

6 Esempi

In questo paragrafo applichiamo le nozioni sull'uso di MATLAB esposte nelle pagine precedenti alla risoluzione di alcuni esempi relativi ad argomenti trattati nel Corso di Elementi di Analisi Matematica (A) e Geometria.

6.1 Esempio 1: Calcolo di integrali in una dimensione

Data la funzione continua $f : [a, b] \rightarrow \mathbb{R}$ consideriamo la suddivisione di $[a, b]$ individuata dagli $(n + 1)$ punti equispaziati:

$$a = x_0, \quad x_1, x_2, \dots, x_{n-1}, \quad x_n = b$$

con

$$x_j = a + jh, \quad h = \frac{b-a}{n}, \quad j = 0, \dots, n$$

Scegliamo all'interno di ciascun intervallo $[x_j, x_{j+1}]$ un punto arbitrario ξ_j e costruiamo la seguente somma (detta *somma di Riemann*):

$$S_n = \sum_{j=0}^{n-1} f(\xi_j)(x_{j+1} - x_j) = \frac{b-a}{n} \sum_{j=0}^{n-1} f(\xi_j) \quad (2)$$

Ricordiamo il teorema (cfr.[1]):

$$\lim_{n \rightarrow +\infty} S_n \text{ esiste finito ed è indipendente dalla scelta dei punti } \xi_j.$$

Definiamo quindi $\lim_{n \rightarrow +\infty} S_n$ *integrale* di f su $[a, b]$ e scriviamo:

$$\int_a^b f(x)dx = \lim_{n \rightarrow +\infty} \frac{b-a}{n} \sum_{j=0}^{n-1} f(\xi_j) \quad (3)$$

Sia $f \geq 0 \quad \forall x \in [a, b]$ (si veda la Figura 18). L'addendo j -esimo della somma (2) può essere interpretato come l'area del rettangolo Q_j avente come base il segmento $[x_j, x_{j+1}]$ e come altezza $f(\xi_j)$, infatti:

$$(x_{j+1} - x_j)f(\xi_j) = \text{base}(Q_j) \cdot \text{altezza}(Q_j)$$

La somma S_n rappresenta quindi un'approssimazione dell'area della parte di piano compresa tra l'intervallo $a \leq x \leq b$ e il grafico di f (il cosiddetto *trapezioid*e individuato da f).

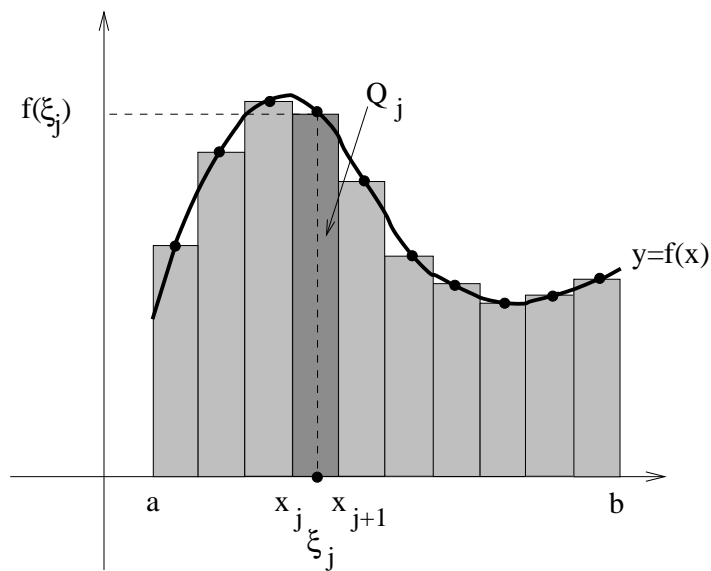


Figura 18: Interpretazione geometrica dell'integrale della funzione $\int_a^b f(x)dx$ con la scelta $\xi_j = \frac{x_j + x_{j+1}}{2}$

Prendiamo $\xi_j = \frac{x_j + x_{j+1}}{2}$ (ottenendo la formula composta del *punto medio* per la valutazione numerica dell'integrale) e valutiamo con MATLAB il valore della somma (2) ripetendo il calcolo con un numero crescente di suddivisioni dell'intervallo $[a, b]$. Definiamo l'errore commesso utilizzando n suddivisioni come:

$$E_n(f) = |I(f) - I_n(f)|$$

dove $I(f)$ è il valore esatto dell'integrale e $I_n(f)$ è il valore calcolato con la formula del punto medio.

La funzione `puntomedio` ripete il calcolo della somma (2) con un numero di suddivisioni n che va da 1 a `maxnint`. Essa richiede in ingresso gli estremi dell'intervallo `a,b`, il numero massimo `maxnint` di suddivisioni da utilizzare, la stringa `fun` che definisce la funzione integranda ed il valore esatto dell'integrale `intex`. Vengono restituiti in uscita i vettori `int`, `err` contenenti rispettivamente i valori dell'integrale calcolato numericamente e del corrispondente errore in modo tale che `int(i)` e `err(i)` sono il valore dell'integrale e dell'errore ottenuti utilizzando "i" suddivisioni.

```
function [int,err]=puntomedio(a,b,maxnint,fun,intex)
%
% Formula composta del punto medio
% [int,err]=puntomedio(a,b,maxnint,fun,intex)
% a,b: estremi dell'intervallo di integrazione
% maxnint: numero massimo di suddivisioni dell'intervallo [a,b]
% fun: stringa contenente la funzione integranda
% intex: valore dell'integrale esatto
% int, err: vettori contenenti i valori dell'integrale
% calcolato numericamente e del corrispondente errore

for i=1:maxnint
    h=(b-a)/i;
    x=[a+h/2:h:b];
```

```

dim = max(size(x));
y=eval(fun);
if size(y)==1,
    y=diag(ones(dim))*y;
end
int(i)=h*sum(y);
err(i)=abs(int(i)-intex);
end
loglog((1:maxnint),err)

