Metodi numerici e simbolici per il calcolo automatico

Si tratta di un capitolo della matematica generalmente assente nei corsi di formazione tradizionale, eppure di grande importanza applicativa, sia nell'ambito della ricerca che in quello industriale.

1 - Algoritmi di calcolo numerico per la derivazione, l'integrazione, la soluzione delle equazioni (funzioni reali di una variabile). Metodi euristici. Una piccola parentesi storica.

2 - Presentazione di un semplice pacchetto software per la manipolazione simbolica delle espressioni matematiche.

3 – Modelli dinamici differenziali e loro soluzione numerica. Cenni sul FEM.

Modulo 1

Il calcolo numerico: introduzione

Il calcolo numerico propone metodi alternativi a quelli della matematica classica, che si differenziano nei diversi ambiti della **matematica simbolica** per la manipolazione formale non delle variabili, delle funzioni e degli operatori, ma direttamente delle diverse quantità nella loro forma numerica. Sono stati così sviluppati metodi particolari negli ambiti dell'**analisi numerica**, della **geometria** e dell'**algebra** (alg. e geom. computazionali).

Per esempio si consideri l'equazione

$$a\cdot\cos(b\cdot x+c) = \sin(b\cdot x+c)$$

La soluzione di tale equazione, ovvero la determinazione dei valori numerici che assegnati alla variabile x soddisfano l'eguaglianza, può essere determinata con il metodo classico (simbolico) manipolando gli operatori, le funzioni e le variabili coinvolte, congruamente con le loro proprietà. Così in base alla definizione di funzione inversa, di funzione trigonometrica, ed in base alle proprietà delle operazioni di somma e prodotto, è possibile ricavare facilmente il valore:

$$x = \frac{Arctg(a) - c}{b}$$

Tale espressione individua tutte le soluzioni esatte dell'equazione in funzione dei parametri (*precisione infinita*).

Assegnando invece un valore ai parametri, p.e. a=7, b=3, c=5, si ottiene evidentemente:

$$x = \frac{Arctg(7) - 5}{3} \approx \frac{1.428899272 - 5}{3} = \frac{-3.571100728}{3} \approx -1.190366909$$

Si nota come la traduzione in forma **numerica** dell'espressione simbolica comporti la perdita dell'esattezza della soluzione. La divisione e la valutazione di un numero irrazionale (*Arctg(7)*) sono esprimibili compiutamente solo con un numero infinito di cifre, la quale cosa evidentemente non è ammessa dall'aritmetica di un microprocessore (*precisione finita*).

Se non ci interessasse conoscere la prima forma di soluzione, quella in funzione dei parametri *a b c*, ma solo la soluzione dell'equazione contenente parametri numerici potremmo seguire un metodo alternativo: si potrebbe stabilire con un qualche criterio che la soluzione del problema è un valore compreso in un certo intervallo. In particolare nel caso d'esempio, sapendo che si tratta di un angolo, potremmo limitare l'argomento delle funzioni trigonometriche all'intervallo [0, 2π]. In alternativa potremmo ipotizzare che il valore di x sia compreso tra 0 e 10^9. Si possono allora provare allora tutti i valori compresi in questo intervallo, per esempio con incrementi di 1 millesimo, cioè si prova a soddisfare l'equazione data con tutti i numeri 0, $1 \cdot 10^{-3}$, $2 \cdot 10^{-3}$,....., 10^9. Verosimilmente nessuno di questi numeri corrisponderà alla soluzione esatta, tuttavia se ne troverà uno o più d'uno che soddisfano l'equazione meglio di altri.

Potremmo allora ripetere il metodo in un intervallo più piccolo, vicino al campo della probabile soluzione, con un incremento più piccolo di un millesimo, p.e. un centomillesimo. Determineremo così una migliore approssimazione della soluzione dell'equazione.
Accettando tempi di calcolo più lunghi, possiamo migliorare arbitrariamente la precisione della soluzione.

Notiamo tre caratteri del metodo numerico, che ritroviamo nella generalità degli algoritmi:

- la soluzione numerica è una stima di quella vera;
- il metodo è iterativo, ed all'aumentare del numero di iterazioni dell'algoritmo (e quindi del tempo di calcolo), aumenta la precisione della soluzione calcolata;
- il metodo numerico è di applicabilità generale. P.e. basta modificare poco l'equazione di partenza per fare in modo che diventi non risolvibile con metodo analitico, p.e. $a \cdot \cos(b \cdot x + c) = \sin(b \cdot x + a)$. Il metodo numerico non risente invece di questa modifica.

I metodi simbolici esatti, pur se applicabili ai problemi contenuti negli eserciziari di matematica generale, sono spesso impraticabili per molti problemi pratici.

Gli algoritmi approssimanti, cioè in grado solo di fornire soluzioni approssimate, si utilizzano quando non esistono algoritmi esatti, oppure quando gli algoritmi esatti non risultano efficienti (tempo, memoria). P.e. radici del polinomio di grado superiore a 3.

1. L'APPROSSIMAZIONE NUMERICA

Abbiamo visto dunque che in genere la soluzione fornita da un metodo numerico è *sbagliata*, anche se si può spingere a piacere vicino a quella esatta. Uno dei compiti importanti dell'analisi numerica è proprio quello di fornire una stima dell'errore commesso.

In genere i metodi numerici, con il procedere delle iterazioni, convergono monotonicamente al valore esatto della soluzione (convergenza dal basso o dall'alto). Dunque forniscono una stima attraverso una serie di maggioranti o di minoranti. Spesso, per uno stesso tipo di problema si possono applicare metodi che permettono di maggiorare o di minorare la soluzione vera, con approssimazione prefissata.

Oltre al problema di **approssimazione del metodo**, sussiste (come abbiamo visto nel nostro esempietto numerico) il problema dell'approssimazione nella **rappresentazione dei numeri**. I calcolatori elettronici digitali dispongono di una rappresentazione interna dei numeri che, come abbiamo già visto, non è in grado di rappresentare un numero con una quantità di cifre significative generalmente superiore ad un certo valore (in QBasic 16). Così potremo avere solo un valore approssimato di numeri come π $\sqrt{7}$ $1/3$ che hanno uno sviluppo decimale infinito.

Sussiste lo stesso problema nella **rappresentazione delle operazioni**: anche partendo da **operandi** esatti non otterremo sempre risultati esatti. (p.e. 1/3). Le espressioni che sono equivalenti in aritmetica esatta non risultano generalmente tali con precisione finita, p.e. *arcsin(sin(x))*, 3·1/3, $(2^{1/2})^2$.

Particolarmente grave è il caso della **cancellazione numerica**, che si verifica nel caso di sottrazione tra numeri molto vicini, così vicini che la propria differenza sia dell'ordine di grandezza della risoluzione di macchina (in QBasic 10^{-38}).

Per evidenziare questo fatto consideriamo la funzione

$$f(x) = 2x\left(\sqrt{x^2+1} - x\right)$$

per la quale non è difficile verificare che:

$$Lim_{x\to\infty} f(x) = 1$$

eppure, sostituendo alla variabile x valori sempre più grandi, i valori che si ottengono per la *f(x)* non si avvicinano affatto ad 1:

x	f(x)
100	≈0.999974
10000	≈1.068116
100000	≈9.876658

La non attendibilità di questi risultati è dovuta alla presenza della sottrazione nell'espressione di *f(x)*. Se la *f(x)* viene scritta nella forma equivalente (nella quale non compare più la sottrazione):

$$f(x) = \frac{2x}{\sqrt{x^2+1} + x}$$

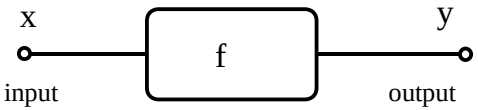
sostituendo alla variabile x i valori precedenti, si ottengono valori effettivamente sempre più prossimi ad 1:

x	f(x)
100	≈0.999975
10000	≈0.999978
100000	≈1.000000

2. LA PROPAGAZIONE DEGLI ERRORI

Condizionamento e stabilità del metodo

Esempio del calcolo dell'intersezione tra rette quasi parallele.
Consideriamo un processo di calcolo con un modello funzionale (*black box model*):



Se l'insieme x viene fornito a meno di una quantità Δx, ed il calcolo fornisce una soluzione y a meno di un margine Δy, si chiama numero di condizione (K) il rapporto tra gli errori relativi dei dati in input e di quelli in output:

$$K = \frac{\Delta y / y}{\Delta x / x}$$

il problema si dice mal condizionato se K è superiore ad uno.
Se cioè sono presenti errori nei dati in ingresso, e/o il processo f introduce degli errori, e tali errori aumentano durante il calcolo, si dice che il metodo è **numericamente instabile**.

Per esempio consideriamo il caso del calcolo dell'area di un quadrato:

$$A = l^2$$

se la misura l è affetta da qualche errore (p.e. dovuta ad un incertezza nella misurazione) allora sarà nota a meno di un margine di errore Δl:

$$l = lvera \pm \Delta l$$

allora l'espressione dell'area diviene:

$$A = l^2 = \left(l_{vera} \pm \Delta l\right)^2 = l_{vera}^2 \pm 2 l_{vera} \cdot \Delta l + \Delta l^2$$

trascurando Δl^2 rispetto a $2 l_{vera} \cdot \Delta l$ otteniamo la forma semplificata:

$$A \approx l_{vera}^2 \pm 2 l_{vera} \cdot \Delta l = A_{vera} \pm \Delta A$$

dall'espressione:

$$\frac{\Delta A}{A} = K \frac{\Delta l}{l}$$

calcoliamo il valore di K:

$$K = \frac{\Delta A}{A_{vera}} \frac{l_{vera}}{\Delta l} = \frac{2l \cdot \Delta l}{l^2} \frac{l}{\Delta l} = 2$$

così un errore del 5% sulla misura di l, si tradurrà, almeno, in un errore del 10% sulla determinazione di A: il problema non è intrinsecamente ben condizionato. Il problema opposto, cioè quello di valutare il lato del quadrato nota la sua area risulta invece molto ben condizionato.

3. CALCOLO DEL LIMITE DI UNA FUNZIONE

Proviamo a calcolare i valori assunti da una funzione nelle vicinanze di un punto di discontinuità per cercare di capire se, avvicinandosi a tale punto, il valore della funzione si avvicina ad un valore limite, diverge, o tende ad infinito.
Consideriamo come esempio il seguente limite:

$$Lim_{x\to 0}\frac{\sin(x)}{x}$$

Un primo programma possibile potrebbe partire da un valore *x=1* dell'ascissa ed avvicinarsi al valore zero, p.e. per dimezzamenti successivi:

```
'       Limiti indeterminati

x = 1
FOR i = 1 TO 30    ' 30 dimezzamenti successivi
   y = SIN(x) / x
   PRINT x, y
   x = x / 2
NEXT i
```

x	Lim
1	0.8414710
0.5	0.9588511
0.25	0.9896159
0.125	0.9973978
0.0625	0.9993491
0.03125	0.9998372
0.015625	0.9999593
0.0078125	0.9999898
3.90625E-03	0.9999974
1.953125E-03	0.9999993

4. CALCOLO DELLA DERIVATA DI UNA FUNZIONE IN UN PUNTO

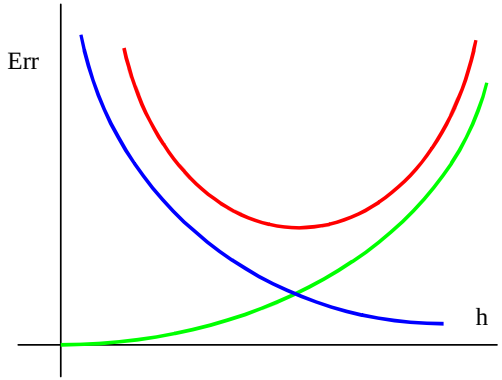
Il precedente processo di calcolo per determinare il limite di una funzione può essere applicato quando si voglia determinare il valore della derivata di una funzione in un punto. Basta infatti rifarsi alla definizione di derivata come limite del rapporto incrementale:

$$\frac{df(x)}{dx} = \underset{h\to 0}{Lim}\frac{f(x+h)-f(x)}{h}$$

```
'derivata numerica

x = 7
       ‘ si calcola la derivata della funzione seno
       ‘ nel punto x=7

h = 1
     ‘ con incremento finito h variabile 0>h≥1

CLS
DO WHILE (h > 1E-09)
     Errore = (COS(x)-DF(x, h)) / COS(x)
     ‘Errore relativo:
     ‘(valore vero-calcolato)/vero

        PRINT h, DF(x, h), Errore
        h = h / 3
LOOP
END


FUNCTION DF (x, h)
        DF = (F(x + h) - F(x)) / h
END FUNCTION

FUNCTION F (x)
        F = SIN(x)
END FUNCTION
```


h	Rapporto incrementale	Errore relativo
1	.3323717	.5591316
.3333333	.6315311	.162317
.1111111	.7158907	5.041973E-02
3.703704E-02	.7415627	1.636763E-02
1.234568E-02	.749833	5.397605E-03
4.115226E-03	.752527	1.824183E-03
1.371742E-03	.7535409	4.79347E-04
4.572474E-04	.753845	7.58962E-05
1.524158E-04	.7547575	-1.134456E-03
5.080526E-05	.7567129	-3.728069E-03
1.693509E-05	.763752	-1.306507E-02
5.645029E-06	.7602324	-8.396572E-03
1.881676E-06	.7602324	-8.396572E-03
6.272255E-07	.5701743	.2437026
2.090752E-07	0	1
6.969172E-08	0	1
2.323057E-08	0	1
7.743524E-09	0	1
2.581175E-09	0	1

Per valori di h troppo alti, evidentemente ci si allontana dalla condizione $h \to 0$ (errore di discretizzazione). Ma si nota anche che per valori di h molto piccoli i risultati siano affetti da un errore notevolissimo (errore nella rappresentazione dei numeri).
L'errore di condizionamento cala per valori di h più piccoli, mentre crescono parallelamente i problemi di stabilità dovuti alla cancellazione numerica.



- Si nota come esista un valore di *h*, dell'ordine di $1/10^4$, che rende minimo lo scarto tra la soluzione analitica e quella numerica.
- Utilizzando variabili in doppia precisione, il valore ottimo di *h* diviene dell'ordine di $1/10^8$.

Esistono molti metodi alternativi a quello esposto per la valutazione numerica della derivata, come p.e. quelli che prevedono l'approssimazione polinomiale continua a tratti della funzione, e la valutazione analitica della derivata del polinomio approssimante. In questa trattazione è stato scelto di presentare ciascun problema con una prima traccia di soluzione, così come gli esempi di implementazione dei diversi algoritmi in linguaggio *QBasic* risultano certamente suscettibili di molti miglioramenti.

5. DETERMINAZIONE DELLE SOLUZIONI DI UN'EQUAZIONE

Il problema della soluzione di una generica equazione del tipo *f1(x)=f2(x)* può essere sempre ricondotto a quello della determinazione delle soluzioni di un'equazione del tipo *f(x) = 0*. Infatti ***f(x) = f1(x) - f2(x) = 0***.
In questa forma le soluzioni si chiamano anche radici o zeri di *f(x)*.

In generale i metodi di calcolo delle radici di un'equazione richiedono che si sia preliminarmente riusciti ad isolare l'intervallo [*a, b*] entro il quale cade almeno una radice dell'equazione data.

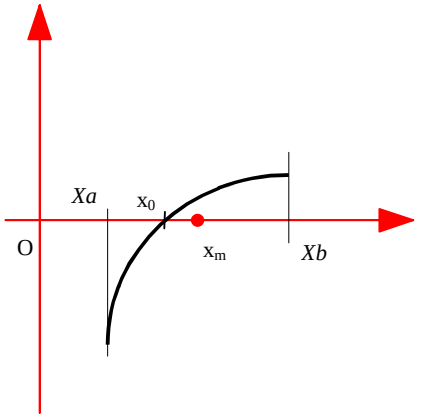
Per determinare tale intervallo si può ricorrere:

- ad una stima preliminare (esplorazione euristica);
- a proprietà analitiche della funzione (p.e. funzioni periodiche);
- a proprietà fisiche del modello che la funzione esprime (p.e. valori di PH).

Occorre notare come già l'avere individuato un intervallo che delimita una radice, ne costituisce una prima approssimazione. Il calcolo procede poi migliorando l'approssimazione del valore della radice (iterazione) fino ad ottenere la precisione desiderata.

Metodo di bisezione

Sia *f(x)* una funzione definita e continua nell'intervallo [*Xa,Xb*], e sussista la disequazione $f(Xa)\cdot f(Xb)<0$, che costituisce una condizione sufficiente per l'esistenza di almeno una radice su [*Xa, Xb*].



Localizziamo il punto medio dell'intervallo [*Xa,Xb*]: *Xm=(Xa+Xb)/2*. Tale punto divide l'intervallo [*Xa,Xb*] in due parti uguali. Calcoliamo il valore della funzione nel punto medio *Xm*:

$$Ym = f(Xm) = f\left(\frac{Xa + Xb}{2}\right)$$

Se *Ym=0* (o anche solo "sufficientemente" nullo), allora *Xm* è il valore della radice cercata.

Se invece *Ym* non è nullo consideriamo quello dei due intervalli, [*Xa, Xm*] o [*Xm, Xb*], alle estremità del quale la funzione *f(x)* assume valori di segno opposto, ovvero quello che *isola* almeno una radice, costituendone una migliore approssimazione.

Il nuovo intervallo (che denominiamo [*Xa1, Xb1*]) viene poi suddiviso in due parti uguali, per le quali si ripete il ragionamento esposto per i due intervalli precedenti.

Procedendo con le iterazioni si perviene o alla radice esatta dell'equazione di partenza, o ad una sua approssimazione per mezzo di una serie di intervalli [*Xa,Xb*], [*Xa1,Xb1*], [*Xa2, Xb2*],.., [*Xai, Xbi*] ,..,[*Xan, Xbn*] contenuti ciascuno nel precedente, tali che per ciascuno risulta:

$$f(Xa_i)\cdot f(Xb_i) < 0 \quad e \quad Xb_i - Xa_i = \frac{Xb - Xa}{2^i}$$

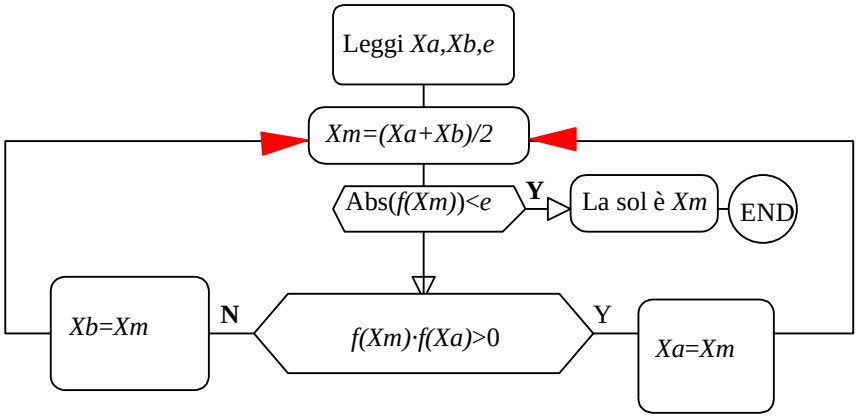
https://www.google.com/search?hl=en&source=hp&q=applet+bisection+method&btnG=Google+Search&aq=f&aqi=&aql=&oq=&gs_rfai=
http://www.scottsarra.org/math/courses/na/nc/bisection.html
http://www.cse.illinois.edu/iem/nonlinear_eqns/Bisection/
https://www.youtube.com/watch?v=DGmNbs5Cywo
http://www.cs.utah.edu/~zachary/isp/applets/Root/Bisection.html
http://www.theorphys.science.ru.nl/people/fasolino/Root/Root.shtml
http://rgm2.lab.nig.ac.jp/RGM2/func.php?rd_id=animation:bisection.method

L'algoritmo, espresso in un linguaggio di progetto (metalinguaggio), diviene:

```
1.leggi i numeri Xa, Xb, e;
2.calcola Xm =(Xa+Xb)/2;
3.calcola Ym =f(Xm);
4.   se (Abs(Ym) < e) allora
5.            la soluzione è Xm;
6.            END.
7.   altrimenti
8.            se (f(Xa)*f(Xm) > 0) allora
9.                 Xa = Xm
10.           altrimenti
11.                Xb = Xm
12.torna al passo 2.
```

al passo 3 si controlla l'esistenza della soluzione. Ciò può essere fatto anche con un controllo del tipo se *ABS(Xm-Xb)<e* allora termina.

Il diagramma di flusso diviene dunque:




```
'Metdodo di bisezione (soluzione iterativa)

xa = -1
xb = 17
e = .00001

CLS

DO
        xm = (xa + xb) / 2
        PRINT xm, F(xm)

        IF (F(xm)*F(xa))>0 THEN
                xa = xm
        ELSE
                xb = xm
        END IF

LOOP WHILE (ABS(xa - xb) > e)
END

FUNCTION F (x)
        F = (x * x - 9)
END FUNCTION
```


Xm	*f(Xm)*
8	55
3.5	3.25
1.25	-7.4375
2.375	-3.359375
2.9375	-.3710938
3.21875	1.360352
3.078125	.4748535
3.007813	4.693604E-02
2.972656	-.1633148
2.990234	-5.849838E-02
2.999023	-5.858421E-03
3.003418	.0205195
3.001221	7.325709E-03
3.000122	7.324368E-04
2.999573	-2.563294E-03
2.999847	-9.155041E-04
2.999985	-9.15525E-05
3.000053	3.204374E-04
3.000019	1.144413E-04
3.000002	1.14441E-05
2.999993	-4.005428E-05

Oltre a quella iterativa è anche possibile un'elegante traduzione dell'algoritmo con tecnica ricorsiva:

```
'Metdodo di bisezione (soluzione ricorsiva)

xa = -1
xb = 17
epsi = .0001

CLS

x0 = Radice(xa, xb, epsi)
PRINT "radice="; x0
END


FUNCTION F (x)
        F = (x * x - 9)
END FUNCTION


FUNCTION Radice (a, b, e)
        xm = (a + b) / 2
        PRINT xm, F(xm)
        IF (ABS(F(xm)) < e) THEN
                Radice = xm
        ELSE
                IF SGN(F(xm)) = SGN(F(a)) THEN
                        a = xm
                ELSE
                        b = xm
                END IF

                Radice = Radice(a, b, e)
        END IF

END FUNCTION
```


x	F(x)
1.25	-7.4375
2.375	-3.359375
2.9375	-.3710938
3.21875	1.360352
3.078125	.4748535
3.007813	4.693604E-02
2.972656	-.1633148
2.990234	-5.849838E-02
2.999023	-5.858421E-03
3.003418	.0205195
3.001221	7.325709E-03
3.000122	7.324368E-04
2.999573	-2.563294E-03
2.999847	-9.155041E-04
2.999985	-9.15525E-05

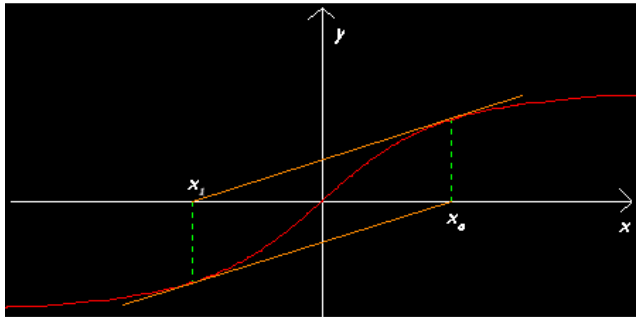
Resta da risolvere il problema dell'isolamento della radice in via generale: se una funzione continua *f(x)* assume in due punti *a* e *b* valori di segno opposto, esiste almeno un punto compreso tra *a* e *b* in cui *f(x)=0*.
Allora si campiona la funzione *f(x)* da *a* a *b* con passo *h*; se per due valori *x* ed *x+h*, la funzione *f(x)* cambia segno allora l'equazione *f(x)=0* ha un numero dispari di radici reali (dunque almeno una) in tale intervallo.

