1.1 L'analisi numerica

1.1.1 Introduzione

Il problema di fondo dell'analisi numerica è quello che i calcoli fatti a macchina e anche a mano non sono quasi mai esatti; dipendono da come vengono fatti e da come vengono inseriti i dati. A seconda poi di come vengono fatti possono richiedere più o meno tempo, essere più o meno precisi.

Spesso il tempo e la precisione sono inversamente proporzionali.

L'analisi numerica si propone quindi di elaborare le migliori tecniche di calcolo e di studiare questi fenomeni.

Comunque, anche quando si usano le migliori tecniche, e non ci si preoccupa del tempo, può darsi che i risultati di un calcolo siano comunque imprecisi. Questo può dipendere dalla natura stessa del problema: se il problema è *mal condizionato* i risultati del calcolo sono suscettibili di grandi variazioni a fronte di piccole variazione nei dati iniziali, il che li rende comunque poco affidabili.

L'analisi numerica ha quindi due aspetti strettamente collegati:

- Analisi del problema e tecniche di soluzione
- Analisi dell'errore e tecniche per renderlo minimo.

1.1.2 Alcuni esempi elementari

Esempio 1.1: Tabulare la funzione $f(x) = \dfrac{1+\sqrt{1+x^2}}{\sqrt{1+x^2}}$.

Se si scrive con un computer qualcosa come:

```
f(x)=(1+sqrt(1+x^2))/sqrt(1+x^2)
```

il valore di `sqrt(1+x^2)` viene calcolato due volte, con evidente cattivo impiego del tempo, quindi conviene un approccio in due passi, solo in apparenza più complesso, del tipo:

```
t=sqrt(1+x^2)
f(x)=(1+t)/t
```

Esempio 1.2: Se a, b, c sono numeri reali, allora, come è ben noto, si ha: $a(b+c) = ab + ac$

Però $a(b+c)$ richiede una somma e un prodotto, mentre $ab + ac$ richiede due prodotti e una somma, quindi maggior tempo di calcolo.

Vedremo anche che tra le due espressioni equivalenti $(a+b)+c$ e $a+(b+c)$, che richiedono lo stesso tempo di calcolo, una di esse, in certi casi, è più conveniente dell'altra dal punto di vista numerico e può anche dare risultato differente.

Esempio 1.3: Consideriamo un sistema lineare *quadrato* $Ax = b$ con A matrice invertibile che quindi ha, come ben noto, una e una sola soluzione

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ & \cdots & \cdots & \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \quad b = \begin{pmatrix} b_1 \\ \cdots \\ b_n \end{pmatrix} \text{ cioè } \begin{cases} a_{11}x_1 & +\cdots & +a_{1n}x_n & = & b_1 \\ & \cdots & \cdots & & \\ a_{n1}x_1 & +\cdots & +a_{nn}x_n & = & b_n \end{cases}$$

Ci sono almeno tre metodi elementari per risolverlo:

1 Il noto algoritmo di eliminazione di Gauss che richiede circa $\dfrac{n^3}{3}$ moltiplicazioni.

2 L'uso dell'espressione $x = A^{-1} \cdot b$ che però richiede circa n^3 moltiplicazioni per il solo calcolo di A^{-1}.

3 La nota regola di Cramer: $x_i = \dfrac{\det(A_i)}{\det(A)}$ che richiede circa $\dfrac{n^3}{3}$ moltiplicazioni per incognita se i determinanti delle $n+1$ matrici vengono calcolati con l'algoritmo di Gauss.
Se poi i determinanti vengono calcolati mediante lo sviluppo di Laplace, ognuno richiede circa $n!$ prodotti.

Quindi il metodo in apparenza peggiore, perché richiede un'elaborazione abbastanza complessa (l'algoritmo di Gauss), è in realtà di gran lunga il più conveniente dal punto di vista del tempo di calcolo.

Esempio 1.4: Calcolare la funzione $f(x) = \dfrac{1}{10^5} - \dfrac{1}{x}$ per $x = 10^5 + 1 = 100001$.

Se la nostra calcolatrice si limita a cinque cifre decimali significative, essa sarà in grado di scrivere correttamente $1/10^5$ come 10^{-5}, ma scriverà anche $1/x = 1/(10^5 + 1)$ come 10^{-5} per cui il risultato sarà 0.

Si può scrivere la funzione nel modo equivalente $f(x) = \dfrac{x - 10^5}{10^5 x}$. Quando si sostituisce a x il valore $10^5 + 1$ si ottiene 1 a numeratore e $10^5 \cdot (10^5 + 1)$ a denominatore, ovvero in totale circa 10^{-10}.

La differenza tra un risultato che è 0 e un risultato diverso da zero (benché relativamente piccolo) è relativamente grande e ciò può talora rendere inaffidabile un calcolo.

Esempio 1.5: Calcolare gli integrali definiti $\displaystyle\int_0^1 \frac{x+1}{x^2+1}dx \quad \int_0^1 \frac{\sqrt{1-x^2}}{\sqrt{2-x^2}}dx$

La prima funzione integranda ammette primitiva elementare, e si scrive facilmente

$$\int_0^1 \frac{x+1}{x^2+1}dx = \left[\frac{1}{2}\ln(x^2+1) + \arctan(x)\right]_0^1$$

La funzione $\sqrt{1-x^2}/\sqrt{2-x^2}$ non è integrabile elementarmente, per cui occorreranno metodi approssimati e quindi apparentemente il primo integrale è molto più semplice del secondo.
In realtà, anche se del primo integrale abbiamo una formula esplicita, il computo delle funzioni logaritmo e arcotangente può essere relativamente lungo anche per un computer. Quindi, se non è richiesta un'alta precisione, il calcolo del secondo integrale mediante formule approssimate di quadratura, può risultare più semplice di quello del primo calcolato esplicitamente. Questo senza togliere valore alla formula esplicita che, se disponibile, è importante in svariate questioni.

Esempio 1.6: Tabulare un polinomio.

Sia per esempio $P(x) = 1 + 5x - 2x^2 + 3x^3 + 6x^4$
Scrivere al computer qualcosa come:

```
P(x)=1 +5*x -2*x^2 +3*x^3 +6*x^4
```

non è la cosa più conveniente (4 somme, 4 prodotti, tre potenze)

Leggermente meglio sarebbe

```
P(x)=1 +5*x -2*x*x +3*x*x*x +6*x*x*x*x
```

con solo 4 somme, 10 prodotti (i prodotti richiedono meno tempo di calcolo delle potenze)

Un certo miglioramento si avrebbe mediante l'uso di un'array ausiliaria `t()`

```
t(1)=x  ;  t(2)=t(1)*x  ;  t(3)=t(2)*x  ;  t(4)=t(3)*x
P(x)=1 +5*t(1) -2*t(2) +3*t(3) +6*t(4)
```

con solo 4 somme, 7 prodotti

La procedura migliore è però quella descritta dallo schema di Ruffini-Hörner qui di seguito.

1.1.3 Lo schema di Ruffini-Hörner

Consideriamo come sopra il polinomio $P(x) = 1 + 5x - 2x^2 + 3x^3 + 6x^4$

Lo schema di Ruffini-Hörner consiste nello scrivere il polinomio come

$$1 + x\Big(5 + x\big(-2 + x\,(3 + x \cdot 6)\big)\Big)$$

Calcoliamo per esempio $1 + 5x - 2x^2 + 3x^3 + 6x^4$ per $x_0 = 2$.

$$\begin{array}{rcrcr|lcr}
6 & & & \to & 6 & x_0 \cdot 6 & \to & 12 \\
3 & + & 12 & \to & 15 & x_0 \cdot 15 & \to & 30 \\
-2 & + & 30 & \to & 28 & x_0 \cdot 28 & \to & 56 \\
5 & + & 56 & \to & 61 & x_0 \cdot 61 & \to & 122 \\
1 & + & 122 & \to & 123 & & \Rightarrow & P(2) = 123
\end{array}$$

Richiede solo 4 somme e 4 prodotti.
Non è difficile scrivere la formula generale per un polinomio qualunque.
Vedremo poi che lo schema è utile in altre circostanze.

1.2 Errori

1.2.1 Errore assoluto e errore relativo

Definizione: Se $x \in \mathbb{R}$ e $\tilde{x}$ è il suo valore "calcolato", definiamo

Errore assoluto:	$\tilde{x} - x$	(o meglio $\mid \tilde{x} - x \mid$)
Errore relativo ($x \neq 0$):	$\dfrac{\tilde{x} - x}{x}$	$\left(\text{ o meglio } \dfrac{\mid \tilde{x} - x \mid}{\mid x \mid} \right)$

Osserviamo che, se il risultato di un calcolo è $\tilde{x} \neq 0$ e si sa che $x = 0$ o viceversa, non ha molto senso calcolare l'errore relativo.

Quando non si conosce x, ma si è in grado di calcolare in qualche modo $\tilde{x}$ e si sa maggiorare l'errore assoluto come $\mid \tilde{x} - x \mid < \varepsilon$, si può scrivere $x = \tilde{x} \pm \varepsilon_1$ con $\varepsilon_1 < \varepsilon$; spesso si scrive, con abuso di notazione, semplicemente $x = \tilde{x} \pm \varepsilon$.

Se invece si ha $\dfrac{\tilde{x} - x}{x} < \varepsilon$, si può scrivere $\tilde{x} = x(1 + \varepsilon_1)$ con $\varepsilon_1 < \varepsilon$.

1.2.2 Possibili sorgenti di errore

Quando il risultato di un calcolo è diverso da quello che ci si attende, occorre determinare la sorgente dell'errore. Le principali cause da prendere in considerazione sono:

1. **Modello troppo semplice**. Per esempio voler rappresentare con un modello lineare un fenomeno che è molto più complesso.
2. **Errore nei dati**. Dipendono da informazioni e/o misurazioni.
3. **Errore da arrotondamento o troncamento**. Ne discuteremo più a lungo in seguito. Per esempio sostituendo 1/3 con 0.3333, per quante numerose siano le cifre decimali non si potrà mai avere errore nullo. Scrivendo 0.6667 in luogo di 2/3 si ha un errore inferiore a quello che si ha scrivendo 0.6666.
4. **Errore da cancellazione**. Quando si calcola la differenza $a - b$ tra due numeri positivi a, b molto prossimi, cambia repentinamente l'ordine di grandezza e questo procura spesso errori. Ne discuteremo più a lungo in seguito.
5. **Errore da troncamento del calcolo**. Un algoritmo indefinito di approssimazione deve cessare ad un certo punto senza aver necessariamente raggiunto il risultato esatto. Questo accade in algoritmi tipo quello delle tangenti di Newton o nell'integrazione numerica.
6. **Errore umano** (o più raramente di macchina). La possibilità di aver per esempio toccato un tasto inavvertitamente e aver letto male un dato va sempre presa in considerazione.

1.3 Basi numeriche e rappresentazione di numeri interi

1.3.1 Numeri interi

Fissiamo $b \in \mathbb{N}$, $b \geq 2$, detto *base*.

Proposizione 1 *Sia ora $n \in \mathbb{Z}, n \neq 0$ un qualunque numero intero, allora esiste un'unica* rappresentazione di n in base b, *cioè un'espressione del tipo*

$$n = s \cdot (d_0 + d_1 b + \cdots + d_r b^r) \qquad 0 \leq d_i < b \qquad d_r \neq 0 \qquad s = \pm 1$$

La cifra d_r è detta *cifra più significativa*, mentre la cifra d_0 è detta *cifra meno significativa* del numero n rappresentato in base b. Il numero $s = \pm 1$ è il segno.

La cifra più significativa è sempre diversa da 0. Teniamo presente che il numero 0, che non ha cifre significative, è sempre un caso particolare.

Esempio 1.7: $b = 10$ è la base comunemente usata (solo per motivi storici).
Il numero 4073 si scrive quindi come

$$4073 = 1 \cdot (3 + 7 \cdot 10 + 0 \cdot 10^2 + 4 \cdot 10^3) \qquad d_0 = 3 \quad d_1 = 7 \quad d_2 = 0 \quad d_3 = 4 \neq 0$$

Le basi comunemente usate oltre al 10 sono 2, 8, 16.
In informatica la base fondamentale è 2, ma le rappresentazioni di numeri in base 2 sono di solito molto lunghe, quindi, negli usi pratici si usano la base 8 (ottale) e soprattutto la base 16 (esadecimale), solo per il motivo che è immediato il passaggio dalla rappresentazione in base 2 a quella in base 16 e viceversa, mentre è complicato il passaggio dalla base 10 alla base 2 e viceversa.

