Controllo dei sistemi energetici

Ottimizzazione statica e dinamica
Parte prima

Ing. Alessandro Pisano
apisano@unica.it

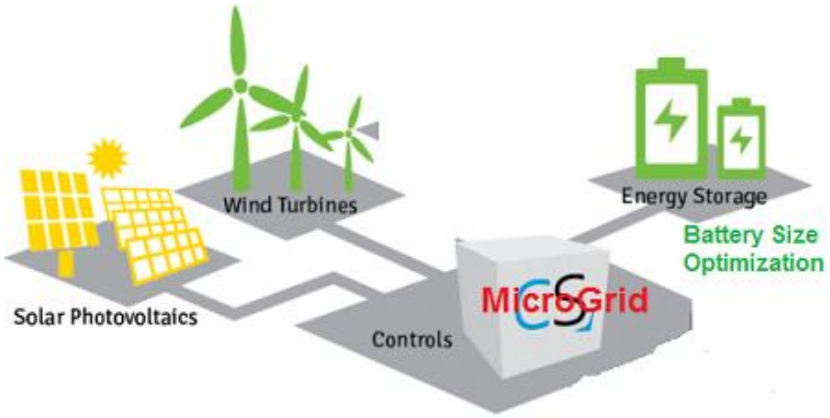
In un problema di ottimizzazione viene massimizzata o minimizzata una "**funzione obiettivo**" che dipende da un insieme di "**variabili di decisione**" soggette a dei **vincoli**.

L'impiego di algoritmi e tecniche di ottimizzazione va assumendo una rilevanza sempre maggiore in pressochè tutte le discipline scientifiche, principalmente grazie al fatto che:

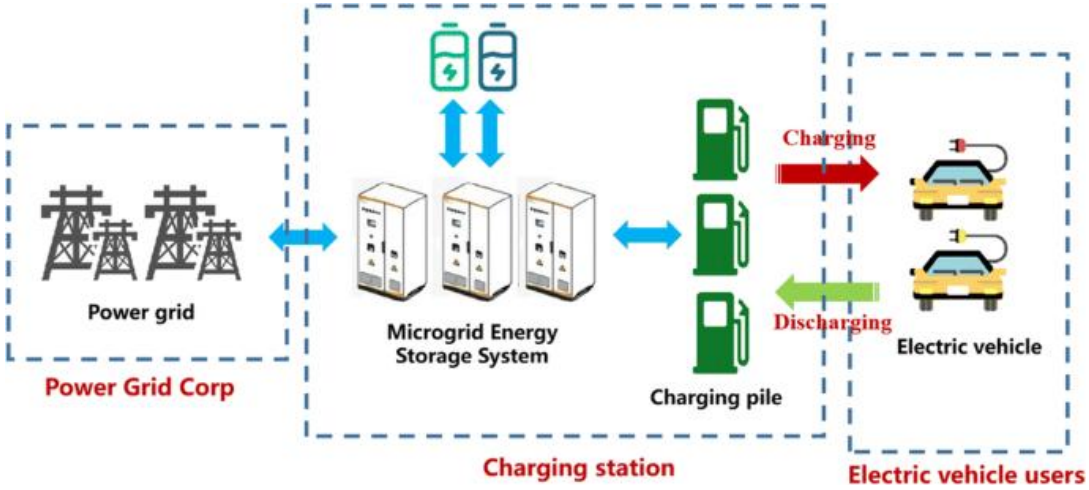
- Una ampia comunita scientifica lavora continuamente per sviluppare algoritmi risolutivi sempre piu efficaci
- La potenza dei moderni calcolatori rende oggi risolubile con relativa semplicità classi di problemi di ottimizzazione che, per complessità e dimensione, non erano trattabili fino a pochi anni fa se non mediante supercalcolatori.

Le applicazioni di tali problemi e algoritmi nell'ambito dell'ingegneria energetica sono infinite:

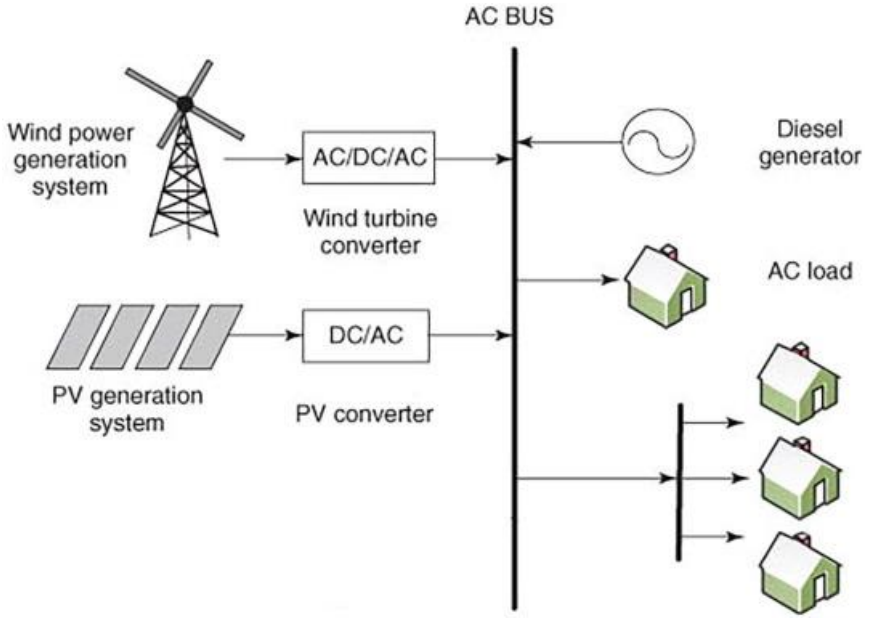
Ottimizzare la capacita di un sistema di storage in relazione ai costi fissi e variabili di esercizio, e sviluppare strategie per l'impiego ottimale dello stesso in relazione alla variabilità dei costi energetici

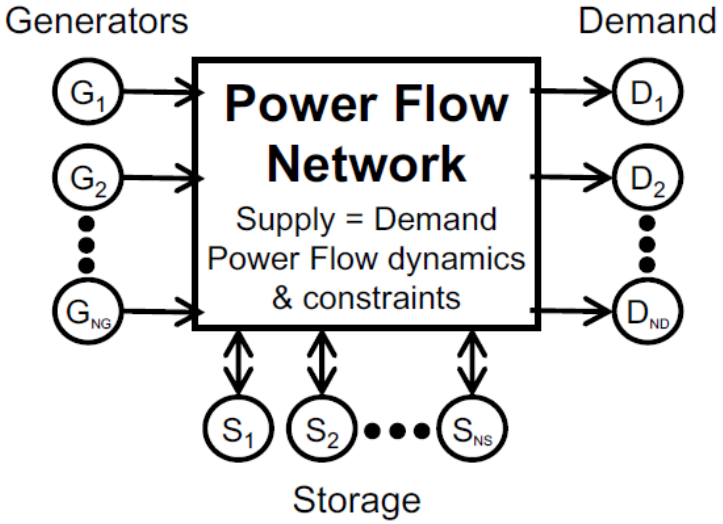


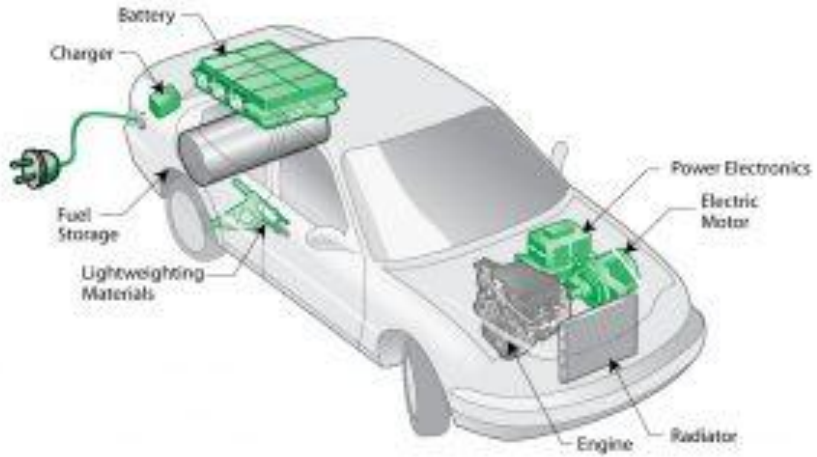
Ottimizzare le procedure di ricarica di veicoli elettrici per minimizzare i costi di gestione garantendo soglie predefinite di operatività della flotta



Pianificare la produzione di un gruppo di generatori ("dispacciamento") per minimizzare i costi di produzione con un profilo di produzione assegnato, o in alternativa per produrre ed immettere in rete energia nelle fascie orarie di maggiore redditività sotto vincoli di disponibilità di combustibile (e in condizioni di inevitabile incertezza per quanto concerne la generazione da fonte rinnovabile)







Programmazione non lineare - Forma generale

I problemi di ottimizzazione statica a **obiettivo singolo** possono essere astratti nella seguente forma canonica generale

Minimizzare	$f(x)$	
Sotto i vincoli	$g_i(x) \leq 0$	$i = 1,2,..,m$
	$h_j(x) = 0$	$j = 1,2,..,\ell$

$f(x)\colon \mathbb{R}^n \to \mathbb{R}$	**funzione obiettivo**
$x \in \mathbb{R}^n$	**variabili di decisione**
$g_i(x) \leq 0$	*Vincoli di diseguaglianza*
$h_j(x) = 0$	*Vincoli di uguaglianza*

$x^* = soluzione\ ottima$

$f(x^*) = costo\ ottimale\ (minimo)$

In versione compatta:

Minimizzare $f(x)$

Sotto i vincoli $\boldsymbol{g}(x) \leq 0 \qquad i = 1,2,..,m$

$\boldsymbol{h}(x) = 0 \qquad j = 1,2,..,\ell$

$$g(x) = \begin{bmatrix} g_1(x) \\ g_2(x) \\ \vdots \\ g_m(x) \end{bmatrix} \qquad h(x) = \begin{bmatrix} h_1(x) \\ h_2(x) \\ \vdots \\ h_\ell(x) \end{bmatrix}$$

Un problema di massimizzazione può essere riformulato in termini di un problema di minimizzazione equivalente in cui la funzione obiettivo ha il segno meno davanti

$$\max_x\{f(x)\} = \min_x\{-f(x)\}$$

Risoluzione mediante Matlab di problemi di programmazione non lineare a obiettivo singolo

Un generico problema di ottimizzazione vincolata può essere risolto in Matlab dalla funzione **`fmincon.`**

La funzione `fmincon` tratta il seguente problema, riformulato in termini più variegati rispetto alla forma canonica precedentemente illustrata

$$\min_x f(x) \text{ such that } \begin{cases} c(x) \le 0 \\ ceq(x) = 0 \\ A \cdot x \le b \\ Aeq \cdot x = beq \\ lb \le x \le ub, \end{cases}$$

vincolo non lineare di disuguaglianza

vincolo non lineare di eguaglianza

vincolo lineare di diseguaglianza

vincolo lineare di uguaglianza (equality)

limiti superiore (upper bound) e inferiore (lower bound)

Ad esempio, un vincolo non lineare di disuguaglianza per un problema di ottimizzazione che coinvolge tre variabili x_1 , x_2 e , x_3 potrebbe essere

$$x_1^2 + x_2^2 + x_3^2 \le 1$$

che significa imporre che la soluzione debba essere ricercata all'interno della sfera di raggio unitario centrata nell'origine.