Metodo di Newton

Questo metodo è più efficiente e non richiede la stima dell'intervallo che contiene la radice ma solo una stima del valore della radice stessa. Se tale stima vale x_0, allora si sostituisce la funzione *f(x)* con la retta tangente nel punto $(x_0, f(x_0))$. Si considera poi l'intersezione x_1 della tangente con l'asse delle ascisse. Iterando il procedimento si costruisce una successione x_k, che converge al valore della radice cercata.

https://www.google.com/search?hl=en&source=hp&q=applet+newton+method&btnG=Google+Search&aq=f&aqi=&aql=&oq=&gs_rfai=
http://www.math.umn.edu/~garrett/qy/Newton.html
http://www-math.mit.edu/18.013A/HTML/chapter13/section01.html
http://www.math.gatech.edu/~carlen/applets/archived/ClassFiles/NewtonRaphson.html
http://www.calvin.edu/~rpruim/java/jcm/newton.shtml

Il metodo non converge se l'equazione non ha radici (p.e. $f(x)=x^2+1$) o se si verificano particolari condizioni nel cambio di concavità della curva.



Con il metodo di Newton non serve più isolare la radice, ma basta esplorare il dominio successivamente con un passo sempre più piccolo, usando diversi punti come innesco del metodo.

L'equazione della retta generica passante per P(x_0, y_0) è:

$$y - f(x_0) = m\cdot(x-x_0)$$

Utilizzando come coefficiente angolare m il valore della derivata di $f(x)$ in x_0 otteniamo l'equazione della retta tangente alla $f(x)$ in x_0:

$$y - f(x_0) = f'(x_0)\cdot(x-x_0)$$

tale retta incontra l'asse di equazione $y=0$ (l'asse x) nel punto di ascissa x_1

$0 - f(x_0) = f'(x_0)\cdot(x_1-x_0)$ da cui, ricavando x_1:

$$x_1 = x_0 - f(x_0)/f'(x_0)$$

che fornisce uno schema ricorrente per migliorare la precisione della stima della radice, a partire da x_0. Ripetendo il procedimento si determina la radice cercata con approssimazione sempre migliore, come limite della successione:

$$x_{i+1} = x_i - f(x_i)/f'(x_i)$$

Es. risolvere le equazioni seguenti:

- $x^2 - 9 = 0$ ==> $x_{i+1} = x_i - \frac{x_i^2-9}{2x_i}$
- $sin(x) - x = 0$ ==> $x_{i+1} = x_i - \frac{\sin x_i - x_i}{\cos x_i - 1}$
- $exp(x) - 1 = 3 + 0.1x^{0.2}$ ==> $x_{i+1} = x_i - \frac{e^{x_i} - 4 - 0.1x_i^{0.2}}{e^{x_i} - 0.02x_i^{-0.8}}$

```
'Soluzione di equazioni con metdodo di Newton

DECLARE FUNCTION F! (x!)
DECLARE FUNCTION Fd! (x!)

x0 = 99
e = .001

CLS
x = x0

DO
        PRINT x, F(x)
        x = x - F(x) / Fd(x)

LOOP WHILE (ABS(F(x)) > e)
END


FUNCTION F (x)
        F = (x * x - 9)
END FUNCTION

FUNCTION Fd (x)
        h = .0001
        Fd = (F(x + h) - F(x)) / h
END FUNCTION
```

```
99            9792
48.86496      2378.784
24.50621      591.5543
12.39118      144.5413
6.579731      34.29287
3.981561      6.852829
3.119965      .7341821
3.002195      1.317697E-02
```

Idee da sviluppare:

☞ Cercare i punti estremanti di una funzione attraverso la ricerca degli zeri della derivata prima.

☞ Cercare i punti di flesso di una funzione attraverso la ricerca degli zeri della derivata seconda.

☞ Vedi "Calculus" di H. Anton a pagina 413: per quale valore del parametro *b* il razzo è visto sotto l'angolo di 0.1 gradi ?

☞ Qual'è il punto della traiettoria del proiettile nel quale la velocità (o l'accelerazione) è massima?

☞ Nota la velocità e la posizione iniziale, determinare l'angolo di lancio di un proiettile affinché passi per un punto assegnato.

☞ Determinare la velocità iniziale o l'angolo di lancio di un proiettile affinchè colpisca un bersaglio in movimento (p.e. carrellino orizzontale, carrello verticale in moto uniforme o in caduta, un altro proiettile).

+ Una lattina per bevande gassate ha la forma di un cilindro con la parte inferiore sostituita da una cavità semisferica. Posto che l'altezza *h* deve essere compatibile con l'attuale impianto di confezionamento (*h*=10cm) e che il volume *V* deve essere pari a 330cm^3, determinare il raggio *r*, tenendo conto della manipolabilità.

$$V = \pi r^2 h - 2/3\pi r^3$$

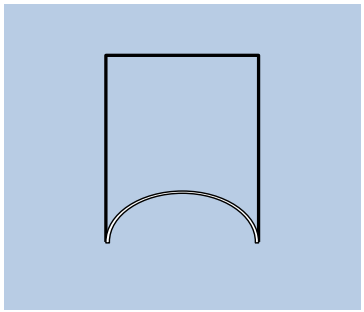
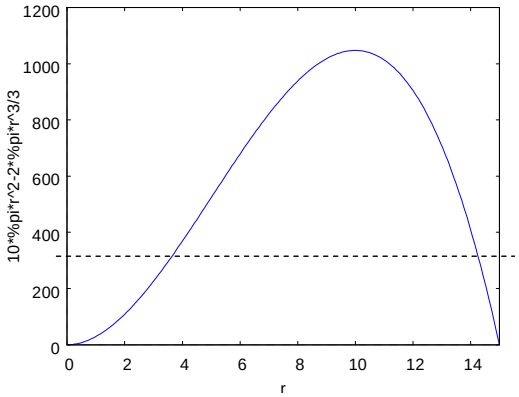


Grafico di ***V(r)*** <u>con *h*=10 cm</u> ed *r* compreso tra 0 e 15cm.



Il problema può essere facilmente risolto inserendo nel programma relativo al metodo di bisezione la funzione seguente:

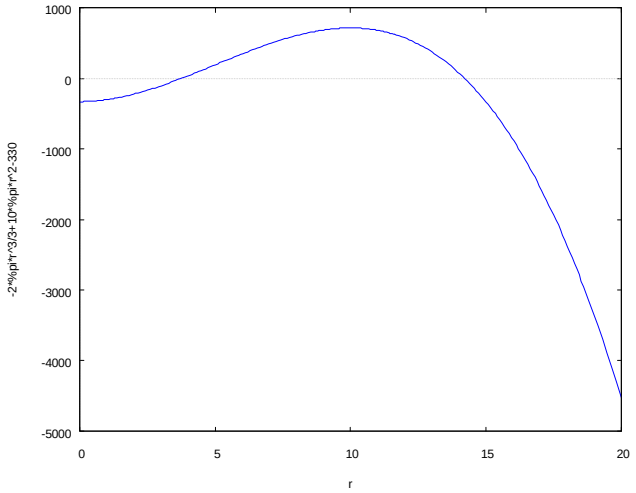
```
FUNCTION F (x)
V = 330
h = 10
F = V - 3.1416 * x * x * h + (2 / 3) * 3.1416 * x * x * x
END FUNCTION
```

che fornisce come soluzione un valore di r pari a 3.74 cm, per un'altezza h=10 cm.

- riprovare per valori di h pari ad 8, 9, 11 e 12 cm.

Maxima:

```
h:10
f(r):=%pi*h*r^2-(2/3)*%pi*r^3-330;
plot2d(f(r),[r,0,20]);
```



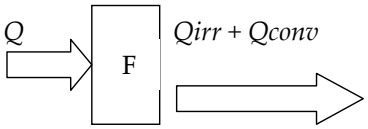

```
find_root(f(r),r,0,10);

find_root(f(r),0,99); (spiegare il risultato!)
```


☞ Determinare la temperatura T_c alla quale si porta un corpo al quale si fornisce una determinata potenza termica.
Inizialmente il corpo si trova alla stessa temperatura dell'ambiente T_a, ma con il tempo aumenta la propria temperatura, fino a raggiungere una condizione di equilibrio per la quale la potenza fornita eguaglia quella dissipata per convezione ed irraggiamento.

P.e. La resistenza elettrica di un generatore di aria calda, destinata ad un processo di essiccazione, dissipa una potenza Q pari a 1 kW. La resistenza è composta da un filo metallico, avvolto in spire, di raggio r=1mm e di lunghezza l=1m. Calcolare la temperatura alla quale si porta il filo quando la velocità dell'aria u, vale 20 m/s.

Potenza fornita = potenza dissipata per irraggiamento e convezione:

$$Qirr = S\sigma_0 (Tc - Ta)^4$$
$$Qconv = hS(Tc - Ta)$$



Essendo:

S = superficie laterale del filo: $2\pi rl$
σ_0 = 5.67E-8 W/m²K
Ta = 20°C
$h = Nu\ \lambda / l$ Nu (in regime di convezione forzata) = $C\ Re^a Pr^b$
$C = 0.023;\ a = 0.8;\ b = 0.4$

$Re = \rho u 2r/\mu$ Pr = $\mu Cs/\lambda$

$\mu(20°C) = 1.8E\text{-}5Pas$ $\lambda = 0.026W/mK$

$\rho(20°C) = 1.2kg/m^3$ $Cs = 1000J/kgK$

e dunque $h \cong 20\ W/(m^2K)$

Il problema si risolve semplicemente calcolando gli zeri dell'equazione:
$Q - (Qirr + Qconv) = 0$

$Q = 1000W$
$Qirr = 2\pi \cdot 0.001 \cdot 1 \cdot (Tc - 20)^4 \cong 0.0063 \cdot (Tc - 20)^4$
$Qconv = 10 \cdot 2\pi \cdot 0.001 \cdot 1 \cdot (Tc - 20) \cong 0.063 \cdot (Tc - 20)$ →

$$1000 - \left(0.0063 \cdot (Tc - 20)^4 + 0.063 \cdot (Tc - 20)\right) = 0$$

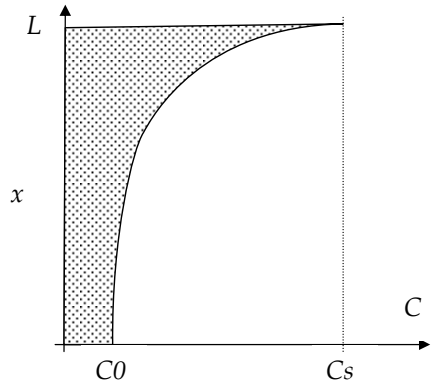
Si può infine calcolare la temperatura dell'aria calda *Tac* ancora attraverso la conservazione dell'energia:
$Q = \rho\, u\, A\, Cs\, (Tac-Ta)$, essendo A l'area della sezione trasversale dell'uscita.

+ Determinazione pH di una soluzione acquosa di un acido debole
$[H^+]^3 + Ka[H^+]^2 - (Kw + Ka\, C_{HA})\,[H^+] - KaKw = 0$

+ Determinazione pH di un anfolita
$[H^+]^4 + (K_{a1}+C_{Af})[H^+]^3 + (K_{a1}K_{a2}-K_w)[H^+]^2 - (K_{a1}K_{a2}C_{Af}+K_wK_{a1})[H^+] - K_wK_{a1}K_{a2} = 0$

☞ Problemi e dati tratti da Physical properties of foods and food processing systems di M.J. Lewis.

Un beaker alto L=0.03 m è riempito di albume d'uovo ed è posto in atmosfera satura di CO2. Si vuole stimare il valore di concentrazione della CO2 in condizioni di saturazione.



La diffusione del gas è molto lenta e per ottenere la saturazione di tutto l'albume potrebbero essere necessarie settimane. In alternativa potrebbe essere utilizzato un volume inferiore o comunque una disposizione in grado di garantire un migliore rapporto superficie volume. Tuttavia entrambe queste soluzioni presentano problemi legati all'evaporazione dell'acqua (e quindi alla trasformazione del materiale) ed al quantitativo minimo necessario per la misura della quantità di CO2 disciolta.

Si può procedere in altro modo: si osserva infatti che l'albume in superficie giunge velocemente ad un valore massimo di concentrazione (ovvero è saturo), mentre quello sul fondo resta sostanzialmente fermo al valore di concentrazione iniziale *c0* (circa 15 mol/m^3) per almeno 72 ore.
Si ipotizza dunque che, dopo 72 ore di esposizione all'atmosfera modificata, la distribuzione della CO2, in funzione della distanza dal fondo del beaker possa essere del tipo:

$$c(x) = c0 * exp(c1*x)$$

c(x): concentrazione [moli/m^3] di CO2 in funzione di *x*, distanza dal fondo [m];
c0: concentrazione iniziale dell'albume (nota e facile da misurare, circa 15 mol/m^3);

Se fosse noto il valore della costante *c1*, si potrebbe calcolare il valore in superficie *c(L)*, assimilabile a quello di saturazione, e per quanto detto risulterebbe:

$$c(L) = c0 * exp(c1*L)$$

E' tuttavia possibile determinare, con classiche metodiche analitiche, il valore medio di concentrazione di CO2 in tutto il volume del beaker in un tempo qualsiasi. A 72 ore risulta pari a circa 25 mol/m^3. Dunque il valore di *c1* può essere trovato imponendo che il valore medio, calcolato sulla base dell'equazione precedente, eguagli quello osservato sperimentalmente:

$$cm = \frac{1}{L}\int_0^L c(x)dx = \frac{c0}{L}\int_0^L e^{c1\cdot x}dx = \frac{c0}{L}\left[\frac{1}{c1}e^{c1\cdot x}\right]_0^L = \frac{c0}{c1\cdot L}\left(e^{c1\cdot L}-1\right)$$

$$\Rightarrow \quad 25 = \frac{15}{0.03\cdot c1}\left(e^{0.03c1}-1\right)$$

E' evidente che in tale equazione sono noti tutti i termini tranne l'incognita che ci interessa: *c1*. Tuttavia tale equazione però non può essere risolta con i normali metodi dell'analisi matematica e deve dunque essere risolta numericamente, per esempio utilizzando il metodo di Newton:

$$F(c1) = 25 - \frac{15}{0.03\cdot c1}\left(e^{0.03c1}-1\right)$$

$$c1 \Leftarrow c1 - \left(F(c1)\Big/F'(c1)\right)$$

Utilizzando le routine BASIC già viste si ottiene un valore per il parametro pari a circa *c1=31.58*.

Dunque l'equazione che descrive l'andamento della concentrazione, a partire dal fondo del contenitore, risulta:

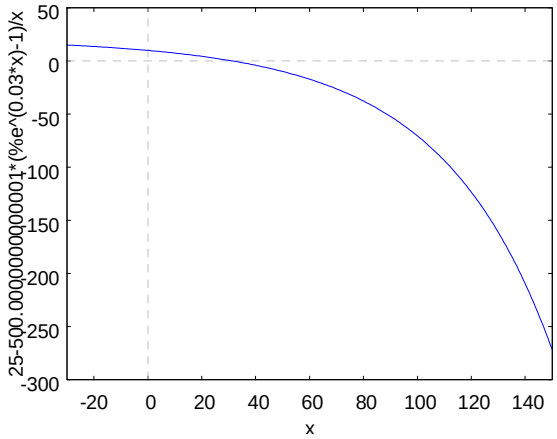
$$c(x) = 15 * exp(31.58*x)$$

ed in particolare la concentrazione in condizioni di saturazione vale:

$$c(0.03) = 15 * exp(31.58*0.03) \cong 40\ mol/m^3$$

Maxima:

```
F(c1) := 25-(15/(c1*0.03))*(exp(0.03*c1)-1)
plot2d(F(c1),[c1,-30,150]);
```




```
solve(F(c1)=0,c1);    =>  non si risolve analiticamente!

find_root(F(c1)=0,c1,-20,150);    ➔  x = 31.5801

load("mnewton")
mnewton(F(c1),c1,100);    ➔  x = 31.5801
```

6. UN INTERMEZZO STORICO

storiadelcalcolo

museo del calcolo

www.museoinformatica.it

http://www.mateureka.it/

www.criad.unibo.it/museum

www.top500.org

7. CALCOLO DI INTEGRALI

La soluzione di molti problemi di tipo fisico o ingegneristico, consiste nel calcolo di un integrale definito:

$$\int_a^b f(x)dx$$

generalmente è molto difficile (e spesso impossibile) determinare una primitiva della funzione *f(x)* con metodi analitici.

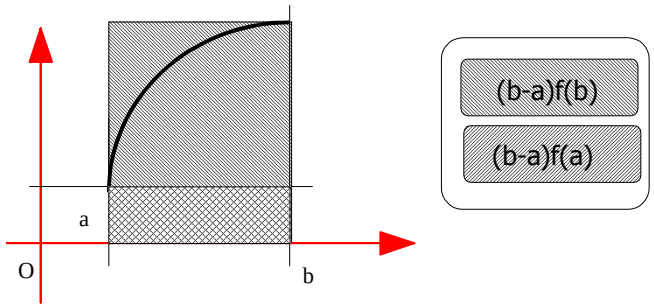
Esempi:

- texture analisys: calcolo dell'energia di rottura di un campione a partire dalla curva F-S;
- calcolo della quantità di fluido sulla base della misura istantanea di portata;
- spazio percorso a partire dalla legge oraria;
- calcolo della riduzione decimale in un processo di pastorizzazione:

$$RD = \log\left(\frac{N}{N_0}\right) = \frac{1}{D}\int_0^{\Delta t} 10^{\frac{T(t)-Tref}{z}}$$

L'idea di base del metodo numerico consiste nell'approssimare la funzione integranda con un polinomio *P(x)* (come del resto per la derivazione numerica e nel metodo di Newton) e quindi calcolare l'integrale definito di questo polinomio (la cui primitiva, come si sa, è un nuovo polinomio di facile determinazione). Il valore che si ottiene viene preso come approssimazione dell'integrale vero della *f(x)* sull'intervallo *[a,b]* (Newton-Cotes).

Considerando il polinomio più semplice, ovvero quello di grado 0, data una certa funzione *f(x)* un primo grossolano metodo di approssimarne l'integrale consiste nel calcolare l'area del rettangolo di altezza *f(a)* oppure *f(b)* e base *b-a*.

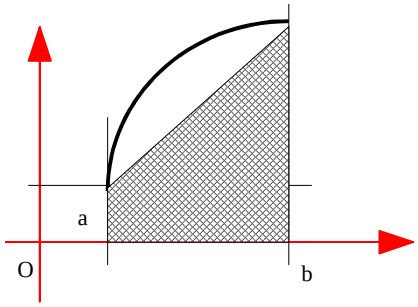


Evidentemente questo metodo va bene solo con funzioni *f(x)* che non variano molto nell'intervallo di integrazione. Il grado di precisione di questo metodo è 0, cioè solo le funzioni costanti, vengono integrate senza errori.

Si nota che, se la funzione è crescente, il primo sistema approssima l'integrale per difetto mentre il secondo fornisce un'approssimazione per eccesso. Si può allora aumentare la precisione calcolando una media dei due risultati precedenti:

$$I \approx (b-a)\frac{f(a)+f(b)}{2}$$

Si nota che tale metodo equivale al calcolo del trapezio di figura, ovvero approssima la curva *y=f(x)* con la retta passante per i punti *(a, f(a))* e *(b,f(b))*.

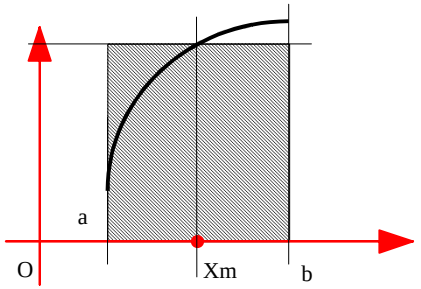


L'area di questo trapezio risulta infatti somma dell'area di un rettangolo e di un triangolo:

$$At = (b-a)\cdot f(a) + \frac{1}{2}(b-a)\cdot(f(b)-f(a)) = \frac{1}{2}(f(a)+f(b))(b-a)$$

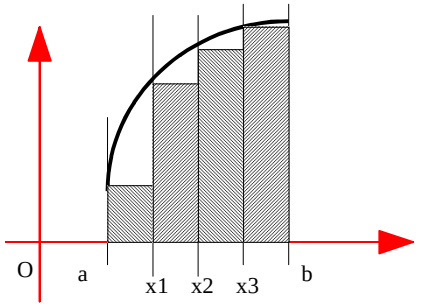
Tale metodo ha grado di precisione 1: l'errore è inferiore a quello del metodo precedente, e le funzioni lineari vengono integrate esattamente.