Storicamente sono state usate anche la base 60 (in Babilonia, ne abbiamo un ricordo nella divisione di angoli e ore in 60 minuti e secondi) e la base 20. Come curiosità notiamo per esempio che, in francese, 92 si pronuncia $quatre - vingt - douze$ che corrisponde a una rappresentazione in base 20: $92 = 12 + 4 \cdot 20 \qquad d_0 = 12 \quad d_1 = 4 \neq 0.$

Per rappresentare quindi un numero in base b occorrono b simboli che rappresentino le b cifre. In base 10 le cifre sono notoriamente $0, 1, ..., 9$. In base 16 occorrono altre 6 cifre che vengono denotate A, B, C, D, E, F. Vediamo i primi 16 numeri naturali in base 10, 8, 16 e 2.

Base 10	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Base 8	0	1	2	3	4	5	6	7	10	11	12	13	14	15	16	17
Base 16	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Base 2	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111

1.3.2 Rappresentazione di un numero intero in base 2

Ricordiamo brevemente l'algoritmo più semplice per passare dalla usuale rappresentazione di un numero intero in base 10 a quella in base 2.
A titolo di esempio usiamo il numero 1354. Dividiamo successivamente per 2 e tenendo conto del resto:

$1354 : 2$	$=$	677	resto 0
$677 : 2$	$=$	338	resto 1
$338 : 2$	$=$	169	resto 0
$169 : 2$	$=$	84	resto 1
$84 : 2$	$=$	42	resto 0
$42 : 2$	$=$	21	resto 0
$21 : 2$	$=$	10	resto 1
$10 : 2$	$=$	5	resto 0
$5 : 2$	$=$	2	resto 1
$2 : 2$	$=$	1	resto 0
$1 : 2$	$=$	0	resto **1**

La cifra più significativa è in neretto per chiarezza.
La rappresentazione è
$\mathbf{1}0101001010 = 0 + 1 \cdot 2 + 0 \cdot 2^2 + \cdots + \mathbf{1} \cdot 2^{10}$.
È facile passare alla rappresentazione in base 16 dividendo le cifre binarie in gruppi di 4 partendo da destra e assegnando a ogni gruppo di 4 la sua cifra esadecimale:

101	0100	1010
5	4	A

Infatti $1354_{10} = 54A_{16} = 10 + 4 \cdot 16 + 5 \cdot 16^2$

Attenzione: Questo noto algoritmo in realtà non può essere utilizzato da un computer. Il computer lavora in base 2 quindi per eseguire il conto precedente dovrebbe avere il numero già scritto in base 2. Viene quindi usato un altro algoritmo che si basa sullo schema di Ruffini-Hörner.

1.3.3 Conversione macchina di un numero intero in base 2

Usiamo come sopra il numero 1354.
Il computer ha già in memoria la rappresentazione binaria dei numeri da 0 a 10:
$0_{10} = 0000_2 \qquad 1_{10} = 0001_2 \qquad 2_{10} = 0010_2 \qquad \cdots \qquad 10_{10} = 1010_2$

ed è in grado di eseguire le operazioni aritmetiche con i numeri in base 2, quindi il numero decimale 1354 viene scritto, usando lo schema di Ruffini-Hörner, come

$$1354 = 4 + 5 \cdot 10 + 3 \cdot 10^2 + 1 \cdot 10^3 \qquad = \qquad 4 + 10 \cdot (5 + 10 \cdot (3 + 1 \cdot 10))$$

Dato che nell'ultima espressione compaiono solo numeri compresi tra 0 e 10 di cui il computer conosce la rappresentazione binaria e con cui è in grado di eseguire i conti, questo permette di determinare la rappresentazione binaria del numero.

Osservazione: Il passaggio dalla rappresentazione binaria a quella decimale, necessario per visualizzare il risultato di un calcolo eseguito dal computer in base 2, è ancora più complesso. Occorre dividere il numero successivamente per 10 (in binario 1010). Le cifre decimali si ricavano a partire dalla meno significativa come resto delle divisioni; la conversione termina quando l'ultimo quoziente è 0. Dato che una divisione richiede al computer un tempo più che doppio rispetto a un prodotto, questa conversione è di norma più onerosa.

1.3.4 Rappresentazione macchina di un numero intero

Vedremo tra poco come il computer rappresenta internamente un numero reale qualunque, ma spesso, quando ha a che fare solo con numeri interi, dopo averli convertiti in base 2, la macchina li può rappresentare, a seconda delle esigenze, usando un byte (8 bit), due byte (16 bit) oppure quattro byte (32 bit) (o anche 11 bit, come vedremo nella rappresentazione dell'esponente di un numero reale).

Descriviamo per semplicità la rappresentazione con un solo byte (che comunque ha evidentemente uso limitato).

Se non usiamo numeri negativi, mediante un byte (8 bit) si possono rappresentare 2^8, cioè 256 numeri, quelli da 0 a 255.

Se invece vogliamo rappresentare anche i negativi, ci sono due possibilità:

Numeri e segno: Si usano 7 bit per il numero e un bit per il segno. Si possono rappresentare i numeri da -127 a 127 e ovviamente non ha senso il numero 0 col segno meno, quindi si possono rappresentare 255 numeri. Questa rappresentazione è però scarsamente usata.

Numeri con segno: I numeri negativi vengono rappresentati in forma complementare col primo bit uguale a 1.

Il più grande numero positivo rappresentabile è 127 che, in forma binaria, è [0111 1111]. Il numero -1 viene rappresentato come [1111 1111]; infatti la sua somma con [0000 0001] è [0000 0000] (dato che va perso il nono bit di riporto). Il numero più piccolo è -128 che si scrive come [1000 0000]. E in effetti $127 + (-128)$ dà -1.

Usando due byte, i numeri rappresentabili vanno da -32768 a 32767 (*short integers*).

Usando quattro byte: da -2147483648 a 2147483647 (*long integers*).

1.4 Basi numeriche e rappresentazione di numeri reali

1.4.1 Numeri reali

La scelta della base numerica influisce in modo notevole sulla rappresentazione dei numeri reali non interi.

Per esempio il numero reale 1/3 ha la nota rappresentazione decimale 0.3333... ovvero $0.\overline{3}$ (periodico), quindi non potrà mai essere scritto in modo esatto in base 10. Usando per esempio la base 12 (di raro uso) la sua rappresentazione sarebbe 0.4 (non periodico), cioè una rappresentazione esatta.

Viceversa il numero reale 1/10 ha la rappresentazione decimale esatta 0.1 (non periodico), ma in base 2 la sua rappresentazione è $0.0\,0011\,0011\,00... = 0.0\,\overline{0011}$ (periodico).

Per chiarire questi concetti introduciamo la rappresentazione dei numeri reali *a virgola variabile* (*floating-point representation* in inglese).

Proposizione 2 *Fissiamo la* base *b, dove* $b \in \mathbb{N}$ *e* $b \geq 2$.
Sia $x \in \mathbb{R}, x \neq 0$ *un numero reale non nullo, allora esiste un'unica* rappresentazione di x in base b, *cioè un'espressione del tipo*

$$x = s \cdot (d_1 \cdot b^{-1} + d_2 \cdot b^{-2} + \cdots)\, b^p$$

$s = \pm 1$	è detto *segno*
$p \in \mathbb{Z}$	è un intero detto *caratteristica* o *esponente*
$m = d_1, d_2, d_3, ...$	è una successione (spesso infinita) detta *mantissa*
	I numeri d_i sono numeri interi tali che $0 \leq d_i < b$.

Nella successione m si ha $d_1 \neq 0$ e d_1 è detto *prima cifra significativa.*
La successione m non è mai definitivamente $b-1\,,\ b-1\,,\ b-1\,, ...$
A volte viene detta mantissa non la successione, ma la sommatoria $m = d_1 \cdot b^{-1} + d_2 \cdot b^{-2} + \cdots$ che, quando è infinita, è sempre una serie di potenze convergente.
La mantissa m così definita è un numero compreso tra $1/b$ e 1: più precisamente $1/b \leq m < 1$.

Esempio 1.8: Qualche esempio in base 10 per familiarizzare:

$24.31 =$	$1 \cdot (0.2431) \cdot 10^2$	segno 1	mantissa 2 4 3 1	(o 0.2431)	caratt. 2
$-0.0349 =$	$-1 \cdot (0.349) \cdot 10^{-1}$	segno -1	mantissa 3 4 9	(o 0.349)	caratt. -1
$1/3 =$	$1 \cdot (0.333...) \cdot 10^0$	segno 1	mantissa 3 3 3 ...	(o 0.333...)	caratt. 0
$125\,000\,000 =$	$1 \cdot (0.125) \cdot 10^9$	segno 1	mantissa 1 2 5	(o 0.125)	caratt. 9

La rappresentazione $1 \cdot (0.29999999...) \cdot 10^2$ non è valida perché le cifre della mantissa sono tutte $b - 1 = 9$ da un certo punto in poi; il numero periodico $29.\overline{9} = 29.999...$ è infatti una rappresentazione errata del numero 30, dato che la mantissa, intesa come sommatoria, vale 0.3.

Esempio 1.9: Determiniamo la mantissa di $(0.9)_{10}$ usando la base 16.

Poniamo innanzitutto $a_1 = (0.9)_{10} = 1 \cdot (d_1 \cdot 16^{-1} + d_2 \cdot 16^{-2} + \cdots)$.

Si ha: $a_1 = (0.9)_{10} = \dfrac{d_1}{16} + \dfrac{d_2}{16^2} + \cdots$. Moltiplichiamo per 16:

$16 \cdot a_1 = d_1 + \dfrac{d_2}{16} + \cdots$. Ma $16 \cdot a_1 = 16 \cdot 0.9 = 14.4$ quindi:

$16 \cdot 0.9 = 14.4 = 14 + 0.4 = d_1 + \dfrac{d_2}{16} + \dfrac{d_3}{16^2} + \cdots$ pertanto $d_1 = (14)_{10} = E_{16}$

Poniamo quindi $a_2 = (0.4)_{10} = \dfrac{d_2}{16} + \dfrac{d_3}{16^2} + \cdots$. Moltiplichiamo per 16:

$16 \cdot a_2 = d_2 + \dfrac{d_3}{16} + \cdots$. Ma $16 \cdot 0.4 = 6.4$ da cui $d_2 = (6)_{10} = 6_{16}$ e così via.

In definitiva $(0.9)_{10} = 0.E6666\cdots$ e quindi $(0.9)_{10} = 1 \cdot (0.E66\cdots) \cdot (16)^1_{10}$
È facile passare alla rappresentazione binaria e vedere la periodicità dello sviluppo del numero:
$(0.9)_{10} = 0.1110\,0110\,0110\cdots = 0.1\,1100\,1100\,1100\cdots = 1 \cdot (0.1\,\overline{1100}) \cdot 10^0$

Osservazione 1: Come nel caso dei numeri interi, l'algoritmo dell'esempio precedente non è eseguibile dal computer, che dovendo lavorare in base 2, esegue di fatto la divisione tra 9 e 10 dopo aver rappresentato 9 e 10 in base 2.

In effetti, quando deve rappresentare per esempio il numero 1234.56, il computer converte in binario il numero intero 123456 e poi lo divide (usando vari accorgimenti) per 100.

Osservazione 2: La rappresentazione a virgola variabile è analoga alla cosiddetta notazione scientifica, usata in fisica e in tecnica e anche sul display delle macchine calcolatrici specialmente per numeri molto grandi o molto piccoli.
La differenza è che nella notazione scientifica si pone la cifra più significativa prima della virgola invece che dopo, quindi l'esponente in notazione scientifica è inferiore di un'unità.
Vediamo la differenza nei quattro esempi precedenti (si suppone una calcolatrice con visore a 8 cifre):

Numero		virgola variabile		notazione scientifica		visore calcolatrice
24.31	=	$1\cdot(0.2431)\cdot10^{2}$	=	2.431×10^{1}	=	24.31
-0.0349	=	$-1\cdot(0.349)\cdot10^{-1}$	=	-3.49×10^{-2}	=	-0.0349
$1/3$	=	$1\cdot(0.333...)\cdot10^{0}$	=	3.3333333×10^{-1}	=	0.3333333
125 000 000	=	$1\cdot(0.125)\cdot10^{9}$	=	1.25×10^{8}	=	1.25 E08

1.4.2 Numeri macchina

Mentre i numeri reali sono infiniti, i numeri macchina disponibili sono in numero finito, quindi, a seconda dell'architettura e dei limiti della macchina, occorre fissare i seguenti numeri interi positivi

- $b \geq 2$, la *base*.
- t il numero di cifre della mantissa.
- $[L, U]$ il *range* (minimo e massimo esponente consentiti).

