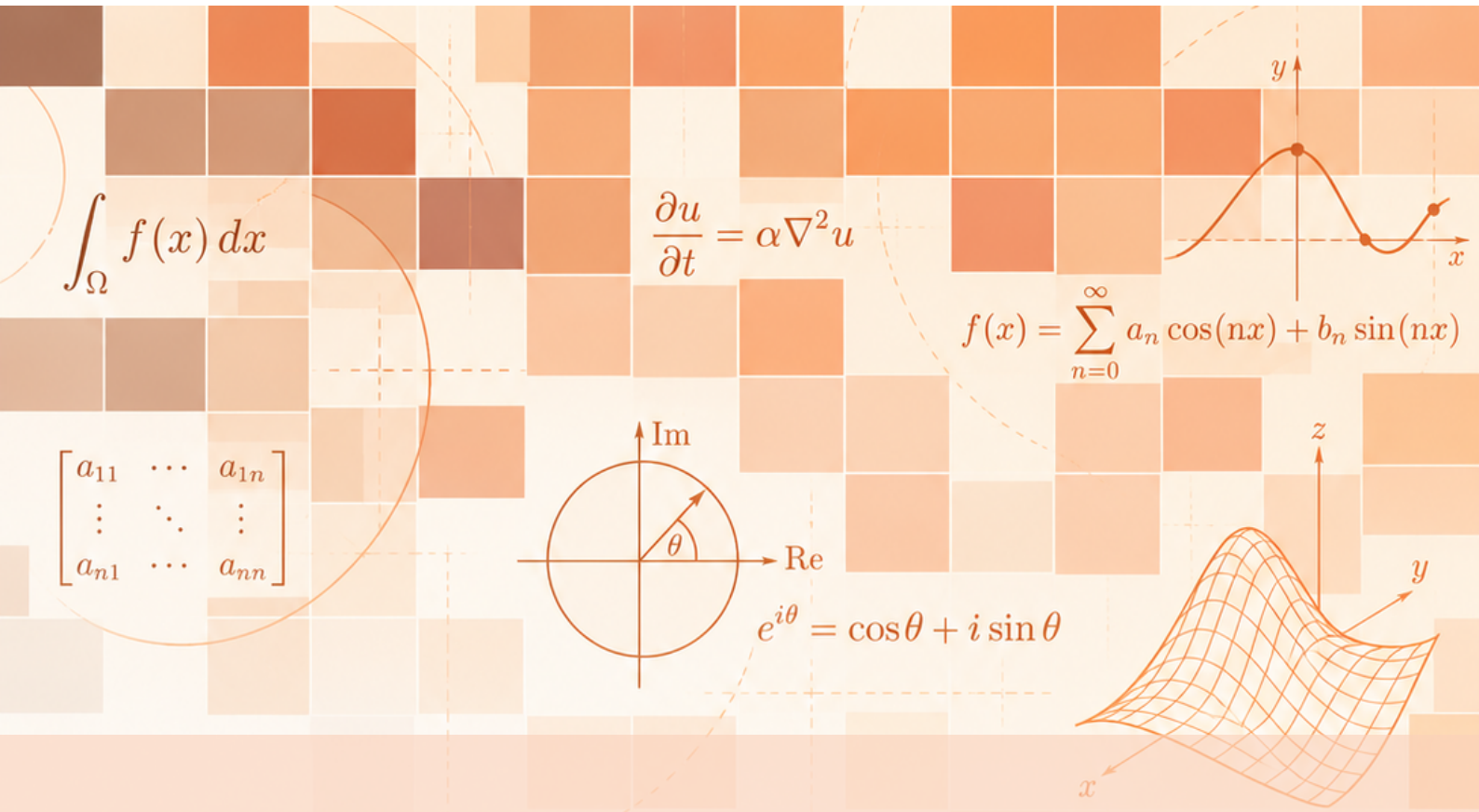
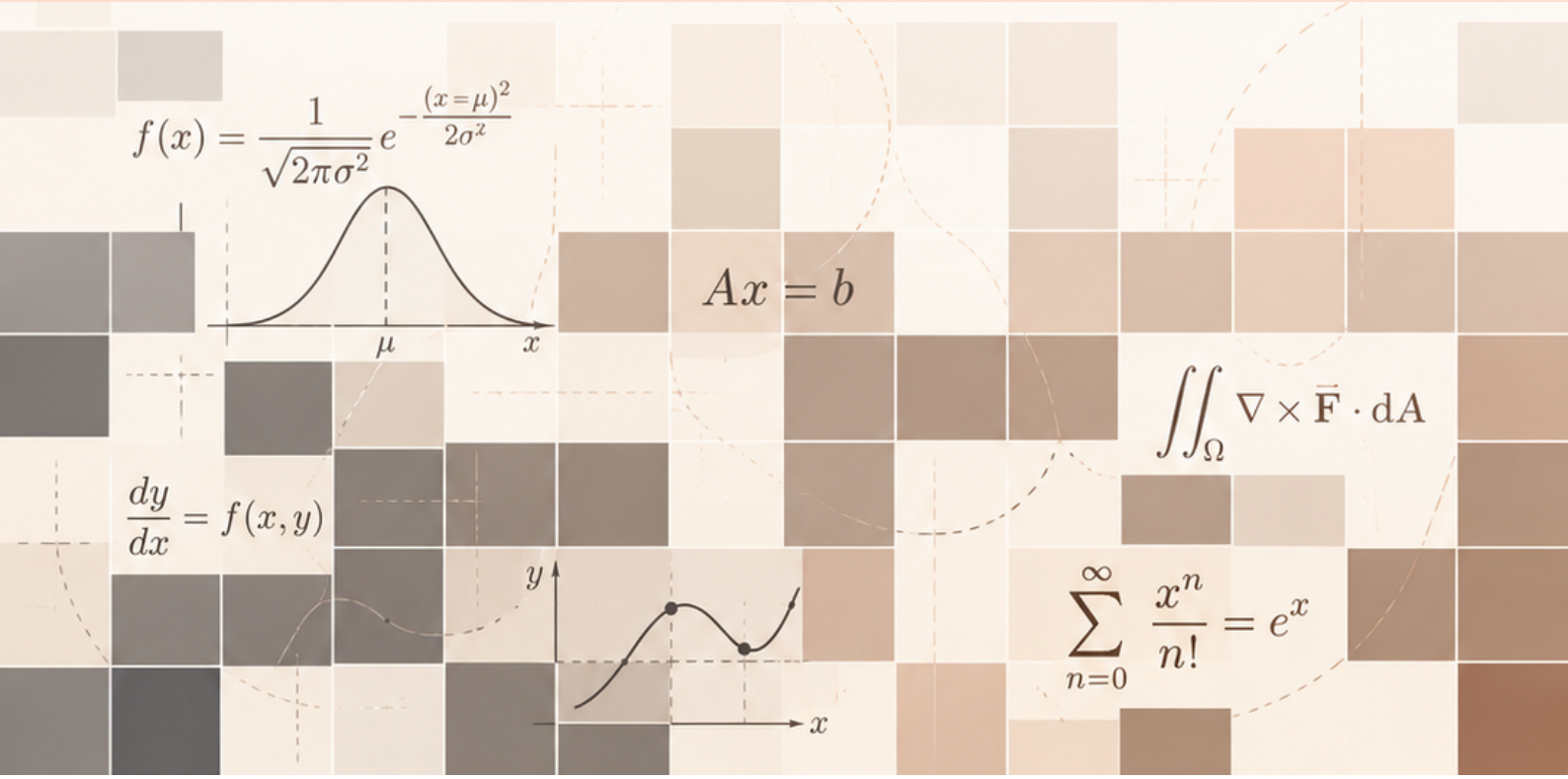


Analisi dei dati per l'ingegneria (146172) (module 1)

Esercizi Svolti

Enrico Bertolazzi



Copyright © 2025 Enrico Bertolazzi

Published by Publisher

book-website.com

Licensed under the Creative Commons Attribution-NonCommercial 4.0 License (the "License"). You may not use this file except in compliance with the License. You may obtain a copy of the License at https://creativecommons.org/licenses/by-nc-sa/4.0. Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "as is" basis, without warranties or conditions of any kind, either express or implied. See the License for the specific language governing permissions and limitations under the License.

First printing, May 2025



Contents

1 Esempio di fattorizzazione LU 7
1.1 Step 1 scambio righe 7
1.2 Step 1b applica eliminazione 7
1.3 Step 2 nessuno scambio 8
1.4 Step 2b applica eliminazione 8
1.5 Step 3 scambio righe 8
1.6 Step 3b eliminazione 8

2 Scelta a priori del numero di sottointervalli 9
2.1 Esercizio 1 — Trapezi compositi per una funzione esponenziale 9
2.2 Esercizio 2 — Simpson composito per una funzione trigonometrica 9
2.3 Esercizio 3 — Trapezi compositi per una funzione razionale 10
2.4 Esercizio 4 — Simpson composito per un polinomio 10

3 Zeri di funzione (esempi) 11
3.1 Calcolo della radice cubica 12
3.1.1 Soluzione 12
3.1.2 Grafico della funzione 12
3.2 Equazione di radiazione di Stefan-Boltzmann (Trasferimento di calore) 15
3.3 Equazione di Colebrook (Flusso di fluidi) 17
3.4 Equazione di Van der Waals (Equilibrio chimico) 19
3.5 Equazione di Shockley per i diodi 21
3.6 Problemi difficili: quando serve la salvaguardia 22
3.6.1 Un solutore ibrido minimo 23
3.7 Equazione di Keplero: orbite quasi paraboliche 23
3.8 Frequenze proprie di una trave a sbalzo 25
3.9 Profondità di immersione di una sfera 26
3.10 Catenaria: la freccia di una linea aerea 27

3.11 Una funzione piatta: il problema 13 del banco TOMS 748 28
3.12 Il problema 10 del banco TOMS 748 29
3.13 Tassi interni di rendimento multipli 30
3.14 Tre patologie del metodo di Newton 31
3.14.1 Arcotangente: converge o diverge secondo il punto iniziale 31
3.14.2 Un ciclo che non finisce mai 33
3.14.3 Fuga garantita 34
3.14.4 Un quadro unico 35

4 Criteri per controllo ordine 39
4.1 Euler Esplicito 39
4.2 Metodo del Punto Medio 40
4.3 Metodo di Ralston 40
4.4 Runge-Kutta del terzo ordine (RK3) 41
4.5 Runge-Kutta del quarto ordine: regola 3/8 43
4.6 Metodo del Punto Medio Implicito 45
4.7 Metodo di Crank-Nicolson 46
4.8 Gauss-Legendre del quarto ordine 46
4.9 Lobatto IIIB del quarto ordine 47

References 49



Indice

Esercizi ed esempi svolti per esteso, raccolti a parte perché non servono a seguire la spiegazione: le note del corso si leggono senza, e questi si leggono dopo.

- Esempio di fattorizzazione LU
- Scelta a priori del numero di sottointervalli
- Zeri di funzione: esempi
- Criteri per controllo ordine
- **Bibliografia**

Gli esercizi proposti al termine dei capitoli, e gli esempi che servono alla spiegazione, sono rimasti nelle note accanto alla teoria che verificano.



1. Esempio di fattorizzazione LU

Il metodo di eliminazione di Gauss[1], applicato a un sistema lineare della forma:

$$\begin{cases} x_2 + 3x_3 + 4x_4 &= 6 \\ 2x_1 + 2x_2 + 4x_4 &= 10 \\ x_2 + 3x_3 + x_4 &= 3 \\ x_1 + 3x_3 - x_4 &= 0 \end{cases}$$

riscritto nella forma $\boldsymbol{Ax} = \boldsymbol{b}$

$$\boldsymbol{A} = \begin{pmatrix} 0 & 1 & 3 & 4 \\ 2 & 2 & 0 & 4 \\ 0 & 1 & 3 & 1 \\ 1 & 0 & 3 & -1 \end{pmatrix}, \quad \boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}, \quad \boldsymbol{b} = \begin{pmatrix} 6 \\ 10 \\ 3 \\ 0 \end{pmatrix},$$

Alloco il sistema lineare in R

```
The 4x4 matrix:
```

Poiché la soluzione è già nota, verifichiamo il residuo:
Inizio l'algoritmo di Gauss

1.1 Step 1 scambio righe

$$\boldsymbol{S}_1\boldsymbol{A} = \begin{bmatrix} 2 & 2 & 0 & 4 \\ 0 & 1 & 3 & 4 \\ 0 & 1 & 3 & 1 \\ 1 & 0 & 3 & -1 \end{bmatrix}$$

Faccio lo stesso lavoro sulla tabella `TABLE`

1.2 Step 1b applica eliminazione

$$\boldsymbol{A}_1 = \boldsymbol{L}_1\boldsymbol{S}_1\boldsymbol{A} = \begin{bmatrix} 2 & 2 & 0 & 4 \\ 0 & 1 & 3 & 4 \\ 0 & 1 & 3 & 1 \\ 0 & -1 & 3 & -3 \end{bmatrix}$$

[1] Carl Friedrich Gauss (1777-1855)

Applico eliminazione sulla tabella TABLE

1.3 Step 2 nessuno scambio

L'elemento $A_{2,2} \neq 0$ quindi non serve nessuno scambio di righe.

1.4 Step 2b applica eliminazione

$$\boldsymbol{A}_2 = \boldsymbol{L}_2\boldsymbol{S}_2\boldsymbol{L}_1\boldsymbol{S}_1\boldsymbol{A} = \begin{bmatrix} 2 & 2 & 0 & 4 \\ 0 & 1 & 3 & 4 \\ 0 & 0 & 0 & -3 \\ 0 & 0 & 6 & 1 \end{bmatrix}$$

Applico eliminazione sulla tabella TABLE

1.5 Step 3 scambio righe

$$\boldsymbol{A}_3 = \boldsymbol{S}_3\boldsymbol{L}_2\boldsymbol{S}_2\boldsymbol{L}_1\boldsymbol{S}_1\boldsymbol{A} = \begin{bmatrix} 2 & 2 & 0 & 4 \\ 0 & 1 & 3 & 4 \\ 0 & 0 & 6 & 1 \\ 0 & 0 & 0 & -3 \end{bmatrix}$$

Applico scambio sulla tabella TABLE

1.6 Step 3b eliminazione

Matrice già in forma triangolare $\boldsymbol{L}_3 = \boldsymbol{I}$ non serve fare niente.

$$\boldsymbol{A}_3 = \boldsymbol{L}_3\boldsymbol{S}_3\boldsymbol{L}_2\boldsymbol{S}_2\boldsymbol{L}_1\boldsymbol{S}_1\boldsymbol{A} = \begin{bmatrix} 2 & 2 & 0 & 4 \\ 0 & 1 & 3 & 4 \\ 0 & 0 & 6 & 1 \\ 0 & 0 & 0 & -3 \end{bmatrix}$$

Applico eliminazione sulla tabella TABLE
Abbiamo quindi ottenuto

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\boldsymbol{L}_3} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\boldsymbol{S}_3} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}}_{\boldsymbol{L}_2} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\boldsymbol{S}_2}$$

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -0.5 & 0 & 0 & 1 \end{bmatrix}}_{\boldsymbol{L}_1} \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\boldsymbol{S}_1} \underbrace{\begin{bmatrix} 0 & 1 & 3 & 4 \\ 2 & 2 & 0 & 4 \\ 0 & 1 & 3 & 1 \\ 1 & 0 & 3 & -1 \end{bmatrix}}_{\boldsymbol{A}} = \underbrace{\begin{bmatrix} 2 & 2 & 0 & 4 \\ 0 & 1 & 3 & 4 \\ 0 & 0 & 6 & 1 \\ 0 & 0 & 0 & -3 \end{bmatrix}}_{\boldsymbol{A}_3}$$



2. Scelta a priori del numero di sottointervalli

Le formule di errore delle regole composite permettono di dimensionare la discretizzazione prima di calcolare l'integrale: fissata una tolleranza tol, si cerca il più piccolo N per cui la maggiorazione teorica non la supera. Le formule usate qui, con $L = b - a$ e $H = L/N$, sono

$$N_T = \left\lceil \sqrt{\frac{L^3 M_2}{12\,\mathrm{tol}}} \right\rceil, \qquad N_S = \left\lceil \left(\frac{L^5 M_4}{2880\,\mathrm{tol}} \right)^{1/4} \right\rceil,$$

e sono ricavate nelle note del corso, dove si trova anche un esercizio che confronta fra loro trapezi, Simpson e Gauss-Legendre.