Risoluzione mediante Matlab di problemi di programmazione non lineare a obiettivo singolo

$$\min_x f(x) \text{ such that } \begin{cases} c(x) \le 0 \\ ceq(x) = 0 \\ A \cdot x \le b \\ Aeq \cdot x = beq \\ lb \le x \le ub, \end{cases}$$

vincolo non lineare di disuguaglianza

vincolo non lineare di eguaglianza

vincolo lineare di diseguaglianza

vincolo lineare di uguaglianza (equality)

limiti superiore (upper bound) e inferiore (lower bound)

Sintassi di impiego della funzione `fmincon`

```
X = fmincon(f,x0,A,b,Aeq,beq,lb,ub,nonlcon)
```

`nonlcon` = $[c(x) \quad c_{eq}(x)]$ Function che riceve in ingresso x e restituisce le quantità $c(x)$ e $c_{eq}(x)$

x_0 = punto di partenza della procedura di ricerca iterativa dell'ottimo. Deve soddisfare i vincoli.

Esempio

$$f(x_1, x_2) = 100\left(x_2 - x_1^2\right)^2 + (1 - x_1)^2$$

Funzione di Rosenbrock. Ha un minimo globale che vale 0 nel punto (1,1).

Rappresenta una funzione complicata da minimizzare, e come tale è usata in letteratura come benchmark per porre a confronto fra di loro algoritmi differenti.

Curve di livello

«banana function»

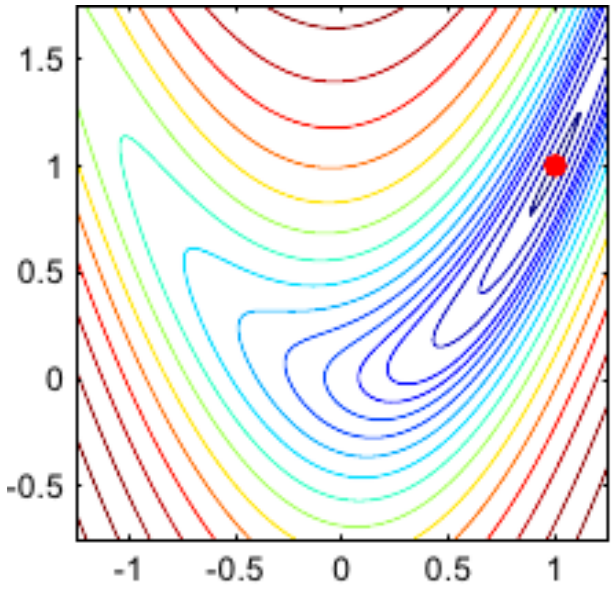
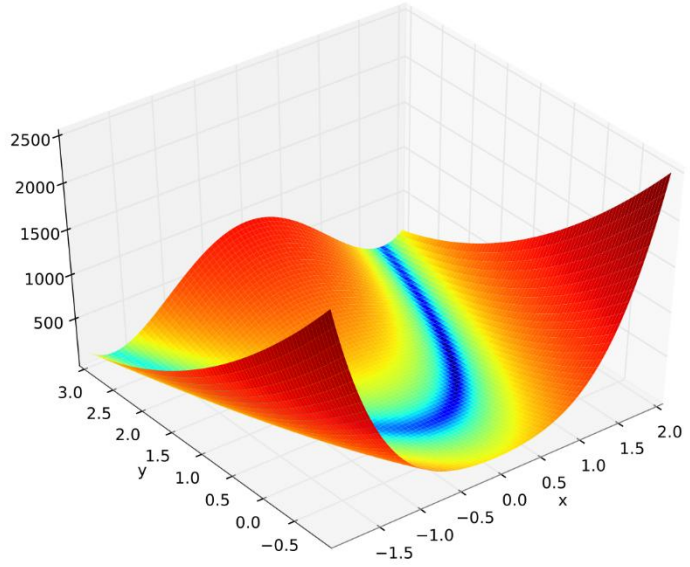


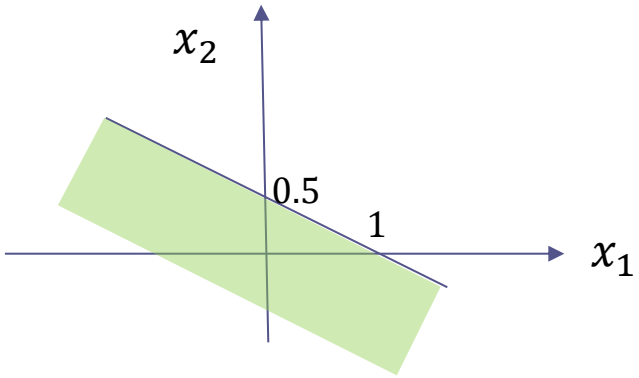
Grafico 3D



Caso 1

Minimizziamo la Funzione di Rosenbrock $f(x_1, x_2) = 100\left(x_2 - x_1^2\right)^2 + (1 - x_1)^2$
sotto il vincolo $x_1 + 2x_2 \leq 1$

Intepretazione geometrica (dominio convesso)



Vincolo lineare di diseguaglianza

$Ax \leq b$

$A = [1\ 2]$

$b = 1$

Script che risolve il problema (**Ros.m**):

```
Rosfun = @(x)100*(x(2) - x(1)^2)^2 + (1 - x(1))^2;
x0 = [-1,2];
lb = [];
ub = [];
A = [1 2];
b = 1;
Aeq = [];
beq = [];
nonlcon=[];
[x,fval] = fmincon(Rosfun,x0,A,b,Aeq,beq,lb,ub,nonlcon)
```

Si definiscono unicamente il punto iniziale e gli array costanti A e b

Output dello script nel prompt di Matlab

```
Local minimum found that satisfies the constraints.

Optimization completed because the objective function is non-decreasing in
feasible directions, to within the value of the optimality tolerance,
and constraints are satisfied to within the value of the constraint tolerance.

<stopping criteria details>

x =

    0.5022    0.2489

fval =

    0.2489
```

Soluzione ottima

Valore ottimale della funzione obiettivo

Caso 2

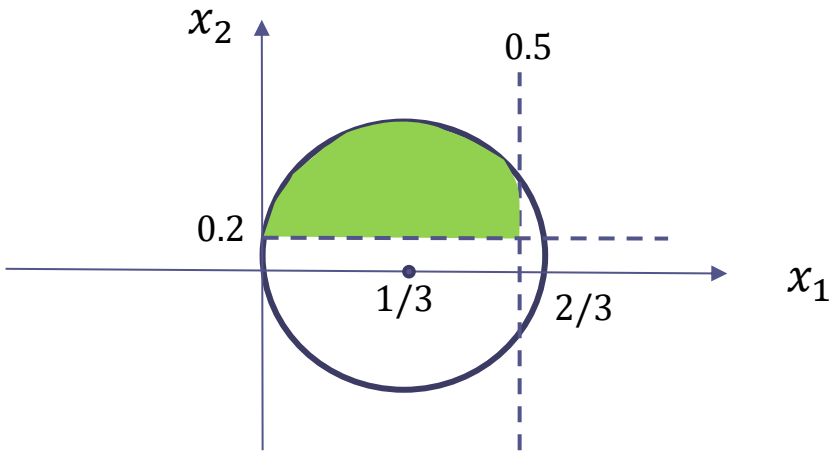
Minimizziamo la Funzione di Rosenbrock $f(x_1, x_2) = 100\left(x_2 - x_1^2\right)^2 + (1 - x_1)^2$
sotto i vincoli:

$$0 \leq x_1 \leq 0.5 \qquad 0.2 \leq x_2 \leq 0.8 \qquad \left(x_1 - \frac{1}{3}\right)^2 + x_2^2 < \left(\frac{1}{3}\right)^2$$

Esprimiamo il vincolo non lineare di diseguaglianza in forma standard $c(x) \leq 0$

$$\left(x_1 - \frac{1}{3}\right)^2 + x_2^2 < \left(\frac{1}{3}\right)^2 \Rightarrow \left(x_1 - \frac{1}{3}\right)^2 + x_2^2 - \left(\frac{1}{3}\right)^2 < 0 \Rightarrow c(x) = \left(x_1 - \frac{1}{3}\right)^2 + x_2^2 - \left(\frac{1}{3}\right)^2$$

Intepretazione geometrica



I vincoli restringono la ricerca del punto di minimo ad una regione del piano, evidenziata in verde. E' una regione convessa.

minimizzare $f(x_1, x_2) = 100\left(x_2 - x_1^2\right)^2 + (1 - x_1)^2$

sotto i vincoli $0 \leq x_1 \leq 0.5$

$0.2 \leq x_2 \leq 0.8$

$\left(x_1 - \frac{1}{3}\right)^2 + x_2^2 - \left(\frac{1}{3}\right)^2 < 0$

Function file che implementa i vincoli non lineari di disuguaglianza e di uguaglianza

```
function [c,ceq] = nonlcon(x)
c = (x(1)-1/3)^2 + (x(2)-1/3)^2 - (1/3)^2;
ceq = [];
end
```

Function file: nonlcon.m

minimizzare $f(x_1, x_2) = 100\left(x_2 - x_1^2\right)^2 + (1 - x_1)^2$

sotto i vincoli $0 \leq x_1 \leq 0.5$

$0.2 \leq x_2 \leq 0.8$

$\left(x_1 - \frac{1}{3}\right)^2 + x_2^2 - \left(\frac{1}{3}\right)^2 < 0$

Script di ottimizzazione (Ros2.m)

```
Rosfun = @(x)100*(x(2) - x(1)^2)^2 + (1 - x(1))^2;
x0 = [1/4,1/4]
lb = [0,0.2];
ub = [0.5,0.8];
A = [];
b = [];
Aeq = [];
beq = [];
nonlcon=@nonlcon;
[x,fval] = fmincon(Rosfun,x0,A,b,Aeq,beq,lb,ub,nonlcon)
```

```
x =

    0.5000    0.2500
```

Anche ammesso che il problema trattato ammetta soluzione (che potrà essere unica o meno) non è garantito che i solutori numerici implementati nella funzione `fmincon` siano in grado di determinarla.

Ciò per effetto del fatto che il problema in forma generale non ha alcuna struttura.

Deve essere inoltre passato al solutore un «punto di partenza» `x0` che soddisfi i vincoli e che «non disti troppo» dal punto ottimale che risolve il problema di ottimizzazione. Ciò non è immediato per problemi complessi.

Il problema generale ammette tutta una serie di sottoclassi, in cui la funzione obiettivo e/o i vincoli assumono forme particolari (ad esempio sono funzioni lineari o quadratiche). Tali sottoclassi sono fondamentali in quanto la forma strutturata del relativo problema fa si che siano disponibili solutori dedicati estremamente efficienti, che garantiscono che la soluzione sarà certamente trovata anche per problemi di dimensione molto elevata (in problemi di interesse il numero di variabili di decisione può essere dell'ordine delle migliaia, decine o centinaia di migliaia, e anche di più) e con un tempo di calcolo «accettabile».

E' pertanto indispensabile e fondamentale valutare se sulla base delle caratteristiche del problema in esame, questo possa essere ricondotto (eventualmente mediante sue semplificazioni) ad una delle sottoclassi con solutori dedicati.

Una importante sotto-classe addizionale di problemi è quella in cui le variabili di decisione (o una parte di queste) sono variabili intere (o eventualmente binarie)

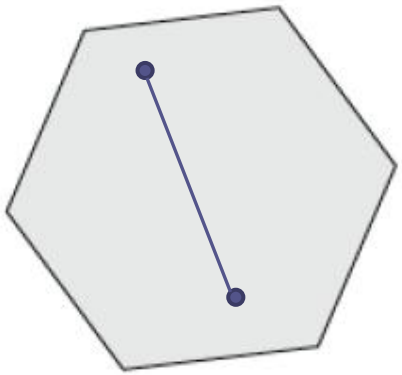
Tali problemi vengono classificati come «**mixed-integer**» (MI) e sono particolarmente complessi da risolvere se la funzione obiettivo e/o i vincoli sono espressi mediante funzioni non lineari delle variabili di decisione. Ne parleremo più avanti e faremo vari esempi.