Un altro metodo di precisione 1 è quello indicato nella figura seguente, dove come nodo si è scelto il punto medio tra *a* e *b*:



$$I \approx (b-a)\cdot f\left(\frac{a+b}{2}\right)$$

I metodi descritti sono evidentemente molto grossolani, tuttavia è possibile migliorarne notevolmente la precisione, semplicemente suddividendo l'intervallo *[a,b]* sul quale si deve effettuare l'integrazione in *n* parti uguali, di ampiezza pari a *(b-a)/n*.
Così secondo il metodo del rettangolo, immaginando di discretizzare la funzione in *n* parti (ovvero *n+1* punti di campionamento), avremo una situazione del tipo:

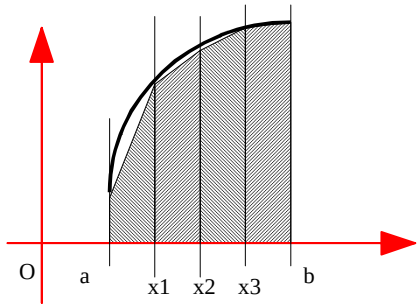


e la stima dell'area sottesa dalla curva sarà:

$$At \approx f(a)(x1-a)+f(x1)(x2-x1)+f(x2)(x3-x2)+f(x3)(b-x3)=$$
$$\Delta x\cdot\left(f(a)+f(x1)+f(x2)+f(x3)\right)=\sum_{i=1}^{n}\Delta x\cdot\left(f(x_i)\right)$$

al crescere di *n* (e quindi del tempo di calcolo) crescerà anche la precisione del risultato.
Se invece il metodo è quello del trapezio, applicato a ciascuno dei sottointervalli [x_i, x_{i+1}] risulta:

$$At \approx \frac{y1+y2}{2}\Delta x+\frac{y2+y3}{2}\Delta x+\ldots+\frac{y_{n-1}+y_n}{2}\Delta x=$$
$$=\frac{b-a}{2n}\left[\sum_{i=1}^{n-1}(y_i+y_{i+1})\right]$$



Se invece il metodo è quello nodo intermedio, applicato a ciascuno dei sottointervalli [x_i, x_{i+1}] risulta:

$$At \approx \frac{b-a}{n}\left[\sum_{i=1}^{n-1} f\left(\frac{x_i+x_{i+1}}{2}\right)\right]$$

Come esempio calcoliamo l'integrale di *sen(x)* nell'intervallo [*0*, *π*], suddividendo l'intervallo in 4 parti uguali risulta:

	0	**1**	**2**	**3**	**4**
x	*0*	*π/4*	*π/2*	*3π/4*	*π*
Sen(x)	*0*	*0.707*	*1*	*0.707*	*0*

$$At \cong \frac{\pi}{8}\left[\sum_{i=1}^{3}(y_i+y_{i+1})\right]=$$
$$=\frac{\pi}{8}\left[(0+0.707)+(0.0707+1)+(1+0.707)+(0.707+0)\right]=$$
$$=\frac{\pi}{8}4.828\cong 1.895951$$

ottenendo, con sole quattro suddivisioni, un errore relativo rispetto alla soluzione analitica (pari a 2) dell'ordine del 5%.

```
'Integrazione numerica

x1 = 0
x2 = 3.14159265/2
n = 10

' Integrazione con rettangoli (valore esatto = 1)
CLS
dx = (x2 - x1) / n
FOR x = x1 TO x2 STEP dx
        integrale = integrale + dx * y(x)
        PRINT integrale
NEXT x

SLEEP
PRINT

' Integrazione con trapezi
integrale = 0
FOR x = x1 TO x2 STEP dx
        integrale = integrale + dx * (y(x)+y(x+dx)) / 2
        PRINT integrale
NEXT x


FUNCTION y (x)
        y = SIN(x)
END FUNCTION
```

Conducendo con il codice precedente qualche esperimento con un diverso numero di suddivisioni, ci si rende facilmente conto (almeno in via qualitativa) del ruolo degli errori di cancellazione numerica e di discretizzazione. In definitiva l'errore totale ammette un minimo per un certo valore del passo di integrazione.

Al crescere di *N* diminuisce la differenza tra rettangoli e trapezi (sin(x) tra 0 e *Pi* con n=4):
http://www.wiley.com/college/mat/anton243310/mod2/applet1/applet1.html
https://www.google.com/search?hl=en&source=hp&q=applet+numerical+integration&btnG=Google+Search&aq=f&aqi=&aql=&oq=&gs_rfai=

☞ Calcolare il calore scambiato durante una trasformazione termodinamica, senza trascurare la relazione tra calore specifico e temperatura.
P.e. pressione costante, allora:

$$\Delta Q = M \cdot \int_{T_1}^{T_2} C_p(T)dT$$

Oppure, fornendo la quantità di calore ΔQ, quale sarà il corrispondente aumento di temperatura?

☞ Calcolare il lavoro richiesto da un compressore per comprimere una massa M di aria all'interno di una bombola di volume V.

☞ Determinare il valore della pressione di saturazione in funzione della temperatura di una sostanza, per mezzo dell'equazione di Clapeyron:

$$\frac{dP}{dT} = \frac{r}{T \cdot v_d}$$

con

$$r \approx a + bT + cT^2 + ... \quad e \quad v_d \approx \frac{RT}{p}$$

☞ Calcolare la potenza termica irraggiata da un corpo grigio in una determinata finestra di frequenze.

☞ Calcolare il lavoro svolto da un gas reale:

$$L_{12} = \int_{v_1}^{v_2} p \cdot dv$$

ricavando il valore della pressione dalla equazione di stato:

$$\left(p + \frac{a}{v^2}\right) \cdot (v - b) = R_w \cdot T$$

Ipotizzare una trasformazione 1) isoterma, 2) adiabatica, 3) politropica.

☞ Un serbatoio contiene un volume di acqua *V*, se la portata d'acqua in uscita vale $Q=Q_0 Sin(2\pi t/24)\cdot(1-0.1\cdot RND)$, calcolare quanta acqua esce nell'intervallo di tempo [t_1-t_2].

☞ Con riferimento all'esercizio precedente calcolare il tempo di svuotamento del serbatoio. Ripetere tutto aggiungendo una portata di ripristino costante.

- Calcolare l'effetto di pasteurizzazione in una crema al cioccolato, sottoposta ai seguenti trattamenti termici:

a) $T = 60°C$;
b) $T = k\ t$ con $k = 10°C/min$;
c) $T = 35\ [1 + sin(t\ 2Pi)]$;
In particolare considerare quello maggiormente efficace, tenendo conto anche del vincolo di una temperatura massima non superabile pari a 75°C.

$$riduzione\_decimale = \log\left(\frac{N}{N_0}\right) = \frac{1}{D}\int_0^{\Delta t} 10^{\frac{T(t)-Tref}{z}}$$

condizioni	Tref [°C]	D [min]	z [°C]
salmonella enteritis	50	2500	4.35
salmonella typhimurium (aw=0.5; pH=6.5-7.5)	67	1173	19
salmonella enteritis	121	0.25	3.29
salmonella enteritis	70	0.57	4.37

Con i dati relativi alla salmonella enteritis su cioccolato, per il trattamento isotermico a T=60°C, si ottengono i seguenti valori:

tempo di trattamento Δt [min]	riduzione decimale
1	0.08
2	0.16
4	0.31
8	0.63
16	1.27
32	2.55
64	5.09

Con i dati relativi alla salmonella enteritis su cioccolato, per il trattamento T=10t, si ottengono i seguenti valori:

tempo di trattamento Δt [min]	temperatura massima °C	riduzione decimale
1	10	4E-14
2	20	9E-12
4	40	3.8E-7
6	60	0.015
7	70	2.99
7.5	75	42

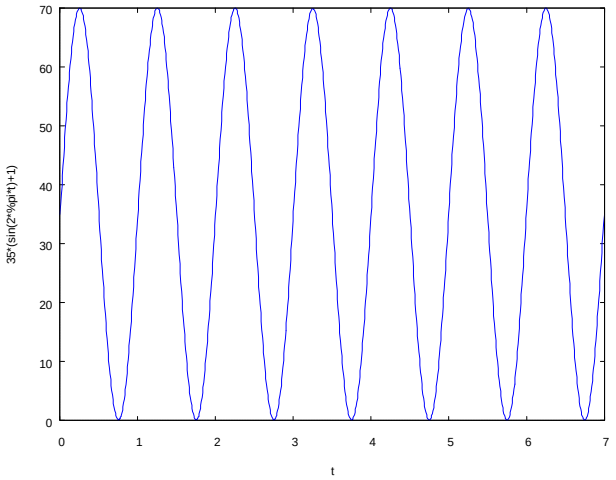
Con i dati relativi alla salmonella enteritis su cioccolato, per il trattamento sinusoidale, si ottengono i seguenti valori:

tempo di trattamento Δt [min]	temperatura massima °C	riduzione decimale
1	70	1.5
2	70	2.9
4	70	5.9
6	70	8.3
7	70	10.3
7.5	70	10.9

Maxima:

```
T(t):=35*(1+sin(2*t*%pi));

Deltat:7.0;

plot2d(T(t),[t,0,Deltat]);

Tref:50; D:2500; z:4.35;

f(t):=10^((T(t)-Tref)/z);

(1/D)*integrate(f(t),t,0,Deltat);

(1/D)*quad_qag(f(t),t,0,Deltat,1);
```

(l'integrale definito è il primo dei 4 valori: 10.34870904835985)



8. METODI EURISTICI

Un metodo euristico non utilizza un fondamento matematico, ma propone iterativamente nuove possibili soluzioni per tentativi, muovendosi all'interno della regione ammissibile per le variabili di ingresso del problema.

Il vantaggio principale di questo tipo di approccio è la maggior velocità con la quale si raggiunge una soluzione, sia per il minor numero di iterazioni necessarie sia per la maggior semplicità delle operazioni eseguite.
Viceversa, proprio per l'assenza di una formulazione matematica del problema, questo tipo di metodi non è in grado di assicurare l'ottimalità della soluzione: la soluzione raggiunta è sempre ammissibile, ma non sempre rappresenta l'ottimo assoluto.

L'applicazione di metodi euristici inoltre non richiede un'approfondita conoscenza delle relazioni tra le variabili che interagiscono nel problema. L'impostazione di un modello di tipo euristico è più semplice (dal punto di vista della matematica utilizzata) rispetto al corrispondente modello analitico. Si adatta bene ad un calcolatore e malissimo alle persone: richiede infatti un gran numero di operazioni ripetitive.

Si tratta di costruire un **generatore di probabili soluzioni** del problema (semplicemente campionando lo spazio dei parametri del problema) e poi di *filtrare* tali possibili soluzioni attraverso la serie di vincoli che definiscono una soluzione accettabile. (e.g. orario delle lezioni).

L'euristica (dal greco ευρίσκω, heurísko, letteralmente "scopro" o "trovo") è quella parte della ricerca il cui compito è di favorire l'accesso a nuovi sviluppi teorici o scoperte empiriche. Si definisce infatti procedimento euristico, un metodo di approccio alla soluzione dei problemi, che non segue un chiaro percorso, ma si affida all'intuito e allo stato temporaneo delle circostanze, al fine di generare nuova conoscenza. È opposto al procedimento algoritmico.

I modelli euristici possono essere esaustivi se sono in grado di provare tutte le soluzioni possibili, altrimenti si basano sull'utilizzo di generatori di numeri casuali con distribuzione determinata (metodi Montecarlo).

I metodi euristici possono anche essere indispensabili per affrontare problemi privi di una soluzione di tipo analitico, come p.e. la ricerca di terne pitagoriche:

Terne pitagoriche

```
' Terne pitagoriche.
' Ricerca dei triangoli rettangoli con misure dei lati
' espresse da numeri interi.


Max = 1000
CLS
' Generatore di soluzioni possibili:
' si esplora TUTTO lo spazio dei parametri
FOR a = 1 TO Max
        FOR b = 1 TO Max
                c = sqr(a*a+b*b)
           ' Filtro che definisce
           ' una soluzione accettabile
                IF (c = INT(c)) THEN PRINT a, b, c
        NEXT b
NEXT a
END
```

e.g.: problema del bancomat: $20a+50b=x$, con a, b, x interi.

Un caso particolare di integrazione euristica: un generatore Montecarlo di π

Consideriamo una superficie quadrata di area *A* tracciata sul pavimento di una stanza. Tale superficie sia divisa in due parti uguali, p.e. una rossa ed una verde. Immaginiamo di lanciare ripetutamente un sassolino e di applicare il seguente algoritmo:

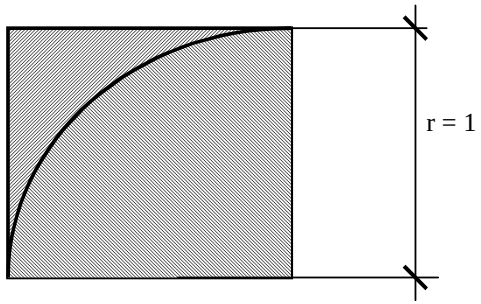
- se il sassolino si ferma fuori dal quadrato, allora il lancio viene ignorato; altrimenti
- si conteggia il numero di lanci (*Nl*);
- se il sassolino cade nell'area verde si incrementa uno specifico contatore (*Nv*).

Al crescere del numero *Nl* a quale valore tenderà il rapporto *Nv/Nl*?

Nell'ipotesi che i lanci siano assolutamente casuali, la probabilità di colpire l'area verde è identica a quella di colpire l'area rossa, di conseguenza il rapporto *Nv/Nl* tende ad ½.

Se le due aree non fossero uguali, p.e. l'area rossa fosse il doppio di quella verde, allora la probabilità di colpire l'area rossa sarebbe doppia rispetto a quella di colpire l'area verde. Dalle relazioni *Pr=2Pv* e *Pr+Pv=1*, si ricava che *Pr=2/3 e Pv=1/3*.
Con tale ipotesi, all'aumentare del numero di lanci, il rapporto *Nv/Nl* tende a 1/3.

Consideriamo ora una configurazione un po' differente della nostra mattonella quadrata:



Si lancia un sassolino: la probabilità che l'area del cerchio venga colpita (*Pc*) è proporzionale all'area stessa. Nell'ipotesi che tutti i sassolini caschino all'interno del quadrato, la probabilità che l'area del cerchio venga colpita, vale il rapporto tra l'area del cerchio e l'area totale del quadrato:

$$Pc = \frac{\pi r^2}{4r^2} = \frac{\pi}{4}$$

Ricordando la definizione di probabilità come limite delle frequenza relativa per un numero di osservazioni tendente ad infinito, indicando con *Nq* il numero totale di lanci e con *Nc* il numero di lanci che colpiscono il solo spicchio circolare, possiamo scrivere:

$$Pc = Lim_{Nq\to\infty} \frac{Nc}{Nq} = \frac{\pi}{4}$$

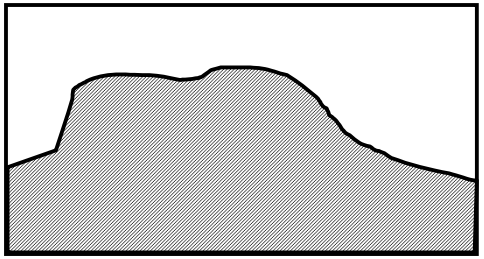
ovvero:

$$\pi = 4 \cdot Lim_{Nq\to\infty} \frac{Nc}{Nq}$$

```
'       Generatore Montecarlo di Pi-greco

RANDOMIZE TIMER
Nc = 0

FOR Nq = 1 TO 30000
       ' Generatore di soluzioni da provare:
       '  scelte  con  un  campionamento  casuale  tra  le
infinite
       ' soluzioni possibili.
        x = RND
        y = RND

        IF (SQR(x * x + y * y) <= 1) THEN Nc = Nc + 1
        pi = 4 * Nc / Nq
        PRINT pi
NEXT Nq
```

- Integrazione Montecarlo

Con la stessa tecnica è evidentemente possibile calcolare l'integrale definito di una funzione qualsiasi:



Simulazione delle code alle casse di un supermercato

```
'      Code

RANDOMIZE TIMER
MaxNCasse = 15    ' Massimo numero di casse utilizzabili
DIM Np(MaxNCasse) ' Numero di persone in fila alla cassa i-esima
Nc = 1            ' Numero casse aperte: inizialmente una sola

CLS
FOR t = 0 TO 3000         ' t in minuti
        PRINT : PRINT "Nc(t):"; Nc; "("; t; ") ";

        ' Affluenza alle casse (in)
        FOR i = 1 TO Nc
              ' Numero di persone che arrivano in un minuto
              ' alla cassa i-esima (nell'ora di punta 3/minuto).
                Tmp = RND * 3 * ABS(COS(t / (60 * (RND + 7.5))))
                Np(i) = Np(i) + Tmp
        NEXT i

        ' Pagamento alle casse (out)
        FOR i = 1 TO Nc
                ' Numero di persone processate in un minto
              ' alla cassa i-esima (almeno 0.5/minuto).

                Np(i) = Np(i) - .5
              ' La fila piu' lunga funzia come gradiente:
              ' l'impiegto è più veloce.
                Np(i) = Np(i) - RND * Np(i) * .05
                IF Np(i) < 0 THEN Np(i) = 0
        NEXT i

        'Apertura nuove casse
        FOR i = 1 TO Nc
                IF (Np(i) > 30 AND Nc < MaxNCasse) THEN
                        Nc = Nc + 1

                        'Ridistribuzione
                        Media = 0
                        Np(Nc) = 0
                        FOR j = 1 TO Nc
                                Media = Media + Np(j)
                        NEXT j
                        Media = Media / Nc

                        FOR j = 1 TO Nc
                                Np(j) = Media
                        NEXT j

                END IF
```

```
        NEXT i

        'Chiusura casse
        FOR i = 1 TO Nc
                IF (Np(i) < 6 AND Nc > 1) THEN
                Nc = Nc - 1
                IF Nc < 0 THEN Nc = 0

                'Ridistribuzione

                Media = 0
                FOR j = 1 TO Nc + 1
                        Media = Media + Np(j)
                NEXT j
                Media = Media / (Nc + 1)

                FOR j = 1 TO Nc
                        Np(j) = Media
                NEXT j

                END IF
        NEXT i

      'Stampa situazione attuale
        FOR i = 1 TO Nc
                PRINT INT(Np(i));
        NEXT i
'INPUT a$
NEXT t
```

☞ Calcolare il numero medio di persone in coda.
☞ Calcolare la massima lunghezza di una coda.
☞ Calcolare la distribuzione di frequenza nella giornata.

☞ Applicare la tecnica illustrata al caso delle scorte di un magazzino (p.e. frutta: la frutta arriva in grossi lotti, in pochi periodi dell'anno, mentre il consumo fluttua in modo casuale ma con continuità attorno ad un valore costante durante tutto l'anno).
☞ Quale sarà la minima quantità da stoccare? Qual'è il massimo tempo di immagazzinamento di un prodotto ?

☞ Applicare la tecnica illustrata al caso della richiesta di potenza frigorifera di un magazzino (suggerimenti: le aperture delle porte e l'introduzione/estrazione di materiali a temperatura diversa è casuale durante la giornata. Occorre tenere conto poi dei cicli giorno/notte e di quelli stagionali).

☞ Applicare la tecnica illustrata al caso dei pasti serviti da una mensa pubblica.

☞ Problemi di cammino minimo e di ottimizzazione (componenti/regime di impianti, approvvigionamento, trasporto, gestione linee di lavorazione, rapporti costo/beneficio), best-fitting.

☛ Una noterella: qualcuno potrebbe a questo punto chiedersi perchè dobbiamo metterci a studiare i dettagli di questi metodi di calcolo, quando esiste tantissimo software che (sapendolo utilizzare) può risolvere il problema con pochi colpi di click.
A costoro indicherei di spendere questi pochi click, nella ricerca in rete del breve scritto *Come vincere nella lotta con la Bestia. Istruzioni per un uso razionale del computer* di Umberto Eco.

Modulo 2

9.Il calcolo automatico con Maxima

http://en.wikipedia.org/wiki/Maxima_(software)
http://www.permucode.com/maxima/ (*Maxima portable*)

Maxima is a free *computer algebra system* based on a 1982 version of Macsyma. It is written in Common Lisp and released under the ***GNU General Public License*** and runs on all POSIX platforms such as Unix, BSD, and Linux as well as under Microsoft Windows.

Maxima is based on a 1982 version of Macsyma, which was ***developed at MIT*** with funding from the United States Department of Energy and other government agencies. A version of Macsyma was maintained by Bill Schelter from 1982 until his death in 2001. In 1998 Schelter obtained permission from the Department of Energy to release his version under the GPL. That version, now called Maxima, is maintained by an independent group of users and developers. Maxima does not include any of the many modifications and enhancements made to the commercial version of Macsyma during 1982–1999. Though the core functionality remains similar, code depending on these enhancements may not work on Maxima, and bugs which were fixed in Macsyma may still be present in Maxima, and vice-versa.

Features

Maxima includes a complete ***programming language*** with ALGOL-like syntax but Lisp-like semantics. It uses Gnuplot for drawing.

Since Maxima is written in Common Lisp, it can be accessed programmatically and extended, as the underlying Lisp can be called from Maxima.

Maxima can compute derivatives and integrals, expand in Taylor series, take limits, and obtain exact solutions to ordinary differential equations. We begin by defining the symbol `f` to be the following function of `x`:

Numeric calculations

Maxima is a full-featured CAS (computer algebra system) that specializes in symbolic operations but it also offers numerical capabilities such as:
arbitrary-precision arithmetic: integers and rational numbers which can grow to sizes limited only by machine memory, and floating point numbers whose precision can be set arbitrarily large ("bfloats").

For calculations which use floating point and arrays heavily, Maxima offers the possibility of generating code in other programming languages (notably Fortran) which may execute more efficiently.

Maxima is a general-purpose system, and special-case calculations such as factorization of large numbers, manipulation of extremely large polynomials, etc. are often better done in specialized systems.

Interfaces (*XMaxima* and *wxMaxima*)

Various graphical user interfaces are available for Maxima. ***wxMaxima*** is a cross-platform GUI based on wxWidgets. The GNU TeXmacs mathematical editor program can be used to provide an interactive GUI for Maxima, as can SAGE. Other options include the Imaxima front end as well as an Emacs interaction mode.

http://maxima.sourceforge.net/

- Comparison of computer algebra systems

- The official Maxima website
- wxMaxima
- Maxima Beginner's FAQ
- Maxima 10 Minute Tutorial
- The HTML Maxima Manual in English
- Short list of useful examples
- Imaxima, Emacs front end that includes typesetting.
- **(Japanese)** Various plotting examples
- A Maxima-Gnuplot interface- drawing examples

- DragMath, an open-source online equation editor that can export Maxima, and other formats.
- http://elearning.cerfacs.fr/miscellaneous/tools/maxima/index.php , online interface to Maxima

Euler

http://euler.rene-grothmann.de/
Run *Euler portable*
Extras -> ToggleMaximaModeOn
Help -> Open Introductions and Examples -> "00 A First Welcome.en"

Maxima as a calculator:

You can use Maxima as a fast and reliable calculator whose precision is arbitrary within the limits of your PC's hardware. Maxima expects you to enter one or more commands and expressions separated by a semicolon character (;) plus ***SHIFT+CR***, just like you would do in many programming languages.