I numeri macchina sono numeri del tipo

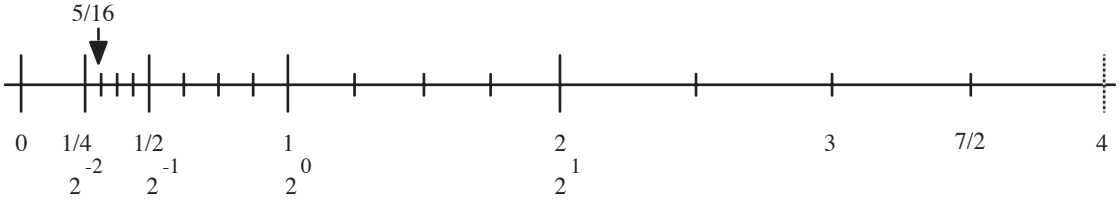
$$x = s \cdot (d_1 \cdot b^{-1} + d_2 \cdot b^{-2} + \cdots + d_t \cdot b^{-t}) b^p$$

cioè numeri a virgola variabile in cui però $L \leq p \leq U$ e la mantissa ha un numero fissato t di cifre.

Per capire come sono disposti i numeri macchina, li elenchiamo tutti supponendo, solo per ragioni di semplicità descrittiva, che:
$b = 2 \qquad t = 3 \qquad L = -2 \qquad U = 1$
I numeri rappresentabili con queste limitazioni sono solo 32 (33 se comprendiamo anche lo zero) e cioè $\pm 0.d_1d_2d_3 \cdot 10^p$ (10 è il numero 2 in rappr. binaria).
Si deve avere $d_1 = 1$ (cifra più significativa), mentre d_2, d_3 possono valere 0 o 1. Inoltre $-2 \leq p \leq 1$.

Per esempio:	$0.100\cdot10^{-1}$	=	$1/4_{10}$	= 2_{10}^{-2}	il più piccolo positivo
	$0.101\cdot10^{-1}$	=	$5/16_{10}$		
	$0.111\cdot10^{10}$	=	$7/2_{10}$		il più grande positivo
	0.000	=	0		eccezione

È possibile disegnare tutti i 16 numeri positivi (scritti qui in decimale):



Come si vede non sono egualmente spaziati sulla retta reale, ma tra 2^t e 2^{t+1} sono equidistanti. Notiamo inoltre il grosso vuoto tra 0 e il minimo numero macchina.

Per memorizzare questi numeri occorrono 6 bit: 3 bit per la mantissa, 1 bit per il segno e 2 bit per l'esponente che può assumere quattro valori: $00 = 0_{10}$; $01 = 1_{10}$; $10 = -2_{10}$; $11 = -1_{10}$, gli ultimi due rappresentati come interi con segno, in forma complementare.
Teniamo presente che la prima cifra della mantissa è necessariamente 1. Unica eccezione è il numero 0 che viene rappresentato con mantissa tutta nulla.

I numeri a doppia precisione: Nella pratica, una delle rappresentazioni macchina più usate è quella dei cosiddetti numeri *a doppia precisione*. La base è 2 e ogni numero è memorizzato con 8 byte cioè con 64 bit. Solitamente la suddivisione dei bit è la seguente:
• 1 bit per il segno
• 11 bit per la caratteristica che è un intero con segno. Quindi il range va da $-2^{-10} = -1024$ a $2^{10} - 1 = 1023$.
• 6 byte e mezzo per la mantissa che ha quindi al massimo 52 cifre.

In questo modo il massimo numero rappresentabile è 2^{1024} "meno un bit" (vedi più avanti la definizione del numero eps) che è circa 1.7977×10^{308} e le 52 cifre binarie della mantissa consentono di rappresentare i numeri in base 10 con circa 16 cifre decimali.
Sono rappresentabili in modo esatto tutti i numeri interi positivi fino a 2^{53}; il numero $2^{53}+1$ è il primo intero positivo non rappresentabile esattamente.

I numeri BCD: Benché la cosa sembri a prima vista strana, è possibile rappresentare i numeri reali in macchina usando la base 10 anziché la base 2. Questi numeri sono comunemente detti BCD (*binary coded decimals*). Si usano, come nella doppia precisione, 6 byte e mezzo per la mantissa. Ogni mezzo byte è una cifra della mantissa decimale (e quindi può assumere valori solo da 0000 a 1001). In questo modo si possono memorizzare 13 cifre decimali. Ci sono meno numeri rappresentabili rispetto alla doppia precisione e il tempo di calcolo è leggermente superiore, ma, da un certo punto di vista, si ha maggior precisione perché non occorre una doppia conversione di base. Comunque attualmente i BCD sono di uso sempre meno frequente.

1.4.3 Rappresentazione in macchina di un numero reale

Sia $x \in \mathbb{R}$, $x \neq 0$; rappresentiamo il numero x con segno s, mantissa $m = d_1\,, d_2\,, \ldots$ e caratteristica p. Una volta fissati i parametri $b, t, [L, U]$, ci sono 4 possibilità:

1. Si ha $L \leq p \leq U$ e $d_i = 0$ per $i > t$. Quindi è possibile rappresentare x esattamente in macchina.
2. $p > U$: *overflow*. È praticamente impossibile rappresentare il numero. Diverse macchine si arrestano ed emettono segnale di errore. Altre restituiscono un numero speciale **`Inf`** (o $-$**`Inf`** se x è negativo) che significa appunto numero con esponente maggiore di U. Se questo non succede, ciò comporta grave errore e risultato inaffidabile.
3. $p < L$: *underflow*. Alcune macchine si arrestano ed emettono segnale di errore. Altre sostituiscono x con 0, ma in certi casi ciò è pericoloso perché sostituire un numero, anche molto piccolo con 0, significa generare un errore relativo teoricamente infinito. Si tenga anche presente che la distanza tra 0 e il minimo numero macchina è relativamente grande.
4. $L \leq p \leq U$, ma la mantissa ha più di t cifre (spesso è infinita).
   Per esempio $1/3_{10} = 0,010101\cdots \cdot 2_{10}^{-1}$ o meglio $1/3_{10} = 0,10101\cdots \cdot 2_{10}^{-2}$.
   Questo è di gran lunga il caso più frequente. Il numero x non può essere rappresentato esattamente, quindi occorre decidere in che modo approssimarlo con un numero macchina e sapere quale errore si commette.

1.4.4 Arrotondamento e troncamento

Siamo nel caso 4.

Per semplicità supponiamo $x > 0$. Per rappresentare x in macchina ci sono due tecniche.

- *Troncamento*: si omettono nella mantissa tutte le cifre oltre la t-esima.
- *Arrotondamento*: provvisoriamente si tronca la mantissa oltre la $(t+1)$-esima cifra, ma, visto che al momento della memorizzazione le cifre devono essere t, si scrive come mantissa $d_1 \cdots d_{t+1} + \frac{1}{2}\, b^{-(t+1)}$ e si tronca alla t-esima cifra.

Esempio 1.10: Un tipico arrotondamento in base 10:

$\frac{1}{3} = 0.3333\cdots$ troncamento e arrotondamento coincidono.

$\frac{2}{3} = 0.6666\cdots$ il troncamento è $0.66666\cdots 6$, l'arrotondamento è

$0.666\cdots 666|6 +$
$0.000\cdots 000|5 =$
$0.666\cdots 667|1$ quindi $2/3$ viene rappresentato come $0.666\cdots 667 \cdot 10^0$.

Esempio 1.11: Un arrotondamento in base 2:

Il numero decimale 3/7 ha la rappresentazione binaria infinita periodica $0.011\,011 = 0.\overline{011}$. Volendolo rappresentare con un numero finito di cifre binarie dopo il punto mediante arrotondamento si ottiene

con 3 cifre binarie	con 4 cifre binarie	con 5 cifre binarie
$0.011\|0 +$	$0.0110\|1 +$	$0.01101\|1 +$
$\underline{0.000\|1 =}$	$\underline{0.0000\|1 =}$	$\underline{0.00000\|1 =}$
$0.011\|1$	$0.0111\|0$	$0.01110\|0$
quindi 0.011	quindi 0.0111	quindi 0.01110

Come vedremo al paragrafo successivo, l'arrotondamento, benché leggermente più complesso, è di norma preferito al troncamento perché l'errore risulta dimezzato.

1.5 Errori macchina

1.5.1 La precisione macchina

Poniamo la seguente

Definizione: Fissati i parametri $b, t, [L, U]$ e un metodo di rappresentazione tra troncamento e arrotondamento, si indica con

$$\mathrm{fl}(x)$$

la rappresentazione macchina del numero reale x

L'abbreviazione fl sta per *floating*.

La proposizione che segue fornisce una maggiorazione dell'errore relativo commesso sostituendo x con $\mathrm{fl}(x)$:

Proposizione 3 *Se non c'è overflow, allora*

$$\left|\frac{\mathrm{fl}(x)-x}{x}\right| \le b^{1-t} \qquad \left|\frac{\mathrm{fl}(x)-x}{x}\right| \le \frac{1}{2}\, b^{1-t}$$

La prima in caso di troncamento, la seconda in caso di arrotondamento.

Il numero $\frac{1}{2}\, b^{1-t}$ (o b^{1-t} se si usa il troncamento) è denotato eps ed è detto *precisione macchina*. Conviene però definirlo in modo indipendente da troncamento o arrotondamento.

Definizione: E' detto eps il più piccolo numero tale che

$$\mathrm{fl}(1+\mathrm{eps}) > 1$$

Il numero eps non è il minimo numero rappresentabile in macchina, ma è la maggiorazione dell'errore relativo commesso rappresentando un numero in macchina. Si ha infatti:

$$\left|\frac{\mathrm{fl}(x)-x}{x}\right| \le \mathrm{eps} \qquad \text{che si può scrivere} \qquad \mathrm{fl}(x) = x(1+\varepsilon) \qquad \text{dove} \qquad |\,\varepsilon\,| \le \mathrm{eps}$$

Esempio 1.12: In base 10. Sia $\pi = 3.14159\cdots$.

Se tronchiamo a 4 cifre dopo il punto $\left|\frac{\pi - 3.1415}{\pi}\right| \le \frac{1}{10^4}$

Se invece arrotondiamo a 4 cifre dopo il punto $\left|\frac{\pi - 3.1416}{\pi}\right| \le \frac{1}{2}\frac{1}{10^4}$

Esempio 1.13: Il maggior numero rappresentabile in macchina usando i numeri a doppia precisione è $(2 - \text{eps}) \cdot 2^{1023}$
Inoltre, usando numeri in doppia precisione e arrotondamento si ha eps$= 2^{-52} \simeq 2.22 \cdot 10^{-16}$.

Una semplice routine per il calcolo di eps:

```
eps = 1
while 1+ eps > 1
        eps = eps /2
end
eps = eps *2
```

1.5.2 Operazioni macchina

Date le 4 operazioni aritmetiche elementari, definiamo le corrispondenti operazioni macchina in questo modo:

$x \oplus y = \text{fl}(x + y)$ $\qquad$ $x \ominus y = \text{fl}(x - y)$
$x \otimes y = \text{fl}(x \cdot y)$ $\qquad$ $x \oslash y = \text{fl}(x/y)$

Gli input x, y saranno numeri macchina, ma non è detto che il risultato delle operazioni usuali $x+y$ etc. sia un numero macchina, per cui spesso $x \oplus y$ è diverso da $x + y$ e così per le altre operazioni. In effetti, si ha per esempio $x \oplus y = (x + y)(1 + \varepsilon) = x(1 + \varepsilon) + y(1 + \varepsilon)$ con $\varepsilon <$ eps.

Le quattro operazioni macchina non godono sempre di proprietà elementari valide nelle operazioni usuali. Per esempio, spesso si ha $(x \oplus y) \oplus z \neq x \oplus (y \oplus z)$.

Possono succedere cose abbastanza strane tipo il fatto che $x \oplus y = x$ se $\mid y \mid < \dfrac{\text{eps}}{b} \mid x \mid$, cioè se si somma a x un numero al di là della precisione macchina in confronto a x.

Per esempio in base 10 con 4 cifre decimali e range abbastanza grande si ha: $1 + 0.00002 = 1$.
Il numero 0.00002 non è al di fuori dei numeri rappresentabili in macchina (è $0.2 \cdot 10^{-4}$), ma è troppo piccolo in confronto a 1 o meglio è oltre la precisione della macchina in confronto a 1.
Da questo esempio si intuisce il fenomeno della *cancellazione*.

1.5.3 La cancellazione

La cancellazione è una tra le più frequenti sorgenti di errore nelle operazioni macchina.

Esempio 1.14: Lavoriamo in base $b = 10$ con $t = 8$ cifre. Siano

$a = 0.23371258 \cdot 10^{-4}$ $\qquad$ $b = 0.33678429 \cdot 10^{2}$ $\qquad$ $c = -0.33677811 \cdot 10^{2}$

Calcoliamo
$(a \oplus b) \oplus c = 0.33678452 \cdot 10^{2} \ominus 0.33677811 \cdot 10^{2} = 0.6410000 \cdot 10^{-3}$
$a \oplus (b \oplus c) = 0.23371258 \cdot 10^{-4} \oplus 0.6180000 \cdot 10^{-3} = 0.64137126 \cdot 10^{-3}$

Il risultato esatto è $a + b + c = 0.64137126 \cdot 10^{-3}$.

