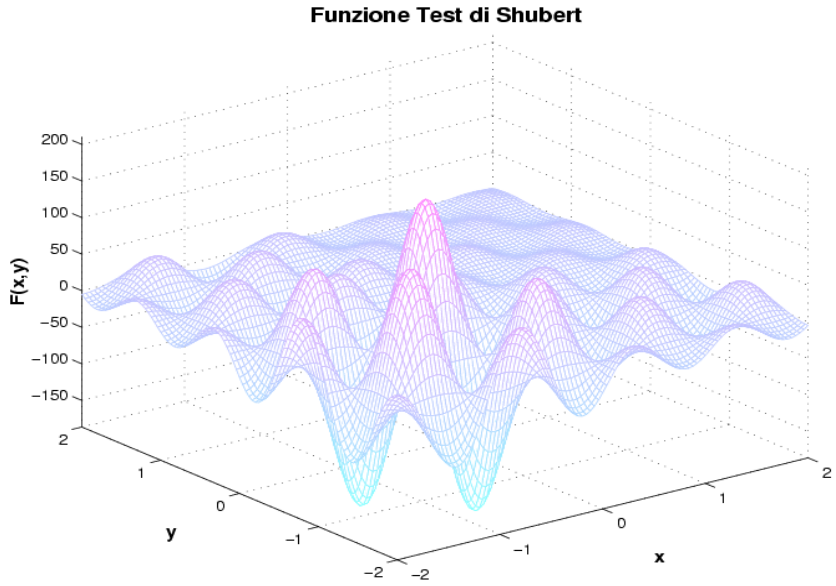

Figura 3-7. Funzione Test di Shubert

La funzione presenta 760 minimi locali, 18 dei quali sono globali. In questo caso, il metodo di *Tunneling* può essere impostato perché calcoli tutti i minimi globali della funzione oggetto oppure solo uno. Entrambe le simulazioni forniscono ottimi risultati da un punto di vista delle prestazioni, come è evidenziato nella tabella 3-1.

Funzione	Tunnel Classico	Tunnel Esponenziale
Levy – Gomez	33	42
Shubert (1 minimo)	120	53
Shubert (18 minimi)	10006	7746

Tabella 3-1. Tunneling: Prestazioni durante l'Ottimizzazione

La tabella 3-1 mostra il numero di valutazioni di funzioni necessarie per raggiungere il minimo globale per le due funzioni test prese in considerazione in questa sezione; nel caso della funzione di Shubert, all'algoritmo è stato richiesto di calcolare, durante la prima simulazione, solo un minimo globale, mentre nella seconda tutti i minimi globali sono stati localizzati.

3.5. Note sull'Algoritmo di Minimizzazione

Come abbiamo già visto, il metodo di *Tunneling* è una tecnica di ottimizzazione basata sul gradiente; entrambe le fasi dell'algoritmo richiedono la minimizzazione di una funzione. Nella prima fase si tratta della funzione obiettivo del problema considerato, mentre nella seconda si considera l'ottimizzazione della funzione ausiliaria di *Tunneling*.

Per questo motivo, all'interno del codice sorgente dell'algoritmo è implementato un metodo di minimizzazione basato sul gradiente: l'algoritmo di Broyden – Fletcher – Goldfarb – Shanno (BFGS), un metodo *Quasi – Newton* [17] che sfrutta le proprietà della matrice hessiana della funzione oggetto senza tuttavia calcolarla esplicitamente. Per i dettagli del metodo BFGS, si veda l'Appendice C.

Questo algoritmo fu originariamente implementato nel codice numerico di *Tunneling* poiché è specificamente progettato per trattare problemi su larga scala, e risulta affidabile e robusto anche per casi test descritti da migliaia di parametri. Inoltre, per le prime applicazioni del metodo alla chimica computazionale, il metodo BFGS fornì ottimi risultati.

