

Laboratorio di Analisi Numerica

Lezione 9

Gianna Del Corso <delcorso@di.unipi.it>

Federico Poloni <fpoloni@di.unipi.it>

27 novembre 2012

Quantità di esercizi: in questa dispensa ci sono *più esercizi* di quanti uno studente medio riesce a farne durante una lezione di laboratorio, specialmente tenendo conto anche degli esercizi facoltativi. Questo è perché sono pensate per “tenere impegnati” per tutta la lezione anche quegli studenti che già hanno un solido background di programmazione. Quindi fate gli esercizi che riuscite, partendo da quelli *non* segnati come facoltativi, e non preoccupatevi se non li finite tutti!

1 Frattali di Newton

In questa lezione cercheremo di disegnare i frattali che si ottengono disegnando i bacini di attrazione del metodo di Newton (sul piano complesso) per un polinomio. Le immagini risultanti dovrebbero assomigliare a quella in figura 1. Notate la simmetria della figura: a seconda del quadrante del piano complesso da cui partiamo, si ha convergenza alla più vicina delle tre radici; però, nei punti che sono circa equidistanti da due delle tre radici, si ha un comportamento caotico.

Cominciamo dividendo il problema in molti sottoproblemi più semplici.

1.1 Manipolazione di polinomi

Dato un polinomio, lo rappresentiamo come il vettore dei suoi coefficienti: ad esempio, a $x^3 - 1$ corrisponde il vettore $[1; 0; 0; -1]$. Notare che se il polinomio ha grado n , il vettore ha lunghezza $n + 1$.

Il metodo di Horner per valutare un polinomio corrisponde a fare i prodotti associandoli in questo modo: per esempio, per un polinomio di grado 4,

$$a(x) = (((a_4 * x + a_3) * x + a_2) * x + a_1) * x + a_0.$$

Esercizio 1. Scrivere una **function** `y=horner(p,x)` che prende un polinomio p (rappresentato come il vettore dei suoi coefficienti) e un numero x e calcola $p(x)$ con il metodo di Horner.

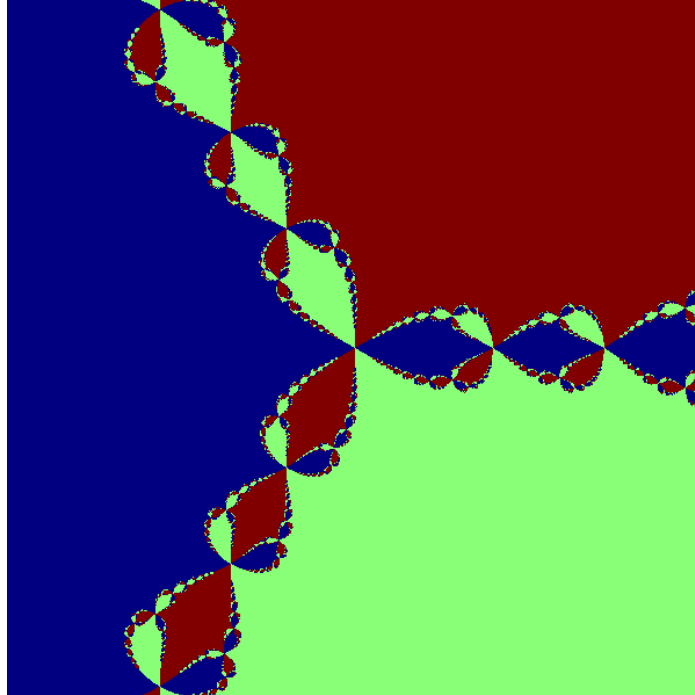


Figura 1: Disegno dei bacini di attrazione del metodo di Newton per il polinomio $x^3 + 1$. I tre colori diversi corrispondono ai punti del piano complesso a partire da cui Newton converge alle tre diverse radici.

Esercizio 2. Scrivere una **function** `dp=derivata(p)` che prende un polinomio p (vettore di coefficienti) e restituisce la sua derivata (vettore di coefficienti).

1.2 Metodo di Newton

Il metodo di Newton è l'iterazione

$$x_{k+1} = x_k - \frac{p(x_k)}{p'(x_k)}.$$

Esercizio 3. Scrivere una **function** `x=newton(p,x0)` che esegue il metodo di Newton sul polinomio p partendo dal punto iniziale x_0 . Come criterio di arresto, si può usare quello di terminare se $|p(x)| \leq 10^{-12}$:

```
function x=newton(p,x0);
    x=x0;
    px=horner(p,x0);
    while(abs(px)>1E-12)
```