2.1 Esercizio 1 — Trapezi compositi per una funzione esponenziale

Stimare il minimo numero di sottointervalli uguali per approssimare

$$I = \int_0^1 e^x \, \mathrm{d}x, \qquad \mathrm{tol} = 10^{-3},$$

con la regola dei trapezi composita. Poiché $f''(x) = e^x$, possiamo prendere $M_2 = e$ e $L = 1$. Segue

$$N \geq \sqrt{\frac{e}{12 \cdot 10^{-3}}} \simeq 15.051, \qquad \boxed{N_T = 16}, \qquad H = \frac{1}{16}.$$

La maggiorazione verificabile a priori è $|E_T| \leq e/(12 \cdot 16^2) \simeq 8.85 \cdot 10^{-4} < \mathrm{tol}$.

2.2 Esercizio 2 — Simpson composito per una funzione trigonometrica

Stimare il minimo numero di intervalli di Simpson per

$$I = \int_0^{\pi} \sin x \, \mathrm{d}x, \qquad \mathrm{tol} = 10^{-6}.$$

Dato che $f^{(4)}(x) = \sin x$, risulta $M_4 = 1$ e $L = \pi$. Quindi

$$N \geq \left(\frac{\pi^5}{2880 \cdot 10^{-6}} \right)^{1/4} \simeq 18.055, \qquad \boxed{N_S = 19}.$$

Occorrono dunque **19 intervalli**, ossia **38 sottointervalli** di lunghezza $h = \pi/38$, con 39 nodi distinti. La stima fornisce $|E_S| \le \pi^5/(2880 \cdot 19^4) < 10^{-6}$.

2.3 Esercizio 3 — Trapezi compositi per una funzione razionale

Determinare il numero di sottointervalli per

$$I = \int_{-1}^{1} \frac{1}{1+x^2}\,dx, \qquad \text{tol} = 5 \cdot 10^{-4}.$$

Si calcola

$$f''(x) = \frac{6x^2 - 2}{(1+x^2)^3}.$$

Su $[-1, 1]$ vale $|f''(x)| \le 2$ (il massimo è raggiunto in $x = 0$), dunque $M_2 = 2$ e $L = 2$. Ne segue

$$N \ge \sqrt{\frac{2^3 \cdot 2}{12 \cdot 5 \cdot 10^{-4}}} \simeq 51.640, \qquad \boxed{N_T = 52}, \qquad H = \frac{2}{52}.$$

In particolare $|E_T| \le 16/(12 \cdot 52^2) < 5 \cdot 10^{-4}$.

2.4 Esercizio 4 — Simpson composito per un polinomio

Determinare il numero di intervalli di Simpson per

$$I = \int_{0}^{2} x^4\,dx, \qquad \text{tol} = 10^{-5}.$$

La derivata quarta è costante: $f^{(4)}(x) = 24$, quindi $M_4 = 24$ e $L = 2$. Otteniamo

$$N \ge \left(\frac{2^5 \cdot 24}{2880 \cdot 10^{-5}}\right)^{1/4} \simeq 12.779, \qquad \boxed{N_S = 13}.$$

Sono necessari **13 intervalli** (26 sottointervalli, 27 nodi). La stima garantisce $|E_S| \le 768/(2880 \cdot 13^4) < 10^{-5}$. Poiché la derivata quarta è costante, in questo esempio la formula dell'errore di Simpson è anche esatta in valore assoluto.

> **Interpretazione del risultato**
>
> La procedura fornisce una **garanzia a priori** soltanto se M_2 o M_4 sono maggiorazioni valide su tutto $[a, b]$. Un valore campionato numericamente della derivata non costituisce, da solo, una maggiorazione rigorosa. Inoltre, per una stessa tolleranza, il numero di intervalli richiesto può essere molto diverso tra trapezi e Simpson: le rispettive stime decrescono come N^{-2} e N^{-4}.



3. Zeri di funzione (esempi)

Alcuni esempi di problemi che si risolvono trovando le radici di $f(x) = 0$.

Solutori in R

Metodo di bisezione per trovare la radice

```r
bisection_method <- function(func, lower, upper, tol = 1e-6, max_iter = 100) {
  for (i in 1:max_iter) {
    cat(sprintf("Intervallo: lower = %g, upper = %g\n", lower, upper))
    midpoint <- (lower + upper) / 2
    if (func(midpoint) == 0 || (upper - lower) / 2 < tol) {
      return(midpoint)  # Restituisce il punto medio se è la radice
    }
    if (sign(func(midpoint)) == sign(func(lower))) {
      lower <- midpoint  # Aggiorna l'intervallo inferiore
    } else {
      upper <- midpoint  # Aggiorna l'intervallo superiore
    }
  }
  return(NULL)  # Se non si trova la radice
}
```

Metodo delle secanti per trovare la radice

```r
secant_method <- function(func, x0, x1, tol = 1e-6, max_iter = 100) {
  for (i in 1:max_iter) {
    if (abs(x1 - x0) < tol) {
      return(x1)  # Restituisce la soluzione se la differenza è inferiore alla
          ↪  tolleranza
    }
    x2 <- x1 - func(x1) * (x1 - x0) / (func(x1) - func(x0))  # Calcola il nuovo punto
    cat(sprintf("Iterazione %d: x = %g\n", i, x2))
    x0 <- x1
    x1 <- x2
  }
  warning("La radice non è stata trovata dopo il numero massimo di iterazioni.")
  return(NULL)  # Se non si trova la radice
}
```

Metodo di Newton-Raphson[1] per trovare la radice

```r
newton_method <- function(func, func_prime, initial_guess, tol = 1e-6, max_iter = 100) {
  x <- initial_guess
  for (i in 1:max_iter) {
    x_new <- x - func(x) / func_prime(x)  # Calcola il nuovo punto
    cat(sprintf("Iterazione %d: x = %g\n", i, x_new))
    if (abs(x_new - x) < tol) {
      return(x_new)  # Restituisce la soluzione se la differenza è inferiore alla
          ↪ tolleranza
    }
    x <- x_new
  }
  warning("La radice non è stata trovata dopo il numero massimo di iterazioni.")
  return(NULL)  # Se non si trova la radice
}
```

3.1 Calcolo della radice cubica

Il calcolo delle radici cubiche è essenziale in ingegneria e in scienze applicate, poiché molte grandezze fisiche e ingegneristiche, come il volume e la densità, richiedono la determinazione di dimensioni lineari a partire da quantità cubiche. In questo documento, analizzeremo il problema di trovare la radice cubica di un numero positivo a risolvendo l'equazione:

$$f(x) = x^3 - a = 0$$

Dove a rappresenta il volume di un cubo, e vogliamo determinare la lunghezza del lato x.

3.1.1 Soluzione

Consideriamo il problema di trovare la radice cubica di un numero $a = 27$. L'equazione che dobbiamo risolvere è:

$$f(x) = x^3 - 27 = 0$$

Stiamo cercando il valore di x tale che $x^3 = 27$.

3.1.2 Grafico della funzione

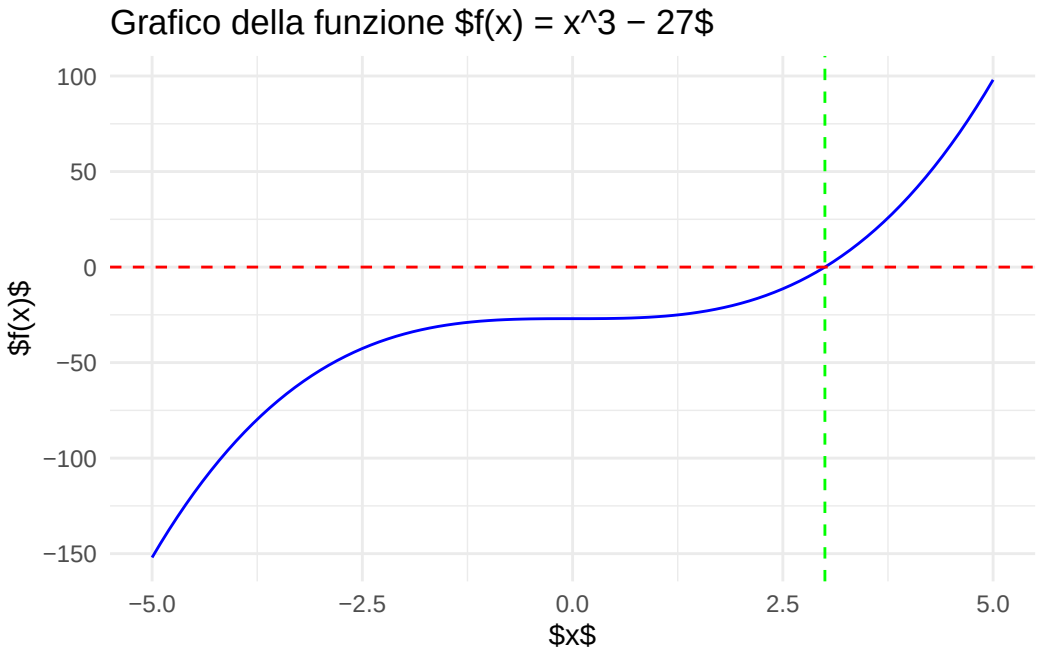
Possiamo visualizzare il grafico della funzione per identificare la posizione della radice:

```r
# Grafico della funzione
library(ggplot2)

# Creare un vettore di valori x
x_values <- seq(-5, 5, by = 0.1)
f_values <- x_values^3 - 27

# Creazione del grafico
ggplot(data = data.frame(x = x_values, f = f_values), aes(x = x, y = f)) +
  geom_line(color = "blue") +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
  geom_vline(xintercept = 3, linetype = "dashed", color = "green") +
  labs(title = "Grafico della funzione $f(x) = x^3 - 27$",
       x = "$x$",
       y = "$f(x)$") +
  theme_minimal()
```

[1] Isaac Newton (1642-1727) e Joseph Raphson (1668 ca.-1715 ca.)



Soluzione in R

Definizione della funzione per il calcolo della radice cubica

```
radice_cubica <- function(x) {
  return(x*x*x-27)
}
```

Metodo di Bisezione

Trova la radice cubica di 27 usando il metodo di bisezione

```
sol <- bisection_method(radice_cubica, 0, 27)
```

```
Intervallo: lower = 0, upper = 27
Intervallo: lower = 0, upper = 13.5
Intervallo: lower = 0, upper = 6.75
Intervallo: lower = 0, upper = 3.375
Intervallo: lower = 1.6875, upper = 3.375
Intervallo: lower = 2.53125, upper = 3.375
Intervallo: lower = 2.95312, upper = 3.375
Intervallo: lower = 2.95312, upper = 3.16406
Intervallo: lower = 2.95312, upper = 3.05859
Intervallo: lower = 2.95312, upper = 3.00586
Intervallo: lower = 2.97949, upper = 3.00586
Intervallo: lower = 2.99268, upper = 3.00586
Intervallo: lower = 2.99927, upper = 3.00586
Intervallo: lower = 2.99927, upper = 3.00256
Intervallo: lower = 2.99927, upper = 3.00092
Intervallo: lower = 2.99927, upper = 3.00009
Intervallo: lower = 2.99968, upper = 3.00009
Intervallo: lower = 2.99989, upper = 3.00009
Intervallo: lower = 2.99999, upper = 3.00009
Intervallo: lower = 2.99999, upper = 3.00004
Intervallo: lower = 2.99999, upper = 3.00001
Intervallo: lower = 2.99999, upper = 3
Intervallo: lower = 2.99999, upper = 3
```

```
Intervallo: lower = 3, upper = 3
Intervallo: lower = 3, upper = 3
```

```
print(sol)
```

```
[1] 3.000001
```

Metodo delle secanti

Trova la radice cubica di 27 usando il metodo delle secanti

```
sol <- secant_method(radice_cubica, x0=0, x1=27, max_iter=40)
```

```
Iterazione 1: x = 0.037037
Iterazione 2: x = 0.0740232
Iterazione 3: x = 2814.65
Iterazione 4: x = 0.0740266
Iterazione 5: x = 0.07403
Iterazione 6: x = 1642.33
Iterazione 7: x = 0.07404
Iterazione 8: x = 0.07405
Iterazione 9: x = 1641.59
Iterazione 10: x = 0.0740601
Iterazione 11: x = 0.0740701
Iterazione 12: x = 1640.7
Iterazione 13: x = 0.0740801
Iterazione 14: x = 0.0740901
Iterazione 15: x = 1639.81
Iterazione 16: x = 0.0741002
Iterazione 17: x = 0.0741102
Iterazione 18: x = 1638.92
Iterazione 19: x = 0.0741203
Iterazione 20: x = 0.0741303
Iterazione 21: x = 1638.03
Iterazione 22: x = 0.0741404
Iterazione 23: x = 0.0741504
Iterazione 24: x = 1637.14
Iterazione 25: x = 0.0741605
Iterazione 26: x = 0.0741706
Iterazione 27: x = 1636.26
Iterazione 28: x = 0.0741807
Iterazione 29: x = 0.0741908
Iterazione 30: x = 1635.37
Iterazione 31: x = 0.0742008
Iterazione 32: x = 0.0742109
Iterazione 33: x = 1634.48
Iterazione 34: x = 0.074221
Iterazione 35: x = 0.0742312
Iterazione 36: x = 1633.59
Iterazione 37: x = 0.0742413
Iterazione 38: x = 0.0742514
Iterazione 39: x = 1632.7
Iterazione 40: x = 0.0742615
```

```
Warning in secant_method(radice_cubica, x0 = 0, x1 = 27, max_iter = 40): La
radice non è stata trovata dopo il numero massimo di iterazioni.
```

```
print(sol)
```

```
NULL
```

non funziona proviamo con nuovi punti di partenza

```
sol <- secant_method(radice_cubica, x0=27/2, x1=27)
```

```
Iterazione 1: x = 11.5926
Iterazione 2: x = 10.2912
Iterazione 3: x = 7.33534
Iterazione 4: x = 5.77205
Iterazione 5: x = 4.49521
Iterazione 6: x = 3.69196
Iterazione 7: x = 3.2295
Iterazione 8: x = 3.04379
Iterazione 9: x = 3.00316
Iterazione 10: x = 3.00005
Iterazione 11: x = 3
Iterazione 12: x = 3
```

```
print(sol)
```

```
[1] 3
```

Metodo di Newton

Definizione della derivata della funzione

```
radice_cubica_derivative <- function(x) {
  return(3*x*x)
}
```

