Prontuario di matematica

Matematica dal punto di vista tecnico

Questo documento e composto da pagine 42. Gli argomenti trattati nascono dall'esperienza dell'autore. L'autore NON può in alcun modo essere responsabile di false citazioni o errori presenti in questo documento. L'autore NON può in alcun modo essere responsabile di danni a cose e/o persone derivanti dalla lettura di questo documento.

Un feedback della lettura sarà sempre apprezzato, anche solo per la soddisfazione che il materiale possa essere utile a qualcuno ☺, si accetteranno ancor più volentieri consigli a carattere costruttivo (e "correttivo" ☺).

Un grazie anche a M. Cristina che per prima ha letto e incentivato questo documento!

05 Gennaio 2004
Cittadella – PD

Federico Milan
milan1@interfree.it

Prontuario di matematica

Matematica dal punto di vista tecnico

. Presentazione

Durante la vita lavorativa, specialmente se siamo tecnici, ci troviamo spesso a "lottare" con problematiche che esigono pochi, ma utili strumenti matematici per essere risolte nel modo "il più veloce possibile".
Questo materiale non è un libro, ne ha la pretesa di essere completo, ma nasce come un prontuario che possa essere consultato velocemente, dare spunti e chiarire dubbi sempre leciti, quando si è alle prese con problemi da risolvere velocemente.

In questo documento non vi saranno dimostrazioni e spesso e volentieri si tralascerà la rigorosità di pensiero prediligendo semplicità e concretezza.

Indice

0.0 . Notazioni

Nel seguente documento si useranno alcune notazioni per rendere veloce e leggibile il materiale. Le notazioni usate sono quelle relative al background dell'autore - del resto l'esperienza insegna che una notazione vale l'altra ☺.

$f(x)$	funzione a una variabile
$\frac{d}{dt}.$	derivata rispetto a t.
$\lvert\ \rvert$	modulo
$\begin{vmatrix} \alpha & \dots \\ \vdots & \ddots \end{vmatrix}$	matrice $n \times m$ elementi
$\lVert A \rVert$	determinante della matrice A
$X_{hex}; X_{oct}; X_{bin}; X_{dec}$	formato numerico
$j = \sqrt{-1}$	unità immaginaria

1.0 . Basi numeriche

Trasformazioni di basi numeriche vengono spesso richieste quando si ha a che fare con la programmazione in genere.

Un numero non è altro che una successione di cifre; questo ci consente di definire il significato di ciascuna cifra come segue:

$$12543 \rightarrow 1*10^4 + 2*10^3 + 5*10^2 + 4*10^1 + 3*10^0$$

quello che è stato scritto vale per la base “dieci”, ma nulla toglie che si possa generalizzare il discorso. Spesso per motivi di praticità si usano basi diverse dalla base “dieci”. Ad esempio in elettronica/programmazione si utilizza la base “due” e suoi multipli.

Data una base si è automaticamente definito il numero di cifre valide usabili. In particolare per le basi di interesse pratico si ha:

- Base “dieci” 10 DEC simboli : 0,1,2,3,4,5,6,7,8,9
- Base ”due” 2 BIN simboli : 0,1
- Base “otto” 8 OCT simboli : 0,1,2,3,4,5,6,7
- Base “sedici” 16 HEX simboli : 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

Passare da una base generica alla base “dieci” è relativamente semplice.

Es.

$$10111_{BIN} \xrightarrow{\text{BIN}\rightarrow\text{DEC}} 1*2^4 + 0*2^3 + 1*2^2 + 1*2^1 + 1*2^0 = 23_{DEC}$$

$$76543_{OCT} \xrightarrow{\text{OCT}\rightarrow\text{DEC}} 7*8^4 + 6*8^3 + 5*8^2 + 4*8^1 + 3*8^0 = 32099_{DEC}$$

$$A_{HEX} \xrightarrow{\text{HEX}\rightarrow\text{DEC}} 10_{DEC}$$

$$B_{HEX} \xrightarrow{\text{HEX}\rightarrow\text{DEC}} 11_{DEC}$$

$$C_{HEX} \xrightarrow{\text{HEX}\rightarrow\text{DEC}} 12_{DEC}$$

$$D_{HEX} \xrightarrow{\text{HEX}\rightarrow\text{DEC}} 13_{DEC}$$

$$E_{HEX} \xrightarrow{\text{HEX}\rightarrow\text{DEC}} 14_{DEC}$$

$$F_{HEX} \xrightarrow{\text{HEX}\rightarrow\text{DEC}} 15_{DEC}$$

$$3A2F_{HEX} \xrightarrow{\text{HEX}\rightarrow\text{DEC}} 3*16^3 + A*16^2 + 2*16^1 + F*16^0 = 14895_{DEC}$$

Qualche volta potrebbe essere necessario costruirsi una piccola routine per la trasformazione automatica da base generica a base "dieci". In letteratura esistono molti algoritmi dedicati a fare questo; di seguito viene proposta una delle possibili alternative in meta linguaggio:

Ipotesi	l'utente inserisce i dati da tastiera. Premerà [Enter] per il risultato.
Risoluzione	basta usare uno stack. Funzioni di sistema da predisporre input_tastierino() restituisce il carattere premuto.

```
Char   ch;
Stack  Stk;
Int    Base <- get_Base();

while( (ch<-input_tastierino()) <> "↵")
{
 stk.push(ch);
}

int valore   <- 0;
int appoggio;
int e        <- 0;

while(!stk.is_Empty())
{
 appoggio=(int)stk.pop();
 valore = valore + appoggio*Base^e;
 e=e+1;
}

return valore;
```

Ovviamente, la conversione da base generica a base "dieci", non è definita solo per numeri interi, ma anche da numeri con virgola:

$$11{,}101 \xrightarrow{BIN \to DEC} 1*2^1 + 1*2^0 + 1*2^{-1} + 1*2^{-2} + 1*2^{-3} = 3_{DEC} + \left.\frac{5}{8}\right|_{DEC} = 3{,}625_{DEC}$$

Se la conversione da base generica a base "dieci" è semplicissima, la conversione da base "dieci" a base generica è un po' meno intuitiva, o meglio necessita di alcuni accorgimenti.

$$10111_{BIN} \xrightarrow{\text{BIN}\to\text{DEC}} 1*2^4 + 0*2^3 + 1*2^2 + 1*2^1 + 1*2^0 \to$$
$$\to (((1*2+0)*2+1)*2+1)*2+1 = 23_{DEC}$$

Dalla relazione sopra scritta risulta quindi ovvio che per effettuare la trasformazione inversa si debba procedere in questo modo:

23	:	2	=	11	resto	1	Passo 1
11	:	2	=	5	resto	1	Passo 2
5	:	2	=	2	resto	1	Passo 3
2	:	2	=	1	resto	0	Passo 4

1 : 2 = 0 resto 1 Passo 5

Come è semplice verificare, leggendo i resti delle operazioni al contrario (dal passo 5 al passo 1), si ottiene la trasformazione da base "dieci" a base "due". Nello stesso modo si procede per qualunque altra base.

E se i numeri sono numeri con "virgola"?

Es.

$0{,}625_{DEC} \xrightarrow{\text{DEc}\rightarrow\text{BIN}} 0{,}101_{\text{DEC}}$ vediamo perchè

0,625	*	2 = 1,25	Parte decimale = 0,25	intera = 1	Passo 1
0,25	*	2 = 0,5	Parte decimale = 0,5	intera = 0	Passo 2
0,5	*	2 = 1,0	Parte decimale = 0,0	intera = 1	Passo 3

Leggendo la parte intera dei risultati partendo dal passo 1 al passo 3 si ottiene la rappresentazione binaria del numero in base "due".

Il discorso dei numeri negativi rappresentati in base diversa da "dieci" è alquanto delicato. Un esempio per tutti fatto per la base 2:

0,3 [DEC] vogliamo calcolarlo in base "due"

0,3	*	2 = 0,9	decimale = 0,9	intera = 0	Passo 1
0,9	*	2 = 1,8	decimale = 0,8	intera = 1	Passo 2
0,8	*	2 = 1,6	decimale = 0,6	intera = 1	Passo 3
0,6	*	2 = 1,2	decimale = 0,2	intera = 1	Passo 4
0,2	*	2 = 0,4	decimale = 0,4	intera = 0	Passo 5
0,4	*	2 = 0,8	decimale = 0,8	intera = 0	Passo 6 … ripete come passo 3

ossia si vede chiaramente che un numero decimale non periodico viene trasformato in un numero periodico. Questo potrebbe introdurre grossi errori di approssimazione ed è uno dei motivi per cui è stato adottato il formato BCD in alcuni centri di calcolo.

Il passaggio da base "due" a base "Otto" ("Sedici") avviene con una tecnica molto semplice ed intuitiva che consiste nel raggruppare le cifre a insiemi di 3 (4) e calcolare il valore di questi gruppetti nella rispettiva base, quindi sostituire le cifre con il valore trovato; di seguito un esempio (che vale di più di mille parole):

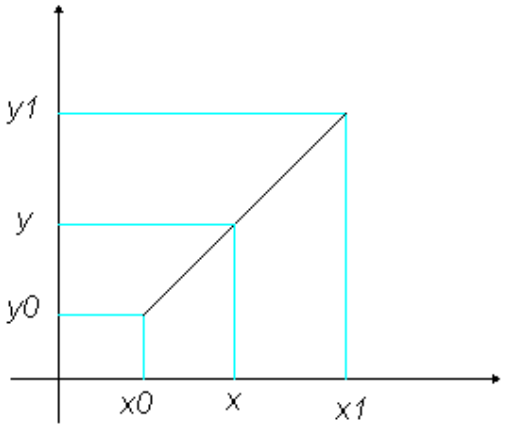
$1001110101_{BIN} \rightarrow 1;001;110;101 \xrightarrow{BIN\rightarrow OCT} 1165_{OCT}$

$1111011001_{BIN} \rightarrow 11;1101;1001 \xrightarrow{BIN\rightarrow HEX} 3D9_{HEX}$

2.0 . Linearizzazione

Il problema della linearizzazione è spesso affrontato quando si devono convertire grandezze, oppure si debba normalizzare dati secondo valori di riferimento.
Uno dei casi più frequenti in automazione è la conversione di una grandezza analogica. Questa è letta in punti decimali e la si vorrebbe leggere in numero decimale o in unità di misura desiderata dall'utente.

La problematica è molto semplice e di seguito ne viene proposto una soluzione grafica.



Dove x0;x1 sono i limiti minimo e massimo della grandezza di ingresso, mentre y0;y1 sono i rispettivi minimo e massimo della grandezza convertita:

$$x_0 \le x \le x_1; y_0 \le y \le y_1;$$

$$\frac{x - x_0}{x_1 - x_0} = \frac{y - y_0}{y_1 - y_0} \rightarrow y = \frac{y_1 - y_0}{x_1 - x_0} * (x - x_0) + y_0$$

dato che ci sono parecchie parti costanti, conviene rivedere la formula come:

$$k_1 = \frac{y_1 - y_0}{x_1 - x_0}; k_2 = y_0 - k_1 * x_0$$

$$y = k_1 * x + k_2$$

Di norma conviene tenere presente, nell'eventuale algoritmo di linearizzazione, anche la presenza di soglie di saturazione della eventuale grandezza di ingresso.

3.0 . Operazioni con i numeri interi

Spesso capita di dover lavorare su calcolatori o PLC utilizzando numeri interi rappresentati da un certo numero di bit; 8-16-32 bit sono valori normalmente usati.

I simboli massimi rappresentati sono dati dalla formula

$$N_{simbolo} = 2^n ; N^+_{simbolo} = 2^{n-1} - 1; N^-_{simbolo} = 2^{n-1};$$

$$byte \rightarrow 2^8 = 256_{simboli}; 2^{8-1} - 1 = 127_{simboli\_positivi}; 2^{8-1} = 128_{simboli\_negativi};$$

$$\left.\begin{aligned} Numero_{\max} &= 2^{8-1} - 1 = +127 \\ Numero_{\min} &= -2^{8-1} = -128 \end{aligned}\right\} \text{per numeri con segno}$$

$$\left.\begin{aligned} Numero_{\max} &= 2^8 - 1 = +255 \\ Numero_{\min} &= 0 \end{aligned}\right\} \text{per numeri senza segno (unsigned)}$$

Stesso discorso per qualunque altra dimensione
di regola le dimensioni più usate sono
n = 8;16;32 (Raramente 64) per numeri interi

Ricordarsi che nei 256 simboli vi è anche lo zero.

