

Analisi numerica con GNU Octave

di Francesco Paparella e Raffaele Vitolo

1. Che cos'è GNU Octave e a che cosa serve.

Il programma GNU Octave è un programma di calcolo numerico. Inizialmente, GNU Octave nacque presso l'Università del Texas come programma di calcolo per l'ingegneria chimica; il nome 'Octave' è il nome del docente di uno dei corsi presso i quali John W. Eaton, il principale autore del programma, ne iniziò lo sviluppo. Poi fu esteso a coprire ambiti matematici più generali, come l'algebra lineare e le equazioni differenziali.

Attualmente GNU Octave è un programma mantenuto e sviluppato dal progetto GNU, rilasciato con licenza GPL (è, quindi, software libero) ed è incluso nelle principali distribuzioni GNU/Linux, ma riceve contributi da un gran numero di sviluppatori indipendenti, spesso ricercatori e docenti universitari.

Tra gli ambiti di utilizzo di GNU Octave (Octave per semplicità, d'ora in poi) si trovano: l'aritmetica di base, l'algebra lineare, il calcolo matriciale, le equazioni algebriche e differenziali, la statistica, la ricerca operativa, la matematica finanziaria, la teoria dei controlli, l'analisi delle immagini, l'analisi dei segnali, l'analisi del suono, e molti altri.

L'importanza del programma è notevole da vari punti di vista. Ne elenchiamo i principali.

- E' uno strumento affidabile perché utilizza, per il calcolo numerico, le librerie del sito <http://netlib.org>. Esse implementano i principali algoritmi per l'analisi numerica sviluppati dalla comunità mondiale di esperti in calcolo scientifico, e sono disponibili a tutti secondo l'etica scientifica. Tali librerie, sviluppate principalmente in Fortran e C, sono soggette ad un continuo processo di ottimizzazione ed aggiornamento, e costituiscono lo standard di riferimento del calcolo numerico.
- E' un ottimo sussidio didattico: a differenza dei programmi commerciali, permette di illustrare agli studenti anche il funzionamento del programma stesso. Si immagini l'utilità in corsi di informatica o di matematica applicata.
- E' eticamente la migliore scelta dal punto di vista scientifico: *nessuno* dovrebbe potersi fidare della “scatola nera” costituita da un programma proprietario in campo scientifico. La riproducibilità dei risultati scientifici *non può* essere affidata al numero di versione di un programma proprietario od alla speranza che gli sviluppatori di un tale programma non abbiano commesso errori.
- E' parzialmente compatibile con il linguaggio di Matlab. Quest'ultimo programma è diffusamente utilizzato in tutto il mondo per scopi didattici e di ricerca. Ma poiché è proprietario, è impossibile studiare i dettagli dell'implementazione, e ciò diseduca gli allievi a chiedersi che cosa ci sia *sotto il cofano* degli strumenti che utilizzano, e getta un'ombra sulla verificabilità dei risultati scientifici con esso ottenuti. La transizione da Matlab a Octave è particolarmente agevole.

2. Introduzione al linguaggio Octave.

Octave è un programma a linea di comando. Questo, nella nostra esperienza, spaventa gli utenti, inizialmente. Ma si tratta di qualcosa di indispensabile: secondo noi non è possibile concepire un programma di calcolo numerico che funzioni in modo efficace con un'interfaccia grafica. Il controllo totale sulle operazioni da svolgere è attuabile solo tramite la linea di comando. E d'altra parte, al di là delle dichiarazioni pubblicitarie, anche gli analoghi proprietari di Octave sono solo coadiuvati da strumenti grafici, ma hanno bisogno di una linea di comando per svolgere il grosso delle operazioni.

Al livello più basso, Octave può essere utilizzato come una calcolatrice scientifica. Sono ovviamente disponibili le operazioni aritmetiche, l'elevamento a potenza (indicato con il carattere ^), le funzioni trigonometriche, esponenziale e le loro inverse.

```
octave:1> (1+1)*3
ans = 6
octave:2> log(1)+cos(0)
ans = 1
```

Alcune costanti sono predefinite: il pi greco (π), la base dei logaritmi naturali (e), l'unità immaginaria (i). I calcoli avvengono sempre in doppia precisione, ed i dettagli dell'implementazione sono dipendenti dall'architettura hardware della macchina.