```

Sia ad esempio $f = \cos(x)$ e $[a, b] = [-\pi/2, \pi/2]$. Valutiamo quindi l'integrale:

$$I(f) = \int_{-\pi/2}^{\pi/2} \cos(x) dx$$

il cui valore esatto è $I(f) = 2$ con le seguenti istruzioni:

```

>> a=-pi/2; b=pi/2;
>> fun='cos(x)';
>> maxnint=1000;
>> intex=2;
>> [int,err]=puntomedio(a,b,maxnint,fun,intex);

```

L'andamento dell'errore è riportato in Figura 19. Dal grafico osserviamo che al crescere del numero di intervalli l'errore decresce, ovvero la successione delle somme S_n tende al valore esatto dell'integrale. Si può mostrare inoltre (si veda ad esempio [3], Cap. 9) che vale la seguente stima dell'errore per la formula composta del punto medio:

$$E_n(f) \leq \frac{b-a}{24} M h^2$$

dove si è supposto che $f \in C^2([a, b])$ e si è definito $M = \max_{x \in [a, b]} |f''(x)|$. Notiamo infatti che aumentando di *un ordine di grandezza* il numero di suddivisioni, ovvero passando per esempio da 100 a 1000 suddivisioni, l'errore si riduce di *due ordini di grandezza*, ovvero passa da 10^{-4} a 10^{-6} , confermando la riduzione quadratica dell'errore rispetto al parametro h .

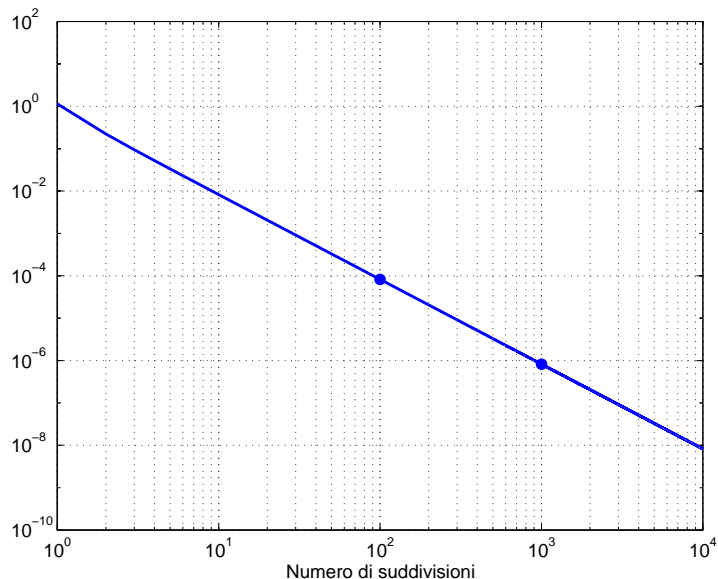


Figura 19: Andamento dell'errore commesso utilizzando nella sommatoria (2) un numero n crescente di suddivisioni

6.2 Esempio 2: Visualizzazione di un vettore trasformato per roto-traslazione

Si vuole studiare il movimento nel piano di un vettore, in particolare l'effetto che hanno su di esso i movimenti rigidi di rotazione e traslazione. Indichiamo con $\mathbf{x}_{old} = (\mathbf{Q} - \mathbf{P})$ il vettore di partenza avente come estremi i punti \mathbf{P} e \mathbf{Q} (entrambi riferiti ad un sistema assoluto di coordinate cartesiane ortogonali) e orientato da \mathbf{P} verso \mathbf{Q} . Analogamente, $\mathbf{x}_{new} = (\mathbf{Q}' - \mathbf{P}')$ è il vettore finale ottenuto in seguito alla trasformazione di \mathbf{x}_{old} (si veda la Figura 20).

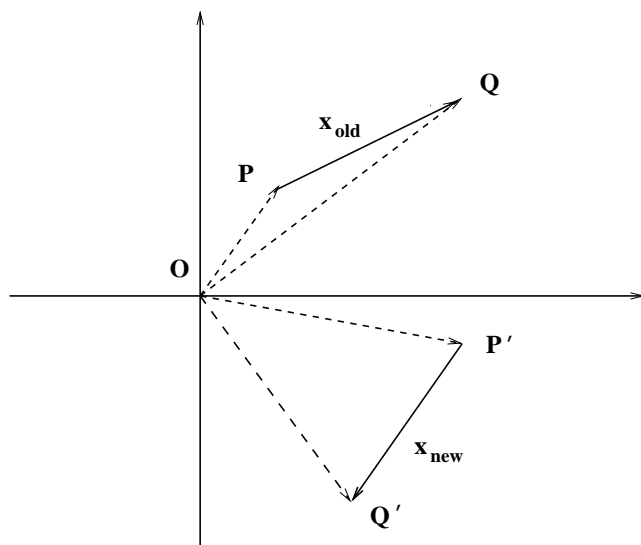


Figura 20: Rototraslazione di un vettore nel piano

Tale trasformazione è caratterizzata da tre parametri reali: l'angolo di rotazione θ misurato in gradi e positivo se corrisponde ad una rotazione antioraria, il centro di rotazione \mathbf{C}_R , il vettore di traslazione $\boldsymbol{\tau}$ (le cui componenti indicano la traslazione rigida

rispettivamente lungo x e y). Il legame fra i due vettori è rappresentato dalla seguente *trasformazione lineare*

$$\begin{aligned}\mathbf{P}' &= (\mathcal{R}(\mathbf{P} - \mathbf{C}_R) + \mathbf{C}_R + \boldsymbol{\tau}) \\ \mathbf{Q}' &= (\mathcal{R}(\mathbf{Q} - \mathbf{C}_R) + \mathbf{C}_R + \boldsymbol{\tau})\end{aligned}\tag{4}$$

dove \mathcal{R} è la matrice di rotazione

$$\mathcal{R} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix}$$

Il legame fra \mathbf{x}_{old} e \mathbf{x}_{new} è quindi

$$\mathbf{x}_{new} = R\mathbf{x}_{old}\tag{5}$$

Si osservi che quest'ultima relazione non ci dà alcuna informazione su dove siano posizionati i due vettori ma definisce unicamente il legame fra i loro versi, moduli e direzioni. Di fatto tale legame non specifica il punto di applicazione di nessuno dei due vettori poiché la (5) è una relazione lineare omogenea che fa intervenire soltanto le differenze di coordinate $\mathbf{Q}' - \mathbf{P}'$ e $\mathbf{Q} - \mathbf{P}$. In particolare la lunghezza dei due vettori è uguale. Ciò dipende dal fatto che la matrice di rotazione non altera la lunghezza dei vettori, ed è infatti un esempio di matrice ortogonale. Le relazioni (4) caratterizzano invece completamente i due vettori, specificandone anche il punto di applicazione. Si noti inoltre che l'ordine delle varie trasformazioni elementari è rotazione, e traslazione. La scelta inversa comporterebbe soltanto un diverso punto di applicazione del vettore finale.