3.1.0 . definizione prime operazioni

Su questi numeri sono definite due operazioni molto utili che, spesso, vengono chiamate inversione (complemento a 1) e negazione (complemento a 2). A queste poi si possono associare operazioni di OR, AND, XOR.

- **3.1.1 .** Complemento a 1

  Il complemento a 1 è un'operazione utile soprattutto quando si utilizza la base "due" e può assumere significati particolari per il calcolo che si sta eseguendo (a volte è utile ragionare in modo "negato").

  [BIN]
  `00110101 [inv]-> 11001010`

  Come si intuisce, l'inversione o complemento a 1 non è altro che invertire ciascun simbolo, cioè trasformare `1->0` e lo `0->1`.

- **3.1.2 .** Complemento a 2

  Il complemento a 2 è l'operazione di negazione. Essa deve essere definita sulla dimensione del numero di cifre rappresentato, cioè ha significato per numeri limitati. Questa operazione consente, dato una rappresentazione base "due" di un numero, di definire delle regole per invertire un numero quindi avere la possibilità di "giocare" con numeri negativi ☺.

Banalmente essa consiste (con rappresentazione in base “due”) nell’invertire tutti i simboli del numero rappresentato e sommare 1. Di seguito viene proposto un esempio:

```
[BIN]
00110101 [neg]-> 11001010 + 00000001 = 11001011

Questo esempio è riferito ad un byte
```

In questo modo, data l’operazione di somma e l’operazione di complemento a due, abbiamo la possibilità di eseguire somme e sottrazioni .

Es.
lavorando con numeri a 8 bit si ha che:

```
5 [DEC] - 3 [DEC] = 2 [DEC]
5[DEC] -> 00000101[BIN]
3[DEC] -> 00000011[BIN]
Neg(00000011[BIN])->11111101

 00000101[BIN]+
 11111101[BIN]=
 -------------
100000010[BIN]
^
Questa cifra viene persa ed il risultato è
00000010[BIN] -> 2[DEC]
```

3.2.0 . Altre operazioni di interesse

Le operazioni logiche fatte su numeri hanno significato uguale alle operazioni logiche fatte su bit. Per queste operazioni conviene sempre rappresentare i numeri in base "due" o base "sedici". Le operazioni più usate sono le operazioni di AND, XOR, OR ciascuna seguita o meno dall'operazione di inversione (ottenendo quindi NAND, EXOR, NOR).

- **3.2.1 .** AND

  L'operazione di And spesso è usata per "filtrare" parte di un numero. Il filtro viene fatto con un numero rappresentato in base "due" o "sedici" chiamato anche maschera. Un esempio potrebbe essere il seguente:

  Es.
  Dato un numero intero a 16 bit si vuole vedere se è pari o dispari.

  Il problema risulta banale in quanto si ha che:

  $\text{Dispari}: 2^n + 1$

  $\text{Pari} \quad : 2^n$

  Quindi risulta ovvio che il bit meno significativo del nostro numero dovrà essere uguale a 0 se il numero è pari, e a 1 se è dispari.

  Il problema quindi è risolubile in modo estremamente semplice utilizzando una maschera di bit tale da poter verificare se l'ultimo bit è uguale o meno a 0.

```
int Test <- get_Num();

int Maschera = 0000_0000_0000_0001[BIN];

Test = Test AND Maschera;

if (Test = 1)
{
     return "Dispari";
} else
     {
          return "Pari";
     }
```

- **3.2.2 .** XOR

  L'operazione di XOR è spesso usata per monitorare variazioni, cioè la proprietà dello XOR è di evidenziare differenze - basta ricordare la mappa di verità.

XOR	0	1
0	0	1
1	1	0

  Lo XOR è utile, quindi, per operazioni logiche piuttosto che operazioni puramente di calcolo (prendete questa affermazione con le pinze ☹).

- **3.2.3 .** OR

  L'operazione di OR può essere utilizzata oltre che come operazione logica, anche per manipolazione numerica.

  Spesso, infatti, si utilizzano numeri a 16 bit, ma solo una parte di questi formano la quantità numerica e i rimanenti sono bit di controllo. Ad esempio si potrebbe pensare di avere un numero a 15 bit e il bit più significativo essere usato come bit di segno (non si confonda questa tecnica con il complemento a 2).

```
Vogliamo con questa tecnica negare il numero 5.

5[DEC] ->0_0_0_5[HEX]

0_0_0_5[HEX] OR
8_0_0_0[HEX] =
---------------
8_0_0_5[HEX]

Questa volta si è prediletta la notazione in
base "sedici" in quanto è molto più compatta.
Basta ricordarsi che 8[HEX] -> 1000[BIN] e il
gioco è fatto.
```

3.3.0 . Primi accenni di calcolo numerico

I numeri interi a rappresentazione limitata vengono spesso usati anche nel calcolo "numerico". Esiste una letteratura specializzata per questo campo di problematiche, ma per la realtà di tutti i giorni bastano le seguenti nozioni di base per evitare problemi ☺.

Il problema principale da evitare è l'overflow. Basti immaginare di avere un byte e di dover moltiplicare il numero 32 per il numero 4. Lavorando con numeri con segno avremmo un owerflow del calcolo e il risultato non sarà +128 ma –128.
È quindi importante in questi casi inserire nel codice istruzioni per saturare il calcolo.

```
byte m   <- get_Num();
byte n   <- get_Num();
dword  app; // 32_bit

app = (dword)m*(dword)n;
if (app >  127) return (byte) 127;
if (app < -128) return (byte)-128;
return (byte)app;


Il codice non fa altro che eseguire in modo corretto la
moltiplicazione, il cast (dword)m non fa altro che
estendere il segno del numero a 8 bit al nuovo numero a
32 bit.
Quindi si procede a controllare se vi è un owerflow e
in presenza a saturare il risultato.
```

L'utilizzo di un formato piuttosto che di un altro è deciso non solo sulla base della precisione (fattore del resto molto importante), ma anche sulla velocità di calcolo dell'elaboratore utilizzato (esempio: un processore a 16 bit lavorerà su dati a 16 bit più velocemente che su dati a 32 o 64 bit a meno di non disporre di particolare co-processori). Le operazioni di cast vanno quindi fatte in modo ponderato.

Oltre agli errori di overflow, con i numeri a rappresentazione limitata, si può incorrere anche in altre problematiche. Tra le più comuni problematiche che ci si imbatte vi sono le operazioni di divisione, soprattutto se queste sono seguite da altre operazioni. Un esempio potrebbe essere: $y = \frac{x_1}{x_2} * x_3$ un calcolo così fatto potrebbe portare a seri errori, basti il seguente esempio:

$$\frac{5}{2} * 2 = 2 * 2 = 4 \neq 5$$

per evitare questi problemi conviene quindi trasformare così l'espressione di calcolo:

$$y = \frac{x_1 * x_3}{x_2}$$

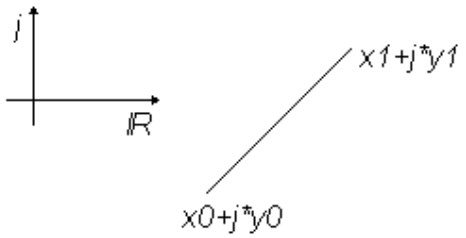
$$\frac{5*2}{2} = \frac{10}{2} = 5$$

dando la precedenza alla moltiplicazione e poi eseguendo la divisione, si riesce ad aumentare la precisione di calcolo.

4.0 . Numeri complessi

I numeri complessi dovrebbero essere parte integrante del background di ciascun tecnico con la T maiuscola ☺, anche perché sono rappresentazioni della realtà e aiutano spesso a risolvere parecchie problematiche.

Un numero complesso può anche essere visto come la rappresentazione di uno spazio piano. Tramite i numeri complessi si possono rappresentare e calcolare distanze in modo veloce e intuitivo.



Come risulta dal disegno, ipotizziamo di avere un segmento nel piano. Stabilita l'origine possiamo segnare i due estremi con due punti, ciascuno contrassegnato con una coppia di numeri rappresentante la distanza rispetto all'asse delle ascisse e delle ordinate. Se questi punti li trasformiamo in coordinate complesse, risulta semplice arrivare alle seguenti conclusioni:

$$P_0 = x_0 + j * y_0;$$
$$P_1 = x_1 + j * y_1;$$
$$\left|\overline{P_0 P_1}\right| = \left|x_1 - x_0 + j * (y_1 - y_0)\right|$$

Potrebbe sembrare una complicazione, ma in realtà molti calcolatori tascabili o software di calcolo dispongono di strumenti per analisi complesse e questo ci consente di sfruttarne le potenzialità. Anzi, possiamo, in modo automatico, passando da coordinate cartesiane a coordinate polari, avere anche a disposizione l'angolo del segmento riferito ad un asse del nostro riferimento.

Le regole di calcolo dei numeri complessi si possono riassumere come:

$$\left|x + j * y\right| = \sqrt{x^2 + y^2} \rightarrow \lambda; \arg(x + j * y) = \arctan(\frac{y}{x}) \rightarrow \langle\theta; x + j * y \leftrightarrow \lambda\langle\theta$$
$$(x_1 + j * y_1) + (x_2 + j * y_2) = (x_1 + x_2) + j * (y_1 + y_2);$$
$$(x_1 + j * y_1) * (x_2 + j * y_2) = (x_1 * x_2 - y_1 * y_2) + j * (x_1 * y_2 + x_2 * y_1)$$
$$\lambda_1\langle\theta_1 * \lambda_2\langle\theta_2 = \lambda_1\lambda_2\langle(\theta_1 + \theta_2); \frac{\lambda_1\langle\theta_1}{\lambda_2\langle\theta_2} = \frac{\lambda_1}{\lambda_2}\langle(\theta_1 - \theta_2)$$

Esistono moltissimi altri teoremi e regole, ma credo che queste siano più che sufficienti per i problemi quotidiani.

5.0 . Algebra lineare

Un'altra disciplina molto utile per noi tecnici è l'algebra lineare. Spesso può sembrare ostica, ma a fronte di un minimo sforzo di apprendimento consente di velocizzare molte operazioni, soprattutto se si è supportati da strumenti di calcolo numerico adeguati (esistono parecchi strumenti CAS sia commerciali che gratuiti).

- **5.0.1 .** Risoluzione di sistemi lineari

  Risolvere un sistema lineare significa prima di tutto conoscere se il sistema sarà risolubile o meno. Un sistema lineare di *m* equazioni verrà rappresentato come segue:

$$\begin{cases} a_{11} * x_1 + a_{21} * x_2 + ... + a_{n1} * x_n + b_{11} + b_{21} + ... = 0 \\ \qquad\qquad\qquad\qquad\qquad\qquad ... \\ a_{1m} * x_1 + a_{2m} * x_2 + ... + a_{nm} * x_n + b_{1m} + b_{2m} + ... = 0 \end{cases}$$

$$b_1 = b_{11} + b_{21} + ...$$
$$\vdots$$
$$b_m = b_{1m} + b_{2m} + ...$$

$$\begin{cases} a_{11} * x_1 + a_{21} * x_2 + ... + a_{n1} * x_n + b_1 = 0 \\ \qquad\qquad\qquad\qquad ... \\ a_{1m} * x_1 + a_{2m} * x_2 + ... + a_{nm} * x_n + b_m = 0 \end{cases}$$

  dove gli [a] sono i coefficienti delle incognite.

  Esistono svariati metodi di risoluzione. Certo che, per sistemi con più di due incognite, è meglio scordarsi il metodo di sostituzione. A semplificare le risoluzioni viene in aiuto una parte della teoria delle matrici.
  Una matrice non è altro che una tabella di simboli. Nel campo delle matrici sono definite alcune operazioni e trasformazioni utili per manipolare le matrici stesse.

