UNIVERSITY OF PISA

Risoluzione di un problema di Constrained Convex Quadratic Programming con Primal-Dual Interior Point Method

Computational Mathematics for Learning and Data Analysis project report

Cornacchia Giuliano, Salinas Mario Leonardo
Gruppo 21

INDICE

1 INTRODUZIONE

Nel seguente report viene descritto ed analizzato il problema numero 3 *noML* assegnato per il progetto finale di *Computational Mathematics for Learning and Data Analysis* e descritta la soluzione da noi proposta, insieme alle varie scelte implementative.

2 DESCRIZIONE DEL PROBLEMA

Date le matrici $Q \in \mathbb{R}^{n\times n}$ ed $A \in \{0,1\}^{k\times n}$, con $Q \succeq 0$, e i vettori $q \in \mathbb{R}^n$ e $b = [1]^k$, il problema di ottimizzazione quadratica convessa primale P è definito come:

$$P := \begin{cases} min\ \ x^\top Qx + q^\top x \\ Ax = b \\ x \geq 0 \end{cases} \tag{2.1}$$

dove i vincoli nella matrice A formano k simplessi disgiunti della forma:

$$\sum_{i\in I^h} x_i = 1, h = 1,\dots,k \tag{2.2}$$

con $I^h, h = 1,\dots,k$ insiemi di indici che formano una partizione di $\{1,\dots,n\}$. Quindi il generico elemento della matrice A $a_{vi} = 1 \iff i \in I^v$.

Data la funzione da minimizzare in 2.1 $f(x) = x^\top Qx + q^\top x$ e la sua *funzione Lagrangiana* $L(x,\lambda_{eq},\lambda_s)$, a P si può associare il seguente *Problema Duale di Wolfe D*:

$$D := \begin{cases} \max\ L(x,\lambda_{eq},\lambda_s) \\ \nabla_x L(x,\lambda_{eq},\lambda_s) = 0 \\ \lambda_s \geq 0 \end{cases} = \begin{cases} \max\ x^\top Qx + q^\top x + \lambda_{eq}^\top (Ax-b) + \lambda_s^\top (-x) \\ 2Qx + q + A^\top \lambda_{eq} - \lambda_s = 0 \\ \lambda_s \geq 0 \end{cases} \tag{2.3}$$

Possiamo a questo punto scrivere il sistema KKT associato a P.

$$\begin{cases} \nabla_x L(x,\lambda_{eq},\lambda_s) = 0 \\ Ax - b = 0 \\ x_i \lambda_{s_i} = 0 \quad i = 1,\dots,n \\ (x,\lambda_s) \geq 0 \end{cases} \tag{2.4}$$

Scegliamo dunque di risolvere il sistema 2.4 applicando il metodo Primal-Dual Interior Point (PDIP): riformuliamo le condizioni di ottimalità 2.4 definendo una funzione $F : \mathbb{R}^{2n+k} \to \mathbb{R}^{2n+k}$ [1]:

$$F(x,\lambda_{eq},\lambda_s) = \begin{bmatrix} \nabla_x L(x,\lambda_{eq},\lambda_s) \\ Ax - b \\ XSe \end{bmatrix} = 0 \tag{2.5a}$$

$$(x,\lambda_s) \geq 0 \tag{2.5b}$$

dove

$$X = diag(x_1,\dots,x_n) \qquad S = diag(\lambda_{s_1},\dots,\lambda_{s_n}) \qquad e^\top = [1,\dots,1] \in \mathbb{R}^n \tag{2.6}$$

Il metodo PDIP, ad ogni iterazione k, genera triple $(x^k,\lambda_{eq}^k,\lambda_s^k)$ che soddisfano *strettamente* la 2.5b.
La procedura con la quale si ricercano le direzioni $(\Delta x,\Delta\lambda_{eq},\Delta\lambda_s)$ prende origine dal metodo di Newton

per equazioni non lineari [1]: alla k-esima iterazione il metodo di Newton forma un modello lineare di F attorno al punto corrente $(x^k, \lambda_{eq}^k, \lambda_s^k)$ e ottiene le direzioni di ricerca risolvendo il seguente sistema lineare:

$$J(x, \lambda_{eq}, \lambda_s) \begin{bmatrix} \Delta x \\ \Delta\lambda_{eq} \\ \Delta\lambda_s \end{bmatrix} = -F(x, \lambda_{eq}, \lambda_s) \tag{2.7}$$

dove J è la Jacobiana di F. Nel nostro caso il sistema da risolvere diventa:

$$\begin{bmatrix} 2Q & A^\top & -I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda_{eq} \\ \Delta\lambda_s \end{bmatrix} = - \begin{bmatrix} r_d \\ r_p \\ XSe \end{bmatrix} \tag{2.8}$$

dove

$$r_d = 2Qx + q + A^\top\lambda_{eq} - \lambda_s \qquad r_p = Ax - b \tag{2.9}$$

Percorrere un passo intero lungo le direzioni trovate risolvendo 2.8 potrebbe violare il vincolo $(x, \lambda_s) \geq 0$, quindi aggiungiamo una parametro $\alpha \in (0, 1]$, detto *step-size*, che servirà a ridurre l'ampiezza del passo $(\Delta x, \Delta\lambda_{eq}, \Delta\lambda_s)$ per garantire il soddisfacimento del vincolo 2.5b.

$$(x^{k+1}, \lambda_{eq}^{k+1}, \lambda_s^{k+1}) = (x^k, \lambda_{eq}^k, \lambda_s^k) + \alpha(\Delta x, \Delta\lambda_{eq}, \Delta\lambda_s) \tag{2.10}$$

Data la corrente iterazione $(x^k, \lambda_{eq}^k, \lambda_s^k)$, che soddisfa 2.5b, introduciamo il *centering parameter* $\sigma \in [0, 1]$ e la *duality measure* $\mu = \frac{x^\top\lambda_s}{n}$; questi due parametri vengono utilizzati per direzionare il Newton step verso un punto per il quale valga $x_i\lambda_{s_i} = \sigma\mu$, piuttosto che a una soluzione diretta di 2.4. A seguito di questa considerazione, il nuovo step verrà calcolato risolvendo il *KKT perturbato* definito come:

$$\begin{bmatrix} 2Q & A^\top & -I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda_{eq} \\ \Delta\lambda_s \end{bmatrix} = - \begin{bmatrix} r_d \\ r_p \\ XSe - \sigma\mu e \end{bmatrix} \tag{2.11}$$

Il sistema 2.11 è non-simmetrico, lo trasformiamo in un sistema lineare equivalente simmetrico, eliminando la terza riga ed esprimendo $\Delta\lambda_s$ in funzione di Δx.

