

Università “Ca’Foscari” Venezia

*Dipartimento di Scienze Ambientali,
Informatica e Statistica*

Giovanni Fasano ¹

*Software & Esempi per la guida alla
soluzione di problemi di Programmazione
Matematica mediante AMPL*

¹Università Ca’Foscari Venezia, Dipartimento di Management, S.Giobbe Cannaregio 873, 30121 Venezia, ITALY. E-mail: fasano@unive.it ; URL: <http://venus.unive.it/~fasano> - A.A. 2014-2015.

Sommario

- Esercitazione 1: Software per la Programmazione Matematica
 - Software per la Programmazione Matematica
 - Esempio 1
 - Esempio 2
- Esercitazione 2: Esempi di Programmazione Matematica
 - Esempio 1 (numerico - lineare)
 - Esempio 2 (Regressione Lineare)
 - Esempio 3 (Case Study - lineare intero) - Jeppesen Sanderson, Inc.
- Esercitazione 3: Esempio di Programmazione Matematica Nonlineare
 - Progetto di una struttura a traliccio
- Esercitazione 4: Esempio di Programmazione Matematica Nonlineare
 - Calcolo traiettoria Shuttle

1.1 Esercitazione 1: Software per la Programmazione Matematica

In questa sezione si intendono fornire alcune informazioni utili per la consultazione e l'utilizzo di software, dedicato alla soluzione di problemi di programmazione matematica. In maniera del tutto intuitiva, possiamo brevemente sintetizzare le fasi che portano alla soluzione di un problema reale mediante i seguenti punti:

1. **Studio del problema reale.** In questa fase sono di cruciale importanza gli aspetti legati alle proprietà ed alle caratteristiche del problema stesso. È sostanzialmente una fase di analisi del problema in oggetto.
2. **Identificazione e Analisi del modello.** Sulla base del punto precedente, in questa fase da un lato si cerca una buona rappresentazione descrittiva del problema reale, dall'altra si deve evitare che le caratteristiche del modello, in generale differenti da quelle del problema in oggetto, se ne discostino eccessivamente. Questo punto evidenzia uno sforzo tipicamente di sintesi, il quale risulta cruciale in quanto in generale la soluzione del problema può differire dalla soluzione del modello.
3. **Soluzione del Modello.** A questo punto è necessario l'utilizzo di uno strumento di calcolo, il quale risulti appropriato per fornire la soluzione del modello. In questa fase il problema reale di partenza è assente. Notiamo che la scelta del solutore opportuno è legata tanto alle caratteristiche quanto ai limiti del modello stesso (e.g. non è possibile l'utilizzo di solutori che richiedono la differenziabilità delle funzioni trattate, se il modello stesso contiene relazioni non differenziabili). Inoltre si osservi che spesso un solutore troppo generico (i.e. applicabile ad una vasta gamma di modelli) può essere inappropriato, di conseguenza la validità delle soluzioni fornite deve costituire oggetto di valutazione comparativa.
4. **Verifica dei risultati.** Da quanto detto nei punti 1.-3., la soluzione fornita da un solutore è soggetta ad almeno due significative verifiche. Da un lato bisogna controllare che i risultati ottenuti, relativi al modello, siano soddisfacenti per il modello stesso (e.g. può non essere utile prendere in considerazione un solutore che per una determinata istanza fornisce una precisione insufficiente per la soluzione del modello). In secondo luogo è indispensabile che i risultati ottenuti, lavorando sul modello, siano *soddisfacenti* per il problema reale; in altri termini è indispensabile *validare* i risultati rispetto al problema reale di partenza. Questa fase spesso rivela che i risultati ottenuti sulla base del modello considerato, sono contraddittori/insufficienti/imprecisi/inadeguati/ecc. per il problema originale. Sarà allora necessario ritornare al punto 2. e modificare il modello stesso, cercando di affinarlo per quanto possibile. In particolare non va sottovalutato che la descrizione stessa del modello avviene attraverso un linguaggio formale; in tal senso la sintassi e la semantica di tale linguaggio giocano un ruolo significativo, al fine di realizzare le specifiche del modello stesso.

I modelli di cui ci occuperemo nel presente corso sono riconducibili a problemi di *programmazione matematica*, ovvero possono essere rappresentati nella forma

$$\min_{x \in \mathcal{A}} f(x), \quad (1.1)$$

nella quale x rappresenta un vettore di **variabili**, \mathcal{A} è l'**insieme ammissibile** per il vettore x ed $f(x)$ è la **funzione obiettivo** (nel nostro caso reale). In sintesi, risolvere la (1.1) significa

scegliere (se esiste!) nell'insieme \mathcal{A} un (in generale *più d'uno*) vettore x che minimizzi la funzione reale $f(x)$. Cerchiamo di determinare ora una serie di strumenti che possono essere utilizzati al punto 3. per la soluzione di problemi del tipo (1.1). Una valida guida sul panorama del software disponibile, per risolvere problemi di programmazione matematica, si può trovare nel libro “*Optimization Software Guide*”, di Jorge J. More' e Stephen J. Wright, SIAM Frontiers in Applied Mathematics 14. In maniera più mirata nelle presenti note, per problemi legati alla

- praticità d'uso dei solutori,
- facile reperibilità del software,
- necessità di utilizzare strumenti e metodologie aggiornate per la soluzione di problemi di programmazione matematica,

proponiamo per la soluzione di (1.1) l'uso dei seguenti strumenti dei quali diamo contestualmente anche una breve descrizione per facilitarne l'utilizzo:

- [1] il software disponibile sul **NEOS Server**,
- [2] il software disponibile sul **sito di AMPL**,
- [3] il software **Student Edition** scaricabile dal sito di **AMPL**,
- [4] il solutore di **Microsoft Excel** (interno al foglio Excel).

Osserviamo preliminarmente che [1] è un'interfaccia molto semplice , disponibile sulla rete Internet, per consentire l'uso di specifici strumenti della programmazione matematica. In sintesi l'interfaccia consente all'utente di inviare al sito il modello matematico di un problema, scritto in uno dei formati standard della programmazione matematica. Successivamente, è possibile disporre di una collezione di solutori, messi a disposizione liberamente dai rispettivi autori, ciascuno dei quali è in genere mirato alla risoluzione di una determinata classe di problemi del tipo

- *lineari*,
- *non lineari*,
- *a variabili intere*,
- *a variabili miste*,
- *non continuamente differenziabili*,
- *vincolati*,
- *non vincolati*,
- *di ottimizzazione globale*,
- *di ottimizzazione su reti*,
- *ecc..*

Come indicato al punto 3., questo richiede spesso l'accurata scelta del solutore più appropriato per il problema corrente. D'altra parte poi, la scelta oculata garantisce: *quasi sempre* risultati più attendibili e/o più accurati, *spesso* tempi significativamente inferiori per ottenere la soluzione cercata.

Per accedere al NEOS Server è necessario collegarsi al sito

<http://www.neos-server.org/neos/>

e scegliere l'opzione **NEOS Solvers**. A questo punto l'utente può scegliere una categoria specifica di problemi (alla quale afferisce il problema da risolvere), oppure nell'indecisione del solutore da scegliere può consultare l'**Optimization Tree**. Qui troverà una classificazione facilmente consultabile dei problemi dell'ottimizzazione, la quale include per ogni categoria di problemi anche alcune note inerenti la struttura, le modalità di soluzione ed una bibliografia relativa.

Supponendo per esempio che il problema in oggetto appartenga alla categoria **Nonlinearly Constrained Optimization**, si otterrà una lista di solutori; ciascuno di essi può essere usato per risolvere una particolare sottoclasse di problemi di ottimizzazione nonlineare vincolata. Si osservi che affianco a ciascun solutore vi sono alcune opzioni (e.g. MINOS [AMPL Input] [GAMS Input]) le quali rappresentano i corrispondenti formati che può assumere l'input del problema da far risolvere al solutore.

I più diffusi tra questi formati sono AMPL e GAMS. In pratica questi ultimi sono *linguaggi di modellazione*, specificatamente studiati per descrivere con la propria sintassi un modello in modo semplice e completo per l'utente. In questo corso si farà unicamente riferimento al modellatore AMPL [FGK89], per la grande diffusione che ha quest'ultimo (link per download gratuito di software, manuali e bibliografia all'URL <http://www.ampl.com>), mostrando alcuni esempi con i quali rappresentare il modello corrente.

Supponendo allora di voler costruire il modello di un problema, che chiameremo per comodità *modello*, la rappresentazione mediante AMPL richiede di generare la coppia di files ¹:

<i>modello.mod</i>	(obbligatorio)
<i>modello.dat</i>	(opzionale).

Il primo deve essere presente nella descrizione di qualsiasi modello in quanto ne costituisce la descrizione stessa, il secondo può essere omissso in quanto contiene i dati associati al problema, i quali possono eventualmente essere inclusi direttamente nel file *modello.mod*.

Una volta disponibile la rappresentazione del modello mediante i files *modello.mod* e *modello.dat*, supponiamo di scegliere il solutore LOQO e l'opzione **WWW Form** (rettangolo in alto e destra). Si aprirà una finestra nella quale sarà richiesto di inserire i files *modello.mod* (**AMPL model(local file)**) e *modello.dat* (**AMPL data(local file)**), nonchè un eventuale file di comandi opzionale (**AMPL commands(local file)**) che contenga particolari specifiche riguardanti la soluzione del problema, ovvero consente di formattare l'output del solutore e modificarne le specifiche (precisione della soluzione, numero massimo di iterazioni consentite, tempo di calcolo massimo, etc.).

Specificando un indirizzo di posta elettronica, una volta risolto il problema, il risultato sarà (anche) inviato a tale indirizzo sotto forma di report in formato testo. Infine per risolvere il problema scegliere **Submit to NEOS** ed attendere. Un esempio di problema rappresentato mediante il solo file *.mod* e risolto mediante KNITRO, è descritto nel paragrafo 1.1.2.