5.1.0 . Operazioni elementari sulle matrici

Le operazioni che ci interessano dal punto di vista tecnico sulle matrici sono soltanto 4 e sono riassunte di seguito:

- Somma tra matrici
- Prodotto matrice per scalare
- Prodotto tra matrici
- Inversa di una matrice – Determinante - Trasposta

5.1.1 . Somma tra matrici

$$\begin{vmatrix} a_{11} & a_{21} & \cdots & a_{m1} \\ a_{12} & a_{22} & \cdots & a_{m2} \\ \vdots & \vdots & \vdots & \vdots \\ a_{1n} & a_{2n} & \cdots & a_{mn} \end{vmatrix} + \begin{vmatrix} b_{11} & b_{21} & \cdots & b_{m1} \\ b_{12} & b_{22} & \cdots & b_{m2} \\ \vdots & \vdots & \vdots & \vdots \\ b_{1n} & b_{2n} & \cdots & b_{mn} \end{vmatrix} = \begin{vmatrix} a_{11}+b_{11} & a_{21}+b_{21} & \cdots & a_{m1}+b_{m1} \\ a_{12}+b_{12} & a_{22}+b_{22} & \cdots & a_{m2}+b_{m2} \\ \vdots & \vdots & \vdots & \vdots \\ a_{1n}+b_{1n} & a_{2n}+b_{2n} & \cdots & a_{mn}+b_{mn} \end{vmatrix}$$

La somma è definita solo per matrici aventi dimensioni uguali, come del resto risulta ovvio dalla definizione stessa. In letteratura esistono disposizioni particolari dei pedici. Tuttavia per i tecnici basta essere coerenti solo nei calcoli, quindi non ci interessa stabilire se il primo numero del pedice sono le colonne o le righe - dal momento che risulta chiaro non appena scriveremo la nostra matrice.

```
Un eventuale algoritmo per eseguire la somma di 2 matrici
potrebbe essere il seguente:

bastano 2 cicli innestati, uno scorre le colonne e uno le
righe (o viceversa, a seconda di cosa piaccia o si riesca
ad ottimizzare … ☺):

array[n][m] Mat_A <- get_Mat();
array[n][m] Mat_B <- get_Mat();
array[n][m] Mat_R;

for (int i = 1, i++, i<=n)
{
      for (int j = 1, j++, j<=m)
      {
            Mat_R[i][j]= Mat_A[i][j]+ Mat_B[i][j];
      }
}
```

5.1.2 . Prodotto matrice per scalare

$$\lambda * \begin{vmatrix} a_{11} & a_{21} & \cdots & a_{m1} \\ a_{12} & a_{22} & \cdots & a_{m2} \\ \vdots & \vdots & \vdots & \vdots \\ a_{1n} & a_{2n} & \cdots & a_{mn} \end{vmatrix} = \begin{vmatrix} \lambda * a_{11} & \lambda * a_{21} & \cdots & \lambda * a_{m1} \\ \lambda * a_{12} & \lambda * a_{22} & \cdots & \lambda * a_{m2} \\ \vdots & \vdots & \vdots & \vdots \\ \lambda * a_{1n} & \lambda * a_{2n} & \cdots & \lambda * a_{mn} \end{vmatrix}$$

Anche questa operazione è estremamente semplice e molto intuitiva, del resto è una operazione di fondamentale importanza per molte problematiche tecniche.

5.1.3 . Prodotto tra matrici

Il prodotto tra matrici risulta meno intuitivo se non si ha alle spalle la conoscenza minima di combinazione lineare (infatti, per chi ne abbia una vaga rimembranza, risulta estremamente lampante il significato della moltiplicazione tra matrici). Diciamo però che, dal punto di vista tecnico, non è altro che l'applicazione di una definizione.

Nei libri di testo spesso si trova che la moltiplicazione di matrici viene eseguita *RIGHE x COLONNE,* ma sicuramente è più semplice vedere con un esempio cosa si intende.

$$\begin{vmatrix} a_{11} & a_{21} & \cdots & a_{n1} \\ a_{12} & a_{22} & \cdots & a_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1l} & a_{2l} & \cdots & a_{nl} \end{vmatrix} * \begin{vmatrix} b_{11} & b_{21} & \cdots & b_{h1} \\ b_{12} & b_{22} & \cdots & b_{h2} \\ \vdots & \vdots & \ddots & \vdots \\ b_{1n} & b_{2n} & \cdots & b_{hn} \end{vmatrix} =$$

$$= \begin{vmatrix} (a_{11} * b_{11} + a_{21} * b_{12} + \ldots + a_{n1} * b_{1n}) & (a_{11} * b_{21} + \ldots + a_{n1} * b_{2n}) & \cdots & (a_{11} * b_{h1} + \ldots + a_{n1} * b_{hn}) \\ (a_{12} * b_{11} + a_{22} * b_{12} + \ldots + a_{n2} * b_{2n}) & (a_{12} * b_{21} + \ldots + a_{n2} * b_{2n}) & \cdots & (a_{12} * b_{h1} + \ldots + a_{n2} * b_{2n}) \\ \vdots & \vdots & \ddots & \vdots \\ (a_{1l} * b_{11} + a_{2l} * b_{12} + \ldots + a_{nl} * b_{1n}) & (a_{1l} * b_{21} + \ldots + a_{nl} * b_{2n}) & \cdots & (a_{1l} * b_{h1} + \ldots + a_{nl} * b_{hn}) \end{vmatrix} =$$

$$= \begin{vmatrix} c_{11} & c_{21} & \cdots & c_{h1} \\ d_{12} & d_{22} & \cdots & d_{h2} \\ \vdots & \vdots & \ddots & \vdots \\ c_{1l} & c_{2l} & \cdots & c_{hl} \end{vmatrix}$$

Il prodotto tra matrici è molto restrittivo. Guardando la regola, si intuisce che le colonne della prima matrice devono essere uguali alle righe della seconda matrice. Non è detto che si possano invertire le matrici, e di regola, se si invertono, non si ottiene lo stesso risultato.
La matrice risultante avrà dimensioni diverse dalle matrici di cui si è fatto il prodotto.

La regola mostra che la moltiplicazione di due matrici di dimensione (*[Colonne x Righe]*) *[n x l] e [h x n]* avrà dimensione *[h x l]*.

Es.
Data una matrice riga e una matrice colonna il risultato è:

$$\begin{bmatrix}1 & 2 & 3\end{bmatrix} * \begin{bmatrix}3\\2\\1\end{bmatrix} = \begin{bmatrix}1*3+2*2+3*1\end{bmatrix} = \begin{bmatrix}10\end{bmatrix} = 10$$

che è uno scalare (ma del resto non deve stupire ☺).

5.1.4 . Inversa di una matrice – Determinate - Trasposta

Come sempre, per definire l'inversa di una matrice, è necessario parecchia teoria che potremmo ridurre al minimo. Anzi, più che ridurre al minimo, dedurremmo:

$$a^{-1} = \frac{1}{a} \rightarrow a * a^{-1} = 1 \text{[inverso di uno scalare]}$$

$$I = \begin{bmatrix}1 & 0 & 0 & 0\\0 & 1 & 0 & 0\\\vdots & \vdots & \ddots & \vdots\\0 & 0 & 0 & 1\end{bmatrix} \quad \text{[matrice identica, "unità"]}$$

$$\mathrm{A} = \begin{bmatrix}\mathrm{a}_{11} & \cdots & \mathrm{a}_{\mathrm{n1}}\\\vdots & \ddots & \vdots\\\mathrm{a}_{1\mathrm{n}} & \cdots & \mathrm{ann}\end{bmatrix}$$

$$A * B = I \rightarrow B = A^{-1}$$

Sempre da un punto di vista puramente tecnico, tralasciamo molte dimostrazioni e proprietà importanti, ma dobbiamo però precisare che l'inversa di una matrice esiste solo se la matrice è *quadrata [n x n]*. Un'altra cosa fondamentale è che non è detto che una matrice sia invertibile, però, se essa è invertibile, allora vale anche $A * B = I \rightarrow B * A = I \rightarrow B = A^{-1} \rightarrow A = B^{-1}$.
Può sembrare una banalità, ma è opportuno ricordarselo.

Purtroppo trovare l'inversa di una matrice non è molto semplice, anzi è necessario eseguire parecchi calcoli. Questo non deve preoccupare, tanto c'è il calcolatore che li farà per noi ☺. Però un po' di teoria, potrebbe aiutarci nel caso in cui fossimo dispersi in chissà quale parte del mondo e impossibilitati a reperire gli accumulatori per il nostro fido calcolatore.

Prima di dare alcuni metodi per il calcolo dell'inversa, è opportuno chiarire cosa sia il determinante di una matrice. Niente di rigoroso: in poche parole il determinante è un numero scalare. Cosa possa servire è presto detto (ovviamente non è solo così limitato):

- è un parametro che ci consente di definire se la matrice è invertibile o meno.

Visto la complessità per determinare l'inversa di una matrice, avere uno strumento per sapere se è o meno invertibile è molto utile. Vi sono più strumenti per verificare se una matrice è invertibile, però qui ne tratterremo solo 2 e cominciamo con quello più difficile (appunto il determinate).

1. $\|k\| = k$
2. $\begin{Vmatrix} a & b \\ c & d \end{Vmatrix} = a*d - b*c$
3. $\begin{Vmatrix} a_{11} & a_{21} & \cdots & a_{n1} \\ a_{12} & a_{22} & \cdots & a_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & \cdots & a_{nn} \end{Vmatrix} =$

$$= a_{11} * \begin{Vmatrix} a_{22} & \cdots & a_{n2} \\ \vdots & \ddots & \vdots \\ a_{2n} & \cdots & a_{nn} \end{Vmatrix} + (-1)^{2+1} a_{21} * \begin{Vmatrix} a_{12} & a_{32} & \cdots a_{n2} \\ \vdots & \vdots & \ddots \vdots \\ a_{1n} & a_{3n} & \cdots a_{nn} \end{Vmatrix} + \cdots + (-1)^{n+1} a_{n1} * \begin{Vmatrix} a_{12} & \cdots & a_{n-1;2} \\ \vdots & \ddots & \vdots \\ a_{1n} & \cdots & a_{n-1;n} \end{Vmatrix}$$

.1 il determinate di uno scalare è lo scalare stesso
.2 il determinante di una matrice *[2 x 2]* è dato dalla formuletta
.3 il determinante di una matrice *[n x n]* è definito in modo ricorsivo come indicato e, come si vede, bisogna stare attenti ai segni definiti appunto da $(-1)^{i+j}$, dove *i* e *j* sono i pedici dell'elemento a_{ij}.

Come anticipato il calcolo del determinante è qualcosa di complesso, ma osservando bene il punto tre, ci si accorge che, se la nostra matrice quadrata è in forma triangolare, il calcolo del determinante è alquanto semplice, anzi banale, e si traduce nella moltiplicazione degli elementi della diagonale.

$$\begin{Vmatrix} a_{11} & a_{21} & \cdots & a_{n1} \\ 0 & a_{22} & \cdots & a_{n2} \\ \vdots & 0 & \ddots & \vdots \\ 0 & \cdots & 0 & a_{nn} \end{Vmatrix} = a_{11} * a_{22} * \cdots * a_{nn}$$

Quindi la domanda è – "… esiste un modo per trasformare una qualunque matrice in matrice triangolare? …" – e la risposta è – ".. Si! ...".

Per fare questo sarebbe opportuno scomodare alcune teorie e teoremi, che alla fine daremo per scontati; quindi, senza entrare nel dettaglio, si applicheranno delle regolette banali che ipotizziamo sempre valide per i nostri scopi.
Tengo solo a sottolineare che la matrice triangolare esiste anche per matrici non quadrate e conviene sapere questa tecnica perché fra qualche paragrafo ritornerà utile ☺.