La function `RotoTrasla` implementa la trasformazione sopra descritta ricevendo in ingresso i seguenti parametri:

- `oldP`, `oldQ` vettori colonna delle coordinate degli estremi del vettore `old` rispetto all'origine del sistema di riferimento
- `theta` angolo di rotazione in gradi e positivo se antiorario rispetto all'asse delle ascisse
- `centroR` vettore colonna delle coordinate del centro di rotazione
- `trasl` vettore colonna delle componenti della traslazione

In uscita vengono restituite le coordinate degli estremi del vettore trasformato \mathbf{x}_{new} . Inoltre, ad ogni passo della trasformazione (rotazione, traslazione) viene generato un grafico che mostra il vettore prima e dopo la trasformazione stessa.

```
function [newP, newQ] = RotoTrasla(oldP, oldQ, theta, centroR, trasl)
%
% [newP, newQ] = RotoTrasla(oldP, oldQ, theta, centroR, trasl)
%
%
% Calcola il vettore new, che ha per estremi
% i punti individuati dai vettori newP e newQ ed e' diretto da newP
% verso newQ,
% Esso e' ottenuto per rotazione di un angolo theta attorno al
% punto centroR e per traslazione pari a trasl del vettore old
```

```

% che ha per estremi i punti individuati dai vettori oldP e oldQ
% ed e' diretto da oldP verso oldQ.
% Nel grafico sono riportate le successive trasformazioni che
% permettono di passare da old a new
%
% oldP, oldQ : vettori colonna delle coordinate degli estremi
% del vettore old rispetto all'origine del sistema di riferimento
% theta: rotazione in gradi (in senso antiorario positiva)
% centroR: vettore colonna delle coordinate del centro di rotazione
% trasl: vettore colonna che assegna la traslazione lungo x e y
% newP, newQ : vettori colonna delle coordinate degli estremi
% del vettore new rispetto all'origine del sistema di riferimento
%
% La trasformazione e':
%
% newP = ( R * (oldP - centroR) + CentroR + trasl)
% newQ = ( R * (oldQ - centroR) + CentroR + trasl)
%
% e quindi
%
% (newQ - newP) = R * (oldQ - oldP)
%
%
close all;

c = cos(pi*theta/180); s = sin(pi*theta/180);
R = [c -s; s c];

Pr = R * (oldP - centroR) + centroR;
Qr = R * (oldQ - centroR) + centroR;
Pt = Pr + trasl; Qt = Qr + trasl;
newP = Pt; newQ = Qt;

Raggioi=norm(oldP-centroR);
Raggioe=norm(oldQ-centroR);
xi=centroR(1)-Raggioi:0.005:centroR(1)+Raggioi;
xe=centroR(1)-Raggioe:0.005:centroR(1)+Raggioe;
yi=centroR(2)+sqrt(Raggioi^2-(xi-centroR(1)).^2);
ye=centroR(2)+sqrt(Raggioe^2-(xe-centroR(1)).^2);
xir=[Pr(1),Pt(1)];
yir=[Pr(2),Pt(2)];
xer=[Qr(1),Qt(1)];
yer=[Qr(2),Qt(2)];

subplot(2,2,1),plot([oldP(1), oldQ(1)],[oldP(2),oldQ(2)],'r'),hold on
plot([Pr(1), Qr(1)],[Pr(2),Qr(2)],'b')
plot(oldQ(1),oldQ(2),'dr',Qr(1),Qr(2),'db'),
text = '1. Rotazione di %g^0 del vettore con centro [%g,%g]';
title(sprintf(text,theta,centroR(1),centroR(2)))

```

```

plot(centroR(1),centroR(2),'k*')
ax=gca; setgrafico(ax,'b','r',1),
axis('manual')
plot(xi,yi,'--k',xi,2*centroR(2)-yi,'k--',xe,ye,...
'k--',xe,2*centroR(2)-ye,'k--')

subplot(2,2,2), plot([Pr(1),Qr(1)],[Pr(2), Qr(2)],'b'),hold on,
plot([Pt(1), Qt(1)],[Pt(2),Qt(2)],'g')
plot(Qr(1),Qr(2),'db',Qt(1),Qt(2),'dg')
text = '2. Traslazione di [%g,%g] del vettore ruotato';
title(sprintf(text,trasl(1),trasl(2)))
ax = gca; setgrafico(ax,'g','b',0)
axis('manual')
plot(xir,yir,'k--',xer,yer,'k--')

```

```

subplot(2,2,3)
plot([oldP(1), oldQ(1)],[oldP(2), oldQ(2)],'r'),hold on
plot([newP(1), newQ(1)],[newP(2),newQ(2)],'g')
plot(oldQ(1),oldQ(2),'dr',newQ(1),newQ(2),'dg')
title('3.Vettore iniziale e vettore finale');
ax=gca; setgrafico(ax,'g','r',0)