Il passo successivo consiste nell'introdurre l'uso di variabili:

```
octave:3> a=5
a = 5
octave:4> b=exp(a)
b = 148.41
octave:5> log(b)-a
ans = 0
```

Per evitare che il risultato delle operazioni di assegnazione sia riportato sullo schermo, si termina l'istruzione con un punto e virgola:

```
octave:6> a=5;
octave:7> b=exp(a);
octave:8> log(b)-a
ans = 0
```

Il contenuto di una variabile può sempre essere ispezionato scrivendo il nome della variabile, il cui valore verrà stampato sullo schermo:

```
octave:9> a
a = 5
```

Come è consueto per un ambiente interpretato, le variabili non devono essere dichiarate, ed il loro tipo è determinato al momento dell'assegnazione dal tipo dell'espressione a destra del segno =. Octave mette a disposizione dell'utente diversi tipi di dati. Oltre al tipo numerico semplice che abbiamo visto finora, esiste anche un tipo logico (`true`, `false`), un tipo stringa e delle strutture dati analoghe alle `struct` del linguaggio C. Ma il tipo senza dubbio più comune e potente è la matrice, cioè una tabella di valori numerici organizzata in righe e colonne. Il modo più semplice per creare una matrice è quello di elencare fra parentesi quadre i valori numerici che la compongono:

```
octave:10> M=[1, 2, 3; 4, 5, 6; 7, 8, 9]
M =
```

```
1 2 3
```

```
4 5 6
7 8 9
```

Come è facile intuire da quest'esempio, la virgola permette di spostarsi da una colonna alla successiva lungo una stessa riga, ed il punto e virgola separa le varie righe. Gli elementi di una matrice possono essere indirizzati singolarmente, con l'ordine (righe, colonne):

```
octave:11> M(2,1)
ans = 4
```

oppure come sottomatrici, grazie a speciali espressioni *range*:

```
octave:12> M(1:2,1:3)
ans =
```

```
1 2 3
4 5 6
```

In questo caso abbiamo chiesto ad Octave di estrarre dalla matrice M le righe dalla 1 alla 2, e le colonne dalla 1 alla 3 (cioè tutte). Le espressioni *range* sono utilissime anche per definire dei “vettori riga”, ovvero delle matrici costituite da una sola riga e colonne multiple, le cui componenti sono progressive:

```
octave:13> a=1:5
a =
```

```
1 2 3 4 5
```

L'inizio, il termine ed il passo in una espressione *range* possono essere specificati liberamente:

```
octave:14> a=-6:3:10
a =
```

```
-6 -3 0 3 6 9
```

La progressione termina senza mai eccedere il limite superiore specificato (10, in questo esempio).

Le espressioni *range* ci portano naturalmente a considerare un ulteriore livello di complessità nell'uso di Octave, quello del controllo del flusso delle istruzioni. Esse, infatti, sono essenziali nello specificare i cicli *for*. Un ciclo *for* si ripete per tante volte quanti sono gli elementi dell'espressione *range* che lo definisce, ed una variabile assume in sequenza i valori di tali elementi. Un esempio tratto dal manuale di Octave chiarirà questo concetto a chi non avesse alcuna familiarità con altri linguaggi di programmazione dotati di strutture simili. Innanzitutto creiamo un vettore riga con 10 elementi tutti pari 1. A questo scopo esiste l'apposita funzione `ones(n,m)`, che crea una matrice di n righe e m colonne con elementi pari a 1:

```
octave:15> fib=ones(1,10);
```

Il nostro scopo è quello di riempire il vettore `fib` con i primi dieci numeri di Fibonacci, che sono definiti in questo modo: il primo numero di Fibonacci è 1, il secondo numero di Fibonacci è anch'esso 1, ogni altro numero di Fibonacci è pari alla somma dei due precedenti. Il ciclo che risolve il problema è il seguente:

```
octave:16> for i=3:10
> fib(i)=fib(i-1)+fib(i-2);
> end
```

La parola chiave `end` chiude il ciclo, mentre la variabile `i` (usata per indicizzare il vettore `fib`) assume in sequenza i valori da 3 a 10. Il risultato è immagazzinato nel vettore `fib`:

```
octave:49> fib
fib =
```

```
1 1 2 3 5 8 13 21 34 55
```

Oltre ai cicli *for*, Octave mette a disposizione le più comuni strutture di controllo del

flusso di istruzioni: i test logici `if-else` e `switch`, ed i cicli `while` e `do-until`.