¹Si può opzionalmente generare un terzo file *modello.run* che contenga comandi ed opzioni di compilazione, nonchè direttive per la visualizzazione finale dei risultati

Per quanto riguarda [2], si tratta di uno strumento molto simile al precedente, disponibile sul sito:

<http://www.ampl.com/TRYAMPL/startup.html>

Anche in questo caso le modalità per risolvere il modello di un problema di programmazione matematica ricalcano quelle descritte per il punto [1]. In questo caso bisogna generare solo i files `.mod` e `.dat`, mentre non è richiesto il file `.run`. A differenza del NEOS Server è indispensabile scrivere i predetti files in formato AMPL (sono pertanto esclusi i formati GAMS, MPS, LP, XMPS, GLP, etc., disponibili per alcuni solutori sul NEOS Server). Infine la scelta del solutore è ristretta ad un sottoinsieme delle possibilità offerte dal NEOS Server.

Relativamente al punto [3], è possibile scaricare gratuitamente dal sito di AMPL la versione *Student Edition* del modellatore-Solutore AMPL. A tal fine è necessario collegarsi al sito:

<http://www.ampl.com/DOWNLOADS/index.html>

e scaricare il file `ampl.exe.gz` oppure (consigliato) `amplcml.zip` e scompattarlo (è opportuno a questo punto leggersi i files `README` e `readme.sw`, nonché scaricare la documentazione all'URL

<http://www.ampl.com/BOOK/>

per avere un'iniziale panoramica riguardo al modellatore-solutore). Si noti che l'opzione [3] prevede l'uso di comandi semplici, immessi da riga di comando, descritti nella documentazione di cui al sito precedente.

Il [4] rappresenta un toolbox all'interno del programma Excel della Microsoft, orientato alla soluzione di problemi di *Programmazione Lineare* e *Programmazione Lineare Intera*, ma consente anche di poter risolvere semplici problemi di *Programmazione NonLineare*. Per utilizzare il solutore, scegliere nella barra degli strumenti **Strumenti (Tools)**, **Componenti aggiuntivi (Add-ins)**, **Aggiunta risolutore (Solver)**. A questo punto, è possibile richiamare il solutore direttamente dal menù **Strumenti (Tools)**. L'help in linea di Excel fornisce le istruzioni per descrivere e risolvere un modello di programmazione matematica. Ulteriori manuali di riferimento, bibliografia e soluzione rapida di problemi collegati all'uso del solver di Excel sono disponibili nel sito <http://www.solver.com/suppstdsolver.htm>. Concludiamo questa sezione suggerendo allo studente di avvalersi preferibilmente delle opzioni [1], [2] e [3], in quanto evidentemente rappresentano strumenti più flessibili per la soluzione di problemi di programmazione matematica.

1.1.1 Esempio 1 (non convesso)

Supponiamo di voler risolvere il semplice problema di programmazione matematica descritto di seguito:

Dato il parametro $L > 0$, determinare i lati a , b , c del triangolo di area massima il cui perimetro sia dato da L (si consideri il caso numerico $L = 100$).

Da semplici considerazioni di tipo geometrico e/o algebrico è possibile dimostrare che la soluzione del problema è data da $a = b = c = L/3$, per ogni valore positivo del parametro L . Un ² possibile modello (non completamente corretto) nel linguaggio di programmazione matematica AMPL è il seguente

²Ribadiamo che un medesimo problema può essere riformulato in maniera equivalente in modi diversi tra loro. Nel secondo esempio di questa sezione se ne darà una dimostrazione.

area_triangolo.mod

```
##### Calcolo lati x,y,z del triangolo di area massima, #####
##### supponendone assegnato il perimetro (L). #####

# definizione parametri
param L >=0;

# definizione di variabili
var x:=0, >=0;
var y:=0, >=0;
var z:=0, >=0;

# definizione funzione obiettivo (formula di Erone per il calcolo dell'area)
maximize area_triangolo: sqrt(L/2*(L/2-x)*(L/2-y)*(L/2-z));

# introduzione vincoli
s.t. vinc_perimetrico: x + y + z = L;
```

area_triangolo.dat

```
##### Calcolo lati a,b,c del triangolo di area massima, #####
##### supponendone assegnato il perimetro (L). #####

# assegnazione parametri
param L := 100;
```

area_triangolo.run

```
##### Calcolo lati a,b,c del triangolo di area massima, #####
##### supponendone assegnato il perimetro (L). #####

# soluzione problema
solve;

# visualizzazione valori incognite all'ottimo
display x , y , z;
```

Per risolvere il precedente modello sono stati utilizzati i seguenti *solutori per problemi nonlineari vincolati*, richiamati dal NEOS Server, fornendo i seguenti risultati:

- MINOS 5.5:

```
MINOS 5.5: outlev=1
timing=1
Error evaluating objective area_triangolo: can't evaluate sqrt(-6.25e+06).
MINOS times:
```

```

read:      0.00
solve:     0.00
write:     0.00
total:     0.00
MINOS 5.5: solution aborted.
0 iterations, objective 0
Nonlin evals: obj = 1, grad = 1.
x = 100
y = 0
z = 0

```

- LANCELOT:

```

LANCELOT: problem solved.
5 iterations; objective 481.1251941
M = 1, N = 3, nslacks = 0, NEL = 1, NG = 2
NFCALL = 6, NGCALL = 6
x = 33.3333
y = 33.3333
z = 33.3333

```

- KNITRO:

```

Final objective value           = 4.81125224324688e+02
Final feasibility error (abs / rel) = 0.00e+00 / 0.00e+00
Final optimality error (abs / rel) = 3.55e-15 / 2.46e-16
# of iterations (major / minor)  = 5 / 6
# of function evaluations        = 17
# of gradient evaluations        = 4
# of Hessian evaluations         = 6
Total program time (sec)         = 0.00
KNITRO 4.0.0: LOCALLY OPTIMAL SOLUTION FOUND.
x = 33.3333
y = 33.3333
z = 33.3333

```

- MOSEK 3:

```

.
.
MOSEK      - problem type           : GECON (general convex optimization problem)
.
.
MOSEK finished. (interior-point iterations - 9, simplex iterations - 0)
Problem status      : PRIMAL_AND_DUAL_FEASIBLE

```



```

Solution status   : OPTIMAL
Primal objective  : 481.1252243
Dual objective    : 481.1252254
x = 33.3333
y = 33.3333
z = 33.3333

```

- PCX (solutore lineare!) :

```

Sorry, pcx can't handle nonlinearities.
exit code 2
<BREAK>

```

Si osservi che mentre l'ultimo solutore è dichiarato per problemi lineari e quindi ragionevolmente non riesce a risolvere il problema nonlineare proposto, i primi 4 solutori non forniscono la stessa soluzione. In particolare il primo sembrerebbe apparentemente il peggiore, dichiarando un fallimento. Però, a ben guardare il modello presenta un vizio, in quanto il punto iniziale $x = 0, y = 0, z = 0$, **non è ammissibile**. Quest'ultimo aspetto viene sottaciuto dai solutori usati, i quali certamente si accorgono della cosa, ma evidentemente o non ne sono influenzati oppure hanno il modo di generano autonomamente i rispettivi punti iniziali. Tra i solutori usati solamente MINOS 5.5 genera un punto (100,0,0) che pur ammissibile, non appartiene al dominio della funzione obiettivo (radice quadrata), pertanto segnala un errore. Gli altri solutori nonlineari apparentemente sembrano non soffrire dello stesso problema e convergono effettivamente alla soluzione. Si noti infine che modificando nel file `area.triangolo.mod` le istruzioni

```

var x:=0, >=0;
var y:=0, >=0;
var z:=0, >=0;

```

nelle (cambio del punto iniziale)

```

var x:=40, >=0;
var y:=40, >=0;
var z:=20, >=0;

```

si ottiene anche per MINOS 5.5 la soluzione corretta

```

MINOS 5.5: outlev=1
timing=1
MINOS times:
read:      0.00
solve:     0.01   excluding minos setup: 0.00
write:     0.00
total:     0.01
MINOS 5.5: optimal solution found.
6 iterations, objective 481.1252243
Nonlin evals: obj = 12, grad = 11.
x = 33.3333
y = 33.3333
z = 33.3333

```

1.1.2 Esempio 2

Supponiamo ora di voler risolvere un altro semplice problema di programmazione matematica:

Sia dato l'intero $N > 0$ e la sfera in \mathbb{R}^3 di centro l'origine e raggio $R > 0$, determinare la posizione di N punti sulla sfera, tali che questi risultino a distanza massima tra loro (si consideri il caso numerico $R = 100$ e $N = 2$).