```

La seguente function setgrafico imposta le proprietà del grafico:

```

function setgrafico(ax,color1,color2,flag)
% imposta le dimensioni del carattere e lo
% spessore delle linee per le figure
% generate dalla function RotoTrasla

axis('equal')

ch = get(ax,'Children');
if(flag==0)
set(ch(3),'LineWidth',2); set(ch(4),'LineWidth',2);
set(ch(1),'MarkerSize',7); set(ch(2),'MarkerSize',7);
set(ch(1),'MarkerFaceColor',color1); set(ch(2),'MarkerFaceColor',color2);
set(gca,'FontSize',[10]);
else
set(ch(4),'LineWidth',2); set(ch(5),'LineWidth',2);
set(ch(2),'MarkerSize',7); set(ch(3),'MarkerSize',7);
set(ch(2),'MarkerFaceColor',color1); set(ch(3),'MarkerFaceColor',color2);
set(gca,'FontSize',[10]);
end

```

Eseguiamo il programma con i seguenti parametri

```

>> [newP, newQ] = RotoTrasla([2;2], [3;3], -60, [1;2], [1;1]);
newP =

```

```

2.5000
2.1340

```

newQ =

3.8660

1.7679

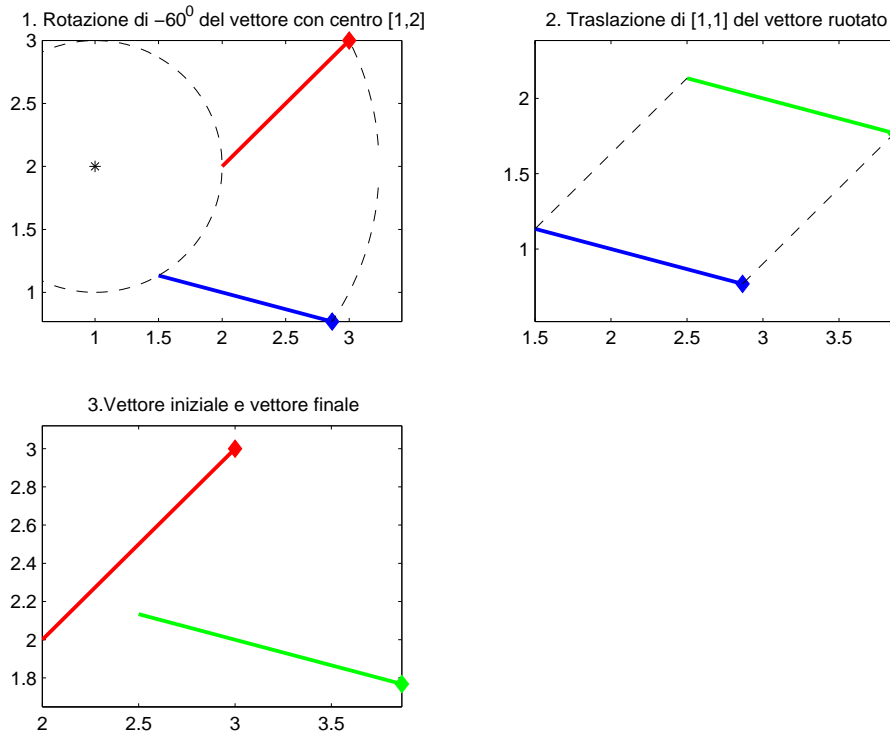


Figura 21: Trasformazione di un vettore nel piano

6.3 Esempio 3: Generazione e visualizzazione di successioni per ricorrenza

In questo esempio ci occupiamo di generare e visualizzare con l'ausilio di MATLAB una successione ottenuta per ricorrenza. Precisamente, tratteremo successioni del tipo

$$\begin{cases} x_0 & \text{assegnato} & \text{punto o condizione iniziale} \\ x_{n+1} = g(x_n) & n \geq 0 & \text{legge di ricorrenza} \end{cases}$$

La function `ricorr` implementa l'algoritmo di generazione della successione (detta anche orbita) a partire da x_0 e la sua visualizzazione tramite un opportuno diagramma a gradino. I parametri di ingresso sono i seguenti:

- `x0` punto iniziale che genera l'orbita
- `g` funzione generatrice definita come stringa
- `a, b` estremi dell'intervallo di esplorazione
- `nmax` numero massimo di iterazioni consentito

- **toll** tolleranza per criterio d'arresto. Nell'implementazione su calcolatore della legge di ricorrenza è infatti necessario introdurre un test per stabilire quando arrestare la generazione delle iterate x_n della successione. A questo proposito, si noti come esistano due sole possibilità per interrompere la generazione delle iterate x_k nel programma **ricorr**: la prima consiste nel soddisfare la condizione

$$|x_{n+1} - x_n| < \text{toll} \quad (6)$$

mentre la seconda equivale ad avere eseguito il numero massimo **nmax** di iterazioni fissato a priori *senza* avere verificato il criterio (6).

I parametri di uscita sono:

- **succ** vettore corrispondente all'orbita generata
- **it** numero di termini della successione effettivamente calcolati.

Il programma è il seguente

```
function [succ,it] = ricorr(x0,g,a,b,nmax,toll)
%
% [succ,it] = ricorr(x0,g,a,b,nmax,toll);
%
% Calcola successioni per ricorrenza:
%
% x0 : punto iniziale
% g : funzione generatrice
% a,b : estremi dell'intervallo di esplorazione
% nmax: numero massimo di iterazioni consentito
% toll: tolleranza per criterio d'arresto
%
% succ: orbita generata
% it : numero di iterazioni effettivo
%
x = a:0.001:b;
plot(x,x,'g',x,eval(g),'b')
hold on
succ = x0;
x = x0;
x = eval(g);
succ = [succ;x];
plot([x0 x0],[0 x],'r')
plot([x0 x],[x x],'r')
err = 1; it = 1;
while (err >= toll & it < nmax)
    xold = x;
    x = eval(g);
    succ = [succ;x];
    plot([xold xold],[xold x],'r')
    plot([xold x],[x x],'r')
    err = abs(x - xold);
```

```

    it = it + 1;
end
hold off
%
if it < nmax % convergenza
    fprintf(' \n Numero di Iterazioni : %d \n',it);
    fprintf(' Radice calcolata      : %12.8f \n\n',succ(it+1));
else
    fprintf(' \n Numero di Iterazioni Massimo raggiunto \n\n');
end