Una sequenza di comandi, di chiamate a funzioni e di istruzioni di controllo di flusso di Octave può essere raccolta in un singolo *file* di testo con estensione `.m`. Da quel momento in poi, è sufficiente digitare il nome del *file* (senza estensione) all'interno dell'ambiente Octave per eseguire la sequenza contenuta nel *file*. In questo modo è possibile costruire semplici programmi *script* per automatizzare operazioni ripetitive. Il comando `path` mostra l'elenco delle cartelle in cui Octave cerca i *file script*.

Infine, il linguaggio Octave mette a disposizione la possibilità di definire funzioni utente, che si comportano in tutto e per tutto come le funzioni matematiche predefinite. La sintassi è semplice e sufficientemente efficace, e la illustriamo con un esempio. Una funzione che calcola i numeri di Fibonacci fino all'*N*-esimo è la seguente:

```
## -- Numeri di Fibonacci: f = fibonacci(N)
##     Restituisce nel vettore f i primi N numeri
##     di Fibonacci

function f = fibonacci(N)
    f=ones(1,N);
    for i=3:N
        f(i)=f(i-1)+f(i-2);
    end
endfunction
```

Per mantenere la compatibilità con Matlab, vi è la curiosa limitazione di dover chiamare il *file* in cui si salva la funzione col nome della funzione stessa. Nel nostro caso il nome del *file* è, obbligatoriamente, `fibonacci.m`. Per il medesimo motivo, ogni funzione deve essere definita in un *file* a parte (è possibile definire più funzioni all'interno di uno *script*, ma esse saranno poi accessibili solo all'interno dello *script*). I commenti prima della definizione della funzione (le linee precedute da `##`) integrano il manuale *on-line* di Octave: digitando `help fibonacci` essi saranno mostrati sullo schermo. Infine, le funzioni di Octave possono essere ricorsive (cioè possono chiamare se stesse), e possono avere un numero variabile di argomenti sia in ingresso che in uscita (per i dettagli rimandiamo al manuale).

3. A proposito di grafica.

La grafica è stata per molto tempo un punto debole di Octave, che non dispone di un sottosistema grafico proprio, ma deve fare ricorso ai servizi di un programma esterno. La scelta cadde su Gnuplot, che ancora oggi è il motore della grafica di Octave, anche se esistono collezioni di funzioni grafiche che si appoggiano ad altri programmi. A mano a mano che le capacità di Gnuplot sono progredite, le possibilità di Octave in campo grafico sono aumentate, ed oggi si ha un prodotto capace di produrre una vastissima tipologia di grafici di eccellente qualità. Un esempio di visualizzazione di dati scientifici è riportato nel box [un esempio di grafica]. Naturalmente, per una panoramica completa delle capacità di Gnuplot, rimandiamo al relativo articolo in questo stesso numero ***.

La situazione, tuttavia, non è ancora pienamente soddisfacente. Il modo più potente per visualizzare dei dati è quello di utilizzare una serie di comandi (`gplot`, `gset`, `graw`, etc.) che replicano i comandi nativi di Gnuplot, con il triplo svantaggio di obbligare l'utente di Octave ad imparare Gnuplot, di creare una inelegante isola di "sintassi anomala" nel linguaggio Octave e di perdere la compatibilità con Matlab. Una serie di comandi a più alto livello (`plot`, `mesh`, `contour`) tenta di replicare gli equivalenti comandi di Matlab. In particolare, grazie agli sforzi della comunità raccoltasi intorno a Octave-Forge, i comandi grafici di alto livello sono aumentati

in numero ed in potenza (fra essi l'utilissimo `print`, che permette di salvare il grafico corrente scegliendo fra un vastissimo numero di formati di file, senza doversi interfacciare a basso livello con Gnuplot).