Il problema proposto, pur nella sua semplicità, ha interessanti risvolti applicativi. Per esempio, la messa in orbita di un prefissato numero di satelliti in orbita geostazionaria, risponde alle specifiche di questo problema. Evidentemente nel caso $N = 2$ i punti devono trovarsi da parti opposte rispetto ad un qualsiasi asse della sfera (i.e. agli antipodi uno rispetto all'altro), pertanto ci aspetteremmo ∞^2 soluzioni. Una possibile formulazione in AMPL per tale problema, è la seguente (si noti che il file `.dat` non è necessario e che nella seguente formulazione, i valori dei parametri sono assegnati direttamente nel file `.mod`):

distribuzione_tridimensionale.mod

```
-----
#### Calcolo distribuzione superficiale estesa per N ####
#### punti a distanza fissa R dall'origine ####

# definizione parametri
param R := 100;
param N := 2;

# definizione di variabili
var x{j in 1..N} := 0;
var y{j in 1..N} := 0;
var z{j in 1..N} := 0;
var t:=0, >=0;

# definizione funzione obiettivo
maximize minima_distanza: t;

# introduzione vincoli
s.t. vinc_distanza{i in 1..N, j in 1..N : j<>i}:
      t <= sqrt((x[i]-x[j])**2 + (y[i]-y[j])**2 + (z[i]-z[j])**2);
s.t. vinc_perimetrico{i in 1..N}: sqrt(x[i]**2 + y[i]**2 + z[i]**2) <= R;
-----
```

ed un possibile file di comandi (`distribuzione_tridimensionale.run`) che consenta di visualizzare funzione obiettivo e variabili all'ottimo è il seguente

distribuzione_tridimensionale.run

```
-----
#### Calcolo distribuzione superficiale estesa per N ####
#### punti a distanza fissa R dall'origine ####

# soluzione problema
solve;

# visualizzazione vettori delle incognite all'ottimo
display x, y, z;
-----
```

Si osservi preliminarmente che il punto iniziale scelto è ammissibile e la specifica $j \neq i$ nei vincoli `vinc_distanza` ha lo scopo di evitare che i punti della distribuzione possano sovrapporsi (la qual cosa fornirebbe un valore della funzione obiettivo pari a 0). Inoltre il significato dei vincoli è il seguente: la prima serie di $N^2 - N$ vincoli impone che la variabile nonnegativa t sia non inferiore alla distanza tra una qualunque coppia di punti, le seconda serie di N vincoli impone che i punti della distribuzione siano contenuti nella regione di spazio delimitata dalla sfera. Infine, la funzione obiettivo viene massimizzata, imponendo (implicitamente) che il valore massimo di t sia il più alto consentito, quindi la distanza tra i punti della distribuzione sia massima, compatibilmente con i vincoli. Per risolvere il problema usiamo il solutore MINOS 5.5 mediante l'interfaccia con il NEOS Server. Otteniamo il risultato:

```
MINOS 5.5: outlev=1
timing=1
Error evaluating constraint vinc_distanza[1,2]: can't evaluate sqrt'(0).
MINOS times:
  read:      0.01
  solve:     0.00
  write:     0.00
  total:     0.01
MINOS 5.5: solution aborted.
0 iterations, objective 0
Nonlin evals: constrs = 1, Jac = 1.
:   x   y   z   :=
1   0   0   0
2   0   0   0;
```

ovvero un apparente fallimento dovuto all'impossibilità di eseguire l'operazione $\sqrt{0}$. Naturalmente il problema sta nel fatto che il solutore tenta di valutare oltre ai vincoli anche il loro Jacobiano nell'origine (in altre parole l'operazione illecita è causata dalla presenza del termine $\sqrt{0}$ in un denominatore). Per evitare questo fatto riformuliamo il problema come segue (il file `.run` rimane inalterato)

distribuzione_tridimensionale.mod

```
-----
#### Calcolo distribuzione superficiale estesa per N ####
#### punti a distanza fissa R dall'origine                ####

# definizione parametri
param R := 100;
param N := 2;

# definizione di variabili
var x{j in 1..N} := 0;
var y{j in 1..N} := 0;
var z{j in 1..N} := 0;
var t:=0, >=0;

# definizione funzione obiettivo
maximize minima_distanza: t**2;
```

```
# introduzione vincoli
s.t. vinc_distanza{i in 1..N, j in 1..N : j<>i}:
      t**2 <= (x[i]-x[j])**2 + (y[i]-y[j])**2 + (z[i]-z[j])**2;
s.t. vinc_perimetrico{i in 1..N}: x[i]**2 + y[i]**2 + z[i]**2 <= R**2;
-----
```

nel quale i vincoli ora sono continuamente differenziabili su tutto \mathbb{R}^7 . La soluzione di MINOS 5.5 con la nuova formulazione è data da

```
MINOS 5.5: outlev=1
timing=1
  MINOS times:
    read:      0.00
    solve:     0.00
    write:     0.00
    total:     0.00
MINOS 5.5: optimal solution found.
0 iterations, objective 0
Nonlin evals: obj = 3, grad = 2, constrs = 3, Jac = 2.
:   x   y   z   :=
1   0   0   0
2   0   0   0;
```

nella quale *apparentemente* il solutore sembra aver risolto il problema senza difficoltà (addirittura 0 iterazioni). In realtà il solutore in questione è rimasto “intrappolato” in un minimo locale (l’origine degli assi), da cui non è stato capace di uscire per trovare la soluzione ottima cercata. Naturalmente allora, perturbando il punto iniziale fornito al problema ci aspetteremmo che la soluzione sia regolarmente trovata. In particolare modificando il punto iniziale come segue (ma si invita lo studente a provare altre modifiche per convincersi della dinamica del solutore)

```
# definizione di variabili
var x{j in 1..N} := 0;
var y{j in 1..N} := 0;
var z{j in 1..N} := 5*(-1)^j;
var t:=5, >=0;
```

si ottiene il seguente risultato corretto, dopo 114 iterazioni

```
MINOS 5.5: outlev=1
timing=1
  MINOS times:
    read:      0.00
    solve:     0.03
    write:     0.00
    total:     0.03
MINOS 5.5: optimal solution found.
114 iterations, objective 40000
Nonlin evals: obj = 292, grad = 291, constrs = 292, Jac = 291.
:   x   y   z   :=
1   0   0  -100
2   0   0  100.
```

1.2 Esercitazione 2: esempi di Programmazione Matematica

1.2.1 Esempio 1 (numerico - lineare)

Consideriamo qui un semplice problema numerico di Programmazione Lineare, i.e. le funzioni coinvolte nel modello sono funzioni affini. Essendo un problema lineare, esso appartiene naturalmente alla categoria dei *problemi convessi*, per i quali sono ben note alcune proprietà associate alle soluzioni relative. Il problema assegnato è il seguente:

$$\begin{aligned} \min \quad & -x -z \\ & 3x + 3y + 3z \leq 5 \\ & x \geq 0.2 \\ & 10z \geq 3x \\ & y \geq 0, z \geq 0 \end{aligned}$$

di cui una possibile formulazione in AMPL (utilizzando per brevità solo il file `.mod`), è data da

numerico.mod

Problema numerico

```
var x >=0.2;  
var y >=0;  
var z >=0;
```

```
minimize obiettivo: -x-z;
```

```
s.t. vincolo_1: 3*x+3*y+3*z <= 5;  
s.t. vincolo_2: 10*z >=3*x;
```

```
#option solver cplex;  
#option solver minos;  
solve;  
display x,y,z;
```

che abbiamo cercato di risolvere utilizzando innanzitutto i solutori (versione *Student Edition*) contenuti nel file `amplcmp.zip` di cui al punto [3] della precedente sezione. Scommentando nel file `numerico.mod` una delle due opzioni

```
#option solver cplex;  
#option solver minos;
```

possiamo scegliere se utilizzare il solutore **Cplex 8.0.0** (solutore specifico per problemi lineari e quadratici), oppure il solutore **minos 5.5** (solutore nonlineare). Si ricorda che una volta installata l'interfaccia **sw**, per poter usare AMPL è necessario dare il comando `ampl`, il quale restituisce il prompt `ampl`, indicando così che AMPL è disponibile. Per far risolvere il modello, supponendo che il percorso del file `numerico.mod` sia `C:\Didattica\numerico.mod`, basterà dare il comando

```
model C:\Didattica\numerico.mod;
```

e la soluzione verrà fornita direttamente. A questo punto si noterà che entrambi i solutori trovano la medesima soluzione

```

optimal solution; objective -1.666666667
ampl: display x,y,z;
x = 1.28205
y = 0
z = 0.384615

```

(**Attenzione!** si osservi che se si tenta di far risolvere il modello due volte di seguito, anche con solutori diversi, verranno segnalati errori di sintassi anche se il modello è corretto. Per evitare questo, digitare il comando

```
reset;
```

prima di risolvere nuovamente il modello).

Se ora ci si collega al NEOS Server e si utilizzano i 3 solutori MOSEK 3, KNITRO e BPMPD-AMPL, si ottenendo le soluzioni

- **MOSEK 3:**

```

MOSEK finished. (interior-point iterations - 0, simplex iterations - 0)
Problem status      : PRIMAL_AND_DUAL_FEASIBLE
Solution status     : OPTIMAL
Primal objective    : -1.666666667
Dual objective      : -1.666666667

x = 1.28205
y = 0
z = 0.384615

```

- **KNITRO:**

```

Final Statistics
-----
Final objective value           = -1.66666664254926e+00
Final feasibility error (abs / rel) = 0.00e+00 / 0.00e+00
Final optimality error (abs / rel) = 1.35e-08 / 1.35e-08
# of iterations (major / minor)   = 6 / 6
# of function evaluations         = 7
# of gradient evaluations         = 7
# of Hessian evaluations          = 6
Total program time (sec)         = 0.00

```

```
KNITRO 4.0.0: LOCALLY OPTIMAL SOLUTION FOUND.
```

```

x = 0.283342
y = 1.05964e-08
z = 1.38332

```

- **BPMPD-AMPL:**

```

BPMPD 2.11: Optimal solution found, objective -1.666666667
4 iterations, 8 corrections (2.00 per iter.)
x = 1.20814
y = 0
z = 0.458524

```

Quindi le soluzioni non coincidono, ma questo a ben guardare non dovrebbe stupire in quanto il problema è **convesso** ma **non strettamente convesso**, pertanto è possibile che ammetta più soluzioni, *tutte*

corrispondenti al medesimo valore della funzione obiettivo, ma con coordinate dei punti diverse tra loro. Si dimostri che dati i due punti soluzione $P1$ e $P2$ del problema sopra, il nuovo punto

$$\alpha P1 + (1 - \alpha)P2, \quad \alpha \in [0, 1]$$

è un punto ammissibile al quale corrisponde il valore della funzione obiettivo -1.66666666.