```

Consideriamo ad esempio la successione di Fibonacci la cui funzione generatrice è

$$g(x) = 1 + \frac{1}{x}$$

e poniamo $x_0 = 1$ con intervallo di esplorazione $[0.5, 2.5]$.

```
>> [succ,it] = ricorr(1,'1+1./x',0.5,2.5,100,1e-10);
```

```

Numero di Iterazioni : 26
Radice calcolata      : 1.61803399

```

Come noto la successione di Fibonacci converge al limite finito $(1+\sqrt{5})/2 = 1.6180339887$ (la cosiddetta *sezione aurea*). La successione delle iterate $\{x_n\}$ generate dal programma tende quindi correttamente al *punto fisso* $\alpha = g(\alpha) = 1.61803399$ che si dice essere *attrattore* dell'orbita. Osserviamo inoltre come il comportamento a spirale dell'orbita sia dovuto al fatto che $-1 < g'(\alpha) < 0$.

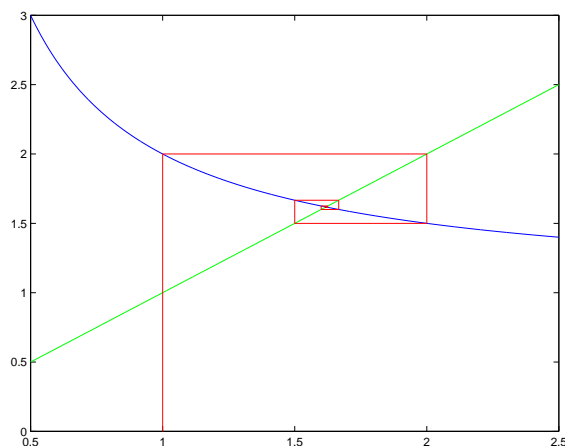


Figura 22: Diagramma a gradino dell'orbita generata dalla successione di Fibonacci

Se invece fosse $0 < g'(\alpha) < 1$ le iterate $\{x_n\}$ tenderebbero al punto fisso α in modo monotono. Consideriamo a questo proposito la funzione generatrice

$$g(x) = \sqrt{x}$$

e studiamo il comportamento della successione generata partendo da $x_0 = 0.5$ con intervallo di esplorazione $[0.25, 1.5]$.

```
>> [succ,it] = ricorr(0.5,'sqrt(x)',0.25,1.5,100,1e-10);
```

Numero di Iterazioni : 33

Radice calcolata : 1.00000000

È evidente dal grafico di Figura 23 la convergenza monotona delle iterate $\{x_n\}$ al punto fisso $\alpha = 1$.

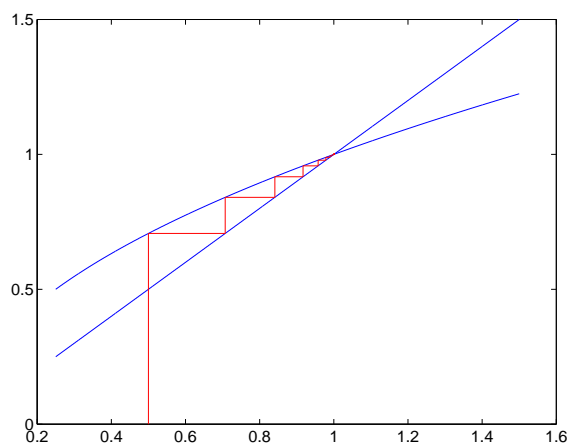


Figura 23: Diagramma a gradino dell'orbita generata dalla funzione $g(x) = \sqrt{x}$. Si noti la convergenza monotona delle iterate al punto fisso

Come ultimo esempio, consideriamo la funzione generatrice

$$g(x) = \begin{cases} 2x & 0 \leq x \leq 1 \\ \frac{1}{2} & x > 1 \end{cases}$$

e studiamo il comportamento della successione generata partendo da $x_0 = 2$

```
>> [succ,it] = ricorr(2,'2*x.*(0<=x & x<=1) + 1/2*(x>1)',...
0,5,100,1e-6);
```

Numero di Iterazioni Massimo raggiunto

Il grafico ottenuto è riportato in Figura 24.

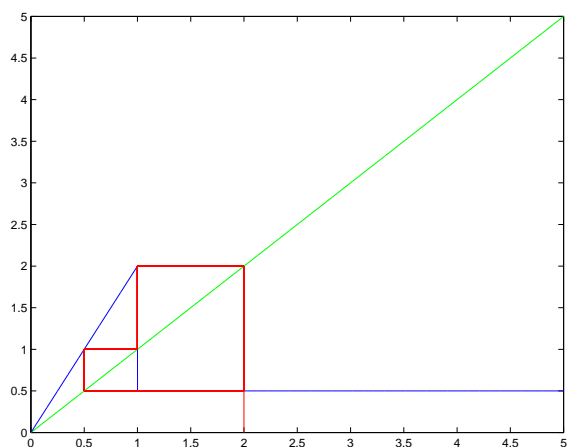


Figura 24: Diagramma a gradino dell'orbita

Il programma segnala che è stato raggiunto il numero massimo di iterazioni. Ciò significa che si è verificata una delle seguenti situazioni:

1. la successione ammette un punto fisso ma la tolleranza imposta è troppo stringente rispetto al numero massimo di iterazioni previsto. In tal caso si deve aumentare `nmax` ed eventualmente incrementare `toll`
2. la successione non ammette un punto fisso e tende a divergere. In tal caso non c'è niente da fare !
3. la successione non ha un punto fisso ma un ciclo limite. È sufficiente ispezionare direttamente `succ` per rendersene conto.

Nell'esempio, osserviamo che l'orbita *non presenta alcun punto fisso* ma che al contrario si manifesta un comportamento oscillante fra i valori $2, 1/2, 1, 2, 1/2, 1, \dots$, cioè un'orbita di periodo 3.

Cogliamo qui l'occasione per segnalare la sintassi che si può utilizzare per definire funzioni che, come accade per g in questo caso, hanno una rappresentazione diversa su vari intervalli dell'insieme di definizione (si veda l'Osservazione seguente).

