quaderni di CABRIRRSAE

Speciale Convegno

Presentazione a cura di:
Franca Noè e Daniele Tasso

"L'insegnamento dell'ALGEBRA (E NON SOLO) nell'era dei COMPUTER"

con la collaborazione di

Direct Media
MEDIA DIRECT srl

CampuStore.it
Tecnologie per la didattica

n° 26

Franca Noè, insegnante di matematica, collaboratrice dell'IRRE ER, fa parte della redazione del bollettino CABRI*RRSAE* e partecipa da alcuni anni ad attività legate alla utilizzazione del software CABRI

Daniele Tasso, insegnante di lettere, fa parte della redazione del bollettino CABRI*RRSAE*, ha curato vari articoli dedicati alla scuola e collabora da anni con l'IRRE ER

Released Information

Progettazione e videoimpaginazione GRAPHICART
Via Fondazza, 37 - 40125 BOLOGNA • Tel. - Fax - Seg. (051) 30.70.73
Modem (051) 42.920.47 • e-mail: graphicart.paolo@tin.it • www.graphicartstudio.it

Illustrazioni Luca Poli - Via Licinia, 7 - Tel. 051.313907 - BO
tratte dal poster "dall'Abaco al Computer"

Gottfried Wilhelm
von Leibniz

dall'Abaco al Computer

Ai docenti partecipanti al Convegno
verranno distribuiti materiali di supporto riguardanti
l'insegnamento della matematica ed il Poster "dall'Abaco al Computer"
dedicato ad una breve e sintetica storia delle idee e delle invenzioni
che hanno portato alla messa a punto
dei moderni elaboratori

IRRE-Emilia Romagna

in collaborazione con

Media Direct di Bassano del Grappa (VI)

Convegno 15 marzo 2004
Royal Hotel Carlton – Bologna

L'insegnamento dell'algebra (e non solo) nell'era dei computer

Programma

MATTINO

Coordina: Nerino Arcangeli,
IRRE Emilia Romagna

09.00 – 09.15 Presentazione
Franco Frabboni, Presidente IRRE-ER

09.15 – 10.00 Princìpi, assiomi e postulati, teoremi, lemmi e corollari, criteri, regole e leggi …
Da dove viene e a cosa serve tanta abbondanza di termini?
Benedetto Scimemi, Università di Padova

10.00 – 10.45 I sistemi di calcolo algebrico ieri e oggi
Giulio Cesare Barozzi, Università di Bologna

10.45 – 11.15 Intervallo

11.15 – 12.00 Programmare con *Derive*
Paolo Boieri, Politecnico di Torino

12.00 – 13.15 Novità di *Derive 6*
Bernhard Kutzler, docente all'Università di Linz, Austria
Direttore di *Soft Warehouse Europe*

13.15 – 14.30 pausa pranzo

POMERIGGIO

14.30 – 15.00 Dopo *Pascal, Derive*?
Laura Gobetti, LS Giordano Bruno di Torino

15.00 – 15.45 Algoritmi iterativi
Sebastiano Cappuccio, ITAer di Forlì

15.45 – 16.30 *Derive* ed il progetto Eccellenza
Aurelia Orlandoni, IRRE Emilia Romagna

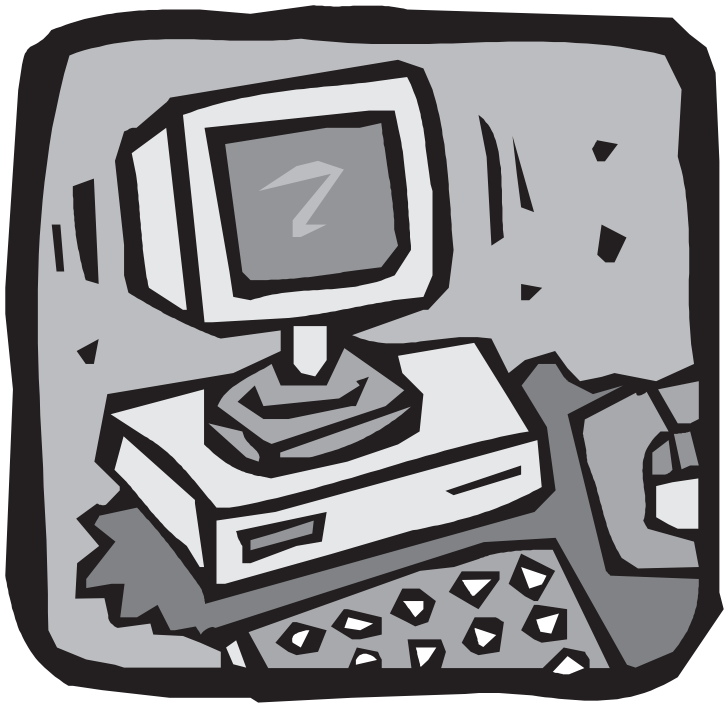
16.30 – 17.15 L'approccio a *Derive* dei futuri docenti di Scuola Secondaria Superiore
Giuseppe Accascina, Università La Sapienza, Roma

17.15 – 17.30 Estrazione premi
17.30 Conclusioni

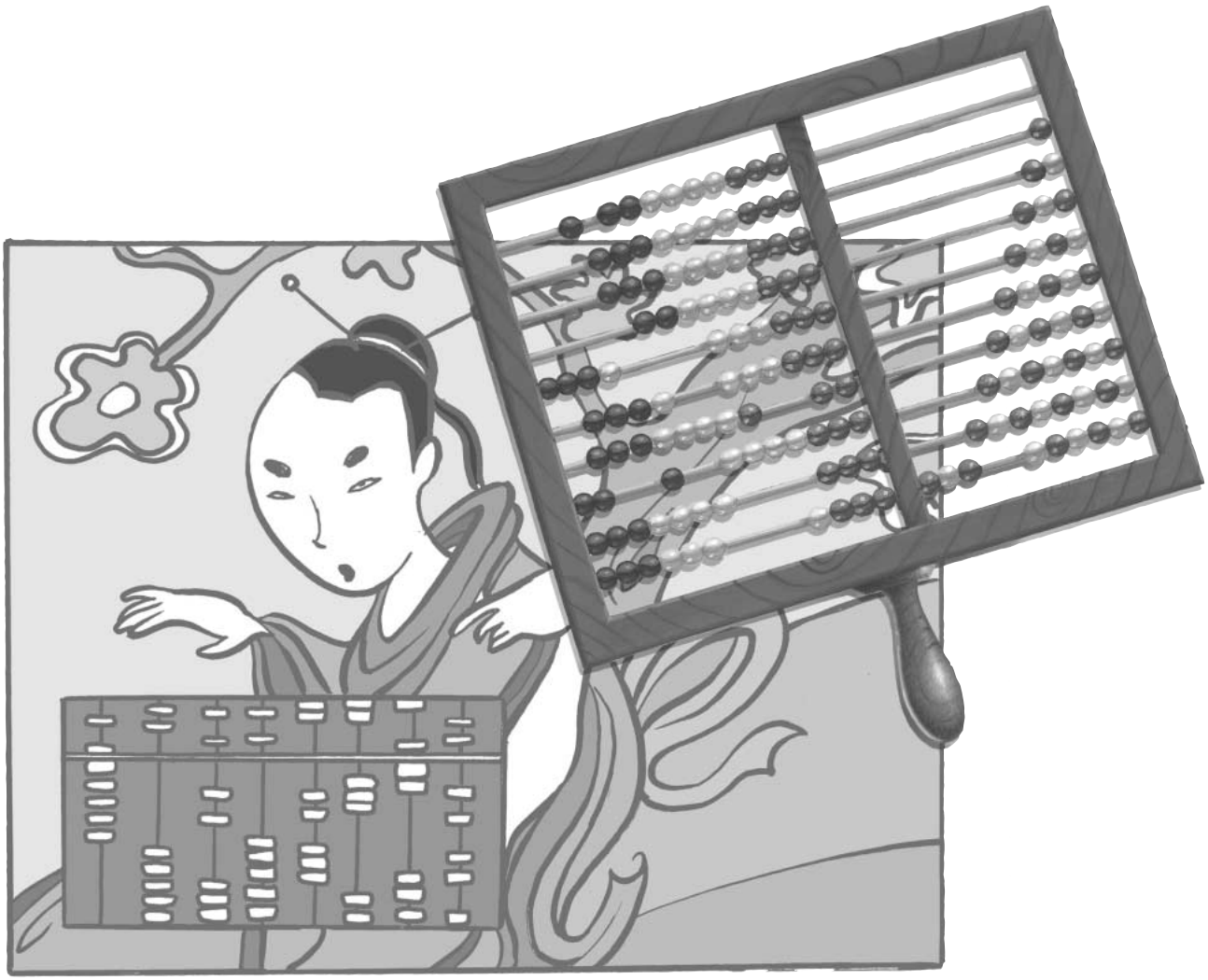
Indice

Da 30 anni a questa parte, i calcolatori hanno cessato di essere dei semplici "schiaccianumeri".
La capacità di eseguire manipolazioni algebriche, inizialmente limitata ai grandi elaboratori, si è progressivamente estesa ai personal computer, ed oggi è accessibile anche alle calcolatrici tascabili.

Le ricadute sulla didattica di queste nuove possibilità offerte dagli elaboratori elettronici, sono oggetto di viva discussione a livello internazionale.
IRRE Emilia Romagna, in collaborazione con Media Direct, intende promuovere a livello nazionale un dibattito sui Sistemi di Calcolo Algebrico (CAS = *Computer Algebra Systems*), con particolare attenzione al software *Derive*, che è attualmente il CAS più diffuso a livello di Scuola Secondaria di 2° grado.



“L’insegnamento dell’ALGEBRA (E NON SOLO) nell’era dei COMPUTER”



Abaco

Princìpi, assiomi e postulati, teoremi, lemmi e corollari, criteri, regole e leggi... Da dove viene e a cosa serve tanta abbondanza di termini?

di Benedetto Scimemi
Università di Padova

Riassunto: In matematica si incontrano molte parole che sembrano avere significati simili o identici. Questa ricchezza di termini in alcuni casi è superflua (*assioma* e *postulato*) e riconducibile a motivi tradizionali. In altri casi le differenze di significato sono discutibili (un *teorema* esigerebbe una dimostrazione, una *regola* si potrebbe prendere per buona senza provarla). Esistono però situazioni in cui l'uso di un termine piuttosto di un altro denuncia una certa confusione concettuale che vale la pena di chiarire (come nel caso del *principio* di identità dei polinomi). Ma anche quando si sia fatta chiarezza sui significati, l'opportunità di scegliere un termine può dipendere dall'impostazione che si vuol dare alla trattazione: un *postulato* può diventare un *teorema*, una *definizione* si trasforma in una *proprietà* e viceversa. Questa conversazione, avvalendosi anche dell'etimologia, cerca di chiarire alcune tipiche confusioni, e incoraggia a prestare attenzione all'uso dei termini, matematici e non, mantenendo un atteggiamento vigile e critico.