La terza riga di 2.11 può essere eliminata poichè durante le iterazioni x_i ed s_i rimangono strettamente positivi; per ricavare $\Delta\lambda_s$ in funzione di Δx dobbiamo prima isolare $\Delta\lambda_s$ sempre dalla terza riga del sistema:

$$\begin{aligned} S\Delta x + X\Delta\lambda_s &= -XSe + \sigma\mu e \\ X\Delta\lambda_s &= \sigma\mu e - XSe - S\Delta x \\ \Delta\lambda_s &= X^{-1}(\sigma\mu e - S\Delta x) - \lambda_s \end{aligned} \tag{2.12}$$

possiamo dunque riscrivere 2.11 sostituendo $\Delta\lambda_s$ come in 2.12:

$$\begin{aligned} 2Q\Delta x + A^\top\Delta\lambda_{eq} - \Delta\lambda_s &= -2Qx - q - A^\top\lambda_{eq} + \lambda_s \\ 2Q\Delta x + A^\top\Delta\lambda_{eq} - X^{-1}(\sigma\mu e - S\Delta x) + \cancel{\lambda_s} &= -2Qx - q - A^\top\lambda_{eq} + \cancel{\lambda_s} \\ (2Q + X^{-1}S)\Delta x + A^\top\Delta\lambda_{eq} &= -2Qx - q - A^\top\lambda_{eq} + X^{-1}\sigma\mu e \end{aligned} \tag{2.13}$$

ponendo $M = 2Q + X^{-1}S$ otteniamo il seguente sistema simmetrico detto anche *augmented KKT*:

$$\begin{bmatrix} 2Q + X^{-1}S & A^\top \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda_{eq} \end{bmatrix} = - \begin{bmatrix} r_d - X^{-1}\sigma\mu e + \lambda_s \\ r_p \end{bmatrix} \tag{2.14a}$$

$$\Delta\lambda_s = X^{-1}(\sigma\mu e - S\Delta x) - \lambda_s \tag{2.14b}$$

la matrice a sinistra in 2.14a è simmetrica poichè:

- M è simmetrica perchè somma di una matrice simmetrica e una matrice diagonale
- il blocco inferiore sinistro e superiore destro sono l'uno il trasposto dell'altro

inoltre se A ha rango massimo essa è non-singolare, e 2.14a ammette soluzione. I vincoli in A, nel nostro caso di studio, formano k simplessi disgiunti, quindi $rank(A) = k$.

Il sistema 2.14a è dunque simmetrico, sparso e la matrice potrebbe essere mal condizionata a causa del prodotto $X^{-1}S$. Il metodo solitamente utilizzato per risolvere sistemi simmetrici sparsi è MINRES, ma non essendo stato affrontato durante il corso, utilizzeremo una sua generalizzazione, GMRES.

Data la simmetria del sistema lineare da risolvere, avremmo potuto anche optare per LDL-factorization, ma non è da escludere che nel nostro problema l'algoritmo possa incontrare degli zero-pivot. Sebbene esistano tecniche che consistono nel trattare un blocco 2x2 come pivot, ovviando cosi al problema, risolveremo il sistema 2.14a con il metodo iterativo GMRES.

3 Metodo Risolutivo

3.1 Primal-Dual Interior Point method

L'intuizione principale dei metodi Primal-Dual è quella di considerare sia il problema di minimizzazione P che il suo duale D per ottenere un limite superiore $v(P)$ ed inferiore $v(D)$ della soluzione. Da P e D si ricava quindi il sistema KKT 2.14a associato e una misura della distanza della soluzione attuale dall'ottimo, detta *complementary gap*; definita come la differenza fra il valore della funzione obiettivo in P e in D, eventualmente normalizzata.

Ad ogni iterazione, una volta trovata una soluzione per 2.14a, calcoliamo una nuova coppia di soluzioni primali/duali e riduciamo il complementary gap.

Con gli elementi presentati finora possiamo delineare la struttura dello pseudocodice del nostro metodo come segue:

Algorithm 1 pseudocodice Interior-Point Primal-Dual method

1: **function** PDIP(Q, q, A, b, eps, maxit)
2: *inizializzare* $(x^0, \lambda_{eq}^0, \lambda_s^0) > 0$
3: $\mu^0 \leftarrow (x^{0\top}\lambda_s^0)/n$
4: $\sigma \in [0,1]$
5: $k \leftarrow 0$
6: **while** *k<maxit & complementary_gap < eps*
7: $\mu^{k+1} \leftarrow \sigma\mu^k$
8: *risolvere* $\begin{bmatrix} 2Q + X^{-1}S & A^\top \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta x^{k+1} \\ \Delta\lambda_{eq}^{k+1} \end{bmatrix} = - \begin{bmatrix} r_d - X^{-1}\mu^{k+1}e + \lambda_s \\ r_p \end{bmatrix}$
9: $\Delta\lambda_s^{k+1} \leftarrow X^{-1}(\mu^{k+1}e - S\Delta x^{k+1}) - \lambda_s^k$
10: *calcolare* $(\alpha_x^{k+1}, \alpha_{\lambda_{eq}}^{k+1}, \alpha_{\lambda_s}^{k+1})$ *tale che* $(x^{k+1}, \lambda_s^{k+1}) > 0$
11: $(x^{k+1}, \lambda_{eq}^{k+1}, \lambda_s^{k+1}) \leftarrow (x^k, \lambda_{eq}^k, \lambda_s^k) + (\alpha_x^{k+1}\Delta_x^{k+1}, \alpha_{\lambda_{eq}}^{k+1}\Delta_{\lambda_{eq}}^{k+1}, \alpha_{\lambda_s}^{k+1}\Delta_{\lambda_s}^{k+1})$
12: $k \leftarrow k+1$

Nell'algoritmo sopra descritto sono cruciali la scelta della tripla iniziale (riga 2) e la scelta di α (riga 10); nelle sezioni seguenti descriveremo le nostre scelte progettuali a riguardo. Le condizioni di stop scelte (riga 6) sono descritte in 3.4.

Mentre per la risoluzione della riga 8 utilizzeremo sia un approccio iterativo (GMRES) che diretto (versione modificata di LDL). Entrambi i metodi verranno presentati nella sezione 3.5.

3.2 Punto inziale

Il nostro obiettivo è trovare una tripla iniziale $(x^0, \lambda_{eq}^0, \lambda_s^0)$ che sia ammissibile per P e D. Di seguito presentiamo le scelte implementative per inizializzare i vettori della tripla.