Nel primo conto la cancellazione è avvenuta alla seconda somma, ma il primo addendo aveva già subito conversione a numero macchina, quindi con perdita di dati. Nel secondo caso la cancellazione è avvenuta subito tra due numeri vicini e quindi con minore perdita di dati. Meglio quindi prima sommare i due numeri di grandezza simile.

Un altro esempio famoso

Esempio 1.15: Scriviamo il noto sviluppo di MacLaurin $e^x = 1 + x + \dfrac{x^2}{2!} + \dfrac{x^3}{3!} + \cdots$ e usiamolo per calcolare e^{-30}: $e^{-30} = 1 - 30 + \dfrac{900}{2} - \dfrac{27000}{6} + \cdots$

Nella sommatoria ci sono numeri di diverso ordine di grandezza, per cui si verifica la cancellazione. È meglio calcolare nel seguente modo:

$e^{30} = 1 + 30 + \dfrac{900}{2} + \dfrac{27000}{6} + \cdots$ e poi eseguire $e^{-30} = 1/e^{30}$.

Calcolando con un computer ci si può render conto di come il primo metodo porti a gravi errori di conto dopo un certo numero di passi.

Cerchiamo di dare una possibile spiegazione teorica del fenomeno della cancellazione:
Siano a, b due numeri reali e poniamo $\tilde{a} = \mathrm{fl}(a)$, $\tilde{b} = \mathrm{fl}(b)$. Si ha

$$\tilde{a} = a(1+\varepsilon_1) \qquad \tilde{b} = b(1+\varepsilon_2) \qquad \tilde{a} \oplus \tilde{b} = (\tilde{a} + \tilde{b})(1+\varepsilon) \qquad \text{con } \mid \varepsilon, \varepsilon_1, \varepsilon_2 \mid < \text{eps}$$

Vogliamo calcolare δ, errore relativo tra $\tilde{a} \oplus \tilde{b}$ e $a + b$, cioè $\delta = \dfrac{(\tilde{a} \oplus \tilde{b}) - (a+b)}{a+b}$

Si ha: $\delta = \dfrac{(\tilde{a} + \tilde{b})(1+\varepsilon) - (a+b)}{a+b} = \cdots = \varepsilon + \left(\dfrac{a\varepsilon_1 + b\varepsilon_2}{a+b}\right)(1+\varepsilon) \qquad \text{con } \mid \varepsilon, \varepsilon_1, \varepsilon_2 \mid < \text{eps}$

Quindi $\mid \delta \mid < \text{eps} + (1 + \text{eps})\ \text{eps}\ \dfrac{\mid a \mid + \mid b \mid}{\mid a + b \mid}$

Questo spiega il fenomeno della cancellazione che si verifica quando a e b sono di segno discorde, ma molto prossimi in valore assoluto, perché $\mid a+b \mid$ può essere molto piccolo rendendo $\mid \delta \mid$ grande.

Esempio 1.16: $a = 0.123456 \qquad b = -0.123454$. Se $t = 5$ (numero di cifre decimali), allora
$\tilde{a} = 0.12346 \qquad \tilde{b} = -0.12345 \qquad a + b = 0.2 \cdot 10^{-5} \qquad \tilde{a} \oplus \tilde{b} = 0.1 \cdot 10^{-4}$
Quindi $\delta = \dfrac{(\tilde{a} \oplus \tilde{b}) - (a+b)}{a+b} = 4$

Esempio 1.17: Equazione di secondo grado. Sia $ax^2 - 2bx + c = 0$ una semplice equazione di grado 2, allora le soluzioni sono notoriamente

$$x_1 = \frac{b - \sqrt{b^2 - 4ac}}{a} \qquad x_2 = \frac{b + \sqrt{b^2 - 4ac}}{a}$$

Supponiamo $b > 0$. Se c è prossimo a 0, allora in x_1 c'è una cancellazione, per cui l'errore può essere anche elevato. Conviene allora calcolare prima x_2 che non ha cancellazione e poi porre $x_1 = \dfrac{c}{b + \sqrt{b^2 - 4ac}}$, ovvero $x_1 = x_2 \cdot a \cdot c$.

1.5.4 L'errore nelle operazioni macchina

Abbiamo già visto che, se $\tilde{a} = a(1+\varepsilon_1) \qquad \tilde{b} = b(1+\varepsilon_2)$, l'errore nella somma è inferiore a $\text{eps} + (1 + \text{eps})\ \text{eps}\ \dfrac{\mid a \mid + \mid b \mid}{\mid a + b \mid}$.

È possibile effettuare un conto analogo per il prodotto e si scopre che

$$\delta = \frac{\tilde{a} \otimes \tilde{b} - a \cdot b}{a \cdot b} = (1+\varepsilon_1)(1+\varepsilon_2) - 1 = \varepsilon_1 + \varepsilon_2 + \varepsilon_1\varepsilon_2 \simeq \varepsilon_1 + \varepsilon_2$$

Un conto analogo per la divisione mostra che

$$\delta = \frac{\tilde{a} \oslash \tilde{b} - a/b}{a/b} = \frac{\varepsilon_1 - \varepsilon_2}{1+\varepsilon_2} \simeq \varepsilon_1 - \varepsilon_2$$

La conclusione è che nel prodotto e nella divisione c'è più controllo dell'errore rispetto a quanto avviene con le somme dove si può verificare il fenomeno della cancellazione.

2.1 Equazioni non lineari

Capita sovente di dover risolvere un'equazione in un'incognita

$$f(x) = 0$$

dove $f(x)$ è un qualche funzione più o meno semplice.
Ci occuperemo dei vari modi di approssimare una soluzione, una volta stabilito che ne esista almeno una in un qualche sottoinsieme di $\mathbb{R}$.

2.1.1 Il metodo di bisezione

Il più semplice algoritmo è quello ben noto di bisezione.

Se $f(x)$ è continua in un intervallo $[a, b]$ e $f(a) \cdot f(b) < 0$ (ovvero assume valori di segno discorde negli estremi), allora, per il noto teorema degli zeri, nell'intervallo $[a, b]$ esiste almeno un x_0 tale che $f(x_0) = 0$; se poi $f(x)$ è strettamente monotona, il numero x_0 è unico.

Per approssimare x_0 si divide l'intervallo a metà: $\left[a\,,\ \frac{a+b}{2}\,,\ b\right]$ e si sostituisce l'intervallo $[a, b]$ con quello tra i due intervalli $\left[a\,,\ \frac{a+b}{2}\right]$ e $\left[\frac{a+b}{2}\,,\ b\right]$ nel quale la funzione ha ancora valori discordi negli estremi e così via.

È praticamente rarissimo che a un certo punto il punto medio dell'intervallo sia proprio lo zero cercato, per cui l'algoritmo normalmente non ha termine e occorre arrestarlo a un certo punto mediante un qualche criterio. Il criterio di solito è uno di questi tre

- Dopo un certo numero p di passi.
- Quando $| \tilde{x} - x_0 | < \varepsilon$ con ε valore prefissato ($\tilde{x}$ il valore trovato in quel momento).
- Quando $f(\tilde{x}) < \varepsilon$ con ε valore prefissato.

Una delle peculiarità dell'algoritmo di bisezione è il fatto che tra i primi due criteri c'è una relazione.

Si ha: $\varepsilon < \frac{| b-a |}{2^{p+1}}$, quindi $p < 1 + \log_2 \frac{| b-a |}{\varepsilon}$.

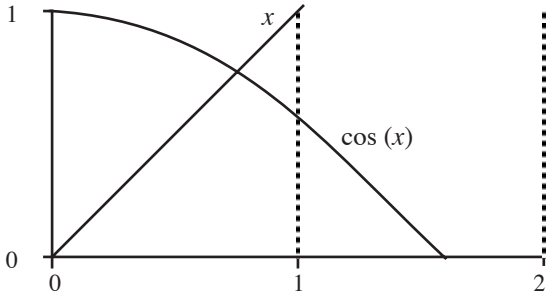
Invece per quanto riguarda il terzo criterio, è a priori impossibile prevedere il numero di passi, a meno di non avere informazioni sulla derivata di $f(x)$ (se esiste).

L'algoritmo è relativamente lento, dato che $\log_2(10) \simeq 3.32$, quindi occorrono circa tre passi per avere una cifra decimale esatta; in compenso, nelle ipotesi date, l'algoritmo converge sicuramente ed è di facilissima implementazione.

2.1.2 L'algoritmo di punto fisso

Cominciamo con un semplice esempio facilmente eseguibile con calcolatrice scientifica tascabile.

Esempio 2.1: Risolvere l'equazione $x = \cos(x)$ ($\cos(x)$ in radianti !).
Dalla figura si può dedurre che $x_0 \simeq 0.7$, e quindi calcoliamo $\cos(0.7) = 0.7648$. Di seguito calcoliamo $\cos(0.7648)$ e così via



$$\begin{aligned}
\cos(0.7) &= 0.7648\\
\cos(0.7648) &= 0.7215\\
\cos(0.7215) &= 0.7508\\
\cos(0.7508) &= 0.7311\\
\cos(0.7311) &= 0.7444\\
\cos(0.7444) &= 0.7355\\
&\text{venti volte...}\\
\cos(0.7391) &= 0.7391
\end{aligned}$$

Quindi dopo una ventina di passi si trovano in modo elementare almeno quattro cifre decimali esatte della soluzione del problema.

Definizione: Si dice che α è un punto fisso della funzione $f(x)$ se $f(\alpha) = \alpha$

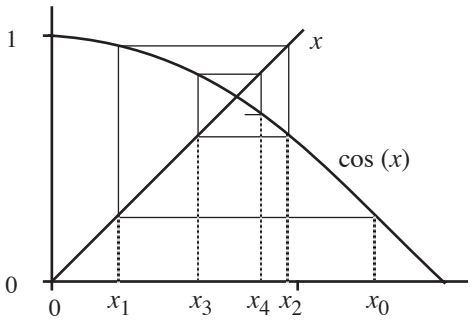
Nell'esempio $\alpha = 0.7391$ è un punto fisso di $\cos(x)$ nell'intervallo $[0, 1]$ e l'algoritmo consente di approssimarlo facilmente.
Benché sembri un caso particolare del problema iniziale di risolvere un'equazione $f(x) = 0$, l'algoritmo di punto fisso è alla base di molti altri metodi.

Proposizione 4 *Sia $g(x)$ continua nell'intervallo $[a, b]$ e sia $x_0 \in [a, b]$. Sia poi $x_0, x_1, ...$ la successione determinata dall'algoritmo di punto fisso, cioè definita come $x_i = g(x_{i-1})$. Supponiamo inoltre che $x_i \in [a, b]$ per ogni i.*
Se la successione converge e $\lim_{i\to\infty} x_i = \alpha$, allora α è punto fisso di $g(x)$.

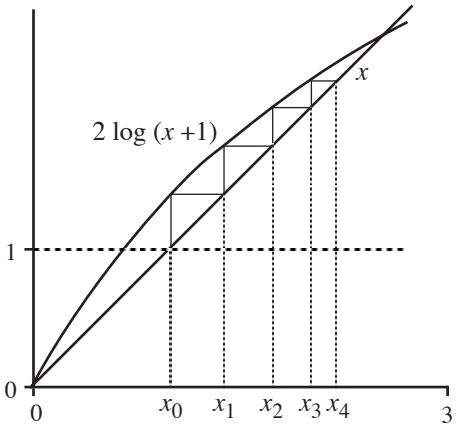
Non sempre la successione è convergente, però basta una condizione sulla derivata di $g(x)$:

Proposizione 5 *(condizione sufficiente) Sia α un punto fisso di $g(x)$ e supponiamo che $g(x)$ sia derivabile in un intervallo $I = [\alpha - \varrho\ ,\ \alpha + \varrho]$ con $\varrho > 0$. Se $|\ g'(x)\ | < 1$ in I e $x_0 \in I$, allora la successione $x_i = g(x_{i-1})$ determinata dall'algoritmo di punto fisso è convergente e si ha $\lim_{i\to\infty} x_i = \alpha$. Inoltre α è unico nell'intervallo.*

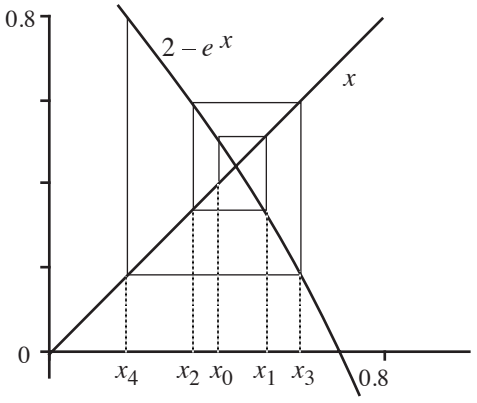
Graficamente:



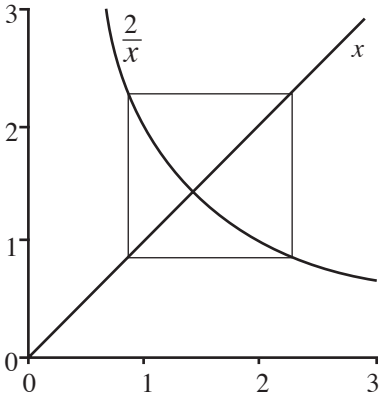
Equazione $x = \cos(x)$ come sopra, ma partendo con un x_0 lontano da α per chiarezza; la successione converge a segno alterno ad α.