Qualche mese fa è uscito un nuovo libro, dal titolo *Cominciamo da Zero*, destinato ai docenti di matematica delle scuole secondarie. L'autore è Vinicio Villani, ben noto per il suo lungo impegno a favore della matematica, per la sua attenzione ai problemi della scuola, nonché per lo stile piacevolissimo – familiare ma preciso - delle sue conversazioni e pubblicazioni. Questo libro risponde a una quantità di *perché* che ogni docente, anche esperto, si deve porre se, nel suo contatto quotidiano con la matematica, vuole assumere un sano atteggiamento di critica invece di adeguarsi alla tradizione o seguire pedissequamente un libro di testo. Ecco qualche esempio delle questioni affrontate: perché meno per meno fa più? a che serve lo studio dei radicali? perché si introducono i numeri reali? che cosa afferma il teorema di identità dei polinomi?

Il titolo di questa conversazione è simile a quello del capitolo 33 di *Cominciamo da Zero*: perché tanti significanti per pochi significati? Ho avuto il privilegio di discutere il testo con l'autore durante la sua preparazione e quindi mi sono posto le stesse domande. Esporrò qui alcune delle mie risposte, che sono molto simili anche se non identiche alle sue. Anzitutto ho esteso l'elenco dei termini matematici in esame (quelli considerati da Villani sono tutti riconducibili, in definitiva, alle due grandi classi: cose da dimostrare, cose da prendere per buone). Poi, per meglio distinguerne i significati, ho cercato di risalire alla loro origine, annotando qualche differenza tra l'uso di una parola nel linguaggio comune e quello che se ne fa abitualmente in matematica. Addentrandomi in un campo che non è il mio (l'*etimologia* è una disciplina affascinante che richiede ben altra competenza) mi scuso fin d'ora per le imprecisioni. Infine ho discusso, con esempi, qualche concetto matematico che spiega come mai, cambiando impostazione, è giusto cambiare terminologia: un *principio* diventa un *teorema* ecc.

La lista che segue contiene anche qualche termine antiquato o … abusivo, ma è tutt'altro che completa; anzi, chiunque voglia accingersi ad elencare i termini matematici che ha incontrato nei libri, stenterà a fermarsi. Tanta abbondanza è caratteristica di una scienza antica, che ha permeato la cultura delle civiltà classiche, e si porta con sé l'eredità di varie discipline (soprattutto la filosofia) e di varie lingue (soprattutto il latino e il greco). Così per certi concetti si sono creati doppioni che sarebbero effettivamente superflui; ma forse la loro sopravvivenza non è un danno, purché l'uso sia consapevole e non crei ambiguità.

Le definizioni con la sigla DIZ: provengono da buoni dizionari italiani (generici), in cui è stata scelta, tra le varie accezioni del termine, quella più prossima all'uso matematico. I commenti *in corsivo* sono opinioni personali e pertanto discutibilissimi. La grafia delle parole greche è approssimativa, per l'indisponibilità del *font* adatto.

proposizione dal latino *pro-ponere* = mettere avanti, esporre, presentare
DIZ: affermazione che si propone alla discussione, alla meditazione di altri; l'enunciato di un problema o teorema. *In matematica è una frase qualsiasi, di cui ci si possa chiedere se è vera o falsa. Chiederselo non significa saper rispondere.*

affermazione dal latino *ad-firmare*, rafforzativo di *firmare* = assicurare
DIZ: opinione data per certa. *In matematica non ha nulla di certo. E' un altro sinonimo di* proposizione.

asserto (asserzione) dal latino *ad-sèrere* = dichiarare
DIZ: opinione data per certa. *Sinonimo di* affermazione*. Si incontra spesso nel corso di una dimostrazione, come sinonimo di* tesi *o di una sua parte:* "l'asserto sarà provato se dimostreremo che..."

enunciato dal latino *ex-nuntiare* = esporre, far conoscere (con una certa solennità)
DIZ: le parole con cui si esprime il testo di un teorema. *Anche di un* postulato, *di una* regola *ecc.*

definizione dal latino *de-finire* = delimitare, porre i confini = *fines*
DIZ: breve e precisa spiegazione di una voce, designazione delle proprietà distintive ed essenziali di una cosa. *In matematica certe definizioni non sono brevi! Uno stesso concetto può avere varie definizioni, che devono essere tutte equivalenti. In testi moderni si trova spesso:* "Definiamo ... se vale una (qualunque) delle seguenti proprietà ..." *Seguono le dimostrazioni che le varie proprietà (= definizioni) si implicano l'una l'altra, per esempio ciclicamente.*

convenzione dal latino *con-venire* = incontrarsi
DIZ. il convenire (di più persone) in un accordo. *Un tipico esempio di convenzione matematica, intesa soltanto al risparmio delle parentesi, è quella di eseguire le moltiplicazioni* prima *delle addizioni (si è deciso così, ma si sarebbe potuto* convenire *il contrario).*

postulato dal latino *postulare* = domandare, esigere
DIZ: proposizione che viene ammessa come vera senza dimostrazione, stante la sua evidenza e chiarezza. *In questa definizione è meglio fermarsi prima della virgola: è tutt'altro che evidente, per esempio, che "la velocità della luce sia indipendente dal sistema di riferimento", un postulato delle teorie relativistiche.*

assioma dal greco αξιωμα = pregio, valore, αξιοω = valuto
DIZ: verità evidente di per se stessa, che è universalmente accettata senza dimostrazione. *Nella filosofia antica si distingueva dal postulato, proprio perché l'assioma aveva un'evidenza intrinseca, il postulato poteva invece essere un'ipotesi di comodo. Attualmente per noi* assioma *e* postulato *sono sinonimi.*

principio dalla radice latina *pre* (di *precedere*, *primus*, *princeps* ecc.)
DIZ: l'atto o il modo con cui si comincia, proposizione fondamentale di una disciplina.
Una teoria assiomatica *inizia (= principia) con un elenco di postulati, che perciò si chiamano talvolta princìpi. Nell'uso comune, un principio può essere parte di una definizione* (p. del minimo, p. di identità dei polinomi)*. Ci sono abusi: non si dovrebbe, per esempio, parlare di* princìpi di equivalenza *quando si passa da un'equazione a un'altra che ha le stesse soluzioni. Si tratta di proprietà che hanno (e meritano) una dimostrazione e suggeriscono un* procedimento *o* metodo *per pervenire al risultato.*

congettura dal latino *conìcere* = gettare, mettere assieme
DIZ: supposizione, giudizio appoggiato a indizi con una certa probabilità di sicurezza. *Enunciato che non ha dimostrazione, ma per il quale non si conoscono (ancora) contro-esempi (per es. la* congettura di Goldbach*). La ricerca dei matematici trasforma lentamente molte congetture in teoremi (per es. la* c. di Fermat*). Alcune congetture si usano come punti di partenza (ipotesi di* Riemann*) per altre dimostrazioni. Talvolta si congettura che qualche postulato sia un teorema (per es. il* p. delle parallele*) ma poi ci si accorge che non sempre è così.*

teorema dal greco θεωρημα = l'oggetto che si considera (anche: spettacolo!) da οραω = vedo
DIZ: proposizione la cui verità deve essere dimostrata per via di deduzione. *Si usa riservare questo termine per gli enunciati di una certa importanza (vedi sotto:* Ruffini*) ma, in realtà, le proprietà, le regole, i criteri ecc., in quanto dimostrabili, sono altrettanti teoremi (più o meno ... spettacolari).*

lemma dal greco λημμα = acquisto, guadagno, da λαμβανω = prendere
DIZ: proposizione preliminare, che si assume come certa, per preparare la dimostrazione di un'altra. *Non la si assume* come certa, *la si dimostra prima*: *è un* teorema *il cui enunciato non sembra molto significativo di per sé, ma costituisce comoda tappa intermedia nel percorso dimostrativo di un teorema che segue. Talvolta il termine compare un po' a sproposito: il* Lemma di Zorn *si usa come un postulato.*

corollario dal latino *corollarium* = coroncina, che si donava, oltre il compenso, agli attori più valenti
DIZ: conseguenza di una proposizione (teorema) già dimostrata. *È un teorema che si ricava con poca fatica dal precedente (ma non per questo è meno importante).*

paradosso dal greco παρα-δοξον = che è contro la comune opinione (= δοξα)
DIZ: proposizione che è o appare contraddittoria al comune giudizio. *I paradossi incontrati nella storia della matemati-*

ca (per es. nella teoria degli insiemi*) hanno costretto a rivedere i sistemi di assiomi, producendo spesso teorie più avanzate, in cui la contraddizione viene cancellata.*

assurdo dal tardo latino *ab-surdus* = stonato, strano (da *surdus* = sordo)
DIZ: (proposizione) che è in contrasto con l'evidenza logica. *La dimostrazione* per assurdo *consiste nell' ipotizzare che la tesi sia falsa e nel dedurne una contraddizione*.

ipotesi dal greco υποθεσιζ = fondamento (dal verbo υπο-τιθημι = sotto-porre)
DIZ: supposizione, presupposto. *È una proposizione che si assume come punto di partenza di una* dimostrazione *che deve condurre, con opportune argomentazioni, alla* tesi.