Osservazione: funzioni con rappresentazione diversa su vari intervalli Sfruttiamo le proprietà degli operatori di confronto che, come visto, restituiscono valore 1 per “vero” e 0 per “falso”. Nell'esempio appena esaminato si ha

```
2*x.*(0<=x & x<=1) + 1/2*(x>1)
```

Si noti che $(0 \leq x \text{ \& \& } x \leq 1)$ è un vettore e che quindi è necessario utilizzare le operazioni “puntate”, inoltre esso vale 1 solo nell'intervallo $0 \leq x \leq 1$ e 0 altrove e viceversa che $(x > 1)$ vale 1 per $x > 1$ e 0 altrove.

In alternativa agli operatori logici di confronto si può utilizzare anche la function MATLAB `stepfun` del toolbox `control`. Il comando `stepfun(t,t0)`, dove `t` è un vettore di valori monotoni crescenti e `t0` un valore di soglia reale, restituisce un vettore della stessa lunghezza di `t` con componenti 0 ove $t < t_0$ e 1 ove $t \geq t_0$. Ad esempio, il grafico della funzione g studiata in precedenza può essere tracciato con le seguenti istruzioni

```
>> x=[0:0.01:2];
>> g=2*x.*(stepfun(x,0)-stepfun(x,1))+1/2*stepfun(x,1);
```

Come ulteriore esempio dell'uso di `stepfun` consideriamo la funzione continua

$$f(x) = \begin{cases} x^2 & 0 \leq x \leq 1 \\ 1 & 1 \leq x \leq 2 \\ 1 - (x-2)^2 & 2 \leq x \leq 4 \end{cases}$$

Essa si costruisce con la seguente sintassi

```
>> x = [0:0.01:4];
>> f = x.^2.*(stepfun(x,0) - stepfun(x,1)) + ...
        (stepfun(x,1) - stepfun(x,2)) + ...
        (1 - (x-2).^2).*stepfun(x,2);
>> plot(x,f)
>> axis([0 4 -3 2])
>> ch = get(gca,'Children');
>> set(ch,'LineWidth',2)
>> set(gca,'FontSize',14)
```

Il grafico che si ottiene è riportato in Figura 25. Si notino i comandi utilizzati per evidenziare meglio la figura.

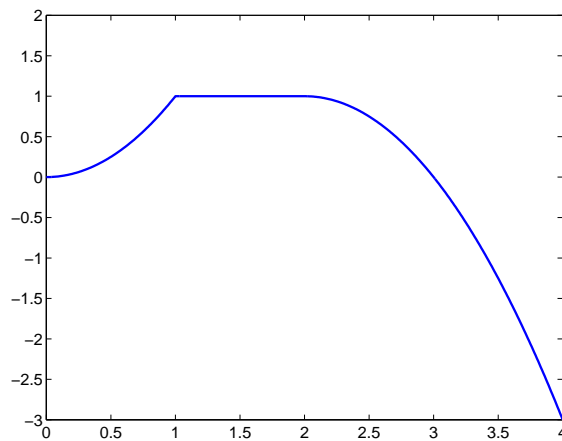


Figura 25: Funzione definita *a tratti*

6.4 Esempio 4: Approssimazione di una funzione con polinomi di Taylor.

Consideriamo una funzione $f : (a, b) \rightarrow \mathbb{R}$ derivabile $n + 1$ volte in $x_0 \in (a, b)$. Vale allora la *formula di Taylor*:

$$f(x) = T_n(x) + R_{n+1}(x)$$

dove

$$T_n(x) = \sum_{k=0}^n \frac{f^{(k)}(x_0)}{k!} (x - x_0)^k$$

è il *polinomio di Taylor* di grado n centrato in x_0 associato ad f , e

$$R_{n+1}(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} (x - x_0)^{n+1}$$

è il *resto secondo Lagrange*.

Se inoltre $f \in C^\infty(a, b)$, il limite per $n \rightarrow \infty$ di T_n esiste unico e prende il nome di *serie di Taylor* associata ad f . Pertanto, in questo caso il polinomio di Taylor di grado n coincide con la *troncata* di ordine n della serie di Taylor di f .

In questo esempio analizziamo la qualità dell'approssimazione di f ottenuta tramite T_n al crescere del grado n . Consideriamo la funzione $f(x) = \sin(x)$; il suo sviluppo in serie di Taylor è

$$\sin(x) = \sum_{k=0}^{\infty} (-1)^k \frac{x^{2k+1}}{(2k+1)!} = x - \frac{x^3}{6} + \frac{x^5}{120} - \dots$$

Studiamo l'approssimazione che si ottiene nell'intervallo $[0, \pi]$ con i polinomi di Taylor di grado 1, 3 e 5 rispetto al punto $x_0 = 0$.

```
>> x = 0:0.01:pi;
>> f = sin(x);
>> T1 = x;
>> T3 = T1 - x.^3/6;
```

```
>> T5 = T3 + x.^5/120;
>> plot(x,f,'b',x,T1,'g',x,T3,'r',x,T5,'m')
>> ch = get(gca,'Children');
>> set(ch,'LineWidth',2)
>> set(gca,'FontSize',14)
```

Il risultato dell'approssimazione che si ottiene è riportato in Figura 26.