- `(%i1) 9+7;` `(%o1) 16`
- `(%i2) -17*19;` `(%o2) -323`
- `(%i3) 10/2;` `(%o3) 5`

For the sake of simplicity, from now on we will omit the numbered input and output prompts produced by Maxima's console, and indicate the output with a => sign. When the numerator and denominator are both integers, a reduced fraction or an integer value is returned. These can be evaluated in floating point by using the *float* function (or *bfloat* for big floating point numbers):

- `2/6;` => $\frac{1}{3}$
- `float(1/3);` => 0.33333333333333
- `1/3.0;` => 0.33333333333333
- `26/4;` => $\frac{13}{2}$
- `float(26/4);` => 6.5
- float(3/9)
- 3/9

As mentioned above, big numbers are not an issue:

- `13^26;` => 91733330193268616658399616009
- `13.0^26` => $9.1733330193268623\ 10^{28}$
- `30!;` => 265252859812191058636308480000000
- `float((7/3)^35);` => $7.5715969098311943\ 10^{12}$

Constants and common functions

Here is a list of common constants in Maxima, which you should be aware of:

- **%e - Euler's Number**
- **%pi -** π
- **float(%pi)**
- %i - the imaginary unit ($\sqrt{-1}$)
- inf - real positive infinity (∞)
- minf - real minus infinity ($-\infty$)
- infinity - complex infinity

We can use some of these along with common functions:

- `sin(%pi/2) + cos(%pi/3);` => $\frac{3}{2}$
- `log(%e);` => 1

Defining functions and variables

Variables can be assigned through a colon ':' and functions through ':='. The following code shows how to use them:

```
a:7; b:8;       => 7    => 8
sqrt(a^2+b^2);  => √113
f(x):= x^2 -x + 1;      => x^2 − x + 1
f(3);   => 7
f(a);   => 43
f(b);   => 57
```

Please note that Maxima only offers the natural logarithm function *log*. *log10* is not available by default but you can define it yourself as shown below:

```
log10(x):= log(x)/log(10);      =>  log10(x) := log(x)/log(10);
log10(10)               => 1
```

Symbolic Calculations

- **1/2+1/3;**

Notes using Maxima

• To abort a computation without leaving Maxima, type ^C. (Here ^ stands for the control key, so that ^C means first press the key marked control and hold it down while pressing the C key.) It is important for you to know how to do this in case, for example, you begin a computation which is taking too long. For example:

```
(%i1) sum (1/x^2, x, 1, 10000);

Maxima encountered a Lisp error:

 Console interrupt.
```

• In order to tell Maxima that you have finished your command, use the semicolon (;), followed by a return. Note that the return key alone does not signal that you are done with your input.

• An alternative input terminator to the semicolon (;) is the dollar sign ($), which, however, supresses the display of Maxima's computation. This is useful if you are computing some long intermediate result, and you don't want to waste time having it displayed on the screen.

• If you wish to repeat a command which you have already given, say on line (%i5), you may do so without typing it over again by preceding its label with two single quotes ('), i.e., '%i5. (Note that simply inputing %i5 will not do the job - try it.)

• If you want to refer to the immediately preceding result computed my Maxima, you can either use its o label, or you can use the special symbol percent (%).

The common arithmetic operations are

+	addition
-	subtraction
*	scalar multiplication
/	division
^	or ** exponentiation
.	matrix multiplication
sqrt(x)	square root of x.

Maxima's output is characterized by exact (rational) arithmetic. E.g.,

- (%i1) 1/100 + 1/101;

```
                            201
(%o1)                       -----
                            10100
```

If irrational numbers are involved in a computation, they are kept in symbolic form:

- (%i2) (1 + sqrt(2))^5;


```
                                     5
(%o2)                    (sqrt(2) + 1)
```

The number of significant figures displayed is controlled by the Maxima variable fpprec, which has the default value of 16:

```
(%i6) fpprec;
(%o6)                           16
```

Here we reset fpprec to yield 100 digits:

```
(%i7) fpprec: 100;
(%o7)                           100
(%i8) ''%i5;
(%o8) 8.20121933088197564152489730020812442785204843859314941221#
2371240173124187540110412666123849550160561B1
```

Solving Equations and Systems

We can easily solve equations and systems of equations through the function *solve*:

- **solve(x^2=4, x);**
- **solve(x^2-4=0,x);** **=>** $[x=-2, x=2]$
- **solve(x^2=k, x);**
- **solve(x^2-4=k, x);**
- **solve(x^2-4=k, k);**
- **solve((b-a)/(c-b)= a/c, b);**
- solve(x^k=2,k);
- **solve(x^3=1,x);** **=>** $\left[x=\frac{\sqrt{3}\,i-1}{2}, x=-\frac{\sqrt{3}\,i+1}{2}, x=1\right]$
- (solve([cos(x)^2-x=2-sin(x)^2], [x])); => $[x=-1]$
- **solve([x - 2*y = 14, x + 3*y = 9],[x,y]);** **=>** $[[x=12, y=-1]]$
- **solve(x^3-3*x+1,x);**
- solve([x^2+2*x+y+3,x*y-3],[x,y]);

- eq1: x^2 + 3*x*y + y^2 = 0;
- eq2: 3*x + y = 1;
- solve([eq1, eq2]);

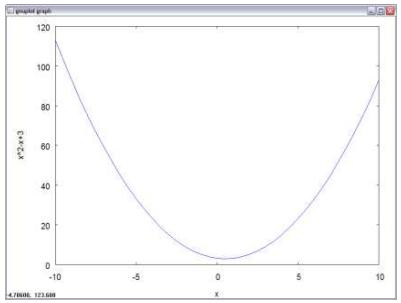
Numerical solution and function definition:

```
f(x) := x*x-3
f(2)
solve(f(x),x)
find_root(f(x),x,0,99)

mnewton(2*a^a-5,a,1);
load("mnewton")
```

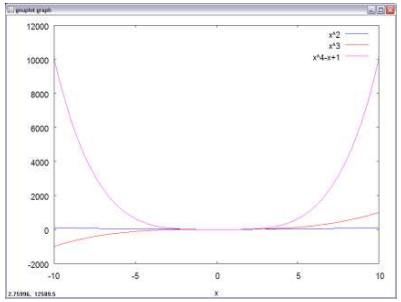
2D and 3D Plotting

Maxima enables us to plot 2D and 3D graphics, and even multiple functions in the same chart. The functions *plot2d* and *plot3d* are quite straightforward as you can see below. The second (and in the case of plot3d, the third) parameter, is just the range of values for x (and y) that define what portion of the chart gets plotted.

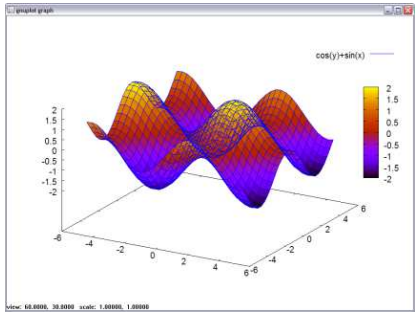
- **plot2d(x^2-x+3,[x,-10,10]);**



- **plot2d([x^2, x^3, x^4 -x +1] ,[x,-10,10]);**



- **f(x,y):= sin(x) + cos(y);**
- **plot3d(f(x,y), [x,-5,5], [y,-5,5]);**



- **plot2d(sin(x),[x,0,2*%pi]);**
- plot3d(atan(-x^2+y^3/4),[x,-4,4],[y,-4,4],[grid,50,50]);
- plot3d(atan(-x^2+y^3/4),[x,-4,4],[y,-4,4],[grid,50,50],[gnuplot_pm3d,true]);
- plot3d(atan(-x^2+y^3/4),[x,-4,4],[y,-4,4],[grid,50,50],[gnuplot_pm3d,true],[gnup
- lot_preamble,"set pm3d at s;unset surface;set contour;set cntrparam levels
- 20;unset key"]);
- surface-pm3d-2:
- plot3d(atan(-x^2+y^3/4),[x,-4,4],[y,-4,4],[grid,50,50],[gnuplot_pm3d,true],[gnup
- lot_preamble,"set pm3d at b"]);
- plot3d(cos(-x^2+y^3/4),[x,-4,4],[y,-4,4],[grid,150,150]);
- plot3d(cos(-x^2+y^3/4),[x,-4,4],[y,-4,4],[gnuplot_preamble,"set view map;
- unset surface"],[gnuplot_pm3d,true],[grid,150,150]);
- plot2d(sin(x),[x,-5,5]);
- plot3d(sin(x*y),[x,-3,3],[y,-3,3]);
- plot2d(sin(x)/x,[x,-5,5]);
- plot3d(sin(sqrt(x^2+y^2))/sqrt(x^2+y^2),[x,-12,12],[y,-12,12]);
- plot3d([cos(y)*(10.0+6*cos(x)),sin(y)*(10.0+6*cos(x)),-6*sin(x)],
- [x,0,2*%pi],[y,0,2*%pi],['grid,40,40]);
- plot3d([5*cos(x)*(cos(x/2)*cos(y)+sin(x/2)*sin(2*y)+3.0)-10.0,
- -5*sin(x)*(cos(x/2)*cos(y)+sin(x/2)*sin(2*y)+3.0),
- 5*(-sin(x/2)*cos(y)+cos(x/2)*sin(2*y))],
- [x,-%pi,%pi],[y,-%pi,%pi],['grid,40,40]);
- plot2d(sec(x),[x,-2,2],[y,-20,20],[nticks,200]);
- plot2d([parametric,cos(t),sin(t),[t,-%pi*2,%pi*2]]);
- plot2d([x^3+2,[parametric,cos(t),sin(t),[t,-5,5]]], [x,-3,3]);

Limits

- limit((1+1/x)^x,x,inf); => %e
- limit(sin(x)/x,x,0); => 1
- limit(2*(x^2-4)/(x-2),x,2); => 4
- limit(log(x),x,0,plus); => $-\infty$
- limit(sqrt(-x)/x,x,0,minus); => $-\infty$

Differentiation

- **diff(sin(x), x); =>** $\cos(x)$
- **diff(x^x, x); =>** $x^x\,(\log x+1)$
- We can calculate higher order derivatives by passing the order as an optional number to the *diff function*:

  diff(tan(x), x, 4); => $8\sec^2 x\tan^3 x+16\sec^4 x\tan x$
- diff(cos(x)^5,x)

Integration

Maxima offers several types of integration. To symbolically solve indefinite integrals use *integrate*:

- **integrate(1/x, x);** => $\log(x)$

For definite integration, just specify the limits of integrations as the two last parameters:

- integrate(sin(x),x)
- integrate(sin(x),x,0,%pi)
- integrate(sin(x),x,a,b)
- integrate(sin(x),x,1,2)
- float(integrate(sin(x),x,0,%pi))
- **integrate(x+2/(x -3), x, 0,1);** => $-2\log 3 + 2\log 2 + \frac{1}{2}$
- **integrate(sin(x),x,0,%pi);**
- integrate(%e^(-x^2),x,minf,inf);=> $\sqrt{\%pi}$
- integrate(tan(x),x);
- integrate(%e^(-x^2),x,minf,inf);

Ordinary differential equations

```
depends(y,x);
diff(y,x)=(4-2*x)/(3*y^2-5);
ode2(%,y,x);
latex(%);
```

Using the quote operator we can write differential equations:

```
(%i11) 'diff (y, x, 2) + 'diff (y, x) + y;
                           2
                          d y   dy
(%o11)                    --- + -- + y
                            2   dx
                          dx
```

Maxima's `ode2` function can solve first and second order ODE's:

```
(%i12) ode2 (%o11, y, x);
                - x/2          sqrt(3) x            sqrt(3) x
(%o12) y = %e       (%k1 sin(---------) + %k2 cos(---------))
                               2                    2
```

8 A partial list of Maxima functions

See the Maxima reference manual `doc/html/maxima_toc.html` (under the main Maxima installation directory). From Maxima itself, you can use `describe(`*`function name`*`)`.

allroots(a)

Finds all the (generally complex) roots of the polynomial equation `a`, and lists them in `numer`ical format (i.e. to 16 significant figures).

batch(a)

Loads and runs a program with filename `a`.

desolve(a,b)

Attempts to solve a linear system `a` of ODE's for unknowns `b` using Laplace transforms.

diff(a,b1,c1,b2,c2,...,bn,cn)

Gives the mixed partial derivative of `a` with respect to each `bi`, `ci` times. For brevity, `diff(a,b,1)` may be represented by `diff(a,b)`. `'diff(...)` represents the unevaluated derivative, useful in specifying a differential equation.

expand(a)

Algebraically expands `a`. In particular multiplication is distributed over addition.

factor(a)

Factors `a`.

integrate(a,b)

Attempts to find the indefinite integral of `a` with respect to `b`.

integrate(a,b,c,d)

Attempts to find the indefinite integral of `a` with respect to `b`. taken from `b=c` to `b=d`. The limits of integration `c` and `D` may be taken is `inf` (positive infinity) of `minf` (negative infinity).

kill(a)

Removes the variable `a` with all its assignments and properties from the current Maxima environment.

limit(a,b,c)

Gives the limit of expression `a` as variable `b` approaches the value `c`. The latter may be taken as `inf` of `minf` as in `integrate`.

ode2(a,b,c)

Attempts to solve the first- or second-order ordinary differential equation `a` for `b` as a function of `c`.

realpart(a)

Returns the real part of `a`.

solve(a,b)

Attempts to solve the algebraic equation `a` for the unknown `b`. A list of solution equations is returned. For brevity, if `a` is an equation of the form `c = 0`, it may be abbreviated simply by the expression `c`.

subst(a,b,c)

Substitutes `a` for `b` in `c`.

taylor(a,b,c,d)

Expands `a` in a Taylor series in `b` about `b=c`, up to and including the term $(b-c)^d$. Maxima also supports Taylor expansions in more than one independent variable; see the Manual for details.

`trigexpand(a)`

Is a trig simplification function which uses the sum-of-angles formulas to simplify the arguments of individual `sin`'s or `cos`'s. For example, `trigexpand(sin(x+y))` gives `cos(x) sin(y) + sin(x) cos(y)`.

`trigreduce(a)`

Is a trig simplification function which uses trig identities to convert products and powers of `sin` and `cos` into a sum of terms, each of which contains only a single `sin` or `cos`. For example, `trigreduce(sin(x)^2)` gives `(1 - cos(2x))/2`.

`trigsimp(a)`

Is a trig simplification function which replaces `tan`, `sec`, etc., by their `sin` and `cos` equivalents. It also uses the identity $\sin()^2 + \cos()^2 = 1$.

La manipolazione simbolica delle espressioni matematiche assistita dal calcolatore: una rassegna sintetica ed un esempio su uno scuotitore per la raccolta meccanica del pomodoro.

Angelo Fabbri *

La manipolazione automatica delle espressioni matematiche in forma simbolica, come completamento del tradizionale metodo numerico, rappresenta attualmente, grazie anche alla discreta disponibilità di pacchetti software efficienti a costi contenuti, un valido strumento per affrontare la costruzione di modelli teorici complessi.

10. introduzione

I calcolatori digitali sono commercialmente disponibili sin dagli anni '50 e da allora si sono succedute diverse generazioni di macchine nello sviluppo della tecnica costruttiva e dei linguaggi di programmazione, durante questo periodo il calcolatore è stato impiegato principalmente nel trattamento di numeri. Particolarmente nel campo della ricerca scientifica, gli algoritmi studiati e l'espressione della capacità di lavoro hanno fatto riferimento finora alle capacità aritmetiche delle macchine. Attualmente nel campo della matematica applicata e dell'ingegneria in generale, dove si impieghi un calcolatore per affrontare un qualsiasi problema di calcolo, la tradizione offre in primo luogo tecniche e soluzioni che coinvolgono direttamente operazioni tra numeri [24].

* Dr. Ing. ANGELO FABBRI, Dottorando di Ricerca. Istituto di Meccanica Agraria dell'Università, Bologna.

Per esempio, se è noto il valore dei tre coefficienti α, β e γ del polinomio

$$P(x) = \alpha + \beta \cdot x^3 + \gamma \cdot x^6 \quad \forall \alpha, \beta, \gamma \in C$$

è possibile, con algoritmi ben noti come per esempio quello di Bairstow, determinare il valore (approssimato all'aritmetica di macchina) delle sei soluzioni reali o complesse dell'equazione

$$P(x) = 0$$

L'approccio simbolico invece permette di rimanere nell'ambito (generalmente più vicino al modo di pensare di coloro che posseggono una formazione matematica tradizionale) delle regole formali dell'algebra, ottenendo in luogo dei sei valori delle radici del polinomio una espressione simbolica che rappresenta in modo parametrico tutte le possibili soluzioni dell'equazione:

$$x_{1,2,3,4,5,6} = \left(\frac{-\beta \pm \sqrt{\beta^2 - 4 \cdot \alpha \cdot \gamma}}{2 \cdot \gamma} \right)^{\frac{1}{3}}$$

Analogamente, applicando uno schema di risoluzione numerico, come per esempio quello di Runge-Kutta, ad una semplice equazione differenziale alle derivate ordinarie del tipo:

$$\begin{cases} \ddot{x} = -\Omega \cdot x \quad \Omega \in R > 0 \\ x(t_0) = x_0 \quad \dot{x}(t_0) = \dot{x}_0 \end{cases}$$

si ottiene una serie di coppie di numeri del tipo $\{x_i, t_i\}$ con i=1÷n, ciascuna delle quali rappresenta una soluzione puntuale approssimata dell'integrale particolare corrispondente alle condizioni iniziali specificate. Tale serie di coppie di valori pur rappresentando la soluzione del problema risulta poco espressiva se lasciata in forma di vettore di numeri; in genere si cerca di ricavare qualche interpretazione del fenomeno che esprime attraverso una rappresentazione grafica su di un piano x-t, ma in ogni caso è difficile estrapolare una legge o capire eventuali periodicità o individuare punti di singolarità. Applicando invece un metodo analitico si ottiene la ben nota soluzione:

$$x = A \cdot Sin\left(\sqrt{\Omega} \cdot t + \alpha\right)$$

Questi semplici esempi forniscono un'idea della efficacia del metodo simbolico attraverso il quale si ottengono informazioni non immediatamente estrapolabili dalla soluzione numerica dei problemi; in particolare si ottengono informazioni sulla qualità della risposta di un sistema, come potrebbe essere per esempio il carattere oscillatorio sinoidale ed il fatto che la frequenza dell'oscillazione sia legata alla radice quadrata del termine Ω.

Dunque, nonostante le procedure numeriche costituiscano uno strumento d'analisi molto potente, esse presentano alcune intrinseche limitazioni: la generale impossibilità di sintetizzare risultati di validità generale e la precisione finita [14, 16]. La precisione finita dell'aritmetica di macchina porta, nell'elaborazione diretta di quantità numeriche, ad inevitabili errori di troncamento nella valutazione delle espressioni trascendenti e di quelle irrazionali. Per algoritmi iterativi, o per problemi mal condizionati, come la soluzione di sistemi lineari del tipo di Vandermonde, la propagazione dell'errore può portare a problemi inaccettabili di perdita di cifre significative attraverso il meccanismo della cancellazione numerica [20]. Inoltre i tradizionali linguaggi di programmazione risentono delle caratteristiche delle diverse macchine utilizzate, nel senso che non permettono di rappresentare direttamente numeri maggiori di una costante prefissata.

Gli algoritmi di calcolo numerico sono spesso di applicabilità generale (p.e. il citato metodo di Runge-Kutta può essere applicato a qualunque tipo di equazione differenziale alle derivate ordinarie, lineare o non lineare) ma anche se il risultato è ben approssimato, esso rappresenta solo una soluzione particolare del sistema fisico al quale si riferisce, caratteristica di un determinato insieme numericamente noto dei parametri geometrici, di quelli dimensionali, delle condizioni al contorno e di quelle iniziali.

Le applicazioni in generale, e quelle mostrate in particolare, evidenziano quanto la soluzione in forma chiusa (ottenuta naturalmente anche in modo non automatizzato) sia generalmente preferibile a quella numerica sia nella fase di messa a punto di un modello matematico sia nella interpretazione del comportamento di un sistema fisico. Generalmente un modello in forma simbolica consente di eseguire dei processi di sintesi, mentre uno schema numerico è certamente più orientato a processi di analisi, o come si usa distinguere in ambito ingegneristico, di affrontare rispettivamente *problemi di progetto* e *problemi di verifica*. Evidentemente, inoltre, trattare un'espressione in forma simbolica comporta il vantaggio di non introdurre errori dovuti alla rappresentazione interna dei numeri.

Gli strumenti software che in anni più recenti hanno affrontato il problema della rappresentazione e del trattamento delle espressioni matematiche direttamente nella loro forma simbolica vengono denominati generalmente *sistemi di manipolazione simbolica*. Una delle differenze più evidenti mostrate da un sistema di calcolo simbolico rispetto ad un tradizionale ambiente di programmazione consiste senza dubbio nel concetto di *variabile*. In un ambiente di calcolo tradizionale, il concetto di variabile è sinonimo di contenitore di informazioni. Così, una sequenza di operazioni come x=1.5 e y=x+7 assume il significato di memorizzazione di risultati numerici in contenitori chiamati x ed y. La variabile rappresenta da un punto di vista strettamente informatico l'estensione del concetto di indirizzo di memoria; pertanto le variabili x ed y non sono altro che sinonimi degli indirizzi di memoria della macchina dove verranno memorizzati i loro valori. Da ciò deriva che non ha senso utilizzare una variabile se prima non le è stato assegnato un valore, e quindi le operazioni precedenti vanno eseguite nel loro proprio ordine. Un sistema di calcolo simbolico viceversa considera la variabile in modo più aderente al suo significato matematico, non trattandola come un oggetto che necessariamente deve contenere un valore, ma piuttosto come un simbolo, ed il calcolo null'altro che applicazioni di regole appropriate ai simboli coinvolti.