tesi dal greco θεσιζ = posizione (dal verbo τιθημι = porre)
DIZ: proposizione che deve essere dimostrata, enunciato la cui verità si fonda su determinate premesse. *Le premesse sono l'ipotesi, i postulati, i teoremi precedentemente dimostrati ecc.*

assunto dal latino *ad-sùmere* = prendere, appropriarsi
DIZ: ciò che uno si propone di fare o di provare. *È un termine un po' antiquato, che corrisponde a uno scopo, un punto d'arrivo, perciò più adatto a una tesi che a un'ipotesi (come farebbe pensare l'inglesismo* assunzione, *da evitare).*

prova dal latino *probare* = verificare la bontà (da *probus* = buono)
DIZ: argomento che si adduce per dimostrare vero un fatto; esperimento che si fa per saggiare le qualità, operazione di verifica. *Qualcuno effettivamente usa* prova *come sinonimo di* dimostrazione *(provare = dimostrare, come in inglese* to prove*), ma conviene tener presente anche il significato di* esperimento *(in inglese* to try *= fare un tentativo) perché qualche prova (per es. la* p. del nove, *vedi 1) non dimostra quel che si vorrebbe.*

verifica dal tardo latino *verificare* = autenticare, documentare la verità
DIZ: accertamento, convalida, riscontro. *Anche qui, come per* prova*, una verifica negativa dimostra che qualcosa non va, una verifica positiva potrebbe non essere sufficiente a dimostrare qualcosa.*

dimostrazione dal latino *de-monstrare* = far palese, mostrare (*monstrum* è il prodigio!)
DIZ: argomentazione che assicura la verità di un asserto, di un teorema. *È il procedimento deduttivo che da certe premesse* (ipotesi) *conduce a certe conseguenze* (tesi)*. Per chi ama la matematica, una bella dimostrazione ha qualcosa di prodigioso.*

criterio dal greco κριτηριον = mezzo per giudicare (da κρινω = giudico)
DIZ: modo secondo cui giudichiamo la verità delle cose. *Si usa più spesso come condizione* sufficiente*. Per es. col* criterio del rapporto *proviamo che una successione converge, ma la condizione non è* necessaria*. Però altri familiari* criteri *(*c. di similitudine *dei triangoli) sono anche condizioni necessarie (e quindi si potrebbero usare come definizioni).*

metodo dal greco μεθοδοζ = ricerca (μετα-οδοζ = oltre la strada)
DIZ: modo di fare una cosa seguendo un certo ordine; serie di operazioni con cui la mente organizza il ragionamento. *Si usa soprattutto quando ce n'è più d'uno (es:* m. di sostituzione*) e dunque è termine didatticamente preferibile rispetto a* regola*, perché sottintende che ci siano altri modi di procedere (*m. di Cramer *per i sistemi lineari).*

argomento dal latino *argùere* = indicare, mostrare (da cui *arguto*)
DIZ: tutto ciò che si adduce in prova di una verità che si afferma. *Questo è il significato originario (meglio:* argomentazione*) ma più spesso, nel linguaggio comune, indica semplicemente il soggetto: "i logaritmi sono un argomento difficile…"*

illazione part.pass. del verbo latino *in-ferre* = portare a, derivare
DIZ: la conclusione tratta da premesse, la conseguenza che si deduce da un argomentazione. *Nell'uso corrente, l'*illazione *è piuttosto una deduzione illecita, non ben fondata. È un termine poco usato in matematica. Nei quotidiani un'* illazione *non dimostrata viene purtroppo chiamata* teorema.

deduzione dal latino *de-dùcere* = condurre da, derivare
DIZ: operazione della mente, per cui si procede dal generale al particolare. *Più semplicemente, ogni procedimento logi-*

co che conduca da certe premesse a certe conclusioni.

induzione dal latino *in-dùcere* = condurre in, derivare
DIZ: l'argomentare dal particolare al generale. *Per noi ha invece un significato più specifico, legato ai numeri naturali: una dimostrazione* per induzione *è basata su uno dei postulati di Peano (vedi 4).*

espressione dal latino *ex-prìmere* = generare, far uscire
DIZ: sequenza di segni, traduzione in simboli di una proposizione o operazione matematica. *Questa definizione è abbastanza generica e corrisponde all'uso tradizionale che se ne fa nella scuola (il termine non è in uso nella matematica moderna). Un'*espressione *(es: un quoziente di polinomi) non si dimostra. Si può elaborare, semplificare ecc.*

formula dal latino *formula* (diminutivo di *forma* = stampo) = regolamento, patto, regola
DIZ: i termini precisi con cui si deve esprimere un concetto, espressione algebrica.
Normalmente, una formula *contiene, a differenza di un* espressione*, un'uguaglianza (del tipo* sin2x = *…) e quindi va dimostrata. Allora diventa un teorema, che si enuncia in simboli invece che con linguaggio discorsivo.*

regola dal latino *regula* = condotta (dal verbo *règere* = condurre)
DIZ: norma, precetto, prescrizione di ciò che si deve fare. *Questo termine mi sembra pericoloso, perché induce lo studente a pensare che non ci sia altro modo di procedere per ottenere un risultato (es: la* r. di Ruffini*, vedi 14; la* r. di Cramer*). Sarebbe meglio parlare di* metodo *o* procedimento *(talvolta* algoritmo*, vedi sotto).*

legge dal latino *lex* (dal verbo *lègere* = raccogliere, mettere assieme)
DIZ: norma ferma e costante che si avvera nei fatti o si impone. *Il termine si usa più spesso in fisica. In matematica è sinonimo di* regola *e ne ha gli stessi difetti: fa pensare a un obbligo invece che a una possibilità (la* legge di cancellazione *ci dice solo che, se ci fa comodo, possiamo cancellare qualcosa senza perdere informazioni, vedi 6). Si può spesso sostituire con il termine* proprietà*, meno coercitivo.*

proprietà dal latino *proprius* = inerente o appartenente a qualcuno
DIZ: particolarità, qualità particolare che appartiene a una cosa. *Questo termine comunissimo mi sembra tanto facile da capire quanto difficile da definire. Una* proprietà *descrive in qualche modo il comportamento di un ente matematico (es: di una relazione, di una operazione, di una funzione). Può far parte della sua* definizione *(e quindi da non dimostrare) oppure essere conseguenza di altri fatti (quindi un* teorema *da dimostrare). Una proprietà può essere comune a enti diversi; se la proprietà è esclusiva, cioè ne gode solo quell'ente (es: quella funzione; vedi sotto) si parla di proprietà* caratteristica. *Vedi anche* definizione *e punto 3.*

sistema dal greco συσθημα = raccolta, collezione (συν-ισθημι = metto insieme)
DIZ: un tutto composto di varie parti, che hanno rapporti reciproci. *Qui l'etimo spiega bene che si stanno considerando più cose assieme (un s. di postulati, un s. di equazioni ecc.). Viene talvolta confuso con* metodo *("ho trovato il* sistema *per risolvere …"), ma non è un uso consigliabile. È corretto invece dire "risolviamo questo sistema lineare con il metodo … ".*

procedimento dal latino *pro-cedere* (= andare avanti)
DIZ: svolgimento di un'operazione. *Modo di procedere: una serie di operazioni (o anche di argomentazioni) organizzate in un certo ordine (*Svolgimento *è un vecchio termine della tradizione scolastica).*

algoritmo dal nome del matematico arabo *Al-Kuwarizmi,* cioè della regione di Kwarizm
DIZ: calcolo algebrico; ma in molti dizionari questa voce non si trova! *È una ricetta per il calcolo, un complesso di istruzioni o formule (non di ragionamenti) che, a partire da certi ingredienti (in inglese* inputs*), dopo un certo numero (finito) di passaggi, produce certi risultati* (*in inglese* outputs).

Illustrerò alcune di queste definizioni con esempi, che si riferiscono a livelli diversi di matematica scolastica, dalla scuola elementare all'università, senza alcuna sistematicità: mi permetterò cioè di saltare da un argomento all'altro, secondo l'occasione offerta dai singoli termini.

[N.B. A proposito di significati: *opportunità* non significa *occasione*: ci sono occasioni di far cose non opportune (es: rubare una bicicletta); ci sarebbe poi l'opportunità di farne altre per cui l'occasione non si presenta (es: restituirla).]

ARITMETICA

1) *Prova* del nove

Sappiamo tutti, fino dalle scuole elementari, che quando risulta positiva la cosiddetta *prova del 9*, non si può pretendere che sia corretto il risultato della operazione; tutt'al più si potrebbe dire che il risultato è corretto con probabilità prossima al 90%! Dunque, più che una *prova* di correttezza dei calcoli (nel senso di *dimostrazione*) è semplicemente un campanello d'allarme, che suona spesso in presenza di errori, ma non sempre. Insomma: se il calcolo è corretto, la prova è necessariamente positiva; se la prova è positiva, il calcolo non è necessariamente corretto. A questo proposito, perché tanti insegnanti non si sono mai presa la briga di *dimostrare* che questa prova (come *necessità*) è valida? forse non è il caso di farlo nelle classi elementari, ma può valer la pena di riprendere l'argomento più avanti, quando si parla di congruenze, di aritmetica *modulare*, proprio come esempio di condizione necessaria che non è sufficiente.

2) *Proprietà* delle operazioni

La parola *proprietà* ha acquisito, anche nel linguaggio comune, un significato meno esclusivo di quanto vorrebbe l'etimologia. Infatti per il Codice Civile una cosa è *propria* se non appartiene ad altri e quindi ne gode *solo* il proprietario. In matematica invece, le proprietà non sono esclusive. Per esempio, delle proprietà *associativa* e *commutativa* godono, in tutti gli ambienti di numeri, sia l'addizione che la moltiplicazione, legate inoltre dalle proprietà *distributive*, a destra e a sinistra, della moltiplicazione s*ulla* addizione. Sarebbe didatticamente opportuno, per far capire l'importanza delle proprietà in questione, considerare (al momento buono del programma) altre operazioni, pur ben conosciute, che *non* godono delle analoghe proprietà. Per esempio, se provvisoriamente scriviamo $x*y$ per indicare la potenza x^y, otteniamo un'operazione * (nell'ambiente *N* dei numeri naturali, ma potremmo metterci anche nei razionali o nei reali) non commutativa e non associativa, perché, in generale, risulta $x*y \neq y*x$, $x*(y*z) \neq (x*y)*z$. Può poi valer la pena di riprendere il discorso quando si parla delle *proprietà delle potenze*. Infatti

$(x\,y)*z = (x*z)(y*z)$ è una *distributività a destra* della * sulla moltiplicazione,

$x*(y+z) = (x*y)(x*z)$ è una sorta di distributività da sinistra che coinvolge tre operazioni,

$(x*y)*z = x*(yz)$ è una sorta di associatività che coinvolge due operazioni.

L'altra variante $(x+y)*z = \ldots$ si chiama *formula* del binomio (di Newton), ma naturalmente è anch'essa una *proprietà* delle operazioni.

3) Proprietà *caratteristiche*

Per recuperare il significato etimologico di *proprietà*, in matematica si usa parlare di *proprietà caratteristica*. In altre parole: *proprietà* si riferisce a una condizione necessaria; *proprietà caratteristica* suggerisce anche la sufficienza. Per esempio, la proprietà di *trasformare prodotti in somme* è caratteristica della funzione logaritmo, nel senso che

(a) log_a è una funzione continua (di R^+ in R) che gode della proprietà: $log_a(xy) = log_a x + log_a y$

(b) se f è una funzione continua (di R^+ in R) per cui risulta $f(xy) = f(x)+f(y)$ per ogni x, y allora la funzione è necessariamente $f = log_a$ pur di scegliere opportunamente la base a.

Una proprietà caratteristica - ma non una proprietà qualunque - si può quindi prendere come definizione di un concetto (es: lo *zero di un polinomio* si può definire riconducendosi alla sua *divisibilità*, con il criterio del punto 13). Occorre peraltro avvertire che in matematica vi sono altri usi anche della parola *caratteristica* (per esempio, la caratteristica di una matrice non è una proprietà caratteristica!).