Trova la radice cubica di 27 usando il metodo di Newton

```
sol <- newton_method(radice_cubica, radice_cubica_derivative, initial_guess = 27)
```

```
Iterazione 1: x = 18.0123
Iterazione 2: x = 12.036
Iterazione 3: x = 8.08611
Iterazione 4: x = 5.52838
Iterazione 5: x = 3.98006
Iterazione 6: x = 3.22152
Iterazione 7: x = 3.01488
Iterazione 8: x = 3.00007
Iterazione 9: x = 3
Iterazione 10: x = 3
```

```
print(sol)
```

```
[1] 3
```

3.2 Equazione di radiazione di Stefan-Boltzmann (Trasferimento di calore)

La potenza irradiata da un corpo nero è data dalla legge di Stefan-Boltzmann:

$$P = \sigma A (T^4 - T_\infty^4)$$

Dove: - $P = 500\,\mathrm{W}$ - $A = 2\,\mathrm{m}^2$ - $T_\infty = 300\,\mathrm{K}$ - $\sigma = 5.67 \times 10^{-8}\,\mathrm{W/m^2K^4}$

Trova la temperatura del corpo T risolvendo l'equazione:

$$f(T) = 5.67 \times 10^{-8} \cdot 2 \cdot (T^4 - 300^4) - 500$$

Soluzione in R

Definizione della funzione $f(T)$:

```
f <- function(T) {
  P <- 500
  A <- 2
  sigma <- 5.67e-8
  return(sigma * A * (T^4 - 300^4) - P)
}
```

Metodo di Bisezione

Trova la temperatura usando il metodo di bisezione:

```
sol <- bisection_method(f, 300, 600)
```

```
Intervallo: lower = 300, upper = 600
Intervallo: lower = 300, upper = 450
Intervallo: lower = 300, upper = 375
Intervallo: lower = 300, upper = 337.5
Intervallo: lower = 318.75, upper = 337.5
Intervallo: lower = 328.125, upper = 337.5
Intervallo: lower = 332.812, upper = 337.5
Intervallo: lower = 332.812, upper = 335.156
Intervallo: lower = 333.984, upper = 335.156
Intervallo: lower = 333.984, upper = 334.57
Intervallo: lower = 334.277, upper = 334.57
Intervallo: lower = 334.424, upper = 334.57
Intervallo: lower = 334.424, upper = 334.497
Intervallo: lower = 334.424, upper = 334.46
Intervallo: lower = 334.424, upper = 334.442
Intervallo: lower = 334.424, upper = 334.433
Intervallo: lower = 334.428, upper = 334.433
Intervallo: lower = 334.431, upper = 334.433
Intervallo: lower = 334.431, upper = 334.432
Intervallo: lower = 334.431, upper = 334.432
Intervallo: lower = 334.431, upper = 334.432
Intervallo: lower = 334.431, upper = 334.432
Intervallo: lower = 334.431, upper = 334.431
Intervallo: lower = 334.431, upper = 334.431
Intervallo: lower = 334.431, upper = 334.431
Intervallo: lower = 334.431, upper = 334.431
Intervallo: lower = 334.431, upper = 334.431
Intervallo: lower = 334.431, upper = 334.431
Intervallo: lower = 334.431, upper = 334.431
```

```
print(sol)
```

```
[1] 334.4315
```

Metodo delle secanti

Trova la soluzione per la temperatura usando il metodo delle secanti:

```
sol <- secant_method(f, x0 = 500, x1 = 400 )
```

```
Iterazione 1: x = 364.523
Iterazione 2: x = 341.536
Iterazione 3: x = 335.308
Iterazione 4: x = 334.459
Iterazione 5: x = 334.432
Iterazione 6: x = 334.431
Iterazione 7: x = 334.431
```

```
print(sol)
```

```
[1] 334.4315
```

Metodo di Newton

Derivata della funzione $f(T)$ rispetto a T

```
f_prime <- function(T) {
  A <- 2
  sigma <- 5.67e-8
  return(4 * sigma * A * T^3)
}
```

Trova la soluzione per la temperatura usando il metodo di Newton-Raphson

```
sol <- newton_method(f, f_prime, initial_guess = 500)
```

```
Iterazione 1: x = 400.018
Iterazione 2: x = 348.871
Iterazione 3: x = 335.303
Iterazione 4: x = 334.435
Iterazione 5: x = 334.431
Iterazione 6: x = 334.431
```

```
print(sol)
```

```
[1] 334.4315
```

3.3 Equazione di Colebrook (Flusso di fluidi)

L'equazione di Colebrook per il coefficiente di attrito f è definita come:

$$\frac{1}{\sqrt{f}} = -2\log\left(\frac{\epsilon/D}{3.7} + \frac{2.51}{Re \cdot \sqrt{f}}\right)$$

Dove:

- $\epsilon/D = 0.005$
- $Re = 1.5 \times 10^5$

Trova il coefficiente di attrito f.

Soluzione in R

Definizione della funzione di Colebrook

```
colebrook <- function(f) {
  epsilon_D <- 0.005
  Re <- 1.5e5
  return(1/sqrt(f) + 2 * log10(epsilon_D/3.7 + 2.51/(Re * sqrt(f))))
}
```

Metodo di Bisezione

Trova il coefficiente di attrito usando il metodo di bisezione

```
sol <- bisection_method(colebrook, 0.01, 0.1)
```

```
Intervallo: lower = 0.01, upper = 0.1
Intervallo: lower = 0.01, upper = 0.055
Intervallo: lower = 0.01, upper = 0.0325
Intervallo: lower = 0.02125, upper = 0.0325
Intervallo: lower = 0.026875, upper = 0.0325
Intervallo: lower = 0.0296875, upper = 0.0325
Intervallo: lower = 0.0296875, upper = 0.0310938
Intervallo: lower = 0.0303906, upper = 0.0310938
Intervallo: lower = 0.0307422, upper = 0.0310938
Intervallo: lower = 0.030918, upper = 0.0310938
Intervallo: lower = 0.030918, upper = 0.0310059
Intervallo: lower = 0.0309619, upper = 0.0310059
Intervallo: lower = 0.0309839, upper = 0.0310059
Intervallo: lower = 0.0309949, upper = 0.0310059
Intervallo: lower = 0.0310004, upper = 0.0310059
Intervallo: lower = 0.0310004, upper = 0.0310031
Intervallo: lower = 0.0310017, upper = 0.0310031
```

```
print(sol)
```

```
[1] 0.03100243
```

Metodo delle secanti

Trova il coefficiente di attrito usando il metodo delle secanti

```
sol <- secant_method(colebrook, x0 = 0.02, x1 = 0.03)
```

```
Iterazione 1: x = 0.030725
Iterazione 2: x = 0.0309953
Iterazione 3: x = 0.031002
Iterazione 4: x = 0.031002
```

```
print(sol)
```

```
[1] 0.03100205
```

Metodo di Newton

Definizione della derivata della funzione di Colebrook rispetto a f

```
colebrook_prime <- function(f) {
  epsilon_D <- 0.005
  Re <- 1.5e5
  num <- 2 * 2.51 / (Re * f^(1.5) * log(10))
```

```
  den <- epsilon_D / 3.7 + 2.51 / (Re * sqrt(f))
  return(-0.5 * f^(-1.5) - num / den)
}
```

Trova il coefficiente di attrito usando il metodo di Newton-Raphson

```
sol <- newton_method(colebrook, colebrook_prime, initial_guess = 0.02)
```

```
Iterazione 1: x = 0.0277966
Iterazione 2: x = 0.0307201
Iterazione 3: x = 0.0309974
Iterazione 4: x = 0.031002
Iterazione 5: x = 0.031002
```

```
print(sol)
```

```
[1] 0.03100205
```

3.4 Equazione di Van der Waals (Equilibrio chimico)

L'equazione di Van der Waals è data da:

$$\left(P + \frac{a}{V^2}\right)(V - b) = RT$$

Dove:

- $P = 100\,\text{kPa}$
- $T = 300\,\text{K}$
- $a = 0.3640\,\text{Pa} \cdot \text{m}^6/\text{mol}^2$ (anidride carbonica)
- $b = 4.267 \cdot 10^{-5}\,\text{m}^3/\text{mol}$ (anidride carbonica)

Trova il volume molare V. A questa pressione il gas è quasi ideale e ci aspettiamo $V \approx RT/P \approx 0.025\,\text{m}^3/\text{mol}$.

Soluzione in R

Definizione della funzione di Van der Waals

```
vanderwaals <- function(V) {
  P <- 100e3     # kPa to Pa
  R <- 8.314     # J/(mol K)
  T <- 300       # K
  a <- 0.3640    # Pa m^6/mol^2 (CO2)
  b <- 4.267e-5  # m^3/mol (CO2)
  return((P + a/(V^2)) * (V - b) - R * T)
}
```

Metodo di Bisezione

Trova il volume usando il metodo di bisezione

```
sol <- bisection_method(vanderwaals, 0.01, 0.1)
```

```
Intervallo: lower = 0.01, upper = 0.1
Intervallo: lower = 0.01, upper = 0.055
Intervallo: lower = 0.01, upper = 0.0325
Intervallo: lower = 0.02125, upper = 0.0325
Intervallo: lower = 0.02125, upper = 0.026875
```

```
Intervallo: lower = 0.0240625, upper = 0.026875
Intervallo: lower = 0.0240625, upper = 0.0254688
Intervallo: lower = 0.0247656, upper = 0.0254688
Intervallo: lower = 0.0247656, upper = 0.0251172
Intervallo: lower = 0.0247656, upper = 0.0249414
Intervallo: lower = 0.0247656, upper = 0.0248535
Intervallo: lower = 0.0248096, upper = 0.0248535
Intervallo: lower = 0.0248315, upper = 0.0248535
Intervallo: lower = 0.0248315, upper = 0.0248425
Intervallo: lower = 0.024837, upper = 0.0248425
Intervallo: lower = 0.024837, upper = 0.0248398
Intervallo: lower = 0.024837, upper = 0.0248384
```

```
print(sol)
```

```
[1] 0.02483772
```

Metodo delle secanti

Trova il volume usando il metodo delle secanti

```
sol <- secant_method(vanderwaals, x0 = 0.02, x1 = 0.03)
```

```
Iterazione 1: x = 0.0248323
Iterazione 2: x = 0.0248384
Iterazione 3: x = 0.0248384
```

```
print(sol)
```

```
[1] 0.02483837
```

Metodo di Newton

Derivata della funzione di Van Der Waals

```
vanderwaals_derivative <- function(V) {
  P <- 100e3      # Pa
  R <- 8.314      # J/(mol K)
  T <- 300        # K
  a <- 0.3640     # Pa m^6/mol^2 (CO2)
  b <- 4.267e-5   # m^3/mol (CO2)

  # Calcolo della derivata
  term1 <- P + a / (V^2)
  term2 <- (2 * a * (V - b)) / (V^3)

  return(term1 - term2)
}
```

Trova il volume usando il metodo di Newton

```
sol <- newton_method(vanderwaals, vanderwaals_derivative, initial_guess = 0.02)
```

```
Iterazione 1: x = 0.024847
Iterazione 2: x = 0.0248384
Iterazione 3: x = 0.0248384
```

```
print(sol)
```

```
[1] 0.02483837
```

3.5 Equazione di Shockley per i diodi

La corrente che attraversa un diodo è descritta dall'equazione di Shockley:

$$I = I_0 \left(e^{\frac{V}{nV_T}} - 1 \right)$$

Dove: - $I = 2\,\text{mA}$ (corrente attraverso il diodo) - $I_0 = 10^{-12}\,\text{A}$ (corrente di saturazione inversa) - $n = 1$ (fattore di idealità del diodo) - $V_T = 26\,\text{mV}$ (tensione termica)

Trova la tensione V del diodo.

Soluzione in R

```r
# Definizione della funzione di Shockley
shockley <- function(V) {
  I <- 2e-3  # Corrente attraverso il diodo in Ampere
  I0 <- 1e-12  # Corrente di saturazione inversa in Ampere
  n <- 1  # Fattore di idealità del diodo
  VT <- 26e-3  # Tensione termica in Volt
  return(I0 * (exp(V / (n * VT)) - 1) - I)
}
```

Metodo di Bisezione

Trova la tensione del diodo usando il metodo di bisezione

```r
sol <- bisection_method(shockley, 0, 1)
```

```
Intervallo: lower = 0, upper = 1
Intervallo: lower = 0.5, upper = 1
Intervallo: lower = 0.5, upper = 0.75
Intervallo: lower = 0.5, upper = 0.625
Intervallo: lower = 0.5, upper = 0.5625
Intervallo: lower = 0.53125, upper = 0.5625
Intervallo: lower = 0.546875, upper = 0.5625
Intervallo: lower = 0.554688, upper = 0.5625
Intervallo: lower = 0.554688, upper = 0.558594
Intervallo: lower = 0.556641, upper = 0.558594
Intervallo: lower = 0.556641, upper = 0.557617
Intervallo: lower = 0.556641, upper = 0.557129
Intervallo: lower = 0.556641, upper = 0.556885
Intervallo: lower = 0.556763, upper = 0.556885
Intervallo: lower = 0.556824, upper = 0.556885
Intervallo: lower = 0.556824, upper = 0.556854
Intervallo: lower = 0.556824, upper = 0.556839
Intervallo: lower = 0.556824, upper = 0.556831
Intervallo: lower = 0.556824, upper = 0.556828
Intervallo: lower = 0.556826, upper = 0.556828
```

```r
print(sol)
```

```
[1] 0.5568266
```

Metodo delle secanti

Trova la tensione del diodo usando il metodo delle secanti

```
sol <- secant_method(shockley, x0 = 0.6, x1 = 0.7)
```

```
Iterazione 1: x = 0.598232
Iterazione 2: x = 0.596581
Iterazione 3: x = 0.576856
Iterazione 4: x = 0.567524
Iterazione 5: x = 0.560234
Iterazione 6: x = 0.557467
Iterazione 7: x = 0.556868
Iterazione 8: x = 0.556827
Iterazione 9: x = 0.556827
```

```
print(sol)
```

```
[1] 0.5568267
```

Metodo di Newton

Derivata della funzione di Shockley

```
shockley_derivative <- function(V) {
  I0 <- 1e-12  # Corrente di saturazione inversa in Ampere
  n <- 1       # Fattore di idealità del diodo
  VT <- 26e-3  # Tensione termica in Volt
  exp_term <- exp(V / (n * VT))  # Calcola l'esponenziale
  derivative <- (I0 / (n * VT)) * exp_term  # Calcola la derivata
  return(derivative)
}
```

Trova la tensione del diodo usando il metodo di Newton

```
sol <- newton_method(shockley, shockley_derivative, initial_guess = 0.6)
```

```
Iterazione 1: x = 0.578941
Iterazione 2: x = 0.564048
Iterazione 3: x = 0.557743
Iterazione 4: x = 0.556843
Iterazione 5: x = 0.556827
Iterazione 6: x = 0.556827
```

```
print(sol)
```

```
[1] 0.5568267
```

3.6 Problemi difficili: quando serve la salvaguardia

Gli esempi precedenti sono *gentili*: la funzione è monotona vicino alla radice, la derivata non si annulla, un punto iniziale ragionevole basta. In questi casi bisezione, secanti e Newton arrivano tutti in fondo e la scelta del metodo è una questione di efficienza.

I sette problemi che seguono sono di natura diversa. Vengono da applicazioni reali o da banchi di prova pubblicati, e su ciascuno **almeno uno dei metodi puri fallisce**: diverge, converge a una radice sbagliata, va in overflow, oppure striscia così lentamente da essere inutilizzabile. Sono gli esempi che giustificano i metodi ibridi di Metodi ibridi e algoritmi con salvaguardia.