Verrà mostrato un caso pratico, qualunque altro caso sarà una "copia" adattata:

$$\begin{vmatrix} a_{11} & a_{21} & a_{31} \\ a_{12} & a_{22} & a_{32} \\ a_{13} & a_{23} & a_{33} \end{vmatrix} = \begin{vmatrix} a_{11} & a_{21} & a_{31} \\ \lambda * a_{12} - \eta * a_{11} & \lambda * a_{22} - \eta * a_{21} & \lambda * a_{32} - \eta * a_{31} \\ a_{13} & a_{23} & a_{33} \end{vmatrix} =$$

$$\begin{vmatrix} a_{11} & a_{21} & a_{31} \\ 0 & c_{22} & c_{32} \\ a_{13} & a_{23} & a_{33} \end{vmatrix} = \begin{vmatrix} a_{11} & a_{21} & a_{31} \\ 0 & c_{22} & c_{32} \\ \mu * a_{13} - \nu * a_{11} & \mu * a_{23} - \nu * a_{21} & \mu * a_{33} - \nu * a_{31} \end{vmatrix} =$$

$$\begin{vmatrix} a_{11} & a_{21} & a_{31} \\ 0 & c_{22} & c_{32} \\ 0 & d_{23} & d_{33} \end{vmatrix} = \begin{vmatrix} a_{11} & a_{21} & a_{31} \\ 0 & c_{22} & c_{32} \\ 0 & \varpi * d_{23} - \vartheta * c_{22} & \varpi * d_{33} - \vartheta * c_{32} \end{vmatrix} = \begin{vmatrix} a_{11} & a_{21} & a_{31} \\ 0 & c_{22} & c_{32} \\ 0 & 0 & g_{33} \end{vmatrix}$$

Come si vede dall'esempio le operazioni da eseguire sono molto semplici e intuitive. Per pura generalità si sono usate lettere, ma risulta abbastanza ovvio che (per l'esempio i valori più intuitivi da dare alle costanti sono):

$\lambda = a_{11}; \eta = a_{12}$
$\mu = a_{11}; \nu = a_{13}$
$\varpi = c_{22}; \vartheta = d_{23}$

Con questo metodo siamo riusciti a trasformare una qualunque matrice in forma triangolare. Risulta anche intuitivo che basta una riga (colonna) nulla e il determinante è uguale a 0. Quindi, se determinante è diverso da zero la matrice è invertibile, altrimenti la matrice non è invertibile.

A questo punto abbiamo a disposizione il metodo per calcolare il determinate, e per verificare se una matrice è invertibile.

Manca ora solo un'ultima operazione molto banale, che è chiamata trasposta di una matrice. Fare la trasposta significa invertire le colonne con le righe:

$$\begin{vmatrix} a_{11} & \cdots & a_{m1} \\ \vdots & \ddots & \vdots \\ a_{1n} & \cdots & a_{mn} \end{vmatrix} \xrightarrow{Trasposta} \begin{vmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{nm} \end{vmatrix}$$

La trasposta si indica anche così :

$$\begin{vmatrix} a_{11} & \cdots & a_{m1} \\ \vdots & \ddots & \vdots \\ a_{1n} & \cdots & a_{mn} \end{vmatrix}_{-1} = \begin{vmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{nm} \end{vmatrix}$$

Non volendo utilizzare teoremi o altri strumenti teorici (molto potenti, ma che ci vorrebbe un libro solo per loro) mostriamo i metodi più comuni per "invertire" una matrice applicandoli a casi generici.

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix}^{-1} = \frac{1}{\begin{Vmatrix} a & b \\ c & d \end{Vmatrix}} * \begin{vmatrix} d & -b \\ -c & a \end{vmatrix} = \frac{1}{a*d-b*c} * \begin{vmatrix} d & -b \\ -c & a \end{vmatrix} \xrightarrow{DIMOSTRAZIONE}$$

$$\rightarrow \frac{1}{a*d-b*c} * \begin{vmatrix} d & -b \\ -c & a \end{vmatrix} * \begin{vmatrix} a & b \\ c & d \end{vmatrix} = \frac{1}{a*d-b*c} \begin{vmatrix} d*a-b*c & d*b-b*d \\ -c*a+a*c & -c*b+a*d \end{vmatrix} =$$

$$= \frac{1}{a*d-b*c} \begin{vmatrix} d*a-b*c & 0 \\ 0 & -c*b+a*d \end{vmatrix} = \begin{vmatrix} \frac{d*a-b*c}{a*d-b*c} & 0 \\ 0 & \frac{-c*b+a*d}{a*d-b*c} \end{vmatrix} =$$

$$= \begin{vmatrix} 1 & 0 \\ 0 & 1 \end{vmatrix}$$

Come era intuibile il caso di matrici *[2 x 2]* è tra i più semplici, ma non per questo banalissimo. Come si vede, comunque, la regola è alquanto generica. Si noti che se la matrice fosse non invertibile il determinante è pari a zero e la divisione per zero non è ammessa.
Per gli altri casi sviluppiamo un esempio numerico – certo che comunque rimarremo generici – anche perché i calcoli in gioco diventano parecchi e si appesantirebbe in tal modo troppo la lettura.

Vogliamo invertire, se possibile, la matrice $\begin{vmatrix} 1 & 2 & 3 \\ 0 & 1 & 0 \\ 3 & 0 & 0 \end{vmatrix}$. Per fare questo abbiamo visto, dalla teoria precedente, che conviene prima calcolare il determinante, quindi verificare se è diverso da zero, e solo all'ora proseguire:

$$\begin{Vmatrix} 1 & 2 & 3 \\ 0 & 1 & 0 \\ 3 & 0 & 0 \end{Vmatrix} = -9$$

Il determinante vale –9 (è diverso da zero) quindi la matrice è invertibile.
Per iniziare a invertire la matrice si deve prima di tutto calcolarne la trasposta; otteniamo in questo modo: $\begin{vmatrix}1&2&3\\0&1&0\\3&0&0\end{vmatrix}_{-1} = \begin{vmatrix}1&0&3\\2&1&0\\3&0&0\end{vmatrix}$. Eseguita questa operazione si dovrà sostituire a ciascun elemento della matrice trasposta il suo minore complementare, ossia il determinante della sottomatrice quadrata ottenuta togliendo la riga e la colonna che incrociano l'elemento scelto.

$\begin{vmatrix}1&0&3\\2&1&0\\3&0&0\end{vmatrix}$ sostituiamo così a ciascun termine

$$a_{11} = (-1)^{1+1}\begin{vmatrix}1&0\\0&0\end{vmatrix}; a_{21} = (-1)^{2+1}\begin{vmatrix}2&0\\3&0\end{vmatrix}; a_{31} = (-1)^{3+1}\begin{vmatrix}2&1\\3&0\end{vmatrix}$$
$$a_{12} = (-1)^{1+2}\begin{vmatrix}0&3\\0&0\end{vmatrix}; a_{22} = (-1)^{2+2}\begin{vmatrix}1&3\\3&0\end{vmatrix}; a_{32} = (-1)^{3+2}\begin{vmatrix}1&0\\3&0\end{vmatrix}$$
$$a_{13} = (-1)^{1+3}\begin{vmatrix}0&3\\1&0\end{vmatrix}; a_{23} = (-1)^{2+3}\begin{vmatrix}1&3\\2&0\end{vmatrix}; a_{33} = (-1)^{3+3}\begin{vmatrix}1&0\\2&1\end{vmatrix}$$

Ottenendo quindi :

$$\begin{vmatrix}(-1)^{1+1}\begin{vmatrix}1&0\\0&0\end{vmatrix} & (-1)^{2+1}\begin{vmatrix}2&0\\3&0\end{vmatrix} & (-1)^{3+1}\begin{vmatrix}2&1\\3&0\end{vmatrix}\\ (-1)^{1+2}\begin{vmatrix}0&3\\0&0\end{vmatrix} & (-1)^{2+2}\begin{vmatrix}1&3\\3&0\end{vmatrix} & (-1)^{3+2}\begin{vmatrix}1&0\\3&0\end{vmatrix}\\ (-1)^{1+3}\begin{vmatrix}0&3\\1&0\end{vmatrix} & (-1)^{2+3}\begin{vmatrix}1&3\\2&0\end{vmatrix} & (-1)^{3+3}\begin{vmatrix}1&0\\2&1\end{vmatrix}\end{vmatrix} =$$

$$= \begin{vmatrix}0 & 0 & (-1)^4*(-3)\\ 0 & (-1)^{2+2}*(-9) & 0\\ (-1)^{1+3}*(-3) & (-1)^{2+3}(-6) & (-1)^6 1\end{vmatrix} = \begin{vmatrix}0&0&-3\\0&-9&0\\-3&6&1\end{vmatrix}$$

Per ottenere l'inversa non ci resta che moltiplicare questa matrice per il determinante prima trovato

$$\begin{vmatrix}0 & 0 & \frac{-3}{-9}\\ 0 & \frac{-9}{-9} & 0\\ \frac{-3}{-9} & \frac{6}{-9} & \frac{1}{-9}\end{vmatrix} = \begin{vmatrix}0&0&\frac{1}{3}\\0&1&0\\\frac{1}{3}&-\frac{2}{3}&-\frac{1}{9}\end{vmatrix} \xrightarrow{Verifica} \begin{vmatrix}0&0&\frac{1}{3}\\0&1&0\\\frac{1}{3}&-\frac{2}{3}&-\frac{1}{9}\end{vmatrix} * \begin{vmatrix}1&2&3\\0&1&0\\3&0&0\end{vmatrix} = \begin{vmatrix}1&0&0\\0&1&0\\0&0&1\end{vmatrix}$$

Questo esempio può essere esteso a qualunque tipo di matrice quadrata. La difficoltà maggiore, del resto, risiede proprio sul calcolo del determinante che dovrà essere eseguito (*n x n* + 1) volte (dove *n* è la dimensione della matrice).

Come piccola disgressione si può analizzare cosa significa implementare il calcolo dell'inversa su un calcolatore. Risulta chiaro che il problema è abbastanza complesso; per semplificare, però, si dovrà predisporre di alcune funzioni base.
La prima funzione base da implementare sarà il calcolo del determinante che potrà essere fatto in modo ricorsivo o iterativo (in letteratura si trovano centinaia di algoritmi pronti all'uso).
In secondo luogo sarà necessario procedere con la costruzione di una funzione capace di estrarre i minimi complementari da sostituire alla trasposta della matrice iniziale.
Servirà, quindi, anche una funzione in grado di restituire la trasposta della matrice da invertire.
Per il resto il gioco è fatto.

```
Mat(m,n)              :     oggetto che rappresenta una matrice
                            di m colonne e n righe.
Mat Tras(Mat M)       :     Funzione in grado di restituire la
                            Trasposta della matrice M.
int Det(Mat M)        :     Funzione in grado di restituire il
                            determinate della matrice M.
Mat MC(Mat M; int i,j): Funzione in grado di restituire il
                            minimo complementare dell'elemento
                            M[i,j].


int n <- Read_num();
Mat M = new Mat(n,n);

M <- Read_Mat();

int Determinante = Det(M);

if (Determinante <> 0)
{
      Mat M_1 = Tras(M);
      Mat M_2 = new Mat(n,n);
      int app;
      for(int i=1, i<=n, i++)
      {
            for(int j=1, j<=n, j++)
            {
                  app = ((-1)^(i+j))*Det(MC(M_1,i,j));
                  M_2[i][j]=app/Determinante;
            }
      }

      Return M_2;

} else
  {
      return Null;
  }
```

Questo potrebbe essere il punto di partenza per costruirsi il proprio "calcolatore matriciale".

Ora vediamo un altro metodo per invertire una matrice. È alquanto banale da implementare sul calcolatore ed estremamente semplice da ricordare.

Invertiremo sempre la solita matrice di prima $\begin{vmatrix} 1 & 2 & 3 \\ 0 & 1 & 0 \\ 3 & 0 & 0 \end{vmatrix}$ tenendo valido quanto detto poco sopra – ossia che l'esempio mantiene genericità anche per matrici quadrate di dimensione diversa.

$\begin{vmatrix} 1 & 2 & 3 \\ 0 & 1 & 0 \\ 3 & 0 & 0 \end{vmatrix}$ scriviamo di fianco a questa matrice la matrice identica

$\left\| \begin{array}{ccc|ccc} 1 & 2 & 3 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 3 & 0 & 0 & 0 & 0 & 1 \end{array} \right\|$ ora operiamo sulle righe della prima matrice in modo da trasformarla in una matrice identica (con operazioni elementari come quelle per rendere triangolare la matrice) quindi riportiamo le stesse operazioni sulla seconda matrice

$\left\| \begin{array}{ccc|ccc} 1 & 2 & 3 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & -6 & -9 & -3 & 0 & 1 \end{array} \right\|$ moltiplicato 1° riga per - 3 quindi sommata alla 3° riga.