Tuttavia è ancora difficile, a nostro parere, avere un controllo della grafica sufficientemente fine coi soli comandi di alto livello. Un'altra difficoltà risiede nella modalità, un po' rozza, di comunicazione fra i due programmi. Infatti Octave salva i dati che devono essere visualizzati da Gnuplot in dei file di testo temporanei (nella cartella `/tmp`). Per grafici singoli questa non è una limitazione, ma se si desidera fare delle animazioni, allora è necessario un calcolatore con un sottosistema di dischi molto veloce!

4 Dieci minuti di introduzione al calcolo matriciale.

Il principale punto di forza di Octave è il calcolo matriciale. Poiché è inconsueto che linguaggi di uso generale abbiano le matrici come tipo di dati predefinito, ed assumiamo che il lettore non abbia familiarità con linguaggi (o librerie) per il calcolo scientifico, ci sembra utile spendere quale parola per illustrare le più semplici manipolazioni matriciali che Octave ci permette di fare.

A questo fine illustriamo come risolvere un sistema lineare di equazioni algebriche (v. box [sistemi lineari]). La ricerca delle soluzioni di un sistema lineare si può ricondurre ad alcune manipolazioni sulle matrici che lo definiscono. Problemi di questa natura si incontrano comunemente in ambito scientifico ed ingegneristico, dove è comune arrivare a matrici con migliaia di righe e di colonne. In questo esempio ci limiteremo ad usare matrici 2×2 .

Le matrici possono essere sommate (e sottratte). La somma può essere effettuata solo tra matrici di uguali dimensioni (cioè con uguale numero di righe e di colonne). Il risultato è una matrice che ha per elementi le somme degli elementi corrispondenti:

```
octave:1> [1, 2; 3, 4] + [2, -1; 4, 2]
ans =
```

```
3  1
7  6
```

Una matrice può essere moltiplicata per un numero; il risultato è la matrice i cui elementi sono gli elementi della matrice di partenza moltiplicati per il numero dato.

L'operazione meno 'consueta' che si può effettuare tra due matrici è senza dubbio la moltiplicazione. Questa operazione è anche detta prodotto 'righe per colonne', si veda il box corrispondente per la definizione. Il prodotto tra matrici è indicato in Octave con `*`. Ad esempio:

```
octave:2> [1, 2; 3, 4] * [2, -1; 4, 2]
ans =
```

```
10  3
22  5
```

Esistono delle matrici quadrate particolari, che indichiamo con I , i cui elementi sono tutti zero, salvo quelli che giacciono sulla diagonale principale, che sono tutti uno. Usando la definizione di prodotto fra matrici, è facile convincersi che il prodotto fra una matrice qualunque A e la matrice identità di dimensione opportuna (quella che ha tante colonne quante sono le righe di A) è uguale alla stessa matrice A . Ad esempio:

```
octave:3> [1, 2; 3, 4] * [1, 0; 0, 1]
ans =
```

```
1 2
3 4
```

Tali matrici sono dette matrici identità, e giocano lo stesso ruolo di 1 nella moltiplicazione tra numeri reali. Pertanto ci si può chiedere se, data una matrice quadrata A , esista una matrice, indicata con A^{-1} , tale che $A^{-1} \cdot A = I$. L'elemento A^{-1} si chiama matrice inversa di A . L'inverso di un numero reale x esiste a condizione che x sia diverso da zero. In tal caso l'inverso è semplice da calcolare: si tratta del numero $1/x$. Per le matrici quadrate la situazione è analoga: l'inverso di una matrice A esiste a condizione che una certa quantità, detta determinante della matrice A , sia diversa da zero. Per matrici 2x2 calcolare il determinante è facile, se $A=[a,b;c,d]$ allora il determinante è $ad-bc$. Qualcuno ricorderà di aver imparato alle scuole superiori un metodo pratico (detto regola di Sarrus) per calcolare il determinante di matrici 3x3. Per matrici di dimensioni superiori il calcolo si complica, ma Octave ci soccorre con la funzione `det()`:

```
octave:4> A = [1, 2; 3, 4];
octave:5> det(A)
ans = -2
```