Scelta x^0

Per la scelta di x^0 sfruttiamo la struttura della matrice dei vincoli A; ogni x_i appare in un solo vincolo ed il vincolo è definito come in 2.2, quindi:

$$x_i = \frac{1}{|I^h|} \tag{3.1}$$

dove I^h è l'insieme di indici, ovvero il simplesso, che contiene l'indice i.

Scelta λ_{eq}^0 e λ_s^0

Inizializziamo λ_{eq}^0 con numeri positivi $\in (0,1)$; mentre per garantire l'ammissibilità duale troviamo λ_s^0 risolvendo $\nabla_x L(x^0, \lambda_{eq}^0, \lambda_s^0) = 0$.

$$\overline{\lambda_s} = 2Qx^0 + q + A^\top \lambda_{eq}^0 \tag{3.2}$$

Nel vettore λ_s^0 potrebbero essere presenti componenti negative, quindi definiamo Δ_s come:

$$\Delta_s = -(3/2)\ max(0, \min_i \overline{\lambda_{s_i}}) \tag{3.3}$$

a questo punto scriviamo $\lambda_s^0 = \overline{\lambda_s} + e\Delta_s$.

3.3 Scelta dello step-size α

Scegliamo di utilizzare uno step-size distinto per ogni vettore della tripla $(x, \lambda_{eq}, \lambda_s)$. Poichè vogliamo assicurare che, muovendoci lungo le direzioni trovate risolvendo il sistema 2.14a, i soli vincoli di non negatività in 2.5b siano soddisfatti, scegliamo il massimo α tale che i vincoli non vengano violati. Per il generico vettore $d \in \{x, \lambda_{eq}, \lambda_s\}$

$$\alpha_{max}^d = \min(1, \min_{i:\Delta d_i < 0} -\frac{d_i}{\Delta d_i}) \tag{3.4}$$

E quindi scegliamo lo step α come segue:

$$\alpha^d = \min(1, \eta \alpha_{max}^d) \quad dove\ \ \eta \in [0.9, 1) \tag{3.5}$$

3.4 Criteri di stop

L'algoritmo può terminare sia perchè è stato raggiunto il numero massimo di iterazioni, oppure perchè il *complementary gap* è minore di ϵ, ovvero $v(P) - v(D) < \epsilon$. Nella nostra analisi queste due soglie sono fissate rispetivvamente a 100 e 10^{-14}.

3.5 Risoluzione del sistema lineare

Per la risoluzione del sistema in 2.14a abbiamo deciso di utilizzare e confrontare due diversi approcci; uno iterativo (GMRES) e l'altro diretto (LDL).

GMRES è un metodo iterativo per la soluzioni di sistemi lineari $Ax = b$ non simmetrici di grandi dimensioni. La soluzione esatta del sistema è $x_* = A^{-1}b$. L'idea del metodo GMRES è di approssimare la soluzione x_* al passo n con un vettore $x_n \in K_n$, dove K_n è il Krylov subspace $\langle b, Ab, \dots, A^{n-1}b \rangle$, che minimizza la norma del residuo $r_n = b - Ax_n$. Questo si traduce in determinare x_n risolvendo un least squares problem [2].

Mentre per la risoluzione diretta del sistema, usiamo una versione modificata di LDL per matrici simmetriche, implementata da `MATLAB` dall'operatore *mldivide*.

3.6 Convergenza del metodo

I risultati teorici sulla convergenza dei metodi PDIP [1] dimostrano che effettuando scelte appropriate sulla tripla iniziale $(x^0, \lambda_{eq}^0, \lambda_s^0)$, per $\epsilon > 0$ fissato, il limite superiore al numero delle iterazioni per garantire la convergenza dell'algoritmo è $log(n\, log(1/\epsilon))$ dove n è l'input size del problema.

Il nostro metodo è di tipo *infeasible*, ovvero ammette un punto iniziale che non soddisfi i vincoli in P e D, ma data la scelta di uno step size diverso per il problema primale e duale, e l'aggiunta del centering parameter - scelte che in pratica aumentano la velocità di convergenza - ci aspettiamo che il nostro metodo converga comunque in un numero di iterazioni di ≈ 100, per $n \to \infty$, come empiricamente osservato sui metodi di questa classe.

4 Codice prodotto e validazione

In questa sezione descriviamo il codice che implementa il metodo PDIP, la generazione delle istanze dei problemi e gli esperimenti.

4.1 Codice `MATLAB`

La nostra soluzione è composta da cinque files:

- `genProblem.m` script che genera un'instanza del problema (Q, q, A, b) usando funzioni ausiliare definite negli altri script.
- `genQF.m` script che dati dimensione n, $\delta \in (0, 1]$ e un vettore $v \geq 0$ genera un vettore $q \in \mathbb{R}^n$ ed una matrice $Q \in \mathbb{R}^{n \times n}$ con densità δ e autovalori v. Per generare Q con queste proprietà è stata utilizzata la funzione `MATLAB` `sprandsym`.
- `generateAdisjointed.m` script che genera la matrice dei vincoli A dato il numero di simplessi m e il numero di variabili n; il metodo si assicura che la matrice generata rappresenti un insieme di partizioni di indici disgiunti e che ogni insieme I^h contenga almeno due indici, questo per evitare il soddisfacimento banale di tale vincolo.
- `feasible_sp.m` script che dati Q, q ed A calcola il punto iniziale $(x^0, \lambda_{eq}^0, \lambda_s^0)$ come mostrato in 3.2.
- `PDIP.m` script che implementa il metodo primale duale del punto interno come in Algoritmo 1.

Di seguito, in Codice 4.1, riportiamo un esempio di utilizzo dei files appena descritti che risolve un problema come quello in analisi, di dimensione $n = 1000$ con $k = 200$ vincoli e densità di Q pari a 0.4, usando LDL come metodo per la risoluzione del sistema KKT.

```
» n = 1000; m = 200; delta = 0.4;
» p = genProblem(n, m, delta);
» PDIP(p, 100, 1e-14, ‘‘ldl’’);
Primal-Dual Interior Point method

 iter     fval        gap       dualf     primalf       sTx
 ----------------------------------------------------------
   1    1.669e+01   2.470e+01  4.274e+01  7.364e-16   4.124e+02
   2   -6.276e+00   2.969e+01  3.032e+00  7.135e-15   1.863e+02
   3   -3.370e+01   2.461e+00  9.211e+00  9.905e-15   8.294e+01
   .
   .
   .
  21   -6.565e+01   2.554e-13  1.201e-12  1.138e-15   1.677e-11
  22   -6.565e+01   4.200e-14  1.979e-13  1.196e-15   2.754e-12
  23   -6.565e+01   6.927e-15  3.322e-14  1.053e-15   4.522e-13


Execution terminated because duality gap reduced under the threshold
Primal-Dual Interior Point method terminated in 23 iterations
elapsed time is 2.3502 seconds
fval = -6.565e+01 and complementary gap = 6.927e-15
```

Codice 4.1: esempio di esecuzione del codice prodotto.