Equazione $x = 2 \cdot \log(x + 1)$ con $x_0 = 1$. La successione converge monotonamente ad α, ma la convergenza è lenta.



Equazione $x = 2 - e^x$. La successione diverge anche partendo da x_0 prossimo ad α perché la derivata è in modulo maggiore di 1.



Equazione $x^2 = 2$ che può essere pensata come punto fisso di $f(x) = 2/x$, ma la successione è stabile e non converge perché la derivata in α è proprio -1.

2.1.3 Criteri d'arresto dell'algoritmo di punto fisso

Dato che l'algoritmo non ha quasi mai termine, occorre porre un criterio di arresto:
Il criterio viene di solito scelto tra uno di questi due

- Quando $\mid x_i - x_{i+1} \mid < \varepsilon$ con ε prefissato. Il criterio può essere poco valido quando $g'(x)$ è positivo, perché può capitare che $\mid x_i - x_{i+1} \mid$ sia più piccolo di $x_{i+i} - \alpha$, come succede nel secondo esempio grafico sopra.
- Quando $\dfrac{\mid x_i - x_{i+1} \mid}{\min\{\mid x_i \mid, \mid x_{i+1} \mid\}} < \varepsilon$ con ε prefissato. Come criterio può essere più affidabile, come ora vedremo.

Vediamo ora (vedi anche gli esempi precedenti) che il segno della derivata di $g(x)$ consente di stabilire in che modo la successione x_i converge.

Quando $g'(x)$ è positivo e quindi $0 < g'(x) < 1$, la successione è *monotona*
Quando $g'(x)$ è negativo e quindi $-1 < g'(x) < 0$, la successione è *alternante*.

In quest'ultimo caso è possibile valutare l'errore assoluto, dato che α è compreso tra x_i e x_{i+1}.

Per capire come vadano le cose, applichiamo il noto teorema di Lagrange all'intervallo $[x_{i-1}, \alpha]$ (o all'intervallo $[\alpha, x_{i-1}]$):

$$\frac{g(x_{i-1}) - g(\alpha)}{x_{i-1} - \alpha} = g'(\xi) \qquad \text{con } \mid \xi - \alpha \mid < \mid x_{i-1} - \alpha \mid$$

Quindi, dato che $g(\alpha) = \alpha$ e $g(x_{i-1}) = x_i$

$$x_i - \alpha = g'(\xi)(x_{i-1} - \alpha) \qquad \text{da cui} \qquad x_i - x_{i-1} = (x_i - \alpha) - (x_{i-1} - \alpha) = (x_{i-1} - \alpha)(g'(\xi) - 1)$$

In conclusione

$$\mid x_{i-1} - \alpha \mid = \frac{\mid x_i - x_{i-1} \mid}{\mid g'(\xi) - 1 \mid}$$

Col primo criterio di arresto si ha:

$$\mid x_{i-1} - \alpha \mid < \frac{\varepsilon}{\mid g'(\xi) - 1 \mid} \qquad \text{Se } -1 < g'(\xi) < 0, \text{ allora} \qquad \mid x_{i-1} - \alpha \mid < \varepsilon$$

Quindi il criterio di arresto maggiora l'errore assoluto se $g'(x) < 0$. Non si può invece dare una maggiorazione di $\mid x_{i-1} - \alpha \mid$ se $g'(x) > 0$.

Col secondo criterio di arresto si ha:

$$\frac{\mid x_{i-1} - \alpha \mid}{\mid \alpha \mid} < \frac{\varepsilon \cdot \min\{\mid x_i \mid, \mid x_{i+1} \mid\}}{\mid \alpha \mid \mid g'(\xi) - 1 \mid}$$

Se $g'(\xi) < 0$, allora $\min\{\mid x_i \mid, \mid x_{i+1} \mid\} \simeq \mid \alpha \mid$, quindi l'ultima espressione è circa ε.

Se $g'(x) > 0$ e soprattutto se $g'(x) \simeq 1$ l'errore può essere ancora grande anche in presenza di ε piccolo. Per valutare la distanza assoluta o relativa di x_i da α occorre conoscere almeno approssimativamente il valore di $g'(x)$ in un intorno di α.

2.1.4 Ordine di convergenza dell'algoritmo di punto fisso

Definizione: Se in un algoritmo di punto fisso si ha $\lim_{i \to \infty}(x_i) = \alpha$ (e si ha $x_i \neq \alpha$ per ogni i), allora il numero $\gamma = \lim_{i \to \infty} \dfrac{\mid x_{i+1} - \alpha \mid}{\mid x_i - \alpha \mid}$ è detto *fattore di convergenza*.

Se l'algoritmo converge ad α, si ha sempre $\gamma \leq 1$.
Se $0 < \gamma < 1$ si dice che la convergenza è *lineare* (caso normale)
Se $\gamma = 1$ si dice che la convergenza è *sublineare* (caso lento)
Se $\gamma = 0$ si dice che la convergenza è *superlineare* (caso veloce)

Definizione: Nelle ipotesi precedenti, se esiste p , $p \geq 1$ tale che $\lim_{i\to\infty} \frac{|\, x_{i+1} - \alpha \,|}{|\, x_i - \alpha \,|^p} = \ell$ con $\ell \neq 0$, si dice che p è l'*ordine di convergenza.*

Se l'ordine di convergenza è 1, ciò significa che le cifre decimali (o binarie) esatte del risultato aumentano all'incirca linearmente ad ogni passo dell'algoritmo. Se l'ordine di convergenza è invece 2, ad ogni passo il numero di cifre decimali (o binarie) esatte viene circa raddoppiato.

L'ordine di convergenza è strettamente legato alla derivata prima e alle successive:

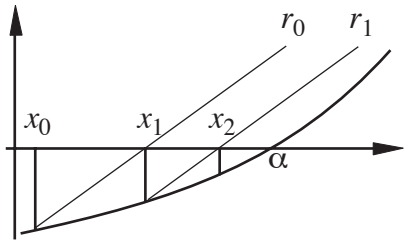
Proposizione 6 *Nelle ipotesi precedenti, supponiamo che $g(x)$ sia di classe C^p in $[\alpha - \varrho\,,\ \alpha + \varrho]$ e $x_0 \in [\alpha - \varrho\,,\ \alpha + \varrho]$. Se l'ordine di convergenza della successione di punto fisso è p, allora:*

$$g'(\alpha) = g''(\alpha) = \cdots = g^{(p-1)}(\alpha) = 0 \qquad g^{(p)}(\alpha) \neq 0$$

In parole povere, se $p \geq 2$, la tangente a $g(x)$ in α è orizzontale.

2.1.5 Riduzione di un'equazione ad algoritmo di punto fisso

In generale l'equazione $f(x) = 0$ può essere trasformata in vari modi in un problema di punto fisso:



Sia α uno zero di $f(x)$. Scegliamo un punto x_0 prossimo ad α e consideriamo la retta r_0 con coefficiente angolare h passante per $(x_0, f(x_0))$ con h scelto in qualche modo.
La retta è $r_0 :\ y - f(x_0) = h(x - x_0)$
L'intersezione tra la retta r_0 e l'asse x ha ascissa $x_1 = x_0 - \dfrac{f(x_0)}{h}$.
Proseguiamo con la retta $r_1 :\ y - f(x_1) = h(x - x_1)$.

L'intersezione tra la retta r_0 e l'asse x ha ascissa $x_2 = x_1 - \dfrac{f(x_1)}{h}$

In pratica stiamo cercando il punto fisso della funzione $g(x) = x - \dfrac{f(x)}{h}$.

Naturalmente non è detto che l'algoritmo converga ad α, ma in molti casi, attraverso un'opportuna scelta di h è possibile riuscirci.
In generale, anche h non andrà scelto costante, ma verrà fatto variare in funzione dell'x via via trovato. Quindi dobbiamo cercare di studiare la convergenza dell'algoritmo di punto fisso della funzione

$$g(x) = x - \frac{f(x)}{h(x)}$$

con $h(x)$ scelta opportunamente in modo che $|\, g'(x) \,| < 1$.

A seconda della scelta di $h(x)$ si ottengono vari algoritmi. I più noti sono quelli delle corde, delle tangenti, delle secanti e quello della falsa posizione.

Per quanto riguarda i criteri d'arresto, essenzialmente ce ne sono tre, di cui due sono quelli dell'algoritmo di punto fisso cioè

- $|\, x_i - x_{i+1} \,| < \varepsilon$ con ε prefissato.
- $\dfrac{|\, x_i - x_{i+1} \,|}{\min\{|\, x_i \,|, |\, x_{i+1} \,|\}} < \varepsilon$ con ε prefissato.

Si può inoltre usare il seguente criterio:

- $|\, f(x_i) \,| < \varepsilon$ con ε prefissato.

Quest'ultimo criterio è il più semplice e osserviamo che, dato che si ha $f(x_i) = (x_i - g(x_i)) \cdot h(x_i)$ e $g(x_i) = x_{i+1}$, esso equivale a chiedere che

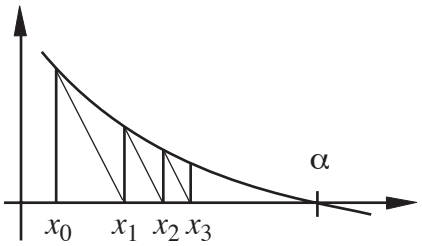
$$|\, f(x_i) \,| = |\, x_i - x_{i+1} \,| \cdot |\, h(x_i) \,| < \varepsilon \qquad \text{ovvero} \qquad |\, x_i - x_{i+1} \,| < \varepsilon \,/\, |\, h(x_i) \,|$$

Quindi sarà in pratica equivalente al primo, se si dispone di una maggiorazione di $|\, 1/h(x) \,|$ in un intorno di α.

2.1.6 Metodo delle corde

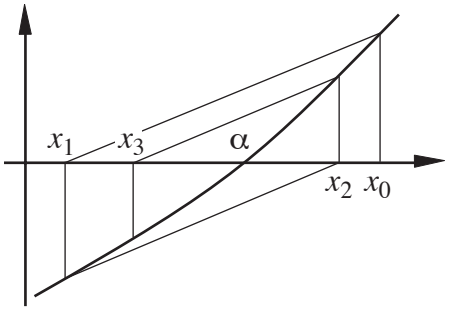
È il metodo più semplice ed è quello con la scelta $h(x) = m$ (m inclinazione costante).

Si cerca quindi il punto fisso della funzione $g(x) = x - \dfrac{f(x)}{m}$,
ovvero l'algoritmo è $x_{i+1} = x_i - \dfrac{f(x_i)}{m}$.
Condizione sufficiente affinché l'algoritmo converga è che $| g'(x) | = \left|1 - \dfrac{f'(x)}{m}\right| < 1$.



Equivalentemente la condizione è che in un intorno di α comprendente x_0 valgano le tre seguenti:

- $f'(x) \neq 0$
- $f'(x) \cdot m > 0$ (devono avere lo stesso segno)
- $| m | > \dfrac{1}{2} \max\{f'(x)\}$



Il metodo delle corde, se converge, converge di ordine 1.
I due esempi grafici mostrano come la convergenza possa essere monotona o alternante.

2.1.7 Metodo delle tangenti

Detto anche metodo di Newton-Raphson, è sicuramente il più noto e presuppone il calcolo di $f'(x)$. Infatti come h si usa il valore della derivata nel punto, cioè la retta tangente al grafico. In pratica $h(x) = f'(x)$

L'algoritmo consiste nel determinare un punto fisso della funzione $g(x) = x - \dfrac{f(x)}{f'(x)}$, quindi la successione x_i è così definita: $x_{i+1} = x_i - \dfrac{f(x_i)}{f'(x_i)}$.

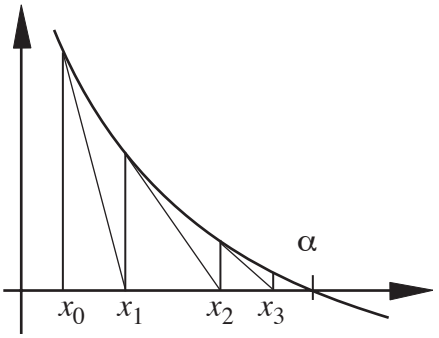
Condizione sufficiente affinché l'algoritmo converga è che $| g'(x) | = \left|\dfrac{f(x)f''(x)}{(f'(x))^2}\right| < 1$.