Un algoritmo di calcolo simbolico richiede in generale molto più tempo e molte più risorse di macchina se comparato con il suo simmetrico numerico; occorre però sottolineare come il confronto sia improprio in quanto la soluzione simbolica in genere rappresenta tutti gli stati possibili del sistema in studio e contiene quella numerica come caso particolare.

Un manipolatore algebrico può essere utile anche in molti casi non complicati, ma che affrontati solamente con carta e penna richiederebbero molto tempo e sarebbero difficili da ripulire dagli errori; in tali casi l'aiuto di un pacchetto di calcolo simbolico costituisce un metodo evidentemente più veloce e sicuro.
Molti Autori sviluppano i propri modelli all'interno di ambienti di manipolazione algebrica per generale praticità, anche senza utilizzare le capacità di analisi simbolica proprie del sistema; spesso ciò viene fatto perché tali ambienti dispongono sia di vaste librerie di funzioni in grado di affrontare i più diffusi problemi di analisi numerica che di molte funzioni per la rappresentazione grafica dei dati. I programmi più aggiornati permettono anche di riunire coerentemente, all'interno di un unico documento, grafici, formule, tabelle, disegni e calcoli in forma sia numerica che simbolica [28, 35]. É stato condotto anche un certo numero di esperienze al fine di valutare l'efficacia di tali programmi nel preparare il materiale didattico necessario per l'insegnamento della matematica sia per agli studenti dei corsi di base che per quelli dei corsi universitari [13, 33].

Spesso un'analisi puramente simbolica non riesce a rispondere completamente a tutte le necessità; le esperienze maturate da molti Autori hanno mostrato come un approccio semi-analitico, che combini le caratteristiche dell'elaborazione numerica e di quella simbolica, è la strategia generalmente più vantaggiosa. Per esempio nel campo dello studio dei meccanismi articolati risulta conveniente ricavare prima le equazioni differenziali del moto del sistema con il manipolatore simbolico e risolverle poi per via numerica. Invece nel campo della meccanica del corpo deformabile si impiegano i manipolatori simbolici per ricavare le matrici di rigidezza di elementi con un numero elevato di gradi di libertà (con particolare vantaggio nel campo dei grandi spostamenti e/o dei materiali anisotropi) per procedere poi in modo tradizionale applicando la tecnica numerica dell'elemento finito [22]. In entrambi i casi citati, per applicazioni realistiche, si perviene frequentemente a dover manipolare espressioni simboliche della lunghezza di qualche pagina di testo, e quindi non trattabili in modo non assistito.

La impossibilità di ricavare la soluzione in forma chiusa per determinati problemi non deve far pensare che tali problemi possano essere affrontati solo con metodi numerici. E' possibile procedere con il metodo simbolico attraverso l'analisi delle soluzioni particolari (per esempio separando lo studio della fase transitoria da quello della fase stazionaria), oppure ancora ricorrendo ad approssimazioni simboliche dei modelli non lineari con sviluppi in serie.

In particolare, nel caso dello studio del comportamento di un sistema meccanico, possiamo ottenere molte informazioni anche senza una risoluzione completa delle equazioni differenziali del moto, ma attraverso la sola espressione degli autovalori ed autovettori principali ottenuti per via simbolica.

I codici per la manipolazione simbolica giungono in un momento di maturità di quei sistemi nati dagli studi dell'*intelligenza artificiale* che fanno della *rappresentazione della conoscenza* il loro modello di riferimento; la semantica di alcuni di tali codici risente infatti fortemente dei più recenti paradigmi di programmazione. Dal punto di vista dell'implementazione, un sistema di manipolazione simbolica può pensarsi costruito come una base di conoscenza (generalmente espandibile dall'utilizzatore) istruita con le regole formali dell'algebra, della trigonometria e del calcolo differenziale. A partire da un nucleo fondamentale di tali conoscenze, il sistema dispone poi della capacità inferenziale necessaria per estendere il set di regole a casi più complessi attraverso semplificazioni, scomposizioni e sostituzioni.

In aggiunta ai tradizionali stili di programmazione *procedurale* e *funzionale*, alcuni sistemi di calcolo simbolico permettono la cosiddetta programmazione *a regole*, ovvero la possibilità di costruire un universo formato da oggetti caratterizzati da una definizione e da una serie di operazioni elementari con le relative proprietà [19]. Per esempio oltre alla classica algebra dei numeri reali è possibile costruirsi una propria algebra dei vettori o delle matrici. In particolare è possibile istruire il sistema sulle proprietà di determinati oggetti, come per esempio l'unità immaginaria I o come ∞, i quali divengono simboli con pari dignità di 5 e 7 e vengono trattati correttamente secondo le regole formali fornite dallo sviluppatore. Così l'operazione 1/0 diviene ammissibile e fornisce ∞, similmente a $\sqrt{-1}$ che fornisce I [3].

Attraverso una opportuna codifica della struttura dei dati, il calcolatore è in grado di trattare dunque simboli astratti che rappresentano quantità numeriche, in questo modo si possono sviluppare i passaggi algebrici con precisione infinita mantenendo i numeri razionali in forma di frazione (p.e. 10/3), gli irrazionali in forma di radice ed i numeri trascendenti in forma simbolica (p.e. π o Sin(7.5)).

Di alcuni dei temi fin qui accennati si è trattato diffusamente al seminario *Computerized Symbolic Manipulation in Mechanics* tenuto presso il *Centre International des Sciences Mecaniques* di Udine nel Luglio del '93, ove sono stati illustrati argomenti specifici legati alla applicazione del metodo simbolico automatico agli studi di meccanica, ed in particolare sono stati discussi alcuni algoritmi di *Computer Algebra* (F. Schwarz [1]), sono state proposte alcune applicazioni nel campo della meccanica strutturale (A. K. Noor [2]) ed in quello della dinamica dei sistemi di corpi rigidi (E. Kreuzer [3]), anche nel caso di oscillazioni non lineari (W. Kleczca [4], R. H. Rand [4]).

11. stato dell'arte e sistemi attuali di manipolazione simbolica

Attualmente il numero di utenti di informazioni ed elaborazioni non numeriche è in forte aumento, si pensi per esempio alla diffusione dei sistemi di elaborazione dei testi, od ai programmi di gestione di grandi basi di dati o alle applicazioni dei sistemi esperti [6, 10]. In particolare, per quanto attiene alla Computer Algebra, la letteratura scientifica presenta ormai moltissimi esempi di applicazioni nel campo della meccanica strutturale (costruzione delle equazioni differenziali della configurazione deformata dei sistemi elastici, applicazioni della tecnica di Rayleigh-Ritz, generazione delle matrici delle rigidezze per analisi agli elementi finiti, sviluppo di schemi di risoluzione alle differenze finite [4, 9, 23]), nel campo della meccanica del corpo rigido (generazione delle equazioni del moto di sistemi meccanici olonomi ed anolonomi [5, 7, 15, 17, 18], nello studio delle singolarità dei problemi di equilibrio [26, 32]), e nello studio delle equazioni dei sistemi di controllo in retroazione [2].

[1] *Gesellshaft für Matematik und Datenverarbeitung*, St. Augustin, Germany.

[2] *University of Virginia*, Hampton, Va, USA.

[3] *Technische Universität Hamburg-Harburg*, Germany.

[4] *Cornell University*, Ithaca, NY, USA.

Il primo esempio documentato di sistema di manipolazione simbolica, per la derivazione formale, risale ai primi anni '50 [21]; come questo, i primi codici scritti per il trattamento simbolico delle espressioni algebriche sono stati guidati da problemi specifici, come quelli classici di meccanica del corpo rigido. Attualmente esistono invece una quantità di programmi di uso generale, attrezzati con amplissime librerie di funzioni in grado di affrontare molti dei tipici problemi di matematica; tali sistemi sono pensati per essere usati come degli abili assistenti matematici in grado sia di affrontare problemi sostanzialmente meccanici come lo sviluppo e la semplificazione delle espressioni algebriche sia problemi come l'integrazione di funzioni che richiedono un discreto livello di esperienza.
Lo sviluppo dei primi sistemi (REDUCE e MACSYMA, entrambi scritti in linguaggio *lisp*) cominciò negli anni '60 e questi pacchetti di uso generale furono anche i primi sistemi interattivi ad essere commercializzati; in seguito lo sviluppo della tecnologia hardware ed il forte decremento dei costi ha sostenuto lo sviluppo di pacchetti sviluppati in codice C e C++ come MAPLE e MATHEMATICA. Esistono poi in commercio altri pacchetti come MATLAB, MATHCAD, EUREKA! e TKSOLVER che sono provvisti di qualche capacità di analisi simbolica pur senza essere dei veri sistemi di manipolazione simbolica.

I pacchetti menzionati dispongono generalmente di una varietà di strumenti e possibilità accessorie, sia per quanto riguarda le capacità di analisi simbolica sia quelle numeriche; l'elenco che segue riporta in forma schematica una serie di capacità che allo stato attuale possono essere ritenute degli standard minimi per i pacchetti d'uso generale:

Algebra: Aritmetica complessa; aritmetica floating-point esatta; funzioni trigonometriche ed iperboliche e le loro inverse; espansione e semplificazione delle espressioni; soluzione dei sistemi di equazioni non lineari ed eliminazione delle variabili; calcolo del determinante; inversione; triangolarizzazione.

Analisi: Differenziazione; derivazione parziale di funzioni incognite; limiti; sviluppi in serie di potenze; integrali definiti ed indefiniti; trasformazioni di Laplace e di Fourier; approssimazione ed interpolazione dei dati.

Calcolo tensoriale e vettoriale.

Grafica: Diagrammi polari e cartesiani in due e tre dimensioni; tracciamento di curve di livello; *shading* delle superfici; *hardcopies*; animazione.

Utility: Programmabilità; output in FORTRAN o C; interfaccia con programmi *custom*; output per word processor; generazione di documenti in grado di includere contemporaneamente testo, espressioni matematiche e grafica.

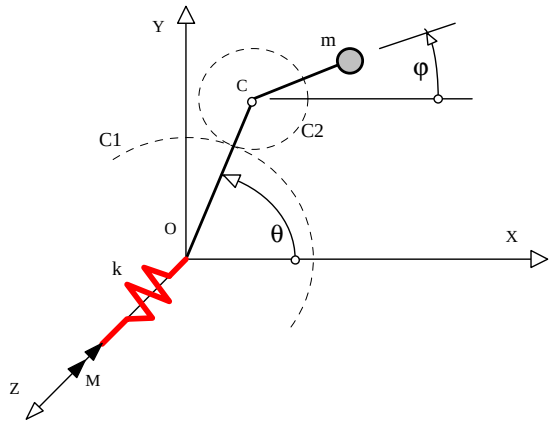
L'esperienza di molti sviluppatori ha mostrato come il *software* di uso generale (p.e. MATHEMATICA e MAPLE) risulti meno efficiente se confrontato con *software* sviluppato per una specifica applicazione, sia in termini di occupazione dello spazio di memoria sia in termini di velocità di esecuzione. Fa parte dell'esperienza ordinaria di coloro che utilizzano gli attuali pacchetti commerciali il fatto di trovarsi in condizioni tali da non poter terminare un processo di calcolo a causa della mancanza di memoria libera in macchina; inoltre risulta in generale impossibile prevedere con adeguata precisione le necessità di memoria per una determinata applicazione. Occorre comunque sottolineare come spesso un problema risulti semplice solo in apparenza, per esempio [34] chiedendo al sistema di calcolare tutti gli autovalori di una matrice di ordine n, il programma dovrà in generale determinare gli zeri di un polinomio di grado n; se n>3 in generale la soluzione analitica risulta impossibile, ovvero è impossibile determinare la forma algebrica esplicita degli autovalori di una matrice qualsiasi, ma è possibile solo per quelle più semplici di tipo sparso.
Attualmente l'efficienza computazionale dei processori tende a diventare sempre meno critica e il costo delle memorie dinamiche e di massa è in costante calo, dunque diviene sempre più ragionevole pensare di utilizzare un codice *general purpose* piuttosto che sviluppare od acquistare un programma specifico.

Nella tabella che segue si riporta un elenco dei principali pacchetti di uso generale per la manipolazione simbolica delle espressioni matematiche rimandando per una valutazione più puntuale alla bibliografia indicata [1, 8, 11, 25, 30, 31]:

Sistema	Autore	Codice	Piattaforma
Axiom	Numerical Algorithms Group Inc., Downers Grove, IL, USA	LISP	SunSparc, IBM RS6000
Derive	Soft Warehouse Inc., Honolulu, HI, USA	LISP	MS-DOS
FORM	CAN, Amsterdam, The Netherlands	_	MS-DOS, Amiga, Macintosh, Sun, Apollo, IBM RISC
Macsyma	Macsyma Inc., Arlington, VA, USA	LISP	Sun-3, Sun (SPARC), Apollo, VAX, HP 9000, DEC RISC, MS-DOS 386/387
Maple V	Waterloo Maple Software, Waterloo, Ontario, Canada	C	Unix WKS(Sun, HP, DEC, IBM, Apollo, Silicon Graphics), MS-DOS, Macintosh
Mathematica	Wolfram Research Inc., Champaign, IL, USA	C++	Macintosh, MS-DOS 386, Unix WKS(Sun, HP, DEC, IBM, Apollo, Silicon Graphics)
MuPad	University of Paderborn, Germany	_	Unix WKS(Sun, IBM)
Reduce	Rand, Santa Monica, CA, USA	LISP	Cray Y-MP, MS-DOS, Macintosh
SENAC	University of Waikato, New Zeland	_	Sun-3, Sun (SPARC), VAX, MS-DOS
Theorist	Prescience Corporation, San Francisco, CA, USA	_	Macintosh

12. analisi dinamica di un semplice sistema meccanico ad un grado di libertà

Un ambito nel quale la manipolazione simbolica delle espressioni ha avuto vasta applicazione è quello della generazione delle equazioni del moto di un sistema meccanico formato dall'assemblaggio di diversi corpi rigidi [27, 29]; nel seguito si presenta un semplice esempio di computerizzazione delle tecniche di meccanica analitica applicate all'analisi di un dispositivo impiegato come scuotitore su di una macchina per la raccolta meccanica del pomodoro da industria [5]. La procedura illustrata è congegnata in modo tale da fornire un esempio semplice ma significativo sia delle capacità di calcolo simbolico e numerico, sia delle capacità grafiche, sia soprattutto della praticità d'uso di un tipico ambiente di manipolazione matematica.
In figura è rappresentato lo schema cinematico (molto semplificato) del meccanismo:



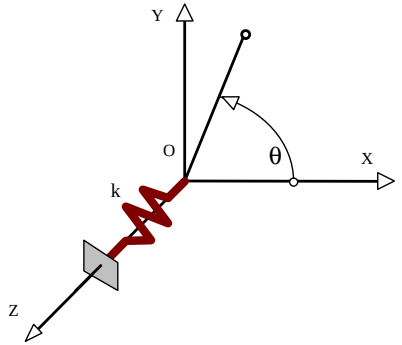
[5] É attualmente in corso una ricerca più dettagliata sulla specifica analisi dinamica di tale meccanismo.

L'articolazione a doppio pendolo, vincolata al telaio della macchina con una coppia rotoidale in O ed incernierata internamente in C, permette alla massa puntiforme m di muoversi solo sul piano x-y. Il membro individuato dal simbolo k rappresenta un elemento elastico, dotato di sola rigidezza torsionale, concentrato idealmente nel punto O.
Il reale cinematismo della macchina è costruito poi in modo da legare linearmente l'angolo φ all'angolo θ, ovvero l'asta Cm si muove come se fosse solidale ad un disco (C2 disegnato a tratteggio) il quale rotolando senza strisciare su di una guida circolare (C1 con centro in O) genera la traiettoria cicloidale piana della massa puntiforme m.
Se con il parametro h si indica il rapporto tra il raggio della circonferenza C2 e quello della circonferenza C1, allora il vincolo cinematico tra gli angoli φ e θ è espresso dalla semplice relazione lineare:

$$\varphi = h \cdot \vartheta \qquad h \in R$$

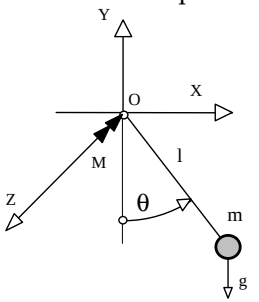
dunque al variare del tempo t, ovvero dell'unico grado di libertà θ(t), la massa m descrive una traiettoria cicloidale piana; in particolare tale traiettoria risulterà epicicloidale od ipocicloidale a seconda che la costante di proporzionalità h sia rispettivamente positiva o negativa [6].

Nel meccanismo reale viene applicato un momento M (in prima approssimazione costante nel tempo) all'estremità libera dell'elemento elastico di rigidezza torsionale k; mentre invece questa particolare analisi verrà condotta, per semplicità, ipotizzando che la rotazione dell'estremo libero della molla di torsione venga bloccata per mezzo di un vincolo ad incastro sul telaio della macchina.



[6] Si conviene di porre h < 0 quando C2 risulta interna a C1.

Tale semplificazione viene introdotta al fine di rendere più evidente il significato delle diverse equazioni e comunque si può dimostrare che, ai fini dello studio della dinamica del sistema, la legge del moto risultante differisce da quella reale per una sola costante additiva.
Senza impiegare troppo spazio per dimostrarlo ci si può rendere conto intuitivamente di ciò pensando al caso semplice del pendolo matematico:



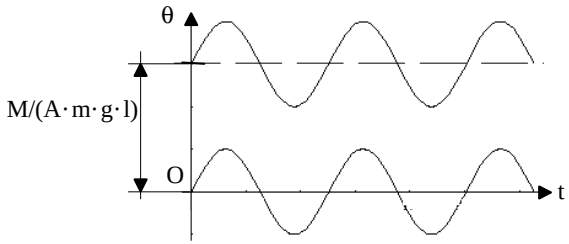
la classica soluzione linearizzata, valida nel caso di oscillazioni di *piccola* ampiezza, fornisce:

$$\vartheta(t) = A \cdot Sin(B \cdot t) \quad A, B \in R$$

mentre se al perno del pendolo (punto O) venisse applicata una coppia costante M (con M << m·g·l), si otterrebbe:

$$\vartheta(t) = A \cdot Sin(B \cdot t) + \frac{M}{A \cdot m \cdot g \cdot l}$$

dove m, g ed l rappresentano rispettivamente la massa puntiforme pendolare, l'accelerazione di gravità e la lunghezza di sospensione del pendolo. Dal punto di vista fisico si osserva cioè che il centro delle oscillazioni si sposta dalla posizione θ=0 alla posizione θ=M/(A·m·g·l), in particolare però si vede come le due leggi del moto differiscano solo per un termine costante e quindi esprimano le stesse velocità ed accelerazioni.



Coerentemente con la procedura manuale, anche con un sistema di manipolazione simbolica, generalmente si inizia ad impostare un problema di modellazione matematica definendo i gradi di libertà del sistema. L'espressione che segue (in carattere diverso da quello del testo normale) comunica al programma che esiste una variabile, reale o complessa, che indico con il simbolo t, ed una funzione incognita θ, della quale sappiamo solo che è funzione di t:

```
(*     Meccanismo Scuotitore*)
ClearAll["@"];Share[];
Hold[θ[t]];
```

si definiscono poi alcuni altri simboli e si esplicita il fatto che siano da considerare come costanti:

```
SetAttributes[{r,d,m,h,k,θ0},Constant]
```

r e d rappresentano le lunghezze degli elementi ad asta Cm e CO, considerati privi di massa ed infinitamente rigidi, ed m è la massa concentrata all'estremità dell'asta. I simboli restanti verranno impiegati nel seguito risultando di significato evidente.

Attraverso l'operatore di assegnamento =, si istituisce poi il legame lineare tra gli angoli θ e φ secondo quanto si è già detto:

```
φ= h·θ[t];
```

Poiché si determineranno le equazioni del moto attraverso il metodo delle equazioni di Lagrange [12] occorre ora calcolare l'espressione generale dell'energia cinetica e potenziale del sistema, a tal fine si scrivono in modo parametrico le coordinate della massa m:

```
x=d·Cos[θ[t]]+r·Cos[φ];y=d·Sin[θ[t]]+r·Sin[φ];
```

Per calcolare l'energia cinetica si calcolano le componenti cartesiane ed il modulo del vettore velocità applicato al punto m attraverso l'operatore *derivata totale, Dt[]*:

```
vx = Dt[x,t];  vy = Dt[y,t];
vq = Simplify[vx^2 + vy^2];
```

ottenendo:

```
vq = (d²+h²·r²+2·d·h·r·Cos[(1-h)·θ[t]])·θ'²[t]
```

L'espressione dell'energia cinetica risulta allora:

```
T = m/2·vq
```

e quella del potenziale delle forze attive agenti sul sistema risulta:

```
U = -k/2·θ[t]²
```

Secondo l'usuale simbologia della meccanica analitica, si chiede poi al sistema di eseguire la seguente serie di operazioni:

$$\frac{d}{dt}\left[\frac{\partial}{\partial\dot{\vartheta}}(T+U)\right]-\frac{\partial}{\partial\vartheta}(T+U)=0$$

ovvero, attraverso l'operatore simbolico *derivata parziale D[]*, si scrivono (e si semplificano) l'espressione della Lagrangiana e dell'equazione del moto del sistema:

```
Lag = Simplify[T+U]
eqdiffmoto=Simplify[Dt[D[Lag,θ'[t]],t]-D[Lag,θ[t]]==0]
```

ed il sistema risponde restituendo in pochi secondi l'espressione (ordinata e semplificata) dell'equazione differenziale del moto del sistema scuotitore:

```
eqdiffmoto=
k·θ[t]-2·d·(1-h)·h·m·r·Sin[(1-h)·θ[t]]·θ'²[t]+
+2·m·(d²+h²·r²+2·d·h·r·Cos[(1-h)·θ[t]])·θ''[t] == 0
```

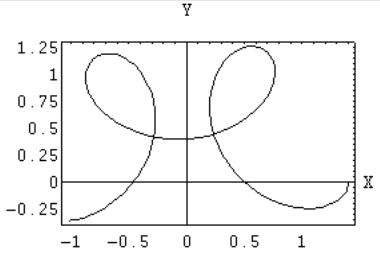
Per rendersi conto della funzionalità della metodologia occorre notare come i passaggi algebrici sintetizzati in queste poche righe di programma, pur risultando elementari, se sviluppati a mano avrebbero richiesto almeno un'ora di tempo e certamente molta attenzione.
Si sottolinea inoltre come l'uso di un sistema di manipolazione simbolica per questo caso semplice (e specialmente lo sforzo richiesto per impararne la sintassi) sia forse eccessivo, ma occorre pensare che un modello anche un poco più realistico del meccanismo potrebbe prevedere più di una massa, effetti dissipativi (attrito statico o dipendente dalla velocità), masse distribuite in luogo di masse puntiformi ed un numero di gradi di libertà superiore ad uno, risultando praticamente impossibile ricavare le equazioni del moto con procedimento non assistito.