Poiché il determinante della matrice $[1, 2; 3, 4]$ è diverso da zero, possiamo chiedere ad Octave quale sia il suo inverso, ed immagazzinarlo nella matrice A_i :

```
octave:6> Ai = inv(A)
Ai =
```

```
-2.00000  1.00000
 1.50000 -0.50000
```

ed infine verificare che il prodotto $A_i \cdot A$ sia proprio la matrice identità:

```
octave:7> Ai*A
ans =
```

```
1.00000  0.00000
0.00000  1.00000
```

Un sistema lineare di equazioni algebriche è esprimibile in forma matriciale come $A \cdot X = B$, dove X è una matrice che ha una sola colonna e contiene tutte le incognite, B è una matrice di una sola colonna contenente tutti i termini noti ed A è una matrice che contiene tutti i coefficienti delle equazioni. Se la matrice A è quadrata ed invertibile, ossia, come si è visto, $\det(A)$ non è nullo, allora si può moltiplicare entrambi i membri dell'equazione per A^{-1} , ottenendo:

$A^{-1} \cdot A \cdot X = A^{-1} \cdot B$; ma essendo $A^{-1} \cdot A = I$ ed $I \cdot X = X$ si ottiene la soluzione del sistema:

$X = A^{-1} \cdot B$. In Octave questo procedimento si traduce così:

```
octave:8> A = [3, 5, -1; 0, 2, 3; -1, 3, 2];
octave:9> B = [2; 0; 3];
octave:10> det(A)
ans = -32
octave:11> X = inv(A) * B
X =
```

```
-1.28125
 1.03125
-0.68750
```

Si può anche procedere in maniera più sintetica con la 'divisione matriciale' di Octave:

```
octave:12> X = A \ B;
```

Ovviamente il risultato è lo stesso, anche se l'algoritmo usato per la divisione matriciale è più rapido che non il passare dalla matrice inversa. Per maggiori dettagli su questo si consulti il manuale di Octave.

Per risolvere sistemi lineari di grandi dimensioni di solito si ricorre ad altri metodi, ognuno dei quali è ottimizzato per una certa classe di problemi. Le funzioni di Octave che sono più utilizzate a tale proposito sono ``lu'` e ``qr'`. In entrambi i casi queste funzioni scompongono la matrice dei coefficienti del sistema in un prodotto di due matrici triangolari, in modo che il sistema sia risolubile mediante la soluzione consecutiva dei due sistemi triangolari (la soluzione dei quali è un problema numericamente molto più semplice). Ma la matematica che sta alla base di queste scomposizioni non permette di approfondire in questa sede l'argomento, per il quale rimandiamo ai testi di analisi numerica in bibliografia.

5 Octave-Forge e la compatibilità con Matlab.

Nessuna discussione riguardo ad Octave sarebbe completa senza una menzione di Octave-Forge. Reperibile all'indirizzo <http://octave.sourceforge.net/index/index.html>, ed inclusa in molte distribuzioni, Octave-Forge è una raccolta in continua evoluzione di *scripts*, estensioni e pacchetti specialistici che aumentano le capacità di Octave.

Da una lato si completano le capacità di Octave in versione base: nel settore *plotting*, ad esempio, si trovano numerose funzioni che rendono più facile e più immediato ottenere rappresentazioni grafiche attraenti dei propri dati. Dall'altro lato vi sono pacchetti che ambiscono a dare ad Octave capacità completamente nuove: *symbolic algebra*, ad esempio, è un primo tentativo di implementare un sottosistema per il calcolo simbolico (gli studenti alle prese con il calcolo di derivate ed integrali ne saranno lieti).

Particolarmente interessanti sono le funzioni di tipo statistico (per esempio per la regressione nonlineare o l'analisi di varianza multicanale ANOVA). Il pacchetto di analisi delle serie temporali è particolarmente completo, includendo anche funzioni per il calcolo e la visualizzazione del bispettro. Gli appassionati di crittografia e compressioni dei dati accoglieranno come una manna le funzioni del pacchetto dedicato alle comunicazioni. Così come gli interessati all'analisi ed all'elaborazione delle immagini saranno deliziati dal pacchetto di *image processing*.