4.2 ESPERIMENTI

Allo scopo di testare l'implementazione proposta sono stati effettuati esperimenti considerando PDIP sia con risoluzione iterativa (*PDIP-GMRES*) che diretta (*PDIP-LDL*) del sistema lineare. Inoltre tempi ed accuratezza del metodo da noi implementato sono stati confrontati con quelli della funzione *built-in* di `MATLAB` `quadprog`.

I parametri coinvolti nella nostra analisi sono:

- dimensione dell'input n;
- m, percentuale rispetto ad n di vincoli; abbiamo anche che $k = m \cdot n = rank(A) = |I| \leq n/2$;
- densità δ della matrice Q: la percentuale di elementi non-nulli in Q.

Gli esperimenti possono essere divisi in quattro gruppi:

1. nel primo insieme di esperimenti abbiamo voluto testare la scalabilità del nostro metodo fissando m e densità al variare di n
2. nel secondo gruppo di esperimenti abbiamo voluto investigare l'effetto del numero dei vincoli sul tempo di convergenza; abbiamo fatto variare m lasciando fisse n e densità.
3. nel terzo gruppo di esperimenti abbiamo investigato gli effetti di δ fissati m ed n
4. infine il quarto tipo di esperimenti è mirato a valutare sia la convergenza dell'implementazione proposta, che l'accuratezza della soluzione trovata; quindi, fissati n, m e δ, analizziamo l'andamento del gap e la differnza fra le soluzioni del nostro metodo e `quadprog`

La Tabella 4.1 riassume i valori assegnati ai parametri durante gli esperimenti.

Per ogni esperimento sono state effettuate 20 ripetizioni per poi calcolare media e deviazione standard delle metriche. Tali esperimenti sono stati eseguiti su un Laptop con processore `Intel(R) Core(TM) i5-3337U CPU @ 1.80GHz (4 CPUs)` e 6 GB di RAM.

Esperimento	n	m	δ
Scalabilità	[50, 75, 100, 200, 300, ... , 1000, 1250, 1500, 1750, 2000]	40%	40%
N° Vincoli	1000	[5%, 10%, 20%, 30%, 40%, 50%]	40%
Densità	1000	20%	[10%, 20%, ... , 100%]
Convergenza & Accuratezza	1000	20%	40%

Tabella 4.1: Tabella che riassume i parametri coinvolti negli esperimenti.

5 Risultati

Questa sezione è dedicata ai risultati sperimentali. Nelle tabelle in seguito, con il termine *slowfactor* ci riferiremo al rapporto fra il tempo di esecuzione, in secondi, di PDIP-LDL e QUADPROG. In queste tabelle in grassetto sono evidenziati il massimo e minimo *slowfactor*.

I files chiamati `ResultsEXP{1,2,3}.mat` contengono le strutture `MATLAB` che memorizzano i risultati dei sotto-esperimenti. Per il quarto sotto-esperimento sono state utilizzate le soluzioni già calcolate negli altri sotto-esperimenti.

Il numero nel nome del file indica il sotto-esperimento, ciascun file contiene tre strutture che raccolgono i risultati di ogni metodo nel seguente ordine: PDIP-LDL, PDIP-GMRES e QUADPROG. La tabella 5.1 descrive i campi della generica struttura.

experiment	*Stringa che identifica il sotto-esperimento.*
method	*Stringa che specifica quale dei tre metodi è stato utilizzato.*
times	*Matrice di dimensione* $\|parameters\| \times nrepeat$ *che contiene i tempi di esecuzione del metodo per ogni valore* $\in$ *parameters e per ognuna delle* $nrepat$ *ripetizioni dell'esperimento.*
iterations	*Matrice di dimensione* $\|parameters\| \times nrepeat$ *che contiene il numero di iterazioni del metodo per ogni valore* $\in$ *parameters e per ognuna delle* $nrepat$ *ripetizioni dell'esperimento.*
parameters	*Vettore che contiene i valori che verranno assunti dal parametro preso in esame nel sotto-esperimento corrente.*

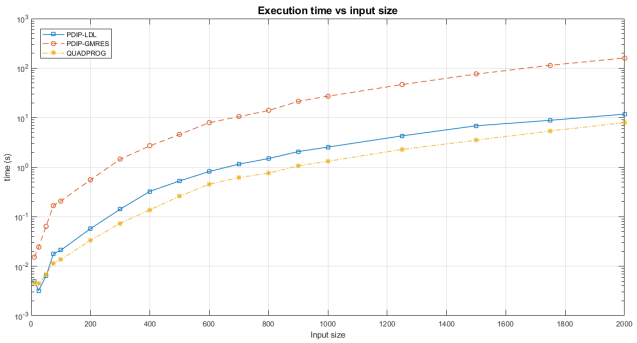
Tabella 5.1: Tabella che descrive i campi delle strutture che contengono i risultati sperimentali.

5.1 Scalabilità

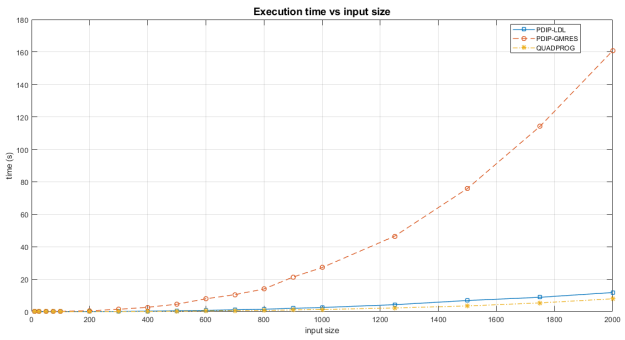
In questo gruppo di esperimenti abbiamo voluto testare la scalabilità della nostra implementazione: fissando numero di vincoli m e densità di Q, δ, abbiamo fatto variare la *dimensione dell'input* n. Analizziamo ora grafici e tabelle relative a tempi di esecuzione e numero di iterazioni.