Desiderando avere una rappresentazione grafica del moto del cinematismo occorre definire le condizioni iniziali ed i valori dei diversi parametri (tutti i valori sono indicati in unità S.I.). Si ipotizza che all'istante t=0 la massa m sia ferma nella posizione x(θ0),y(θ0):

```
condcont={eqdiffmoto,θ[0]==θ0,θ'[0]==0};
r=.5; d=.9; m=1; h=-4.8; Δt=3; θ0=0.5; k=150;
numsol=NDSolve[condcont,θ[t],{t,0,Δt}];
```

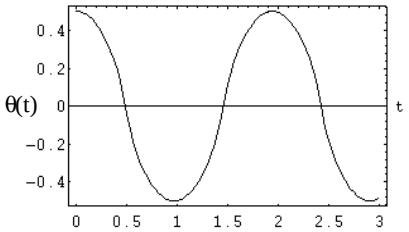
La funzione *NDSolve[]* fornisce la soluzione numerica di un sistema di equazioni differenziali alle derivate ordinarie. Nota la soluzione θ(t) possiamo tracciare la traiettoria seguita dalla massa m nei primi 3 secondi del moto:

```
ParametricPlot[{x/.θ[t]->a,y/.θ[t]->a},{a,0,Δt},  Frame-
>True, AxesLabel->{"X","Y"}];
```

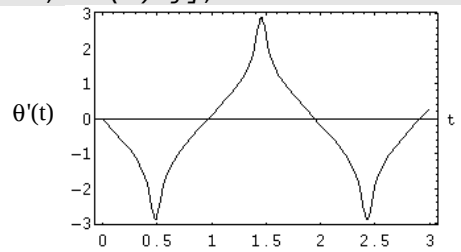


Si chiede poi al sistema di tracciare gli andamenti di θ(t), θ'(t) e θ''(t) con t=0÷3s:

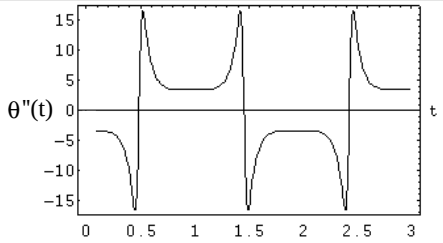
```
angolo = θ[t]/.numsol;
Plot[Evaluate[angolo],{t,0,Δt},Frame->True,   AxesLabel-
>{"t","θ(t)"}];
```



```
velangolare = Dt[dummy0,t];
Plot[Evaluate[velangolare],{t,0,Δt},        Frame->True,
AxesLabel->{"t","θ'(t)"}];
```




```
accelangolare = Dt[dummy1,t];
Plot[Evaluate[accelangolare],{t,0,Δt},Frame->True,
AxesLabel->{"t","θ''(t)"}];
```



É interessante notare come attraverso l'operatore di assegnamento il sistema di calcolo, ad ogni passaggio ed in modo trasparente per l'utilizzatore, si faccia carico di trasformare le informazioni nei modi più opportuni così da integrare con apparente semplicità dati in forma grafica, numerica e simbolica.

Come ultima applicazione si studiano le piccole oscillazioni del sistema attorno alla posizione di equilibrio. Per fare questo occorre linearizzare le espressioni di T ed U nell'intorno della posizione θ=0:

```
Uappr = -k/2·θ²
Tappr = m/2·[(d+h·r)·θ']²
```

e procedendo come già visto si ricava:

```
θ'' = -k/[m·(d+h·r)²]·θ
```


che è la nota equazione dell'oscillatore armonico monodimensionale citata nel primo capitolo e dunque si può affermare che uno dei periodi fondamentali del sistema meccanico in studio vale:

$$T = 2\pi \cdot (d + h \cdot r) \cdot \sqrt{\frac{m}{k}}$$

Con un impegno minimo rispetto alla procedura manuale è evidentemente possibile verificare le reazioni del sistema al variare dei diversi parametri; per esempio si può ripetere il calcolo con un valore positivo del coefficiente h (ciò ha il significato di rendere concordi le rotazioni dei due elementi ad asta) e magari rappresentare su di un unico grafico diverse curve relative a diverse configurazioni del meccanismo.
La versatilità dell'ambiente di calcolo permette poi di sviluppare lo studio in molte direzioni: si potrebbe sviluppare in serie di Fourier la soluzione numerica θ[t] per approssimare una soluzione analitica semplice in termini di multipli della pulsazione propria fondamentale del sistema, oppure ancora si potrebbe esplicitare θ''[t] e, per derivazione, cercare le condizioni in corrispondenza delle quali si ottengono i picchi massimi dell'accelerazione, che è il parametro chiave per quanto riguarda l'efficacia dell'operazione di distacco della bacca dalla pianta.

13. bibliografia

1) ABBOTT, P., *Maple V and Mathematica*, Notices of the American Mathematical Society (1992), 39 (8).

2) ANDREWS, G. C., *A. Brief Survey of Self-Formulating Simulation Programs for Multibody Dynamic Systems*, in: Prof. IASTED Intern. Symp. Simulation and Modeling (1985).

3) BUCHBERGER, B., COLLINS, G.E., AND LOOS, R. (eds.), *Computer Algebra - Symbolic and Algebraic Computation* (1982), Springer-Verlag, New York.

4) CHUGH, A.K. AND GESHUND H., *Automatic Generation of the Coefficient Matrix of Finite Difference Equations*, International Journal of Numerical Methods in Engineering (1974), 662-670.

5) DABERKOW, A., *Zur CAD-gestützen Modellierung von Mehrkörpersystemen*. Fortschritt-Berichte der VDI-Zeitschriften (1993), 20, 80, VDI-Verlag, Düsseldorf.

6) DAVIS, M.S., *Analytical Mathematics on Computers*. Applied Mechanics Reviews, (1978), 31, 1-9.

7) DUFFEK, W., FURER, C., SCHWARTZ, W., AND WALLRAPP, O., *Analysis and Simulation of Rail and Road Vehicles with the program MEDYNA*. In: Proc. 9th IAVSD-Symposium on the Dynamics of Vehicles on Roads and on Tracks, (1986), Nortdström, O. (ed.), Lisse: Swets and Zeitlinger, 71-85.

8) FATEMAN, R.J., *A Review of Macsyma*, IEEE Transactions on Knowledge and Data Engineering, 1, (1), 1989.

9) FISETTE, P., SAMIN, J.C., WILLEMS, P.Y., *Contribution To Symbolic Analysis of Deformable Multi-body System*, International Journal of Numerical Methods in Engineering (1991), 1621-1635.

10) FITCH, J. *Mathematics goes automatic*. Physics World, (1993), 48-52.

11) FOSTER, K.R. AND BAU, H.H. (1989), *Symbolic Manipulation Programs for the Personal Computer*, Science, 243, 3.

12) GRAFFI, D. *Elementi di Meccanica Razionale* (1982), Patron Editore, Bologna.

13) GRAY, T. AND JERRY, G. (1991), *Exploring Mathematics with Mathematica*. Redwood City/...: Addison-Wesley Pub. Comp.

14) GROSSMANN, R. (ed.) (1989), *Symbolic computation: application to scientific computating*. Philadelphia: Soc. for Ind. and Appl. Math. (SIAM).

15) INSTITUTE OF MECHATRONICS, *Advanced Lagrangian Solver in Kinetic Analysis for Multibody System Dynamics Simulation Software*, Program Manual, V2.0 1993, Reichenhainer Str.88, Chemnitz, Germany.

16) KAHRIMANIAN, H.G. (1954), *Analytical Differentiation by Digital Computer*. Sympos. on Automatic Programm. for Digital Comput., Off. of Naval Res., Dept. of the Navy, 6-14.

17) KORTÜM, W. AND SCHIELEN, W. (1985), *General Purpose Vehicle System Dynamics Software Based on Multibody Formalisms*. Vehicle System Dynamics, 14, 229-263.

18) KREUZER, E. AND SCHIELEN, W. (1990), *NEWEUL-Software for the Generation of Symbolical Equations of Motion*. In: Multibody Systems Handbook, Schielen, W. (ed.), Berlin /...:Springer-Verlag, 181-202.

19) MAEDER, R. (1989), *Programming in Mathematica*. Redwood City/...: Addison-Wesley Pub. Comp.

20) MONEGATO, G. (1985), *Calcolo Numerico*. Levrotto e Bella, Torino.

21) NOLAN, J. (1953), *Analytical Differentiation on a Digital Computer*. Cambridge, Mass.: Inst. of Technol. (MIT), M.A. Thesis.

22) NOOR, A.K. AND ANDERSEN, C. M. (1979), *Computerized Symbolic Manipulation in Structural Mechanics - Progress and Potential*. Computer and Structures, 18, 95-118.

23) NOOR, A.K., ELISHAKOFF, I. (1990), *Symbolic Computations and Their Impact on Mechanics - Proc. of the Winter Annual Meeting of the American Society of Mechanical Engineers*. A.S.M.E. New York.

24) PAVELLE, R., ROTHSTEIN, M., AND FITCH, J. (1981), *Computer Algebra*. Scientific American, 136-152.

25) RAND, R. H. (1984), *Computer Algebra in Applied Mathematics: An Introduction to MACSYMA*. Pitman.

26) RAND, R. H. AND ARMBRUSTER, D. (1987), *Perturbation Methods, Bifurcation Theory and Computer Algebra*. New York/...:Springer-Verlag.

27) ROSENTHAL, D.E. AND SHERMAN, M.A. (1986), *High performance multibody simulation via symbolic equation manipulation and Kane's method*. J. of Astronautical Sciences, 34, 223-239.

28) SCARASCIA MUGNOZZA, G., RUSSO, G., VOX, G. (1992), *Modello numerico per l'analisi previsionale delle temperature nel terreno agrario solarizzato*. Atti del convegno Informatica ed Agricoltura, Accademia dei Georgofili, Firenze.

29) SCHIEHLEN, W. (ed.) (1990), *Multibody Systems Handbook*. Berlin/...: Springer-Verlag.

30) SIMON, B. (1992), *Comparative CAS Reviews*, Notices of the American Mathematical Society, 39, 7.

31) SIMON, B. (1992), *Symbolic Math Software - It's Not Just for Mainframes Anymore*, PC Magazine, August.

32) SUNADA, W.H., DUBOWSKY, S. (1982): *On the Dynamic Analysis and Behaviour of Industrial Robotic Manipulators With Elastic Members*. ASME Journal of Mechanical Design 1-10.

33) VVEDENSKY, D. (1991), *Partial Differential Equations with Mathematica*. Redwood City/...: Addison-Wesley Pub. Comp.

34) WOLFRAM, S. (1988), *Mathematica - A System for Doing Mathematics by Computer*. Redwood City/...: Addison-Wesley Pub. Comp.

35) YOUNG, J.H., STIKELEATHER, L.F., (1991), *Use of TK Solver in Agricultural Engineering Instruction*. Transactions of ASAE, 34, 301-306.

14. riassunto

Nell'ambito generale delle applicazioni dell'ingegneria alle scienze naturali, e dell'ingegneria agraria in particolare, la letteratura scientifica è testimone di una intensa attività nel campo delle ricerche teoriche. In tale ottica è sembrato opportuno proporre, con il presente lavoro, una rassegna sintetica sui più aggiornati *manipolatori algebrici* come strumenti *software* particolarmente adatti allo sviluppo di modelli matematici.
Nell'articolo vengono anche riassunti alcuni degli argomenti trattati ad un recente seminario organizzato dal *Centre International des Sciences Mecaniques* di Udine dal tema *Computerized Symbolic Manipulation in Mechanics*, in particolare vengono confrontati pregi e limiti degli approcci simbolico e numerico. Viene poi illustrata una rassegna dei principali pacchetti *software* per la manipolazione delle espressioni matematiche in forma simbolica attualmente disponibili sul mercato e delle loro applicazioni più notevoli nell'ambito dei modelli matematici di interesse per l'ingegneria meccanica. Infine un semplice esempio illustra le capacità tipiche di tali pacchetti evidenziando come i maggiori vantaggi possano derivare da un uso misto delle tecniche simboliche, numeriche e grafiche.

Modulo 3

15. Applicazioni (un po' più) avanzate di calcolo numerico: risoluzione approssimata di equazioni differenziali

Equazioni differenziali di primo grado: soluzione analitica, applicazioni fisiche, discretizzazione alle differenze finite.

Supponiamo di registrare con continuità la velocità (istantanea) di una motocicletta in un certo intervallo di tempo. Si disporrà così di una funzione $v(t)$ definita sull'intervallo $[t_a, t_b]$.
Si voglia ora conoscere lo spazio percorso durante tale intervallo di tempo.

A tal fine scriviamo la definizione di velocità:

$$v(t) = \frac{ds(t)}{dt} = s'(t)$$

è evidente che per conoscere lo spazio percorso s_{ab} nell'intervallo di tempo t_b-t_a, basta integrare nel tempo la funzione $v(t)$ tra gli estremi t_a e t_b (integrale definito):

$$S_{ab} = s(t_b) - s(t_a) = \int_{t_a}^{tb} s'(t)dt = \int_{t_a}^{tb} v(t)dt$$

Se invece desiderassimo conoscere l'equazione oraria della motocicletta $s=s(t)$, ovvero la curva integrale della $v(t)$, allora ci troveremmo di fronte ad un problema che possiamo assimilare a quello usuale della soluzione di un'equazione in una variabile, ma dove la quantità incognita non è più un singolo valore, ma una funzione.

La soluzione analitica di questo particolare problema è comunque immediata (integrale indefinito):

$$S(t) = \int_0^t v(t)dt + Cost.$$

Generalmente si adotta una notazione più comoda: supponendo che il nostro punto materiale si muova su di una traiettoria rettilinea, che facciamo corrispondere con l'asse x di un sistema di riferimento cartesiano, indicando con $x(t)$ la posizione del punto materiale all'istante t (ovvero la distanza dall'origine), l'equazione diviene:

$$\frac{dx(t)}{dt} = x'(t) = F(t)$$

essendo $F(t)$ la funzione del tempo che contiene i valori della velocità acquisiti ad ogni istante.

Come si è detto è nota $x'(t)$ ed è incognita la funzione $x(t)$. Un tale tipo di equazioni, dove la funzione incognita compare derivata una volta, si chiamano *equazioni differenziali di primo grado*.

Se, come accade in ambito sperimentale, immaginiamo di avere osservato il fenomeno non con continuità. In particolare, se della $F(t)$ è noto solo un *campionamento*, ovvero se i valori della velocità fossero stati acquisiti non con continuità, ma ad intervalli regolari Δt, allora la $F(t)$ sarebbe nota solo in corrispondenza di una serie di istanti t_a, $t_a+\Delta t$, $t_a+2\Delta t$, $t_a+3\Delta t$, $t_a+4\Delta t$,...., t_b.
Per tale successione di istanti si usa spesso anche una notazione indiciale abbreviata t_0, t_1, t_2, t_3, t_4..,t_i, .., t_N.

In tal caso una relazione tra spazio percorso e velocità istantanea è data dalla definizione di *velocità media* su ciascun intervallo finito di tempo Δt:

$$\bar{v}(t) = \frac{\Delta s}{\Delta t} = \frac{x(t+\Delta t) - x(t)}{\Delta t} = F(t)$$

Dove i valori $x(t)$ sono sconosciuti, mentre sono noti Δt ed $F(t)$. Riordinando tale l'espressione se ne ottiene una forma ricorsiva, dove il valore di x in un istante futuro è funzione di quello presente:

$$x(t+\Delta t) = F(t)\cdot\Delta t + x(t)$$

Adottando la notazione indiciale abbreviata t_0, t_1, t_2, t_3, t_4..,t_i, .., t_N; per gli istanti successivi t_a, $t_a+\Delta t$, $t_a+2\Delta t$, $t_a+3\Delta t$, $t_a+4\Delta t$,...., t_b, l'espressione ricorsiva diviene:

$$x(t_{i+1}) = F(t_i)\cdot\Delta t + x(t_i)$$

o anche:

$$x_{i+1} = F_i\cdot\Delta t + x_i$$

la sequenza delle x_i, costituisce una rappresentazione *per punti* della soluzione del problema, rappresentata dall'equazione oraria del moto della nostra motocicletta (o punto materiale) $x(t)$:

i	t	F(t)	x(t)	x(t+Δt)
0	0	F(0)	x(0)	F(0)·Δt+x(0)
1	Δt	F(Δt)	x(Δt)	F(Δt)·Δt+x(Δt)
2	2 Δt	F(2Δt)	x(2Δt)	F(2Δt)·Δt+x(2Δt)
3	3 Δt	F(3Δt)	x(3Δt)	F(3Δt)·Δt+x(3Δt)
4	4 Δt	F(4Δt)	x(4Δt)	F(4Δt)·Δt+x(4Δt)
.. .	...	...	...	...

E' evidente che per rendere determinata la soluzione del problema occorre conoscere la posizione iniziale $x(0)$.

Una forma un poco più generale di un'equazione differenziale di primo grado è del tipo:

$$\boxed{\frac{dx(t)}{dt} = x'(t) = F(t, x(t))}$$

è evidente che in questo caso non è più praticabile la via dell'integrazione immediata:

$$x(t) \overset{?}{=} \int_{x_1}^{x_2} F(t, x(t))dt$$

mentre rimane valida invece la soluzione in termini di rapporti incrementali (o come si dice più tecnicamente in contrapposizione alle quantità differenziali dell'analisi matematica, in termini di **differenze finite**):

$$x(t_{i+1}) = F(t_i, x(t_i)) \cdot \Delta t + x(t_i)$$

o anche, come è già stato mostrato, in notazione più sintetica:

$$\boxed{x_{i+1} = F_i \cdot \Delta t + x_i}$$

Ovvero, anche nel caso in cui il problema sia continuo, la soluzione in termini numerici può essere cercata supponendo di dividere l'intervallo sul quale è definita la variabile indipendente t, in un numero N di parti uguali: indicando con t_0, t_1, t_2, ..., t_N i punti di suddivisione dell'intervallo (nodi), posta l'ampiezza di ciascun sottointervallo $\Delta t=(t_N-t_0)/N$, l'equazione alle differenze finite, in grado di approssimare l'equazione differenziale e la sua soluzione, diviene:

$$x'(t) = F(t, x(t)) \rightarrow \frac{x(t+\Delta t) - x(t)}{\Delta t} = F(t, x(t)) \rightarrow$$
$$x(t+\Delta t) = x(t) + \Delta t \cdot F(t, x(t))$$

e la serie che costituisce la soluzione numerica del problema, a partire dalla conoscenza della posizione iniziale x_0, è:

$$x_1 = x_0 + \Delta t \cdot F(t_0, x_0)$$
$$x_2 = x_1 + \Delta t \cdot F(t_1, x_1)$$
$$x_3 = x_2 + \Delta t \cdot F(t_2, x_2)$$
$$....$$
$$x_n = x_{n-1} + \Delta t \cdot F(t_{n-1}, x_{n-1})$$

tale serie, dal punto di vista geometrico, rappresenta una serie di punti, o conseguentemente una linea spezzata, di vertici $P_i(t_i, x_i)$:

$$t_{i+1} = t_i + \Delta t \quad e \quad x_{i+1} = x_i + \Delta t \cdot F(t_i, x_i)$$

Tale spezzata approssima la curva primitiva (o integrale) richiesta, ed il generico segmentino P_i-P_{i+1}, ha come coefficiente angolare il valore approssimato della $x'(t) = F(t, x(t))$.

Questo metodo semplice di soluzione delle equazioni differenziali è dovuto ad **Eulero**, ed è detto di primo grado in quanto fornisce una soluzione esatta nel caso che questa sia lineare. E' evidente che più piccolo è l'intervallo Δt è più la spezzata sarà vicina alla soluzione reale del problema.

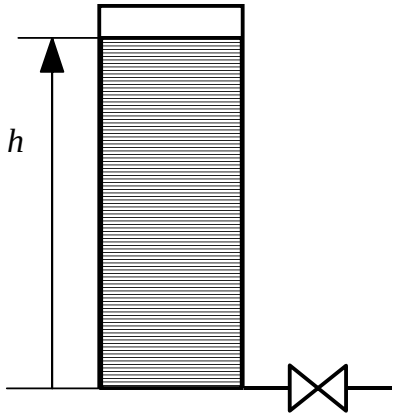
Ancora una volta vediamo come il metodo numerico nasca dalla sostituzione della soluzione incognita con una sua approssimazione semplice. In particolare la soluzione di una equazione differenziale può essere approssimata, oltre che da un segmentino di retta, anche da un polinomio di grado 2 o 3, pervenendo ai metodi (più stabili ma più laboriosi, dal punto di vista computazionale) di Runge-Kutta.

http://it.wikipedia.org/wiki/Metodi_di_soluzione_numerica_per_equazioni_differenziali_ordinarie

www.comsol.com

Esempio 1 : tempo di svuotamento di un serbatoio

Consideriamo ora come applicazione un serbatoio cilindrico ad asse verticale, con area della sezione trasversale pari ad Ω_s, riempito di un liquido incomprimibile con densità ρ fino alla quota h_0. Il serbatoio è munito alla base di un tubo per lo svuotamento con area della sezione trasversale pari ad Ω_t. Si vuole calcolare il tempo necessario ad un completo svuotamento.