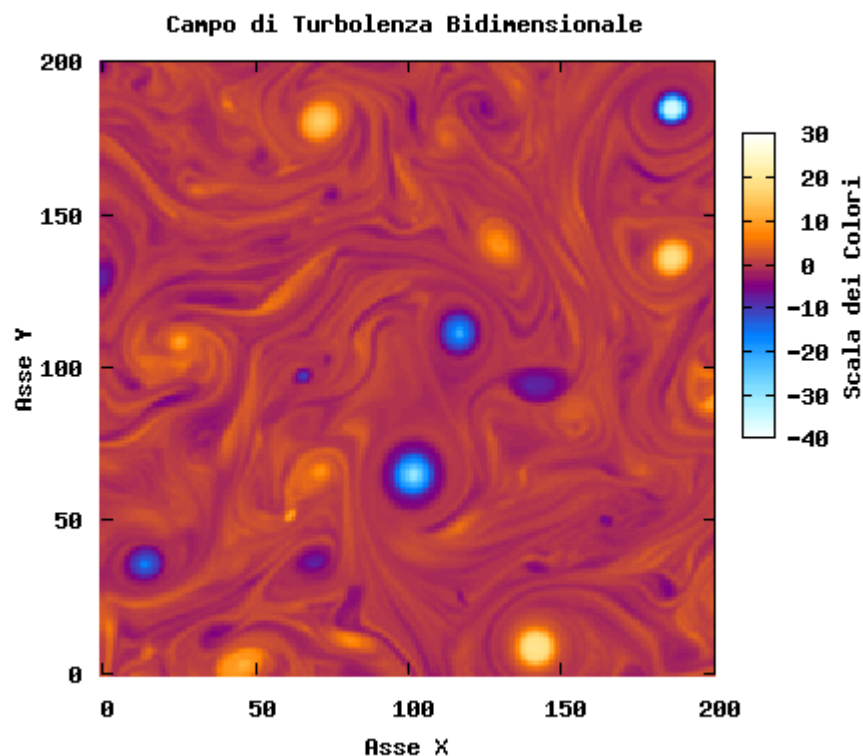
Fra le cose da segnalare in Octave-Forge vi è il libricino "*Da Coda Al Fine: Pushing Octave's Limits*" di Christoph L. Spiel e Stefan van der Walt (in html e pdf), che risulterà utilissimo a chi non voglia abbandonare la comodità di un ambiente matematico interattivo, ma abbia fra le mani un problema numerico che richiede risorse di calcolo così ampie che una sua implementazione nel linguaggio di Octave risulterebbe troppo lenta. Ecco allora che il testo di Spiel e van der Walt ci accompagna nello scrivere delle estensioni di Octave in linguaggio C++. In questo modo è possibile implementare con un linguaggio compilato e veloce il cuore dell'algoritmo numerico, e demandare all'ambiente interpretato di alto livello i compiti della visualizzazione e dell'*input-output* dei dati, e di eventuali manipolazioni accessorie. In quest'ottica Octave può essere usato, in ambito numerico, un po' come si usa Perl in altre situazioni: un linguaggio 'colla', usato per costruire interfacce tra librerie e programmi di basso livello, amalgamandoli tra loro in una singola applicazione articolata.

Infine, nella raccolta di Octave-Forge esistono molte funzioni che migliorano la compatibilità fra Octave e Matlab. Vista la diffusione di questo ambiente proprietario, esiste una grande pressione da parte della comunità degli utenti (o dei potenziali utenti) sugli sviluppatori di

Octave, per raggiungere la piena compatibilità con Matlab. L'ovvio vantaggio sarebbe quello di poter utilizzare tutti gli *scripts* già sviluppati dagli utenti dell'ambiente proprietario con un programma libero. L'altra faccia della medaglia è che la piena compatibilità richiede di seguire pedissequamente le scelte progettuali fatte dagli sviluppatori di Matlab, alcune delle quali sono discutibili. Pertanto J. W. Eaton (il principale sviluppatore di Octave) ha reso chiaro già da tempo che innovazioni ed evoluzioni del linguaggio Octave non devono essere vincolate alla compatibilità con Matlab. Lo scopo della compatibilità parziale, invece, è quello di rendere facile l'apprendimento di Octave per chi già conosce Matlab. In effetti, per esperienza personale, possiamo affermare che sono sufficienti pochi minuti a qualunque utente di Matlab per essere perfettamente produttivo con Octave.