3.6.1 Un solutore ibrido minimo

Per i confronti useremo due strumenti. Il primo è `uniroot`, il solutore di libreria di R, che implementa l'algoritmo di Brent–Dekker (Metodo di Brent–Dekker): accetta un intervallo con cambio di segno e restituisce anche il numero di valutazioni usate. Il secondo è una versione compatta di Newton con salvaguardia (Gli ibridi con derivate: uno schema comune), che rifiuta il passo di Newton quando esce dal bracket o non dimezza l'incremento precedente:

```
newton_sicuro <- function(f, df, a, b, tol = 1e-12, nmax = 100) {
  fa <- f(a); fb <- f(b)
  if (fa*fb > 0) stop("gli estremi non racchiudono una radice")
  if (fa < 0) { xl <- a; xh <- b } else { xl <- b; xh <- a }
  x <- 0.5*(a+b); dxold <- abs(b-a); dx <- dxold
  fx <- f(x); dfx <- df(x); if (!is.finite(dfx)) dfx <- 0
  storia <- data.frame()
  for (k in 1:nmax) {
    fuori <- ((x-xh)*dfx - fx) * ((x-xl)*dfx - fx) > 0
    lento <- abs(2*fx) > abs(dxold*dfx)
    if (fuori || lento) { dxold <- dx; dx <- 0.5*(xh-xl); x <- xl + dx; tipo <-
        ↪ "bisezione" }
    else                { dxold <- dx; dx <- fx/dfx;      x <- x - dx;  tipo <-
        ↪ "Newton"    }
    fx <- f(x); dfx <- df(x); if (!is.finite(dfx)) dfx <- 0
    if (fx < 0) xl <- x else xh <- x
    storia <- rbind(storia, data.frame(k = k, passo = tipo, x = x, fx = fx))
    if (abs(dx) < tol || fx == 0) break
  }
  storia
}
```

3.7 Equazione di Keplero: orbite quasi paraboliche

Determinare dove si trova un corpo celeste sulla sua orbita a un dato istante significa risolvere l'**equazione di Keplero**

$$f(E) = E - e\sin E - M = 0,$$

dove M è l'*anomalia media* (proporzionale al tempo trascorso dal perielio), e l'eccentricità dell'orbita ed E l'*anomalia eccentrica* incognita. È il calcolo che sta dietro a ogni effemeride e a ogni propagatore orbitale; la sua soluzione numerica è un problema con quattro secoli di storia [2].

Per orbite quasi circolari ($e \ll 1$) la funzione è quasi lineare e qualunque metodo funziona. Il caso difficile è quello **quasi parabolico**, $e \to 1$, tipico delle comete: in $E = 0$ la derivata

$$f'(E) = 1 - e\cos E$$

vale $1-e$, cioè quasi zero. Il passo di Newton calcolato dalla scelta naturale $E_0 = M$ viene diviso per questa quantità minuscola e scaglia l'iterata lontanissimo, fuori dall'intervallo fisicamente ammissibile $[0, \pi]$.

```
e <- 0.999; M <- 0.01
fK  <- function(E) E - e*sin(E) - M
dfK <- function(E) 1 - e*cos(E)

newton_puro <- function(f, df, x0, n) {
  x <- x0; tr <- x
  for (k in 1:n) {
    d <- df(x); if (!is.finite(d) || d == 0) break
```

```
    x <- x - f(x)/d; tr <- c(tr, x)
    if (!is.finite(x)) break
  }
  tr
}
signif(newton_puro(fK, dfK, M, 12), 7)
```

```
 [1]   0.01000000   9.52458400   4.70312600  -0.93676740  -0.58907810
 [6]  -0.32907450  -0.03183997   6.63146200 -96.36529000 -33.49573000
[11] -11.54740000  14.50457000   4.52375900
```

La prima iterata salta a 9.52, ben fuori dall'intervallo fisico, poi il metodo vaga fra -96 e $+14$ prima di ricadere per puro caso vicino alla radice dopo ventidue passi. Con $e = 0.9999$ e $M = 0.1$ non ci ricade affatto: le iterate raggiungono $1.2 \cdot 10^{10}$ e dopo trenta passi il metodo è a $3.8 \cdot 10^{6}$, cioè non converge.

Il bracket, invece, esiste sempre ed è noto a priori senza alcun calcolo: poiché $|e \sin E| \le 1$, per $M \in [0, \pi]$ la radice sta certamente in $[0, \pi]$.

```
u <- uniroot(fK, c(0, pi), tol = 1e-15)
cat(sprintf("Brent: E = %.15f   f(E) = %.2e   valutazioni = %d\n",
            u$root, u$f.root, u$iter))
```

```
Brent: E = 0.387461123237761   f(E) = 8.67e-18   valutazioni = 14
```

```
knitr::kable(newton_sicuro(fK, dfK, 0, pi),
             digits = c(0, 0, 12, 20), align = "r",
             caption = "Keplero con $e=0.999$, $M=0.01$: Newton con salvaguardia")
```

Table 3.1: Keplero con $e = 0.999$, $M = 0.01$: Newton con salvaguardia

k	passo	x	fx
1	Newton	1.0090000	1.535467e-01
2	Newton	0.6807862	4.201146e-02
3	Newton	0.4929827	1.020043e-02
4	Newton	0.4079477	1.618147e-03
5	Newton	0.3884475	7.421816e-05
6	Newton	0.3874636	1.830228e-07
7	Newton	0.3874611	1.122333e-12
8	Newton	0.3874611	-4.684000e-17
9	Newton	0.3874611	8.670000e-18

Vale la pena osservare che nella colonna `passo` **non compare nessuna bisezione**: tutti i passi di Newton vengono accettati. Il guadagno non sta dunque nell'aver sostituito Newton con qualcosa di più lento, ma in due effetti del bracket. Il primo è che il metodo parte dal punto medio dell'intervallo, $\pi/2$, dove $f' = 1 - e\cos(\pi/2) = 1$ e il modello lineare è eccellente, invece che dal punto $E_0 = M$ suggerito dalla fisica, dove $f' \approx 1 - e$ è quasi nullo. Il secondo è che il test di ammissibilità impedisce comunque a qualunque passo di uscire da $[0, \pi]$. La bisezione resta disponibile come assicurazione e in questo caso non serve mai: **la salvaguardia lavora prevenendo il disastro, non riparandolo**.

3.8 Frequenze proprie di una trave a sbalzo

Le frequenze di vibrazione flessionale di una trave incastrata a un estremo e libera all'altro si ottengono dalle radici dell'**equazione delle frequenze**

$$g(x) = \cos x \cosh x + 1 = 0, \qquad x = \beta L,$$

e la k-esima frequenza propria vale $\omega_k = (\beta_k L)^2 \sqrt{EJ/(\rho A L^4)}$. È un calcolo di routine in progettazione strutturale e in analisi modale.

Il problema numerico non sta nella matematica ma nella **scala**. Il fattore $\cosh x$ cresce come $e^x/2$, quindi g assume valori enormi non appena ci si allontana dall'origine:

```
g <- function(x) cos(x)*cosh(x) + 1
cat(sprintf("g(20) = %.3e\ng(40) = %.3e\ng(100) = %.3e\ng(800) = %.3e\n",
            g(20), g(40), g(100), g(800)))
```

```
g(20) = 9.899e+07
g(40) = -7.849e+16
g(100) = 1.159e+43
g(800) = -Inf
```

```
cat("prodotto g(1)*g(40) =", g(1)*g(40), "\n")
```

```
prodotto g(1)*g(40) = -1.439362e+17
```

Tre conseguenze pratiche, tutte già incontrate in Metodo di bisezione (o dicotomico) e Criteri di arresto e accuratezza raggiungibile:

- a $x \approx 710$ il cosh **va in overflow** e g restituisce `Inf` o `NaN`;
- il **prodotto** $g(a)g(b)$ usato ingenuamente per il test di cambio di segno vale già $-1.4 \cdot 10^{17}$ per le prime radici, e va in overflow molto prima della funzione stessa: è il motivo per cui il test va fatto **confrontando i segni**;
- un criterio d'arresto sul residuo $|g|$ è privo di senso, perché la scala di g cambia di venti ordini di grandezza lungo l'intervallo.

Il rimedio corretto non è numerico ma algebrico: si divide per $\cosh x$, ottenendo una funzione equivalente, limitata e ben condizionata,

$$\tilde{g}(x) = \cos x + \frac{1}{\cosh x} = 0.$$

Le radici sono le stesse, ma $\tilde{g}$ oscilla in $[-1, 1]$ e tende a $\cos x$: le radici si addensano attorno a $(2k-1)\pi/2$, il che fornisce gratuitamente un bracket per ciascuna.

```
g_scal <- function(x) cos(x) + 1/cosh(x)
radici <- sapply(1:6, function(k) {
  centro <- (2*k-1)*pi/2
  uniroot(g_scal, c(centro - 1.4, centro + 1.4), tol = 1e-14)$root
})
knitr::kable(
  data.frame(k = 1:6,
             `$\\beta_k L$` = sprintf("%.10f", radici),
             `$(2k-1)\\pi/2$` = sprintf("%.10f", (2*(1:6)-1)*pi/2),
             `$(\\beta_k L)^2$` = sprintf("%.6f", radici^2),
             check.names = FALSE),
  align = "r", caption = "Prime sei radici dell'equazione delle frequenze")
```

Table 3.2: Prime sei radici dell'equazione delle frequenze

k	$\beta_k L$	$(2k-1)\pi/2$	$(\beta_k L)^2$
1	1.8751040687	1.5707963268	3.516015
2	4.6940911330	4.7123889804	22.034492
3	7.8547574382	7.8539816340	61.697214
4	10.9955407349	10.9955742876	120.901916
5	14.1371683910	14.1371669412	199.859530
6	17.2787595321	17.2787595947	298.555531

Il primo valore, $\beta_1 L = 1.8751040687$, è la costante tabulata in tutti i manuali di vibrazioni meccaniche. Si noti come dalla terza radice in poi $\beta_k L$ coincida con $(2k-1)\pi/2$ fino a quasi tutte le cifre: il termine $1/\cosh x$ diventa trascurabile e le radici si allineano agli zeri del coseno.

3.9 Profondità di immersione di una sfera

Una sfera omogenea di raggio R e densità relativa s galleggia immersa fino alla profondità h. Uguagliando il peso alla spinta di Archimede — cioè il volume della calotta immersa, $\pi h^2(3R-h)/3$, al volume spostato $s\,\frac{4}{3}\pi R^3$ — si ottiene

$$f(h) = h^3 - 3Rh^2 + 4sR^3 = 0.$$

È un classico dei testi di metodi numerici per l'ingegneria, e la sua difficoltà non è la convergenza ma **quale radice si trova**. Il polinomio è cubico e per $0 < s < 1$ ha tre radici reali, di cui una sola ha senso fisico: quella in $[0, 2R]$.

```
R <- 1; s <- 0.6
fh  <- function(h) h^3 - 3*R*h^2 + 4*s*R^3
dfh <- function(h) 3*h^2 - 6*R*h
cat("tutte le radici reali:", signif(sort(Re(polyroot(c(4*s*R^3, 0, -3*R, 1)))), 10),
    ↪  "\n")
```

```
tutte le radici reali: -0.7952197 1.134138 2.661082
```

```
cat("intervallo fisicamente ammissibile: [0,", 2*R, "]\n\n")
```

```
intervallo fisicamente ammissibile: [0, 2 ]
```

```
cat("Newton da h0 = 2.5 :", signif(tail(newton_puro(fh, dfh, 2.5, 10), 1), 10), "<- NON
    ↪  fisica\n")
```

```
Newton da h0 = 2.5 : 2.661082 <- NON fisica
```

```
cat("Newton da h0 = 1.0 :", signif(tail(newton_puro(fh, dfh, 1.0, 10), 1), 10), "<-
    ↪  corretta\n")
```

```
Newton da h0 = 1.0 : 1.134138 <- corretta
```

Partendo da $h_0 = 2.5$ il metodo di Newton converge perfettamente — a $h = 2.661$, che è maggiore del diametro: la sfera sarebbe immersa più del suo stesso ingombro. Il risultato è matematicamente ineccepibile e fisicamente assurdo, e nessun criterio d'arresto può accorgersene.

Un metodo con bracket, invece, riceve l'informazione fisica come dato di ingresso e **non può** restituire una radice fuori da $[0, 2R]$:

```
u <- uniroot(fh, c(0, 2*R), tol = 1e-14)
cat(sprintf("Brent su [0, 2R]: h = %.12f  (%d valutazioni)\n", u$root, u$iter))
```

```
Brent su [0, 2R]: h = 1.134137845705  (6 valutazioni)
```

È la lezione più importante di questo capitolo: il bracket non è soltanto una garanzia di convergenza, è il modo in cui si comunica al solutore ciò che si sa del problema.

3.10 Catenaria: la freccia di una linea aerea

Un cavo omogeneo sospeso fra due sostegni alla stessa quota assume la forma di una catenaria. Detta $2b$ la campata e d la freccia (l'abbassamento in mezzeria), il parametro a della catenaria risolve

$$f(a) = a\left(\cosh\frac{b}{a} - 1\right) - d = 0,$$

ed è la quantità da cui si ricavano tiro e tensione del conduttore. Il problema è reale e quotidiano nel progetto delle linee elettriche.

La difficoltà è duplice e di natura puramente numerica. Primo, il **condizionamento**: per frecce piccole a diventa grande e, sviluppando, $f(a) \approx b^2/(2a) - d$, da cui $f'(a) \approx -b^2/(2a^2)$, che tende a zero come $1/a^2$.

```
b <- 50
f_cat <- function(a, d) 2*a*sinh(b/(2*a))^2 - d        # forma stabile, si veda oltre
tab <- t(sapply(c(10, 1, 0.1, 0.01), function(d) {
  a  <- uniroot(function(a) f_cat(a, d), c(10, 1e7), tol = 1e-12)$root
  dp <- (f_cat(a + 1e-3, d) - f_cat(a - 1e-3, d))/2e-3
  c(d = d, a = a, derivata = dp, condizionamento = 1/abs(dp))
}))
knitr::kable(tab, digits = c(3, 4, 9, 1), align = "r",
             caption = "Condizionamento della radice al diminuire della freccia")
```

Table 3.3: Condizionamento della radice al diminuire della freccia

d	a	derivata	condizionamento
10.00	126.6324	-0.081015286	12.3
1.00	1250.1666	-0.000800107	1249.8
0.10	12500.0167	-0.000008000	124999.8
0.01	125000.0017	-0.000000080	12499999.5

Passando da una freccia di 10 m a una di 1 cm il numero di condizionamento $1/|f'(\alpha)|$ peggiora di sei ordini di grandezza: per la Intervallo di incertezza e condizionamento della radice l'incertezza sul parametro a cresce nella stessa misura, **qualunque** metodo si usi.