$\left\| \begin{array}{ccc|ccc} 1 & 2 & 3 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & -9 & -3 & 6 & 1 \end{array} \right\|$ moltiplicato 2° riga per 6 quindi sommata alla 3° riga.

$\left\| \begin{array}{ccc|ccc} 1 & 2 & 3 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & \frac{1}{3} & -\frac{2}{3} & -\frac{1}{9} \end{array} \right\|$ moltiplicato 3° riga per - $\frac{1}{9}$

$\left\| \begin{array}{ccc|ccc} 1 & 2 & 0 & 0 & 2 & \frac{1}{3} \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & \frac{1}{3} & -\frac{2}{3} & -\frac{1}{9} \end{array} \right\|$ moltiplicato 3° riga per - 3 quindi sommato alla 1° riga

$\left\| \begin{array}{ccc|ccc} 1 & 0 & 0 & 0 & 0 & \frac{1}{3} \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & \frac{1}{3} & -\frac{2}{3} & -\frac{1}{9} \end{array} \right\|$ moltiplicato 2° riga per - 2 quindi sommata alla 1° riga

la matrice inversa risulta quindi essere $\begin{vmatrix} 0 & 0 & \frac{1}{3} \\ 0 & 1 & 0 \\ \frac{1}{3} & -\frac{2}{3} & -\frac{1}{9} \end{vmatrix}$

Questo metodo di risoluzione forse risulta più semplice visto che è un metodo pressoché automatico di procedere, è facilmente implementabile sul calcolatore, ma necessita di alcuni accorgimenti per evitare problematiche di calcolo in ambiente puramente automatico.

5.2.0 . Risoluzione di sistemi lineari

Ora abbiamo tutte le basi per risolvere qualunque tipo di sistema lineare. Anzi di più, perché disponiamo di strumenti in grado di dirci se è risolubile o meno.

Prima quindi di procedere alla risoluzione brutale di un sistema lineare, conviene sempre chiederci se ammette o meno soluzione. In questo caso conviene ricordare un teorema fondamentale che afferma che: un sistema è risolubile se e soltanto se la matrice completa e incompleta del sistema hanno rango uguale.
Senza entrare nel merito di cosa sia un rango e di come si calcoli, vediamo subito come si procede.

$$\begin{cases} a_{11} * x_1 + a_{21} * x_2 + ... + a_{n1} * x_n + b_{11} + b_{21} + ... = 0 \\ ... \\ a_{1m} * x_1 + a_{2m} * x_2 + ... + a_{nm} * x_n + b_{1m} + b_{2m} + ... = 0 \end{cases}$$

$$\left.\begin{array}{l} -b_1 = b_{11} + b_{21} + ... \\ \vdots \\ -b_m = b_{1m} + b_{2m} + ... \end{array}\right\}$$ solo per comodità adotto il segno meno, ma nulla cambia se non lo si facesse.

$$\begin{cases} a_{11} * x_1 + a_{21} * x_2 + ... + a_{n1} * x_n = b_1 \\ ... \\ a_{1m} * x_1 + a_{2m} * x_2 + ... + a_{nm} * x_n = b_m \end{cases}$$

trasformiamo il sistema in matrice riga

$$\begin{vmatrix} a_{11} * x_1 + a_{21} * x_2 + ... + a_{n1} * x_n \\ \vdots \\ a_{1m} * x_1 + a_{2m} * x_2 + ... + a_{nm} * x_n \end{vmatrix} = \begin{vmatrix} b_1 \\ \vdots \\ b_m \end{vmatrix}$$

Per definizione di prodotto tra matrici risulta evidente che :

$$\begin{vmatrix} a_{11} & a_{21} & \cdots & a_{n1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1m} & a_{2m} & \cdots & a_{nm} \end{vmatrix} * \begin{vmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{vmatrix} = \begin{vmatrix} b_1 \\ \vdots \\ b_m \end{vmatrix}$$

Arrivati a questo punto siamo all'80% del lavoro. Ora non ci resta che imparare un metodo per valutare se il sistema è risolubile o meno, quindi procedere alla risoluzione del sistema stesso.
La matrice incompleta del sistema è la matrice degli *a*, mentre la matrice completa è la matrice degli *a* con aggiunto la colonna dei termini noti – ossia la matrice colonna dei *b*.
Dire che queste due matrici hanno uguale rango significa che, estraendo la matrice quadrata di rango diverso da zero con la massima dimensione dalla matrice incompleta, ed estraendo un'altra quadrata di rango diverso da zero di massima dimensione anche dalla completa, queste due matrici (che potrebbero anche essere le stesse) quadrate hanno la stessa dimensione.
Tutti questi giri di parole possono essere ricondotti a una operazione banale, che è la triangolarizzazione della matrice completa, come verrà mostrato di seguito:

$$\begin{vmatrix} a_{11} & a_{21} & \cdots & a_{n1} & b_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{1m} & a_{2m} & \cdots & a_{nm} & b_m \end{vmatrix} \xrightarrow{TRIANGOLARIZZAZIONE} \begin{vmatrix} a_{11} & a_{21} & \cdots & a_{n1} & b_1 \\ 0 & c_{22} & \cdots & c_{n1} & d_2 \\ 0 & 0 & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & c_{nm} & d_m \end{vmatrix}$$

Il nostro sistema sarà risolubile se ad ogni valore di b/d diverso da zero, la riga della matrice incompleta associata ha almeno un elemento diverso da zero.

5.3.0 . Metodi risolutivi e casi particolari

Da quanto detto prima saranno possibili tre casi:

- Sistema non risolubile
- Sistema risolubile avente 1 sola soluzione
- Sistema risolubile avente più soluzioni

5.3.1 . Sistema non risolubile

Non vale la pena di eseguire nulla. Si sa già che non avremmo nessuna soluzione valida al problema. Probabilmente le condizioni al contorno del problema sono imprecise o errate.

5.3.2 . Sistema risolubile avente 1 sola soluzione

In questa condizione ci troviamo quando abbiamo tante equazioni linearmente indipendenti quante sono le incognite. Ossia quando la matrice incompleta è una matrice quadrata di dimensione pari al numero di incognite e il determinante è diverso da zero.
Ci troviamo quindi in questa situazione:

$$\begin{vmatrix} a_{11} & a_{21} & \cdots & a_{n1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & \cdots & a_{nn} \end{vmatrix} * \begin{vmatrix} x_1 \\ \vdots \\ x_n \end{vmatrix} = \begin{vmatrix} b_1 \\ \vdots \\ b_n \end{vmatrix} \rightarrow \begin{vmatrix} x_1 \\ \vdots \\ x_n \end{vmatrix} = \begin{vmatrix} a_{11} & a_{21} & \cdots & a_{n1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & \cdots & a_{nn} \end{vmatrix}^{-1} * \begin{vmatrix} b_1 \\ \vdots \\ b_n \end{vmatrix}$$

Come si vede torna utile saper calcolare l'inversa in quanto è banale risolvere il sistema. Nei casi in cui i termini noti variano si è avvantaggiati perché, pre-calcolando l'inversa, abbiamo, in "tempo reale", tutte le soluzioni.

A volte, però, può non essere necessario avere tutte le soluzioni; in questo caso arriva in aiuto un altro procedimento che si basa sul calcolo dei determinati e che viene illustrato di seguito:

$$x_1 = \frac{\begin{Vmatrix} b_1 & a_{21} & \cdots & a_{n1} \\ \vdots & \vdots & \ddots & \vdots \\ b_n & a_{2n} & \cdots & a_{nn} \end{Vmatrix}}{\begin{Vmatrix} a_{11} & a_{21} & \cdots & a_{n1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & \cdots & a_{nn} \end{Vmatrix}}$$

$$x_2 = \frac{\begin{Vmatrix} a_{11} & b_1 & \cdots & a_{n1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1n} & b_n & \cdots & a_{nn} \end{Vmatrix}}{\begin{Vmatrix} a_{11} & a_{21} & \cdots & a_{n1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & \cdots & a_{nn} \end{Vmatrix}}$$

Oltre a questi metodi adottati, ne esiste un altro molto comodo che si basa proprio sulla triangolarizzazione della matrice completa, come l'esempio seguente mostrerà:

$$\begin{vmatrix} a_{11} & a_{21} & \cdots & a_{n1} & b_1 \\ 0 & c_{22} & \cdots & c_{n1} & d_2 \\ 0 & 0 & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & c_{nn} & d_n \end{vmatrix}$$ una volta triangolarizzata vediamo che il sistema è pressochè risolto

$$\begin{vmatrix} a_{11} & a_{21} & \cdots & a_{n1} \\ 0 & c_{22} & \cdots & c_{n1} \\ 0 & 0 & \ddots & \vdots \\ 0 & 0 & 0 & c_{nn} \end{vmatrix} * \begin{vmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{vmatrix} = \begin{vmatrix} b_1 \\ d_2 \\ \vdots \\ d_n \end{vmatrix}$$ risulta banale quindi vedere che

$$\begin{cases} a_{11} * x_1 + & a_{21} * x_2 & + \cdots + & a_{n1} * x_n = d_1 \\ 0 & c_{22} * x_2 & + \cdots + & c_{n1} * x_n = d_2 \\ 0 & 0 & \ddots & \vdots \\ & & & c_{nn} * x_n = d_n \end{cases}$$ da cui banalmente risulta

$x_n = \frac{d_n}{c_{nn}}$ poi si sotituirà questo valore alla riga precedente e si calcolerà x_{n-1},
si procederà così fino al termine.

Come accennato pochi paragrafi fa, la tecnica della triangolarizzazione è molto utile: ci consente con pochi calcoli di trovare se il sistema è risolubile, e quindi trovarne le soluzioni senza ulteriori complicazioni. Questa tecnica del resto è utile anche per l'ultima classe di sistemi, ossia quelli risolubili, ma aventi più soluzioni.

5.3.3 . Sistema risolubile avente più soluzioni

Le soluzioni sono uno spazio vettoriale (veramente non solo in questo caso ☺), cioè sono date da una o più matrice colonna (riga secondo "gusti") combinate tra loro. Anche qui esistono alcuni casi di interesse pratico. In particolare, dal punto di vista tecnico, quello che merita maggior considerazione è il caso in cui si abbia *n* incognite ed *n*-1 equazioni associate al vettore soluzione nullo. In questo preciso caso, abbiamo come soluzione una matrice colonna avente *n* elementi, dove ciascun elemento ha come valore il determinante della sottomatrice quadrata ricavata togliendo la colonna dell'elemento preso in considerazone. Un esempio:

$$\begin{cases} 1*x_1 + 2*x_2 + 3*x_3 = 0 \\ 0*x_1 + 1*x_2 + 0*x_3 = 0 \end{cases}$$

$$\begin{vmatrix} 1 & 2 & 3 \\ 0 & 1 & 0 \end{vmatrix} \rightarrow \begin{vmatrix} x_1 \\ x_2 \\ x_3 \end{vmatrix} = \begin{Vmatrix} 2 & 3 \\ 1 & 0 \end{Vmatrix} \begin{Vmatrix} 1 & 3 \\ 0 & 0 \end{Vmatrix} \begin{Vmatrix} 1 & 2 \\ 0 & 1 \end{Vmatrix} = \begin{vmatrix} -3 \\ 0 \\ 1 \end{vmatrix}$$

infatti abbiamo che la soluzione generica è $\lambda * \begin{vmatrix} -3 \\ 0 \\ 1 \end{vmatrix}$ con $\lambda \in \Re$.

$$\begin{vmatrix} 1 & 2 & 3 \\ 0 & 1 & 0 \end{vmatrix} * \lambda * \begin{vmatrix} -3 \\ 0 \\ 1 \end{vmatrix} = \begin{vmatrix} 0 \\ 0 \end{vmatrix}$$ qualunque sia λ.

In generale, per risolvere questo tipo di problematiche, conviene sempre usare il metodo di triangolarizzazione della matrice:

$\begin{cases} 1*x_1 + 2*x_2 + 3*x_3 = 1 \\ 0*x_1 + 1*x_2 + 0*x_3 = 2 \end{cases} \rightarrow \begin{vmatrix} 1 & 2 & 3 & 1 \\ 0 & 1 & 0 & 1 \end{vmatrix}$ è gia triangolare e si vede che ammette soluzione.