BOX: un esempio di grafica

Questo breve esempio illustra le potenzialità di Octave nell'attività di ricerca. Un programma di simulazione idrodinamica ha salvato su di un *file* binario le soluzioni numeriche delle equazioni della turbolenza bidimensionale. Si tratta di un modello idealizzato di fluidi il cui moto è dominato da vortici, come nel caso di ampie zone degli oceani, o dell'atmosfera di Giove. Con il breve script riportato di seguito i dati del modello vengono trasferiti in una matrice, manipolati, visualizzati sullo schermo, ed infine l'immagine viene salvata in un *file postscript*.



(Figura 1: Simulazione di turbolenza bidimensionale)

```

#Apre il file 'frame15.dat' in lettura ed in modo binario nativo
fid=fopen('frame15.dat','r','n');
#Legge tutti i dati in singola precisione e li mette nel vettore F
F=fread(fid,inf,'real*4');
#Chiude il file
st=fclose(fid);
#Riarrangia F in forma di matrice quadrata.
#La risoluzione della simulazione era 512x512.
F=reshape(F,512,512);
#Selezioniamo la parte del campo turbolento che ci interessa.
FF=F(200:400,200:400);
#Imponiamo che la scala dell'asse x sia la stessa dell'asse y.
axis("equal")
#Non visualizziamo la griglia fra nodo e nodo (non sara' una
#immagine wireframe).
gset nosurface
#Non visualizziamo la legenda.
gset nokey
#Imponiamo la vista dall'alto.
gset view map
#Selezioniamo la modalita' grafica pm3d
gset pm3d
#Selezioniamo la palette che preferiamo fra le 389017 disponibili
gset palette rgbformulae -25,14,25
#Visualizziamo a schermo una immagine della matrice FF
gsplot FF
#Diamo un nome al grafico ed agli assi.
title("Campo di Turbolenza Bidimensionale")
xlabel("Asse X")
ylabel("Asse Y")
gset cblabel "Scala dei Colori"
#Salviamo il risultato in un file .eps usando font corpo 20.
print("octavefig.eps", "-depsc2", "-F:20")

```

BOX: sistemi lineari e matrici

E' importante studiare i sistemi di equazioni lineari. Questi sono usati per approssimare leggi fisiche molto più complicate come l'equazione di Navier-Stokes per i fluidi, quindi per le previsioni del tempo, per lo studio dell'alimentazione dei motori, per l'aerodinamica delle automobili, degli aerei, ecc.. Si noti che i programmi per la soluzione dei sistemi lineari sono stati tra i primissimi programmi fatti girare sul primo calcolatore mai costruito, l'ENIAC.

La forma più generale di un sistema lineare è la seguente:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \cdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n = b_m \end{cases}$$

(Figura 2: Sistema lineare)

dove x_i sono le incognite (quantità da determinare), a_{ij} sono i coefficienti e b_k sono i termini noti (quantità note, nei casi concreti sono dati sperimentali). Visto che l'informazione contenuta in un sistema lineare è tutta nei coefficienti e nei termini noti, considerate le due tabelle, o matrici, seguenti:

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix} \quad B = \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix}$$

(Figura 3: Matrice associata ad un sistema lineare)

si esprime il sistema lineare della figura 1 come $A \cdot X = B$, dove il puntino è l'operazione di prodotto di matrici. Questa operazione si definisce in modo tale che la prima riga della matrice prodotto $A \cdot X$ abbia come espressione proprio la parte sinistra della prima equazione del sistema (si veda il box corrispondente). Pertanto l'uguaglianza $A \cdot X = B$ è un'uguaglianza tra due matrici che hanno m righe ed una colonna, ed equivale all'insieme delle uguaglianze che costituiscono il sistema lineare.