Secondo, la **cancellazione**. Scrivere $\cosh(x) - 1$ per $x = b/a$ piccolo è il prototipo della sottrazione catastrofica, perché si sottrae 1 da un numero che vale $1 + x^2/2$. L'identità $\cosh x - 1 = 2\sinh^2(x/2)$ elimina il problema:

```
xs <- c(1e-2, 1e-3, 1e-4, 1e-5)
confronto <- data.frame(
  `$x$`                 = sprintf("%.0e", xs),
```

```
  `$\\cosh x-1$`        = sprintf("%.16e", cosh(xs)-1),
  `$2\\sinh^2(x/2)$`    = sprintf("%.16e", 2*sinh(xs/2)^2),
  `cifre corrette`      = sprintf("%.1f",
      -log10(pmax(abs((cosh(xs)-1) - 2*sinh(xs/2)^2)/(2*sinh(xs/2)^2), 1e-17))),
  check.names = FALSE)
knitr::kable(confronto, align = "r",
             caption = "Cancellazione nella valutazione di $\\cosh x-1$")
```

Table 3.4: Cancellazione nella valutazione di $\cosh x - 1$

x	$\cosh x - 1$	$2\sinh^2(x/2)$	cifre corrette
1e-02	5.0000416668138925e-05	5.0000416668055563e-05	11.8
1e-03	5.0000004159223010e-07	5.0000004166666821e-07	9.8
1e-04	4.9999999696126451e-09	5.0000000041666668e-09	8.2
1e-05	5.0000004137018550e-11	5.0000000000416681e-11	7.1

A $x = 10^{-5}$ la forma ingenua ha perso nove cifre significative su sedici. Nessun solutore può recuperare quell'informazione: la funzione va riscritta prima di chiamare il solutore, non dopo.

3.11 Una funzione piatta: il problema 13 del banco TOMS 748

Il banco di prova usato da Alefeld, Potra e Shi per validare l'algoritmo 748 [1] contiene funzioni costruite apposta per mettere in crisi i metodi di ricerca di zeri. La più istruttiva è

$$f(x) = \begin{cases} x\, e^{-1/x^2}, & x \neq 0, \\ 0, & x = 0, \end{cases} \qquad x \in [-1, 4].$$

È una funzione C^∞ con un unico zero in $x = 0$, dove però **tutte le derivate si annullano**. Il risultato è che in aritmetica finita la funzione non è semplicemente piccola vicino allo zero: è *esattamente* zero.

```
f13 <- function(x) ifelse(x == 0, 0, x*exp(-1/x^2))
knitr::kable(
  data.frame(x = c(0.2, 0.1, 0.06, 0.05, 0.04, 0.037, 0.03, 0.02),
             `$f(x)$` = sprintf("%.3e", f13(c(0.2,0.1,0.06,0.05,0.04,0.037,0.03,0.02))),
             check.names = FALSE),
  align = "r", caption = "Underflow di $x\\,e^{-1/x^2}$ in prossimità dello zero")
```

Table 3.5: Underflow di $x\, e^{-1/x^2}$ in prossimità dello zero

x	$f(x)$
0.200	2.778e-12
0.100	3.720e-45
0.060	1.383e-122
0.050	9.576e-176
0.040	1.472e-273
0.037	2.155e-319
0.030	0.000e+00
0.020	0.000e+00

Sotto $|x| \approx 0.037$ l'esponenziale va in underflow e f restituisce zero esatto: **ogni** punto dell'intervallo $(-0.037, 0.037)$ è, per il calcolatore, una radice. L'intervallo di incertezza della

Intervallo di incertezza e condizionamento della radice qui non è dell'ordine di 10^{-8} o 10^{-5}, ma di 10^{-2}.

Per il metodo di Newton il disastro è di altro tipo. Si ha $f/f' = x^3/(x^2+2)$, quindi lo schema è $x_{k+1} \approx x_k - x_k^3/2$: per x piccolo l'incremento è *cubicamente* piccolo e il metodo striscia.

```
df13 <- function(x) if (x == 0) 0 else exp(-1/x^2)*(1 + 2/x^2)
tr <- newton_puro(f13, Vectorize(df13), 1, 25)
cat("Newton da x0 = 1, iterate 18--25:\n"); print(signif(tail(tr, 8), 6))
```

```
Newton da x0 = 1, iterate 18--25:
```

```
[1] 0.224612 0.219086 0.213951 0.209164 0.204686 0.200486 0.196537 0.192813
```

```
u <- uniroot(f13, c(-1, 4), tol = 1e-14)
cat(sprintf("\nBrent su [-1,4]: x = %.3e  f(x) = %g  (%d valutazioni)\n",
            u$root, u$f.root, u$iter))
```

```

Brent su [-1,4]: x = -1.906e-02  f(x) = -0  (14 valutazioni)
```

Dopo venticinque iterazioni Newton è ancora a $x \approx 0.19$, e procede guadagnando una frazione di cifra per passo; un metodo con bracket chiude la partita in quattordici valutazioni, semplicemente perché dimezza l'intervallo senza chiedersi quanto la funzione sia piatta.

3.12 Il problema 10 del banco TOMS 748

Sempre dallo stesso banco [1], con un difetto opposto al precedente:

$$f(x) = e^{-nx}(x-1) + x^n = 0, \qquad x \in [0,1].$$

Per n grande la funzione è piatta e quasi costante (≈ -1) su quasi tutto l'intervallo, poi risale bruscamente verso $+1$ in un intorno sempre più stretto della radice. In $x = 0$ e in $x = 1$ la derivata non dà alcuna informazione utile sulla posizione dello zero.

```
n <- 20
f10  <- function(x) exp(-n*x)*(x-1) + x^n
df10 <- function(x) exp(-n*x)*(1 - n*(x-1)) + n*x^(n-1)
cat("Newton da x0 = 0:\n"); print(signif(head(newton_puro(f10, df10, 0, 9), 10), 6))
```

```
Newton da x0 = 0:
```

```
 [1] 0.000000 0.047619 0.095125 0.142507 0.189752 0.236846 0.283771 0.330509
 [9] 0.377034 0.423317
```

```
cat("Newton da x0 = 1:\n"); print(signif(head(newton_puro(f10, df10, 1, 9), 10), 6))
```

```
Newton da x0 = 1:
```

```
 [1] 1.000000 0.950000 0.902500 0.857375 0.814506 0.773781 0.735093 0.698341
 [9] 0.663446 0.630398
```

Da entrambi gli estremi Newton avanza a passi quasi costanti — da sinistra $+0.048$ per volta, da destra -5% per volta — comportandosi come un metodo lineare lentissimo: in nove iterazioni, cioè diciotto valutazioni fra funzione e derivata, non ha ancora raggiunto la radice.

```
righe <- lapply(c(1, 5, 10, 20), function(n) {
  f <- function(x) exp(-n*x)*(x-1) + x^n
  u <- uniroot(f, c(0, 1), tol = 1e-14)
  data.frame(n = n, radice = sprintf("%.12f", u$root), valutazioni = u$iter)
})
knitr::kable(do.call(rbind, righe), align = "r",
             caption = "Brent sul problema 10 di TOMS 748")
```

Table 3.6: Brent sul problema 10 di TOMS 748

n	radice	valutazioni
1	0.401058137542	7
5	0.516153518758	10
10	0.539522226908	12
20	0.552704666678	13

Brent chiude con un numero di valutazioni che cresce appena con n, e con la garanzia di non uscire mai da $[0, 1]$.

3.13 Tassi interni di rendimento multipli

Il **tasso interno di rendimento** (TIR, o IRR) di un progetto è il tasso r che annulla il valore attuale netto dei flussi di cassa $C_0, C_1, \ldots, C_n$:

$$\mathrm{VAN}(r) = \sum_{t=0}^{n} \frac{C_t}{(1+r)^t} = 0.$$

Se i flussi cambiano segno una sola volta — un esborso iniziale seguito da ricavi — la funzione è monotona e la radice è unica. Ma per i progetti con **flussi non convenzionali**, in cui un esborso rilevante ricompare alla fine (lo smantellamento di un impianto, la bonifica di un sito, il ripristino di una cava), i cambi di segno sono due e di radici ce ne sono due. È il problema classico del razionamento del capitale segnalato da Lorie e Savage [3].

```
flussi <- c(-1000, 2500, -1540)
van  <- function(r) sum(flussi / (1+r)^(0:2))
dvan <- function(r) sum(-(0:2) * flussi / (1+r)^(1:3))
van  <- Vectorize(van); dvan <- Vectorize(dvan)

cat("radici esatte:", Re(polyroot(rev(flussi)))-1, "\n")
```

```
radici esatte: 0.1 0.4
```

```
cat(sprintf("VAN(0%%) = %.1f   VAN(20%%) = %.1f   VAN(100%%) = %.1f\n",
            van(0), van(0.2), van(1)))
```

```
VAN(0%) = -40.0   VAN(20%) = 13.9   VAN(100%) = -135.0
```

```
cat("\nNewton da r0 =  5% :", sprintf("%.4f", tail(newton_puro(van, dvan, 0.05, 20),
    ↪  1)), "\n")
```

```

Newton da r0 =  5% : 0.1000
```

```
cat(  "Newton da r0 = 50% :", sprintf("%.4f", tail(newton_puro(van, dvan, 0.50, 20),
    ↪  1)), "\n")
```

```
Newton da r0 = 50% : 0.4000
```

Il progetto ha due tassi interni di rendimento, 10% e 40%, entrambi matematicamente legittimi. Newton restituisce l'uno o l'altro a seconda del punto iniziale, e non ha modo di segnalare che esiste una seconda soluzione: la risposta dipende da un dettaglio che nulla ha a che vedere con l'economia del progetto.

```
cat(sprintf("Brent su [0%%, 20%%] : r = %.6f\n", uniroot(van, c(0, 0.2), tol =
    ↪  1e-14)$root))
```

```
Brent su [0%, 20%] : r = 0.100000
```

```
cat(sprintf("Brent su [20%%, 100%%]: r = %.6f\n", uniroot(van, c(0.2, 1), tol =
    ↪  1e-14)$root))
```

```
Brent su [20%, 100%]: r = 0.400000
```

Separando i due intervalli si ottengono entrambe le radici, ciascuna con la garanzia di esistenza che il cambio di segno fornisce. La morale finanziaria — che il TIR non è un criterio affidabile per i flussi non convenzionali — è a sua volta una conseguenza diretta di un fatto numerico.

3.14 Tre patologie del metodo di Newton

Negli esempi precedenti il metodo di Newton fallisce per colpa del *problema*: una derivata quasi nulla, una seconda radice, un intervallo in cui la funzione è numericamente zero. I tre che seguono sono di natura diversa e più istruttiva: la funzione è regolare quanto si vuole, ha **una sola radice semplice** o una sola radice e basta, eppure Newton non converge — e non per sfortuna, ma per struttura.

Servono a fissare un punto che il Convergenza locale enuncia e che è facile dimenticare: la convergenza di Newton è **locale**. Il teorema garantisce un intorno della radice in cui il metodo converge, non dice quanto sia grande, e fuori da quell'intorno non promette nulla. Sono la versione dettagliata dei modi di fallimento elencati in Quando Newton fallisce.

3.14.1 Arcotangente: converge o diverge secondo il punto iniziale

Consideriamo

$$f(x) = \arctan x,$$

che ha l'unica radice $\alpha = 0$, è analitica su tutto $\mathbb{R}$, strettamente crescente, con $f'(0) = 1$: tutte le ipotesi di convergenza locale sono soddisfatte nel modo più comodo possibile. L'iterazione di Newton è

$$N(x) = x - \frac{\arctan x}{1/(1+x^2)} = x - (1+x^2)\arctan x. \tag{3.1}$$

Il guaio si legge nella 3.1: il fattore $1 + x^2$ cresce come x^2, mentre $\arctan x$ tende a $\pi/2$. Per x grande il passo di Newton vale circa $-\frac{\pi}{2}x^2$, cioè è enormemente più lungo della distanza dalla radice. La tangente è quasi orizzontale e interseca l'asse molto lontano, dalla parte opposta.

Esiste quindi una soglia $x^\star$ che separa i due comportamenti. Per simmetria (f è dispari, quindi N è dispari) la soglia è il punto in cui $N(x^\star) = -x^\star$, cioè la radice di

$$2x = (1 + x^2)\arctan x. \tag{3.2}$$

```
f_at  <- function(x) atan(x)
df_at <- function(x) 1/(1+x^2)

xstar <- uniroot(function(x) 2*x - (1+x^2)*atan(x), c(1, 2), tol = 1e-15)$root
cat(sprintf("soglia x* = %.15f\n", xstar))
```

```
soglia x* = 1.391745200270735
```

Il comportamento è allora completamente determinato dal punto iniziale:

- se $|x_0| < x^\star$ il metodo converge, e converge quadraticamente;
- se $|x_0| = x^\star$ le iterate cadono nel ciclo $x^\star \to -x^\star \to x^\star \to \cdots$;
- se $|x_0| > x^\star$ le iterate si allontanano, cambiando segno a ogni passo, e divergono.

```
x0 <- c(1.0, 1.35, 1.3918, 1.5)
tr <- lapply(x0, function(a) { t <- newton_puro(f_at, df_at, a, 8); length(t) <- 9; t })

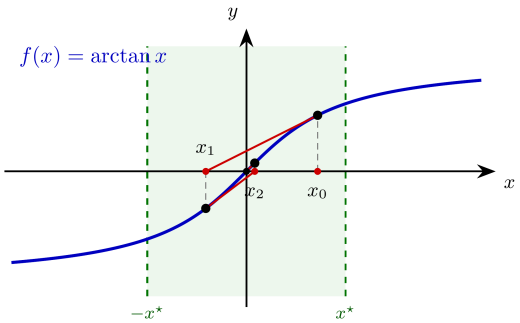
tabella <- data.frame(k = 0:8)
for (i in seq_along(x0))
  tabella[[sprintf("$x_0=%.4f$", x0[i])]] <- formatC(tr[[i]], format = "g", digits = 6)

knitr::kable(tabella, align = "r", row.names = FALSE,
             caption = "Newton su $\\arctan x$ da quattro punti iniziali.")
```

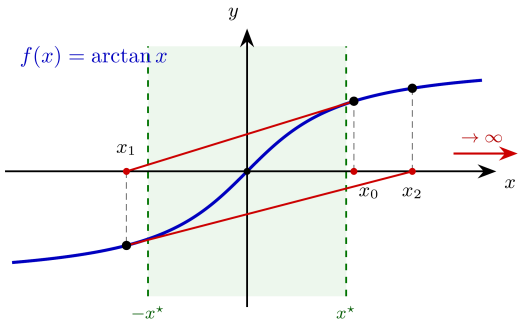
Table 3.7: Newton su arctan x da quattro punti iniziali.

k	$x_0 = 1.0000$	$x_0 = 1.3500$	$x_0 = 1.3918$	$x_0 = 1.5000$
0	1	1.35	1.3918	1.5
1	-0.570796	-1.28409	-1.39189	-1.69408
2	0.11686	1.12412	1.39213	2.32113
3	-0.00106102	-0.785872	-1.39275	-5.11409
4	7.9631e-10	0.291554	1.3944	32.2957
5	0	-0.016251	-1.39876	-1575.32
6	0	2.86105e-06	1.41033	3.89498e+06
7	0	-1.56129e-17	-1.44126	-2.38303e+13
8	0	0	1.52586	8.92028e+26

Le due colonne centrali sono le più istruttive. Con $x_0 = 1.35$, sotto la soglia, il metodo prima si allontana — arriva a -1.28, poi a 1.12 — e solo dopo quattro passi inverte la rotta e crolla sulla radice: la convergenza quadratica si vede soltanto alle ultime due righe. Con $x_0 = 1.3918$, che supera la soglia di appena $5 \cdot 10^{-5}$, le iterate restano a lungo vicine a ± 1.39 e sembrano un ciclo, poi scivolano via e divergono. La differenza fra convergere e divergere sta nella quinta cifra decimale del punto iniziale, e nessuna informazione ricavabile da f nei punti visitati permette di prevederla.



$|x_0| < x^\star$: converge (qui $x_0 = 1$)



$|x_0| > x^\star$: diverge (qui $x_0 = 1.5$)

Che cosa insegna questo esempio

La funzione non ha difetti: è analitica, con una sola radice, semplice, e derivata mai nulla. Il fallimento è interamente dovuto al **punto iniziale**, e la frontiera fra i due regimi è una soglia netta che nessun test sui valori di f può individuare. Un metodo con salvaguardia (Metodi ibridi e algoritmi con salvaguardia) rende la cosa irrilevante: appena il passo di Newton esce dal bracket, viene sostituito da una bisezione.