Un importante concetto che permea l'ambito dell'ottimizzazione, sia in relazione alla esistenza della soluzione che alla disponibilità di algoritmi efficienti per trovarla, è quello di **convessità**

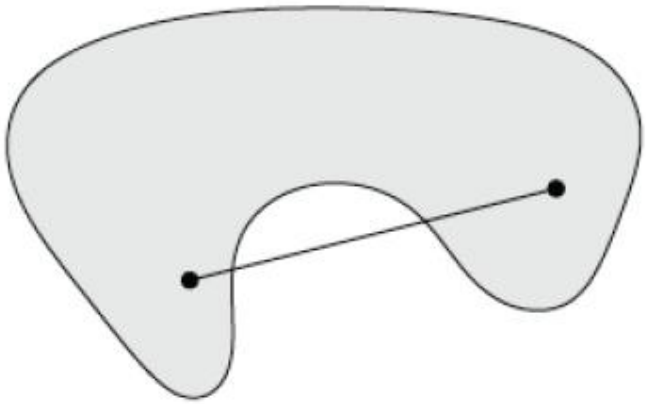
Si può affermare che nel momento in cui il dominio ammissibile D individuato dai vincoli è convesso, e la funzione obiettivo è convessa in D, allora si ha la certezza che il problema di ottimizzazione possa essere risolto senza particolari difficolta algoritmiche anche per dimensioni molto elevate del problema.

Domini (insiemi) convessi e non convessi

Un dominio D è convesso se considerando due punti arbitrari del dominio, la linea che unisce i due punti appartiene interamente al dominio D.



dominio convesso



dominio non convesso

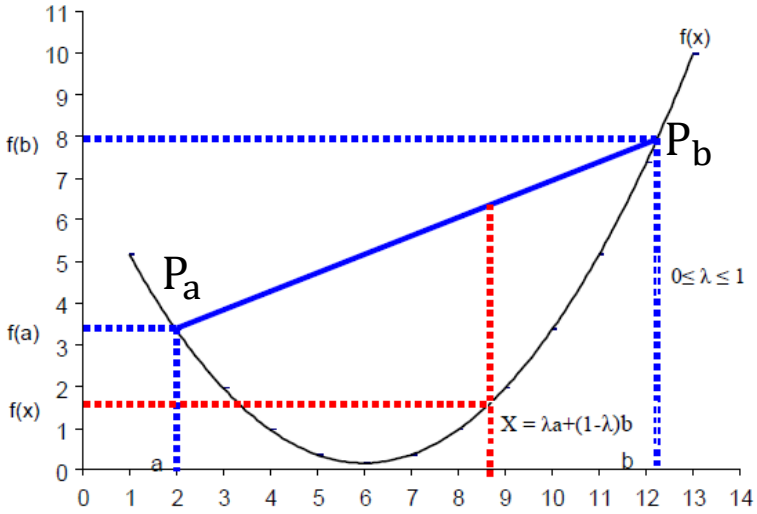
Funzioni convesse e non convesse

Sia D un dominio convesso e siano $a, b \in D$ due punti arbitrari del dominio.

La funzione $f(x)\colon D \to \mathbb{R}$ è convessa se e solo se per ogni numero reale $0 \leq \lambda \leq 1$ vale la seguente relazione

$$f(\lambda\, a + (1-\lambda)b) \leq \lambda f(a) + (1-\lambda) f(b)$$

In termini geometrici, una funzione è convessa in un dominio se la linea che unisce i punti $\mathrm{P_a} = (\mathrm{a}\,, f(a))$ e $\mathrm{P_b} = (\mathrm{b}\,, f(b))$ si trova sempre sopra la funzione.



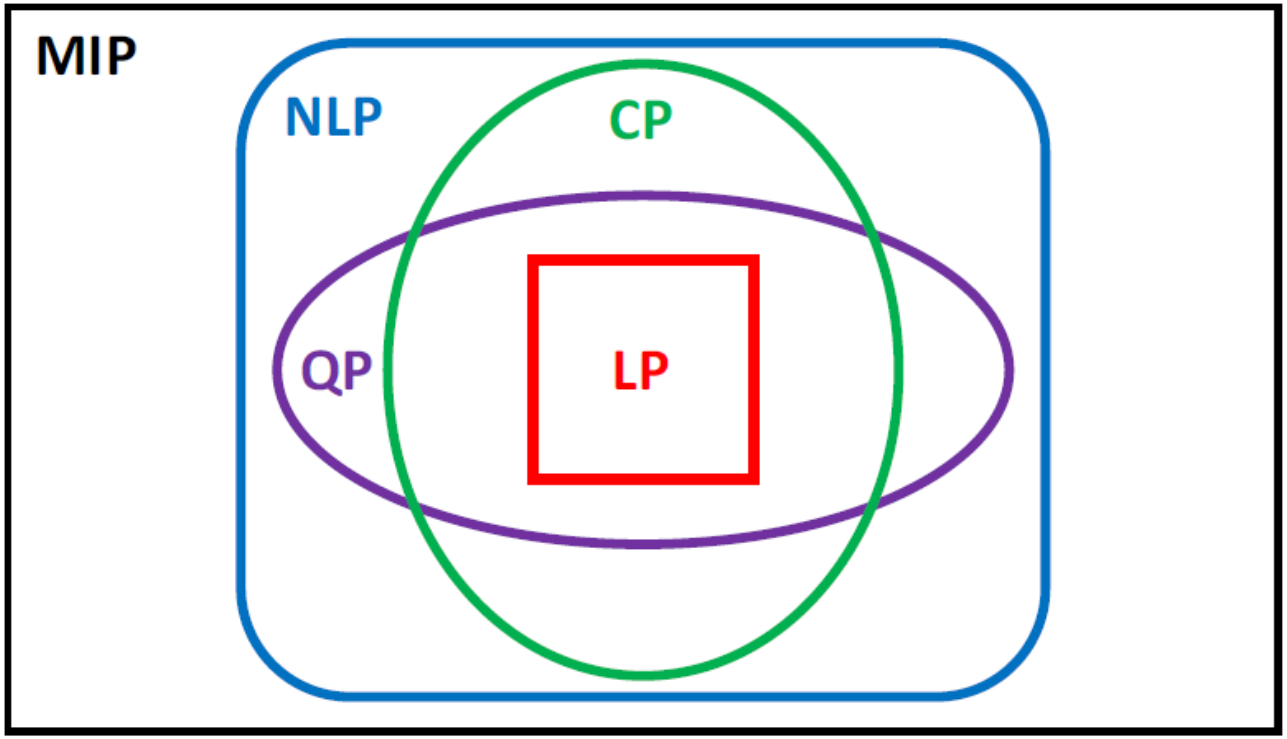
Funzione convessa di una variabile

N.B. Anche una funzione lineare $f(x) = Ax + b$ soddisfa la condizione di convessità

Principali classi di problemi di ottimizzazione

LP	Linear programming
QP	Quadratic Programming
QCQP	Quadratically-Constrained Quadratic Programming
MILP	Mixed-integer Linear programming
MIQP	Mixed-Integer Quadratic Programming
SOCP	Second-Order Cone Programming

CP	Convex Programming	*Funzione obiettivo e dominio ammissibile convessi*
NLP	Nonlinear Programming	*Il caso più generale. Nessuna ipotesi sulla funzione obiettivo e sui vincoli.*


MIP
NLP
CP
QP
LP

Programmazione lineare

LP

Problema generale

$$\min_x c^T x \qquad x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^n$$

Decision variables

Vincoli

$Ax \leq b$ — *Vincoli di diseguaglianza*

$A_{eq}x = b_{eq}$ — *Vincoli di uguaglianza*

c è un vettore colonna di n elementi (in modo che $c^T x$ sia uno scalare).

A, A_{eq} sono matrici che hanno n colonne.

b, b_{eq} sono vettori che hanno tanti elementi quante sono, rispettivamente, le righe di $A, ed\ A_{eq}$.

Programmazione quadratica

QP

Problema generale

$$\min_x \frac{1}{2} x^T H x + c^T x \qquad x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^n$$

Decision variables

Vincoli

$Ax \leq b$ — *Vincoli di diseguaglianza*

$A_{eq} x = b_{eq}$ — *Vincoli di uguaglianza*

Un problema di tipo QP si è caratterizzato da una funzione obiettivo quadratica e da funzioni di vincolo lineari (più precisamente, affini). Se anche le funzioni di vincolo contengono termini quadratici si considera una classe di problemi più generale denominata QCQP (Quadratically-Constrained Quadratic Programming)

La funzione obiettivo è convessa se e solo se la matrice $\frac{H+H^T}{2}$ è semi-definita positiva, cioè tutti i suoi autovalori (che sono reali) sono maggiori o uguali a zero.

Programmazione quadratica con vincoli quadratici

QCQP

Problema generale

$$\min_x \ \frac{1}{2}x^T Hx + c^T x \qquad x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^n$$

Decision variables

Vincoli

$$\frac{1}{2}x^T Qx + Ax \leq b$$

Vincoli di diseguaglianza

$$\frac{1}{2}x^T Q_{eq}x + A_{eq}x = b_{eq}$$

Vincoli di uguaglianza

Programmazione lineare mixed-integer

MILP

Problema generale

$$\min_x \; c^T x \qquad \textit{Funzione obiettivo}$$

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \qquad \textbf{Decision variables}$$

$$x_i \in \mathbb{R} \qquad i \in V^R$$

$$x_i \in \mathbb{Z} \qquad i \in V^Z$$

$$V^R \cap V^Z = \emptyset$$

$$V^R \cup V^Z = V = \{1,2,\dots,n\}$$

Vincoli

$$Ax \leq b \qquad \textit{Vincoli di diseguaglianza}$$

$$A_{eq}x = b_{eq} \qquad \textit{Vincoli di uguaglianza}$$

Un sottoinsieme delle variabili di decisione può assumere valori reali, mentre la restante parte è ristretta ad assumere valori interi. Il dominio D di ammissibilità della soluzione è non convesso.

Programmazione quadratica mixed-integer

MIQP

Problema generale

$$\min_x \ \frac{1}{2}x^T Hx + c^T x \qquad \textit{Funzione obiettivo}$$

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \qquad \textbf{Decision variables}$$

$$x_i \in \mathbb{R} \qquad i \in V^R$$

$$x_i \in \mathbb{Z} \qquad i \in V^Z$$

$$V^R \cap V^Z = \emptyset$$

$$V^R \cup V^Z = V = \{1,2,\dots,n\}$$

Vincoli

$$Ax \leq b \qquad \textit{Vincoli di diseguaglianza}$$

$$A_{eq}x = b_{eq} \qquad \textit{Vincoli di uguaglianza}$$

L'unica differenza dal problema MILP è che la funzione obiettivo è quadratica

La funzione obiettivo è convessa se e solo se la matrice $\frac{H+H^T}{2}$ è semi-definita positiva, cioè tutti i suoi autovalori (che sono reali) sono maggiori o uguali a zero.

