Università degli Studi di Napoli "Federico II"

Dipartimento di Progettazione Aeronautica

Appunti dalle lezioni di Dinamica del Volo

Integrazione di sistemi di equazioni differenziali con il metodo di Runge–Kutta

---

Ing. **Agostino De Marco**

agodemar@unina.it

Indice

1 Problemi di valori iniziali e integrazione numerica

Sia data una funzione f delle variabili $x, y_1, \ldots, y_M$. In generale sia la f una funzione vettoriale ad M componenti: $f = \{f_1, \ldots, f_M\}^\mathsf{T}$. Un sistema di equazioni differenziali ordinarie del primo ordine, nell'incognita $y = \{y_1, \ldots, y_M\}^\mathsf{T} = y(x)$, funzione della variabile scalare indipendente x, si scrive nella forma:

$$y' = f(x, y) \tag{1}$$

dove l'operatore $(\cdot)' \equiv \mathrm{d}/\mathrm{d}x$ indica la derivazione rispetto alla variabile x.

Una qualsiasi equazione differenziale ordinaria, di qualsiasi ordine, può sempre essere ridotta ad un sistema di equazioni differenziali del primo ordine del tipo (1). Ad esempio l'equazione

$$\frac{\mathrm{d}^2 g}{\mathrm{d}x} + q(x)\frac{\mathrm{d}g}{\mathrm{d}x} = r(x) \tag{2}$$

nell'incognita scalare $g(x)$ può essere riscritta nella forma di sistema:

$$\begin{cases} \dfrac{\mathrm{d}y_1}{\mathrm{d}x} = y_2(x) \\ \dfrac{\mathrm{d}y_2}{\mathrm{d}x} = r(x) - q(x)y_2(x) \end{cases} \tag{3}$$

nella funzione vettoriale incognita $y(x) = \{y_1(x), y_2(x)\}^\mathsf{T}$, dove l'incognita originale $g(x) \equiv y_1(x)$ mentre $y_2(x)$ è una nuova variabile dipendente.

Il generico problema di equazioni differenziali ordinarie viene quindi sempre ricondotto allo studio di un sistema di M equazioni accoppiate del primo ordine per le funzioni y_i, $i = 1, \ldots, M$, avente la forma:

$$\frac{\mathrm{d}y_i}{\mathrm{d}x} = f_i(x, y_1, \ldots, y_M) \qquad i = 1, \ldots, M \tag{4}$$

dove le funzioni a secondo membro sono assegnate. Il sistema (4) insieme con la condizione iniziale nel punto x_0:

$$y(x_0) = y_0 \tag{5}$$

costituisce un *problema di valori iniziali* che ammette una soluzione unica, sotto opportune ipotesi di regolarità della funzione f. Un problema ben posto è anche quello costituito dalla

(4) e dalle condizioni al contorno:

$$\begin{aligned} y_i(a) &= y_{i,A} \qquad i = 1, \dots m \\ y_i(b) &= y_{i,B} \qquad i = m+1, \dots M \end{aligned} \tag{6}$$

dove, ad esempio, alcune condizioni ($m < M$) sono assegnate per un valore a e le rimanenti per un valore b. In tal caso si parla di *problema di valori al contorno* nell'intervallo delle $x \in [a, b]$. In questi appunti si parlerà esclusivamente della risoluzione di un problema di valori iniziali.

I metodi numerici di risoluzione del problema (4)-(5), detti anche metodi di integrazione "al passo", determinano in maniera approssimata i valori della y in corrispondenza di un numero discreto di valori della x:

$$x_{\text{Iniz}} \equiv x_0, x_1, \dots, x_n, x_{n+1}, \dots, x_{\text{Fin}} \tag{7}$$

dove n indica il generico valore discreto ed $h_n = x_{n+1} - x_n$ il generico passo di integrazione. Per semplicità si supporrà in quanto segue che il passo di integrazione sia costante e pari ad $h = (x_{\text{Fin}} - x_{\text{Iniz}})/N$, con N il numero totale di passi di integrazione.

Qualsiasi procedura numerica di integrazione al passo segue la stessa semplice idea di base: i differenziali dy e dx vengono riscritti nelle formule (4) come incrementi finiti Δy e Δx. Moltiplicando le equazioni per Δx si ottengono delle formule algebriche che danno la variazione delle funzioni f_i, cioè della variabile dipendente y, al variare, passo passo, Δx, della variabile indipendente. Nel limite per $\Delta x \to 0$ la soluzione numerica, campionata in un numero sempre maggiore di punti, tenderà alla soluzione esatta del problema differenziale di partenza. L'implementazione di una simile procedura corrisponde al ben noto metodo di integrazione *di Eulero*.