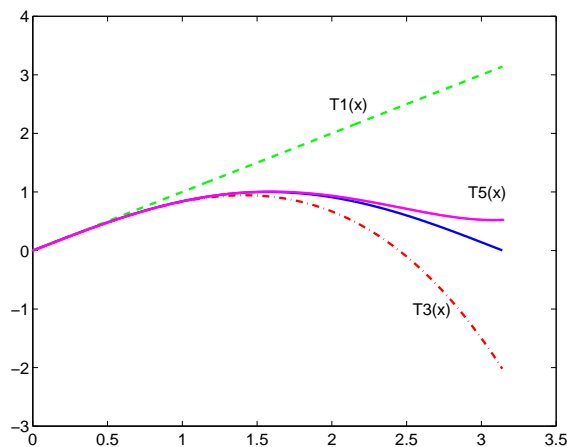


Figura 26: Approssimazione di $\sin(x)$ sull'intervallo $[0, \pi]$ con i primi 3 polinomi di Taylor

Si osservi come le approssimazioni che si ottengono diventino sempre più accurate al crescere dell'ordine del polinomio di Taylor e all'avvicinarsi di x al punto x_0 . Per ogni n fissato, misuriamo l'errore commesso nell'approssimazione come

$$\|f - T_n\|_{\infty, (a,b)} = \max_{x \in (a,b)} |f(x) - T_n(x)|$$

Poiché in MATLAB l'intervallo (a, b) viene rappresentato in modo discreto (cioè da un numero finito di punti), la quantità $\|f - T_n\|_{\infty, (a,b)}$ viene coerentemente sostituita con il comando `norm(f-Tn,inf)` che calcola la norma infinito del vettore $(f-Tn)$.

```
>> e1 = norm(f - T1,inf)
```

```
e1 =
```

```
3.1384
```

```
>> e3 = norm(f - T3,inf)
```

```
e3 =
```

```
2.0214
```

```
>> e5 = norm(f - T5,inf)
```

```
e5 =
```

```
0.5223
```

Come atteso, si osserva una diminuzione progressiva dell'errore. Possiamo rendere più sistematica l'analisi studiando il comportamento dell'errore per valori successivi di n da 0 a 10, per esempio, con la seguente procedura

```
>> x = 0:0.01:pi;
>> f = sin(x);
>> Tn = zeros(size(x));
>> for n = 0:10
    Tn = Tn + (-1)^n*x.^(2*n+1)/prod(1:2*n+1);
    e(n+1) = norm(f - Tn,inf);
    stima(n+1)=(pi^(2*n+3))/(prod(1:2*n+3));
end
>> loglog(0:10,e,'-*')
```

Il grafico ottenuto è riportato in Figura 27.

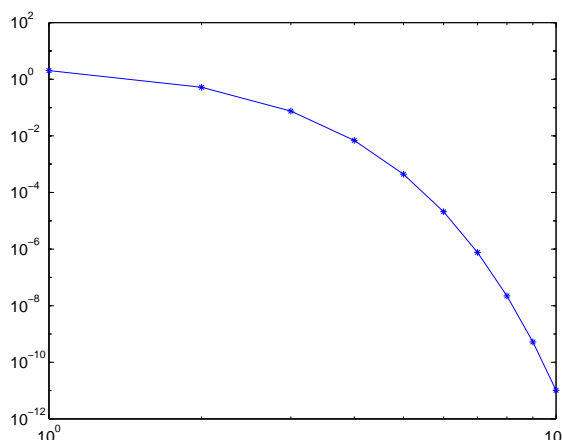


Figura 27: Errore di approssimazione al variare di n

Si noti la rapida convergenza a zero dell'errore che per $n = 10$ vale $1.0228 \cdot 10^{-11}$. È possibile fornire una stima del massimo errore commesso a partire dall'espressione del resto nella forma di Lagrange. Nel caso in esame si ha

$$|R_{n+1}(x)| = |\cos(\xi)| \frac{x^{2n+3}}{(2n+3)!}$$

che può essere maggiorato uniformemente nell'intervallo $[0, \pi]$ come

$$|R_{n+1}(x)| \leq \frac{\pi^{2n+3}}{(2n+3)!}$$

Riportiamo in Figura 28 i grafici dell'errore effettivamente commesso e della corrispondente stima per i valori di n da 0 a 5. Si osservi come, in questo caso, la maggiorazione fornisce una stima accurata dell'errore effettivamente commesso. Ciò dipende soprattutto dalla maggiorazione della funzione coseno con 1 che non è eccessivamente pessimistica.

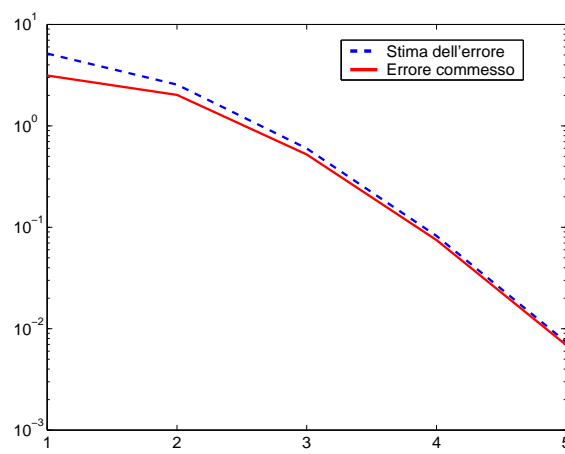


Figura 28: Confronto tra l'errore commesso e la sua maggiorazione

Studiamo l'approssimazione che si ottiene con il polinomio di grado 9 sull'intervallo $[0, 2\pi]$.

```
>> x = 0:0.01:2*pi;
>> f = sin(x);
>> p9 = x-1/6*x.^3+1/120*x.^5-1/5040*x.^7+1/362880*x.^9;
>> plot(x,f,'b',x,p9,'r')
>> set(gca,'FontSize',14)
>> ch = get(gca,'Children');
>> set(ch,'LineWidth',2)
```

Il grafico è mostrato in Figura 29.

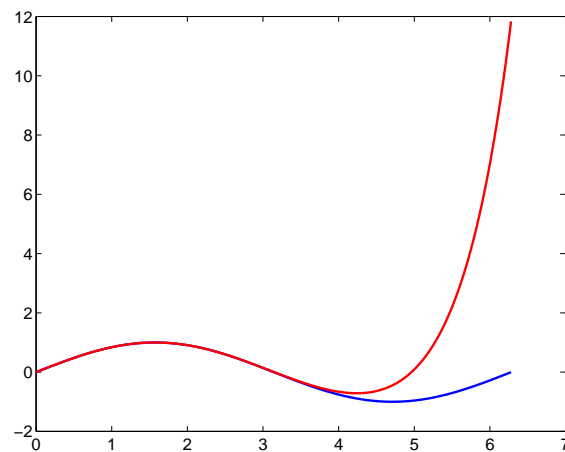


Figura 29: Approssimazione di $\sin(x)$ sull'intervallo $[0, 2\pi]$ con il polinomio di Taylor di grado 9

Si osserva che l'approssimazione è accettabile fino a circa $x = 4$.