Esempio preliminare

$$\min_{x_1,x_2}\left\{-x_1-\frac{x_2}{3}\right\} \qquad x=\begin{bmatrix}x_1\\x_2\end{bmatrix} \qquad c=\begin{bmatrix}-1\\-\frac{1}{3}\end{bmatrix} \qquad f(x)=c^Tx=\begin{bmatrix}-1 & -\frac{1}{3}\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}$$

Funzione obiettivo lineare

Vincoli

Vincoli di diseguaglianza

$$x_1+x_2\le 2$$

$$x_1+\frac{1}{4}x_2\le 1$$

$$x_1-x_2\le 2$$

$$\frac{x_1}{4}+x_2\ge -1$$

$$-x_1-x_2\le -1$$

$$-x_1+x_2\le 2$$

$$|x_1|\le 2$$

$$-1\le x_2\le 3$$

Vincoli di uguaglianza

$$x_1+\frac{x_2}{4}=\frac{1}{2}$$

Vincoli lineari

Rappresentiamo i vincoli nella forma standard

$Ax \leq b$ *Vincoli di diseguaglianza*

$A_{eq}x = b_{eq}$ *Vincoli di uguaglianza*

Rappresentiamo i vincoli nella forma standard

$Ax \leq b$ *Vincoli di diseguaglianza*

$A_{eq}x = b_{eq}$ *Vincoli di uguaglianza*

Vincoli di diseguaglianza

$x_1 + x_2 \leq 2$

$x_1 + \frac{1}{4}x_2 \leq 1$

$x_1 - x_2 \leq 2$

$\frac{x_1}{4} + x_2 \geq -1 \rightarrow -\frac{x_1}{4} - x_2 \leq 1$

$-x_1 - x_2 \leq -1$

$-x_1 + x_2 \leq 2$

Esprimiamo i vincoli di diseguaglianza mediante relazioni aventi unicamente il simbolo $\leq$

$|x_1| \leq 2 \rightarrow \begin{cases} x_1 \leq 2 \\ x_1 \geq -2 \end{cases} \rightarrow \begin{cases} x_1 \leq 2 \\ -x_1 \leq 2 \end{cases}$

$-1 \leq x_2 \leq 3 \rightarrow \begin{cases} x_2 \leq 3 \\ -x_2 \leq 1 \end{cases}$

Determiniamo le matrici A e b che definiscono il vincolo di diseguaglianza

$$x_1 + x_2 \leq 2$$
$$x_1 + \frac{1}{4}x_2 \leq 1$$
$$x_1 - x_2 \leq 2$$
$$-\frac{x_1}{4} - x_2 \leq 1$$
$$-x_1 - x_2 \leq -1$$
$$-x_1 + x_2 \leq 2$$
$$x_1 \leq 2$$
$$-x_1 \leq 2$$
$$x_2 \leq 3$$
$$-x_2 \leq 1$$

$Ax \leq b$ *Vincoli di diseguaglianza*

$$A = \begin{bmatrix} 1 & 1 \\ 1 & \frac{1}{4} \\ 1 & -1 \\ -\frac{1}{4} & -1 \\ -1 & -1 \\ -1 & 1 \\ 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix} \qquad b = \begin{bmatrix} 2 \\ 1 \\ 2 \\ 1 \\ -1 \\ 2 \\ 2 \\ 2 \\ 3 \\ 1 \end{bmatrix}$$

$A_{eq}x = b_{eq}$ *Vincoli di uguaglianza*

$x_1 + \frac{x_2}{4} = \frac{1}{2}$ $A_{eq} = \begin{bmatrix} 1 & \frac{1}{4} \end{bmatrix}$ $b_{eq} = \frac{1}{2}$

Il problema di ottimizzazione lineare è completamente caratterizzato dalle particolari matrici e vettori c, A, b, A_{eq}, b_{eq} che lo definiscono

Un problema di programmazione lineare si risolve in Matlab mediante la funzione `linprog` (inclusa nel Toolbox «Optimization»)

Sintassi di base

```
x=linprog(c,A,b,Aeq,beq)
```

Se non vi sono vincoli di uguaglianza:

```
x=linprog(c,A,b)
```

Se vi sono unicamente vincoli di uguaglianza:

```
x=linprog(c,[],[],Aeq,beq)
```

Sebbene i vincoli del tipo

$$x_{im} \leq x_i \leq x_{iM}$$

Possano essere facilmente inglobati all'interno del termine $Ax \leq b$, la funzione `linprog` consente di passare i valori minimi e massimi delle variabili di decisione come argomenti aggiuntivi in ingresso

```
x=linprog(c,A,b,Aeq,beq,lb,ub)
```

upper bound

lower bound

```
lb=[x1m x2m … xnm]

ub=[x1M x2M … xnM]
```

Risolviamo in Matlab il problema di esempio precedentemente illustrato

Iniziamo dal sottoproblema che include unicamente i vincoli di diseguaglianza

$$\min_{x_1,x_2}\left\{-x_1-\frac{x_2}{3}\right\} \qquad x=\begin{bmatrix}x_1\\x_2\end{bmatrix} \qquad c=\begin{bmatrix}-1\\-\frac{1}{3}\end{bmatrix} \qquad f(x)=c^Tx=\begin{bmatrix}-1 & -\frac{1}{3}\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}$$

Vincoli

$$x_1+x_2\le 2$$
$$x_1+\frac{1}{4}x_2\le 1$$
$$x_1-x_2\le 2$$
$$\frac{x_1}{4}+x_2\ge -1$$
$$-x_1-x_2\le -1$$
$$-x_1+x_2\le 2$$
$$|x_1|\le 2$$
$$-1\le x_2\le 3$$

$$Ax\le b$$

$$A=\begin{bmatrix}1 & 1\\ 1 & \frac{1}{4}\\ 1 & -1\\ -\frac{1}{4} & -1\\ -1 & -1\\ -1 & 1\\ 1 & 0\\ -1 & 0\\ 0 & 1\\ 0 & -1\end{bmatrix} \qquad b=\begin{bmatrix}2\\1\\2\\1\\-1\\2\\2\\2\\3\\1\end{bmatrix}$$

Script file: EsPrel01.m

```
clear all
close all
clc

c=[-1 -1/3];

A=[1 1;1 1/4;1 -1;-1/4 -1;-1 -1;-1 1;1 0;-1 0;0 1;0 -1];
b=[2;1;2;1;-1;2;2;2;3;1];


x=linprog(c,A,b)
```

Valori ottimali:

```
Optimal solution found.


x =

    0.6667
    1.3333
```

Imponiamo i valori minimi e massimi delle variabili di decisione attraverso gli argomenti di ingresso aggiuntivi `lb,ub` alla funzione `linprog`

A e b assumono forme più semplici

```
clear all
close all
clc
c=[-1 -1/3];

A=[1 1;1 1/4;1 -1;-1/4 -1;-1 -1;-1 1];
b=[2;1;2;1;-1;2];

lb=[-2 -1];
ub=[2 3];

x=linprog(c,A,b,[],[],lb,ub)
```

$$A = \begin{bmatrix} 1 & 1 \\ 1 & \frac{1}{4} \\ 1 & -1 \\ -\frac{1}{4} & -1 \\ -1 & -1 \\ -1 & 1 \end{bmatrix} \qquad b = \begin{bmatrix} 2 \\ 1 \\ 2 \\ 1 \\ -1 \\ 2 \end{bmatrix}$$

Script file: EsPrel02.m

Ora risolviamo il problema completo comprendente anche il vincolo di uguaglianza

$$x_1 + \frac{x_2}{4} = \frac{1}{2} \quad \Rightarrow \quad A_{eq} = \begin{bmatrix} 1 & \frac{1}{4} \end{bmatrix} \qquad b_{eq} = \frac{1}{2}$$

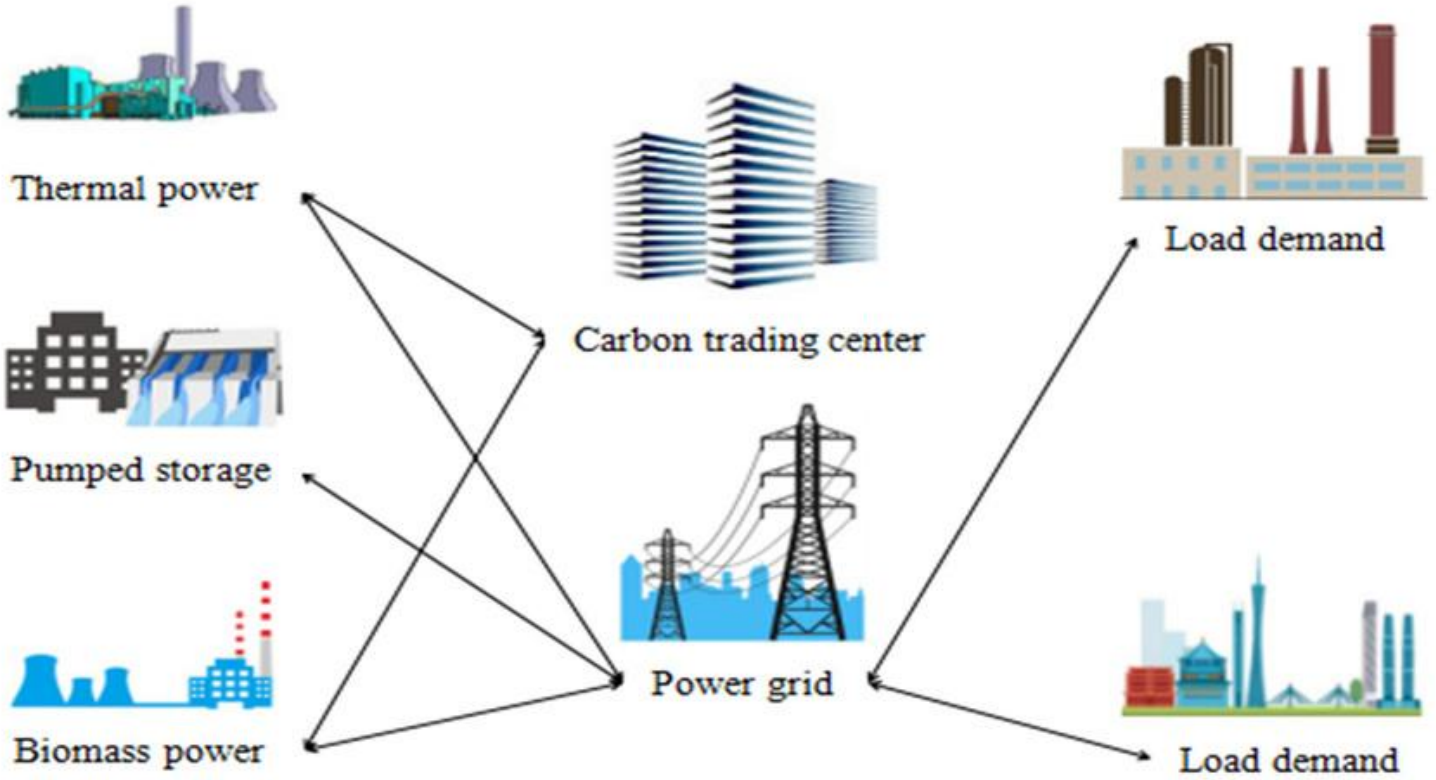
```
clear all
close all
clc
c=[-1 -1/3];
A=[1 1;1 1/4;1 -1;-1/4 -1;-1 -1;-1 1;1 0;-1 0;0 1;0 -1];
b=[2;1;2;1;-1;2;2;2;3;1];

A=[1 1;1 1/4;1 -1;-1/4 -1;-1 -1;-1 1];
b=[2;1;2;1;-1;2];
lb=[-2 -1];
ub=[2 3];

Aeq=[1 1/4];
beq=1/2;

x=linprog(c,A,b,Aeq,beq,lb,ub)
```

```
Optimal solution found.


x =

     0
     2
```

Script file: EsPrel03.m

Esempio

Dispacciamento energetico ottimale di un set di generatori



Esempio

Dispacciamento energetico ottimale di un set di generatori

Si desidera schedulare la produzione giornaliera di $n = 4$ generatori in modo da soddisfare una richiesta prefissata di produzione minimizzando i costi di produzione

Più precisamente, si desidera determinare la potenza p_i con cui dovrà operare ogni generatore $(i = 1,2,..,n)$ in ciascuna delle 24 ore del giorno successivo

Dati del problema:

Per ogni generatore è assegnato il costo marginale di produzione

$$c_{mi} \qquad \left[\frac{€}{MWh}\right] \qquad i = 1,2,3,4$$

Ciascun generatore può erogare un valore minimo e massimo di potenza

$$P_{mi} \leq p_i \leq P_{Mi} \qquad [MW] \qquad i = 1,2,3,4$$

E' assegnata la richiesta di potenza (carico) in ogni fascia oraria

$$D(k) \qquad [MW] \qquad k = 1,2,\dots,24$$

Variabili di decisione:

$$p_i(k) \qquad i = 1,2,3,4 \qquad k = 1,2,\dots,24$$

96 variabili da determinare

Funzione di costo da minimizzare:

$$J = \sum_{k=1}^{24} \sum_{i=1}^{4} c_{mi}\, p_i(k) = \sum_{k=1}^{24} J_k \qquad J_k = \sum_{i=1}^{4} c_{mi}\, p_i(k)$$

Vincoli

$$P_{mi} \le p_i(k) \le P_{Mi} \qquad \forall\, i = 1,2,3,4 \qquad \forall\, k = 1,2,\dots,24$$

$$\sum_{i=1}^{4} p_i(k) = D(k) \qquad \forall\, k = 1,2,\dots,24$$

Osserviamo che il problema di ottimizzazione è disaccoppiato dal punto di vista temporale. Possiamo pertanto determinare separatamente il mix ottimale di produzione in ciascuna delle 24 ore del giorno minimizzando separatamente le funzioni obiettivo J_k, e successivamente combinare fra loro i risultati.