Non è facile verificare direttamente la diseguaglianza, quindi si ricorre a criteri sufficienti.
Il più noto è:

Proposizione 7 *Supponiamo che $f(x)$ sia di classe C^2 in $I = [\alpha\,,\ \alpha + \varrho]$ (o in $I = [\alpha - \varrho\,,\ \alpha]$)e $x_0 \in I$.*
Se nell'intervallo si ha $f(x)f''(x) > 0$ e $f'(x) \neq 0$ allora l'algoritmo converge.

Diversamente dal metodo delle corde, perché sia garantita la convergenza, occorre anche scegliere opportunamente il punto iniziale x_0 a destra o a sinistra di α, a seconda delle circostanze.
Graficamente la situazione è quella a lato.
È chiaro che nella situazione disegnata l'algoritmo converge partendo da x_0 a sinistra perché la $f(x)$ è positiva e così pure $f''(x)$, mentre l'algoritmo non converge necessariamente partendo da x_0 a destra perché $f(x)$ è negativa e $f''(x) > 0$

Il metodo delle tangenti, se converge, converge di ordine 2 o superiore, quindi, quando è applicabile, è uno dei più veloci.



Esempio 2.2: Si può calcolare $\sqrt{2}$ cercando lo zero positivo della funzione $x^2 - 2$.
Basta eseguire l'algoritmo di punto fisso sulla funzione $g(x) = x - \dfrac{x^2-2}{2x} = \dfrac{x^2+2}{2x}$.
Se si parte da un qualunque $x_0 > 2$ converge perché soddisfa le condizioni sufficienti.
Se si parte da $x_0 = 1$ converge ugualmente, perché dopo il primo passo si trova $x_1 = 3/2$ e si rientra nelle condizioni sufficienti della proposizione, non così se si inizia invece con $x_0 < 0$.

2.1.8 Metodo delle secanti

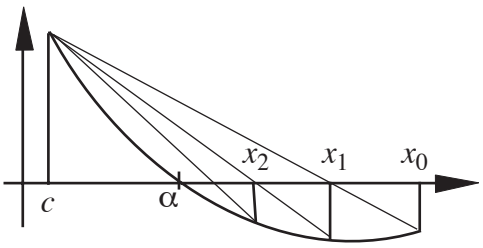
Sia α lo zero da cercare; fissiamo $x = c$ prossimo ad α e scegliamo come valore di partenza dell'algoritmo un x_0 tale che α sia compreso tra c e x_0.
Consideriamo la retta congiungente i due punti $(c, f(c))$ $(x_0, f(x_0))$. L'intersezione tra la retta e l'asse x è il nuovo punto x_1.
L'algoritmo è $x_{i+1} = x_i - \dfrac{f(x_i)(x_i - c)}{f(x_i) - f(c)}$. Quindi come funzione h si ha $h(x) = \dfrac{f(x) - f(c)}{x - c}$.
La funzione di cui trovare il punto fisso è

$$g(x) = \frac{c \cdot f(x) - x \cdot f(c)}{f(x) - f(c)} \qquad \text{e si ha} \qquad g'(x) = f(c)\,\frac{f'(x)(x-c) - f(x) + f(c)}{(f(x) - f(c))^2}$$

Una condizione sufficiente per la convergenza è $\left|\dfrac{f(c)}{c-\alpha}\right| > \dfrac{1}{2} \mid f'(\alpha) \mid$. Come per le tangenti si ha:

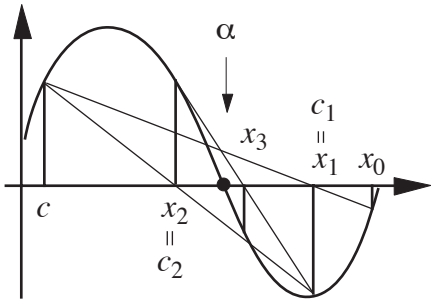
Proposizione 8 *Supponiamo che la funzione $f(x)$ definita nell'intervallo $I = [a, b]$ sia di classe C^2 e si abbia $f'(x), f''(x) \neq 0$.*
Se si scelgono nell'intervallo c, x_0 tali che $f(c) \cdot f''(c) \geq 0$ e inoltre $f(x_0) \cdot f''(x_0) \leq 0$, allora l'algoritmo delle secanti converge (monotonamente).

L'algoritmo delle secanti è talvolta preferito a quello delle tangenti, anche se la convergenza è di ordine 1, perché la funzione di cui calcolare il punto fisso può essere più semplice, non prevedendo il calcolo della derivata di $f(x)$. Per certe funzioni non definite mediante formule esplicite il calcolo della derivata può essere estremamente difficoltoso. Inoltre l'algoritmo delle secanti è la premessa al metodo seguente.



2.1.9 Metodo della falsa posizione (regula falsi)

Come nel metodo delle secanti fissa un punto c prossimo allo zero da cercare α e si scrive la retta congiungente i due punti $(c, f(c))$ $-$ $(x_0, f(x_0))$. Però ci si riserva di cambiare il punto c, quando sia necessario, se le condizioni della proposizione non sono più verificate. Nella fattispecie, se x_{i+1} è tale che $f(x_{i+1}) \cdot f(c) > 0$, allora si pone $c = x_i$ e si prosegue l'algoritmo con il nuovo c.
L'algoritmo delle secanti, modificato con la regula falsi, converge di ordine 1 e ha il pregio di convergere nella sola ipotesi che $f(x)$ sia continua.
Come si vede nell'esempio, si comincia con c e x_0 tra cui è compreso α e si trova x_1.
Poi si continua con x_1 e c e si trova x_2. A questo punto $f(x_2)$ e $f(c)$ sono concordi, perciò α non è più compreso tra c e x_i .
Si sostituisce c con $c_1 = x_1$ e si prosegue con x_2 e c_1.
Si trova x_3 e, dato che $f(x_3)$ e $f(c_1)$ sono concordi, si deve di nuovo porre $c_2 = x_2$, dopodiché l'algoritmo dovrebbe procedere senza più cambiamenti.

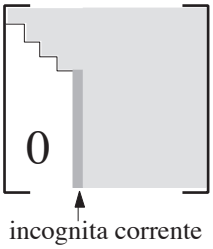


3.1 Algebra lineare numerica

3.1.1 Le varianti dell'algoritmo di Gauss

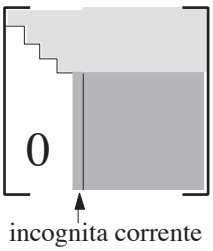
Dato un sistema lineare *quadrato* $Ax = b$ con A matrice invertibile (c'è sempre il problema di scoprire se lo sia), vediamo quali sono i metodi di risoluzione. L'algoritmo di Gauss è il metodo base, ma ha parecchie varianti. Elenchiamo le principali varianti:

1. **Pivotizzazione parziale**: L'algoritmo di Gauss prevede la ricerca di un pivot per ogni incognita. Dopo alcuni passi dell'algoritmo di Gauss la matrice è parzialmente a scala.
   Il pivot va cercato nella zona grigio scuro tra i coefficienti della incognita corrente.
   La *pivotizzazione parziale* prevede che tra i possibili pivot si scelga sempre quello di valore assoluto più alto e poi si faccia uno scambio di righe per usarlo come pivot. La scelta di un pivot di valore assoluto alto riduce l'impatto degli inevitabili errori di arrotondamento.
   Non è facile dare una spiegazione di questo fatto, ma si può averne un'intuizione dal fatto che, se un pivot nullo è improponibile, un pivot piccolo è comunque sconsigliato.



2. **Pivotizzazione totale**: Questa strategia prevede la ricerca di un pivot non solo tra i coefficienti dell'incognita su cui si sta lavorando, ma anche tra i coefficienti delle incognite successive (la zona grigio scuro della figura)

   Se viene trovato un buon pivot in un'altra incognita, si scambiano tra loro le incognite e poi si procede secondo l'algoritmo classico. Naturalmente si dovrà tenere conto di questi scambi al momento di scrivere il risultato finale, cioè la n-upla delle soluzioni.
   La ricerca di un pivot in un insieme più vasto richiede più tempo contro un vantaggio non sempre reale, quindi il metodo della pivotizzazione totale è scarsamente usato, mentre la pivotizzazione parziale è in pratica lo standard nell'algoritmo gaussiano.



3. **Pivotizzazione scalata**: Osserviamo che, se una riga di un sistema viene moltiplicata per una costante non nulla, il sistema ottenuto è equivalente, ma può cambiare la scelta del pivot nella strategia della pivotizzazione parziale.

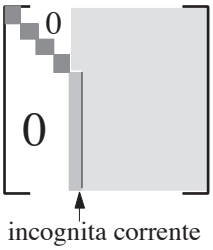
   Questo è il motivo per cui a volte si usa, in luogo della pivotizzazione totale, a parità di tempo, la cosiddetta *pivotizzazione parziale scalata.* In questo caso un elemento di una matrice viene considerato grande, non se è grande in assoluto, ma se lo è rispetto al resto della riga.

   Per la precisione, in ognuna delle righe interessate alla ricerca del pivot, si calcola la grandezza della riga i-esima che è definita come $d_i = \max_j \{ \mid a_{ij} \mid \}$.

   Quindi la grandezza del possibile pivot p_i della i-esima riga è calcolata come $\frac{\mid p_i \mid}{d_i}$ e viene scelto come pivot quello per cui il rapporto è maggiore.

4. **Algoritmo di Gauss-Jordan**: Usando l'algoritmo classico di Gauss, si produce una matrice ridotta, dopodiché occorre l'algoritmo retrogrado ovvero la sostituzione all'indietro per risolvere il sistema.

   La variante di Jordan dell'algoritmo gaussiano invece riduce immediatamente in modo totale la matrice.
   Cioè il pivot viene usato non solo per annullare i coefficienti della sua colonna situati nelle righe inferiori, ma anche quelli situati nelle righe sopra. Nella figura in scuro i pivot già usati.
   L'algoritmo di Gauss-Jordan venne usato ai primordi del calcolo, perché riducendo immediatamente tutta la matrice, permetteva di liberare dalla memoria del computer i dati delle colonne già ridotte.



Oggi è meno usato, dato che comporta un tempo leggermente superiore all'algoritmo classico di Gauss, mentre la quantità di memoria disponibile non è più un problema.

5. **La fattorizzazione LU**: Per risolvere il sistema $Ax = b$ l'algoritmo classico di Gauss prevede che si riduca la matrice $[A \mid b]$.
   La fattorizzazione LU, che non descriviamo in dettaglio, prevede invece che si riduca solo la matrice A ottenendo quindi una matrice U triangolare superiore (U sta per "upper triangular"). Le operazioni elementari eseguite vengono memorizzate in una matrice (quasi) triangolare inferiore L. Il costo di questa operazione è praticamente nullo perché non richiede operazioni aritmetiche, ma solo spazio in memoria. Non stiamo qui a descrivere in dettaglio la costruzione di L, diciamo solo che tra le matrici A, L, U c'è la relazione $A = L \cdot U$, per cui si parla di fattorizzazione LU .
   Vediamo ora come si usa la decomposizione $A = L \cdot U$ per risolvere il sistema $A \cdot x = b$.
   Il sistema diventa $L \cdot U \cdot x = b$.
   La matrice L è (quasi) triangolare inferiore, nel senso che lo è a meno di un riordinamento delle righe e inoltre gli elementi della diagonale (una volta riordinata) sono tutti 1.
   Pertanto è facile risolvere il sistema lineare $L \cdot t = b$ con una variante dell'algoritmo retrogrado di Gauss che consiste semplicemente nel partire dalla prima equazione e prima incognita anziché dall'ultima.
   Sia quindi b_1 la soluzione del sistema $L \cdot t = b$. Risulta pertanto $L \cdot b_1 = b$.
   Il sistema originale $L \cdot U \cdot x = b$ si scrive $L \cdot U \cdot x = L \cdot b_1$ ed è equivalente al sistema ridotto $U \cdot x = b_1$ che si risolve con la sostituzione all'indietro.
   La x trovata è la soluzione del sistema originale.
   In pratica, una volta ridotta A, si trova la nuova matrice dei termini noti semplicemente risolvendo $L \cdot t = b$.
   Questo metodo, nonostante l'apparenza più macchinosa, è in realtà una variante dell'algoritmo gaussiano che richiede un numero di operazioni aritmetiche (somme, prodotti e divisioni) uguale a quello della riduzione totale di $[A \mid b]$ attraverso l'algoritmo gaussiano classico.
   Il grosso vantaggio di questo metodo sta però nel fatto che, una volta individuata la fattorizzazione LU della matrice A, qualunque altro sistema $Ax = c$, avente la stessa matrice dei coefficienti, ma diverso termine noto, può essere risolto in tempo brevissimo usando la fattorizzazione LU già trovata, dato che il calcolo di L e U è la parte più onerosa dell'intero processo e si può evitare di ripeterlo.