$x_2 = \frac{1}{1} = 1 \rightarrow 1*x_1 + 2*1 + 3*x_3 = 1 \rightarrow x_1 = 1 - 2 - 3*x_3 = -1 - 3*x_3$ la soluzione del nostro sistema è

$$\begin{vmatrix} -1-3*x_3 \\ 1 \\ x_3 \end{vmatrix} = \begin{vmatrix} -1 \\ 1 \\ 0 \end{vmatrix} + \left\langle \begin{vmatrix} -3 \\ 0 \\ 1 \end{vmatrix} \right\rangle$$

Purtroppo la notazione è un po' pesante e deriva dalla teoria degli spazi vettoriali, ma si può semplificare dicendo che la soluzione è *[-1,1,0]*, a cui si può sommare un qualunque vettore così costruito *l*[-3,0,1]*, da cui si vede come le soluzioni sono infinite e in funzione del parametro reale *l*.

5.4.0 . Problema dei minimi quadrati

Come ultimo accenno sulle potenzialità delle matrici nel mondo tecnico, vediamo come risolvere un problema che sicuramente ognuno di noi ha incontrato. In letteratura spesso lo si incontra con il termine minimi quadrati, ma lo possiamo così formulare:

- immaginiamo di aver fatto una serie di misure su di un sistema fisico per poi tracciarne l'andamento. Siamo sicuri che il sistema è lineare e ha una legge che rispecchia una retta.

Caso tipico potrebbe essere la misura di tensione su di una resistenza al variare della corrente, o la velocità in funzione della tensione su di un motore elettrico, …

In questi casi disponiamo di una tabella di punti (immaginiamo quindi che siano la tensione letta su una resistenza e la corrente che vi facciamo passare).

$p_n([i],[v])$
$p_1(0,0);\ p_2(1,1);\ p_3(2,2.4);\ p_4(3,3.6);$

Vogliamo quindi trovare la retta che meglio approssima la serie di punti rilevati.
La retta avrà equazione: $y=m*x+q$.
Ovviamente abbiamo un sistema sovradeterminato: troppe equazioni e, a causa di errori di misura, non avrà soluzione.

$$\begin{cases} 0*m+q=0 \\ 1*m+q=1 \\ 2*m+q=2.4 \\ 3*m+q=3.6 \end{cases}$$

Senza entrare sulle motivazioni tecniche e sul perché "funziona", si dovrà procedere in questo modo:

1. Calcolare la trasposta della matrice incompleta
2. Calcolare il prodotto tra la trasposta della matrice incompleta per la matrice incompleta stessa, ottenendo di regola una matrice quadrata.
3. Verificare che il determinante della matrice quadrata trovata sia diverso da zero, altrimenti il problema non è solubile.
4. Calcolare il prodotto tra matrice trasposta della incompleta per la matrice dei termini noti.
5. Si avrà così il nuovo sistema che consente di trovare i valori m,q che approssimano la retta dei valori trovati.

Di seguito l'esempio:

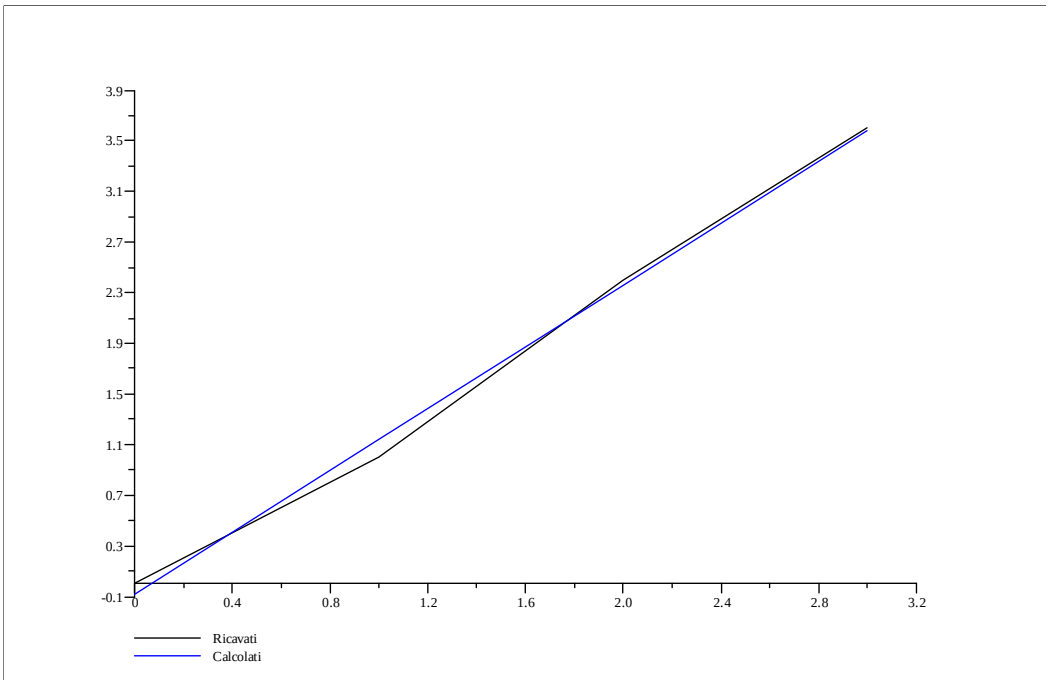
$$\begin{cases} 0*m+q=0 \\ 1*m+q=1 \\ 2*m+q=2.4 \\ 3*m+q=3.6 \end{cases}$$

$$\begin{vmatrix} 0 & 1 \\ 1 & 1 \\ 2 & 1 \\ 3 & 1 \end{vmatrix}_{-1} = \begin{vmatrix} 0 & 1 & 2 & 3 \\ 1 & 1 & 1 & 1 \end{vmatrix} \rightarrow M = \begin{vmatrix} 0 & 1 & 2 & 3 \\ 1 & 1 & 1 & 1 \end{vmatrix} * \begin{vmatrix} 0 & 1 \\ 1 & 1 \\ 2 & 1 \\ 3 & 1 \end{vmatrix} = \begin{vmatrix} 14 & 6 \\ 6 & 4 \end{vmatrix}$$

$$\begin{vmatrix} 0 & 1 & 2 & 3 \\ 1 & 1 & 1 & 1 \end{vmatrix} * \begin{vmatrix} 0 \\ 1 \\ 2.4 \\ 3.6 \end{vmatrix} = \begin{vmatrix} 16.6 \\ 7 \end{vmatrix}$$

$$\begin{vmatrix} 14 & 6 \\ 6 & 4 \end{vmatrix} * \begin{vmatrix} m \\ q \end{vmatrix} = \begin{vmatrix} 16.6 \\ 7 \end{vmatrix} \rightarrow \begin{vmatrix} m \\ q \end{vmatrix} = \begin{vmatrix} 14 & 6 \\ 6 & 4 \end{vmatrix}^{-1} * \begin{vmatrix} 16.6 \\ 7 \end{vmatrix} = \begin{vmatrix} 1.22 \\ -0.08 \end{vmatrix}$$

Di seguito la rappresentazione grafica:



La retta che meglio approssima i valori ricavati è: $v(i) = 1.22 * i - 0.08$.

6.0 . Funzioni

Spesso i problemi tecnici implicano la necessità di ricavarsi delle funzioni, oppure di semplificarle o di vedere dove siano i minimi/massimi di tali funzioni. Le funzioni sono quindi delle leggi che spesso sono indicate come $y = f(x)$.

Dal punto di vista tecnico - cioè quando si cerca una soluzione veloce (non vale la pena di scomodare tanta teoria) - serve solo avere una idea di come varia la legge. Per questo motivo conviene saper "giocare" con programmi adeguati. Il consiglio più utile da dare è procurarsi un programma o un calcolatore grafico e imparare ad usarlo.
Programmi di matematica ne esistono di molti tipi: gratuiti, commerciali, open, … e ognuno ha una sua particolarità. Possiamo però distinguerli in due categorie principali: quella puramente "numerica" e quella simbolica. Per la verità spesso si intersecano le due categorie. Tra le categorie numeriche troviamo i CAS (computer algebra system). Il concetto si basa su calcoli numerici eseguiti su matrici, vettori, … e sono i principali strumenti per un progettista o tecnico evoluto. Per usarli a pieno necessitano di una buona dose di teoria soprattutto per interpretare i risultati che, essendo calcoli eseguiti con precisione definita dall'utente, potrebbero presentare alcune spiacevoli sorprese. Nella categoria simbolica esistono strumenti più o meno validi, spesso usati nell'educazione. Hanno una grande potenzialità dal punto di vista teorico, semplificano il tracciamento di grafici ai meno esperti, ma per ottenere il massimo, richiedono spesso nozioni teoriche non alla portata di tutti.

Di seguito viene illustrato un procedimento alquanto comune e molto semplificato per eseguire lo studio di funzione tramite un Computer Algebra System.

6.1.0 . Grafico della funzione con un CAS

Prima di tutto abbiamo bisogno della funzione che, per semplicità, inventeremo al momento ☺:

$$f(t) = \frac{\sin(t)}{t}$$

Per graficarla abbiamo la necessità di discretizzare la variabile *t*, in quanto lavoriamo con un calcolatore. Quindi, dobbiamo costruirci un vettore contenente tutti gli istanti temporali di interesse. Visto che c'è una funzione trigonometrica di periodo $2*\pi$, il nostro dominio lo imponiamo vari tra $-2*\pi < t < 2*\pi$ (ovviamente è solo un esempio, dove non si pretende dare un metodo di analisi, ma quanto vedere le possibili strade).

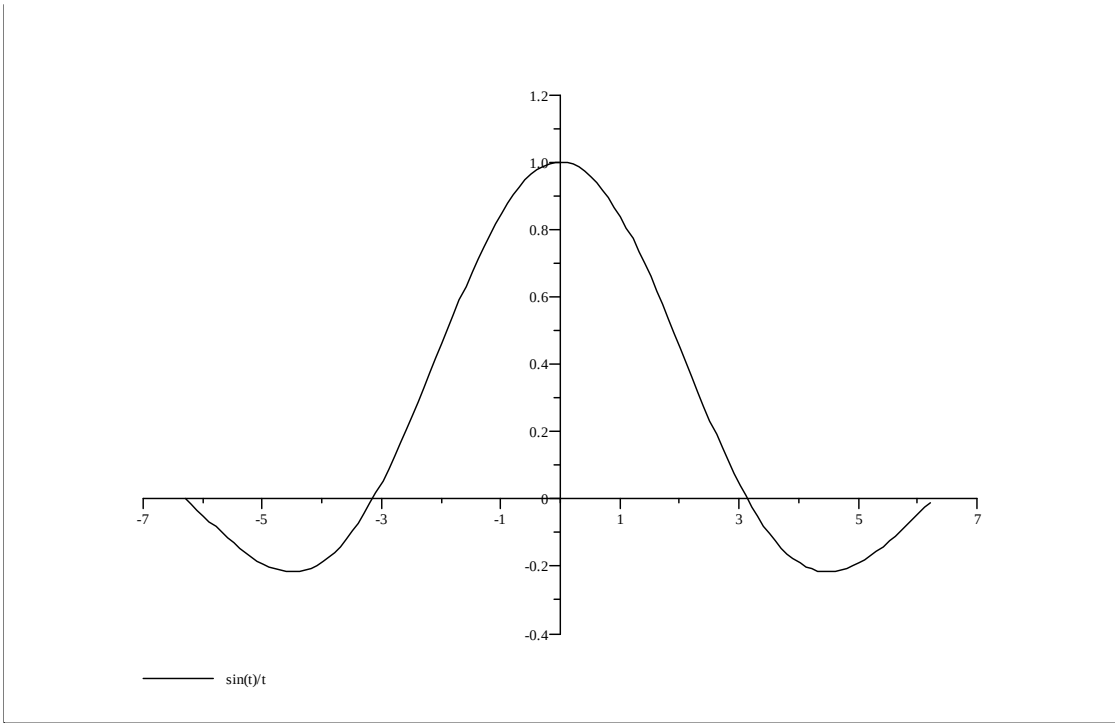
Useremo un meta linguaggio abbastanza comune su alcuni CAS (del resto con il manuale sarà abbastanza semplice fare le dovute correzioni).