4) *Assiomi* di Peano, *principio* di induzione

Nella scuola superiore giustamente si suppone che tutti sappiano *che cosa sono* i numeri naturali: si è fatta la loro conoscenza nella scuola elementare semplicemente imparando i loro nomi a partire da *1*; l'apprendimento è avvenuto in modo che, se il maestro/la maestra ne nomina uno, tutti gli scolari sanno il *nome* del suo *successivo.* Questa nozione è così primitiva da meritare il termine *principio* di induzione. Effettivamente, l'insegnante ha utilizzato, più o meno coscientemente, uno dei famosi *assiomi* (= *postulati*) di Peano: ogni numero naturale (salvo *1*) si può pensare come successivo del successivo … del successivo di *1*. Così i numeri vengono tutti chiamati all'appello. Se si precisa che in questo appello nominale ogni numero, una volta chiamato, non compare mai più (questo è un altro degli assiomi di Peano), l'insieme *N* rimane individuato.

5) Ancora proprietà delle operazioni: *postulati* o *teoremi*?

Mettiamoci ancora nell'ambiente *N* dei numeri naturali e chiediamoci: le familiari *proprietà* delle operazioni (di cui abbiamo parlato nel punto 2) sono *teoremi* o *postulati*? Qui la risposta dipende dall'impostazione che si vuol dare all'argomento. Se i numeri naturali sono quelli del punto 4, manca ancora da *strutturare N*, cioè occorre introdurvi le operazioni + e x. L'insegnante elementare sa come farlo; ovviamente, senza dimostrazioni. Ma se qualcuno avesse la pazienza di ripercorrere con attenzione tutti i passaggi, si accorgerebbe che alcune proprietà devono essere postulate mentre altre si potrebbero derivare come *teoremi*. Non è qui il caso (e tanto meno nella scuola) di entrare in questi dettagli. Piuttosto, osserviamo che, in alternativa a quanto fa l'insegnante elementare, potremmo procedere così (ma è soltanto un esercizio, non un consiglio didattico!):

(a) *convenire* che, per ogni coppia di numeri $m, n,$ i simboli $m + n$, $m \times n$ indichino numeri naturali;

(b) decidere il comportamento del numero *1*, *convenendo* che il successivo di n (per ogni n) si indichi con $n + 1$ e inoltre che il prodotto $n \times 1$ sia n;

(c) *postulare* che le due operazioni siano associative, commutative e la seconda si distribuisca sulla prima.

Si potrebbe dimostrare allora (in realtà basterebbe postulare assai meno) che le due operazioni restano definite univocamente, e sono proprio quelle che tutti conosciamo.

6) *Regole* dei segni, *leggi* di cancellazione

È bene aver presenti queste due impostazioni anche quando, dopo i naturali, si introducono lo zero e gli interi negativi. Normalmente, si fa la prima conoscenza di 0 e dell'opposto $-n$ del numero n imponendo (*postulando*) che esistano e che ad essi si estenda l'operazione di addizione secondo le *regole*:

(i) $n + 0 = n = 0 + n$ $\qquad$ $n + (-n) = 0 = (-n) + n = 0$

Un po' più avanti si insegna il loro comportamento moltiplicativo con le *regole*:

(ii) $m \times 0 = 0 = 0 \times m$ $\qquad$ $m \times (-n) = (-m) \times n = - m \times n$ $\qquad$ $(-m) \times (-n) = m \times n$

Nulla osta (e di fatto lo si fa, a livello elementare) pensare a tutte queste relazioni come a *norme operative* che vengono semplicemente imposte (*regole*). Tuttavia è istruttivo, come esercizio o semplicemente come … gioco, impostare il problema altrimenti: si postula che anche nell'insieme dei numeri interi (come nei naturali) esistano operazioni +, x per cui valgano le solite proprietà. Si suppone inoltre che valgano le (i) (che diventano *relazioni di definizione* o *postulati*) e da queste si cerca di derivare le relazioni (ii) (*teoremi*). Si ottiene, come primo teorema, la *legge* (meglio: *proprietà*) di *cancellazione* additiva:

da $m + a = n + a$ $\qquad$ segue $\qquad$ $m = n$

Infatti sommando $-a$ a entrambi i membri e usando la proprietà associativa si ha

$m = m + 0 = m + (a + (-a)) = (m + a) + (-a) = (n + a) + (-a) = n + (a + (-a)) = n + 0 = n$

Come secondo *teorema* si trova che ogni numero è l'opposto del suo opposto:

$0 = -(-n) + (-n) = 0 = n + (-n) = 0$ da cui, cancellando, $-(-n) = n$.

Quanto alla moltiplicazione, usando la distributività troviamo

$a \times n + 0 = a \times n = (a+0) \times n = a \times n + 0 \times n$ $\qquad$ da cui, cancellando, $\qquad$ $0 \times n = 0$

e, analogamente, si troverebbe $n \times 0 = 0$.

Quanto alle regole dei segni, calcoliamo

$-m \times n + m \times n = 0 = 0 \times n = (-m+m) \times n = (-m) \times n + m \times n$ da cui, cancellando, $-m \times n = (-m) \times n$

e, analogamente, partendo da $0 = m \times 0$ si perviene a $-m \times n = m \times (-n)$.

Infine, mettendo assieme le precedenti, si ha anche $(-m) \times (-n) = -(m \times (-n)) = -(-m \times n) = m \times n$.

Così tutte le proprietà (ii) sono diventate teoremi. È bene osservare che esistono altre proprietà, anche ben note, che *non* si possono dedurre in questo modo; per esempio, nei soliti insiemi numerici vale la *legge* di cancellazione moltiplicativa

se $a \neq 0$ $\quad$ da $m \times a = n \times a$ segue $m = n$

ma sarebbe inutile tentare di derivarla dalle i), ii) e dalle solite proprietà delle operazioni.

N.B. Chi conosce la cosiddetta *aritmetica modulare*, cioè sa operare con le *classi di resti*, consideri, per esempio, l'*anello Z/4Z* , in cui calcolerà $1 \cdot 2 = 2 = 3 \cdot 2$. Dunque il fattore *2* non si può cancellare: si è forse violata una *legge*? Si osservi che in queste strutture (anelli di classi di resti) viene a mancare una proprietà fondamentale dei numeri naturali: chiamando gli elementi all'appello, $n+1$ dopo n, li ritroviamo più di una volta.

7) Il *principio* di conservazione delle proprietà formali

Più generalmente, gli *ampliamenti* delle strutture numeriche (cioè la costruzione progressiva: naturali *N*, interi *Z*, razionali *Q*, reali *R*) viene realizzata in modo che:

i) il nuovo insieme di numeri contenga come sottoinsieme il vecchio (questo è il significato della parola *estensione* = *ampliamento*; qualcuno, più preciso, potrà pensare che nel nuovo c'è solo una *fotocopia* del vecchio);

ii) le nuove operazioni estendano (o *prolunghino*) quelle che c'erano nel vecchio ambiente, cioè: nel nuovo ambiente

con i vecchi numeri si deve poter operare come si faceva prima;
iii) le principali *proprietà* che avevano le operazioni nel vecchio ambiente siano *conservate* nel nuovo ambiente (è quanto abbiamo postulato in 5).
Lo *scopo* di un ampliamento è poi che
iv) nel nuovo ambiente ci siano possibilità nuove, cioè si possano risolvere problemi che non erano risolubili nel vecchio (per es. equazioni del tipo $a + x = b$, $ax = b$, $x^2 = b$).
Adottare il *principio di conservazione delle proprietà formali* significa imporre la condizione iii). Chiamarlo *principio* è dunque abbastanza opportuno: significa appunto che nella nuova struttura abbiamo rinunciato a dimostrare certe proprietà: imponiamo noi, a priori, che valgano (resta da dimostrare, semmai, che questa pretesa è realizzabile).

8) *Proprietà* delle potenze: postulati, definizioni o teoremi?
Se si insiste con la precedente impostazione, tutte le proprietà delle potenze
a) vanno dimostrate in *N*;
b) diventano *postulati* nelle strutture che estendono *N*, come *Z*, *Q* e *R*
Per esempio, se (come si è fatto in 2)) scriviamo la potenza x^y come risultato di un'operazione *, le proprietà $x*1 = x$ $x*(y+z) = (x*y)(x*z)$ si possono
a) dimostrare in *N* , partendo dalla proprietà associativa della moltiplicazione, che consente di togliere le parentesi nel prodotto *(nnn …)(nnn …)*.
b) imporre in *Z*, *Q* e *R* con il principio di conservazione.
Allora possiamo scrivere

$$(x*y)\ 1 = x*y = x*(y + 0) = (x*y)(x*0)$$

e cancellando il fattore $(x*y)$ (se non nullo) concludiamo che vale $x*0 = x^0 = 1$.
Inoltre, scrivendo

$$1 = x*0 = x*(1 + (-1)) = (x*1)(x*(-1))$$

siamo costretti ad attribuire alla scrittura x^{-1} (non per convenzione) il significato di inverso di *x*. Questo però non vale se $x = 0$, e infatti non solo non siamo costretti a scrivere $0^0 = 1$, ma è addirittura opportuno (come si vede in altro contesto) non farlo nemmeno per *convenzione*.

ALGEBRA

9) *Definizione* di polinomio
In questo argomento (soggetto?) è facile trovare nei libri di testo qualche incoerenza. Alcuni incominciano con la nozione di *monomio*, parlano poi di monomi *simili*, e i polinomi si introducono come *somme* di monomi non simili. Ma se si fa così ci si trova presto in difficoltà: non sarebbero polinomi scritture come *2*, oppure *2x*, che non sono somme; e nemmeno $2x - 3x$ perché i due addendi sono simili ecc. Queste difficoltà sono facilmente rimediabili, ma ce ne sono altre che richiedono una certa attenzione. Anzitutto si dovrebbe sempre parlare di polinomi in ... (qualche *indeterminata*, per esempio *x*) a coefficienti in ... (qualche ambiente, per esempio *Z*), altrimenti la nozione è equivoca. Per esempio, una scrittura del tipo $y^3 - axy + logx + a^2 x^{1/2}$ non è un polinomio in *x*, ma diventa un polinomio in *a*, a coefficienti *reali* non appena si precisi che i simboli *x* e *y* indicano i numeri *2* e *3*. Introdurre l'argomento *polinomi* è di fatto più difficile di quel che si crede.
Forse vale la pena di allontanarsi un po' dalla pratica scolastica per descrivere una definizione di polinomio come la darebbe un algebrista moderno. Questa impostazione, se non è didatticamente raccomandabile per gli studenti, può forse servire all'insegnante per chiarirsi le idee.
Supponiamo dunque di disporre di un *anello* ***A***, cioè di un insieme i cui elementi si sappiano sommare e moltiplicare con le solite proprietà. Prendiamo, per esempio, gli interi *Z* (non occorre infatti che gli elementi di ***A*** siano invertibili). Chiamiamo provvisoriamente *polinomio a coefficienti in* ***A*** la *successione dei suoi coefficienti*, cioè una successione *a* di elementi a_i di ***A***, i quali, da un certo indice in poi, siano tutti nulli

$$a_0 \quad a_1 \quad a_2 \quad ... \quad ... \quad a_{n-1} \quad a_n \quad 0 \quad 0 \quad 0 \quad ...$$

Se a_n è l'ultimo coefficiente non nullo, chiamiamo *grado* del polinomio *a* il numero *n* (se tutti i coefficienti sono nulli, siamo in presenza del *polinomio nullo*, cui non si attribuisce grado). Si dirà: dov'è sparita la *x* , che normalmente ci aspettiamo di vedere in un polinomio? Con questa impostazione non abbiamo bisogno di scriverla esplicitamente. Vediamo perché. Tutti riconosceranno la corrispondenza tra le nostre successioni e le notazioni usuali, che qui illustriamo con qualche esempio:

$3x^3 + 5x - 6$	-6	5	0	3	0	0	0	...
$2x^4 + 3x^5 + 2x$	0	2	0	0	2	3	0	...
x	0	1	0	0	0	0	0	...
-17	-17	0	0	0	0	0	0	...