La Figura 5.1 ci permette di confrontare i tempi di completamento dei metodi analizzati; possiamo notare che l'andamento è esponenziale rispetto ad n per tutti e tre i metodi. In particolare:

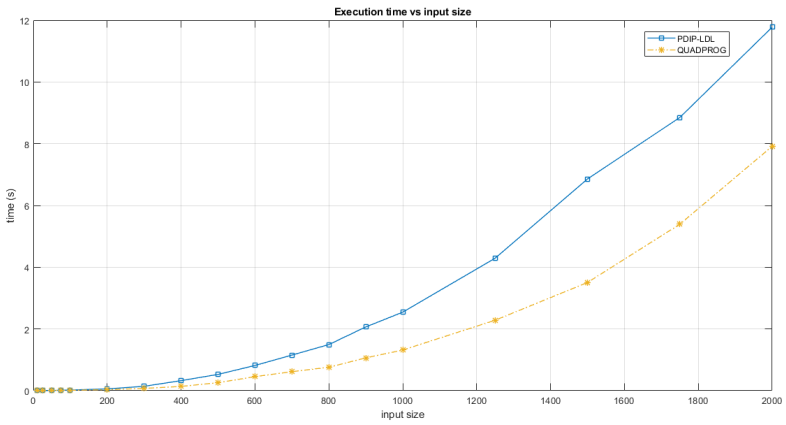
- in Fig.5.1b la curva relativa al metodo PDIP-GMRES cresce con un esponente maggiore rispetto agli altri due metodi, questo a causa della presenza di metodo iterativo al suo interno, ossia GMRES, che all'aumentare di n dovrà approssimare la soluzione di un sistema sempre più grande.
- confrontando i due metodi più veloci in Fig.5.1c e Tab.5.4 possiamo notare come la nostra implementazione abbia scalabilità paragonabile a quella di QUADRPROG a meno di un fattore $\approx \times 1.85$.



(a) Confronto fra i tre metodi con asse delle y in *log-scale*.



(b) Confronto fra i tre metodi.



(c) Grafico di confronto fra PIDP-LDL e QUADPROG.

Figura 5.1: Tempo di esecuzione, in secondi, all'aumentare della dimensione del problema n.

input size	**50**	**75**	**100**	**200**	**300**	**400**	**500**	**600**
PDIP-LDL	0.0064	0.0177	0.0211	0.0575	0.1418	0.3251	0.5271	0.8170
QUADPROG	0.0069	0.0115	0.0137	0.0334	0.0737	0.1377	0.2590	0.4541
slowfactor	**0.9259**	1.5432	1.5422	1.7229	1.9247	**2.3617**	2.0351	1.7989
input size	**700**	**800**	**900**	**1000**	**1250**	**1500**	**1750**	**2000**
PDIP-LDL	1.1532	1.4939	2.0655	2.5441	4.2911	6.8573	8.8513	11.7768
QUADPROG	0.6157	0.7584	1.0631	1.3172	2.2816	3.5041	5.3935	7.9170
slowfactor	1.8728	1.9697	1.9428	1.9315	1.8808	1.9569	1.6411	1.4875

Tabella 5.2: Tabelle contenenti la media dei tempi di esecuzione (s) di PDIP-LDL e QUADPROG relativi al primo sotto-esperimento.

input size	**50**	**75**	**100**	**200**	**300**	**400**	**500**	**600**
PDIP-LDL	10.65%	29.22%	17.34%	5.81%	8.03%	3.87%	5.03%	1.80%
QUADPROG	13.35%	25.80%	17.72%	2.55%	3.12%	4.95%	19.12%	7.99%
input size	**700**	**800**	**900**	**1000**	**1250**	**1500**	**1750**	**2000**
PDIP-LDL	3.96%	2.74%	1.67%	1.94%	2.27%	1.56%	2.31%	2.41%
QUADPROG	2.94%	1.37%	1.42%	0.94%	1.13%	0.85%	0.92%	0.83%

Tabella 5.3: Tabella contenente la deviazione standard dei tempi di esecuzione di PDIP-LDL e QUADPROG relativi al primo sotto-esperimento.

La Tab. 5.3 mostra come le prestazioni ottenute sono stabili anche in termini di deviazione standard; si nota un leggero aumento di quest'ultima sulle istanze piccole del problema in quanto una piccola variazione incide maggiormente sulla metrica.

La variazione dell'input-size non sembra invece influire in maniera significativa sul numero di iterazioni necessarie alla convergenza dei metodi. Il grafico in Fig.5.2 mostra infatti come, a prescindere dal valore di n, PDIP impieghi circa 21 iterazioni mentre QUADPROG circa 7.

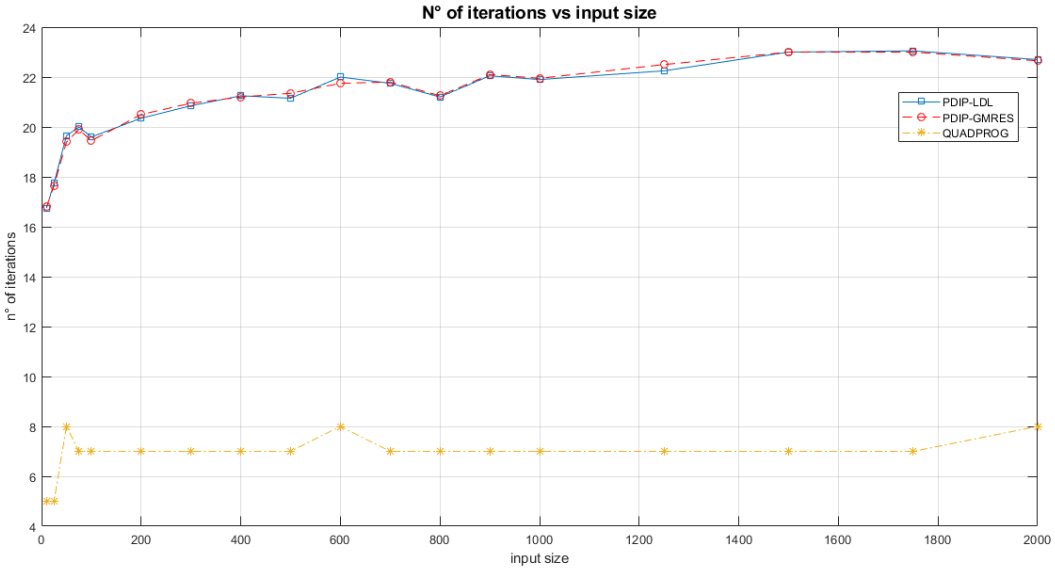
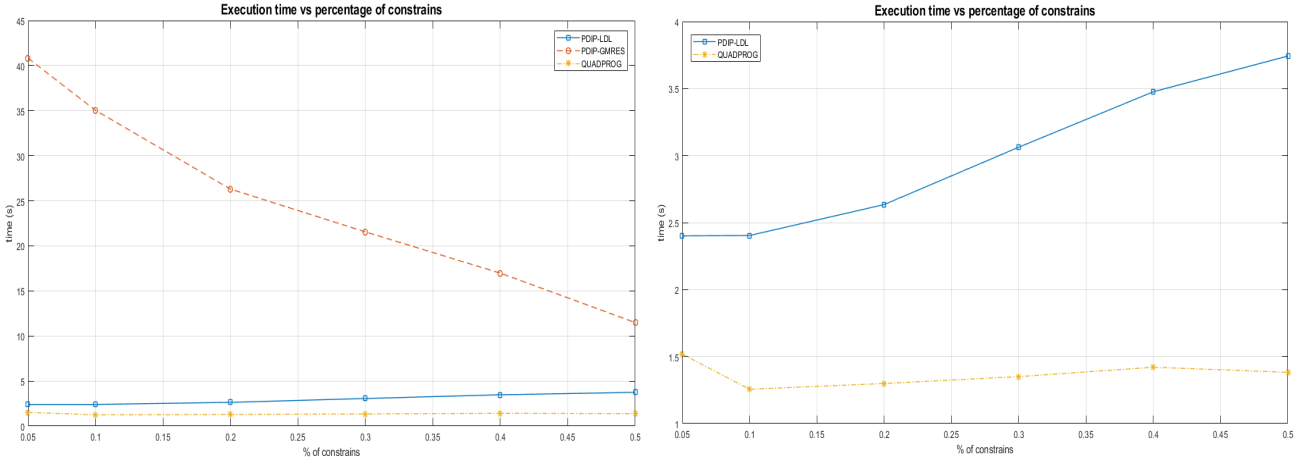