3.14.2 Un ciclo che non finisce mai

Nell'esempio precedente il ciclo esisteva, ma per un solo valore del punto iniziale. Si può fare di peggio: costruire una funzione per cui Newton cicla **da qualunque punto si parta**. Basta

$$f(x) = \begin{cases} \sqrt{x} & x \geq 0, \\ -\sqrt{-x} & x < 0, \end{cases} \qquad \text{ovvero} \qquad f(x) = \operatorname{sign}(x)\sqrt{|x|}. \tag{3.3}$$

È continua, strettamente crescente, dispari, e ha l'unica radice $\alpha = 0$. La derivata è

$$f'(x) = \frac{1}{2\sqrt{|x|}},$$

definita e positiva per ogni $x \neq 0$. Il conto del passo di Newton si fa a mano:

$$N(x) = x - \frac{\operatorname{sign}(x)\sqrt{|x|}}{\dfrac{1}{2\sqrt{|x|}}} = x - 2\operatorname{sign}(x)\,|x| = x - 2x = -x. \tag{3.4}$$

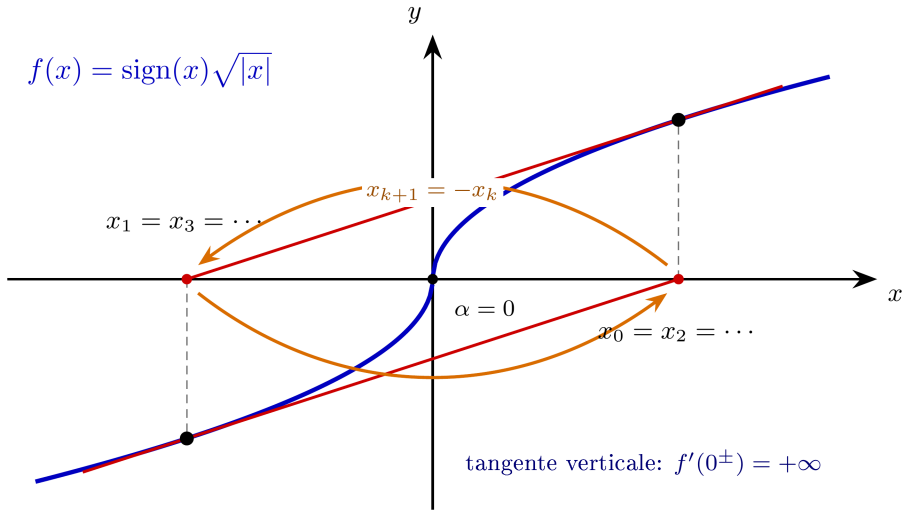
L'iterazione è dunque **esattamente** $x_{k+1} = -x_k$: da qualunque $x_0 \neq 0$ si parta, le iterate rimbalzano fra x_0 e $-x_0$ per sempre, senza avvicinarsi di un millimetro alla radice che sta in mezzo.

```
f_sq  <- function(x) sign(x)*sqrt(abs(x))
df_sq <- function(x) 1/(2*sqrt(abs(x)))

for (a in c(1.3, -0.7, 1e6))
  cat(sprintf("%-6s", format(a)),
      sprintf("%9.3g", newton_puro(f_sq, df_sq, a, 6)), "\n")
```

```
1.3          1.3      -1.3       1.3      -1.3       1.3      -1.3       1.3
-0.7        -0.7       0.7      -0.7       0.7      -0.7       0.7      -0.7
1e+06      1e+06    -1e+06     1e+06    -1e+06     1e+06    -1e+06     1e+06
```

Si noti l'ultima riga: anche partendo da 10^6 — cioè a un milione di distanza dalla radice — il metodo non diverge; semplicemente non si muove. E, cosa più insidiosa, **non dà alcun segnale di errore**: le iterate sono finite, la derivata è finita e non nulla, f è finita. Un criterio d'arresto basato su $|x_{k+1} - x_k|$ non scatta mai, uno basato su $|f(x_k)|$ nemmeno. Il programma gira fino al numero massimo di iterazioni e restituisce un numero che non è una radice.



L'origine del guaio è la tangente **verticale** nell'origine: $f'(x) \to +\infty$ per $x \to 0$. La radice è di «ordine frazionario» $1/2$, e il Convergenza locale non si applica perché richiede $f \in C^2$ in un intorno della radice — qui nemmeno f' è limitata.

3.14.3 Fuga garantita

Cambiando l'esponente si passa dal ciclo alla divergenza. Sia

$$f(x) = \begin{cases} \sqrt[3]{x} & x \geq 0, \\ -\sqrt[3]{-x} & x < 0, \end{cases} \qquad \text{ovvero} \qquad f(x) = \mathrm{sign}(x)\sqrt[3]{|x|}, \tag{3.5}$$

che è la radice cubica reale, anch'essa continua, dispari, strettamente crescente, con l'unica radice $\alpha = 0$. Ora

$$f'(x) = \frac{1}{3}|x|^{-2/3}, \qquad N(x) = x - \frac{\mathrm{sign}(x)|x|^{1/3}}{\frac{1}{3}|x|^{-2/3}} = x - 3x = -2x. \tag{3.6}$$

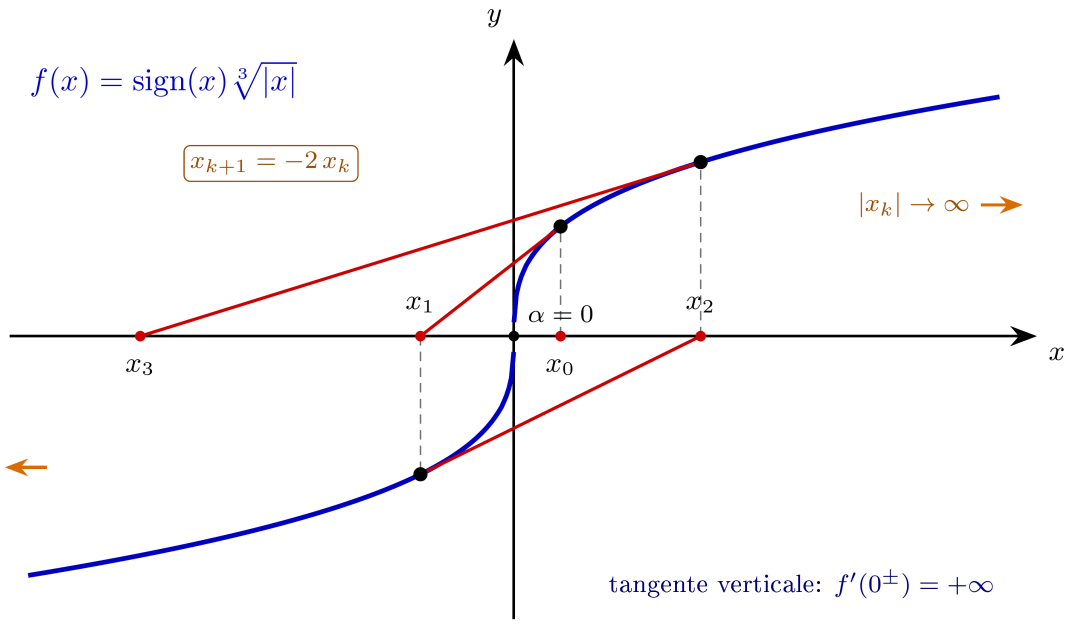
L'iterazione è $x_{k+1} = -2x_k$: il modulo **raddoppia** a ogni passo e il segno si alterna. Non esiste alcun punto iniziale, per quanto vicino alla radice, da cui il metodo converga.

```
f_cb  <- function(x) sign(x)*abs(x)^(1/3)
df_cb <- function(x) (1/3)*abs(x)^(-2/3)

for (a in c(0.5, 1e-6))
  cat(sprintf("%-6s", format(a)),
      sprintf("%8.4g", newton_puro(f_cb, df_cb, a, 7)), "\n")
```

```
0.5         0.5       -1        2       -4        8      -16       32      -64
1e-06     1e-06    -2e-06    4e-06    -8e-06  1.6e-05 -3.2e-05  6.4e-05 -0.000128
```

La seconda riga è la più eloquente: partendo a 10^{-6} dalla radice, dopo venti iterazioni si è a distanza 1, e da lì si continua a raddoppiare. Qui il criterio d'arresto scatta — nella direzione sbagliata: $|x_{k+1} - x_k|$ *cresce*, e un'implementazione che non controlli l'overflow finisce in `Inf`.



3.14.4 Un quadro unico

I due esempi non sono curiosità separate: appartengono alla stessa famiglia a un parametro

$$f_p(x) = \operatorname{sign}(x)\,|x|^p, \qquad p > 0,$$

per la quale il passo di Newton si calcola una volta per tutte:

$$\frac{f_p(x)}{f_p'(x)} = \frac{\operatorname{sign}(x)|x|^p}{p\,|x|^{p-1}} = \frac{x}{p}, \qquad \text{quindi} \qquad \boxed{N_p(x) = \left(1 - \frac{1}{p}\right)x.} \tag{3.7}$$

L'iterazione è **lineare** con fattore $\lambda_p = 1 - 1/p$, e tutto dipende da $|\lambda_p|$:

```
p  <- c(1/4, 1/3, 1/2, 1, 2, 3)
tr <- lapply(p, function(q) {
  f  <- function(x) sign(x)*abs(x)^q
  df <- function(x) q*abs(x)^(q-1)
  formatC(newton_puro(f, df, 1, 5), format = "g", digits = 4)
})

knitr::kable(
  cbind(data.frame(`$p$`      = c("1/4","1/3","1/2","1","2","3"),
                   `$1-1/p$`  = formatC(1-1/p, format = "f", digits = 3),
                   check.names = FALSE),
        as.data.frame(do.call(rbind, tr))),
  col.names = c("$p$", "$1-1/p$", paste0("$x_", 0:5, "$")),
  align = "r", row.names = FALSE,
  caption = "Newton su $f_p(x)=\\operatorname{sign}(x)|x|^p$ a partire da $x_0=1$.")
```

Table 3.8: Newton su $f_p(x) = \operatorname{sign}(x)|x|^p$ a partire da $x_0 = 1$.

p	$1-1/p$	x_0	x_1	x_2	x_3	x_4	x_5
1/4	-3.000	1	-3	9	-27	81	-243
1/3	-2.000	1	-2	4	-8	16	-32

p	$1 - 1/p$	x_0	x_1	x_2	x_3	x_4	x_5
1/2	-1.000	1	-1	1	-1	1	-1
1	0.000	1	0	0	0	0	0
2	0.500	1	0.5	0.25	0.125	0.0625	0.03125
3	0.667	1	0.6667	0.4444	0.2963	0.1975	0.1317

Le righe riassumono tutti i casi possibili.

p	$\lambda_p = 1 - 1/p$	comportamento
$p < 1/2$	$\lambda_p < -1$	diverge, oscillando, tanto più in fretta quanto più p è piccolo
$p = 1/3$	-2	la fuga di Sezione 3.14.3
$p = 1/2$	-1	il ciclo di Sezione 3.14.2
$1/2 < p < 1$	$-1 < \lambda_p < 0$	converge linearmente, oscillando
$p = 1$	0	radice semplice: converge in **un** passo
$p > 1$	$0 < \lambda_p < 1$	radice multipla di ordine p: converge linearmente con fattore $1 - 1/p$

L'ultima riga è il risultato classico sulle radici multiple di Radici multiple e rallentamento della convergenza, che qui appare come un caso particolare. Le prime due sono il suo prolungamento naturale per esponenti **minori di uno**, dove la derivata esplode invece di annullarsi: la stessa formula, letta al di là della soglia $p = 1/2$, smette di descrivere una convergenza lenta e descrive un ciclo o una fuga.

Morale

In tutti e tre gli esempi la funzione ha una sola radice e il metodo di Newton, applicato senza rete, produce una successione perfettamente finita e perfettamente inutile. Nessuno dei criteri d'arresto usuali se ne accorge. Il rimedio è sempre lo stesso: partire da un intervallo con cambio di segno e rifiutare il passo di Newton quando esce dal bracket — cioè uno qualsiasi dei metodi di Metodi ibridi e algoritmi con salvaguardia, che su tutti e tre questi problemi convergono senza sforzo.

```
cat(sprintf("arctan  su [-1, 5]        : %.3e\n", uniroot(f_at, c(-1, 5), tol =
    ↪ 1e-14)$root))
```

```
arctan  su [-1, 5]        : 2.877e-22
```

```
cat(sprintf("sign(x)sqrt(|x|) su [-2,1]: %.3e\n", uniroot(f_sq, c(-2, 1), tol =
    ↪ 1e-14)$root))
```

```
sign(x)sqrt(|x|) su [-2,1]: -1.957e-15
```

```
cat(sprintf("radice cubica su [-3, 2]  : %.3e\n", uniroot(f_cb, c(-3, 2), tol =
    ↪ 1e-14)$root))
```

```
radice cubica su [-3, 2]  : -2.988e-15
```

In sintesi

Ogni esempio illustra un modo diverso in cui un metodo puro può sbagliare, e il rimedio corrispondente.

Problema	Che cosa va storto	Rimedio
Keplero, $e \to 1$	$f' \approx 1 - e \approx 0$: Newton proietta l'iterata fuori da $[0, \pi]$	bracket noto a priori + salvaguardia
Trave a sbalzo	overflow di cosh; il prodotto $f(a)f(b)$ esplode	riscalare la funzione; test sui **segni**
Sfera galleggiante	converge a una radice **non fisica**	il bracket codifica il vincolo fisico
Catenaria	condizionamento $\propto a^2$; cancellazione in $\cosh x - 1$	riformulare con $2\sinh^2(x/2)$; accettare il limite
TOMS 748 n.13	$f \equiv 0$ in aritmetica finita su un intervallo ampio	nessun metodo fa meglio: è il limite della Intervallo di incertezza e condizionamento della radice
TOMS 748 n.10	Newton avanza a passi costanti, come un metodo lineare	bisezione garantita
TIR multiplo	due radici: la risposta dipende dal punto iniziale	un bracket per ciascuna radice
$\arctan x$	Newton diverge se $\lvert x_0\rvert > x^\star$, pur essendo f analitica	salvaguardia: il passo lungo viene rifiutato
$\operatorname{sign}(x)\sqrt{\lvert x\rvert}$	ciclo $x \mapsto -x$ da **ogni** punto iniziale	il bracket si dimezza comunque
$\operatorname{sign}(x)\sqrt[3]{\lvert x\rvert}$	fuga $x \mapsto -2x$ da **ogni** punto iniziale	il bracket si dimezza comunque

Il filo comune è che in **nessuno** di questi casi il problema sta nell'aritmetica del metodo: sta nel fatto che un metodo aperto non ha modo di sapere dove la radice *deve* stare. Il bracket è esattamente quell'informazione, ed è gratis — basta averla.