La velocità w con cui il fluido defluisce dal tubo di uscita può essere facilmente ricavata applicando il teorema di Bernoulli:

$$\frac{w^2}{2} = g \cdot h \rightarrow w = \sqrt{2g \cdot h} \rightarrow w = \alpha\sqrt{2g \cdot h}$$

il coefficiente α *può* essere introdotto per tenere conto, in prima approssimazione, delle perdite per attrito al fine di rendere più realistica la soluzione del problema: $\alpha \in [0, 1]$, $\alpha=1$ fluido perfetto di Bernoulli, $\alpha=0$ fluido a viscosità infinita.
La portata volumetrica Q che sfugge attraverso il tubo di uscita risulta evidentemente:

$$Q = w \cdot \Omega_t = \Omega_t\sqrt{2g \cdot h}$$

Approssimando la portata istantanea con quella media su un piccolo intervallo di tempo Δt, possiamo dire che in tale intervallo è uscito dal tubo un volume d'acqua $\Delta V = Q \cdot \Delta t$, e dunque un identico volume di liquido deve mancare dal serbatoio, dove la quota del pelo libero risulterà diminuita della quantità Δh tale che $\Delta V = \Delta h \cdot \Omega_s$. A questo punto, trascorso l'intervallo di tempo Δt, essendo diminuita la quota h, risulterà diminuita anche la velocità con la quale il fluido esce dal tubo, e dunque la portata deve essere ricalcolata.

Lo schema di calcolo iterativo per risolvere il problema risulta dunque il seguente:

a) $t = 0$; $h = h_0$
b) $Q = \Omega_t\sqrt{2g \cdot h}$
c) $\Delta V = Q \cdot \Delta t$
d) $\Delta h = \Delta V / \Omega_s$
e) $h \leftarrow h - \Delta h$
f) Stampa t, h
g) $t \leftarrow t + \Delta t$
h) Se $h > 0$ torna al punto *b; altrimenti END.*

```
' Calcolo del tempo di svuotamento di un serbatoio per gravita'

H0 = 10          ' m
DeltaT = .1      ' s
AreaSerb = 1     ' m^2
AreaTubo = .1    ' m^2
g = 9.816        ' m/t^2
Alfa = .8

H = H0

DO WHILE (H > 0)
   PRINT "t="; t, "H="; H
   W = SQR(2 * g * H)
   Q = AreaTubo * W
   DeltaV = Q * DeltaT
   Deltah = DeltaV / AreaSerb
   H = H - Deltah
   t = t + DeltaT
LOOP
```

La ricomposizione di tutti i passaggi permette di ricavare l'equazione differenziale di primo grado nella funzione $h(t)$ che, assieme alla condizione iniziale $h(0)$, descrive completamente il funzionamento del sistema. Infatti ricavando la quantità h dalla equazione *e)*, sostituendo Δh con la definizione *d)*, e ΔV dalla *c)* e Q dalla *b)*, si ottiene l'equazione ricorsiva che descrive il comportamento del sistema, ovvero la funzione $h(t)$:

$$h_{fut} \leftarrow h_{now} - \frac{\Omega_t}{\Omega s}\sqrt{2g \cdot h_{now}} \cdot \Delta t$$

che può essere scritta in notazione indiciale come:

$$h_{i+1} \leftarrow h_i - \frac{\Omega_t}{\Omega s}\sqrt{2g \cdot h_i} \cdot \Delta t$$

Osserviamo inoltre che tale espressione può essere ricavata direttamente anche impostando le equazioni di conservazione della massa ($\Omega_s \cdot w_s = \Omega_t \cdot w \rightarrow -\Omega_s \cdot \frac{dh}{dt} = \Omega_t \cdot w$) e dell'energia ($w = \sqrt{2g \cdot h}$), dalle quali, eliminando w tra le due equazioni, si ricava:

$$\frac{dh(t)}{dt} = h'(t) = -\frac{\Omega_t}{\Omega s}\sqrt{2g \cdot h(t)}$$

che trasformata in differenze finite fornisce appunto:

$$\frac{h(t+\Delta t) - h(t)}{\Delta t} \cong -\frac{\Omega_t}{\Omega s}\sqrt{2g \cdot h(t)} \rightarrow h(t+\Delta t) \cong h(t) - \Delta t \cdot \frac{\Omega_t}{\Omega s}\sqrt{2g \cdot h(t)}$$

la quale può essere codificata immediatamente in forma algoritmica:

```
'Calcolo del tempo di svuotamento di un serbatoio per
gravita'.

H0 = 10          ' m
DeltaT = .1      ' s
AreaSerb = 1     ' m^2
AreaTubo = .1    ' m^2
g = 9.816        ' m/t^2

Hpast = H0

DO WHILE (Hnow > 0)
   PRINT "t="; t, "H="; Hnow

   HFut=Hnow-(AreaTubo/AreaSerb)*SQR(2*g*Hnow)*DeltaT

   t = t + DeltaT
   Hnow = HFut
LOOP
```


☞Complicare il problema considerando il ruolo del coefficiente α, costante oppure funzione della velocità secondo le usuali formule empiriche dell'idraulica.

☞Complicare il problema aggiungendo un peso appoggiato sul pelo libero del liquido.

☞Complicare il problema considerando l'effetto di un gas in pressione nel serbatoio del liquido.

☞Complicare il problema considerando l'effetto di una molla che spinge su di uno stantuffo scorrevole sul pelo libero del liquido.

☞Complicare il problema supponendo che il liquido invece che defluire liberamente passi in un altro serbatoio, ad un certa quota ed in comunicazione con l'atmosfera.

☞Complicare il problema supponendo che il liquido invece che defluire liberamente passi in un altro serbatoio, inizialmente pieno d'aria, ed ermeticamente chiuso.

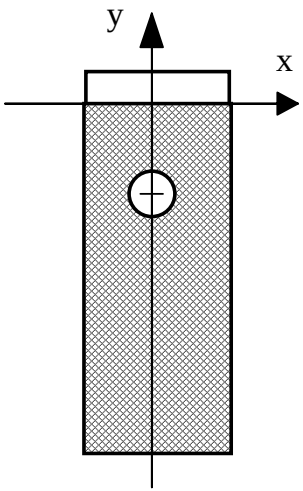
☞Complicare il problema considerando l'effetto di una pompa per il riempimento o lo svuotamento.

Esempio 2 : il viscosimetro a caduta

Consideriamo ora come seconda applicazione un *viscosimetro a caduta*. Tale dispositivo assomiglia ad un serbatoio cilindrico, con altezza pari a diverse volte il diametro di base, riempito del liquido del quale si vuole misurare la viscosità. Si lascia cadere una pallina nel fluido e se ne misura il tempo di caduta; da questa indicazione si può risalire alla viscosità del fluido.
Costruiamo il modellino matematico del sistema (ovvero scriviamo l'equazione del moto della pallina):

$$F = m \cdot a$$

le forze che agiscono sulla pallina sono quella di gravità (*Fg*), la spinta di Archimede (*Fa*), la resistenza del fluido (*Ff*).
Orientando un sistema di riferimento come in figura risulta:



$$F_g = -mg$$

$$F_a = \left(\pi \frac{4}{3} r^3\right) g\rho$$

$$Ff = \begin{cases} \mathrm{Re} < 2000 & -(6\pi\mu r)v \\ \mathrm{Re} > 5000 & -\left(Cx \frac{1}{2} \pi r^2 \rho\right) v^2 \end{cases}$$

$$\mathrm{Re} = \frac{\rho v r}{\mu}$$

allora considerando come funzione incognita la variabile velocità, l'equazione del moto diviene:

$$F_g + F_a + F_f = \left(-mg + \pi \frac{4}{3} r^3 \rho g\right) - (6\pi r \eta) \cdot v(t) = m \frac{dv(t)}{dt}$$

$$K_1 - K_2 v(t) = v'(t)$$

che è risolvibile secondo lo schema già visto: si sostituiscono le differenze finite ai differenziali, così l'equazione differenziale si trasforma in una semplice equazione algebrica:

$$v_i = v_{i-1} + \Delta t \cdot \left(K_1 - K_2 v_{i-1}\right)$$

i passi risolventi (semplicissimi) sono dunque:


```
'                Viscosimetro a caduta

m = 1            'kg
g = 9.816        'm/s^2
Pi = 3.14159
r = .1           'm
Ro = 1000        'kg/m^3
Eta = .2         'Pa*s
Tm = 10          's
N = 5000

K1 = (-m * g + Pi * 4 / 3 * r ^ 3 * Ro *g) / m
K2 = (6 * Pi * r * Eta) / m

DeltaT = Tm / N

DIM V(N)
V(0) = 0

FOR i = 1 TO N
   T = T + DeltaT

   F = K1 - K2 * V(i - 1)

   V(i) = V(i - 1) + DeltaT * F

   PRINT "T="; T, "V="; V(i)
NEXT i

END
```

☞Provare a sviluppare la soluzione alle differenze finite per il caso di elevato numero di Reynolds e confrontare i risultati otenuti nel caso di basso numero di Reynolds.
☞Tracciare su di un grafico la legge velocità-tempo di caduta.
☞Calcolare il tempo che occorre per raggiungere una determinata velocità (p.e. il 90% del valore asintotico).
☞Tracciare diverse curve velocità-tempo, per diversi valori della temperatura del fluido, tenendo conto della particolare legge di variazione della viscosità con la temperatura.

```
'   Soluzione della generica equazione differenziale di
I grado
'     nella forma:  x'(t)=F(x(t),t)

x0 = 10
TMax = 50

N = 500
DIM x(N)


'       Boundary conditions
x(0) = x0
Dt = TMax / N

FOR i = 0 TO N-1
        t = t + Dt

        x(i + 1) = x(i) + Dt * F(x(i), t)

        PRINT i, t, x(i), F(x(i), t)
NEXT i


SCREEN 12
FOR i = 0 TO N
        IF (x(i) < MinX) THEN MinX = x(i)
        IF (x(i) > MaxX) THEN MaxX = x(i)
NEXT i

WINDOW (0, MinX)-(t, MaxX)

FOR i = 0 TO N
        PSET (i * Dt, x(i))
NEXT i
SLEEP
END


FUNCTION F (x, t)
        F = 2 * t
END FUNCTION
```


Equazioni differenziali di secondo grado: soluzione analitica, applicazioni fisiche, discretizzazione alle differenze finite.

Ancora con riferimento al caso della caduta di un grave in un mezzo viscoso, si pensi di voler ricavare, invece della $v(t)$, la legge oraria del moto $y(t)$. In tal caso, essendo $v(t)=dy(t)/dt$, l'equazione differenziale del moto diviene:

$$K_1 - K_2 \frac{dy(t)}{dt} = \frac{d^2y(t)}{dt^2} \text{ o anche } K_1 - K_2 y'(t) = y''(t)$$

Compare cioè la funzione incognita $y(t)$ derivata due volte.
Sviluppando i rapporti incrementali, l'equazione differenziale si trasforma in una serie di equazioni algebriche concatenate. Con tale metodo si possono risolvere tutte le equazioni differenziali del tipo:

$$\frac{d^2y(t)}{dt^2} = F\left(\frac{d}{dt}y(t), y(t), t\right) \quad o \quad y''(t) = F\left(y'(t), y(t), t\right)$$

o più generalmente:

$$y^{(n)}(t) = F\left(y^{(n-1)}, y^{(n-2)}, ..., y(t), t\right)$$

Vediamo come si trasforma la derivata seconda in termini di differenze finite. Data la funzione $y(t)$, definiamone il rapporto incrementale di primo grado come:

$$y'(t) \to \frac{y(t+\Delta t) - y(t)}{\Delta t}$$

e quello di secondo grado vale:

$$y''(t) \to \frac{y'(t+\Delta t) - y'(t)}{\Delta t}$$

dove evidentemente risulta:

$$y'(t+\Delta t) \to \frac{y(t+\Delta t+\Delta t) - y(t+\Delta t)}{\Delta t} = \frac{y(t+2\Delta t) - y(t+\Delta t)}{\Delta t}$$

allora sostituendo nella espressione della derivata seconda risulta:

$$y''(t) \to \frac{1}{\Delta t}\left[\frac{y(t+2\Delta t) - y(t+\Delta t)}{\Delta t} - \frac{y(t+\Delta t) - y(t)}{\Delta t}\right] = \frac{y(t+2\Delta t) - 2y(t+\Delta t) + y(t)}{\Delta t^2}$$

Si vede dunque che per la stima numerica del valore della derivata seconda di una funzione *y(t)* in un punto, occorre calcolare il valore della funzione in tre punti.

Disponendo allora di una equazione del tipo:

$$\frac{d^2y(t)}{dt^2} = F\left(\frac{d}{dt}y(t), y(t), t\right) \quad o \quad y''(t) = F\left(y'(t), y(t), t\right)$$

la traduzione in forma di differenze finite diviene:

$$\frac{y(t+2\Delta t) - 2y(t+\Delta t) + y(t)}{\Delta t^2} = F\left[y(t+\Delta t), y(t), t\right]$$

esplicitando il termine $y(t+2\Delta t)$ si ottiene:

$$y(t+2\Delta t) = \Delta t^2 \cdot F\left[y(t+\Delta t), y(t), t\right] + 2y(t+\Delta t) - y(t)$$

si vede che noti i valori di y_0 e y_1, possiamo calcolare con questa forma ricorsiva tutti i valori seguenti di *y(t)*.

Generalmente della storia di un sistema dinamico sono imposte le *condizioni iniziali* in termini di posizione e di velocità (dette anche condizioni al contorno). P.e. nel caso della nostra sferetta sia imposta la posizione dell'ordinata di partenza (al tempo *t=0*) *y(0)=0*, e la velocità iniziale p.e. nulla *v(0)=0*.
Da quest'ultimo valore si ricava la posizione nell'istante successivo, semplicemente dall'espressione della velocità, come rapporto incrementale nel primo intervallo di tempo.

$$y(\Delta t) = y(0) + v(0) \cdot \Delta t$$

Il flusso delle operazioni necessarie per determinare il valore di *y(t)* nei passi successivi, ovvero per risolvere l'equazione differenziale con questo tipo di condizioni al contorno diviene:


```
' Soluzione della generica equazione
' differenziale di II grado
' nella forma:  x''(t)=F(x(t),x'(t),t)

CLS
N = 15
Tmax = 5

DIM x(N), y(N), v(N)

y(0) = 0
v(0) = 0

Dt = Tmax / N
y(1) = y0 + v(0) * Dt

FOR i = 2 TO N
   t = i * Dt
   y(i) = (Dt ^ 2) * F(y(i - 2), v(i - 2), t) + 2 * y(i
- 1) - y(i - 2)
   v(i - 1) = (y(i) - y(i - 1)) / Dt
   PRINT t, y(i), v(i - 1)
NEXT i

FUNCTION F (x, v, t)
 F = 9.8
END FUNCTION
```

La forma risolvente può essere ottenuta anche per una via un poco più meccanica, spezzando l'equazione differenziale di II ordine in due equazioni differenziali di I ordine: a partire da una equazione differenziale di secondo grado nella forma consueta $x''(t) = F\left(x'(t), x(t), t\right)$, si può scrivere anche $Vx'(t) = F\left(Vx(t), x(t), t\right)$.
Approssimando la derivata con il suo rapporto incrementale possiamo scrivere:

$$\frac{Vx(t+\Delta t) - Vx(t)}{\Delta t} \approx F\left(Vx(t), x(t), t\right)$$

da cui, esplicitando $Vx(t+\Delta t)$, si ottiene:

$$Vx(t+\Delta t) \approx Vx(t) + \Delta t \cdot F(Vx(t), x(t), t)$$

tale espressione, assieme all'approssimazione incrementale di $Vx(t)$:

$$Vx(t) \approx \frac{x(t+\Delta t) - x(t)}{\Delta t} \rightarrow x(t+\Delta t) \approx x(t) + \Delta t \cdot Vx(t)$$

conducono alla soluzione del problema, che in forma indiciale compatta assume la forma (prima si risolve l'equazione in $v(t)$, e nota questa risolvo in $x(t)$):

$$Vx_{i+1} = Vx_i + \Delta t \cdot F(Vx_i, x_i, t_i) \quad \text{con} \quad \begin{matrix} t_i = i \cdot \Delta t \\ x_{i+1} = x_i + \Delta t \cdot Vx_i \end{matrix} \quad \text{per} \quad i \in]0, N]$$

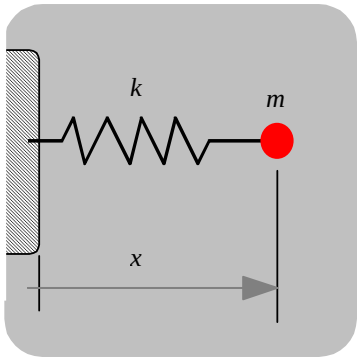
Evidentemente, per rendere determinata la soluzione numerica del problema, devono essere sempre specificate le condizioni iniziali, ovvero il valore delle quantità x e Vx all'istante iniziale t=0.


```
' Soluzione della generica equazione
' differenziale di II grado
' nella forma:  x''(t)=F(x(t),x'(t),t)

' Soluzione della generica equazione
' differenziale di II grado
' nella forma:  x''(t)=F(x(t),x'(t),t)

N = 50
Tmax = 5
y0 = 0
v0 = 0

DIM y(N), v(N)
y(0) = y0
v(0) = v0

Dt = Tmax / N

FOR i = 0 TO N - 1
   t = i * Dt

   v(i + 1) = v(i) + F(y(i), v(i), t) * Dt
   y(i + 1) = y(i) + v(i) * Dt

   PRINT t, y(i), v(i)

NEXT i

FUNCTION F (x, v, t)
 F = 9.8
END FUNCTION
```

☞Relativamente al viscosimetro a caduta, provare a sviluppare la soluzione alle differenze finite per il caso di elevato/basso numero di Reynolds.
☞Tracciare su di un grafico la legge spazio-tempo di caduta.
☞Tracciare su di un grafico la legge spazio-velocità di caduta.
☞Tracciare su di un grafico la legge spazio-accelerazione di caduta.
☞ Calcolare il tempo che occorre per percorrere tutta l'altezza del viscosimetro per valori diversi della viscosità.
☞Calcolare la viscosità in funzione del tempo impiegato per percorrere tutta l'altezza del viscosimetro.

Esempio 3 : massa + molla su guida rettilinea con attrito

Vediamo un esempio di meccanica assai istruttivo: consideriamo il sistema composto da una massa vincolata a muoversi su di una guida orizzontale rettilinea, collegata ad una molla:



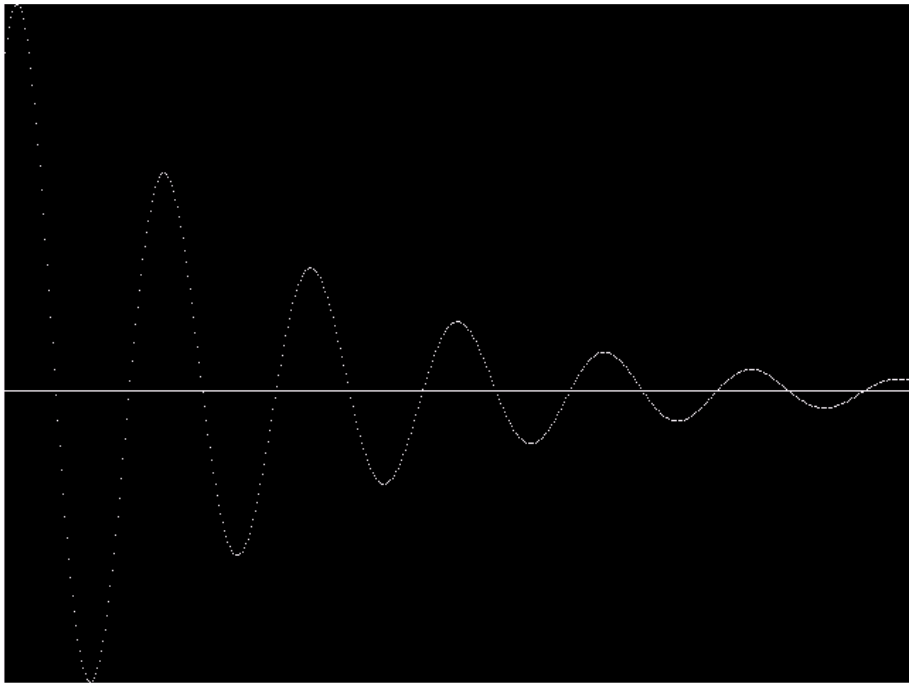
supponiamo che la massa *m* si muova in un mezzo viscoso e scriviamo l'equazione del moto lungo l'asse *x: F=m·a*. Sulla massa *m* intervengono due sole forze, il richiamo della molla e l'attrito: se si ipotizza che la molla sia a comportamento elastico lineare allora l'azione di richiamo vale *-k·x*, mentre se si ipotizza che il fluido che circonda la massa *m* (per esempio l'aria) abbia comportamento viscoso, allora la forza resistente vale *-h·Vx*:

$$-k\cdot x(t)-h\cdot x'(t)=m\cdot x''(t)\rightarrow\begin{cases}x''(t)=F(x'(t),x(t))\\ con\\ F(x'(t),x(t))=-\left(\frac{k}{m}\cdot x(t)+\frac{h}{m}\cdot x'(t)\right)\end{cases}$$

imponendo le condizioni al contorno *x(0)=d, x'(0)=0*, applicando lo schema risolutivo già visto si determina la soluzione *x(t)*, rappresentata da una funzione sinusoidale smorzata, ovvero con ampiezza decrescente nel tempo con legge esponenziale.


```
'       Equadiff II grado con grafico t-x:
'       Molla 1D con attrito viscoso.

COMMON SHARED h, k, m
x0 = 10
xd0 = 5
TMax = 50

h = .2: k = .6: m = 1

N = 2000
DIM x(N), xd(N)


x(0) = x0
xd(0) = xd0
Dt = TMax / N
FOR i = 0 TO N - 1
   t = t + Dt
   xd(i + 1) = xd(i) + Dt * F(xd(i), x(i), t)
   x(i + 1) = x(i) + Dt * xd(i)
   PRINT i; t, x(i); xd(i)
NEXT i


SCREEN 12
MinX = 1E+38
MaxX = -1E+38

FOR i = 0 TO N
   IF (x(i) < MinX) THEN MinX = x(i)
   IF (x(i) > MaxX) THEN MaxX = x(i)
NEXT i

WINDOW (0, MinX)-(t, MaxX)
LINE (0, 0)-(t, 0)
LINE (0, MinX)-(0, MaxX)
FOR i = 0 TO N
   PSET (i * Dt, x(i))
NEXT i
SLEEP
END

FUNCTION F (xd, x, t)
```

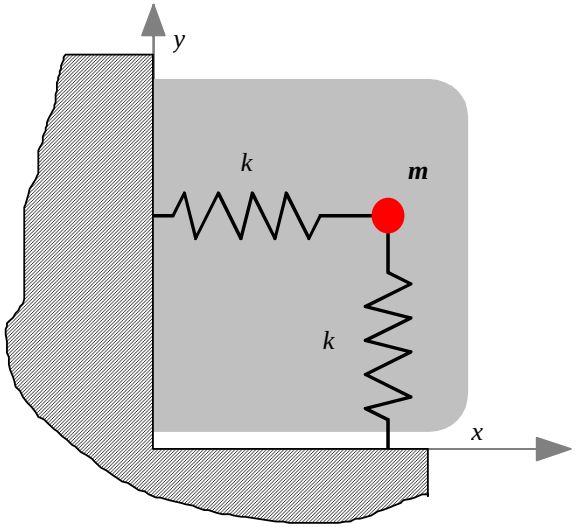
```
   F = -h/m * xd - k/m * x
END FUNCTION
```



☞Provare a sviluppare la soluzione con attrito nullo e con attrito radente.
☞Provare a sviluppare la soluzione con forza resistente del fluido pari a $-h \cdot Vx^n$, con $n>1$.
☞Tracciare su di un grafico la legge accelerazione-tempo.
☞Tracciare su di un grafico la legge accelerazione-spazio.
☞Tracciare su di un grafico la legge spazio-velocità.
☞Tracciare su di un grafico la legge velocità-accelerazione.
☞Calcolare il tempo che occorre per percorrere un'oscillazione completa per valori diversi del coefficiente di resistenza aerodinamica.
☞Pendolo: studiare le grandi oscillazioni e confrontarle con la soluzione teorica. Includere anche l'attrito.