Il problema di ottimizzazione complessivo è pertanto decomposto in 24 problemi fra loro disaccoppiati, all'interno dei quali l'indice temporale k assume un valore prefissato e costante da 1 a 24

Variabili di decisione:

$$p_i(k) \qquad i = 1,2,3,4$$

4 variabili da determinare

Funzione di costo da minimizzare:

$$J_k = \sum_{i=1}^{4} c_{mi}\, p_i(k)$$

Problema LP

Vincoli

$$P_{mi} \leq p_i(k) \leq P_{Mi} \qquad \forall\, i = 1,2,3,4$$

$$\sum_{i=1}^{4} p_i(k) = D(k)$$

Programmazione lineare

LP

Problema generale

$$\min_x c^T x \qquad x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^n$$

Decision variables

Vincoli

$Ax \leq b$ — *Vincoli di diseguaglianza*

$A_{eq}x = b_{eq}$ — *Vincoli di uguaglianza*

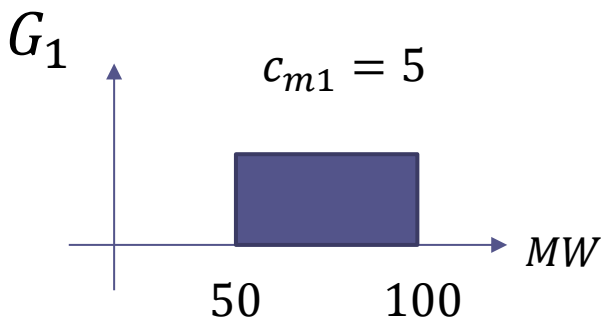
c è un vettore colonna di n elementi (in modo che $c^T x$ sia uno scalare).

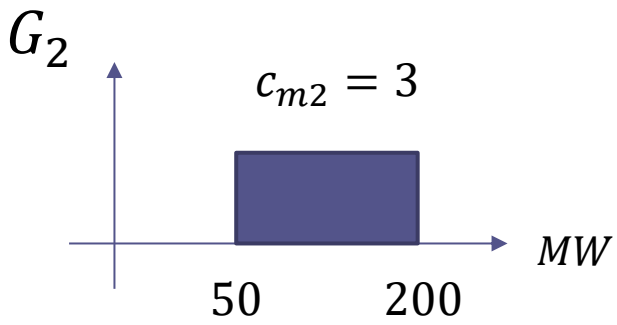
A, A_{eq} sono matrici che hanno n colonne.

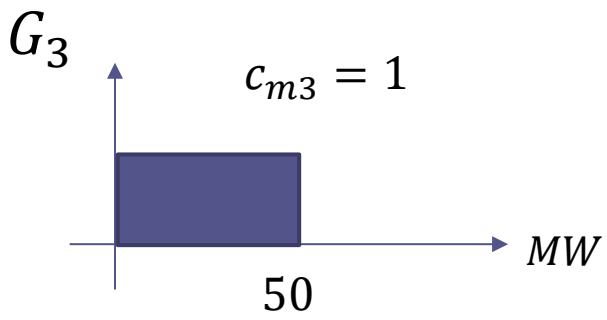
b, b_{eq} sono vettori che hanno tanti elementi quante sono, rispettivamente, le righe di $A, ed\ A_{eq}$.

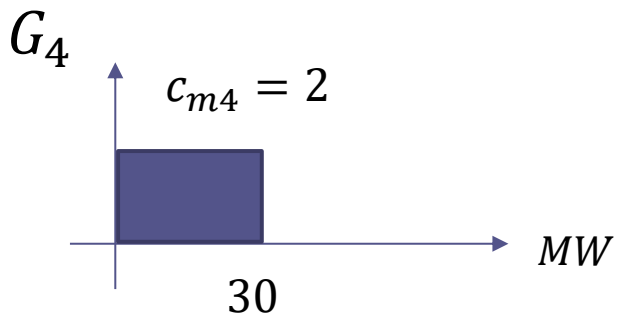
Dati

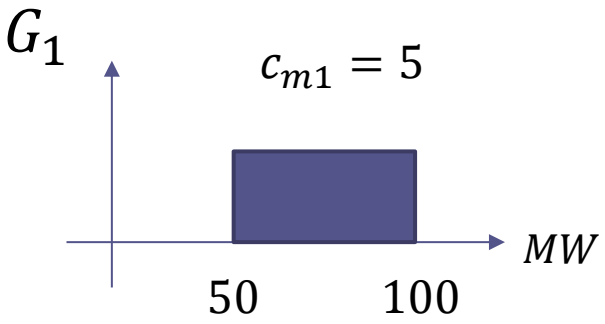
		$i=1$	$i=2$	$i=3$	$i=4$
c_{mi}	$\left[\frac{€}{MWh}\right]$	5	3	1	2
P_{mi}	$[MW]$	50	50	0	0
P_{Mi}	$[MW]$	100	200	50	30

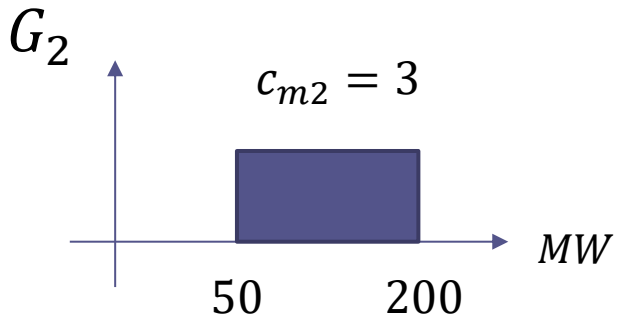


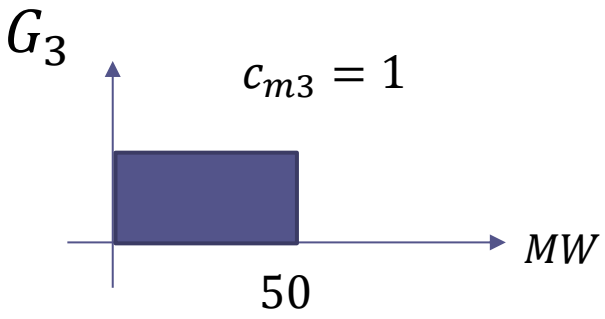


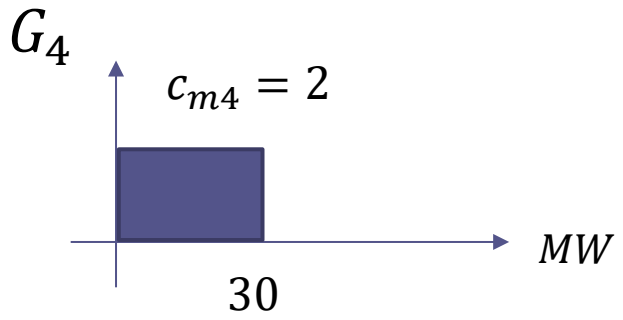












La domanda oraria $D(k)$ non deve ovviamente eccedere i 380 MW (e nemmeno essere inferiore ai 100 MW)

	$k = 1$	$k = 2$	$k = 3$
$D(k)$	120	200	260

Quali sono le soluzioni?

Implementiamo il solutore in Matlab

Variabili di decisione

$$x = \begin{bmatrix} p_1(k) \\ p_2(k) \\ p_3(k) \\ p_4(k) \end{bmatrix}$$

Funzione di costo da minimizzare:

$$J_k = \sum_{i=1}^{4} c_{mi}\, p_i(k) = c^T x$$

$$c^T = [c_{m1} \quad c_{m2} \quad c_{m3} \quad c_{m4}]$$

Vincoli di **disuguaglianza**

$$P_{mi} \leq p_i(k) \leq P_{Mi} \qquad \forall\, i = 1,2,3,4$$

$$p_1(k) \leq P_{M1}$$

$$-p_1(k) \leq -P_{m1}$$

$$p_2(k) \leq P_{M2}$$

$$-p_2(k) \leq -P_{m2}$$

$$p_3(k) \leq P_{M3}$$

$$-p_3(k) \leq -P_{m3}$$

$$p_4(k) \leq P_{M4}$$

$$-p_4(k) \leq -P_{m4}$$

$$Ax \leq b \qquad x = \begin{bmatrix} p_1(k) \\ p_2(k) \\ p_3(k) \\ p_4(k) \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 \end{bmatrix} \qquad b = \begin{bmatrix} P_{M1} \\ -P_{m1} \\ P_{M2} \\ -P_{m2} \\ P_{M3} \\ -P_{m3} \\ P_{M4} \\ -P_{m4} \end{bmatrix}$$

Vincoli di
uguaglianza

$$\sum_{i=1}^{4} p_i(k) = D(k)$$

$$A_{eq}x = b_{eq} \qquad x = \begin{bmatrix} p_1(k) \\ p_2(k) \\ p_3(k) \\ p_4(k) \end{bmatrix}$$

$$A_{eq} = [1 \;\; 1 \;\; 1 \;\; 1] \qquad b_{eq} = D(k)$$

```
clear all, close all, clc

n=4;   %numero di generatori

mc1=5;mc2=3;mc3=1;mc4=2;

mc=[mc1 mc2 mc3 mc4];

x1m=50; x1M=100;
x2m=50; x2M=200;
x3m=0; x3M=50;
x4m=0; x4M=30;

D0=120;

ct=[mc1 mc2 mc3 mc4];

A=zeros(8,4);

for i=1:n
      A(1+2*(i-1):2+2*(i-1),i)=[1; -1];
end

b=[x1M -x1m x2M -x2m x3M -x3m x4M -x4m];

Aeq=[1 1 1 1];
beq=D0;

x = linprog(ct,A,b,Aeq,beq)
```

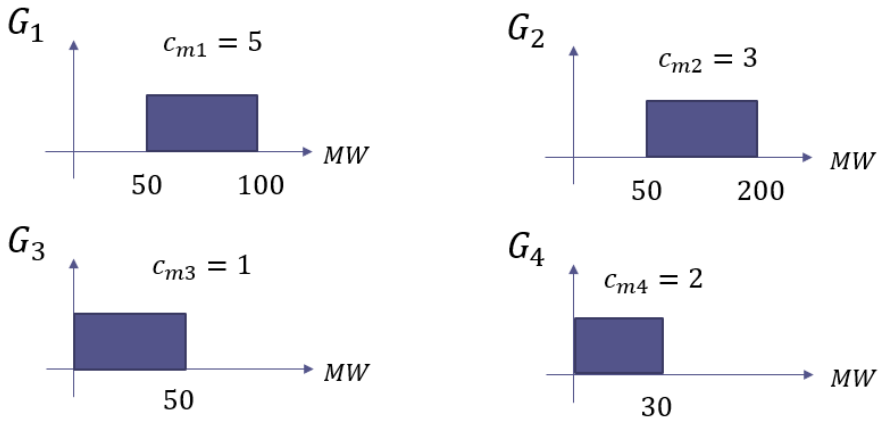
```
Optimal solution found.


x =

    50
    50
    20
     0
```

Script file: Dispatch_LP_ex11h.m

Versione semplificata dello script in cui i valori minimi e massimi delle variabili di decisione sono imposti attraverso gli argomenti di ingresso aggiuntivi `lb,ub` alla funzione `linprog`

```
clear all, close all, clc

mc1=5;mc2=3;mc3=1;mc4=2;

mc=[mc1 mc2 mc3 mc4];

x1m=50; x1M=100;
x2m=50; x2M=200;
x3m=0; x3M=50;
x4m=0; x4M=30;

D1=200;

ct=[mc1 mc2 mc3 mc4];

Aeq=[1 1 1 1];
beq=D1;

A=[];
b=[];
lb=[x1m x2m x3m x4m];
ub=[x1M x2M x3M x4M];

x = linprog(ct,A,b,Aeq,beq,lb,ub)
```

```
Optimal solution found.


x =

    50
    70
    50
    30
```



Script file: Dispatch_LP_ex11hv02.m

Per come abbiamo impostato il problema, non è previsto lo shut-down dei generatori e come risultato i generatori G1 e G2 sono sempre attivi ad un livello di potenza non inferiore al loro valore minimo.