Così x è nato da sé, come uno dei tanti polinomi (gli daremo più tardi un ruolo un po' speciale). In questa definizione di polinomio è implicito che, affinché un polinomio b

b_0 b_1 b_2 ... ... b_{m-1} b_m 0 0

sia uguale al precedente a, occorre e basta che i coefficienti coincidano a due a due. In particolare, devono avere lo stesso grado $m = n$ e deve essere

$a_0 = b_0$ $a_1 = b_1$ ... ... $a_{n-1} = b_{n-1}$ $a_n = b_n$

Dunque non occorre scomodare alcun *principio* di identità per dire che "due polinomi coincidono (se e) solo se hanno gli stessi coefficienti". È implicito nella loro *definizione.*

10) Operazioni con i polinomi: *definizioni* o *teoremi*?

Occorre naturalmente capire in che modo la definizione del punto 9 conduca esattamente al familiare concetto di polinomio. Intanto introduciamo nell'insieme di queste successioni un'addizione e una moltiplicazione. Come *somma* $a + b$ di due polinomi prenderemo la successione

$a_0 + b_0$ $a_1 + b_1$ $a_2 + b_2$ ... ... $a_i + b_i$ ...

il cui grado risulterà minore o eguale ai gradi di a e b. È evidente che questa operazione eredita le proprietà (associativa e commutativa) dell'addizione in $\boldsymbol{A}$. Definiremo la moltiplicazione tra polinomi in un modo apparentemente più artificiale: come prodotto ab prenderemo la successione

a_0b_0 $a_0b_1 + a_1b_0$ $a_0b_2 + a_1b_1 + a_2b_0$ .... $a_0b_r + a_1b_{r-1} + ... + a_rb_0$ ...

Per scrivere il coefficiente di posto r-esimo vogliamo cioè sommare tutti prodotti a_ib_j per cui $i+j=r$. Si potrebbe a questo punto dimostrare (*teorema*) che questa moltiplicazione risulta associativa, commutativa e distributiva sull'addizione; è un compito pesante e poco istruttivo, ma ha il pregio di far vedere che l'anello dei polinomi a coefficienti in $\boldsymbol{A}$ è univocamente individuato dai suoi coefficienti, senza far intervenire alcun simbolo x (indeterminata? variabile?).

[N.B. Queste definizioni formali sembrano arbitrarie fintantoché non si verifica la relazione tra polinomi e funzioni polinomiali (vedi 11). Un'analoga situazione si crea quando si definisce il prodotto di due matrici – righe per colonne – e solo più tardi si scopre che questa *regola* apparentemente arbitraria interpreta la composizione di due funzioni, e dunque è quasi imposta dall'uso che vogliamo farne.]

Sappiamo tutti che in realtà, nella tradizione e soprattutto nella scuola, i polinomi si introducono in un altro modo, che, guarda caso, assomiglia al procedimento alternativo che abbiamo adottato nei punti 5, 6, 7 per ampliare le strutture numeriche. Infatti, molto grosso modo, di solito si procede così:

1) si inventa un ambiente $\boldsymbol{A}[x]$ che è un ampliamento di $\boldsymbol{A}$ (gli elementi di $\boldsymbol{A}$ sono i polinomi di grado zero) e in cui vi sono un'addizione e una moltiplicazione che godono delle solite proprietà (*principio* di conservazione delle proprietà formali). Tra i nuovi elementi, ce n'è uno che si chiama x; si suppone (in realtà si richiede, *si postula*) che questo elemento goda di proprietà particolari, che sono le seguenti:

2) tutti gli elementi di $\boldsymbol{A}[x]$ si ottengono facendo, in tutti i modi possibili, somme e prodotti di elementi che sono o elementi di $\boldsymbol{A}$ oppure x. Con opportuni accorpamenti, si vede allora che ogni elemento di $\boldsymbol{A}[x]$ si scrive nella solita forma $a_0 + a_1x + a_2x^2 + ... + a_nx^n$.

3) (Questa proprietà ha un nome importante: *trascendenza*!) L'elemento $a_0 + a_1x + a_2x^2 + ... + a_nx^n$ non coincide mai con lo zero, a meno che non siano nulli tutti i suoi coefficienti.

È evidente che applicando 1), cioè comportandoci come se x fosse un numero, possiamo applicare le proprietà distributive ecc. Il fatto è che, se abbiamo la pazienza di fare i calcoli, otteniamo come risultato finale esattamente quello che nel punto 10 abbiamo stabilito per le successioni. Per esempio, per il prodotto $(2x^4+x^2-x)\,(3x^3+5x-6)$, applicando la distributività e accorpando i monomi si ottiene $6x^7 + \ldots + 5x^3 - 11x^2 + 6x + 0$. Con l'impostazione del punto 10 scriviamo

$a = 2x^4 + x^2 - x$ 0 -1 1 0 ...

$b = 3x^3 + 5x - 6$ -6 5 0 3 ...

$ab = 0\cdot6$ $0\cdot5+(-6)\cdot(-1)$ $0\cdot0+(-1)\cdot5+1\cdot(-6)$ $0\cdot3+(-1)\cdot0+1\cdot5+0\cdot(-6)$

$= 0$ 6 -11 5

Ancora una volta, le *regole* secondo cui moltiplicavamo i polinomi nel paragrafo precedente (compresa l'addizione dei monomi simili) invece che *definizioni* sono diventati teoremi.

La proprietà 3) ha un' immediata conseguenza: due elementi $\boldsymbol{A}[x]$ coincidono $a_0 + a_1x + a_2x^2 + ... + a_nx^n = b_0 + b_1x + b_2x^2 + ... + a_nx^n$ se e solo se sono eguali tutti i loro coefficienti: $a_i = b_i$ per ogni indice i.

Problema da studiare: è vero che le proprietà 2), 3) sono *caratteristiche* di x, oppure esiste qualche altro elemento di $\boldsymbol{A}[x]$ che le soddisfa?

11) *Principio* di identità o *corollario*?

Non è da pensare che la nozione di polinomio che abbiamo appena descritta sia quella che nel passato andava per la

maggiore: fino a una certa epoca storica, un polinomio $a = a_0 + a_1x + a_2x^2 + ... + a_nx^n$ a coefficienti in $\boldsymbol{A}$ veniva tranquillamente identificato con la *funzione*, chiamiamola $a\hat{}$, di $\boldsymbol{A}$ in sé, che ad ogni elemento u di $\boldsymbol{A}$ associa l'elemento $a\hat{}(u) = a_0+a_1u + a_2u^2 + ... + a_nu^n$. Una funzione di questo tipo si chiama *polinomiale* ovvero *razionale intera.* Polinomi e funzioni polinomiali sono nozioni diverse: la prima è più di pertinenza dell'Algebra, la seconda dell'Analisi. Tuttavia, come vedremo, la distinzione diventa importante soltanto in ambienti numerici un po' ... esotici. Come ben dice il Villani, la prevalenza dell'Analisi sull'Algebra (se non altro la sua maggiore età) spiega ampiamente una certa diffusa confusione. Evidenziamo le connessioni e le differenze:

i) Per capire come le due nozioni siano strettamente connesse, osserviamo che anche in Analisi si introducono somme e prodotti di funzioni (quando scriviamo, per es. $sin(a+x) = sinacosx + \ldots$ intendiamo una relazione tra funzioni, cioè che quella formula valga per tutti gli x reali; e i polinomi non c'entrano). Ed è facile convincersi che alle somme e prodotti nell'ambiente dei polinomi corrispondono analoghe somme e prodotti per le relative funzioni: $(a+b)\hat{} = a\hat{} + b\hat{}$, $(ab)\hat{} = a\hat{}b\hat{}$.

[N.B. È questo che giustifica – tra le tante possibili – la scelta delle definizioni del punto10. In effetti, i prodotti tra successioni si potrebbero definire molto più semplicemente, sempre conservando le solite proprietà delle operazioni; ma ne risulterebbe un anello meno interessante del nostro.]

ii) Per ribadire la loro differenza concettuale, basterà dire che:

il polinomio $a = a_0 + a_1x + a_2x^2 + ... + a_nx^n$ è nullo, per definizione, solo se è tutta nulla la successione dei suoi coefficienti. Viceversa, per particolari scelte di $\boldsymbol{A}$ e per particolari scelte del polinomio a, può succedere che la funzione $a\hat{}$ sia nulla (cioè $a\hat{}(u) = a_0 + a_1u + a_2u^2 + ... + a_nu^n = 0$ per ogni scelta di u in $\boldsymbol{A}$) anche se alcuni dei suoi coefficienti a_i sono diversi da zero. Come conseguenza più generale, ci possono essere polinomi diversi $c \neq d$ che danno luogo alla medesima funzione: infatti, se $a = d-c$, da $a\hat{} = 0\hat{}$ segue $d\hat{} = c\hat{}$. Questa eventualità va però considerata molto remota, almeno dal punto di vista scolastico. Possiamo convincerci infatti che non si verifica mai se $\boldsymbol{A}$ è una delle strutture numeriche importanti: interi Z, razionali Q, reali R. Basterà ricordare il *teorema degli zeri* (che riprenderemo nel punto 13); premessa la *definizione*:

u (un elemento di $\boldsymbol{A}$) è uno *zero* (= *radice*) di a (un elemento di $\boldsymbol{A}[x]$) significa $a\hat{}(u) = 0$

il teorema degli zeri afferma:

ogni valore u che sia zero del polinomio a produce una fattorizzazione $a = (x-u)b$.