6. **La matrice inversa e il metodo di Cramer**: Risolvere il sistema $Ax = b$ scrivendo $x = A^{-1}b$ è lecito, ma non conveniente. Infatti, mentre la fattorizzazione LU per ridurre A richiede *circa* $n^3/3$ prodotti, per calcolare l'inversa mediante l'algoritmo di Gauss occorrono invece circa n^3 prodotti.

   Nella soluzione dei sistemi lineari, non conviene quindi determinare l'inversa della matrice dei coefficienti, ma usare metodi tipo la fattorizzazione LU.

   La riduzione retrograda a partire dalla matrice ridotta U richiede un numero di prodotti dell'ordine di $n^2/2$, numero trascurabile, rispetto alle operazioni richieste per la riduzione della matrice.
   Del tutto da evitare, salvo casi particolarissimi, è la nota regola di Cramer.
   La regola dice che $x_i = \det(A_i)/\det(A)$ dove con A_i si indica la matrice ottenuta sostituendo in A la colonna C_i con la colonna b.
   Quindi la regola di Cramer richiede il calcolo di $n+1$ determinanti, ciascuno dei quali richiede circa $n^3/3$ prodotti.

7. **I metodi iterativi**: Assomigliano un po' agli algoritmi di punto fisso. Si parte da una stima della soluzione e, attraverso un algoritmo se ne trova (se l'algoritmo converge) una stima più prossima. Non si trova praticamente mai la soluzione esatta (d'altra parte anche con l'algoritmo di Gauss non la si ottiene mai, causa gli arrotondamenti), ma hanno certi tipi di vantaggi come vedremo più avanti.

3.1.2 Il condizionamento

Sia A una matrice invertibile. Consideriamo il sistema lineare $Au = b$ (con $b \neq 0$) e sia x la sua soluzione.
Consideriamo poi il sistema $Au = b + \delta b$ in cui il termine noto b ha subito una "piccola" perturbazione δb e sia $x + \delta x$ la sua soluzione. Ci proponiamo di studiare quanto sia piccola la perturbazione δx subita dalla soluzione del sistema.
Occorre confrontare la grandezza di δx con quella di δb.
Usualmente la grandezza di un vettore si misura mediante la *norma euclidea*:

$$\text{Se } v = (x_1, ..., x_n): \qquad \| v \| = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$$

La norma ha tre proprietà

1. $\| u + v \| \leq \| u \| + \| v \|$
2. $\| \lambda v \| = | \lambda | \, \| v \|$
3. $\| v \| \geq 0$ e $\| v \| = 0$ se e solo se $v = 0$

Avvertiamo che esistono altri modi di misurare la norma, o meglio altre norme, a volte più convenienti, comunque ci limitiamo alla norma euclidea.

Teniamo ora presente il fatto che è importante non tanto conoscere la norma della perturbazione $\| \delta x \|$ subita da x, quanto il rapporto $\dfrac{\| \delta x \|}{\| x \|}$, cioè la misura relativa della perturbazione e che questo rapporto va confrontato con quello analogo per b: $\dfrac{\| \delta b \|}{\| b \|}$
La "piccolezza" di δb sarà quindi misurata da $\| \delta b \| \, / \, \| b \|$ e analogamente per δx.

Consideriamo le due eguaglianze

$$Ax = b \qquad A(x + \delta x) = b + \delta b \qquad \text{da cui} \qquad A \, \delta x = \delta b$$

Pertanto $\| Ax \| = \| b \|$ $\| A \delta x \| = \| \delta b \|$.

Per confrontare $\| \delta b \| \, / \, \| b \|$ con $\| \delta x \| \, / \, \| x \|$ occorre quindi conoscere $\| Ax \|$ e $\| A \delta x \|$ o meglio individuare una relazione tra $\| x \|$ e $\| Ax \|$ e una tra $\| \delta x \|$ e $\| A \delta x \|$. Queste relazioni dipenderanno ovviamente da proprietà della matrice A.

Poniamo quindi la seguente definizione.

Definizione: La *norma matriciale* di una matrice quadrata invertibile $A \in M_{nn}(\mathbb{R})$ è

$$\| A \| = \max_{x \neq 0} \frac{\| Ax \|}{\| x \|}$$

al variare di x in $\mathbb{R}^n$ (x è sempre un vettore colonna).

Dalla definizione si ricavano immediatamente le due relazioni

$$\| Ax \| \leq \| A \| \cdot \| x \| \qquad\qquad \| A \delta x \| \leq \| A \| \cdot \| \delta x \|$$

Queste ci consentono di procedere nel nostro conto.

Prima di continuare osserviamo che rimane il problema di calcolare $\| A \|$, dato che la definizione precedente non può essere usata per calcolare direttamente la norma di una matrice. Questo è un problema più complesso che rinviamo al paragrafo successivo.

Si ha:

$$\| b \| = \| Ax \| \leq \| A \| \, \| x \|$$

Per il confronto tra δb e δx conviene scrivere diversamente la relazione $A\delta x = \delta b$ usando la matrice inversa A^{-1}

$$\delta x = A^{-1}\delta b \qquad \text{da cui} \qquad \| \delta x \| = \| A^{-1}\delta b \| \leq \| A^{-1} \| \, \| \delta b \|$$

Le due relazioni ottenute

$$\| b \| \leq \| A \| \, \| x \| \qquad \| \delta x \| \leq \| A^{-1} \| \, \| \delta b \|$$

si possono scrivere:

$$\frac{1}{\| x \|} \leq \frac{\| A \|}{\| b \|} \qquad \| \delta x \| \leq \| A^{-1} \| \, \| \delta b \|$$

Moltiplicando le due diseguaglianze membro a membro, si ha la relazione cercata:

$$\frac{\| \delta x \|}{\| x \|} \leq \| A \| \, \| A^{-1} \| \, \frac{\| \delta b \|}{\| b \|}$$

Quindi, se $\| \delta b \| / \| b \|$ è piccolo e anche il numero $\| A \| \, \| A^{-1} \|$ lo è, allora $\| \delta x \| / \| x \|$ rimane piccolo. Se invece il numero $\| A \| \, \| A^{-1} \|$ è grande, a fronte di una piccola perturbazione di b si può verificare una grossa perturbazione di x.

Si pone quindi la definizione

Definizione: Se A è una matrice quadrata e invertibile, il numero

$$\text{cond}(A) = \| A \| \, \| A^{-1} \|$$

è detto *numero di condizionamento* di A.

La relazione precedente si scrive quindi usualmente come

$$\frac{\| \delta x \|}{\| x \|} \leq \text{cond}(A) \, | \, \frac{\| \delta b \|}{\| b \|}$$

Rimane il problema di calcolare $\| A \|$ e $\| A^{-1} \|$ e quindi cond(A).

Prima però esaminiamo un esempio.

Esempio 3.1: Siano $A = \begin{pmatrix} -1 & 2 & 2 \\ 2 & 1 & 3 \\ 2 & 3 & 6 \end{pmatrix}$ e $b = \begin{pmatrix} 3 \\ 4 \\ 8 \end{pmatrix}$ La soluzione del sistema lineare $Ax = b$ è $x = \begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix}$.

Apparentemente la matrice A non presenta inconvenienti: è simmetrica, ha elementi non troppo distanti tra loro e ha determinante -1. Se però consideriamo il sistema $Ax = b + \delta b$ con $b + \delta b = \begin{pmatrix} 3.1 \\ 4.2 \\ 7.9 \end{pmatrix}$ si scopre che la soluzione è $x + \delta x = \begin{pmatrix} 2.9 \\ 5.3 \\ -2.3 \end{pmatrix}$ assai differente da x.

Esaminiamo le norme. Si ha: $\| x \| \simeq 2.24$, $\| b \| \simeq 9.4$, $\| \delta x \| \simeq 4.45$, $\| \delta b \| \simeq 0.24$

Quindi $\dfrac{\| \delta b \|}{\| b \|} \simeq 0.03$ mentre $\dfrac{\| \delta x \|}{\| x \|} \simeq 1.99$

La norma di b è variata circa del 3%, mentre quella di x ha subito una variazione del 199% !

Dato che $\dfrac{\| \delta x \|}{\| x \|} \Big/ \dfrac{\| \delta b \|}{\| b \|} \leq \text{cond}\, A$, questo significa che cond(A) è superiore a $199/3 \simeq 66$.

3.1.3 Calcolo di norme e condizionamenti

Caso simmetrico

Se A è simmetrica, $(A = A^T)$, allora, come è noto (teorema *spettrale*), A ha solo autovalori reali $\lambda_1, \lambda_2, ..., \lambda_n$. Ordiniamo gli n autovalori secondo il loro modulo: $| \lambda_1 | \leq | \lambda_2 | \leq \cdots \leq | \lambda_n |$, per

cui con λ_1 si intenderà un'autovalore (può non essere unico) di A minimo in modulo e con λ_n un'autovalore massimo in modulo.

Si dimostra che: $\| A \| = | \lambda_n |$

Gli autovalori di A^{-1} sono notoriamente i reciproci di quelli di A, per cui la successione degli autovalori sarà: $\left|\frac{1}{\lambda_1}\right| \geq \left|\frac{1}{\lambda_2}\right| \geq \cdots \geq \left|\frac{1}{\lambda_n}\right|$. Quindi $\| A^{-1} \| = \lambda_1^{-1}$. In conclusione:

$$\text{cond}(A) = \left|\frac{\lambda_n}{\lambda_1}\right|$$

Una matrice è *ben condizionata* se gli autovalori non sono troppo distanti tra loro in modulo.

Esempio 3.2: Nell'esempio precedente gli autovalori di A sono circa $8.18 \quad , \quad -2.24 \quad , \quad 0.05$, per cui $\text{cond}(A) = \frac{8.18}{0.05} \simeq 150$, piuttosto elevato, come si è visto.

Caso generale

Se A non è simmetrica, consideriamo la matrice $A^T \cdot A$.
Si verifica facilmente che $A^T A$ è una matrice simmetrica e definita positiva, quindi i suoi autovalori $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ sono tutti positivi. Per ogni i poniamo $s_i = \sqrt{\lambda_i}$.
Le radici quadrate $s_1 \leq s_2 \leq \cdots \leq s_n$ si dicono *valori singolari* di A.
Si dimostra che $\| A \| = s_n = \sqrt{\lambda_n}$. Analogamente $\| A^{-1} \| = 1/s_1 = \sqrt{1/\lambda_1}$, per cui

$$\text{cond}(A) = \frac{\text{max valore singolare di } A}{\text{min valore singolare di } A} = \frac{s_n}{s_1} = \sqrt{\frac{\lambda_n}{\lambda_1}}$$

Notiamo che per matrici simmetriche il calcolo di $\text{cond}(A)$ fornisce lo stesso risultato nei due casi.

Osservazioni:

- Dal calcolo di $\text{cond}(A)$ si deduce che $\text{cond}(A) \geq 1$ per ogni matrice, mentre non esiste un limite superiore.
- Se A è una matrice *ortogonale*, ovvero le colonne di A sono a due ortogonali e di norma 1, come è noto, A è una matrice isometrica, cioè per ogni x si ha $\| Ax \| = \| x \|$.
  Per come è stata definita la norma di A, si deduce quindi che, se A è ortogonale, allora $\| A \| = 1$. Dato che anche A^{-1} è ortogonale, anche $\| A^{-1} \| = 1$, quindi $\text{cond}(A) = 1$. Le matrici ortogonali sono quindi sempre ben condizionate.

3.1.4 Metodi iterativi, il metodo di Jacobi

La pratica mostra che il metodo di eliminazione di Gauss diventa inaffidabile quando il sistema sia troppo grosso anche usando tutte le cautele possibili.
Per questa ragione conviene in molti casi ricorrere ai metodi iterativi che consentono di usare sempre la matrice originale e modificano invece la soluzione fino a farla tendere a quella esatta. Questi metodi, quando funzionano, permettono di superare anche l'eventuale mal condizionamento della matrice che rende ancor più instabile l'algoritmo gaussiano
Inoltre in molti casi consentono di avere una soluzione accettabile in un tempo più breve di quello richiesto dall'eliminazione gaussiana.