Il vantaggio nel passaggio dalla forma algebrica alla forma matriciale del sistema lineare è costituito dal fatto che in forma matriciale si può operare sulle matrici come oggetti numerici, dotati di proprie operazioni (si veda il riquadro sul calcolo con matrici), piuttosto che operare in maniera simbolica, compito enormemente più complesso dal punto di vista della programmazione. E' bene ricordare che tutte le proprietà di un sistema lineare si deducono dalla matrice dei coefficienti e dei termini noti. Per approfondimenti, si consulti il libro De Cecco-Vitolo: Note di Geometria e Algebra, disponibile a tutti sul sito web <http://poincare.unile.it/vitolo/>

BOX: prodotto di matrici 'righe per colonne'

Siano date due matrici come in figura

$$Z = \begin{pmatrix} z_{11} & z_{12} & \dots & z_{1n} \\ z_{21} & z_{22} & \dots & z_{2n} \\ \dots & \dots & \dots & \dots \\ z_{m1} & z_{m2} & \dots & z_{mn} \end{pmatrix} \quad W = \begin{pmatrix} w_{11} & w_{12} & \dots & w_{1p} \\ w_{21} & w_{22} & \dots & w_{2p} \\ \dots & \dots & \dots & \dots \\ w_{n1} & w_{n2} & \dots & w_{np} \end{pmatrix}$$

(Figura 4: due matrici)

Allora la matrice prodotto delle due matrici Z e W è la matrice $C = Z \cdot W$ con elementi c_{ij} definiti dalla seguente relazione: $c_{ij} = z_{i1} w_{1j} + z_{i2} w_{2j} + \dots + z_{in} w_{nj}$.

Si noti che il prodotto è definibile solo se Z ha tante colonne quante sono le righe di Z e che la matrice C ha tante righe quante ne ha Z e tante colonne quante ne ha W.

Curriculum

Francesco Paparella insegna fisica matematica presso il corso di laurea in Matematica e

Informatica della facoltà di Scienze dell'Università di Lecce. La sua attività di ricerca si rivolge alla teoria dei sistemi dinamici ed alla fluidodinamica. Per maggiori informazioni, si veda <http://smaug.unile.it/>

Raffaele Vitolo è docente di calcolo matriciale presso la facoltà di Ingegneria dell'Università di Lecce. I suoi interessi di ricerca sono centrati sui modelli geometrici per la meccanica e la teoria dei campi. Per maggiori informazioni, si veda <http://poincare.unile.it/vitolo/>

Riferimenti bibliografici e risorse in rete

Il sito di Octave:

<http://www.octave.org>

Il sito di Octave-Forge:

<http://octave.sourceforge.net>

Netlib raccoglie le implementazioni di riferimento dei principali algoritmi per il calcolo numerico: <http://netlib.org>

Octave è lo strumento didattico principale nel corso *Laboratorio di Sistemi Dinamici* presso il corso di laurea in Matematica e Informatica dell'università di Lecce. Da questa pagina si accede agli *script* utilizzati durante le lezioni:

http://smaug.unile.it/ode_2004.shtml

A chi desideri capire le basi matematiche del calcolo numerico, senza limitarsi a utilizzare algoritmi standard come 'scatole nere', consigliamo:

A. Quarteroni, R. Sacco, F. Saleri, *Matematica numerica*, Springer, 1998.

V. Comincioli, *Analisi numerica: complementi e problemi*, McGraw-Hill, 1991.

Chi voglia approfondire in particolare la teoria e le tecniche del calcolo matriciale e dell'algebra lineare può consultare:

G. De Cecco, R. Vitolo, *Note di geometria ed algebra*. Dispense del corso di Geometria ed Algebra per la laurea di I livello della Facoltà di Ingegneria dell'Università di Lecce, <http://poincare.unile.it/vitolo/>

G. H. Golub, C. F. Van Loan, *Matrix Computations*, Johns Hopkins Univ. Press, 3rd edition, 1996.