(Il viceversa è ovvio) Qui il grado di b si è abbassato rispetto a quello di a. Perciò, ripetendo il procedimento su b (i dettagli sono ben spiegati nel libro di Villani) si vede che il numero di zeri della funzione $a\hat{}$ non può superare il grado del polinomio a. Si conclude che, se $\boldsymbol{A}$ contiene infiniti elementi (e questo è il caso usuale), nessun polinomio a (che non sia nullo) può dar luogo alla funzione nulla. Ne consegue il *corollario*:

se $\boldsymbol{A} = Z$, oppure Q, oppure R, due polinomi diversi producono due funzioni polinomiali diverse.

Questo enunciato viene chiamato nei testi classici *principio*, e da Villani *teorema di identità dei polinomi*. Che non si tratti di un nuovo assioma è certo, e come tale il termine *principio* non è adatto e quello di *teorema* sì. Non sono così certo, tuttavia, che davvero ci si riferisca sempre a questo enunciato quando, in certi libri di testo, lo si cita come *principio*. Ho l'impressione che qualche autore lo faccia per indicare un fatto, in un certo senso, molto più semplice, e cioè che, costruendo i polinomi nel modo tradizionale (cioè come nel punto 10, non come nel punto 9), all'elemento x occorre attribuire la proprietà 3), che abbiamo chiamato *trascendenza*. Se la mia impressione fosse corretta, il termine *principio* sarebbe abbastanza giustificato, trattandosi di un'ipotesi che si fa a priori su un elemento x (della cui esistenza non ci si preoccupa più di tanto).

È certo che l'intero argomento risulta piuttosto difficile, e temo che sia la poca chiarezza dei concetti che dà luogo a imperfezioni di linguaggio. E talvolta anche di ragionamento: ho trovato infatti testi in cui dapprima si descrive correttamente la differenza tra polinomi e funzioni polinomiali, ma poi si ricorre a quell'enunciato (quello sulle funzioni) a sproposito, facendolo intervenire quando non ce ne sarebbe bisogno.

.

12) Divisione (con resto) tra polinomi: perché non si chiama *teorema*?

Una delle più importanti *proprietà* dei polinomi, per la quale invece, nella tradizione scolastica, non si scomoda alcun termine importante, è certamente il *teorema della divisione* (con resto), il cui enunciato è il seguente:

Se a , b sono polinomi a coefficienti in un *campo* $\boldsymbol{A}$ (cioè in $\boldsymbol{A}$ ogni elemento non nullo è invertibile, come avviene in Q o R) e $b \neq 0$, esistono due polinomi q (quoziente) e r (resto) a coefficienti in $\boldsymbol{A}$ tali che $a = bq + r$ dove il grado di r è minore di quello di b, oppure $r = 0$.

Non sempre questo fatto viene evidenziato come *teorema*: spesso si insegna che "per dividere due polinomi si fa così e così": una sorta di *regola* esecutiva. Ma non mettersi nell'ordine di idee di *dimostrarlo* impedisce di accorgersi di alcuni fatti importanti:

(i) l'enunciato ha un analogo in Z (la divisione con resto negli interi), e l'analogia si spinge molto avanti: quando vale questo teorema si parla di anelli *euclidei* e si può applicare l'algoritmo di Euclide (vedi 14);

(ii) il teorema vale (cioè la divisione "si può fare") anche quando i coefficienti di a, b non appartengano a un campo, purché sia invertibile (in $\boldsymbol{A}$) il coefficiente di grado massimo del polinomio b (in Z, per esempio, sia 1 oppure -1);
(iii) il teorema non si generalizza ai polinomi in più indeterminate (un argomento di cui qui non parliamo, ma il Villani vi dedica un bel capitolo): in effetti, se per grado complessivo di un polinomio si prende l'accezione comune, può accadere che q e r non si trovino.

13) *Teorema* (o *corollario*?) di Ruffini, *criterio* di divisibilità

Il grande matematico modenese non sarebbe molto contento se sapesse che il suo nome resta associato - nella cultura corrente - a una regola abbastanza banale, piuttosto che ai suoi risultati profondi, questi sì di importanza storica, sulla *risolubilità per radicali* delle equazioni algebriche. Fortunatamente, in Italia si parla anche di un *teorema* di Ruffini, detto anche *teorema del resto*; ma anche questo enunciato, in verità, non sembra adeguato a un grande nome. Eccolo:
(1) nella divisione di a (elemento di $\boldsymbol{A}[x]$) per $x-u$ il resto è $a\hat{}(u)$.
Nei libri, dopo poche righe, si trova il seguente *criterio* di divisibiltà, che poi è il *teorema* degli zeri (vedi 11):
(2) l'elemento u (di $\boldsymbol{A}$) è uno zero del polinomio a se e solo se a è divisibile (in $\boldsymbol{A}[x]$) per $b = x-u$.
Ora è evidente che (2) è un *corollario* di (1), anzi è un suo caso particolare perché, per *definizione*, "u è uno *zero* di a" significa $a\hat{}(u) = 0$ (vedi 11). D'altra parte, (1) è un *corollario* del teorema della divisione. Infatti da $a = bq + r$, possiamo senz'altro dedurre (senza invocare alcun *principio*: questa è una delle sviste che si ritrovano nei testi) l'analoga relazione per le funzioni polinomiali $a\hat{} = b\hat{}q\hat{} + r\hat{}$, che significa $a\hat{}(u) = b\hat{}(u)q\hat{}(u) + r\hat{}(u)$ per ogni elemento u *di* $\boldsymbol{A}$. In particolare, quando $q = x-u$, è evidentemente $q\hat{}(u) = u-u = 0$. Da cui $a\hat{}(u) = r\hat{}(u)$.

14) La *regola* di Ruffini, l'*algoritmo* euclideo

Gli studenti piu' impulsivi, di fronte a un problema matematico, hanno la cattiva abitudine, prima di capirne la sostanza, di chiedersi "come si fa?". Nulla di più ghiotto, dunque, di una ricetta meccanica che conduce alla risposta senza richiedere giustificazioni. Tale è la famosa regola di Ruffini, che in Italia gode di grande popolarità e in altri paesi porta il nome di *algoritmo* di Horner. Giustamente il Villani sostiene che attribuirle troppa importanza è fuorviante, perché normalmente se ne può fare a meno, sostituendo quel procedimento con altri più istruttivi. Senza ripetere i suoi ottimi argomenti, mi limiterò a dire che anche per questa regola, se la si vuole usare, andrebbe predisposta una qualche dimostrazione. Per usare, per esercizio, l'impostazione dei punti 9, 10, basta osservare che scrivendo $a, b, x-u$ come successione di coefficienti

a	a_0	a_1	a_2	...	...	a_{n-1}	a_n	0	...
$x-u$	$-u$	1	0	0	0	0	0	0	...
b	b_0	b_1	b_2	...	...	b_{n-1}	0	0	...

la relazione $a = (x-u)b$ comporta

$a_0 = -ub_0 + r$ $\quad a_1 = -ub_1 + b_0\,1$ $\quad a_2 = -ub_2 + b_1\,1 + 0b_0$ $\quad ...$

$\quad ...$ $\quad a_{n-1} = -ub_{n-1} + b_{n-2} + ...\,0$ $\quad a_n = 0... + b_{n-1}$

da cui, senza bisogno di preparare il famoso schema con le righe incrociate ecc. si ottiene la precisa descrizione della *regola*:

$b_{n-1} = a_n$ $b_{n-2} = a_{n-1} + ub_{n-1}$ $\quad ...$ $\quad b_1 = a_2 + ub_2$ $\quad b_0 = a_1 + ub_1$ $\quad r = a_0 + ub_0$.

Dobbiamo dunque eliminare la regola di Ruffini dai nostri programmi? Ridimensionarla sì, eliminarla forse no. Infatti, ci piaccia o non ci piaccia, in tempi recenti sta prendendo sempre maggior importanza l'*algoritmica*, che consiste, in un certo senso, nell'arte di insegnare un modo efficiente di procedere anche a un essere non raziocinante, qual è il *calcolatore*. Scegliere un buon algoritmo per realizzare un calcolo e scriverlo in modo che il computer lo accetti è un'arte raffinata, che richiede una cura particolare. Come ben sappiamo, basta infatti un piccolo errore di notazione (non di concetto!) per mandare in crisi il calcolatore. Questa attenzione, io credo, è facilitata dagli schemi grafici come quello della nostra *regola*, i quali rendono appunto automatico un procedimento, sostituendolo a un ragionamento più generale. In fin dei conti, anche la nostra scrittura dei polinomi come successioni (vedi 9) e le *regole* per sommarli e moltiplicarli (vedi 10) si possono interpretare come espediente grafico per evitare gli errori che potrebbero nascere – usando le scritture tradizionali con la x – nell'applicazione delle proprietà distributive ecc. D'altra parte, esistono *procedimenti* (per es. è ben noto quello delle divisioni successive, detto *algoritmo euclideo*) che hanno profondo interesse matematico e completano molto opportunamente la comprensione di concetti (il *massimo comun divisore*) originariamente definiti mediante *proprietà caratteristiche*, senza ricorrere a procedimenti di calcolo. In realtà, anche preparare lo studente alla mentalità algoritmica - essenziale nell'informatica - può essere un nobile scopo didattico. Che tuttavia ci allontana dallo spirito del libro *Cominciamo da Zero*: il maggior pericolo per lo studio della matematica rimane la cattiva abitudine a non porsi tanti "perché".

I sistemi di calcolo algebrico ieri e oggi

di Giulio Cesare Barozzi
Università di Bologna

Gli elaboratori elettronici, o calcolatori come più comunemente si dice, si sono presentati alla ribalta poco più di mezzo secolo fa, e hanno trovato, almeno inizialmente, impieghi in campo militare, scientifico e gestionale. Essi servirono per calcolare le tavole di tiro dei cannoni, per gestire in modo ottimale i depositi militari, poi entrarono nei vari rami dell'ingegneria e della fisica, fino a trattare problemi di grande complessità come le previsioni meteorologiche. In un secondo tempo essi trovarono impiego nelle banche e nei centri statistici, nei punti di vendita, fino ad entrare in ogni aspetto dell'attività umana con l'esplosione di Internet. In tutti gli impieghi che abbiamo citato i calcolatori sono essenzialmente degli "schiaccia numeri".

L'idea di utilizzare i calcolatori per facilitare i calcoli algebrici è più recente, e può essere collocata intorno agli anni '70. Non a caso, i primi che sentirono la necessità di trovare uno strumento di calcolo algebrico furono i fisici teorici, gli astronomi e i relativisti, tutti quei ricercatori che, per esigenze di lavoro, dovevano sobbarcarsi pesanti calcoli algebrici, dove l'aggettivo *algebrico* va inteso in un'accezione ampia.