Il metodo di Jacobi consiste nel decomporre A come $A = S - T$, dove S è la matrice diagonale di A e T è la matrice complementare con diagonale nulla.
Esplicitamente:

$$\text{Se } A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots \\ a_{21} & a_{22} & a_{23} & \cdots \\ a_{31} & a_{32} & a_{33} & \cdots \\ \cdots & \cdots & & \end{pmatrix}$$

Allora:

$$S = \begin{pmatrix} a_{11} & 0 & 0 & \cdots \\ 0 & a_{22} & 0 & \cdots \\ 0 & 0 & a_{33} & \cdots \\ \cdots & \cdots & & \end{pmatrix} \quad T = \begin{pmatrix} 0 & -a_{12} & -a_{13} & \cdots \\ -a_{21} & 0 & -a_{23} & \cdots \\ -a_{31} & -a_{32} & 0 & \cdots \\ \cdots & \cdots & & \end{pmatrix}$$

Si scrive:

$$Ax = Sx - Tx = b \quad \text{cioè} \quad Sx = Tx + b$$

Sia ora x_0 un qualunque vettore. Sostituiamo x_0 a secondo membro e otteniamo:

$$Sx = Tx_0 + b$$

Dato che il sistema nella matrice S è facilmente risolubile, è agevole trovare la soluzione x_1 di questo sistema. Ricominciamo dal sistema $Sx = Tx + b$, sostituendo x_1 a secondo membro:

$$Sx = Tx_1 + b$$

Risolviamo nuovamente il sistema determinando x_2 e così via. Descriviamo esplicitamente il metodo di Jacobi nel caso particolare di un sistema 3×3:

Esempio 3.3: Siano

$$A = \begin{pmatrix} 3 & 1 & 0 \\ 1 & 3 & 1 \\ 0 & 1 & 3 \end{pmatrix} \quad b = \begin{pmatrix} 5 \\ 4 \\ -7 \end{pmatrix}$$ Il sistema $Sx - Tx = b$ è: $$\begin{cases} 3x &= -y + 5 \\ 3y &= -x - z + 4 \\ 3z &= -y - 7 \end{cases}$$

Partiamo con la terna $x_0 = 0\ ;\ y_0 = 0\ ;\ z_0 = 0$.
Sostituiamo a secondo membro $(0, 0, 0)$ e otteniamo:
$x_1 = 5/3\ ;\ y_1 = 4/3\ ;\ z_1 = -7/3$
Sostituiamo a secondo membro $(5/3, 4/3, -7/3)$ e otteniamo:
$x_2 = 11/9\ ;\ y_2 = 14/9\ ;\ z_2 = -25/9$
Sostituiamo a secondo membro $(11/9, 14/9, -25/9)$ e otteniamo:
$x_3 = 31/27\ ;\ y_3 = 50/27,\ ;\ z_3 = -77/27$
L'ultima terna è $(1.14...\ ,\ 1.85...\ ,\ -2.85...)$ che è discretamente vicina alla soluzione esatta: $(1, 2, -3)$.

Non sempre il metodo di Jacobi converge, in realtà si può dimostrare che la successione converge alla soluzione del sistema, quale che sia la scelta iniziale di x_0, se e solamente la matrice $S^{-1}T$ ha tutti autovalori minori di 1 in modulo. Non è praticamente mai possibile, né conveniente verificare direttamente la condizione del teorema. Esistono però dei criteri *sufficienti* di facile uso che garantiscano che essa sia verificata.

Proposizione 9 *(condizione sufficiente) L'algoritmo di Jacobi converge nei seguenti due casi interessanti:*

- *Se A è* diagonalmente dominante *per righe, se cioè in ogni riga l'elemento a_{ii} è in modulo strettamente maggiore della somma dei moduli degli altri elementi della riga.*
- *Se A è* diagonalmente dominante *per colonne, se cioè in ogni colonna l'elemento a_{ii} è in modulo strettamente maggiore della somma dei moduli degli altri elementi della colonna.*

In effetti la matrice dell'esempio sopra è diagonalmente dominante sia per righe che per colonne.

$$\begin{pmatrix} 3 & 1 & 0 \\ 1 & 3 & 1 \\ 0 & 1 & 3 \end{pmatrix} \quad \begin{array}{l} |\,3\,| > |\,1\,| + |\,0\,| \\ |\,3\,| > |\,1\,| + |\,1\,| \\ |\,3\,| > |\,0\,| + |\,1\,| \end{array}$$

Se le diseguaglianze nelle due definizioni precedenti non sono strette, se cioè a_{ii} è maggiore o uguale alla somma dei moduli degli altri elementi, allora la matrice si dice *debolmente diagonalmente dominante*. In questo caso la convergenza non è garantita, anche se si verifica in moltissimi casi.

3.1.5 Metodi iterativi, il metodo di Gauss-Seidel

Il metodo consiste nel decomporre A come $A = S - T$, dove S è la parte triangolare inferiore di A e T è la matrice complementare.
Esplicitamente:

$$\text{Se } A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots \\ a_{21} & a_{22} & a_{23} & \cdots \\ a_{31} & a_{32} & a_{33} & \cdots \\ \cdots & \cdots & & \end{pmatrix}$$

Allora:

$$S = \begin{pmatrix} a_{11} & 0 & 0 & \cdots \\ a_{21} & a_{22} & 0 & \cdots \\ a_{31} & a_{32} & a_{33} & \cdots \\ \cdots & \cdots & & \end{pmatrix} \quad T = \begin{pmatrix} 0 & -a_{12} & -a_{13} & \cdots \\ 0 & 0 & -a_{23} & \cdots \\ 0 & 0 & 0 & \cdots \\ \cdots & \cdots & & \end{pmatrix}$$

Descriviamo esplicitamente anche il metodo di Gauss-Seidel nel caso particolare di un sistema 3×3.
Si scrive : $Sx = Tx + b$, cioè:

$$\begin{cases} a_{11}x & = & -a_{12}y - a_{13}z + b_1 \\ a_{21}x + a_{22}y & = & -a_{23}z + b_2 \\ a_{31}x + a_{32}y + a_{33}z & = & +b_3 \end{cases}$$

Nella pratica non si scrivono le matrici S e T, ma si scrive il sistema come nel metodo di Jacobi:

$$\begin{cases} a_{11}x & = & -a_{12}y - a_{13}z + b_1 \\ a_{22}y & = & -a_{21}x - a_{23}z + b_2 \\ a_{33}z & = & -a_{31}x - a_{32}y + b_3 \end{cases}$$

e la differenza sta nel fatto che a secondo membro non viene sostituita la terna (x_i, y_i, z_i), ma vengono utilizzati i valori di x, y, z via via trovati.
Quindi, dal punto di vista dell'onerosità del calcolo, il metodo di Gauss-Seidel è del tutto equivalente a quello di Jacobi, ma in generale, la convergenza è molto più veloce.

Anche qui illustriamo il metodo di Gauss-Seidel usando lo stesso sistema dell'esempio precedente:

Esempio 3.4: Sostituiremo, per semplicità, i risultati intermedi dell'esempio con i loro sviluppi decimali arrotondati alla seconda cifra decimale.

$$\begin{cases} 3x & = & -y + 5 \\ 3y & = & -x - z + 4 \\ 3z & = & -y - 7 \end{cases}$$

Partiamo con la terna $x_0 = 0\ ;\ y_0 = 0\ ;\ z_0 = 0$.
Sostituiamo y_0, z_0 a secondo membro della E_1 e otteniamo:
$x_1 = 5/3 = 1.67$
Sostituiamo x_1, z_0 a secondo membro della E_2 e otteniamo:
$y_1 = 7/9 = 0.78$
Sostituiamo x_1, y_1 a secondo membro della E_3 e otteniamo:
$z_1 = -70/27 = -2.59$
Si noti come per ricavare la terna (x_1, y_1, z_1) si siano usato i risultati intermedi.

Sostituiamo y_1, z_1 a secondo membro della E_1 e otteniamo $x_2 = 1.41$
Sostituiamo x_2, z_1 a secondo membro della E_2 e otteniamo $y_2 = 1.73$
Sostituiamo x_2, y_2 a secondo membro della E_3 e otteniamo $z_2 = -2.91$

Al terzo passo si otterrà la terna $(1.09\ ,\ 1.94\ ,\ -2.98)$ e come si vede la convergenza è più veloce che con il metodo di Jacobi.

Si dimostra che l'algoritmo di Gauss-Seidel converge in ciascuna delle due ipotesi sufficienti di diagonale dominanza, enunciate nel paragrafo precedente, in cui converge quello di Jacobi.
Aggiungiamo che l'algoritmo di Gauss-Seidel converge anche nel caso in cui la matrice A sia simmetrica e definita positiva. Comunque, se la matrice non è diagonalmente dominante, anche nel caso in cui l'algoritmo converga, la convergenza può essere assai lenta.

Per terminare aggiungiamo che i due metodi non sempre convergono, ma convergono in diversi casi che capitano nella pratica.

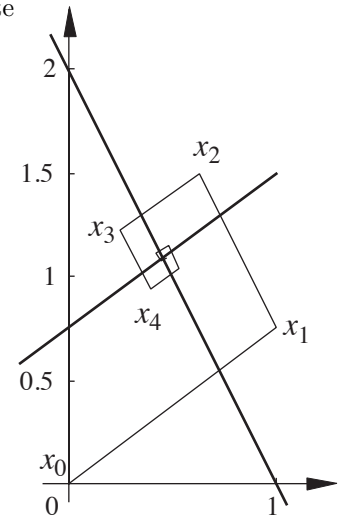
Cenno sul metodo di rilassamento: È possibile accelerare la convergenza di un metodo iterativo, "correggendo" ad ogni passo la soluzione ottenuta in modo da renderla più prossima a quella esatta.
L'idea base è la seguente: se x_{k-1} e x_k sono le soluzioni approssimate ottenute al $(k-1)^{\text{mo}}$ e k^{mo} passo di un algoritmo, si può proseguire l'algoritmo sostituendo x_k con $x_k^* = (1-\omega)x_{k-1} + \omega x_k$ dove ω è un numero compreso tra 1 e 2 (di solito intorno a 1.1), detto *coefficiente di rilassamento*.
Il reperimento del coefficiente di rilassamento corretto è la parte più difficile, ma se lo si riesce a trovare (occorrono esperienza e sperimentazione) di solito esso vale per una vasta classe di sistemi lineari e spesso consente di far convergere l'algoritmo di Gauss-Seidel, anche in casi nei casi in cui il metodo base non converge.

Terminiamo con un semplicissimo esempio che mostra l'analogia tra gli algoritmi di punto fisso e i metodi iterativi di algebra lineare.

Esempio 3.5: Usiamo Jacobi sul sistema diagonalmente dominante
$\begin{cases} 2x + y = 2 \\ -3x + 4y = 3 \end{cases}$ riscritto come $\begin{cases} 2x = 2 - y \\ 4y = 3 + 3x \end{cases}$

Geometricamente è l'intersezione di due rette.

Partiamo con $x_0 = (0, 0)$.
Sostituiamo $(0, 0)$ a secondo membro e otteniamo:
$x_1 = (1, 3/4)$
Sostituiamo $(1, 3/4)$ a secondo membro e otteniamo:
$x_2 = (5/8, 3/2)$
I primi 8 passi sono (arrotondando):
$x_1 = (1.0000, 0.7500)$ $x_2 = (0.6250, 1.5000)$
$x_3 = (0.2500, 1.2188)$ $x_4 = (0.3906, 0.9375)$
$x_5 = (0.5312, 1.0430)$ $x_6 = (0.4785, 1.1484)$
$x_7 = (0.4258, 1.1089)$ $x_8 = (0.4456, 1.0693)$
È interessante disegnare le due rette e la spezzata $x_0, x_1, ...$ che converge alla soluzione esatta.



3.1.6 Criteri d'arresto degli algoritmi iterativi

Non è facile dare una stima dell'errore commesso sostituendo alla soluzione esatta di un sistema una soluzione della successione ottenuta con un metodo iterativi e quindi non è facile dare un criterio d'arresto affidabile per un metodo che non fornisce praticamente mai la soluzione esatta.
In pratica ci si accontenta di un valore *stimato* dell'errore.
Se x_i e x_{i+1} fanno parte della successione, si può ritenere che, se si ha

$$\| x_{i+1} - x_i \| < \varepsilon$$

dove ε è un valore prefissato, l'errore assoluto sia minore di ε.
Analogamente, se

$$\frac{\| x_{i+1} - x_i \|}{\| x_{i+1} \|} < \varepsilon$$

si può ritenere che l'errore relativo sia minore di ε.
Il problema è che, come nei metodi di punto fisso in cui la convergenza è monotona, anche in questo caso il criterio non è del tutto affidabile.

Un'altro criterio di arresto è quello per cui si può reputare che x_i sia prossimo alla soluzione se Ax_i e b sono prossimi, se cioè $\| Ax_i - b \| < \varepsilon$, con ε valore prefissato.
Questo criterio è più affidabile dei precedenti, ma richiede un tempo di calcolo superiore.
Nella pratica si possono combinare i due criteri.