2 Il metodo di Runge–Kutta

La formula di integrazione corrispondente al metodo di Eulero, che fa "avanzare" una soluzione, nota al generico passo n, dal punto x_n a quello successivo $x_{n+1} \equiv x_n + h$ è la seguente:

$$y_{n+1} = y_n + h f(x_n, y_n) \tag{8}$$

La (8) è una tipica formula "non simmetrica", nel senso che sfrutta informazioni, le derivate $f(x_n, y_n)$, valutate solo al primo estremo dell'intervallo $[x_n, x_{n+1}]$. Ciò rende la "predizione" del valore y_{n+1} accurata a meno di un termine di correzione di $O(h^2)$. La formula di Eulero (8) corrisponde ad un metodo del primo ordine.[1]

Il metodo di Eulero *non* è raccomandabile per un uso pratico per due motivi principali: (*i*) esso non è accurato, a parità di passo di integrazione, quanto altri metodi di altrettanto semplice implementazione, come il metodo di Runge–Kutta di cui si parlerà qui di seguito; (*ii*) il metodo di Eulero può risultare spesso instabile quando la funzione f è sensibilmente variabile.

Si consideri comunque una formula del tipo (8) al fine di ottenere un "incremento di prova" k_1 della y, a partire dai valori x_n ed y_n. Si otterrà un valore $y_n + k_1$ in corrispondenza dell'estremo x_{n+1}. Infine si applichi la stessa formula di integrazione ma valutando stavolta la funzione f in corrispondenza dei valori intermedi $x_n + \frac{1}{2}h$ ed $y_n + \frac{1}{2}k_1$. Si otterrà un nuovo incremento, k_2, che potrà essere considerato un valore più accurato di k_1. Si può far vedere [2] che k_2 è un incremento approssimato a meno di un termine di $O(h^3)$:

$$\begin{aligned} k_1 &= hf(x_n, y_n) \\ k_2 &= hf\left(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_1\right) \\ y_{n+1} &= y_n + k_2 + O(h^3) \end{aligned} \tag{9}$$

cioè che la procedura appena descritta, nota come *midpoint method* o metodo di *Runge–Kutta* del *secondo ordine*, è un metodo del secondo ordine, cfr fig. 1(b). Combinando due formule del primo ordine, cioè costruendo un valore definitivo della y_{n+1} in *due stadi* come somma

$$y_{n+1} = y_n + 0 \cdot k_2 + 1 \cdot k_2 \tag{10}$$

"valutando" differentemente la funzione f in ciascuno stadio, si ottiene quindi la cancellazione dal termine di errore del contributo del secondo ordine (h^2).

Esistono molti modi di valutare il secondo membro $f(x, y)$ all'interno dell'intervallo $[x_n, x_{n+1}]$ secondo formule del primo ordine come la (8) e di combinarle in più stadi N_s. È

[1] Per convenzione un metodo si dice di ordine r se il termine di correzione $E = y_{n+1} - y_n - k$ è di $O(h^{r+1})$, dove k è pari al prodotto di h per la f valutata nel modo proposto dallo schema in questione.

possibile dimostrare che, al crescere del numero degli stadi, con opportune combinazioni si possono via via eliminare dall'errore di approssimazione

$$E = y_{n+1} - y_n - \sum_{s=1}^{N_s} c_s k_s \qquad \left(\text{dove} \quad \sum_{s=1}^{N_s} c_s = 1 \right) \tag{11}$$

i contributi di $O(h^2)$, $O(h^3)$, $O(h^4)$ ecc. Questa è l'idea in base alla quale è stato sviluppato il metodo di Runge–Kutta.

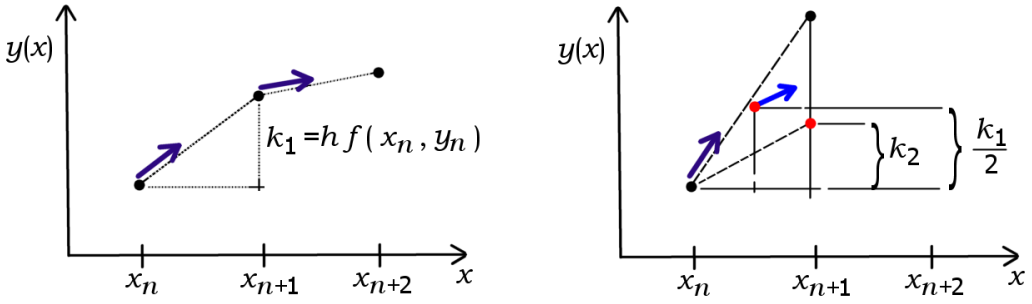
La versione più diffusa di questo metodo di integrazione al passo corrisponde alla cosiddetta *formula di Runge–Kutta* del *quarto ordine*. Tale procedura viene scritta classicamente nella forma:

$$\begin{aligned} k_1 &= h f\left(x_n, y_n\right) \\ k_2 &= h f\left(x_n + \frac{h}{2}, y_n + \frac{k_1}{2}\right) \\ k_3 &= h f\left(x_n + \frac{h}{2}, y_n + \frac{k_2}{2}\right) \\ k_4 &= h f\left(x_n + h, y_n + k_3\right) \\ y_{n+1} &= y_n + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6} + O(h^5) \end{aligned} \tag{12}$$

Il metodo di Runge–Kutta del quarto ordine, noto anche come *4-stage stepping scheme*, ($N_s = 4$, $c_1 = c_4 = \frac{1}{6}$, $c_2 = c_3 = \frac{1}{3}$, cfr. (11)), richiede quattro valutazioni della funzione f per ciascun passo h, cfr. fig. 2. Una tale procedura è nella grande maggioranza dei casi superiore alla corrispondente formula del secondo ordine (9) *se*, a parità di accuratezza, si riesce ad mantenere un passo doppio. Si ricorda al lettore che quest'ultima affermazione è vera dal punto di vista della pratica ingegneristica, non dal punto di vista strettamente matematico, cfr. [1].

3 Listato del codice `rkdrive.for`

Come esempio di applicazione del metodo di integrazione di Runge–Kutta viene proposto in questo paragrafo il listato del codice di calcolo `rkdrive.for`, scritto in FORTRAN 77. Il



(a) Formula di Eulero

(b) Formula di Runge–Kutta del secondo ordine

Figura 1: Metodo di Eulero (a). Con la derivata al punto iniziale di ogni intervallo si estrapola direttamente il valore successivo della funzione incognita. *Midpoint method* (b). Con la derivata iniziale si ottiene un punto intermedio. In esso si valuta la derivata che estrapola il valore successivo a partire da quello iniziale.

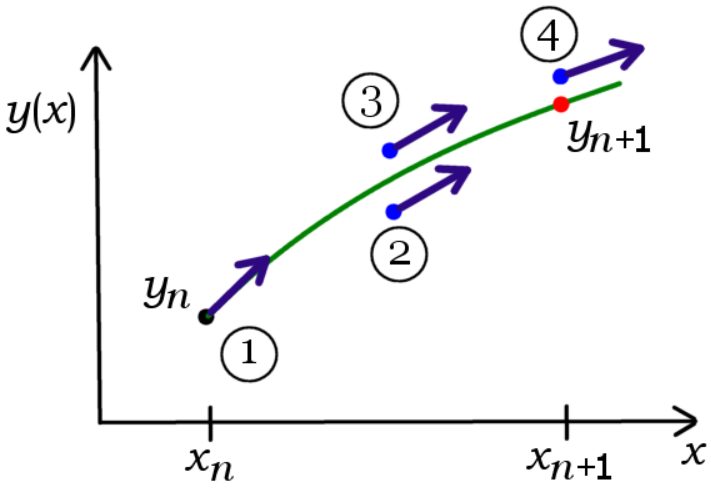


Figura 2: Metodo di Runge–Kutta del quarto ordine. La derivata è valutata quattro volte, una volta al punto iniziale (1), due volte in due punti intermedi di prova (2) e (3) ed infine al punto di prova finale (4). Dalla combinazione di tali valori si ottiene il valore successivo dell'incognita y.

problema di valori iniziali risolto con questo programma è il seguente:

$$\begin{aligned} \frac{\mathrm{d}^2 y}{\mathrm{d}x^2} + 3\cos^2 x - 2 = 0 \\ y(0) = \frac{\mathrm{d}y}{\mathrm{d}x}(0) = 0 \end{aligned} \tag{13}$$