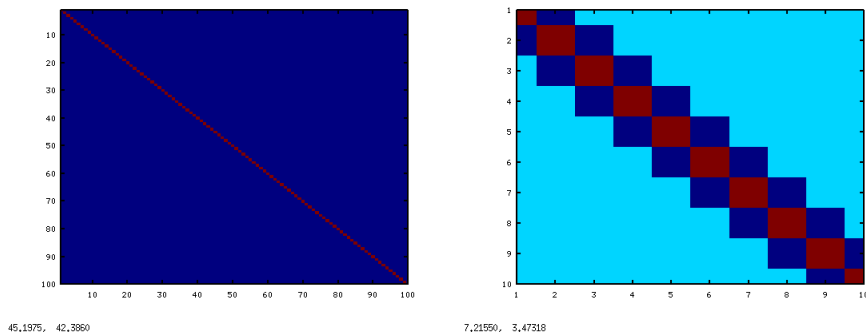


Figura 2: `imagesc(eye(100))` e `imagesc(laplacian(10))`

```
%calcola il nuovo x e il nuovo px
endwhile
endfunction
```

Testare il metodo di Newton sul polinomio $p(x) = x^3 + 1$. Quali sono le sue radici? Riuscite a trovare un valore iniziale per il metodo di Newton che lo faccia convergere ad ognuna di esse? Ricordate che il modo più semplice per inserire un numero complesso in Octave è $2+3I$.

1.3 “Disegnare” una matrice

La funzione `imagesc` prende come parametro una matrice $m \times n$ A , e genera un’immagine $m \times n$ in cui il pixel i, j è colorato di un colore che varia su una scala da rosso a blu a seconda di quanto il valore di $A_{i,j}$ è grande/piccolo rispetto agli altri elementi della matrice. Per esempio, con

```
octave:1> imagesc(eye(100))
```

viene visualizzata un’immagine in cui la diagonale (elementi più grandi) è rossa, e tutti gli altri elementi (elementi più piccoli) sono blu. Provate anche `imagesc(laplacian(10))` o `imagesc(rand(100))`.

1.4 Frattale di Newton

Per disegnare il frattale di Newton relativo al polinomio $p(x) = x^3 + 1$, abbiamo bisogno innanzitutto di una funzione che “decida” a quale valore c’è convergenza.

Esercizio 4. Scrivere una funzione `function val=decidi(x)` che restituisca 1, 2 o 3 a seconda se il numero complesso x è più vicino a -1 , a $\frac{1}{2} + I\frac{\sqrt{3}}{2}$, o a $\frac{1}{2} - I\frac{\sqrt{3}}{2}$.

Esercizio 5. Scrivere una `function img=newtonfractal()` che non prende alcun argomento e restituisce una matrice 101×101 chiamata `img` calcolata in questo modo:

- genera 101 valori equispaziati nell'intervallo $[-2, 2]$ con l'istruzione `range=-2:0.04:2`.
- per ogni coppia (i, j) :
 - calcola il punto z_0 del piano complesso `z0=range(i)+1I*range(j)`;
 - esegue il metodo di Newton per il polinomio $x^3 + 1 = 0$ partendo dal punto `z0`;
 - utilizzando la funzione `decidi()`, scrive 1, 2 o 3 in `img(i, j)` a seconda della radice del polinomio a cui si ha convergenza a partire dal valore iniziale `z0`.¹.
- restituisce la matrice `img`

La funzione qui scritta sarà probabilmente abbastanza lenta (potrebbe metterci un mezzo minuto...) e restituirà una matrice `img` che potrete poi visualizzare a schermo con l'istruzione `imagesc(img)`. Assomiglia alla figura 1?

2 Esercizi facoltativi

Esercizio 6 (facoltativo). Generate l'immagine corrispondente per il metodo di Newton su altri polinomi. Non sapete le radici? Potete farle calcolare a Octave: la funzione `roots(p)` calcola le radici di un polinomio (rappresentato come vettore di coefficienti): per esempio,

```
octave:2> roots([1 -1 -1 ]) %radici di x^2-x-1=0
ans =
-0.61803
 1.61803
```

Se volete, potete riscrivere le funzioni scritte finora in modo che il polinomio `p` non sia fissato ma sia uno degli argomenti.

2.1 Frattale di Julia

Il *frattale di Julia* relativo al numero complesso c è definito come l'insieme dei punti z_0 per cui la successione definita da $z_{k+1} = z_k^2 + c$ non diverge.

Esercizio 7 (facoltativo). Scrivete una `function img=julia(c)` che restituisca il frattale di Julia associato a c . La funzione:

- genera 101 valori equispaziati nell'intervallo $[-2, 2]$ con l'istruzione `range=-2:0.04:2`.
- per ogni coppia (i, j) :
 - calcola il punto z_0 del piano complesso `z0=range(i)+1I*range(j)`;
 - applica per 10 volte la funzione $f(z) = z^2 + c$ a partire dal punto z_0

¹potete dare per scontato che il metodo di Newton converga per tutti i valori iniziali nel nostro range.

- scrive in `img(i,j)` l'*arcotangente* del modulo del numero complesso z_{10} così calcolato. Difatti i numeri hanno variazioni molto grosse (da 0 a $10^{300}\dots$), e disegnarli così come sono non produrrebbe un risultato interessante.