1.2.2 Esempio 2 (Regressione Lineare)

Si considererà ora un esempio reale di uso della Regressione Lineare. Molto spesso in medicina, per determinare l'efficacia di un farmaco, si tende a valutare gli effetti di un protocollo di sperimentazione del farmaco stesso su un campione significativo di individui curati. In particolare, la sperimentazione spesso prende in esame cavie da laboratorio alle quali viene somministrato il farmaco in dosi controllate. Il seguente esempio di uso in farmacologia della regressione lineare è stato tratto dal sito

<http://www2.unipr.it/bottarel/epi/cause/regress.htm>

Si supponga che tra gli effetti indesiderati di un certo farmaco, si annoveri quello di innalzare la pressione arteriosa. Verifichiamo questa ipotesi attraverso un esperimento: somministriamo dosi crescenti del farmaco ad alcune cavie, misurando poi la variazione della pressione diastolica che si verifica dopo la somministrazione.

In dettaglio, vengono utilizzati 16 ratti, suddivisi in 8 gruppi di 2 animali ciascuno. Il primo gruppo è di controllo e non viene trattato; ai gruppi successivi il farmaco viene somministrato secondo le seguenti dosi:

Gruppo	Dose
2	1 mg/kg
3	2 mg/kg
4	3 mg/kg
5	4 mg/kg
6	5 mg/kg
7	6 mg/kg
8	7 mg/kg

Quindi, per esempio, a ciascuna delle due cavie del gruppo 6 vengono somministrati 5 mg/kg di farmaco. I risultati della pressione diastolica sono riassunti nella tabella che segue.

Variazione della pressione arteriosa (mm/Hg) dopo la somministrazione								
Dose (mg/Kg)	0	1	2	3	4	5	6	7
cavia 1	-2	1	6	5	6	10	8	11
cavia 2	0	5	7	9	9	7	15	12

Per valutare quantitativamente l'effetto del farmaco sull'innalzamento della pressione, determiniamo i due parametri m e q della retta di regressione

$$y = mx + q,$$

in cui $x \in \mathbb{R}$ rappresenta la dose di farmaco mentre $y \in \mathbb{R}$ rappresenta la pressione. I due parametri m e q risolvono il problema di regressione lineare

$$\min_{m,q} \sum_{i=1}^{16} [y_i - (mx_i + q)]^2,$$

in cui si ha per le sequenze $\{x_i\}$ ed $\{y_i\}$ la seguente tabella

	cavia 1								cavia 2							
i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
x_i	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
y_i	-2	1	6	5	6	10	8	11	0	5	7	9	9	7	15	12

Una possibile formulazione in AMPL, utilizzando sia il file .mod che il file .dat, è data da

regressione_lineare.mod

```
-----  
####      Problema di regressione lineare      ####  
  
# definizione insiemi  
set CASI := 1..16;  
  
# definizione parametri  
param x{CASI} >=0;  
param y{CASI};  
  
# definizione di variabili  
var m;  
var q;  
  
# definizione funzione obiettivo  
minimize somma_quadrati: sum{i in CASI} (y[i] - m*x[i] - q)^2;  
-----
```

regressione_lineare.dat

```
-----  
####      Problema di regressione lineare      ####  
  
# assegnazione parametri  
param x :=  
  1  0  
  2  1  
  3  2  
  4  3  
  5  4  
  6  5  
  7  6  
  8  7  
  9  0  
 10  1  
 11  2  
 12  3  
 13  4  
 14  5  
 15  6  
 16  7 ;  
  
param y :=  
  1 -2  
  2  1  
  3  6  
  4  5  
  5  6  
  6 10  
  7  8
```



```

8    11
9    0
10   5
11   7
12   9
13   9
14   7
15  15
16  12 ;

```

Usando il solutore MINOS 5.5 si ottiene la seguente soluzione:

```

MINOS 5.5: optimal solution found.
5 iterations, objective 76.625
Nonlin evals: obj = 10, grad = 9.
m = 1.625
q = 1.125

```

in cui essendo $m > 0$, appare chiaro come *al crescere del dosaggio del farmaco la pressione diastolica tenda a crescere*.

1.2.3 Esempio 3 (Case Study - lineare intero)

In questa sezione si considera la realizzazione di un modello di programmazione matematica, relativo ad un problema applicativo reale. Il problema in esame si riferisce ad una impresa denominata Jeppesen Sanderson, Inc., facente parte del gruppo Boeing. Questa azienda si occupa di fornire dettagliate informazioni a piloti di linea di molte compagnie aeree civili. In particolare si noti che la determinazione di carte per la navigazione aerea richiede un costante aggiornamento, al fine di verificare su base settimanale le informazioni atmosferiche e climatiche necessarie. Le informazioni vengono raccolte separatamente in *fascicoli* e questi poi riuniti in *ordinativi* (ovvero commesse), richiesti dalle compagnie aeree. Per la delicatezza dell'informazione e la necessità di un aggiornamento costante, la redazione dei fascicoli richiede una pianificazione accurata ed efficiente. Normalmente la Jeppesen fornisce circa 5-30 milioni di fascicoli settimanalmente a 200000 clienti in tutto il mondo. Qui ci limiteremo a descrivere ora una versione semplificata su base settimanale della gestione di fascicoli. In particolare si supporrà che

- l'orizzonte temporale si limita a **7 giorni**;
- il numero degli **ordinativi possibili** è pari a **4**;
- il **numero di fascicoli** per ogni ordine è al più **6**;
- sono presenti **3 macchine** per la lavorazione dei fascicoli (lavorazione in parallelo);
- sono noti i **tempi medi di lavorazione** e le **disponibilità orarie** relative.

Descrizione del problema

È previsto l'invio di 6 fascicoli distinti ($F = \{f1, \dots, f6\}$), organizzati in 4 ordinativi ($N = \{1, \dots, 4\}$), che devono essere processati nell'arco temporale di 7 giorni ($T = \{1, \dots, 7\}$). Sono disponibili 3 macchinari ($M = \{1, \dots, 3\}$) per la lavorazione (eventualmente in parallelo) dei fascicoli, inoltre ogni ordinativo richiede un sottoinsieme diverso ($SF[i]$, $i = 1, \dots, N$) di fascicoli. Per ogni giorno sono note le ore macchina ordinarie ($ore_ord[i]$, $i \in T$) e quelle straordinarie ($ore_str[i]$, $i \in T$) disponibili. Si considera anche il salario (orario) ordinario ($paga_ord$) e straordinario ($paga_str$) per la lavorazione dei fascicoli, nonché il tempo ($tempo[i, j]$) necessario a produrre ciascun ordinativo (i) su ciascuna macchina (j). Infine è prevista una penalizzazione ($pen[i, k]$) qualora venga evaso l'ordinativo i -simo nel giorno k -simo, inoltre per generare il fascicolo h -simo nel giorno k -simo è previsto un costo $ct[k, h]$.

Formulazione del modello

Osserviamo che per definire una formulazione di un problema è necessario passare ad una formalizzazione che prevede l'introduzione di 3 elementi:

- Definizione Variabili

$$\begin{aligned} x[i, j, k] &= \text{numero di fascicoli dell'ordinativo } i\text{-simo, } i = 1, \dots, 4, \text{ lavorati sul macchinario} \\ &\quad j = 1, \dots, 3, \text{ nel giorno } k = 1, \dots, 7. \\ y_1[h, k] &= \text{vale 1 se il fascicolo di tipo } h \in SF[1] \text{ è lavorato nel giorno } k = 1, \dots, 7, \text{ vale 0 altrimenti.} \\ &\dots \\ y_4[h, k] &= \text{vale 1 se il fascicolo di tipo } h \in SF[4] \text{ è lavorato nel giorno } k = 1, \dots, 7, \text{ 0 altrimenti.} \\ h_str[k] &= \text{numero ore di straordinario lavorate nel giorno } k = 1, \dots, 7. \end{aligned}$$

- Identificazione Vincoli

È necessario introdurre vincoli con i quali garantire che ciascun fascicolo di ciascun ordine viene lavorato nell'arco dell'orizzonte temporale di 7 giorni, pertanto sarà:

$$\sum_{k=1}^7 y_i[h, k] = 1, \quad i = 1, \dots, 4, \quad h \in SF[i].$$

Inoltre bisogna garantire vincoli di consistenza che leghino i vettori di variabili x ed y_i , sarà allora (si veda il quesito nella sezione 'Discussione', al termine dell'esempio !!):

$$\sum_{j=1}^3 \sum_{k=1}^7 x[i, j, k] = \sum_{h \in SF[i]} \sum_{k=1}^7 y_i[h, k] \quad i = 1, \dots, 4.$$

Sono poi definiti ovvi vincoli orari relativi alla disponibilità oraria giornaliera

$$h_str[k] \leq ore_str[k], \quad k = 1, \dots, 7,$$

nonchè vincoli sulla capacità produttiva giornaliera

$$\sum_{i=1}^4 \sum_{j=1}^3 tempo[i, j] \cdot x[i, j, k] \leq ore_ord[k] + ore_str[k], \quad k = 1, \dots, 7.$$

- Identificazione Funzione Obiettivo

$$\begin{aligned} \min \text{ costo_totale} &= \sum_{i=1}^4 \sum_{j=1}^3 \sum_{k=1}^7 pen[i, k] \cdot x[i, j, k] + paga_ord \cdot \sum_{i=1}^4 \sum_{j=1}^3 \sum_{k=1}^7 tempo[i, j] \cdot x[i, j, k] \\ &+ paga_str \cdot \sum_{k=1}^7 h_str[k] + \sum_{i=1}^4 \sum_{h \in SF[i]} \sum_{k=1}^7 ct[k, h] \cdot y_i[h, k] \end{aligned}$$

Realizzazione in AMPL

Una possibile formulazione in AMPL del problema della Jeppesen, introducendo entrambi i files `.mod` e `.dat`, è data da (si noti che nel file `.mod`) la sezione relativa alla dichiarazione dei `set` è divisa in due parti, in quanto la dichiarazione `set F` deve precedere la dichiarazione `param ct{1..T,F}`:

Jeppesen.mod

```

-----
#####
#### CASE STUDY: Jeppesen. Gestione informazioni per l'aviazione ####
#####

#### dichiarazione set del problema ####
set F;                                # insieme di tutti i fascicoli possibili

#### dichiarazione parametri del problema ####
param T;                              # giorni dell'orizzonte temporale
param M;                              # tipologie macchinari
param N;                              # numero degli ordini possibili
param ore_ord{1..T};                  # ore macchina disponibili ogni giorno
param ore_str{1..T};                  # ore straordinario disponibili ogni giorno
param paga_ord;                       # salario per lavoro ordinario
param paga_str;                       # salario per lavoro ordinario param
tempo{1..N,1..M};                     # tempo necessario a produrre ciascun
                                      # ordine su ciascuna macchina
param pen{1..N,1..T};                 # penalizzazione per aver prodotto ciascun
                                      # ordine in ciascun giorno
param ct{1..T,F};                     # costo per produrre ciascun
                                      # fascicolo in ciascun giorno

#### dichiarazione set del problema ####
set SF{i in 1..N} within F;           # insieme dei sottoinsiemi di fascicoli possibili

#### dichiarazione di variabili ####
var x{1..N,1..M,1..T} >=0, integer;    # numero di fascicoli di ciascun ordine
                                      # per ciascuna macchina ed in ogni giorno
var y1{h in SF[1],k in 1..T} binary;    # 1 se l'h-simo fascicolo del primo ordine
                                      # e' prodotto nel k-simo giorno
                                      # 0 altrimenti
var y2{SF[2],1..T} binary;              # analogamente ...
var y3{SF[3],1..T} binary;              # analogamente ...
var y4{SF[4],1..T} binary;              # analogamente ...
var h_str{1..T}, >=0, integer;          # ore di straordinario in ogni giorno

#### dichiarazione funzione obiettivo ####
minimize costo_totale: sum{i in 1..N, j in 1..M, k in 1..T} pen[i,k] * x[i,j,k] +
                        paga_ord*sum{i in 1..N, j in 1..M, k in 1..T} tempo[i,j] * x[i,j,k] +
                        paga_str*sum{k in 1..T} h_str[k] +
                        sum{h in SF[1], k in 1..T} ct[k,h] * y1[h,k] +
                        sum{h in SF[2], k in 1..T} ct[k,h] * y2[h,k] +
                        sum{h in SF[3], k in 1..T} ct[k,h] * y3[h,k] +
                        sum{h in SF[4], k in 1..T} ct[k,h] * y4[h,k];

#### introduzione vincoli ####
s.t. min_produzione_1{h in SF[1]}: sum{k in 1..T} y1[h,k] = 1;
s.t. min_produzione_2{h in SF[2]}: sum{k in 1..T} y2[h,k] = 1;
s.t. min_produzione_3{h in SF[3]}: sum{k in 1..T} y3[h,k] = 1;
s.t. min_produzione_4{h in SF[4]}: sum{k in 1..T} y4[h,k] = 1;

```

```

s.t.   vinc_prod_1: sum{j in 1..M, k in 1..T} x[1,j,k] = sum{h in SF[1], k in 1..T} y1[h,k];
s.t.   vinc_prod_2: sum{j in 1..M, k in 1..T} x[2,j,k] = sum{h in SF[2], k in 1..T} y2[h,k];
s.t.   vinc_prod_3: sum{j in 1..M, k in 1..T} x[3,j,k] = sum{h in SF[3], k in 1..T} y3[h,k];
s.t.   vinc_prod_4: sum{j in 1..M, k in 1..T} x[4,j,k] = sum{h in SF[4], k in 1..T} y4[h,k];

s.t.   ore_straordinario{k in 1..T}: h_str[k] <= ore_str[k];      # vincolo giornaliero sulle ore
                                           # di straordinario

s.t.   cap_produttiva{k in 1..T}: sum{i in 1..N, j in 1..M}
      tempo[i,j] * x[i,j,k] <= ore_ord[k] + ore_str[k]; # vincolo orario
-----

```

Jeppesen.dat

```

-----
#####
#### CASE STUDY: Jeppesen. Gestione informazioni per l'aviazione ####
#####

```

```

#### assegnazione set ####
set F:= f1 , f2 , f3 , f4 , f5 , f6;
set SF[1]:= f1 , f2;
set SF[2]:= f1 , f3 , f5;
set SF[3]:= f1 , f2 , f3 , f4 , f5 , f6;
set SF[4]:= f2 , f4 , f6;

```

```

#### assegnazione parametri ####

```

```

param T:= 7;
param N:= 4;
param M:= 3;
param ore_ord:=

```

```

    1      3
    2     30
    3     40
    4     20
    5     40
    6     40
    7     80;

```

```

param ore_str:=

```

```

    1      0
    2     20
    3     10
    4      5
    5     10
    6     10
    7      0;

```

```

param paga_ord:= 1100;

```

```

param paga_str:= 750;

```

```

param tempo:   1   2   3 :=

```

```

    1   8  20  15
    2  15  14  14
    3  15   5   4
    4  15  40  10;

```

```

param pen:   1   2   3   4   5   6   7:=

```

```

    1   2.3 3.1 5.3 2.6 3.0 3.0 1.4

```

```

2  1.3 4.1 5.3 1.6 3.0 5.0 1.7
3  1.5 3.1 5.3 2.9 3.0 5.0 1.4
4  3.4 3.6 5.3 3.6 3.0 3.0 1.4;
param ct:   f1  f2  f3  f4  f5  f6 :=
1  100 100 100 100 100 100
2  100 100 100 100 100 100
3  200 200 200 200 200 200
4  200 200 200 200 200 200
5  200 200 200 200 200 200
6  300 300 300 300 300 300
7  400 400 400 400 400 400;

```

Inserendo in `ampl` (tramite l'interfaccia `sw`) i seguenti comandi

```

ampl: reset;
ampl: model ..\..\Jeppesen.mod
ampl: data ..\..\Jeppesen.dat
ampl: option solver cplex;
ampl: solve;
ampl: display x;

```

si ottiene il seguente risultato (`objective 124623` rappresenta il valore all'ottimo della funzione obiettivo, 5 MIP indica che sono state svolte 5 iterazioni del metodo risolutivo)

```

CPLEX 8.0.0: optimal integer solution within mipgap or absmipgap; objective 124623
5 MIP simplex iterations
0 branch-and-bound nodes

```

```

x [*,1,*] (tr)
:   1   2   3   4   :=
1   0   0   0   0
2   0   0   0   0
3   0   0   0   0
4   0   0   0   0
5   0   0   0   0
6   0   0   0   0
7   2   0   0   0

```

```

[*,2,*] (tr)
:   1   2   3   4   :=
1   0   0   0   0
2   0   0   0   0
3   0   0   0   0
4   0   1   0   0
5   0   0   0   0
6   0   0   0   0
7   0   0   0   0

```

```

[*,3,*] (tr)
:   1   2   3   4   :=
1   0   0   0   0
2   0   0   0   0
3   0   0   0   0
4   0   0   0   0

```

```

5  0  2  0  0
6  0  0  0  0
7  0  0  6  3
;

h_str [*] :=
1  0
2  0
3  0
4  0
5  0
6  0
7  0
;

```

Discussione

- Si invita lo studente a modificare il parametro `tempo` e verificare i cambiamenti della soluzione al variare di tale parametro. Per esempio si noti che moltiplicando per un fattore 10 le 12 componenti del parametro `tempo`, il solutore fornisce la seguente soluzione

```

presolve, constraint min_produzione_4['f6']:
    all variables eliminated, but lower bound = 1 > 0
presolve, constraint min_produzione_4['f4']:
    all variables eliminated, but lower bound = 1 > 0
presolve, constraint min_produzione_4['f2']:
    all variables eliminated, but lower bound = 1 > 0
presolve, constraint min_produzione_2['f5']:
    all variables eliminated, but lower bound = 1 > 0
presolve, constraint min_produzione_2['f3']:
    all variables eliminated, but lower bound = 1 > 0
1 presolve messages suppressed.

```

Ad un'attenta interpretazione il risultato precedente rivela che l'insieme ammissibile del modello risulta *vuoto*, quindi la formulazione non è corretta oppure il problema originale non ammette soluzione.

È corretta la formulazione complessiva del modello (si rifletta AUTONOMAMENTE sulla definizione del blocco di vincoli che legano le variabili x e y_i) ?

Suggerimento: In realtà guardando solo i valori all'ottimo delle variabili $x[i, j, k]$ sembrerebbe di sì, ma se si richiede al solutore di fornire anche le variabili $y_i[h, k]$ si ottiene:

```

ampl: display y1;
y1 :=
f1 1  1
f1 2  0
f1 3  0
f1 4  0
f1 5  0
f1 6  0
f1 7  0
f2 1  1
f2 2  0
f2 3  0
f2 4  0
f2 5  0

```

```

f2 6    0
f2 7    0
;

ampl: display y2;
y2 [*,*] (tr)
:  f1  f3  f5      :=
1   0   1   1
2   1   0   0
3   0   0   0
4   0   0   0
5   0   0   0
6   0   0   0
7   0   0   0
;

ampl: display y3;
y3 [*,*] (tr)
:  f1  f2  f3  f4  f5  f6      :=
1   0   1   1   0   1   1
2   1   0   0   1   0   0
3   0   0   0   0   0   0
4   0   0   0   0   0   0
5   0   0   0   0   0   0
6   0   0   0   0   0   0
7   0   0   0   0   0   0
;

ampl: display y4;
y4 [*,*] (tr)
:  f2  f4  f6      :=
1   0   1   1
2   1   0   0
3   0   0   0
4   0   0   0
5   0   0   0
6   0   0   0
7   0   0   0
;

```

e confrontando i valori dei vettori di variabili x ed y_i , $i = 1, \dots, 4$ si ottiene un'evidente contraddizione. Infatti, per esempio si noti che risulta $x[1, 1, 7] = 2$, ovvero i due fascicoli (f1 ed f2) relativi al primo ordinativo, vengono prodotti sulla prima macchina, il *settimo* giorno. Ma questo risultato è in contraddizione con $y_1[f1, 1] = y_1[f2, 1] = 1$ (i.e. i fascicoli f1 ed f2 relativi al primo ordinativo sono prodotti il *primo* giorno). L'errore sta nell'aver formulato erroneamente il secondo blocco di vincoli. In particolare tali vincoli

$$\sum_{j=1}^3 \sum_{k=1}^7 x[i, j, k] = \sum_{h \in SF[i]} \sum_{k=1}^7 y_i[h, k] \quad i = 1, \dots, 4$$

devono essere sostituiti con i nuovi vincoli (che impongono condizioni più stringenti in quanto valide in ciascun giorno e non solo complessivamente nell'arco dell'intera settimana)

$$\sum_{j=1}^3 x[i, j, k] = \sum_{h \in SF[i]} y_i[h, k] \quad i = 1, \dots, 4, \quad k = 1, \dots, 7.$$

Si otterranno così i seguenti risultati (di cui si invita a verificare la correttezza).