4. Criteri per controllo ordine

$$\underbrace{\underbrace{\underbrace{\underbrace{A\mathbf{1} = c}_{\text{consistenza}},\quad \boldsymbol{b}\cdot\mathbf{1} = 1}_{\text{ordine 1}},\quad \boldsymbol{b}\cdot c = \frac{1}{2}}_{\text{ordine 2}},\quad \boldsymbol{b}\cdot c^2 = \frac{1}{3},\quad \boldsymbol{b}\cdot\boldsymbol{d} = \frac{1}{6}}_{\text{ordine 3}}$$
$$\boldsymbol{b}\cdot c^3 = \frac{1}{4},\quad (\boldsymbol{b}\otimes c)\cdot\boldsymbol{d} = \frac{1}{8},\quad \boldsymbol{e}\cdot c^2 = \frac{1}{12},\quad \boldsymbol{e}\cdot\boldsymbol{d} = \frac{1}{24} \quad \text{ordine 4}$$

$$\boldsymbol{d} = A\boldsymbol{c},\qquad \boldsymbol{e} = A^T\boldsymbol{b},\qquad c^2 = c\otimes c,\qquad c^3 = c\otimes c\otimes c$$

4.1 Euler Esplicito

Il metodo di Euler esplicito rappresentato utilizzando un tableau di Butcher[1]:

0	0
	1

- Check consistency $A\mathbf{1} = c$

$$\underbrace{(0)}_{A}\,\mathbf{1} = \underbrace{(0)}_{c}$$

- Check ordine 1 $\boldsymbol{b}\cdot\mathbf{1} = 1$

$$\underbrace{(1)}_{\boldsymbol{b}}\cdot\mathbf{1} = 1$$

- Check ordine 2 $\boldsymbol{b}\cdot c = \frac{1}{2}$

$$\underbrace{(1)}_{\boldsymbol{b}}\cdot\underbrace{(0)}_{c} = 0 \neq \frac{1}{2}$$

[1] John Charles Butcher (1933)

l'ordine è quindi 1.

4.2 Metodo del Punto Medio

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1/2 & 1/2 & 0 \\ \hline & 0 & 1 \end{array}$$

- Check consistency $\boldsymbol{A1 = c}$

$$\underbrace{\begin{pmatrix} 0 & 0 \\ 1/2 & 0 \end{pmatrix}}_{\boldsymbol{A}} \underbrace{\begin{pmatrix} 1 \\ 1 \end{pmatrix}}_{\mathbf{1}} = \underbrace{\begin{pmatrix} 0 \\ 1/2 \end{pmatrix}}_{\boldsymbol{c}}$$

- Check ordine 1 $\boldsymbol{b \cdot 1 = 1}$

$$\underbrace{\begin{pmatrix} 0 \\ 1 \end{pmatrix}}_{\boldsymbol{b}} \cdot \underbrace{\begin{pmatrix} 1 \\ 1 \end{pmatrix}}_{\mathbf{1}} = 1$$

- Check ordine 2 $\boldsymbol{b \cdot c = \frac{1}{2}}$

$$\underbrace{\begin{pmatrix} 0 \\ 1 \end{pmatrix}}_{\boldsymbol{b}} \cdot \underbrace{\begin{pmatrix} 0 \\ 1/2 \end{pmatrix}}_{\boldsymbol{c}} = \frac{1}{2}$$

- Check ordine 3 $\boldsymbol{b \cdot c^2 = \frac{1}{3}}$

$$\underbrace{\begin{pmatrix} 0 \\ 1 \end{pmatrix}}_{\boldsymbol{b}} \cdot \underbrace{\begin{pmatrix} 0 \\ 1/4 \end{pmatrix}}_{\boldsymbol{c^2}} = \frac{1}{4} \neq \frac{1}{3}$$

- Check ordine 3 $\boldsymbol{b \cdot d = \frac{1}{6}}$ (non serve ordine 2)

l'ordine è quindi 2.

4.3 Metodo di Ralston

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 2/3 & 2/3 & 0 \\ \hline & 1/4 & 3/4 \end{array}$$

- Check consistency $\boldsymbol{A1 = c}$

$$\underbrace{\begin{pmatrix} 0 & 0 \\ 2/3 & 0 \end{pmatrix}}_{\boldsymbol{A}} \underbrace{\begin{pmatrix} 1 \\ 1 \end{pmatrix}}_{\mathbf{1}} = \underbrace{\begin{pmatrix} 0 \\ 2/3 \end{pmatrix}}_{\boldsymbol{c}}$$

- Check ordine 1 $\boldsymbol{b}\cdot\mathbf{1}=\mathbf{1}$

$$\underbrace{\begin{pmatrix}1/4\\3/4\end{pmatrix}}_{\boldsymbol{b}}\cdot\underbrace{\begin{pmatrix}1\\1\end{pmatrix}}_{\mathbf{1}}=1$$

- Check ordine 2 $\boldsymbol{b}\cdot\boldsymbol{c}=\frac{1}{2}$

$$\underbrace{\begin{pmatrix}1/4\\3/4\end{pmatrix}}_{\boldsymbol{b}}\cdot\underbrace{\begin{pmatrix}0\\2/3\end{pmatrix}}_{\boldsymbol{c}}=\frac{1}{2}$$

- Check ordine 3 $\boldsymbol{b}\cdot\boldsymbol{c}^2=\frac{1}{3}$

$$\underbrace{\begin{pmatrix}1/4\\3/4\end{pmatrix}}_{\boldsymbol{b}}\cdot\underbrace{\begin{pmatrix}0\\4/9\end{pmatrix}}_{\boldsymbol{c}^2}=\frac{1}{3}$$

- Check ordine 3 $\boldsymbol{b}\cdot\boldsymbol{d}=\frac{1}{6}$

$$\boldsymbol{d}=\underbrace{\begin{pmatrix}0&0\\2/3&0\end{pmatrix}}_{\boldsymbol{A}}\underbrace{\begin{pmatrix}0\\2/3\end{pmatrix}}_{\boldsymbol{c}}=\underbrace{\begin{pmatrix}0\\0\end{pmatrix}}$$

$$\underbrace{\begin{pmatrix}1/4\\3/4\end{pmatrix}}_{\boldsymbol{b}}\cdot\underbrace{\begin{pmatrix}0\\0\end{pmatrix}}_{\boldsymbol{d}}=0\neq\frac{1}{6}$$

l'ordine è quindi 2.

4.4 Runge-Kutta del terzo ordine (RK3)

$$\begin{array}{c|ccc}0&0&0&0\\1/2&1/2&0&0\\1&-1&2&0\\\hline&1/6&2/3&1/6\end{array}$$

- Check consistency $\boldsymbol{A}\mathbf{1}=\boldsymbol{c}$"{=latex}

$$\underbrace{\begin{pmatrix}0&0&0\\1/2&0&0\\-1&2&0\end{pmatrix}}_{\boldsymbol{A}}\underbrace{\begin{pmatrix}1\\1\\1\end{pmatrix}}_{\mathbf{1}}=\underbrace{\begin{pmatrix}0\\1/2\\1\end{pmatrix}}_{\boldsymbol{c}}$$

- Check ordine 1 $\boldsymbol{b}\cdot\mathbf{1}=\mathbf{1}$

$$\underbrace{\begin{pmatrix}1/6\\2/3\\1/6\end{pmatrix}}_{\boldsymbol{b}}\cdot\underbrace{\begin{pmatrix}1\\1\\1\end{pmatrix}}_{\mathbf{1}}=1$$

- Check ordine 2 $\boldsymbol{b}\cdot\boldsymbol{c}=\frac{1}{2}$

$$\underbrace{\begin{pmatrix}1/6\\2/3\\1/6\end{pmatrix}}_{\boldsymbol{b}}\cdot\underbrace{\begin{pmatrix}0\\1/2\\1\end{pmatrix}}_{\boldsymbol{c}}=\frac{1}{3}+\frac{1}{6}=\frac{1}{2}$$

- Check ordine 3 $\boldsymbol{b}\cdot\boldsymbol{c}^2=\frac{1}{3}$

$$\underbrace{\begin{pmatrix}1/6\\2/3\\1/6\end{pmatrix}}_{\boldsymbol{b}}\cdot\underbrace{\begin{pmatrix}0\\1/4\\1\end{pmatrix}}_{\boldsymbol{c}^2}=\frac{1}{6}+\frac{1}{6}=\frac{1}{3}$$

- Check ordine 3 $\boldsymbol{b}\cdot\boldsymbol{d}=\frac{1}{6}$

$$\boldsymbol{d}=\underbrace{\begin{pmatrix}0&0&0\\1/2&0&0\\-1&2&0\end{pmatrix}}_{\boldsymbol{A}}\underbrace{\begin{pmatrix}0\\1/2\\1\end{pmatrix}}_{\boldsymbol{c}}=\underbrace{\begin{pmatrix}0\\0\\1\end{pmatrix}}_{\boldsymbol{d}}$$

$$\underbrace{\begin{pmatrix}1/6\\2/3\\1/6\end{pmatrix}}_{\boldsymbol{b}}\cdot\underbrace{\begin{pmatrix}0\\0\\1\end{pmatrix}}_{\boldsymbol{d}}=\frac{1}{6}$$

- Check ordine 4 $\boldsymbol{b}\cdot\boldsymbol{c}^3=\frac{1}{4}$

$$\underbrace{\begin{pmatrix}1/6\\2/3\\1/6\end{pmatrix}}_{\boldsymbol{b}}\cdot\underbrace{\begin{pmatrix}0\\1/8\\1\end{pmatrix}}_{\boldsymbol{c}^3}=\frac{1}{12}+\frac{1}{6}=\frac{1}{4}$$

- Check ordine 4 $(\boldsymbol{b}\otimes\boldsymbol{c})\cdot\boldsymbol{d}=\frac{1}{8}$

$$\underbrace{\begin{pmatrix}1/6\\2/3\\1/6\end{pmatrix}}_{\boldsymbol{b}}\otimes\underbrace{\begin{pmatrix}0\\1/2\\1\end{pmatrix}}_{\boldsymbol{c}}\cdot\underbrace{\begin{pmatrix}0\\0\\1\end{pmatrix}}_{\boldsymbol{d}}=\frac{1}{6}\neq\frac{1}{8}$$

- Check ordine 4 $\boldsymbol{b}\quad\boldsymbol{e}\cdot\boldsymbol{c}^2=\frac{1}{12}$ (non serve ordine 3)
- Check ordine 4 $\boldsymbol{e}\cdot\boldsymbol{d}=\frac{1}{24}$ (non serve ordine 3

l'ordine è quindi 3.

4.5 Runge-Kutta del quarto ordine: regola 3/8

$$\begin{array}{c|cccc} 0 & 0 & 0 & 0 & 0 \\ 1/3 & 1/3 & 0 & 0 & 0 \\ 2/3 & -1/3 & 1 & 0 & 0 \\ 1 & 1 & -1 & 1 & 0 \\ \hline & 1/8 & 3/8 & 3/8 & 1/8 \end{array}$$

- Check consistency $\boldsymbol{A}\mathbf{1} = \boldsymbol{c}$

$$\underbrace{\begin{pmatrix} 0 & 0 & 0 & 0 \\ 1/3 & 0 & 0 & 0 \\ -1/3 & 1 & 0 & 0 \\ 1 & -1 & 1 & 0 \end{pmatrix}}_{\boldsymbol{A}} \underbrace{\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}}_{\mathbf{1}} = \underbrace{\begin{pmatrix} 0 \\ 1/3 \\ 2/3 \\ 1 \end{pmatrix}}_{\boldsymbol{c}}$$

- Check ordine 1 $\boldsymbol{b} \cdot \mathbf{1} = 1$

$\boldsymbol{b}$	1/8	3/8	3/8	1/8	1	OK

- Check ordine 2 $\boldsymbol{b} \cdot \boldsymbol{c} = \frac{1}{2}$

$\boldsymbol{b}$	1/8	3/8	3/8	1/8	1	OK
$\boldsymbol{c}$	0	1/3	2/3	1		
$\boldsymbol{b} \otimes \boldsymbol{c}$	0	1/8	1/4	1/8	1/2	OK

- Check ordine 3 $\boldsymbol{b} \cdot \boldsymbol{c}^2 = \frac{1}{3}$

$\boldsymbol{b}$	1/8	3/8	3/8	1/8	1	OK
$\boldsymbol{c}$	0	1/3	2/3	1		
$\boldsymbol{b} \otimes \boldsymbol{c}$	0	1/8	1/4	1/8	1/2	OK
$\boldsymbol{b} \otimes \boldsymbol{c}^2$	0	1/24	1/6	1/8	1/3	OK

- Check ordine 3 $\boldsymbol{b} \cdot \boldsymbol{d} = \frac{1}{6}$

$$\underbrace{\begin{pmatrix} 0 & 0 & 0 & 0 \\ 1/3 & 0 & 0 & 0 \\ -1/3 & 1 & 0 & 0 \\ 1 & -1 & 1 & 0 \end{pmatrix}}_{\boldsymbol{A}} \underbrace{\begin{pmatrix} 0 \\ 1/3 \\ 2/3 \\ 1 \end{pmatrix}}_{\boldsymbol{c}} = \underbrace{\begin{pmatrix} 0 \\ 0 \\ 1/3 \\ 1/3 \end{pmatrix}}_{\boldsymbol{d}}$$

$\boldsymbol{b}$	1/8	3/8	3/8	1/8	1	OK
$\boldsymbol{c}$	0	1/3	2/3	1		
$\boldsymbol{b} \otimes \boldsymbol{c}$	0	1/8	1/4	1/8	1/2	OK
$\boldsymbol{b} \otimes \boldsymbol{c}^2$	0	1/24	1/6	1/8	1/3	OK
$\boldsymbol{d}$	0	0	1/3	1/3		
$\boldsymbol{b} \otimes \boldsymbol{d}$	0	0	1/8	1/24	1/6	OK

- Check ordine 4 $\quad \boldsymbol{b} \cdot \boldsymbol{c}^3 = \frac{1}{4}$

$\boldsymbol{b}$	1/8	3/8	3/8	1/8	1	OK
$\boldsymbol{c}$	0	1/3	2/3	1		
$\boldsymbol{b} \otimes \boldsymbol{c}$	0	1/8	1/4	1/8	1/2	OK
$\boldsymbol{b} \otimes \boldsymbol{c}^2$	0	1/24	1/6	1/8	1/3	OK
$\boldsymbol{d}$	0	0	1/3	1/3		
$\boldsymbol{b} \otimes \boldsymbol{d}$	0	0	1/8	1/24	1/6	OK
$\boldsymbol{b} \otimes \boldsymbol{c}^3$	0	1/72	1/9	1/8	1/4	OK

- Check ordine 4 $\quad (\boldsymbol{b} \otimes \boldsymbol{c}) \cdot \boldsymbol{d} = \frac{1}{8}$

$\boldsymbol{b}$	1/8	3/8	3/8	1/8	1	OK
$\boldsymbol{c}$	0	1/3	2/3	1		
$\boldsymbol{b} \otimes \boldsymbol{c}$	0	1/8	1/4	1/8	1/2	OK
$\boldsymbol{b} \otimes \boldsymbol{c}^2$	0	1/24	1/6	1/8	1/3	OK
$\boldsymbol{d}$	0	0	1/3	1/3		
$\boldsymbol{b} \otimes \boldsymbol{d}$	0	0	1/8	1/24	1/6	OK
$\boldsymbol{b} \otimes \boldsymbol{c}^3$	0	1/72	1/9	1/8	1/4	OK
$\boldsymbol{b} \otimes \boldsymbol{c} \otimes \boldsymbol{d}$	0	0	1/12	1/24	1/8	OK

- Check ordine 4 $\quad \boldsymbol{e} \cdot \boldsymbol{c}^2 = \frac{1}{12}$

$$\frac{1}{8} \underbrace{\begin{pmatrix} 1 & 3 & 3 & 1 \end{pmatrix}}_{\boldsymbol{b}^T} \underbrace{\begin{pmatrix} 0 & 0 & 0 & 0 \\ 1/3 & 0 & 0 & 0 \\ -1/3 & 1 & 0 & 0 \\ 1 & -1 & 1 & 0 \end{pmatrix}}_{\boldsymbol{A}} = \frac{1}{8} \underbrace{\begin{pmatrix} 1 & 2 & 1 & 0 \end{pmatrix}}_{\boldsymbol{e}^T}$$