Figura 5.2: Il grafico mostra l'andamento del numero di iterazioni all'aumentare di n.

5.2 Numero di Vincoli

In questo gruppo di esperimenti abbiamo voluto analizzare l'effetto che la variazione di m ha sul tempo di convergenza dei metodi; δ ed n rimangono fissati come in Tab. 4.1.



(a) Grafico di confronto fra i tempi di esecuzione dei tre metodi.

(b) Grafico di confronto fra PIDP-LDL e QUADPROG.

Figura 5.3: I grafici mostrano l'andamento del tempo di esecuzione, in secondi, all'aumentare del numero dei vincoli m; espresso in percentuale rispetto ad n.

in Fig.5.3a GMRES impega meno tempo per covergere all'aumentare del numero di vincoli; questo perchè è più efficiente su matrici sparse: poichè il numero di elementi non-nulli in A è sempre n, all'aumentare di m, la sparsità del sistema aumenta, confermando che le performance di GMRES su sistemi sparsi sono migliori.

Confrontando invece PDIP-LDL e QUADPROG in Fig.5.3b, notiamo come l'aumentare del numero di vincoli impatti in modo negativo sulle performance della nostra implementazione, probabilmente a causa dell'aumento della dimensione del sistema da risolvere che penalizza l'approccio diretto. Anche il tempo di convergenza di QUADPROG aumenta con m, ma in modo meno netto.

La Tabella 5.4 conferma la stabilità del nostro metodo più veloce: si osservano infatti deviazioni standard trascurabili, dello stesso ordine di grandezza di QUADPROG; è possibile osservare anche che lo slowfactor aumenta in modo monotono al variare di m, rimanendo compreso nell'intervallo $[1.58, 2.7]$.

m	**0.05**	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**
PDIP-LDL	$2.401 \pm 2.6\%$	$2.403 \pm 1.8\%$	$2.634 \pm 2\%$	$3.062 \pm 3\%$	$3.476 \pm 1.9\%$	$3.743 \pm 2.8\%$
QUADPROG	$1.519 \pm 3.3\%$	$1.256 \pm 2.4\%$	$1.299 \pm 0.9\%$	$1.351 \pm 1.6\%$	$1.420 \pm 0.9\%$	$1.382 \pm 3.6\%$
slowfactor	**1.581**	1.913	2.027	2.267	2.446	**2.707**

Tabella 5.4: Tabella contenente media e deviazione standard dei tempi di esecuzione (s) di PDIP-LDL e QUADPROG relativi al secondo sotto-esperimento.

Infine, osserviamo che in Fig. 5.4 il numero di iterazioni diminuisce all'aumentare di m, questo però non vale per QUADPROG il cui numero di iterazioni non viene influenzato dalla frazione di vincoli rispetto alla dimensione dell'input.

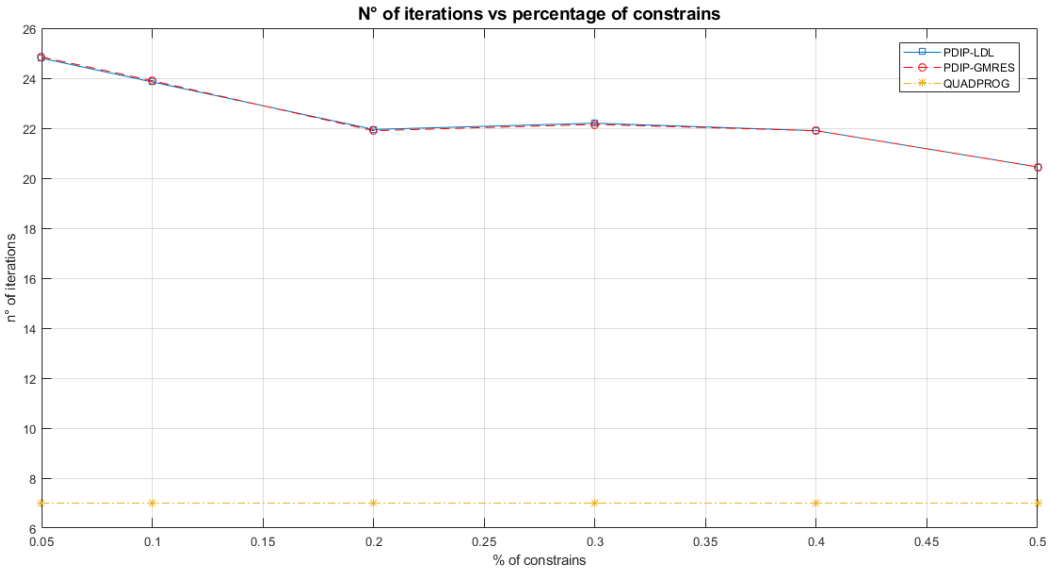


Figura 5.4: Numero di iterazioni all'aumentare di m.

5.3 Densità

In questo gruppo di esperimenti abbiamo voluto testare qualora la densità della matrice Q avesse effetti sui tempi di convergenza e il numero di iterazioni dei vari approcci coinvolti nella nostra analisi.