6.5 Esempio 5: Approssimazione di una funzione in serie di Fourier

Un *polinomio trigonometrico di ordine n* è una funzione della forma:

$$P_n(x) = \sum_{k=0}^n a_k \cos(kx) + \sum_{k=1}^n b_k \sin(kx)$$

dove a_k e b_k sono assegnati numeri reali o complessi. Si tratta di una funzione regolare (infinitamente derivabile), periodica di periodo 2π (o sottomultiplo di 2π). Si dice *serie trigonometrica* l'analoga espressione dove la somma finita sia sostituita da una serie (cioè k varia da 0 a ∞ anziché da 0 a n):

$$\sum_{k=0}^{\infty} a_k \cos(kx) + \sum_{k=1}^{\infty} b_k \sin(kx)$$

Ricordiamo inoltre la definizione di *serie di Fourier* associata a una funzione f periodica di periodo 2π

$$f(x) \sim \frac{a_0}{2} + \sum_{k=1}^{\infty} a_k \cos(kx) + \sum_{k=1}^{\infty} b_k \sin(kx)$$

ove

$$\begin{cases} a_0 = \frac{1}{2\pi} \int_{-\pi}^{\pi} f(x) dx \\ a_k = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos(kx) dx & k = 1, 2, \dots \\ b_k = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin(kx) dx & k = 1, 2, \dots \end{cases} \quad (7)$$

La serie di Fourier è quindi una particolare serie trigonometrica in cui i coefficienti a_k e b_k sono determinati dalla sola funzione f . Introduciamo la *troncata n -esima* della serie di Fourier di f definita come

$$F_n(x) = \sum_{k=0}^n a_k \cos(kx) + \sum_{k=1}^n b_k \sin(kx)$$

Essa è un polinomio trigonometrico di ordine n i cui coefficienti a_k e b_k , $k = 0, 1, \dots, n$, sono dati dalle relazioni (7) e sono tali da minimizzare lo scarto quadratico medio

$$\int_{-\pi}^{\pi} |f(x) - F_n(x)|^2 dx \quad (8)$$

(si veda, ad esempio, [1]). Studiamo al crescere di n l'approssimazione che si ottiene sostituendo alla serie di Fourier la corrispondente troncata di ordine n . Ad esempio, consideriamo l'*onda quadra*:

$$f(x) = \begin{cases} A, & 0 < x < \pi \\ -A, & -\pi < x < 0, \end{cases} \text{ prolungata periodicamente}$$

dove A è un'assegnata costante. Per definizione si ha

$$a_k = \frac{1}{\pi} \int_{-\pi}^{\pi} A \operatorname{sgn}(x) \cos(kx) dx = 0$$

(integrale di una funzione dispari su un intervallo simmetrico);

$$\begin{aligned} b_k &= \frac{1}{\pi} \int_{-\pi}^{\pi} A \operatorname{sgn}(x) \sin(kx) dx = \frac{2}{\pi} \int_0^{\pi} A \sin(kx) dx \\ &= \frac{2A}{\pi} \left[-\frac{\cos(kx)}{k} \right]_0^{\pi} = \frac{2A}{\pi k} (1 - (-1)^k) = \begin{cases} 0, & k = 2n \\ \frac{4A}{\pi(2n+1)}, & k = 2n+1 \end{cases} \end{aligned}$$

Dunque

$$f(x) \sim \sum_{n=0}^{\infty} \frac{4A}{\pi(2n+1)} \sin(2n+1)x$$

Consideriamo il caso $A = 1$. Tracciamo il grafico di f insieme a quello della troncata n -esima della sua serie di Fourier, per i valori $n = 2$ e $n = 6$.

```
>> x = -2*pi:0.01:2*pi;
>> f1 = stepfun(x,0) - stepfun(x,pi);
>> f2 = stepfun(x+2*pi,0) - stepfun(x+2*pi,pi);
>> f = 2*(f1+f2)-1;
>> s(:,1) = 4/pi*sin(x);
>> for n=1:6
    s(n+1,:) = s(n,:) + 4/(pi*(2*n+1))*sin((2*n+1)*x);
end
>> plot(x,f,x,s([3,7],:))
>> legend('f(x)', 'n = 2', 'n = 6')
>> ch = get(gca,'Children');
>> set(ch,'LineWidth',2)
>> set(gca,'FontSize',14)
```

Il grafico ottenuto è riportato in Figura 30.

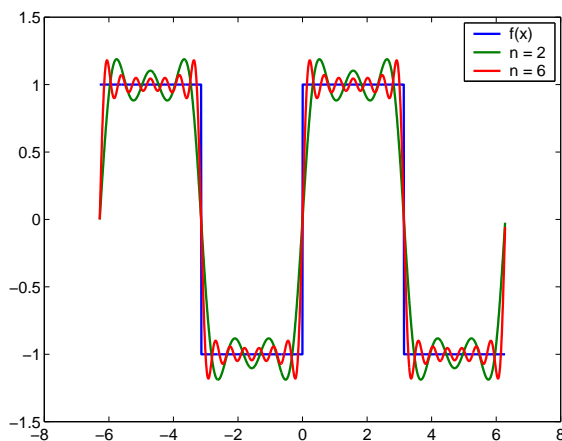


Figura 30: Approssimazione di un'onda quadra con i polinomi trigonometrici di ordine 2 (verde) e 6 (rosso)

Si ricorda che l'approssimazione qui considerata converge nel senso dello scarto quadratico medio, cioè nel senso della relazione *integrale* (8). In questo caso, la convergenza è anche puntuale tranne nei punti di salto $x = k\pi, k \in \mathbb{Z}$ ove la serie di Fourier vale 0.