Nel febbraio del 1982 la rivista *Le Scienze* (traduzione italiana di *Scientific American*) pubblicò un articolo di notevole interesse, poi raccolto, insieme ad altri, nel Quaderno de Le Scienze n. 14, dal titolo *Matematica e calcolatore*, a cura di G. Lolli e C. Mangione. Il titolo era semplicemente

Algebra al calcolatore
autori: R. Pavelle, M. Rothstein e J. Fitch

tutti matematici con inclinazione verso la fisica teorica e l'informatica. La lettura di questo articolo, a distanza di oltre vent'anni, è oltremodo istruttiva. Si spiega innanzitutto che cosa sia un sistema di calcolo algebrico. Potremmo dire sbrigativamente: un sistema che riesca ad automatizzare un calcolo del tipo

$$(a - b)(a + b) = a^2 - b^2,$$

ed anche la fattorizzazione in senso inverso. I linguaggi procedurali tradizionali (Pascal, Fortran, C, …) non sono in grado di realizzare ciò. Possono certamente eseguire i prodotto $(a - b)(a + b)$, ma a patto che alle variabili a e b sia assegnato un valore, ad esempio un valore reale.

Vediamo più da vicino che cosa accade quando scriviamo un programma del tipo

```
…
a = 3.1
b = 2.5
c = (a - b)(a + b)
PRINT c
…
```

In realtà non appena si digita una lettera, ad esempio la lettera *a*, il sistema provvede a compilare una riga in un'immaginaria tabella, in cui alla lettera *a* si associa un puntatore ad una locazione di memoria, e in tale locazione si scrive il valore 3.1. Possiamo immaginare una situazione del genere:

variabile	indirizzo	valore
…	1002	…
a	1003	3.1
b	1004	2.5
c	1005	3.36
…	1006	…

Al valore *a* viene associato l'indirizzo 1003 e nella corrispondente locazione di memoria si scrive il numero 3.1, al valore *b* viene associato l'indirizzo 1004, e nella corrispondente locazione viene scritto il valore 2.5. Quando viene richiesta l'esecuzione dell'istruzione $c = (a - b)(a + b)$ vengono letti i valori che stanno negli indirizzi corrispon-

denti alle variabili *a* e *b* e il risultato calcolato dall'unità aritmetica viene memorizzata nella cella 1005, corrispondente alla lettera *c*. Da questa cella il valore calcolato verrà prelevato dall'istruzione successiva (PRINT *c*) per essere esibito dall'unità che mostra i dati in uscita, sia essa lo schermo o la stampante.

L'istruzione $c = (a - b)(a + b)$ non può essere eseguita se alle variabili *a* e *b* non è stato preventivamente assegnato un valore.

I sistemi di calcolo algebrico, al contrario, non necessitano che alle variabili sia preventivamente assegnato un valore, anche se ciò non è proibito, vale a dire sono in grado di scrivere in una forma semplificata il prodotto $(a - b)(a + b)$, dove *a* e *b* sono dei puri simboli, e anche di fare il passaggio in senso inverso. Tipici comandi al riguardo potrebbero essere

EXPAND($(a - b)(a + b)$)

e

FACTOR($a^2 - b^2$)

Con riferimento alla tabella precedente, potremmo dire che, in assenza di un valore numerico, una variabile ha come "valore" il proprio nome:

variabile	indirizzo	valore
…	1002	…
a	1003	*a*
b	1004	*b*
c	1005	*c*
…	1006	…

Lo sviluppo dei sistemi di calcolo algebrico è stato lento ma costante; l'articolo di Pavelle et al. elenca i quattro principali sistemi esistenti all'inizio degli anni '80: Macsyma, Reduce, Scratchpad, SMP. Nessuno di essi sopravvive oggi, anche se SMP fu il precedecessore di *Mathematica*. Come unico sistema in grado di essere utilizzato su un PC si parla di muMATH: questo è l'antenato di DERIVE. A proposito di muMATH, ecco che cose scrivono Pavelle e colleghi nel 1981:

"Il linguaggio muMATH presenta alcune capacità tipiche dei grandi sistemi, anche se le dimensioni, la memoria e la velocità del microcalcolatore non gli consentono di affrontare problemi complessi. Comunque, prima della fine di questo decennio, questi sistemi (o loro discendenti più potenti) probabilmente raggiungeranno i microcalcolatori e forse addirittura i piccoli calcolatori tascabili." Raramente profezia è stata più azzeccata!

Inizialmente i sistemi di calcolo algebrico erano veramente tali, cioè si limitavano ad automatizzare le operazioni tipiche dell'algebra e dell'analisi: fattorizzazioni e sviluppi, calcolo di derivate e integrali, sviluppi in serie di Taylor, ecc. Essi non disponevano di capacità grafiche e non erano particolarmente attenti alle esigenze del calcolo numerico.

Per dare un esempio dei primi tipi di utilizzo leggiamo ancora nell'articolo sopracitato la storia di un matematico e astronomo francese, Charles Delaunay, che, nel diannovesimo secolo, aveva speso vent'anni della propria vita professionale per calcolare piccole variazioni dell'orbita lunare dovute a perturbazioni dell'angolo tra il piano orbitale della Luna e il piano orbitale della Terra, variazioni dovute a vari influssi. Questo Delaunay pubblicò i propri risultati nel 1867 ma, ovviamente, nessuno si prese la briga di spendere altri 20 anni di lavoro per verificarli.

L'argomento trattato da Delaunay dormì sonni tranquilli fino allo scoppio della seconda Guerra Mondiale, quando il problema tornò a essere interessante in connessione con la caccia ai micidiali sottomarini tedeschi U-Boote, che utilizzavano sistemi di determinazione delle propria posizione basati su metodi astronomici. Più recentemente il problema del corretto posizionamento e dell'inseguimento dei satelliti articifiali ha ridato vita al problema studiato da Delaunay.

Utilizzando Macsyma fu possibile verificare in 20 ore di calcolo (nel 1973) i calcoli di Delaunay e scoprire che aveva fatto solo un piccolo errore, fortunatamente con scarse conseguenze.

L'articolo di Pavelle et al. si concludeva con questa frase: "A mano a mano che i piccoli calcolatori di basso costo andranno perfezionandosi, l'algebra al calcolatore si renderà disponibile per l'insegnamento, per lo studio, per la ricerca e forse per molte altre applicazioni a cui non siamo ancora in grado di pensare, per tutte le persone interessate, a casa come in ufficio."

La situazione attuale è molto cambiata rispetto a quella descritta nell'articolo più volte citato. I sistemi di calcolo algebrico (in inglese CAS = *Computer Algebra Systems*) sono in realtà degli ambienti integrati di calcolo, in cui

convivono il calcolo algebrico, il calcolo numerico, la grafica, le gestione dei testi, ecc. Come tali essi valgono più della somma delle loro parti, nel senso che un ambiente unico in cui realizzare diversi compiti acquista sicuramente un valore aggiunto, quanto meno a livello didattico, anche se i singoli compiti non sono realizzati al livello di un sistema dedicato ad uno di essi.

I due sistemi che vanno per la maggiore sono *Mathematica* e Maple, a cui si aggiunge DERIVE, a un livello di minore complessità. Anche altri sistemi sono largamente diffusi, come Matlab e MathCad, ma non sono propriamente sistemi di calcolo algebrico, anche se hanno alcune capacità in tale direzione.

Il problema che si pone oggi è quello della scelta di un linguaggio di programmazione in cui realizzare quella che, ai tempi del P.N.I., si chiamava *l'educazione al pensiero algoritmico*. Tutti sono d'accordo nel cantare il *De Profundis* al Pascal, linguaggio ritenuto obsoleto; ma dalle ceneri del Pascal non è emerso un chiaro successore. Sarà uno dei linguaggi contenuti nei CAS a prendere il suo posto? DERIVE sembra essere il candidato più naturale, anche se certe limitazioni, anche banali, come l'assenza di un ambiente adeguato di editing dei listati, facevano sorgere molte perplessità su tale linguaggio, almeno fino alla versione 5. Uno degli scopi di questo convegno è di fare il punto sulla versione 6 e di valutare la fattibilità di una scelta di DERIVE come linguaggio in cui realizzare l'obiettivo a cui sopra ci riferivamo.

Altro problema: un solo linguaggio o più di uno, diciamo due, almeno nelle scuole dove ciò è possibile? Possiamo ignorare certe tendenze che sono esplose con lo sviluppo di Internet, come i linguaggi orientati alla programmazione a oggetti, Java in primis?

Io non ho una risposta, e forse nessuno ce l'ha. Lo scopo dei convegni è quello di dibattere idee, presentare soluzioni tra loro in alternativa, e di consentire agli insegnanti di operare una scelta meditata. Magari bastasse un convegno per dirimere ogni dubbio!

Quale che sia il CAS prescelto, rimane un problema di fondo, a cui solo l'esperienza didattica potrà trovare una soluzione soddisfacente. Questo problema è in qualche modo analogo a quello che si è posto con l'introduzione delle calcolatrici numeriche nella scuola elementare e media: visto che i calcoli numerici si fanno a macchina, quale livello di abilità manuale ha ancora senso esigere dai nostri allievi?

Qui il problema è in un certo senso lo stesso, ma è anche più complicato: visto che le derivate si fanno a macchina, ha senso pretendere che i nostri studenti le sappiano fare a mano? Credo che dovremo batterci perché i nostri studenti concentrino la propria attenzione sul significato di ciò che fanno più che sugli aspetti formali.

Per dirla in termini difficili: i CAS lavorano a livello sintattico (non sanno quello che fanno) e non semantico; sta all'utente dare significato a ciò che fa. In questo senso si richiede un utente più colto di quanto si richiedesse prima, anche se si può tollerare un utente meno abile sul piano tecnico. Per scavare una buca con vanga e badile occorre forza fisica, ma per scavarla con una pala meccanica occorre una notevole capacità di coordinamento visivo-motorio, anche se non servono i muscoli.

La ricerca di un punto di equilibrio tra le diverse abilità sarà lenta e graduale, ma io confido che se ci sarà un ambiente aperto alle collaborazioni e agli scambi di esperienze didattiche (e questo convegno vuol essere un'occasione al riguardo) si raggiungerà una soluzione soddisfacente.