Ciò comporta che il problema di ottimizzazione sia unfeasible se $D(k) < 100$

$D(k) = 90$

```
No feasible solution found.

Linprog stopped because no point satisfies the constraints.


x =

     []
```

Riformuliamo il problema prevedendo che i generatori possano essere disattivati o comunque disconnessi dai carichi

Associamo pertanto a ciascun generatore una variabile addizionale δ_i $(i = 1,2,3,4)$ **binaria** che codifichi lo stato di attivazione o shut-down del generatore

Variabili di decisione

$$x = \begin{bmatrix} p_1(k) \\ p_2(k) \\ p_3(k) \\ p_4(k) \\ \delta_1(k) \\ \delta_2(k) \\ \delta_3(k) \\ \delta_4(k) \end{bmatrix}$$

Un problema di ottimizzazione in cui alcune variabili decisionali sono vincolate ad assumere valori interi (o binari, che è un caso particolare) ricade come detto nella famiglia dei problemi «Mixed-integer». Sebbene si tratti di un problema di ottimizzazione non convesso, esistono comunque dei solutori relativamente affidabili nel caso in cui la funzione obbiettivo ed i vincoli siano lineari (problema MILP – mixed-integer linear programming)

Programmazione lineare mixed-integer

MILP

Problema generale

$$\min_x \; c^T x$$

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

Decision variables

$$x_i \in \mathbb{R} \qquad i \in V^R$$

$$x_i \in \mathbb{Z} \qquad i \in V^Z$$

$$V^R \cap V^Z = \emptyset$$

$$V^R \cup V^Z = V = \{1,2,\dots,n\}$$

Vincoli

$Ax \leq b$ — *Vincoli di diseguaglianza*

$A_{eq}x = b_{eq}$ — *Vincoli di uguaglianza*

Vediamo se riusciamo a formulare il problema in esame in termini MILP

Per la funzione obiettivo ed i vincolo sulla produzione complessiva oraria, la prima soluzione che potrebbe venire in mente prevede una loro riformulazione nei termini seguenti

$$J_k = \sum_{i=1}^{4} c_{mi}\, p_i(k)\delta_i(k)$$

$$\sum_{i=1}^{4} p_i(k)\delta_i(k) = D(k)$$

Evidentemente tale strada non è la migliore, in quanto la funzione obiettivo ed il vincolo non dipendono più in maniera lineare dalle variabili di decisione ma in maniera aggregata dal loro prodotto

Questo approccio condurrebbe, più nel dettaglio, ad una funzione obiettivo quadratica del tipo

$$J_k = \frac{1}{2} x^T P x + c^T x$$

Che forma assumono la matrice P ed il vettore c ?

$$P = \begin{bmatrix} 0 & 0 & 0 & c_{m1} & 0 & 0 \\ 0 & 0 & 0 & 0 & c_{m2} & 0 \\ 0 & 0 & 0 & 0 & 0 & c_{m3} \\ c_{m1} & 0 & 0 & 0 & 0 & 0 \\ 0 & c_{m2} & 0 & 0 & 0 & 0 \\ 0 & 0 & c_{m3} & 0 & 0 & 0 \end{bmatrix} \qquad c = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

La matrice P possiede gli autovalori $\pm c_{m1}, \pm c_{m2}, \pm \mathrm{c_{m3}}$ e come tale la funzione obiettivo è non convessa. In aggiunta, le variabili decisionali sono in parte reali ed in parte binarie, ed i problemi di tipo MIQCQP sono di non semplice risoluzione.

Cerchiamo una strada differente.

Ci serve un vincolo lineare sulle variabili di decisione tale da garantire che se $\delta_i(k) = 0$ allora il corrispondente livello di potenza $p_i(k)$ sia posto pari a zero, in modo che non sia necessario includere esplicitamente le variabili binarie all'interno della funzione obiettivo.

Un vincolo del tipo

$$P_{mi}\delta_i(k) \leq p_i(k) \leq P_{Mi}\delta_i(k) \qquad i = 1,2,3,4$$

è esattamente ciò che ci serve.

Esso da un lato impone il fatto che se $\delta_i(k) = 0$ (generatore i-esimo disattivato) allora il corrispondente livello di potenza $p_i(k)$ sia posto pari a zero.

Ciò permette di lasciare inalterata l'espressione della funzione obiettivo

$$J_k = \sum_{i=1}^{4} c_{mi}\, p_i(k)$$

Osserviamo anche come il vincolo proposto garantisca simultaneamente che se il generatore è attivo ($\delta_i(k) = 1$) il corrispondente livello di potenza sia limitato inferiormente e superiormente dai valori minimi e massimi prescritti

$$P_{mi} \leq p_i(k) \leq P_{mi} \qquad i = 1,2,3,4$$

Caratterizziamo quindi in maniera completa il problema MILP in esame

Variabili di decisione $x = \begin{bmatrix} p_1(k) \\ p_2(k) \\ p_3(k) \\ p_4(k) \\ \delta_1(k) \\ \delta_2(k) \\ \delta_3(k) \\ \delta_4(k) \end{bmatrix}$

$p_i(k) \in \mathbb{R}$ $\quad \Rightarrow \quad V^R = \{1,2,3,4\}$

$\delta_i(k) \in \mathbb{Z}$ $\quad\quad\quad V^Z = \{5,6,7,8\}$

Determiniamo le matrici che caratterizzano i vincoli:

Vincoli di disuguaglianza

$$P_{mi}\delta_i(k) \le p_i(k) \le P_{Mi}\delta_i(k) \qquad i = 1,2,3,4$$

$$\Rightarrow \quad \begin{aligned} -p_i(k) + P_{mi}\delta_i(k) &\le 0 \\ p_i(k) - P_{Mi}\delta_i(k) &\le 0 \end{aligned} \qquad i = 1,2,3,4$$

$$\begin{aligned} -p_1(k) + P_{m1}\delta_1(k) &\le 0 \\ p_1(k) - P_{M1}\delta_1(k) &\le 0 \\ -p_2(k) + P_{m2}\delta_2(k) &\le 0 \\ p_2(k) - P_{M2}\delta_2(k) &\le 0 \\ -p_3(k) + P_{m3}\delta_3(k) &\le 0 \\ p_3(k) - P_{M3}\delta_3(k) &\le 0 \\ -p_4(k) + P_{m4}\delta_4(k) &\le 0 \\ p_4(k) - P_{M4}\delta_4(k) &\le 0 \end{aligned}$$

$$\Rightarrow \quad Ax \le b \qquad x = \begin{bmatrix} p_1(k) \\ p_2(k) \\ p_3(k) \\ p_4(k) \\ \delta_1(k) \\ \delta_2(k) \\ \delta_3(k) \\ \delta_4(k) \end{bmatrix}$$

$$A = \begin{bmatrix} -1 & 0 & 0 & 0 & P_{m1} & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & P_{M1} & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & P_{m2} & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & P_{M2} & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & P_{m3} & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & P_{M3} & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & P_{m4} \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & P_{M4} \end{bmatrix} \qquad b = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Vincoli di uguaglianza

$$\sum_{i=1}^{4} p_i(k) = D(k)$$

$$A_{eq}x = b_{eq}$$

$$x = \begin{bmatrix} p_1(k) \\ p_2(k) \\ p_3(k) \\ p_4(k) \\ \delta_1(k) \\ \delta_2(k) \\ \delta_3(k) \\ \delta_4(k) \end{bmatrix}$$

$$A_{eq} = [1 \;\; 1 \;\; 1 \;\; 1 \;\; 0 \;\; 0 \;\; 0 \;\; 0]$$

$$b_{eq} = D(k)$$

Un problema di programmazione lineare mixed integer MILP si risolve in Matlab mediante la funzione **intlinprog**

Sintassi di base

```
x=intlinprog(c,intcon,A,b,Aeq,beq)
```

`intcon` é un vettore che contiene le posizioni all'interno del vettore x delle variabili decisionali x_i che assumono valori interi (cioè contiene gli elementi dell'insieme V^Z)

Se non vi sono vincoli di uguaglianza si omettono gli ultimi due argomenti in ingresso alla funzione:

```
x= intlinprog(c,intcon,A,b)
```

Se vi sono unicamente vincoli di uguaglianza, i termini `A,b` vengono rappresentati dal vettore vuoto `[]`:

```
x=intlinprog(c,intcon,[],[],Aeq,beq)
```

La funzione `intlinprog` consente di passare i valori minimi e massimi delle variabili di decisione come argomenti aggiuntivi in ingresso

```
x=intlinprog(c,intcon,A,b,Aeq,beq,lb,ub)
```

upper bound

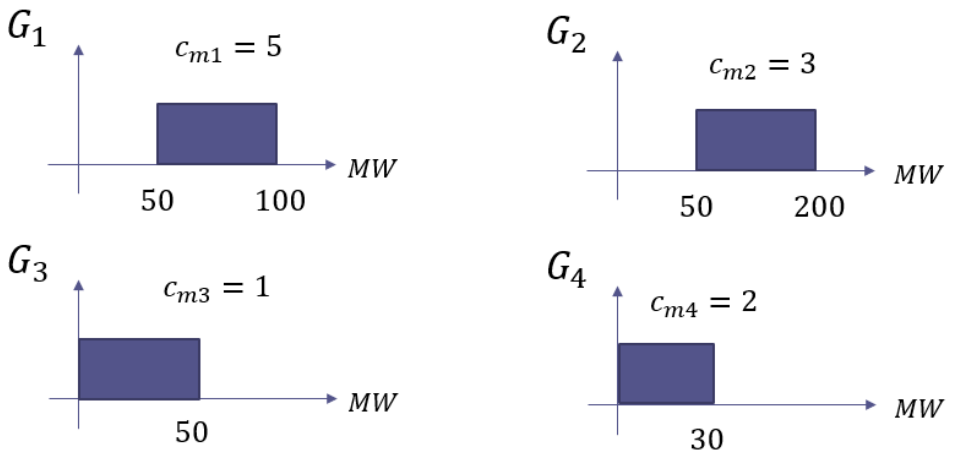
lower bound

```
lb=[x1m x2m … xnm]

ub=[x1M x2M … xnM]
```

N.B. Per imporre che una certa variabile di decisione intera sia una variabile **binaria** si deve imporre come estremo inferiore zero e come estremo superiore 1

Avendo deciso che in corrispondenza della condizione di shut-down di un generatore il corrispondente livello di potenza p_i debba essere settato a zero, i limiti inferiori (lower bound) da considerare per le variabili p_i sono pari a zero.

```
clear all, close all, clc

mc1=5; mc2=3;
mc3=1; mc4=2;
mc=[mc1 mc2 mc3 mc4];

ct=[mc 0 0 0 0];

x1m=50; x1M=100;
x2m=50; x2M=200;
x3m=0; x3M=50;
x4m=0; x4M=30;

A=[1 0 0 0 -x1M 0 0 0;
  -1 0 0 0 x1m 0 0 0;
  0 1 0 0 0 -x2M 0 0;
  0 -1 0 0 0 x2m 0 0;
  0 0 1 0 0 0 -x3M 0;
  0 0 -1 0 0 0 x3m 0;
  0 0 0 1 0 0 0 -x4M;
  0 0 0 -1 0 0 0 x4m];

b=zeros(8,1);

D=50;
Aeq=[1 1 1 1 0 0 0 0];
beq=D;
intcon=5:8;
lb=zeros(1,8);
ub=[x1M x2M x3M x4M 1 1 1 1];

x = intlinprog(ct,intcon,A,b,Aeq,beq,lb,ub);
POTENZE=x(1:4)
ONOFF=x(5:8)
```




```
POTENZE =

     0
     0
    50
     0


ONOFF =

     0
     0
     1
     1
```

File: Dispatch_LP_ex11hMILP.m

Ora sviluppiamo il codice in maniera differente e più compatta, ponendo a frutto il linguaggio di modellazione dell'Optimization Toolbox di Matlab .