$\boldsymbol{b}$	1/8	3/8	3/8	1/8	1	OK
$\boldsymbol{c}$	0	1/3	2/3	1		
$\boldsymbol{b} \otimes \boldsymbol{c}$	0	1/8	1/4	1/8	1/2	OK
$\boldsymbol{b} \otimes \boldsymbol{c}^2$	0	1/24	1/6	1/8	1/3	OK
$\boldsymbol{d}$	0	0	1/3	1/3		
$\boldsymbol{b} \otimes \boldsymbol{d}$	0	0	1/8	1/24	1/6	OK
$\boldsymbol{b} \otimes \boldsymbol{c}^3$	0	1/72	1/9	1/8	1/4	OK
$\boldsymbol{b} \otimes \boldsymbol{c} \otimes \boldsymbol{d}$	0	0	1/12	1/24	1/8	OK
$\boldsymbol{e}$	1/8	1/4	1/8	0		
$\boldsymbol{c}^2$	0	1/9	4/9	1		
$\boldsymbol{e} \otimes \boldsymbol{c}^2$	0	1/36	1/18	0	1/12	OK

- Check ordine 4 $\quad \boldsymbol{e} \cdot \boldsymbol{d} = \frac{1}{24}$

b	$1/8$	$3/8$	$3/8$	$1/8$	1	OK
c	0	$1/3$	$2/3$	1		
$b \otimes c$	0	$1/8$	$1/4$	$1/8$	$1/2$	OK
$b \otimes c^2$	0	$1/24$	$1/6$	$1/8$	$1/3$	OK
d	0	0	$1/3$	$1/3$		
$b \otimes d$	0	0	$1/8$	$1/24$	$1/6$	OK
$b \otimes c^3$	0	$1/72$	$1/9$	$1/8$	$1/4$	OK
$b \otimes c \otimes d$	0	0	$1/12$	$1/24$	$1/8$	OK
e	$1/8$	$1/4$	$1/8$	0		
c^2	0	$1/9$	$4/9$	1		
$e \otimes c^2$	0	$1/36$	$1/18$	0	$1/12$	OK
$e \otimes d$	0	0	$1/24$	0	$1/24$	OK

l'ordine è quindi è almeno 4. Essendo a 4 stadi ed esplicito l'ordine è al più 4. Quindi l'ordine e 4.

4.6 Metodo del Punto Medio Implicito

$1/2$	$1/2$
	1

bisogna riempire la tabella fino a che soddisfa le condizioni di ordine

b			1	OK
c				
$b \otimes c$			$1/2$	OK
$b \otimes c^2$			$1/3$	OK
d				
$b \otimes d$			$1/6$	OK
$b \otimes c^3$			$1/4$	OK
$b \otimes c \otimes d$			$1/8$	OK
e				
c^2				
$e \otimes c^2$			$1/12$	OK
$e \otimes d$			$1/24$	OK

```
riempimento
```

b	1	1	OK
c	1/2		
$b \otimes c$	1/2	1/2	OK
$b \otimes c^2$	1/4	1/4	NO
d			
$b \otimes d$			OK
$b \otimes c^3$			OK
$b \otimes c \otimes d$			OK
e			
c^2			
$e \otimes c^2$			OK
$e \otimes d$			OK

ordine 2.

4.7 Metodo di Crank-Nicolson

0	0	0
1	1/2	1/2
	1/2	1/2

riempimento

b	1/2	1/2	1	OK
c	0	1		
$b \otimes c$	0	1/2	1/2	OK
$b \otimes c^2$	0	1/2	1/2	NO
d				
$b \otimes d$				OK
$b \otimes c^3$				OK
$b \otimes c \otimes d$				OK
e				
c^2				
$e \otimes c^2$				OK
$e \otimes d$				OK

ordine 2.

4.8 Gauss-Legendre del quarto ordine

$\frac{1}{2} - \frac{\sqrt{3}}{6}$	$\frac{1}{4}$	$\frac{1}{4} - \frac{\sqrt{3}}{6}$
$\frac{1}{2} + \frac{\sqrt{3}}{6}$	$\frac{1}{4} + \frac{\sqrt{3}}{6}$	$\frac{1}{4}$
	$\frac{1}{2}$	$\frac{1}{2}$

$$\boldsymbol{d} = \underbrace{\begin{pmatrix} \frac{1}{4} & \frac{1}{4} - \frac{\sqrt{3}}{6} \\ \frac{1}{4} + \frac{\sqrt{3}}{6} & \frac{1}{4} \end{pmatrix}}_{\boldsymbol{A}} \underbrace{\begin{pmatrix} \frac{1}{2} - \frac{\sqrt{3}}{6} \\ \frac{1}{2} + \frac{\sqrt{3}}{6} \end{pmatrix}}_{\boldsymbol{c}} = \underbrace{\begin{pmatrix} \frac{1}{6} - \frac{\sqrt{3}}{12} \\ \frac{1}{6} + \frac{\sqrt{3}}{12} \end{pmatrix}}_{\boldsymbol{d}}$$

$$\boldsymbol{e} = \underbrace{\begin{pmatrix} \frac{1}{4} & \frac{1}{4} + \frac{\sqrt{3}}{6} \\ \frac{1}{4} - \frac{\sqrt{3}}{6} & \frac{1}{4} \end{pmatrix}}_{\boldsymbol{A}^T} \underbrace{\begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \end{pmatrix}}_{\boldsymbol{b}} = \underbrace{\begin{pmatrix} \frac{1}{4} + \frac{\sqrt{3}}{12} \\ \frac{1}{4} - \frac{\sqrt{3}}{12} \end{pmatrix}}_{\boldsymbol{e}}$$

riempimento

$\boldsymbol{b}$	$1/2$	$1/2$	1	OK
$\boldsymbol{c}$	$\frac{1}{2} - \frac{\sqrt{3}}{6}$	$\frac{1}{2} + \frac{\sqrt{3}}{6}$		
$\boldsymbol{b} \otimes \boldsymbol{c}$	$\frac{1}{4} - \frac{\sqrt{3}}{12}$	$\frac{1}{4} + \frac{\sqrt{3}}{12}$	$1/2$	OK
$\boldsymbol{b} \otimes \boldsymbol{c}^2$	$\frac{1}{6} - \frac{\sqrt{3}}{12}$	$\frac{1}{6} + \frac{\sqrt{3}}{12}$	$1/3$	OK
$\boldsymbol{d}$	$\frac{1}{6} - \frac{\sqrt{3}}{12}$	$\frac{1}{6} + \frac{\sqrt{3}}{12}$		
$\boldsymbol{b} \otimes \boldsymbol{d}$	$\frac{1}{12} - \frac{\sqrt{3}}{24}$	$\frac{1}{12} + \frac{\sqrt{3}}{24}$	$1/6$	OK
$\boldsymbol{b} \otimes \boldsymbol{c}^3$	$\frac{1}{8} - \frac{5\sqrt{3}}{72}$	$\frac{1}{8} + \frac{5\sqrt{3}}{72}$	$1/4$	OK
$\boldsymbol{b} \otimes \boldsymbol{c} \otimes \boldsymbol{d}$	$\frac{1}{16} - \frac{5\sqrt{3}}{144}$	$\frac{1}{16} + \frac{5\sqrt{3}}{144}$	$1/8$	OK
$\boldsymbol{e}$	$\frac{1}{4} + \frac{\sqrt{3}}{12}$	$\frac{1}{4} - \frac{\sqrt{3}}{12}$		
$\boldsymbol{c}^2$	$\frac{1}{3} - \frac{\sqrt{3}}{6}$	$\frac{1}{3} + \frac{\sqrt{3}}{6}$		
$\boldsymbol{e} \otimes \boldsymbol{c}^2$	$\frac{1}{24} - \frac{\sqrt{3}}{72}$	$\frac{1}{24} - \frac{\sqrt{3}}{72}$	$1/12$	OK
$\boldsymbol{e} \otimes \boldsymbol{d}$	$\frac{1}{48} - \frac{\sqrt{3}}{144}$	$\frac{1}{48} - \frac{\sqrt{3}}{144}$	$1/24$	OK

4.9 Lobatto IIIB del quarto ordine

0	$1/6$	$-1/6$	0
$1/2$	$1/6$	$1/3$	0
1	$1/6$	$5/6$	0
	$1/6$	$2/3$	$1/6$

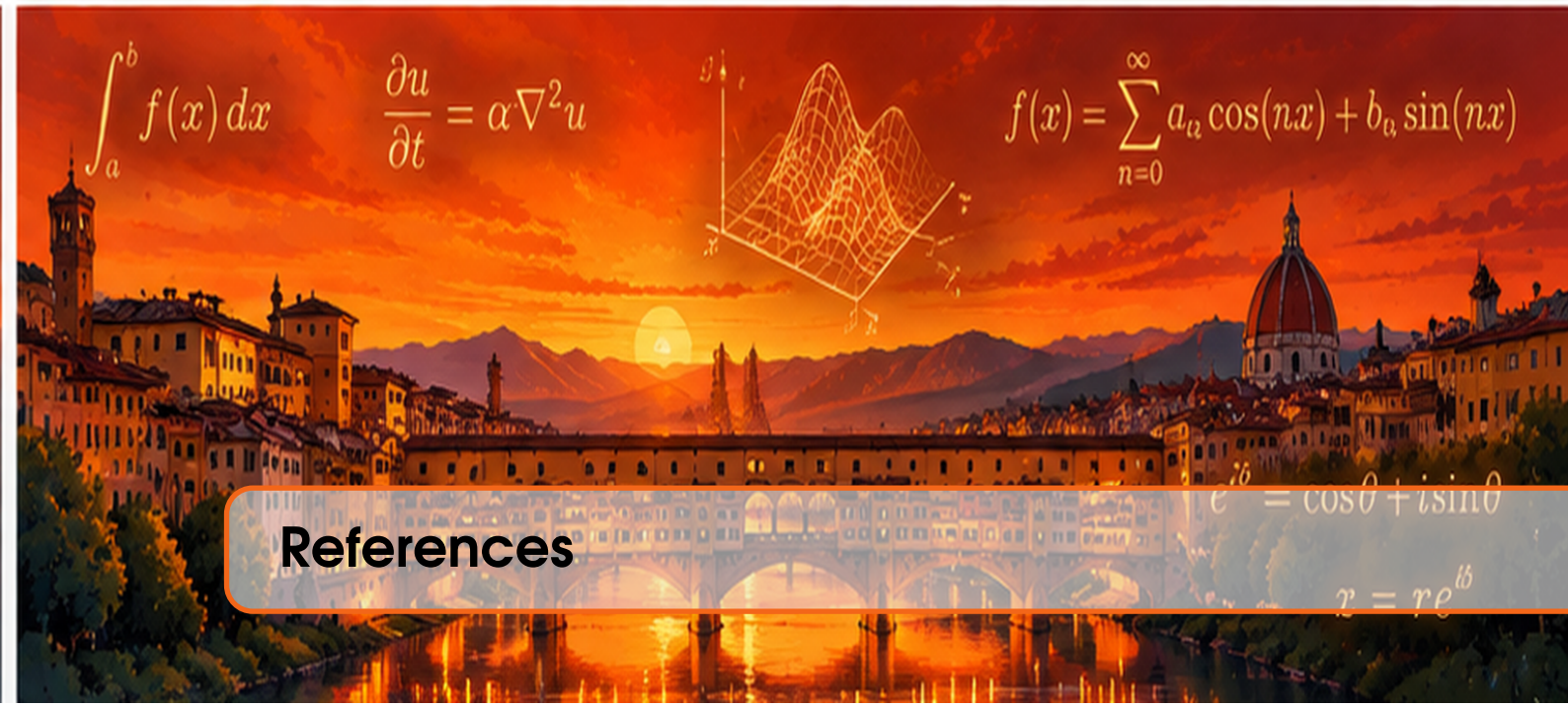
$$\boldsymbol{d} = \underbrace{\begin{pmatrix} 1/6 & -1/6 & 0 \\ 1/6 & 1/3 & 0 \\ 1/6 & 5/6 & 0 \end{pmatrix}}_{\boldsymbol{A}} \underbrace{\begin{pmatrix} 0 \\ 1/2 \\ 1 \end{pmatrix}}_{\boldsymbol{c}} = \underbrace{\begin{pmatrix} -1/12 \\ 1/6 \\ 5/12 \end{pmatrix}}_{\boldsymbol{d}}$$

$$\boldsymbol{e} = \underbrace{\begin{pmatrix} 1/6 & 1/6 & 1/6 \\ -1/6 & 1/3 & 5/6 \\ 0 & 0 & 0 \end{pmatrix}}_{\boldsymbol{A}^T} \underbrace{\begin{pmatrix} 1/6 \\ 2/3 \\ 1/6 \end{pmatrix}}_{\boldsymbol{b}} = \underbrace{\begin{pmatrix} 1/6 \\ 1/3 \\ 0 \end{pmatrix}}_{\boldsymbol{e}}$$

riempimento

$\boldsymbol{b}$	$1/6$	$2/3$	$1/6$	1	OK
$\boldsymbol{c}$	0	$1/2$	1		
$\boldsymbol{b} \otimes \boldsymbol{c}$	0	$1/3$	$1/6$	$1/2$	OK
$\boldsymbol{b} \otimes \boldsymbol{c}^2$	0	$1/6$	$1/6$	$1/3$	OK
$\boldsymbol{d}$	$-1/12$	$1/6$	$5/12$		
$\boldsymbol{b} \otimes \boldsymbol{d}$	$-1/72$	$1/9$	$5/72$	$1/6$	OK
$\boldsymbol{b} \otimes \boldsymbol{c}^3$	0	$1/12$	$1/6$	$1/4$	OK
$\boldsymbol{b} \otimes \boldsymbol{c} \otimes \boldsymbol{d}$	0	$1/18$	$5/72$	$1/8$	OK
$\boldsymbol{e}$	$1/6$	$1/3$	0		
$\boldsymbol{c}^2$	0	$1/4$	1		
$\boldsymbol{e} \otimes \boldsymbol{c}^2$	0	$1/12$	0	$1/12$	OK
$\boldsymbol{e} \otimes \boldsymbol{d}$	$-1/72$	$1/18$	0	$1/24$	OK

ordine almeno 4.



References

[1] G. E. Alefeld, F. A. Potra, e Yixun Shi. 1995. Algorithm 748: Enclosing Zeros of Continuous Functions. *ACM Transactions on Mathematical Software* 21, 3 (1995), 327–344. https://doi.org/10.1145/210089.210111

[2] Peter Colwell. 1993. *Solving Kepler's Equation Over Three Centuries*. Willmann-Bell, Richmond, VA.

[3] James H. Lorie e Leonard J. Savage. 1955. Three Problems in Rationing Capital. *The Journal of Business* 28, 4 (1955), 229–239.



Indice analitico

eliminazione di Gauss, 7

fattorizzazione
    LU, 7
formule di quadratura
    di Gauss-Legendre, 47

metodi di Runge-Kutta, 41
metodo
    delle secanti, 11, 14, 16, 18, 20, 22
    di bisezione, 11, 13, 16, 18, 19, 21
    di Crank-Nicolson, 46
    di Newton, 12, 15, 17, 18, 20, 22

tableau di Butcher, 39