Si può facilmente verificare che la soluzione analitica del problema (13) è costituita dalla funzione:

$$y(x) = \frac{1}{4}x^2 + \frac{3}{8}\cos(2x) - \frac{3}{8} \tag{14}$$

Nel codice `rkdrive` viene fissato un numero di passi di integrazione pari ad `NSTEP` ed un valore finale della variabile indipendente (`x2`) pari a `6.28`. La *subroutine* che implementa la formula di Runge–Kutta è denominata `rk4` mentre la *subroutine* che implementa l'algoritmo di integrazione al passo è denominata `rkdumb`. La *subroutine* `derivs`, il cui nome compare tra gli argomenti di `rkdumb`, implementa la posizione:

$$\begin{cases} \dfrac{\mathrm{d}y_1}{\mathrm{d}x} = f_1(x, y_1, y_2) = y_2 \\ \dfrac{\mathrm{d}y_2}{\mathrm{d}x} = f_2(x, y_1, y_2) = -3\cos^2 x + 2 \end{cases} \tag{15}$$

Il grafico della soluzione esatta e della soluzione numerica del problema proposto è riportato infine in fig. 3.

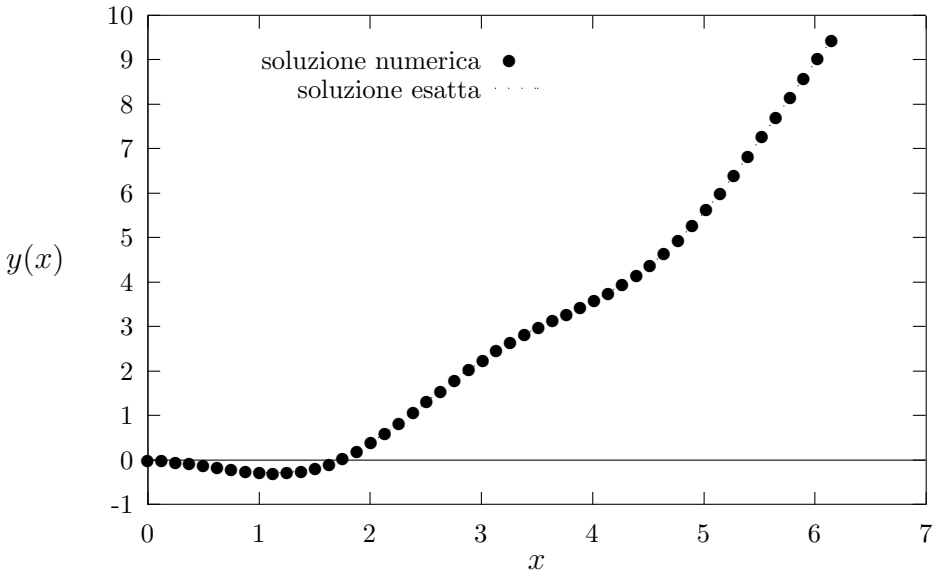


Figura 3: Soluzione numerica del problema di valori iniziali (13) nell'intervallo $[0, 2\pi]$ con metodo di Runge–Kutta del quarto ordine.