Facciamo uso di particolari variabili simboliche ("optimization variables"), che consentono di creare espressioni compatte e facili da programmare sia per la funzione obiettivo che per i vincoli di uguaglianza e diseguaglianza che coinvolgono tali variabili.

I vincoli vengono definiti uno ad uno mediante istruzioni Matlab pressoché coincidenti con la definizione analitica del vincolo, evitando la complessa operazione di creazione delle matrici e vettori A, b, A_{eq}, b_{eq} .

Istruzioni Matlab vettoriali per la rappresentazione dei vincoli.

Esempi di possibili sintassi

$$x = \begin{bmatrix} x_1(1) & x_1(2) \\ x_2(1) & x_2(2) \\ x_3(1) & x_3(2) \end{bmatrix}$$

$a = cost.$

$x \leq a$ ⇒

$$x_1(1) \leq a$$
$$x_2(1) \leq a$$
$$x_3(1) \leq a$$
$$x_1(2) \leq a$$
$$x_2(2) \leq a$$
$$x_3(2) \leq a$$

Istruzioni Matlab vettoriali per la rappresentazione dei vincoli.

Esempi di possibili sintassi

$$x = \begin{bmatrix} x_1(1) & x_1(2) \\ x_2(1) & x_2(2) \\ x_3(1) & x_3(2) \end{bmatrix}$$

$a = cost.$

$x == a$

$$x_1(1) = a$$
$$x_2(1) = a$$
$$x_3(1) = a$$
$$x_1(2) = a$$
$$x_2(2) = a$$
$$x_3(2) = a$$

Istruzioni Matlab vettoriali per la rappresentazione dei vincoli.

Esempi di possibili sintassi

$$x = \begin{bmatrix} x_1(1) & x_1(2) \\ x_2(1) & x_2(2) \\ x_3(1) & x_3(2) \end{bmatrix} \qquad M = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \\ M_{31} & M_{32} \end{bmatrix}$$

$$x \geq M \quad \Rightarrow$$

$$x_1(1) \geq M_{11}$$
$$x_2(1) \geq M_{21}$$
$$x_3(1) \geq M_{31}$$
$$x_1(2) \geq M_{12}$$
$$x_2(2) \geq M_{22}$$
$$x_3(2) \geq M_{32}$$

File: Dispatch_short.m

```
clear all, close all, clc

mc1=5; mc2=3; mc3=1;  mc4=2;
mc=[mc1 mc2 mc3 mc4]';

x1m=50; x1M=100;
x2m=50; x2M=200;
x3m=0; x3M=50;
x4m=0; x4M=30;

xm=[x1m x2m x3m x4m]';
xM=[x1M x2M x3M x4M]';
D=50;
```

Dati del problema

```
p = optimvar('p',4,1);
delta=optimvar('delta',4,1,'Type','integer','LowerBound',0,...
     'UpperBound',1);
```

Definizione delle variabili di ottimizzazione (funzione `optimvar`)

```
powercons = sum(p) == D;
deltacons1 = xm.*delta <= p;
deltacons2 = p <= xM.*delta;
```

Definizione vincoli

```
obj=sum(mc.*p);          % obj=mc'*p;  sintassi alternativa
```

Definizione funzione obiettivo

```
dispatch=optimproblem;
dispatch.Objective = obj;
dispatch.Constraints.powercons = powercons;
dispatch.Constraints.deltacons1 = deltacons1;
dispatch.Constraints.deltacons2 = deltacons2;
```

Creazione del problema di ottimizzazione (funzione `optimproblem`) ed assegnazione della funzione obiettivo e dei vincoli.

```
options = optimoptions('intlinprog','Display','final');

dispatchsol= solve(dispatch,'options',options);
```

Scelta del solutore e risoluzione del problema di ottimizzazione

```
pstar=dispatchsol.p
deltastar=dispatchsol.delta
```

L'estensione da 1 ora a 24 ore di funzionamento prevede modifiche minime al codice.

File: `Dispatch_short24h.m`

$$P = \begin{bmatrix} p_1(1) & p_1(2) & \dots & p_1(24) \\ p_2(1) & p_2(2) & \dots & p_2(24) \\ p_3(1) & p_3(2) & \dots & p_3(24) \\ p_4(1) & p_4(2) & \dots & p_4(24) \end{bmatrix}$$

$$\delta = \begin{bmatrix} \delta_1(1) & \delta_1(2) & \dots & \delta_1(24) \\ \delta_2(1) & \delta_2(2) & \dots & \delta_2(24) \\ \delta_3(1) & \delta_3(2) & \dots & \delta_3(24) \\ \delta_4(1) & \delta_4(2) & \dots & \delta_4(24) \end{bmatrix}$$

Funzione obiettivo

$$J = \sum_{k=1}^{24} \sum_{i=1}^{4} c_{mi}\, p_i(k) = c_{m1}p_1(1) + c_{m2}p_2(1) + c_{m3}p_3(1) + c_{m4}p_4(1) +$$
$$+ c_{m1}p_1(2) + c_{m2}p_2(2) + c_{m3}p_3(2) + c_{m4}p_4(2) + \cdots +$$
$$+ c_{m1}p_1(24) + c_{m2}p_2(24) + c_{m3}p_3(24) + c_{m4}p_4(24)$$

$$J = \sum_{k=1}^{24} \sum_{i=1}^{4} c_{mi}\, p_i(k) = c_{m1}p_1(1) + c_{m2}p_2(1) + c_{m3}p_3(1) + c_{m4}p_4(1) +$$
$$+c_{m1}p_1(2) + c_{m2}p_2(2) + c_{m3}p_3(2) + c_{m4}p_4(2) + \cdots +$$
$$+c_{m1}p_1(24) + c_{m2}p_2(24) + c_{m3}p_3(24) + c_{m4}p_4(24)$$

$$P = \begin{bmatrix} p_1(1) & p_1(2) & \dots & p_1(24) \\ p_2(1) & p_2(2) & \dots & p_2(24) \\ p_3(1) & p_3(2) & \dots & p_3(24) \\ p_4(1) & p_4(2) & \dots & p_4(24) \end{bmatrix}$$

$$J = [c_{m1} \quad c_{m2} \quad c_{m3} \quad c_{m4}] \cdot \begin{bmatrix} p_1(1) + p_1(2) + \cdots + p_1(24) \\ p_2(1) + p_2(2) + \cdots + p_2(24) \\ p_3(1) + p_3(2) + \cdots + p_3(24) \\ p_4(1) + p_4(2) + \cdots + p_4(24) \end{bmatrix}$$

Vincoli di disuguaglianza

$$P_{mi}\delta_i(k) \leq p_i(k) \leq P_{Mi}\delta_i(k)$$

$$\begin{bmatrix} P_{m1}\delta_1(1) & P_{m1}\delta_1(2) & \dots & P_{m1}\delta_1(24) \\ P_{m2}\delta_2(1) & P_{m2}\delta_2(2) & \dots & P_{m2}\delta_2(24) \\ P_{m3}\delta_3(1) & P_{m3}\delta_3(2) & \dots & P_{m4}\delta_3(24) \\ P_{m4}\delta_4(1) & P_{m4}\delta_4(2) & \dots & P_{m4}\delta_4(24) \end{bmatrix} \leq \begin{bmatrix} p_1(1) & p_1(2) & \dots & p_1(24) \\ p_2(1) & p_2(2) & \dots & p_2(24) \\ p_3(1) & p_3(2) & \dots & p_3(24) \\ p_4(1) & p_4(2) & \dots & p_4(24) \end{bmatrix} \leq \begin{bmatrix} P_{M1}\delta_1(1) & P_{M1}\delta_1(2) & \dots & P_{M1}\delta_1(24) \\ P_{M2}\delta_2(1) & P_{M2}\delta_2(2) & \dots & P_{M2}\delta_2(24) \\ P_{M3}\delta_3(1) & P_{M3}\delta_3(2) & \dots & P_{M4}\delta_3(24) \\ P_{M4}\delta_4(1) & P_{M4}\delta_4(2) & \dots & P_{M4}\delta_4(24) \end{bmatrix}$$

Soluzione compatta – 24 ore di funzionamento

```
clear all, close all, clc

numh=24;
numgen=4;

mc1=5; mc2=3; mc3=1;  mc4=2;
mc=[mc1 mc2 mc3 mc4]';

x1m=50; x1M=100;
x2m=50; x2M=200;
x3m=0; x3M=50;
x4m=0; x4M=30;

xm=[x1m x2m x3m x4m]';
xM=[x1M x2M x3M x4M]';
```

```
for i=1:numh
mcext(:,i)=mc;
xmext(:,i)=xm;
xMext(:,i)=xM;
end
```

Matrici «estese» che replicano la medesima colonna per `numh=24` volte.

```
mcext =

  Columns 1 through 14

     5     5     5     5     5     5     5     5     5     5     5     5     5     5
     3     3     3     3     3     3     3     3     3     3     3     3     3     3
     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     2     2     2     2     2     2     2     2     2     2     2     2     2     2
```

```
D=[24 30 50 60 63 70 75 95 100 120 ...
    150 190 200 230 280 360 340 350 320 300 ...
    250 220 200 150;];

figure(1)
bar(D)
title('Domanda [MWh]')
```

Carico

```
p = optimvar('p',numgen,numh);
delta=optimvar('delta',numgen,numh,'Type','integer','LowerBound',0,...
     'UpperBound',1);

obj=sum(sum(mcext.*p));
```

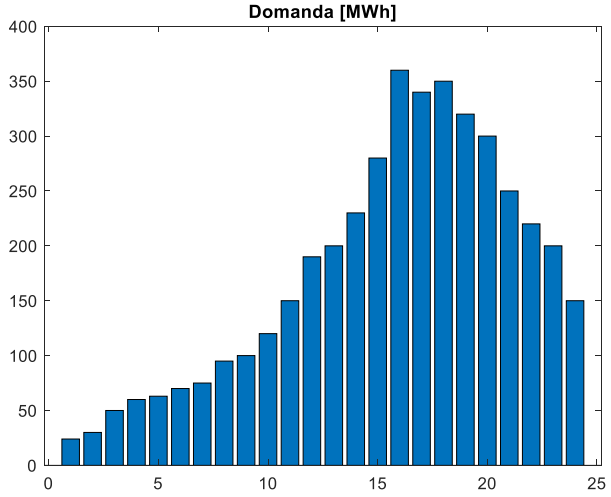
Definizione della funzione obiettivo (la doppia applicazione ad una matrice della funzione sum restituisce la somma fra tutti gli elementi della matrice). La matrice estesa `mcext` viene moltiplicata elemento per elemento con la matrice `p`.

```
powercons = sum(p) == D;
deltacons1 = xmext.*delta <= p;
deltacons2 = p <= xMext.*delta;
```