$K\_p = 2*\pi;$
$t = [-K\_p : 0{,}1 : +K\_p]$ costruiamo il vettore tempi con passo di 0,1 ossia $[-2*\pi, -6.38.., -6.28.., ..., 2*\pi]$

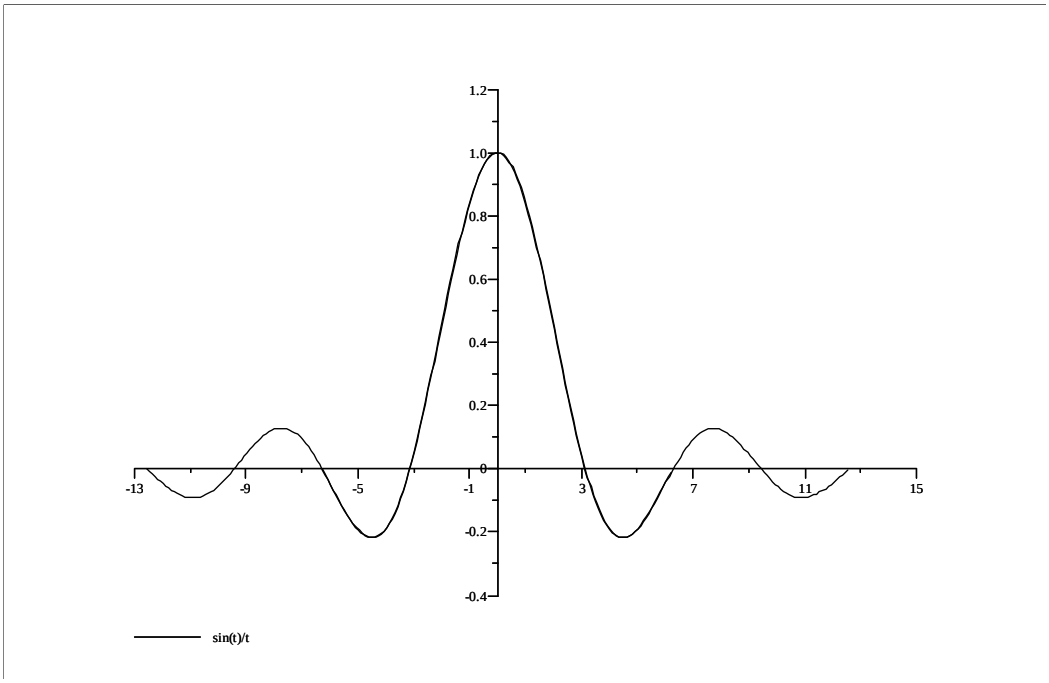
a questo punto possiamo costruirci il vettore contenente i risultati della funzione:

$f = \frac{\sin(t)}{t}$ f è quindi è un vettore di dimensione pari a t

A questo punto, avendo i due vettori, si utilizza una funzione (che molto spesso si chiama plot) per disegnare a video le coppie di punti rappresentanti la funzione; si otterrà come risultato:



A questo punto si vede che nel dominio di interesse esiste un massimo e 2 minimi. Se volessimo approfondire di più, potremmo incrementare il dominio:



Ora si possono trarre le dovute considerazioni sulla forma della nostra funzione. Per completezza, però, bisogna dire che il sistema di plottaggio che ha consentito di disegnare la funzione si comporta in modo anomalo, oppure ha un grado di intelligenza non disprezzabile.
Infatti la funzione studiata non dovrebbe essere definita nello zero dal momento che abbiamo una divisione per zero: $f = \frac{sin(t)}{t}$ per t = 0 .

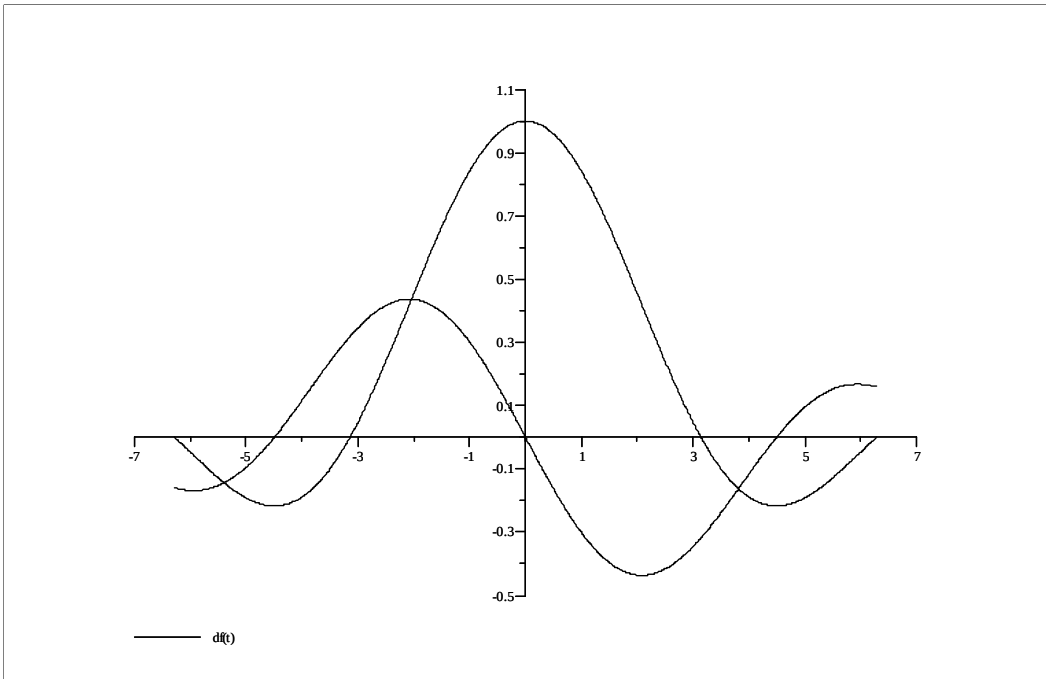
A questo punto dobbiamo essere noi utenti a stabilire se il grafico, nell'intorno dello zero, è una rappresentazione corretta o meno della realtà.

In aiuto viene la teoria dei limiti $\lim_{t \to 0} \frac{\sin(t)}{t} = 1$ e in questo caso è un limite ben conosciuto ☺. Quindi il grafico è corretto anche nell'intorno dello zero.
Questo è solo un esempio di ciò che si intende per nozioni teoriche di base per lo studio di funzioni con il calcolatore.

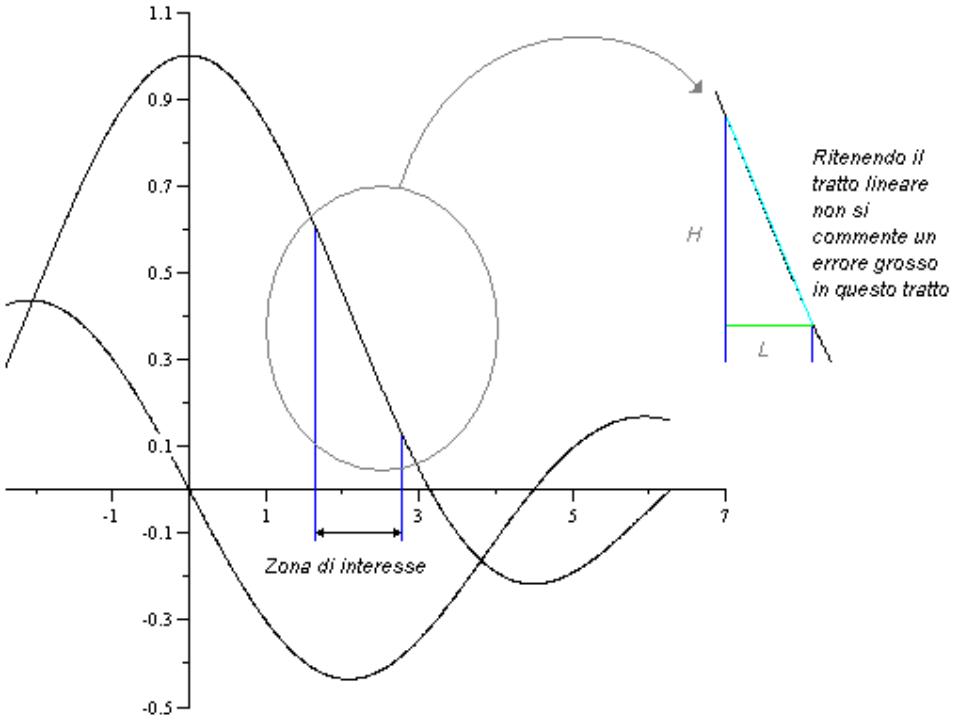
6.2.0 . Accenni sulle derivate e loro significato

Dal punto di vista progettuale le cose invece cambiano, e si è costretti ad usare metodi matematici anche sofisticati in quanto potrebbe essere importante conoscere le pendenze della nostra funzione. In questo caso entrano in gioco le nozioni di derivata; quindi avremmo che:

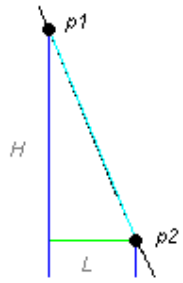
$$\frac{d}{dt}(f(t)) = \frac{d}{dt}(\frac{\sin(t)}{t}) = \frac{\frac{\mathrm{d}}{\mathrm{dt}}(\sin(t)) * t - \frac{d}{dt}(t) * \sin(t)}{t^2} = \frac{t * \cos(t) - \sin(t)}{t^2}$$



Come da teoria, dove la derivata assume valori positivi la funzione è crescente; dove assume valori negativi la funzione è decrescente; dove si annulla vi è la possibilità di trovare un punto di max/min relativo.
Il concetto di derivata è molto importante non solo per lo studio di funzioni, ma anche perché aiuta a semplificare le funzioni. Vediamo velocemente una applicazione:



Quello che si fa spesso è restringere il dominio di interesse al tratto lineare, in modo da semplificare la funzione. Quello che si deve fare, quindi, è trovare la pendenza della retta che interseca la curva.



$p_1 = [x_1, y_1]$
$p_2 = [x_2, y_2]$

La pendenza della retta la possiamo quindi trovare come $\frac{y_1 - y_2}{x_1 - x_2}$. Questo, quindi, è il coefficiente angolare della retta che passa per i punti *p1* e *p2* e che, per il nostro

scopo, approssima in modo buono il tratto di curva. Cosa centri la derivata in tutto questo è presto detto; infatti, portando il calcolo di prima al limite, otteniamo proprio il concetto di derivata:

$$\frac{y_1 - y_2}{x_1 - x_2} = \frac{f(x_1) - f(x_2)}{x_1 - x_2} = \frac{\Delta f}{\Delta x}$$

$$\lim_{\Delta x \to 0} \frac{\Delta f}{\Delta x} = \frac{d}{dx} f(x)$$

La derivata, quindi, rappresenta il coefficiente angolare del tratto infinitesimale di curva considerato. Quindi, per piccole variazioni di *x*, possiamo scrivere: $f(x_a + \Delta x) = f(x_a) + \Delta x * \frac{d}{dx} f(x_a)$. Ossia, una volta conosciuta *f(x)* in un punto, per piccole variazioni di x possiamo "linearizzare" il suo comportamento, quindi semplificare notevolmente lo studio o la stessa realizzazione.

6.3.0 . Accenni sullo sviluppo in serie di funzioni

L'ultimo campo che analizzeremo velocemente consiste nel vedere se si può semplificare le funzioni con altre più semplici.
Anche in questo campo bisognerebbe avere una base teorica alle spalle abbastanza pesante, ma vedremo che, per quello che serve ad un "tecnico evoluto", non è limitativo ☺.

Sicuramente le funzioni più "belle" (perché semplici da studiare) sono quelle nella forma: $f(x) = \sum_{i=0}^{M} a_i * x^i = a_0 + a_1 * x + a_2 * x^2 + ... + a_M * x^M$. Potremmo chiederci se fosse possibile approssimare almeno una parte di funzioni con un polinomio come quello scritto poc'anzi e, se fosse possibile, come procedere per ricavarsi i coefficienti.