Listato 1: Il *file* sorgente "`rkdrive.for`".

```
!---------------------------------------------------------------------
! Programma: rkdrive
! Autore: Agostino De Marco
! Ref.: Numerical Recipes in Fortran
! Task: Driver per la routine rkdumb
!---------------------------------------------------------------------
      PROGRAM rkdrive
      INTEGER NSTEP,NVAR
      PARAMETER(NVAR=2,NSTEP=50)
      PARAMETER (NMAX=50,NSTPMX=200)
      INTEGER i,j
      REAL x(NSTPMX),x1,x2,y(NMAX,NSTPMX),vstart(NVAR)
      COMMON /path/ x,y
      EXTERNAL derivs
      x1=0.0
      vstart(1)=0.    <-- y1(0) = 0
      vstart(2)=0.    <-- y2(0) = 0
      x2=6.28
      CALL rkdumb(vstart,NVAR,x1,x2,NSTEP,derivs)
```

```
20
21 !-----------------------------------------------------------------
22 ! Stampa i risultati a video
23
24       WRITE(*,'(/1x,t9,a,t17,a,t31,a/)') 'X','Integrated','Exact'
25       DO 11 i=1,(NSTEP/10)
26         j=10*i
27         WRITE(*,'(1x,f10.4,2x,2f12.6)') x(j),y(1,j),val(x(j))
28 11    CONTINUE
29
30 !-----------------------------------------------------------------
31 ! Salva i risultati su file
32
33       OPEN(8,FILE="output.txt")
34       DO 111 j=1,NSTEP
35         WRITE(8,'(1x,f10.4,2x,2f12.6)') x(j),y(1,j),val(x(j))
36 111   CONTINUE
37
38       END
39
40 !-----------------------------------------------------------------
41 ! Definisce la soluzione esatta: y(x) = x^4 + 3 cos^2 x + 2
42
43       REAL FUNCTION val(t)
44       REAL t
45       val=t*t/4. + 3.*cos(2.*t)/8.-3./8.
46       END FUNCTION
47
48 !-----------------------------------------------------------------
49 ! Definisce le derivate f_i(x, y_1, y_2)
50
51       SUBROUTINE derivs(x,y,dydx)
52       REAL x,y(*),dydx(*)
53       dydx(1)=y(2)                    <-- y_1' = y_2
54       dydx(2)=-3.*cos(x)*cos(x)+2.    <-- y_2' = -3 cos^2 x + 2
55       RETURN
56       END
57 !-----------------------------------------------------------------
58 ! Implementa l'algoritmo di integrazione al passo
59
60       SUBROUTINE rkdumb(vstart,nvar,x1,x2,nstep,derivs)
```

```
61      INTEGER nstep,nvar,NMAX,NSTPMX
62      PARAMETER (NMAX=50,NSTPMX=200)
63      REAL x1,x2,vstart(nvar),xx(NSTPMX),y(NMAX,NSTPMX)
64      EXTERNAL derivs
65      COMMON /path/ xx,y
66  !U  USES rk4
67      INTEGER i,k
68      REAL h,x,dv(NMAX),v(NMAX)
69      DO 11 i=1,nvar
70        v(i)=vstart(i)
71        y(i,1)=v(i)      ⟵ assegna y(x_0) = y_0
72  11  CONTINUE
73      xx(1)=x1
74      x=x1
75      h=(x2-x1)/nstep      ⟵ calcola il passo h
76      DO 13 k=1,nstep          (k≡ indice del generico passo di integrazione n)
77        CALL derivs(x,v,dv)     ⟵ valuta la derivata dv= f(x_n, y_n)
78        CALL rk4(v,dv,nvar,x,h,v,derivs) ⟵ v è la soluzione al passo successivo
79        IF(x+h.eq.x)pause 'stepsize not significant in rkdumb'
80        x=x+h
81        xx(k+1)=x      ⟵ x_{n+1} = x_n + h
82        DO 12 i=1,nvar
83          y(i,k+1)=v(i)      ⟵ y_{n+1} =v
84  12      CONTINUE
85  13  CONTINUE
86      RETURN
87      END
88
89  !-------------------------------------------------------------------
90  ! Implementa la formula di Runge-Kutta a quattro stadi
91
92      SUBROUTINE rk4(y,dydx,n,x,h,yout,derivs)
93      INTEGER n,NMAX
94      REAL h,x,dydx(n),y(n),yout(n)
95      EXTERNAL derivs
96      PARAMETER (NMAX=50)
97      INTEGER i
98      REAL h6,hh,xh,dym(NMAX),dyt(NMAX),yt(NMAX)
99      hh=h*0.5
100     h6=h/6.
101     xh=x+hh      ⟵ x_n + h/2
```

```
      DO 11 i=1,n
        yt(i)=y(i)+hh*dydx(i)        ⟵ dyt = y_n + k1/2
11    CONTINUE
      CALL derivs(xh,yt,dyt)         ⟵ valuta la derivata dyt= f(x_n + h/2, y_n + k1/2)
      DO 12 i=1,n
        yt(i)=y(i)+hh*dyt(i)         ⟵ y_n + k2/2
12    CONTINUE
      CALL derivs(xh,yt,dym)         ⟵ valuta la derivata dym= f(x_n + h/2, y_n + k2/2)
      DO 13 i=1,n
        yt(i)=y(i)+h*dym(i)          ⟵ dyt = y_n + k3
        dym(i)=dyt(i)+dym(i)         ⟵ dym= (k2 + k3)/h
13    CONTINUE
      CALL derivs(x+h,yt,dyt)        ⟵ valuta la derivata dyt= f(x_n + h, y_n + k3)
      DO 14 i=1,n
        yout(i)=y(i)+h6*(dydx(i)+dyt(i)+2.*dym(i)) ⟵ y_n + h/6 (k1/h + k4/h + 2(k2+k3)/h)
14    CONTINUE
      RETURN
      END
```

Riferimenti bibliografici

[1] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, *Numerical Recipes in Fortran: The Art of Scientific Computing*, Cambridge University Press, 1992.

[2] G. Dahlquist, A. Bjorck, *Numerical Methods*, Prentice-Hall, 1974.