Definizione dei vincoli

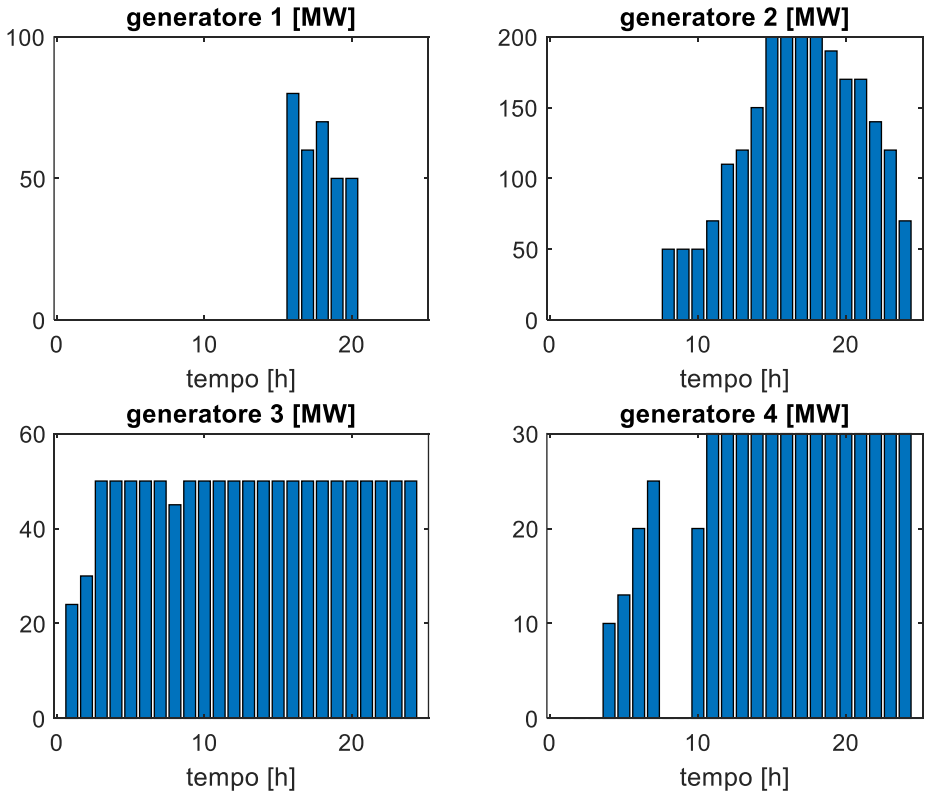
```
dispatch=optimproblem;

dispatch.Objective = obj;

dispatch.Constraints.powercons = powercons;
dispatch.Constraints.deltacons1 = deltacons1;
dispatch.Constraints.deltacons2 = deltacons2;

options = optimoptions('intlinprog','Display','final');

dispatchsol= solve(dispatch,'options',options);
pstar=dispatchsol.p;
deltastar=dispatchsol.delta;
```

Non modificata



Carico orario

Profili di potenza dei generatori



Potrebbero essere molteplici i vincoli aggiuntivi tali da rendere il problema non più disaccoppiabile da un punto di vista temporale.

Vincoli sulla disponibilità giornaliera di combustibile

$$\sum_{k=1}^{24} p_i(k) \leq \Gamma_i$$

Vincoli sulla massima variazione di potenza consentita fra due intervalli temporali adiacenti («ramping constraint»)

$$-\zeta_i^{down} \leq p_i(k) - p_i(k-1) \leq \zeta_i^{up} \qquad k = 2,3,\dots,24$$

Nell'includere tali vincoli si deve prestare attenzione alla «feasibility» del problema, cioè alla esistenza di una soluzione.

Risolviamo il problema inserendo un vincolo sulla disponibilità giornaliera di combustibile per il generatore 3.

Sia f_3 il consumo marginale di combustibile del generatore 3

$$f_3 \qquad \left[\frac{kg}{MWh}\right] \qquad\qquad f_3 = 100$$

e Q^{max} $[kg]$ la quantità di combustibile a disposizione $\qquad Q^{max} = 90000\ kg = 90\ t$

La quantità di combustibile Q consumata nelle 24 ore è pari a

$$Q = f_3 \sum_{k=1}^{24} p_3(k)$$

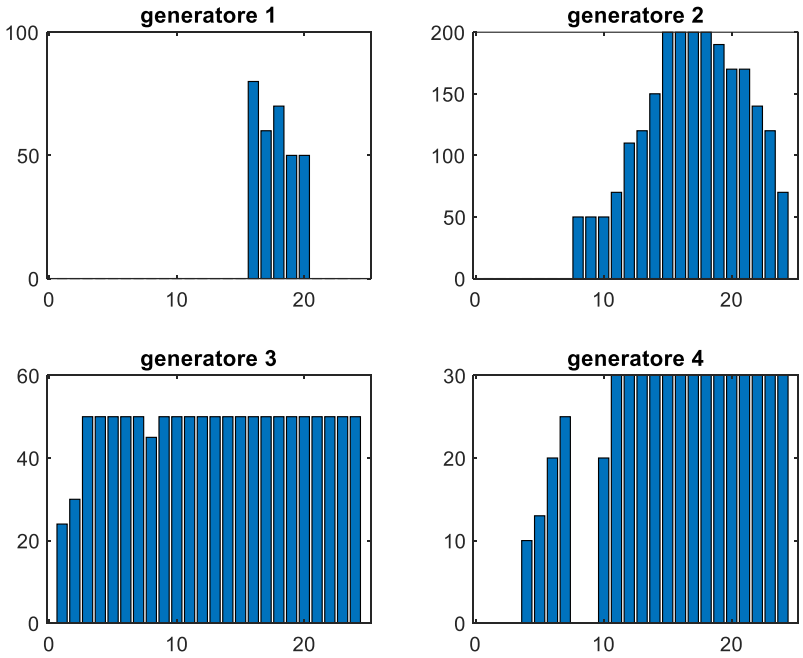
Il vincolo $Q \leq Q_{max}$ assume la forma

$$\sum_{k=1}^{24} p_3(k) \leq \frac{Q^{max}}{f_3} \qquad\qquad \frac{Q^{max}}{f_3} = 900$$

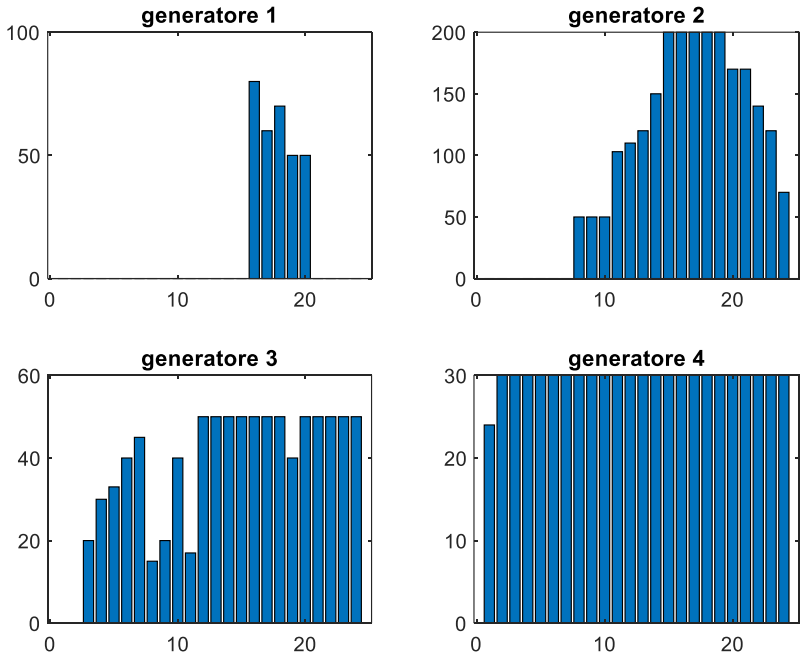
Codice Matlab **File:** Dispatch_short24hFuel.m

```
Qmax=90000;
f3=100;

p = optimvar('p',numgen,numh);
delta=optimvar('delta',numgen,numh,'Type','integer','LowerBound',0,...
     'UpperBound',1);

powercons = sum(p) == D;
deltacons1 = xmext.*delta <= p;
deltacons2 = p <= xMext.*delta;
fuelG3=sum(p(3,:))<= Qmax/f3;

obj=sum(sum(mcext.*p));

dispatch=optimproblem;
dispatch.Objective = obj;
dispatch.Constraints.powercons = powercons;
dispatch.Constraints.deltacons1 = deltacons1;
dispatch.Constraints.deltacons2 = deltacons2;
dispatch.Constraints.fuelG3 = fuelG3;

options = optimoptions('intlinprog','Display','final');

dispatchsol= solve(dispatch,'options',options);
dispatchsol.p;
dispatchsol.delta;
```

Nessun vincolo sulla disponibilità di combustibile



Con vincolo sulla disponibilità di combustibile per il generatore G3

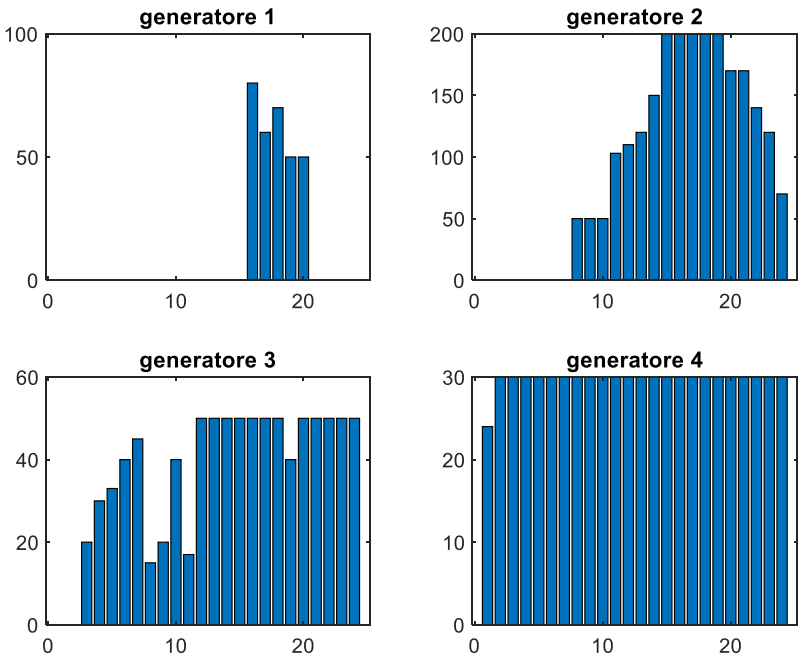


Il generatore 3 viene impiegato in misura minore rispetto al caso senza vincolo sul combustibile, e consuma (verificarlo) tutto il combustibile disponibile

```
Qmax=90000;
f3=100;
csi1UP=50;
csi1DOWN=50;

p = optimvar('p',numgen,numh);
delta=optimvar('delta',numgen,numh,'Type','integer','LowerBound',0,...
     'UpperBound',1);

powercons = sum(p) == D;
deltacons1 = xmext.*delta <= p;
deltacons2 = p <= xMext.*delta;
fuelG3=sum(p(3,:))<= Qmax/f3;
rampUPG1=p(1,2:24)-p(1,1:23)<= csi1UP;
rampDOWNG1=p(1,2:24)-p(1,1:23)>=-csi1DOWN;

obj=sum(sum(mcext.*p));

dispatch=optimproblem;
dispatch.Objective = obj;
dispatch.Constraints.powercons = powercons;
dispatch.Constraints.deltacons1 = deltacons1;
dispatch.Constraints.deltacons2 = deltacons2;
dispatch.Constraints.fuelG3 = fuelG3;
dispacciamento.Constraints.rampUPG1 = rampUPG1;
dispacciamento.Constraints.rampDOWNG1 = rampDOWNG1;

options = optimoptions('intlinprog','Display','final');

dispatchsol= solve(dispatch,'options',options);
dispatchsol.p;
dispatchsol.delta;
```

Oltre al vincolo sulla disponibilità di combustibile per G3 introduciamo un ramp constraint su G1

Con vincolo sulla disponibilità di combustibile per il generatore G3.
No ramping constraint



Con vincolo sulla disponibilità di combustibile per il generatore G3.
Ramp-constraint su G3 ($\zeta_3^{up} = \zeta_i^{down} = 20$)