Da Fig. 5.5 si evince che l'aumento di δ ha un impatto critico sulle prestazioni di QUADPROG, mentre entrambe le versioni della nostra implementazione non sembrano risentire troppo dell'aumento di densità della matrice Q, con PDIP-LDL più veloce di un ordine di grandezza rispetto a PDIP-GMRES.

In Fig. 5.6, il numero di iterazioni necessarie alla convergenza rimane stabile rispetto alla variazione di densità di Q in tutti i metodi testati.

Confrontando infine i tempi di PDIP-LDL e QUADPROG in Tab. 5.5:

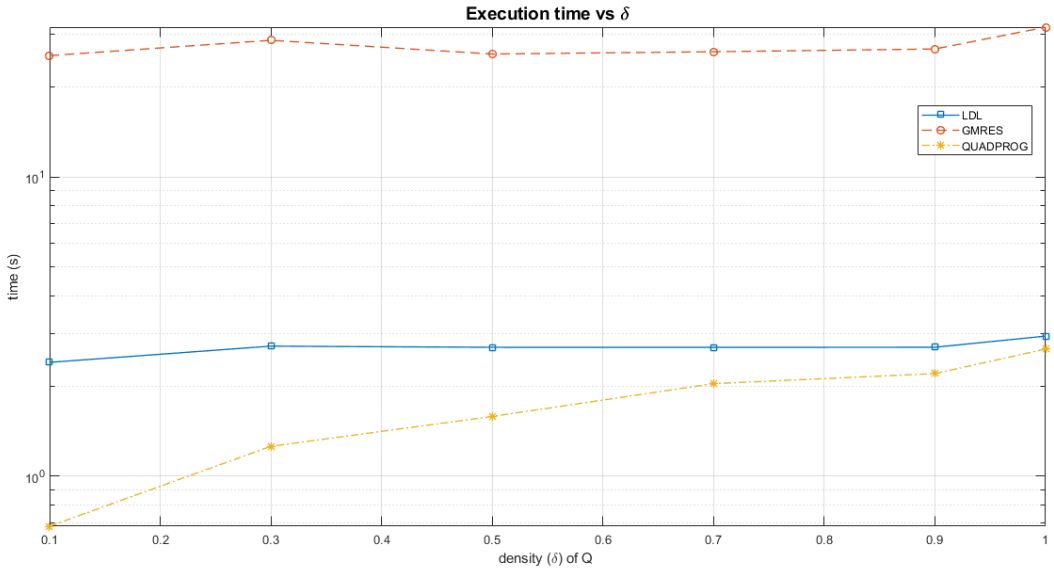


Figura 5.5: Tempo di esecuzione (in *log-scale* sull'asse delle y) dei tre metodi all'aumentare della densità (δ) di Q.

- il tempo di completamento della versione con risoluzione diretta della nostra implementazione rimane costante e stabile all'aumentare della densità
- il tempo di completamento di QUADPROG, invece, aumenta monotonamente
- di conseguenza lo slowfactor è massimo in corrispondenza del valore minimo di δ, minimo viceversa; in generale la nostra implementazione è più lenta di un fattore $\approx \times 1.8$.

density	**0.1**	**0.3**	**0.5**	**0.7**	**0.9**	**1.0**
PDIP-LDL	$2.401 \pm 1.2\%$	$2.719 \pm 2.8\%$	$2.695 \pm 2\%$	$2.694 \pm 2.2\%$	$2.696 \pm 2.3\%$	$2.938 \pm 2.4\%$
QUADPROG	$0.681 \pm 2.2\%$	$1.258 \pm 5\%$	$1.583 \pm 2.8\%$	$2.038 \pm 1.5\%$	$2.202 \pm 1.1\%$	$2.659 \pm 5.9\%$
slowfactor	**3.526**	2.161	1.701	1.321	1.224	**1.104**

Tabella 5.5: Tabella contenente media e deviazione standard dei tempi di esecuzione (s) di PDIP-LDL e QUADPROG relativi al terzo sotto-esperimento.

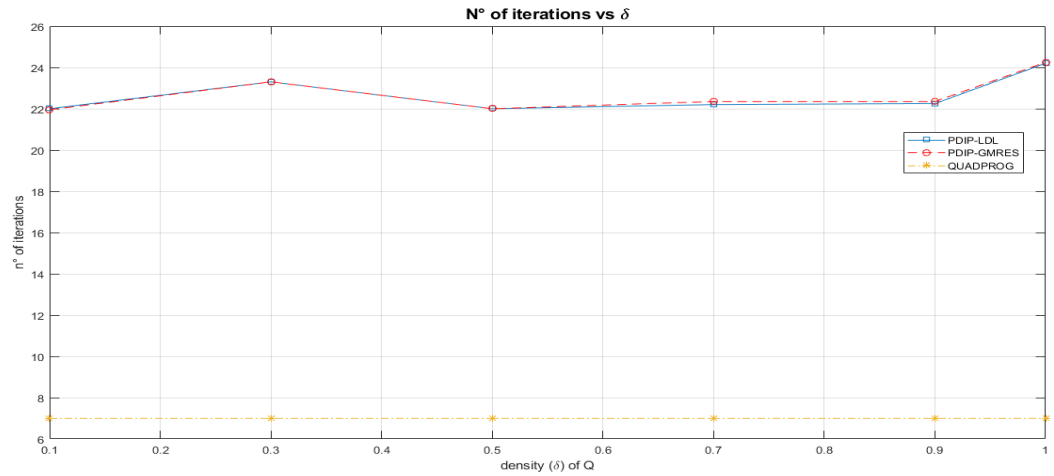


Figura 5.6: Grafico dell'andamento del numero di iterazioni all'aumentare di δ.

5.4 Convergenza e Accuratezza della Soluzione

Per i risultati in questa sezione sono state effettuate 20 esecuzioni fissando il problema come in Tab. 4.1. Per analizzare la convergenza dell'implementazione proposta, abbiamo collezionato i complementary gap ad ogni iterazione di PDIP sia in versione GMRES, che LDL.

Il grafico in Fig. 5.7 mostra come varia - in scala logaritmica sull'asse delle y - ad ogni iterazione la media dei complementary gap: in entrambe le varianti della nostra implementazione il gap decresce monotonamente ad ogni iterazione successiva alla seconda e le prestazioni sono in linea con quelle attese.