CPLEX 8.0.0: optimal integer solution; objective 125142

56 MIP simplex iterations

0 branch-and-bound nodes

ampl: display x;

x [*,1,*] (tr)

:	1	2	3	4	:=
1	0	0	0	0	
2	2	0	0	0	
3	0	0	0	0	
4	0	0	0	0	
5	0	0	0	0	
6	0	0	0	0	
7	0	0	0	0	

[*,2,*] (tr)

:	1	2	3	4	:=
1	0	0	0	0	
2	0	0	0	0	
3	0	0	0	0	
4	0	0	0	0	
5	0	0	0	0	
6	0	0	0	0	
7	0	0	0	0	

[*,3,*] (tr)

:	1	2	3	4	:=
1	0	0	0	0	
2	0	0	6	1	
3	0	0	0	0	
4	0	1	0	0	
5	0	2	0	2	
6	0	0	0	0	
7	0	0	0	0	

;

ampl: display y1;

y1 :=

f1 1	0
f1 2	1
f1 3	0
f1 4	0
f1 5	0
f1 6	0
f1 7	0
f2 1	0
f2 2	1
f2 3	0
f2 4	0
f2 5	0
f2 6	0
f2 7	0

;


```

ampl: display y2;
y2 [*,*] (tr)
:  f1  f3  f5      :=
1   0   0   0
2   0   0   0
3   0   0   0
4   1   0   0
5   0   1   1
6   0   0   0
7   0   0   0
;

```

```

ampl: display y3;
y3 [*,*] (tr)
:  f1  f2  f3  f4  f5  f6      :=
1   0   0   0   0   0   0
2   1   1   1   1   1   1
3   0   0   0   0   0   0
4   0   0   0   0   0   0
5   0   0   0   0   0   0
6   0   0   0   0   0   0
7   0   0   0   0   0   0
;

```

```

ampl: display y4;
y4 [*,*] (tr)
:  f2  f4  f6      :=
1   0   0   0
2   0   0   1
3   0   0   0
4   0   0   0
5   1   1   0
6   0   0   0
7   0   0   0
;

```

(per es. ora risulta correttamente $x[1,1,2] = 2$ e $y_1[f1,2] = y_1[f2,2] = 1$). Infine si osservi che il valore ottimo della funzione obiettivo risulta **peggiorato** (problema di minimizzazione), inoltre i vettori delle variabili all'ottimo sono completamente differenti. Questo è conseguenza dell'aver risolto un nuovo modello (sostituendo il secondo blocco di vincoli) in cui l'insieme ammissibile risulta essere contenuto nell'insieme ammissibile del modello precedente (si lascia per esercizio la verifica di quest'ultima proprietà).

1.3 Esercitazione 3: Esempio di Programmazione Matematica nonlineare

1.3.1 Progetto di una struttura a traliccio

In questa sezione introdurremo brevemente alcune nozioni molto semplificate di calcolo strutturale, che verranno utilizzate successivamente per progettare una struttura a *traliccio*, ovvero una struttura data da un insieme di punti (*nodi*) raccordati attraverso segmenti (*barre*) supposti tutti dal medesimo materiale [BN01]. Per definire la struttura del traliccio è necessario fissare un riferimento cartesiano nel quale definire le coordinate dei nodi e le lunghezze delle barre, nonché i *carichi* che agiscono su ciascun nodo. Volendo limitare lo studio del traliccio nel piano bidimensionale, i nodi nonché i carichi verranno espressi come vettori a 2 componenti. Vogliamo risolvere il problema di *dimensionamento delle barre* (ovvero determinarne la sezione -equivalentemente il peso- essendo assegnata la lunghezza), affinché il traliccio sia in grado di sopportare il carico (supposto naturalmente statico) in Fig. 1.1. La struttura rappresenta un ponte in cui sono presenti 5 nodi e 7 barre. È del tutto intuitivo che una qualsiasi struttura soggetta all'azione di forze esterne ($f = -100$ in figura) presenta delle flessioni (oppure si rompe), quindi la posizione dei nodi del traliccio, per effetto delle forze esterne, può cambiare. In base a questo, i nodi vengono suddivisi in *nodi fissi* (nodi 4 e 5 che non possono modificare la propria posizione in quanto ancorati al suolo) e *nodi liberi* (nodi 1, 2 e 3 i quali possono modificare la propria posizione). La variazione (vettore) di posizione di ciascun nodo libero prende il nome di *displacement* di quel nodo. Come evidente dalla figura, anche le lunghezze delle barre (supposte in numero pari a $M = 7$) sono assegnate, quindi la descrizione della geometria del traliccio (ponte) è completa.

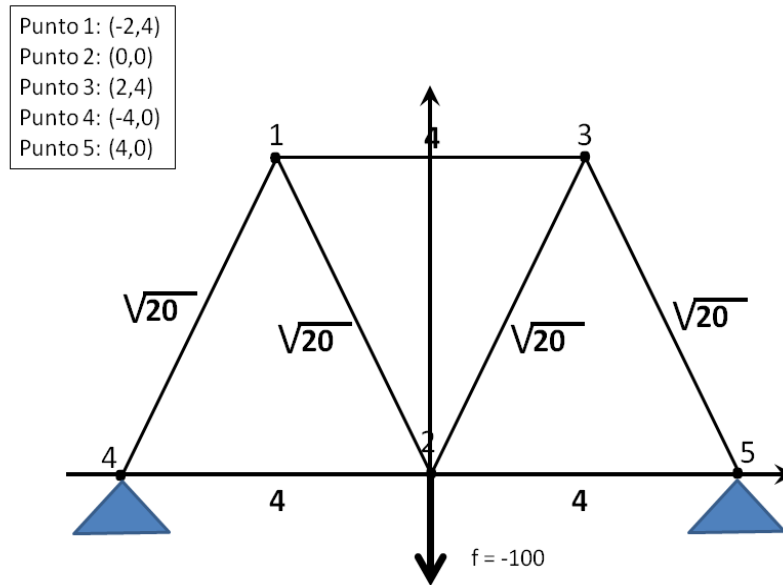


Figure 1.1: Progetto di un traliccio (ponte).

Ora, per ogni barra di estremi A e B si identifica una vettore β_{AB} dato da

$$\beta_{AB} = \sqrt{\kappa} \frac{B - A}{\|B - A\|^2} \in \mathbb{R}^2, \quad (1.2)$$

in cui κ è il cosiddetto *modulo di Young*, caratteristico del materiale usato per la barra. Si osservi che β_{AB} dipende dalla posizione e dall'ordine dei nodi.

Considerato l'insieme dei nodi liberi Nl , introdurremo il vettore dei displacement

$$dv \in \mathbb{R}^{2Nl}$$

di dimensione $2Nl$, il quale ha le prime 2 componenti pari al vettore di displacement relativo al primo nodo libero, le componenti 3 e 4 pari al vettore di displacement relativo al secondo nodo libero, etc.

Viene anche assegnato il vettore delle forze esterne (agenti solo sui nodi liberi in quanto la loro azione sui nodi fissi è ininfluente a meno di possibili rotture strutturali), dato da

$$f_e \in \mathbb{R}^{2Nl},$$

le cui componenti sono assegnate similmente al vettore dv . Infine, in maniera analoga, per ogni barra i di estremi A_i e B_i viene introdotto anche il vettore $b_i \in \mathbb{R}^{2Nl}$ a $2Nl$ componenti. In particolare, ciascuna coppia di componenti di b_i , relative al nodo libero ν (con $\nu = 1, \dots, Nl$), viene definita mediante la

$$b_i[\nu] = \begin{cases} \beta_{A_i B_i} & \text{se } \nu \text{ coincide con } B_i; \\ -\beta_{A_i B_i} & \text{se } \nu \text{ coincide con } A_i; \\ 0 & \text{altrimenti.} \end{cases} \quad (1.3)$$

Per completare la descrizione del problema è necessario introdurre il vettore

$$t \in \mathbb{R}^M$$

dei *volumi* delle barre. Esso costituisce l'elemento da dimensionare in modo tale che la struttura risulti resistente alla sollecitazione delle forze ma non sia eccessivamente sovradimensionata (avendo lunghezza assegnata, un maggior volume indica una maggiore sezione, ovvero un costo aggiuntivo non necessario).