La risposta è sì, ed il procedimento è molto semplice, anzi fin troppo intuitivo ☺:

$$f(x) \stackrel{?}{\cong} a_0 + a_1 * x + a_2 * x^2 + ... + a_M * x^M$$

se ammettiamo possibile quello che abbiamo scritto allora risulta che

$f(0) = a_0$ dovrebbe essere evidente

essendo a_0 una costante all'orapossiamo procedere come segue

$$\frac{\mathrm{d}}{\mathrm{dx}} f(x) = \frac{d}{dx} a_0 + \frac{d}{dx}(a_1 * x) + \frac{d}{dx}(a_2 * x^2) + ... = a_1 + \frac{1}{2} * a_2 * x + \frac{1}{3} * a_3 * x^2 + ...$$

$$\frac{\mathrm{d}}{\mathrm{dx}} f(0) = a_1$$

$$\frac{\mathrm{d}^2}{\mathrm{dx}^2} f(x) = \frac{d}{dx} a_1 + \frac{d}{dx}(\frac{1}{2} * a_2 * x) + \frac{d}{dx}(\frac{1}{3} * a_3 * x^2) + ... = \frac{1}{2} * a_2 + \frac{1}{6} * a_3 * x + ...$$

$$\frac{\mathrm{d}^2}{\mathrm{dx}^2} f(0) = \frac{1}{2} * a_2$$

e così via ...

Da quanto detto è importante notare che la funzione deve essere derivabile. Purtroppo non tutte le funzioni sono derivabili (e le loro derivate ancora derivabili) nell'intorno dello zero. Questo, però, non deve preoccupare perché è sempre possibile traslare il punto. L'esempio che si trova maggiormente in letteratura è il logaritmo:

$$f(x) = \ln(1+x) \stackrel{?}{\approx} \sum_{i=0}^{M} a_i * x^i$$

$$f(x)\big|_{x\to 0} = 0 = a_0$$

$$\frac{d}{dx} f(x) = \frac{1}{1+x}$$

$$\frac{d}{dx} f(x)\Big|_{x\to 0} = 1 = a_1$$

...

$$\ln(1+x) \approx x - \frac{1}{2} * x^2 + \frac{1}{3} * x^3 - \frac{1}{4} * x^4 + ...$$

Lo sviluppo in serie (questo è il nome della tecnica utilizzata in questo paragrafo, in particolare lo sviluppo in serie di Taylor) è molto redditizio in quanto semplifica notevolmente le funzioni, e una loro eventuale implementazione sul calcolatore. Basti notare che, con una semplice sommatoria, possiamo rappresentare funzioni molto complesse e soprattutto decidere il grado di precisione che vogliamo ottenere dal calcolo.

Per completezza bisogna aggiungere che, oltre allo sviluppo in serie di Taylor, esistono molte altre tecniche, tra le quali una molto utilizzata (non in forma di serie, ma di trasformata – ma onestamente il concetto è abbastanza simile -) è la serie di Fourier.
La particolarità della serie di Fourier consiste nel cercare se esiste una serie di funzioni trigonometriche (*sin* , *cos*) che approssima la funzione data.

Il calcolo della serie di Fourier è molto laborioso e complesso e si basa sull'ortogonalità delle funzioni *sin* e *cos* (solo come accenno, *sin* e *cos* formano uno spazio vettoriale ortogonale, quindi l'idea che sta alla base è quella di esprimere le funzioni tramite "vettori di funzioni *sin* , *cos*").
In questo prontuario non verrà trattata la serie di Fourier (forse in un'evoluzione futura !?!), vale però la pena di accennare che estendendo il concetto di serie di Fourier, si entra "nel campo" della trasformata di Fourier.
La trasformata di Fourier consente di semplificare notevolmente calcoli complessi, eseguire analisi armoniche, … (basti ricordare come le reti elettriche si possano semplificare se viste nel dominio della frequenza, quindi trasformate di Fourier, Cisoidali, …)
… questa però è tutta un'altra storia ☺.

7.0 . Integrazione

Pur essendo un campo che ha molti aspetti simili alle funzioni - l'integrazione viene fatta su funzioni – la vediamo in modo a se stante, soprattutto per quanto riguarda l'integrazione numerica.

Perché è importante? Behh … basti l'approccio a problemi di fisica "tecnica" tra cui il più banale è lo studio della dinamica di un punto (spazio, velocità, accelerazione).

a	accelerazione
v	velocità
t	tempo
s	spazio

$$s = \int_{t_0}^{t_1} v \cdot dt = v*t\Big|_{t_0}^{t_1} = v*(t_1 - t_0)$$

$$v = \int_{t_0}^{t_1} a \cdot dt = a*t\Big|_{t_0}^{t_1} = a*(t_1 - t_0)$$

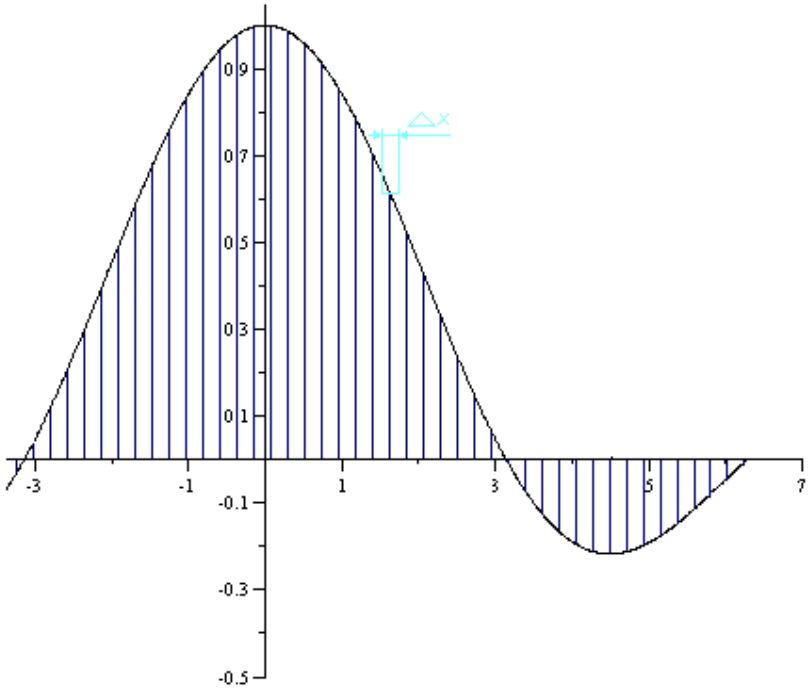
$$t_0 = 0; t_1 = \tau; v = a*\tau$$

$$s(t) = \int_0^t a*\tau \cdot d\tau = \frac{1}{2}*a*\tau^2\Big|_0^t = \frac{1}{2}*a*t^2$$

Come si vede, l'integrazione riveste un ruolo importante per problemi apparentemente semplici. Spesso e volentieri basta ricordarsi le formule; ma se poi dobbiamo simulare il tutto? Diciamo che all'80% delle volte basta il manuale e le formule pronte, ma per il restante dobbiamo per forza di cose arrangiarci con il nostro fido calcolatore e un po' di teoria ☺.

L'integrazione spesso – in modo barbaro – è confusa con il calcolo dell'area sottesa da una funzione. In questo caso, una definizione così barbara è quanto di meglio possiamo avere ☺.
In automazione spesso viene richiesto di calcolare l'energia, niente di più semplice: se conosciamo la potenza basta integrarla per il tempo ☺.
Risulta chiaro che è necessario avere un metodo semplice per integrare le grandezze. E a dire il vero il metodo lo abbiamo – ed è anche semplice –. Prendendo la

definizione di integrale che si trova in molti libri - "… area dei rettangoloidi associati …" (non vale la pena di ricordarsela tutta basta e avanza questa ☺) - abbiamo che la rappresentazione grafica è la seguente:



Numericamente, quindi, possiamo pensare di suddividere la nostra curva in tanti pezzettini, prendere il valore della nostra curva discretizzata, quindi sommare tutti i valori e moltiplicarli per il *delta x* scelto nella suddivisione. Tanto più piccolo è il *delta x* e tanto più preciso sarà il calcolo.

$$t_i = [-T : 0{,}01 : +2*T]$$
$$Area \approx \Delta x * \sum_i f(t_i)$$

I metodi di integrazione numerica non servono solo per calcolare l'area di funzione, ma (con opportuna interpretazione) anche per trovare la media:

$$Media = \frac{1}{M} \sum_{i=1}^{M} f_i$$

Dove f è una successione numerica, ad esempio una serie di rilevazioni di una grandezza per un arco temporale abbastanza grande. Tipico esempio è quello di monitorare la potenza media in utilizzo o l'assorbimento medio di corrente.

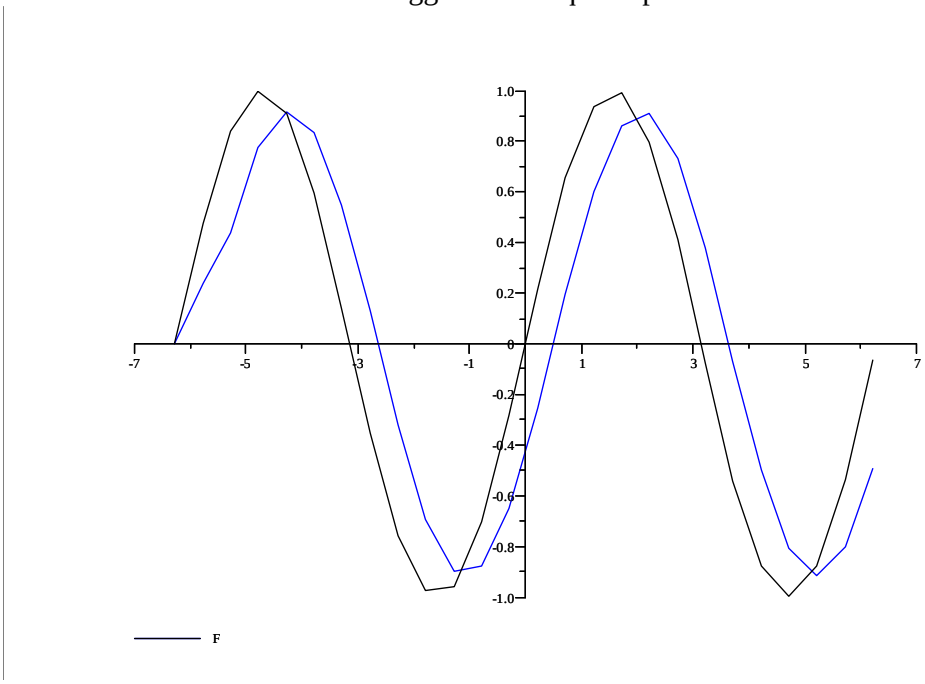
```
double Amp[]<-get_Misure();
double M_Amp = 0;

for(int i=0;i<Amp.lenght;i++)
{
        M_Amp += Amp[i];
}
return M_Amp/Amp.lenght;
```

Qualche volta potrebbe essere necessario utilizzare una media per filtrare i dati letti, magari affetti da disturbi. Questo, però, ritarderebbe parecchio i risultati, in quanto si deve attendere di aver effettuato tutte le letture prima di procedere a eseguire la media (a dimostrazione basti l'algoritmo poco sopra scritto). Nascono, per ovviare a queste problematiche, una serie di algoritmi che si basano sempre sul concetto di somma. Qui viene proposto solo a titolo di curiosità un possibile modo di procedere:

$$f_i\big|_{i=0} = f_0$$
$$f_i\big|_{i>0} = \frac{f_i + f_{i-1}}{2}$$

Di seguito viene mostrato cosa potrebbe accadere utilizzando una tecnica come quella accennata, dove l'andamento nero è la forma originale, e l'andamento azzurro è l'andamento modificato da una legge simile a quella prima scritta.



Questo capitolo ha voluto solo dare alcuni accenni su come utilizzare tecniche semplici di integrazione, ovviamente deve essere chiaro che quanto visto è solo un accenno, giusto per incuriosire.