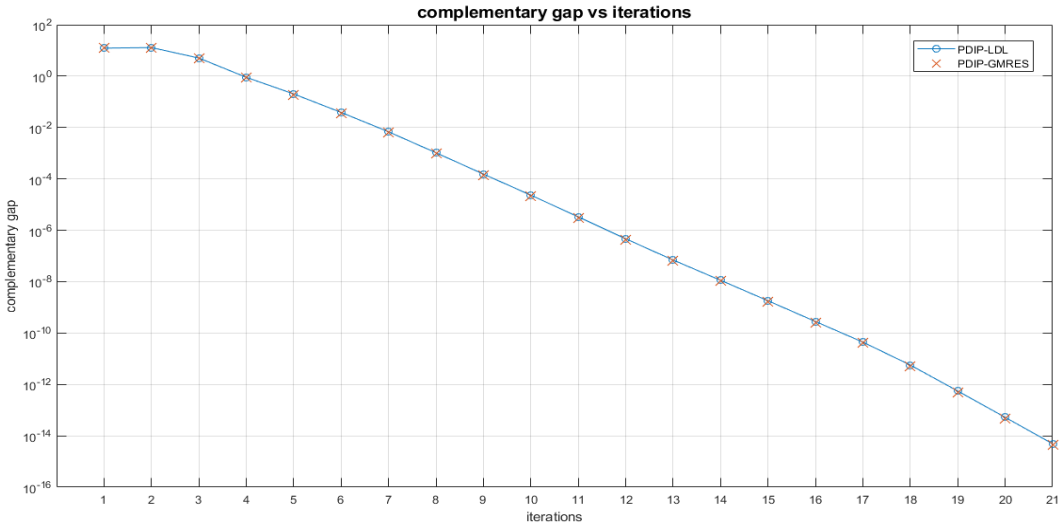


Figura 5.7: Andamento del gap (in *log-scale* sull'asse delle y) all'aumentare delle iterazioni.

La seconda parte di questo esperimento ha l'obiettivo di valutare l'accuratezza della soluzione $(x, \lambda_{eq}, \lambda_s)$ dal nostro metodo. A questo proposito confrontiamo i residui finali e le triple restituite da PDIP-GMRES e PDIP-LDL fra di loro e con quella di QUADPROG.

Come metrica di confronto per le triple abbiamo scelto di usare la norma della differenza fra i vettori soluzione $\|\delta_v\|$:

$$\|v_{M1} - v_{M2}\| \qquad \text{con} \quad v \in \{x, \lambda_{eq}, \lambda_s\} \text{ e } M_i \in \{PDIP-LDL,\ PDIP-GMRES,\ QUADPROG\} \tag{5.1}$$

In Tab. 5.6 e 5.7 sono riportate medie e deviazioni standard - su 20 esecuzioni - delle metriche, calcolate come in 5.1, rispettivamente per i vettori x e λ.

$\|\delta \mathbf{x}\|$	**PDIP-GMRES**	**QUADPROG**
PDIP-LDL	$3.3856 \cdot 10^{-13} \pm 6\%$	$4.3127 \cdot 10^{-3} \pm 0\%$
PDIP-GMRES		$4.3127 \cdot 10^{-3} \pm 0\%$

Tabella 5.6: Media e deviazione standard, sulle 20 ripetizioni, della norma della differenza fra i vettori soluzione.

$\|\delta\lambda\|$	**PDIP-LDL**		**PDIP-GMRES**	
PDIP-GMRES	$1.6224 \cdot 10^{-13} \pm 11\%$	$1.1201 \cdot 10^{-12} \pm 5.4\%$		
QUADPROG	$1.5712 \cdot 10^{-3} \pm 0\%$	$5.3517 \cdot 10^{-3} \pm 0\%$	$1.5712 \cdot 10^{-3} \pm 0\%$	$5.3517 \cdot 10^{-3} \pm 0\%$
	$\|\delta\lambda_{eqlin}\|$	$\|\delta\lambda_s\|$	$\|\delta\lambda_{eqlin}\|$	$\|\delta\lambda_s\|$

Tabella 5.7: Media e deviazione standard, sulle 20 ripetizioni, della norma della differenza fra i moltiplicatori lagrangiani soluzione dei metodi testati.

	$\|\mathbf{r_p}\|$	$\|\mathbf{r_d}\|$
PDIP-LDL	$1.1741 \cdot 10^{-15} \pm 8.9\%$	$1.3726 \cdot 10^{-14} \pm 11.4\%$
PDIP-GMRES	$9.0965 \cdot 10^{-14} \pm 13.3\%$	$1.1016 \cdot 10^{-12} \pm 5.3\%$
QUADPROG	$9.8679 \cdot 10^{-16} \pm 0\%$	$9.3234 \cdot 10^{-15} \pm 0.1\%$

Tabella 5.8: Tabella che riporta la media e deviazione standard su 20 esecuzioni della norma dei residui primali e duali finali dei tre metodi in analisi.

Le soluzioni ottenute dall'esecuzione diretta e iterativa di PDIP risultano pressocchè identiche: sia $\|\delta x\|$ che $\|\delta\lambda\|$ sono dell'ordine di 10^{-13}; ed entrambe queste soluzioni sono uguali a quella trovata da QUADPROG a meno di un fattore di 10^{-3}.

La Tabella 5.8 conferma ulteriormente la precisione della nostra implementazione:

- l'accuratezza raggiunta è paragonabile a quella di QUADPROG infatti sia il residuo primale, che duale sono maggiori di circa un ordine di grandezza rispetto alla built-in di `MATLAB`; probabilmente il nostro metodo potrebbe raggiungere precisione dello stesso ordine di grandezza di QUADPROG scegliendo una soglia più bassa rispetto a quella usata nei nostri esperimenti (10^{-14}), impiegando però più iterazioni e risultando più lento
- durante tutta la fase sperimentale abbiamo osservato deviazioni standard più alte per la nostra implementazione. Questo fenomeno può essere spiegato dalla presenza di una parte randomica nel nostro metodo, precisamente in 3.2 durante la generazione della tripla iniziale: λ_{eq} viene inizializzato con reali casuali, influenzando anche l'inizializzazione di λ_s; QUADPROG invece inizializza sempre $(x^0, \lambda_{eq}^0, \lambda_s^0) = 1$, con risultati finali di conseguenza più simili fra loro.

Concludendo la nostra implementazione, sia in versione diretta, che iterativa, può essere quindi considerata stabile e accurata rispetto alla built-in di `MATLAB` sia in termini di tempi ed iterazioni di convergenza, che di precisione del risultato finale.

RIFERIMENTI BIBLIOGRAFICI

[1] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer, New York, 2 edition, 2006.

[2] L.N. Trefethen and D. Bau. *Numerical Linear Algebra*. Other Titles in Applied Mathematics. Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104), 1997.