Le simulazioni di *History Matching* sono state condotte basandosi sul codice sorgente di *Tunneling* originale, senza modificazioni di sorta. Tuttavia, come si vedrà

nelle sezioni seguenti, il fatto che l'*History Matching* sia definito come problema ai minimi quadrati penalizza in un certo senso il metodo BFGS, in quanto non è in grado di sfruttare tutte le importanti caratteristiche di ottimizzazione degli algoritmi costruiti appositamente per questo tipo di problema; ciò, come vedremo, porta ad un rallentamento della convergenza del metodo BFGS e quindi del metodo di *Tunneling*.

Dato appunto il tipo di implementazione considerato, nei casi previsti d'ora in poi ci riferiremo all'algoritmo globale come BFGS + BFGS.

3.6. Tunneling e History Matching

Come abbiamo visto nelle sezioni precedenti, l'algoritmo di *Tunneling* è in grado di fornire ottime prestazioni nel caso dell'ottimizzazione di funzioni analitiche tratte dalla letteratura matematica. Durante il processo di *History Matching*, tuttavia, il tempo di calcolo per ogni singola iterazione può essere molto grande, a seconda della complessità del giacimento considerato. Dunque, il fattore limitante nelle simulazioni è il numero di valutazioni di funzioni; questo genera due tipi di problemi utilizzando metodi di ottimizzazione basati sui gradienti, ed in particolare nel caso del *Tunneling*.

In primo luogo, a priori non si conosce l'andamento qualitativo della funzione obiettivo né il numero approssimativo di minimi locali che la caratterizzano: quindi non si è in grado di dire se la funzione oggetto possiede valli estese e fondamentalmente piatte o picchi molto profondi. In secondo luogo, limitando il numero di valutazioni di funzioni a poche centinaia, si è obbligati a richiedere una scarsa precisione nel calcolo dei minimi locali, per evitare che il metodo spenda troppe iterazioni per affinare il valore dei parametri in una valle di un punto stazionario. Come si è visto nelle sezioni precedenti, una scarsa precisione nel calcolo di un minimo locale può generare dei problemi nella definizione della funzione di *Tunneling* Classica. Per questo motivo, la totalità delle simulazioni condotte sui casi test di giacimento è stata effettuata basandosi sulla funzione di *Tunneling* Esponenziale. Inoltre, se la funzione possiede valli estese e piatte, una scarsa precisione nel calcolo di un minimo locale può portare a dei valori dei parametri fondamentalmente lontani da quelli ottimali.

Nonostante l'eccellente idea di fondo che regge l'algoritmo di *Tunneling*, un

punto a sfavore del metodo è dato dal fatto che è concepito come tecnica di ottimizzazione per funzioni generiche, scalari. Nella vita reale, una buona parte dei problemi di ottimizzazione consiste nella calibrazione di modelli matematici sfruttando l'accordo tra dati calcolati e proprietà misurate: vale a dire problemi ai minimi quadrati, la cui caratteristica principale è quella di possedere una funzione obiettivo vettoriale, le cui componenti sono funzioni scalari. Il loro numero è superiore al numero di variabili del problema.

Uno di questi problemi ai minimi quadrati è appunto l'*History Matching*; le tecniche di ottimizzazione migliori per la soluzione di questi casi, come il metodo di Levenberg – Marquardt o di Gauss – Newton, si basano su proprietà della funzione vettoriale che l'algoritmo di *Tunneling* non è in grado di sfruttare poiché tratta soltanto funzioni scalari. Un modo per migliorare le caratteristiche del metodo è utilizzare una tecnica di ottimizzazione per problemi ai minimi quadrati durante la fase di minimizzazione; in effetti, l'applicazione dell'algoritmo di Levenberg – Marquardt durante la prima fase del metodo di *Tunneling* in una simulazione di giacimento ha portato ad un significativo miglioramento delle prestazioni di questa tecnica di ottimizzazione.