Esempio 4 : massa + molle su guida piana con attrito

Vediamo un altro esempio di meccanica: consideriamo il sistema composto da una massa vincolata a muoversi su di un piano orizzontale e collegata a due molle scorrevoli:



supponiamo che la massa m si muova in un mezzo viscoso e scriviamo l'equazione vettoriale del moto considerando la forza di richiamo della molla e l'attrito:

$$\vec{F} = m \cdot \vec{a} \rightarrow \begin{cases} F_x = m \cdot a_x \\ F_y = m \cdot a_y \end{cases} \rightarrow \begin{cases} x'' = F_x(x'(t), x(t)) \\ y'' = F_y(y'(t), y(t)) \end{cases} \text{ con}$$

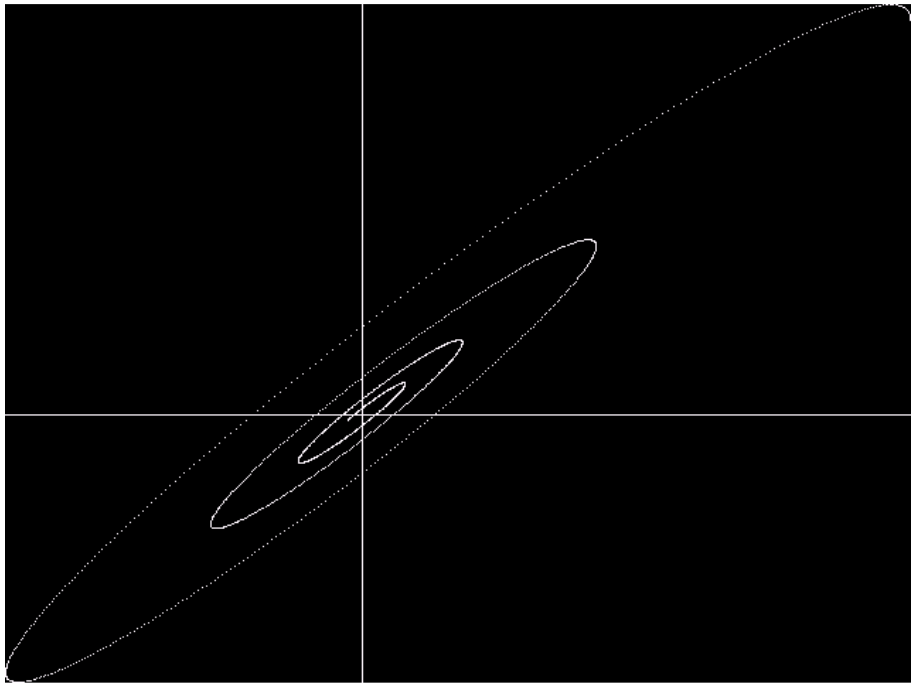
$$\begin{cases} F_x = -\left(\frac{k}{m} \cdot x(t) + \frac{h}{m} \cdot x'(t)\right) \\ F_y = -\left(\frac{k}{m} \cdot y(t) + \frac{h}{m} \cdot y'(t)\right) \end{cases}$$

imponendo le condizioni al contorno $x(0)=d_x$, $y(0)=d_y$, $x'(0)=0$, $y'(0)=0$, applicando lo schema risolutivo già visto si determina la soluzione $x(t)$ $y(t)$, rappresentata da una funzione sinusoidale smorzata, ovvero con ampiezza decrescente nel tempo con legge esponenziale.

```
'       Molle 2D con attrito viscoso

x0 = .1: y0 = .05
xd0 = 0: yd0 = .1
TMax = 3

N = 1500
DIM x(N), xd(N), y(N), yd(N)

x(0) = x0: y(0) = y0
xd(0) = xd0: yd(0) = yd0
Dt = TMax / N

FOR i = 0 TO N-1
        t = t + Dt
        xd(i + 1) = xd(i) + Dt * Fx(xd(i), yd(i), x(i),
y(i), t)
        yd(i + 1) = yd(i) + Dt * Fy(xd(i), yd(i), x(i),
y(i), t)

        x(i + 1) = x(i) + Dt * xd(i)
        y(i + 1) = y(i) + Dt * yd(i)

        PRINT i; t, x(i); y(i)
NEXT i


SCREEN 12
MinX = 1E+38: MinY = MinX
MaxX = -1E+38: MaxY = MaxX

FOR i = 0 TO N
        IF (x(i) < MinX) THEN MinX = x(i)
        IF (x(i) > MaxX) THEN MaxX = x(i)

        IF (y(i) < MinY) THEN MinY = y(i)
        IF (y(i) > MaxY) THEN MaxY = y(i)
NEXT i

WINDOW (MinX, MinY)-(MaxX, MaxY)
LINE (MinX, 0)-(MaxX, 0)
LINE (0, MinY)-(0, MaxY)
```



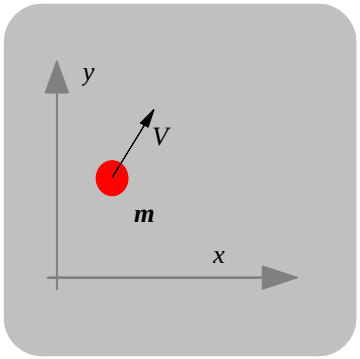
```
FOR i = 0 TO N
        PSET (x(i), y(i))
NEXT i
SLEEP
END

FUNCTION Fx (xd, yd, x, y, t)
       k=50
       m=1
       h=2
        Fx = -(k/m) * x – (h/m) * xd
END FUNCTION

FUNCTION Fy (xd, yd, x, y, t)
       k=50
       m=1
       h=2
        Fy = -(k/m) * y – (h/m) * yd
END FUNCTION
```



☞Provare a sviluppare la soluzione con diverse leggi di attrito e confrontarle con la condizione di attrito nullo.
☞Provare a sviluppare la soluzione con attrito radente.
☞Provare a sviluppare la soluzione con forza resistente del fluido pari a $-h \cdot V^n$, con $n>1$.
☞Tracciare su di un grafico la legge a_x-a_y.
☞Tracciare su di un grafico la legge Vx-Vy.

Esempio 5 : moto del proiettile con attrito

Impostiamo l'equazione differenziale del moto di un punto materiale vincolato a muoversi su di un piano verticale, in un campo gravitazionale unifome e soggetto ad un'azione resistente di tipo aerodinamico:



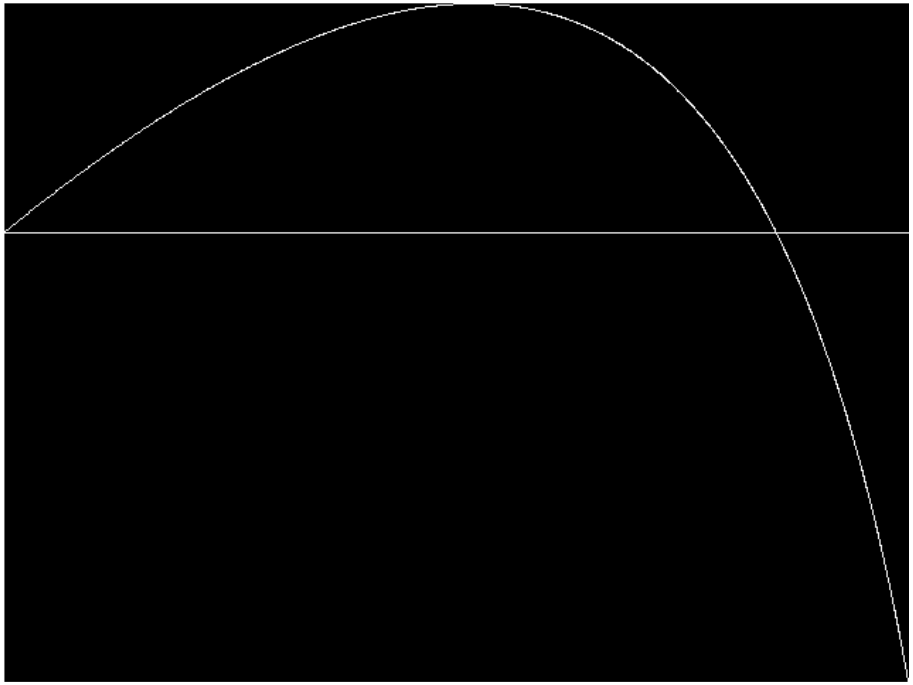
supponiamo che la massa m si muova in un mezzo viscoso e scriviamo l'equazione vettoriale del moto considerando la forza di gravità e l'attrito:

$$\vec{F} = m \cdot \vec{a} \rightarrow \begin{cases} F_x = m \cdot a_x \\ F_y = m \cdot a_y \end{cases} \rightarrow \begin{cases} x'' = F_x(x'(t), x(t)) \\ y'' = F_y(y'(t), y(t)) \end{cases} \text{ con } \begin{cases} F_x = -\left(\frac{h}{m} \cdot x'(t)\right) \\ F_y = -\left(g + \frac{h}{m} \cdot y'(t)\right) \end{cases}$$

imponendo le condizioni al contorno $x(0)=0$, $y(0)=0$, $x'(0)=V_{x0}$, $y'(0)=V_{y0}$, applicando lo schema risolutivo già visto si determina la traiettoria del proiettile in forma parametrica $x(t)$ $y(t)$.


```
'       Moto del proiettile in aria

x0 = 0: y0 = 0
xd0 = .1: yd0 = 8
TMax = 2

N = 1500
DIM x(N), xd(N), y(N), yd(N)

x(0) = x0: y(0) = y0
xd(0) = xd0: yd(0) = yd0
Dt = TMax / N

FOR i = 0 TO N-1
        t = t + Dt
        xd(i + 1) = xd(i) + Dt * Fx(xd(i), yd(i), x(i),
y(i), t)
        yd(i + 1) = yd(i) + Dt * Fy(xd(i), yd(i), x(i),
y(i), t)
        x(i + 1) = x(i) + Dt * xd(i)
        y(i + 1) = y(i) + Dt * yd(i)

        PRINT i; t, x(i), y(i)
NEXT i


SCREEN 12
MinX = 1E+38: MinY = MinX
MaxX = -1E+38: MaxY = MaxX

FOR i = 0 TO N
        IF (x(i) < MinX) THEN MinX = x(i)
        IF (x(i) > MaxX) THEN MaxX = x(i)

        IF (y(i) < MinY) THEN MinY = y(i)
        IF (y(i) > MaxY) THEN MaxY = y(i)
NEXT i

WINDOW (MinX, MinY)-(MaxX, MaxY)
LINE (MinX, 0)-(MaxX, 0)
LINE (0, MinY)-(0, MaxY)
FOR i = 0 TO N
        PSET (x(i), y(i))
```

```
NEXT i
SLEEP
END

FUNCTION Fx (xd, yd, x, y, t)
        Fx = -1 * xd
END FUNCTION

FUNCTION Fy (xd, yd, x, y, t)
        Fy = -9.81 - 1 * yd
END FUNCTION
```



☞Confrontare i risultati ottenuti con la soluzione teorica.
☞Provare a sviluppare la soluzione con diverse leggi di attrito.
☞Provare a sviluppare la soluzione con forza resistente del fluido pari a $-h \cdot V^n$, con $n>1$.
☞Tracciare su di un grafico la legge Vx-Vy.
☞Calcolare il tempo di volo.

☞Determinare l'angolo di lancio, che a parità di velocità iniziale ed in presenza d'aria, garantisce la gittata massima.
☞Determinare le condizioni di lancio che permettono di colpire un bersaglio preassegnato.
☞Introdurre un disturbo random in modo da simulare p.e. condizioni ventose, e verificare l'influenza di queste sulla traiettoria al variare della massa del proiettile.
☞Proiettile a massa variabile, come un missile che consuma del carburante.
🕮 Vedere anche *Calculus* di H. Anton: p.394, 399-401 *applicazioni delle equazioni differenziali*.

Esempio 6 : campo elettrico

$$\vec{F} = m \cdot \vec{a} \rightarrow \vec{F}_r = E \frac{q \cdot Q}{r^2} \frac{\vec{r}}{|\vec{r}|} \rightarrow \begin{cases} F_x = m \cdot a_x \\ F_y = m \cdot a_y \end{cases} \rightarrow \begin{cases} x'' = F_x(x'(t), x(t)) \\ y'' = F_y(y'(t), y(t)) \end{cases}$$

$$\text{con} \begin{cases} F_x = -\frac{E}{m} \frac{q \cdot Q}{x^2 + y^2} Sgn(x) \\ F_y = -\frac{E}{m} \frac{q \cdot Q}{x^2 + y^2} Sgn(y) \end{cases}$$

Essendo *Sgn(x)* la funzione segno di *x*.

Esempio 7 : campo gravitazionale

$$\vec{F} = m \cdot \vec{a} \rightarrow \vec{F}_r = G \frac{m \cdot M}{r^2} \frac{\vec{r}}{|\vec{r}|} \rightarrow \begin{cases} F_x = m \cdot a_x \\ F_y = m \cdot a_y \end{cases} \rightarrow \begin{cases} x'' = F_x(x'(t), x(t)) \\ y'' = F_y(y'(t), y(t)) \end{cases}$$

$$\text{con} \begin{cases} F_x = -G \frac{M}{x^2 + y^2} Sgn(x) \\ F_y = -G \frac{M}{x^2 + y^2} Sgn(y) \end{cases}$$

Essendo *Sgn(x)* la funzione segno di *x*.

Esempio 8 : campo magnetico

$$\vec{F} = m \cdot \vec{a} \rightarrow \vec{F} = q \cdot \vec{V} \wedge \vec{B} = q \cdot \begin{bmatrix} i & j & k \\ V_x & V_y & 0 \\ 0 & 0 & B_z \end{bmatrix} \rightarrow \begin{cases} F_x = m \cdot a_x \\ F_y = m \cdot a_y \end{cases} \rightarrow \begin{cases} x'' = F_x(x'(t), x(t)) \\ y'' = F_y(y'(t), y(t)) \end{cases}$$

$$\text{con} \begin{cases} F_x = q \cdot B_z \dfrac{x'}{m} \\ F_y = -q \cdot B_z \dfrac{y'}{m} \end{cases}$$

Esempio 9 : conduzione del calore stazionaria (1D)

Un'altra equazione interessante è quella che regola la trasmissione del calore per conduzione. Nel caso monodimensionale stazionario si riduce alla:

$$\frac{d^2T(x)}{dx^2} = 0$$

Esempio 10 : conduzione del calore stazionaria (2D)

Esempio 11 : conduzione del calore in regime transitorio (1D)

Mentre nel caso monodimensionale ma transitorio diviene:

$$\lambda \frac{d^2T(x,t)}{dx^2} = c_p \rho \frac{dT(x,t)}{dt}$$

Esempio 12 : conduzione del calore in regime transitorio (2D)

16. Tesine supplementari

N.	*Titolo*	*Difficoltà [1-4]*	*Opportunità [1-4]*
1	Somma degli elementi di un vettore	1	
2	Radici del polinimio di secondo grado	1	
3	Somma degli elementi positivi di un vettore	1	
4	Calcolo del numero di elementi di un vettore compresi in un certo intervallo	2	
5	Calcolo del prodotto degli elementi di un vettore	1	
6	Determinazione del più piccolo elemento di un vettore	1	
7	Calcolo del prodotto dei primi N numeri interi	1	
8	Distribuzione voti studenti	2	
9	Calcolo coefficenti binomiali secondo la definizione	2	
10	Generatore di numeri casuali	2	
11	Stampa di una tavola pitagorica	1	
12	Somma di due matrici	1	
13	Accrescimento popolazione di farfalle	2	
14	Equilibrio volpi-fagiani	3	
15	Ricerca dei numeri primi	3	
16	Calcolo della derivata n-esima di una funzione	3	
17	Ricerca delle terne pitagoriche	2	
18	Prodotto matriciale	2	
19	Ricerca binaria	2	
20	MCD ed MCM	2	
21	Triangolo di Tartaglia	2	
22	Ordinamento di un vettore alfanumerico	2	
23	Confronto tra ordinamento a bolla e per scelta	3	
24	Calcolo della funzione esponenziale con serie di potenze	2	
25	Serie di potenze convergenti	2	
26	Calcolo delle aree con serie di Archimede	2	
27	Espansione gas reali	3	
28	Integrazione equazione di diffusione monodimensionale anche non stazionaria (calore)	3	4
29	Integrazione equazione di diffusione monodimensionale anche non stazionaria (soluti)	3	4

30	Integrazione equazione di diffusione monodimensionale anche non stazionaria (evaporazione)	3	4
31	Integrazione equazione di diffusione monodimensionale anche non stazionaria (scarica condensatori)	3	4
32	Achille e la tartaruga	2	
33	Integrali definiti con Trapezi e Simpson	3	
34	Calcolo del calore necessario per fondere una sostanza	3	4
35	Sistemi lineari con Gauss	4	
36	Sistemi lineari con Seidel	3	
37	Radici di equazioni con metodo dicotomico	3	
38	Radici di equazioni con Newton	3	
39	Radici di equazioni con metodo della secante	3	
40	Radici di polinimi	3	
41	Serie di Mengoli	2	
42	Aree di Archimede	2	
43	Achille e la tartaruga	2	
44	Lunghezza di curve	3	
45	Valor medio di funzioni	3	
46	Verifica della bontà del generatore di numeri casuali.	2	
47	Generatore di numeri casuali con distribuzione predefinita	3	
48	Approssimazione funzioni con serie di Fourier	3	
49	Radici di equazioni con metodo iterativo	3	
50	Calcolo del coefficiente λ dall'eq. di Colebrook	4	4
51	Calcolo di un impianto idraulico di sollevamento	4	4
52	Calcolo della velocità dell'acqua in una condotta	4	4
53	Life	3	4
54	Calcolo del giorno tra due date	3	
55	Grafici di eq. in forma parametrica	3	
56	Grafici di funzioni in forma implicita	4	
57	Cammino hamiltoniano del cavallo	3	
58	Calcolo della radice quadrata	3	
59	Calcolo di π	2	
60	Cambio di base dei numeri	2	
61	Area di un poligono	2	
62	Sintesi di pangrammi	3	
63	Graficazione di funzioni in una variabile	3	
64	Curve ad evolvente	3	
65	Capitalizzazione composta	2	
66	Teoria delle code	3	
67	Orario scolastico	3	
68	Master Mind	3	

69	Stampa tabella ASCII	2	
70	Permutatore per anagrammi	3	
71	Grafico di funzione in 2 variabili	4	
72	Programmazione lineare	3	
73	Interpolazione polinomiale di Lagrange	3	
74	Divisione delle parole in sillabe	2	
75	Regressione ad un parametro	2	
76	Adattamento ad un polinomio	3	
77	Regressione lineare a più variabili	3	
78	Chicchi di grandine	3	
79	Awele	3	
80	Problemi di cammino minimo	3	
81	Problema 8 regine	3	
82	Sviluppo di un numero in lettere	3	
83	Esapedone	3	
84	Equazioni differenziali con differenze finite	4	
85	Equadiff con Runge-Kutta	4	
86	Sistemi di equazioni differenziali	4	
87	Equazioni di Fourier, Laplace e Poisson	4	4
88	Spettro di assorbimento	4	4
89	Conduzione termica tra due corpi di massa finita	4	4
90	Conduzione termica monodimensionale stazionaria	3	4
91	Mappa dei 4 colori	4	
92	Serie di Fourier	3	
93	Moto in mezzo viscoso	4	4
94	Oscillazioni elastiche	4	3
95	Transitori elettromagnetici	4	3
96	Sfera in caduta libera (moto laminare)	4	4
97	Sfera in caduta libera (moto turbolento)	4	
98	Moto del proiettile in aria	4	
99	Angolo corrispondente alla massima gittata (nel vuoto ed in aria)	4	
100	Proiettile sganciato da un aereo	4	
101	Moto di un satellite	4	
102	Moto di un corpo a massa variabile	4	
103	Fuga dalla terra	4	
104	Oscillatore armonico	4	
105	Animazione moto dei corpi liberi	4	
106	Nim	3	
107	Funzione faccia	3	
108	Pendolo matematico	4	
109	Oscillatore arminico smorzato	4	
110	Calcolo della pressione esercitata da un gas perfetto	4	4
111	Relazione tra temperatura ed energia interna	4	4

112	Espansione adiabatica	4	4
113	Espansione isoterma	4	4
114	Irreversibilità (espansioni libere)	4	4
115	Ciclo di Carnot	4	4
116	Ciclo Otto	4	3
117	Ciclo Diesel	4	3
118	Spettro di emissione corpo nero	4	4
119	Potenza emessa dal corpo nero	4	4
120	Moto di un elettrone in un campo elettrico	4	
121	Moto di una carica in un campo magnetico	4	
122	Transitorio RCL	4	
123	Circuito RCL con C.A.	4	
124	Transitorio RL	4	
125	Ponte di Wheatstone	4	
126	Puntofita e curvosauro	3	
127	Crivello di Eratostene	3	
128	Microgolf	3	
129	Automi cellulari	3	
130	Biomorfi	3	
131	Turmiti	3	
132	Difetti e Demoni	3	
133	Capra e cavoli	3	
134	Nimble	3	

17. La simulazione numerica nell'ingegneria agroalimentare

FEA