Per concludere si introduce la matrice di *stiffness* $S(t)$ del traliccio, definita mediante la

$$S(t) = \sum_{i=1}^M t_i b_i b_i^T,$$

la quale, essendo per definizione $t_i \geq 0$, $i = 1, \dots, M$, risulterà semidefinita positiva. La matrice di stiffness $S(t)$, il vettore dei displacement dv ed il vettore delle forze esterne f_e sono legate dalla relazione

$$S(t)dv = f_e. \quad (1.4)$$

Inoltre viene introdotta una grandezza *Com* (*compliance*) che rappresenta l'energia potenziale complessiva immagazzinata nel traliccio all'equilibrio. Tale grandezza risulta pari a

$$Com = \frac{1}{2} f_e^T dv.$$

Una possibile formulazione non lineare del problema di progettare il precedente traliccio, dimensionando i volumi delle barre e calcolando il displacement di ogni nodo libero, è data di seguito.

- Definizione Variabili

t_i	=	volume della barra i -sima ($i = 1, \dots, M$)
dv	=	vettore (a $2Nl$ componenti) del displacement. Le prime due componenti costituiscono il vettore di displacement relativo al primo nodo libero, le componenti 3 e 4 costituiscono il vettore di displacement relativo al secondo nodo libero, etc.
$\beta_{A_i B_i}$	=	vettore (a 2 componenti) della tensione relativa alla barra i -sima ($i = 1, \dots, M$)
b_i	=	vettore di supporto (a $2Nl$ componenti) definito nella (1.3) relativo alla barra i -sima ($i = 1, \dots, M$).

- Identificazione Vincoli

Avremo innanzitutto i vincoli di non negatività sulle componenti del vettore t , quindi sarà

$$t_i \geq 0, \quad i = 1, \dots, M.$$

Poi saranno presenti vincoli (1.3) che legano le variabili $\beta_{A_i B_i}$ e le variabili b_i .

Infine dovrà essere presente un vincolo che esprima la relazione (1.4), dato da

$$\left(\sum_{i=1}^M t_i b_i b_i^T \right) dv = f_e.$$

- Identificazione Funzione Obiettivo

$$\min \text{ Com} = \frac{1}{2} f_e^T dv.$$

OSSERVAZIONE: Si noti che *fissato* il vettore t , se il sistema lineare (1.4) ammette le due soluzioni distinte dv_1 e dv_2 , ad esse corrisponde lo stesso valore della funzione obiettivo **Com**. Infatti, dalla simmetria di $S(t)$ sarà

$$\frac{1}{2} f_e^T dv_1 = \frac{1}{2} (dv_1)^T S(t) dv_1 = \frac{1}{2} (dv_1)^T S(t) dv_2 = \frac{1}{2} (dv_2)^T S(t) dv_1 = \frac{1}{2} f_e^T dv_2.$$

Inoltre si osservi che se il sistema (1.4) non ammette soluzione allora vuol dire che il traliccio non è in grado di sopportare i carichi esterni agenti sui nodi liberi. Infine si sottolinea che essendo $S(t)$ semidefinita positiva, dalla precedente relazione segue che la funzione obiettivo è *convessa*.

Sia ora data la seguente realizzazione del modello presentato, mediante il linguaggio di programmazione matematica AMPL. Si verifichi la correttezza del modello e si forniscano le motivazioni in base alle quali il solutore MINOS 5.5 fornisce la seguente soluzione:

MINOS 5.5: unbounded (or badly scaled) problem.

0 iterations

Nonlin evals: constra = 45, Jac = 44.

Traliccio.mod

```
#####
####  CALCOLO TRALICCIO  ####
#####
```

```
####  dichiarazione parametri del problema  ####
param Nl;                                # numero nodi liberi del traliccio
param Nf;                                # numero nodi fissi del traliccio
param M;                                  # numero barre del traliccio
```

```
####  dichiarazione set del problema  ####
set NODI_TOT:={1..Nl+Nf};                # insieme di tutti i nodi
set BARRE:={1..M};                       # insieme delle barre
```

```

#### dichiarazione parametri del problema ####
param NODI{1..2,NODI_TOT};          # matrice delle posizioni
                                     # dei nodi del traliccio
param YY;                           # modulo di Young relativo
                                     # al materiale delle barre
param w > 0;                         # peso max complessivo consentito
param f{1..2*Nl};                   # vettore dei carichi agenti sui
                                     # nodi liberi del traliccio

#### dichiarazione di variabili ####
var t{i in BARRE} >=0;              # volume t[i] della barra i-sima
var dv{1..2*Nl};                    # vettore dei displacement (2 componenti
                                     # per ogni nodo libero)
var beta{1..2,i in BARRE};          # variabili di supporto
var b{k in 1..2*Nl,i in BARRE};     # variabili di supporto

#### dichiarazione funzione obiettivo ####
minimize compliance: .5 * sum{k in 1..2*Nl} f[k] * dv[k]; # minimizziamo l'energia
                                                         # potenziale complessiva
                                                         # immagazzinata nel
                                                         # traliccio

#### introduzione vincoli ####
s.t.  min_volume: sum{i in BARRE} t[i] <= w;

s.t.  beta_1{k in 1..2}: beta[k,1]= sqrt(YY)*(NODI[k,1]- NODI[k,4])/20;
s.t.  beta_2{k in 1..2}: beta[k,2]= sqrt(YY)*(NODI[k,2]- NODI[k,4])/16;
s.t.  beta_3{k in 1..2}: beta[k,3]= sqrt(YY)*(NODI[k,2]- NODI[k,1])/20;
s.t.  beta_4{k in 1..2}: beta[k,4]= sqrt(YY)*(NODI[k,3]- NODI[k,1])/16;
s.t.  beta_5{k in 1..2}: beta[k,5]= sqrt(YY)*(NODI[k,3]- NODI[k,2])/20;
s.t.  beta_6{k in 1..2}: beta[k,6]= sqrt(YY)*(NODI[k,5]- NODI[k,2])/16;
s.t.  beta_7{k in 1..2}: beta[k,7]= sqrt(YY)*(NODI[k,5]- NODI[k,3])/20;

s.t.  b_1_free{k in 1..2*Nl: k <= 2}: b[k,1]=beta[k,1];
s.t.  b_1_altr{k in 1..2*Nl: k > 2}: b[k,1]=0;

s.t.  b_2_altr1{k in 1..2*Nl: k <= 2}: b[k,2]=0;
s.t.  b_2_free{k in 1..2*Nl: k >= 3 and k <= 4}: b[k,2]=beta[k-2,2];
s.t.  b_2_altr2{k in 1..2*Nl: k > 4}: b[k,2]=0;

s.t.  b_3_free1{k in 1..2*Nl: k <= 2}: b[k,3]= -beta[k,3];
s.t.  b_3_free2{k in 1..2*Nl: k >= 3 and k <= 4}: b[k,3]= beta[k-2,3];
s.t.  b_3_altr{k in 1..2*Nl: k > 4}: b[k,3]=0;

s.t.  b_4_free1{k in 1..2*Nl: k <= 2}: b[k,4]=-beta[k,4];
s.t.  b_4_altr{k in 1..2*Nl: k >= 3 and k <= 4}: b[k,4]= 0;
s.t.  b_4_free2{k in 1..2*Nl: k > 4}: b[k,4]= beta[k-4,4];

s.t.  b_5_altr{k in 1..2*Nl: k <= 2}: b[k,5]=0;
s.t.  b_5_free1{k in 1..2*Nl: k >= 3 and k <= 4}: b[k,5]= -beta[k-2,5];

```

```

s.t.   b_5_free2{k in 1..2*N1: k > 4}: b[k,5]= beta[k-4,5];

s.t.   b_6_altri{k in 1..2*N1: k <= 2}: b[k,6]=0;
s.t.   b_6_free{k in 1..2*N1: k >= 3 and k <= 4}: b[k,6]= -beta[k-2,6];
s.t.   b_6_altr2{k in 1..2*N1: k > 4}: b[k,6]= 0;

s.t.   b_7_altr{k in 1..2*N1: k <= 4}: b[k,7]= 0;
s.t.   b_7_free{k in 1..2*N1: k > 4}: b[k,7]= -beta[k-4,7];

s.t.   equi_forze{h in 1..2*N1}: sum{i in BARRE} t[i]*b[h,i]*
                                   (sum{k in 1..2*N1} b[k,i]*dv[k]) = f[h];
-----

```

Traliccio.dat

```

#####
####  CALCOLO TRALICCIO  ####
#####

####  assegnazione  parametri  ####
param N1:= 3;
param Nf:= 2;
param M:= 7;
param f:=
  1 0
  2 0
  3 0
  4 -100
  5 0
  6 0 ;

param NODI: 1   2   3   4   5 :=
  1  -2   0   2  -4   4
  2   4   0   4   0   0 ;

param YY:= 1;

param w:= 10000;
-----

```

Si provi ora a sostituire l'istruzione

```
var t{i in BARRE} >=0;
```

con la nuova istruzione

```
var t{i in BARRE}:=0.5 >=0;
```

Si otterrà la soluzione corretta (lo studente è invitato a darne un'interpretazione; inoltre si osservi la simmetria del vettore di displacement per i 3 nodi liberi).

```

MINOS 5.5: optimal solution found.
176 iterations, objective 72
Nonlin evals: constrs = 442, Jac = 441.

```

```
t [*] :=  
1 2083.33  
2 0  
3 2083.33  
4 1666.67  
5 2083.33  
6 0  
7 2083.33  
;  
  
dv [*] :=  
1 0.24  
2 -0.72  
3 -1.63692e-10  
4 -1.44  
5 -0.24  
6 -0.72  
;
```

1.4 Esercitazione 4: Esempio di Programmazione Matematica nonlineare

1.4.1 Calcolo traiettoria Shuttle

Descriviamo ora un secondo esempio di modellazione-soluzione mediante AMPL. Il problema è sostanzialmente un classico problema di meccanica newtoniana. Si vuole far decollare uno shuttle dalla Terra facendolo atterrare sulla Luna; per evidenti ragioni di simmetria sferica si intende studiare il problema in 2 sole dimensioni. Sono noti i dati relativi alla massa, raggio, posizione dei centri di Terra e Luna rispetto al sistema di riferimento scelto (origine nel centro della Terra). Si intende approssimare la traiettoria $\mathbf{x}(t) \in \mathbb{R}^2$ (funzione del tempo) dello shuttle supponendo che esso sia soggetto alle sole forze di attrazione dei due corpi celesti e possa con i propri motori generare una spinta controllabile (si veda Fig. 1.2).