- restituisce la matrice `img`

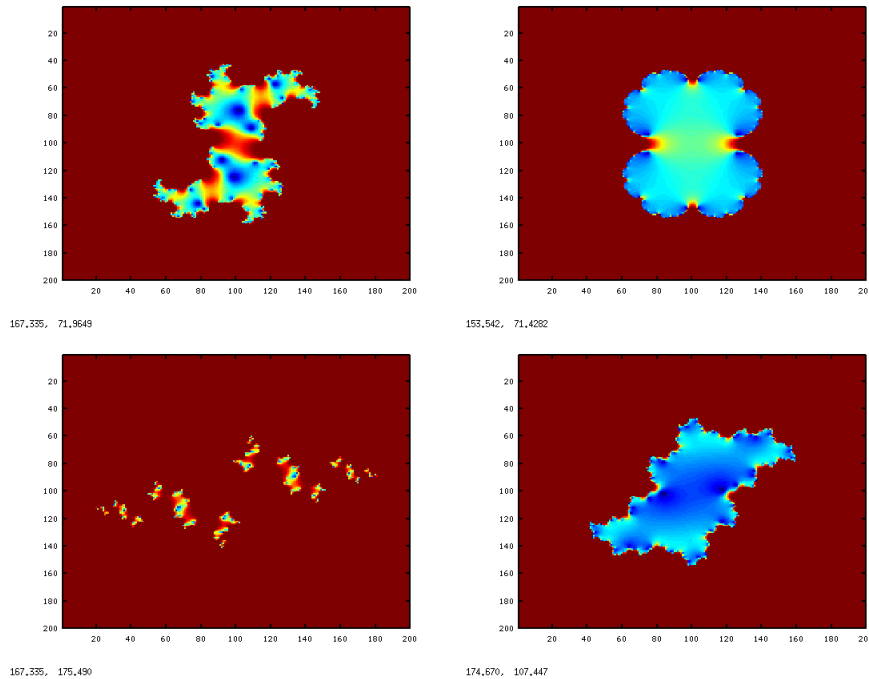


Figura 3: Alcuni frattali di Julia

2.2 Versioni vettorizzate

Il programma `newton.m` è abbastanza lento; difatti non abbiamo dedicato alcuna attenzione all'ottimizzazione del numero di istruzioni eseguite (sono questioni tecniche e noiose — non è lavoro per un matematico). Un primo passo per velocizzarlo è sostituire `horner` e `derivata` con le funzioni equivalenti che già esistono in Octave, `polyval` e `polyderiv` (controllare la sintassi con `help`).

Un altro miglioramento potete provare a farlo da soli:

Esercizio 8 (facoltativo). Utilizzando le funzioni `polyval` e `polyderiv` applicate con una matrice come secondo argomento (`help polyval`) e le operazioni elemento-per-elemento (ad esempio `.*`, `./`), scrivete un'istruzione che esegua un passo del metodo di Newton *contemporaneamente* su tutti i valori contenuti in una matrice X .

Un po' più complicato infine è rendere efficiente la funzione `decidi` per Newton. Riportiamo qui una versione più veloce delle funzioni che disegnano i frattali di Newton e Julia, che utilizza tutte queste ottimizzazioni. Se volete potete usarli per generare immagini più grandi o per zoomare su alcuni dettagli e studiare la forma dei due frattali.

```
function img=disegna_newton(dimensione)
%restituisce un frattale di Newton nxn. Visualizzare con imagesc(img)
p=[1;0;0;1]; %coefficienti del polinomio
n=dimensione;
range=-2:4/(n-1):2; %griglia di punti

deg=length(p)-1;
dp=polyderiv(p);
radici=roots(p);

%matrice dei valori iniziali
X=ones(n,1)*range+range'*ones(1,n)*1I;

%esegue 10 passi di Newton "in parallelo", dovrebbero bastare
for k=1:10
    X=X-polyval(p,X)./polyval(dp,X);
endfor

%trova il punto piu' vicino "in parallelo" con un trucco:
%
%generiamo un "tensore", cioe' un oggetto a tre indici
%M(i,j,k)=X_{j,k}-radici(i)
%
%usiamo la funzione min: [values,positions]=min(A) calcola il minimo
%lungo la prima dimensione (righe)
%e restituisce in positions le /posizioni/ in cui si trovano
%i minimi su ogni riga
%ad es. [v p]=min([4.5 2.5 8 9]) restituisce v=2.5, p=2
%(perche' il minimo sta in posizione 2)
M=zeros(deg,n,n);
for i=1:deg
    M(i,,:)=abs(X-radici(i));
endfor
[values positions]=min(M);

%ritrasforma positions da un "tensore" 1xnxn a una matrice nxn
img=reshape(positions,[n n]);
endfunction

function img=julia(c)
```

```
%function img=julia(c)
%restituisce il disegno del frattale di Julia con parametro c
%plottare con imagesc(julia(c))
n=200;
range=-2:4/(n-1):2; %griglia di punti

%matrice dei valori iniziali
X=ones(n,1)*range+range'*ones(1,n)*1I;

for k=1:10
    X=X.*X+c;
endfor

img=atan(abs(X));
endfunction
```