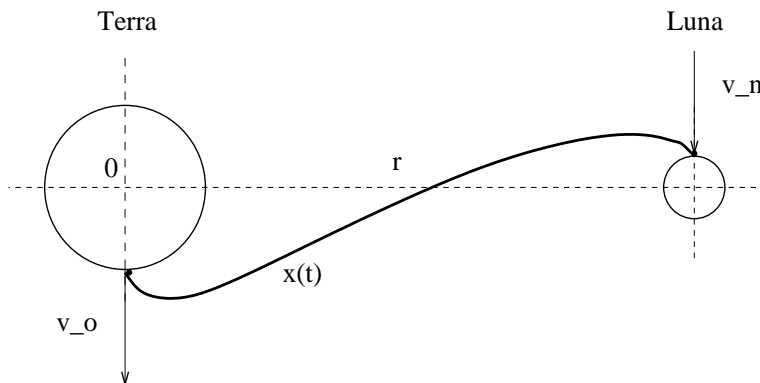


Figure 1.2: Traiettoria Shuttle.

La traiettoria è un vettore a due dimensioni, inoltre la velocità $\mathbf{v}(t) \in \mathbb{R}^2$ e l'accelerazione $\mathbf{a}(t) \in \mathbb{R}^2$ cui è soggetto lo shuttle, sono legate dalle relazioni:

$$\mathbf{v}(t) = \frac{d\mathbf{x}}{dt}(t) \quad \mathbf{a}(t) = \frac{d\mathbf{v}}{dt}(t)$$

Detta m_s la massa dello shuttle, quest'ultimo sarà soggetto ad un'accelerazione data dalla seconda legge del moto:

$$\mathbf{F} = m_s \cdot \mathbf{a}$$

dove \mathbf{F} è la forza totale alla quale è sottoposto lo shuttle. Tale forza è la risultante delle seguenti 3 forze:

- l'attrazione gravitazionale della Terra;
- l'attrazione gravitazionale della Luna;
- la spinta dei motori dello shuttle.

Le prime due forze sono regolate dalla *legge della gravitazione universale*, che per esempio nel caso della coppia di masse Terra-Luna, è espressa dalla seguente relazione (si ricordi che la seguente è un'equazione scalare e che pertanto andrà proiettata sugli assi del sistema di riferimento in quanto \mathbf{F} è un vettore):

$$F = G \cdot \frac{m_T \cdot m_L}{r^2},$$

dove G è una costante universale, m_T ed m_L sono rispettivamente le masse della Terra e della Luna, infine r è la lunghezza del raggio congiungente i centri della Terra e della Luna.

Si vuole calcolare la traiettoria minimizzando un indice di costo *non lineare* dato dalla somma (opportunamente pesata) del tempo totale necessario allo spostamento dello shuttle e dell'energia spesa per la spinta. La relazione che sintetizza questo indice è la seguente:

$$\text{cost} \cdot T + \int_0^T \|\Theta(t)\|^2 dt,$$

dove $\Theta(t)$ è la spinta (forza) generata dai motori dello shuttle. Sulla base di quanto descritto, in ogni punto della traiettoria lo shuttle sarà soggetto ad un'accelerazione pari a (la seguente è una relazione vettoriale che come al solito va proiettata sugli assi del sistema di riferimento):

$$\mathbf{a}(t) = G \cdot \frac{m_T}{\|\mathbf{x}_T - \mathbf{x}(t)\|^3} (\mathbf{x}_T - \mathbf{x}(t)) + G \cdot \frac{m_L}{\|\mathbf{x}_L - \mathbf{x}(t)\|^3} (\mathbf{x}_L - \mathbf{x}(t)) + \frac{\Theta(t)}{m_s}$$

in cui \mathbf{x}_T ed \mathbf{x}_L sono due vettori che rappresentano le coordinate del centro della Terra e della Luna. Infine sono note le posizioni e le velocità di partenza (dalla Terra) ed arrivo (sulla Luna) dello shuttle. A questo punto un possibile modello (in AMPL) del problema, opportunamente discretizzato (si è divisa la traiettoria in 25 segmenti, inoltre si sono scelti valori delle costanti che approssimativamente rappresentino il problema in scala), è il seguente:

Shuttle.mod

```
-----
# Calcolo traiettoria di uno Shuttle - Shuttle.mod -

####  DEFINIZIONE/ASSEGNAZIONE PARAMETRI PRELIMINARI  ####
param n:=25;

####  DEFINIZIONE/ASSEGNAZIONE INSIEMI  ####
set DIM:= 1..2;
set Nx:= 0..n;
set Nv:= 0.5..n-0.5 by 1;
set Na:= 1..n-1;

####  DEFINIZIONE/ASSEGNAZIONE PARAMETRI  ####
param G;
param costante;
param m_earth;
param r_earth;
param x_earth{DIM};
param m_moon;
param r_moon;
param x_moon{DIM};

param x_0{DIM};
param x_n{DIM};
param v_0{DIM};
param v_n{DIM};

####  DEFINIZIONE/ASSEGNAZIONE VARIABILI  ####
var T >=1 := 4;

var x {i in DIM,j in Nx}:= (1 - j/n)*x_0[i] + (j/n)*x_n[i];

var v {i in DIM,j in Nv} = (x[i,j+0.5] - x[i,j-0.5])/(T/n);  # velocita' media nel
                                                                # sottointervallo j-mo
var a {i in DIM,j in Na} = (v[i,j+0.5] - v[i,j-0.5])/(T/n);  # accelerazione media nel
                                                                # sottointervallo j-mo
var theta{DIM,Na};

####  FUNZIONE OBIETTIVO E VINCOLI  ####
    minimize total_cost: costante * T + (sum{i in DIM,j in Na} theta[i,j]^2) * (T/n);
```

```

s.t. pos_iniz{i in DIM}: x[i,0] = x_0[i];          # fissata la posizione di partenza
s.t. pos_fina{i in DIM}: x[i,n] = x_n[i];          # fissata la posizione di arrivo
s.t. vel_iniz{i in DIM}: v[i,0.5] = v_0[i];        # fissata la velocita iniziale
s.t. vel_fina{i in DIM}: v[i,n-0.5] = v_n[i];      # fissata la velocita finale

s.t. spostamento{j in Nx: j<> n}: x[1,j] <= x[1,j+1];      # vincoli inutili !!
s.t. bilanc_forze{i in DIM, j in Na}:
    a[i,j] = G * m_earth * (x_earth[i] - x[i,j]) /
              (sum{ii in DIM}(x[ii,j]-x_earth[ii])^2)^(3/2) +
              - G * m_moon * (x[i,j] - x_moon[i]) /
              (sum{ii in DIM}(x[ii,j]-x_moon[ii])^2)^(3/2) +
              + theta[i,j] / 0.01;

s.t. acceleraz_limitata{i in DIM, j in Na}: -20 <= a[i,j] <= 20;

```

E per quanto riguarda il file di dati, è stato utilizzato il seguente:

Shuttle.dat

Calcolo traiettoria di uno Shuttle - Shuttle.dat -

```

param costante:= 2;
param G = 1;
param m_earth:= 10;
param r_earth:= 2;
param      x_earth:=
    1      0
    2      0      ;
param m_moon:= 1;
param r_moon:= 1;
param:      x_moon:=
    1      50
    2      0      ;
param      x_0 :=
    1      0
    2     -2      ;
param      x_n :=
    1      50
    2      1      ;
param      v_0 :=
    1      0
    2     -0.5    ;
param      v_n :=
    1      0
    2     -0.5    ;

```

Per la cui soluzione si ha utilizzando il solutore nonlineare MINOS (si osservi che nel modello sono presenti anche funzioni razionali)

```

OBJECTIVES:
total_cost = 6.7172

```

VARIABLES:
T = 3.29404

```
x [*,*]
:=      1      2
-----
0      0      -2
1    1.96367e-15 -2.06588
2    0.347222   -2.08846
3    1.04167    -2.05942
4    2.08333    -1.97981
5    3.47222    -1.8619
6    5.20833    -1.71641
7    7.29167    -1.54967
8    9.72222    -1.36605
9   12.5        -1.1692
10   15.625     -0.962608
11   19.0972    -0.749548
12   22.9167    -0.533442
13   27.0833    -0.317612
14   30.9028    -0.105333
15   34.375     0.100011
16   37.5       0.295149
17   40.2778    0.476764
18   42.7083    0.641522
19   44.7917    0.786061
20   46.5278    0.906933
21   47.9167    1.00063
22   48.9583    1.06292
23   49.6528    1.087
24   50         1.06588
25   50         1
```

Bibliography

- [BN01] A. Ben-Tal, A.Nemirovski (2001) *“Lectures on Modern Convex Optimization - analysis, algorithms, and engineering applications”*, MPS-SIAM Series on Optimization.
- [FGK89] R.Fourer, D.M.Gay, B.W.Kernighan (1989) *“AMPL: A mathematical programming language”*, AT&T Bell Laboratories, Murray Hill, NJ.
- [PV91] P.M.Pardalos, S.A.Vavasis (1991) *“Quadratic Programming with One Negative Eigenvalue is NP-Hard”*, Journal of Global Optimization 1, pp. 15-22.
- [R70] R.T.Rockafellar (1970) *“Convex Analysis”*, Princeton University Press, Princeton, N.J.
- [V91] S.A.Vavasis (1991) *“Nonlinear Optimization - complexity issues”*, Oxford University Press, N.Y.