Da al-Khwarizmi il termine algoritmo

La programmazione in *Derive*

di Paolo Boieri
Dipartimento di Matematica Politecnico di Torino

Il programma *Derive* è nato come un sistema di calcolo simbolico (vedi Rich e Stoutemyer, 1994 e Barozzi, 1990); nell'ambito dei CAS (*Computer Algebra Systems*) si è sempre contraddistinto, rispetto ai concorrenti, per le dimensioni estremamente compatte (la prima versione del 1988 era contenuta in un dischetto da 360Kb) e per le limitate risorse richieste al computer su cui gira.
Le successive versioni (vedi Cappuccio 1995, 1998, 2001) sono state arricchite con prestazioni grafiche sempre più evolute, con una interfaccia (integrata nell'ambiente Windows nella versione 4) più amichevole rispetto a quella veramente spartana delle prime versioni, con l'introduzione di nuove funzioni.
In particolare, a partire dalla versione 2, in aggiunta all'ambiente di base, in cui l'utente opera tramite l'immissione di comandi, sono state introdotte delle funzioni che consentono di scrivere dei programmi.
Lo stile di programmazione di *Derive* è diverso da quello di altri ben noti ambienti come, ad esempio, il Pascal o il C; il linguaggio di *Derive* era, fino alla versione 4, di tipo puramente funzionale; nella prima parte di questo intervento esaminiamo le caratteristiche peculiari di questo linguaggio, per renderci conto di che cosa significasse programmare in *Derive* fino alla versione 4.
A partire dalla versione 5 sono stati aggiunti alcuni costrutti che rendono lo stile di programmazione in *Derive* più vicino a quello degli altri linguaggi, senza però modificare l'impostazione funzionale originaria; ne è nato un linguaggio "misto", che presenta delle caratteristiche interessanti e che costituisce un sistema più facilmente utilizzabile anche a scopi didattici; nella seconda parte del lavoro sono esaminate le caratteristiche di questo linguaggio.

1. I prerequisiti

A differenza di quanto avviene con altri linguaggi, non si può partire direttamente dalla programmazione in *Derive* come argomento a sé stante, ma conviene arrivare ad essa dopo avere acquisito una certa conoscenza dell'ambiente dei comandi o almeno di una parte di essi.
D'altra parte i risultati più efficaci si ottengono, come vedremo in seguito, dall'uso combinato dei comandi e delle funzioni di libreria con le funzioni definite dall'utente tramite la programmazione.
Tra i prerequisiti che si possono elencare vi sono certamente le applicazioni di base di *Derive*, quali le manipolazioni algebriche sui polinomi, la soluzione di equazioni e di sistemi, la grafica di funzioni.
Vogliamo qui sottolineare due aspetti che sono di particolare importanza per il futuro lavoro sulla programmazione: si tratta dell'uso dei vettori e del meccanismo della definizione e della composizione di funzioni. Esaminiamoli attraverso un semplice esempio di geometria analitica (vedi Boieri, 2003), in cui vediamo:
come rappresentare graficamente un punto nel piano;
come scrivere una funzione per calcolare la distanza di due punti;
come scrivere una funzione per trovare l'equazione dell'asse di un segmento, assegnati i suoi estremi.

Un vettore in *Derive* è un elenco di numeri, espressioni numeriche o algebriche.
Ognuno di questi oggetti è un elemento (o componente) del vettore; il numero di elementi di un vettore è detto dimensione del vettore.
Un vettore viene introdotto scrivendo gli elementi, separati da una virgola, tra parentesi quadre; ad esempio
#1: v :=[0, 1, 2, 3, 4, 5]
definisce il vettore v i cui elementi sono i primi 6 numeri naturali.
Derive consente di moltiplicare un vettore per uno scalare e di sommare due vettori della stessa dimensione, di calcolare il prodotto scalare; vi sono poi molte funzioni di libreria per la manipolazione dei vettori. Elenchiamo quelle di uso più comune:

DIM(v)	fornisce il numero di elementi di v
v SUB k	fornisce l'elemento k-esimo di v
DELETE(v, k)	fornisce una copia del vettore v, ma con l'elemento k-esimo cancellato.
INSERT(u, v, k)	fornisce una copia del vettore v, ma con l'inserimento dell'espressione u prima dell'elemento k-esimo.
FIRST(v)	fornisce il primo elemento di v
REST(v)	fornisce una copia di v a cui è stato tolto il primo elemento

Un vettore può essere definito, oltre che assegnandone le componenti, anche tramite il comando VECTOR, uno dei più frequentemente utilizzati in *Derive*. La sua sintassi è: VECTOR(u(n), n, n_min, n_max, [passo])
VECTOR genera un vettore delle espressioni u(n) quando n varia da n_min a n_max con passo unitario (a meno che il passo non sia esplicitamente indicato come quinto argomento). Ecco alcuni esempi:

#1: VECTOR(n^2, n, 1, 10)
#2: [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
#3: VECTOR(0, n, 1, 5)
#4: [0, 0, 0, 0, 0]
#5: VECTOR(x^n, n, 1, 5, 2)
#6: [x, x^3, x^5]

Uno degli utilizzi più comuni dei vettori è nello studio della geometria analitica e delle trasformazioni geometriche (vedi Accascina e Berneschi, 1998a e 1998b): infatti, per indicare un punto del piano e ottenerne la visualizzazione nella finestra grafica di *Derive* dobbiamo scrivere il vettore delle sue coordinate e poi passare alla finestra grafica.
Consideriamo i punti $A = (x_A, y_A)$ e $B = (x_B, y_B)$; la loro distanza è data da

$$d(A, B) = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}$$

Vogliamo definire una funzione che ci consenta di calcolare questa distanza; ciascuno dei due punti è identificato da un vettore; il punto A dal vettore v, il punto B da w.
Per definire la funzione richiesta, che chiamiamo DIST, dobbiamo quindi estrarre da v e da w le componenti ed operare su di esse, utilizzando la funzione di libreria SQRT per il calcolo della radice. La funzione DIST ha come argomento una coppia di vettori e fornisce come risultato un numero reale; per utilizzarla è sufficiente richiamarne il nome, indicando tra parentesi i due vettori:

#1: DIST(v, w) := SQRT((w SUB 1- v SUB 1)^2 + (w SUB 2- v SUB 2)^2)
#2: DIST([1, 5], [4, 1])
#3: 5
#4: DIST([a, b], [-a, b])
#5: 2·|a|

La funzione DIST può essere utilizzata, nell'ambito della stessa sessione di lavoro, come le funzioni di libreria di *Derive* e può intervenire nella definizione di nuove funzioni; deve essere salvata quando si esce dal programma e richiamata quando si vuole utilizzarla in un secondo tempo.
A partire da DIST vogliamo definire la funzione ASSE(v, w); arriviamo a questa definizione per passi successivi, svolgendo gli stessi passaggi che faremmo con carta e matita.
L'asse è l'insieme dei punti del piano equidistanti da A e da B; quindi dobbiamo prima imporre che il generico punto dell'asse (che indichiamo "alla *Derive*" con [x, y]) sia equidistante da A e B; partiamo da un caso concreto (difficilmente lo studente sarebbe in grado di gestire 6 parametri!); poniamo, ad esempio, A = [1, 5] e B = [4, 1] e scriviamo la relazione:
DIST([x, y], [1, 5]) = DIST([x, y], [4, 1])
Quando la calcoliamo, vediamo che *Derive* ha valutato i due membri dell'equazione; ora risolviamo questa equazione rispetto alla variabile y, ottenendo l'equazione dell'asse.

#1: DIST(v, w) := $\sqrt{[(w_1 - v_1)^2 + (w_2 - v_2)^2]}$
#2: DIST([x, y], [1, 5]) = DIST([x, y], [4, 1])
#3: $\sqrt{(x^2 - 2\cdot x + y^2 - 10\cdot y + 26)} = \sqrt{(x^2 - 8\cdot x + y^2 - 2\cdot y + 17)}$
#4: SOLVE($\sqrt{(x^2 - 2\cdot x + y^2 - 10\cdot y + 26)} = \sqrt{(x^2 - 8\cdot x + y^2 - 2\cdot y + 17)}$, y)
#5: $y = \frac{3\cdot(2\cdot x + 3)}{8}$

A questo punto possiamo fare il grande salto: è possibile definire la funzione ASSE(v, w)?
Lo studente deve capire che i passaggi svolti sono di carattere generale e che non è necessario assegnare numericamente le coordinate di A e B, ma è possibile sintetizzare il procedimento in una sola riga, scrivendo:
ASSE(v, w) := SOLVE(DIST([x, y], v) = DIST([x, y], w), y)

2. La programmazione funzionale

Vediamo ora le caratteristiche della programmazione puramente funzionale di *Derive*, presente nella versione 4 (per un'analisi più completa vedi Boieri 1994).

Oltre alle caratteristiche viste in precedenza, che potremmo definire "propedeutiche" alla programmazione vera e propria, abbiamo a disposizione in *Derive* tre funzionalità che sono comuni a molti altri linguaggi di programmazione:

il costrutto condizionale IF;

gli operatori logici AND, OR, NOT;

la possibilità di definire funzioni ricorsive.

La caratteristica saliente della programmazione in *Derive* è invece la funzione Iterates (con la sua variante Iterate), che realizza il calcolo iterato di una funzione, ovvero la composizione di una funzione con se stessa. La sintassi di Iterates è:

ITERATES(u(x), x, x_in, [num_passi])

dove:

u(x) è la funzione di cui vogliamo calcolare le iterate;

x è la variabile di iterazione;

x_in è il valore iniziale assegnato a x.

Il meccanismo con cui opera ITERATES è il seguente:

viene posto x = x_in;

si calcola u(x);

se u(x) è un valore già presente nella sequenza il processo si arresta e viene scritto il vettore calcolato fino a questo punto, altrimenti si pone x = u(x) e si torna al passo 2.

Nel caso in cui sia specificato num_passi il processo si arresta dopo il numero di passi specificato, anche se non è stata raggiunta la condizione vista al punto 2). La funzione Iterate ha una sintassi analoga e differisce dalla Iterates solo per il fatto di restituire l'ultimo valore calcolato e non l'intero vettore.

Esempio 1

Se iteriamo la funzione u(x) = 1/x, a partire da x_in = 2 otteniamo

#1: ITERATES(1/x, x, 2)

#2: [2, 1/2, 2]

La sequenza si arresta perché viene riottenuto il valore 2.

Esempio 2

Se non assegniamo un valore numerico a x_in possiamo calcolare la composizione di una funzione con se stessa per un numero fissato di volte, come nel seguente esempio.

#1: ITERATES(1/(1+x), x, x, 5)

#2: [x, 1/(x+1), (x+1)/(x+2), (x+2)/(2*x+3), (2*x+3)/(3*x+5), (3*x+5)/(5*x+8)]

Esempio 3

Il problema dell'uso di Iterates si complica quando la funzione da calcolare non dipende da una sola variabile, ma da due o più; in questo caso si deve ricorrere alla iterazione su un vettore. Un tipico esempio è quello del calcolo iterativo della successione di Fibonacci.

Per calcolare i primi n termini della successione dobbiamo scrivere:

#1: FIB(n):=ITERATES([j, i + j], [i, j], [0, 1], n)

L'iteratore è il vettore [i, j], a cui viene assegnata la condizione iniziale [0, 1], corrispondente alla coppia [FIB(0), FIB(1)]. Ad ogni passaggio il vettore viene aggiornato: la sua prima componente viene posta uguale alla seconda componente del vettore al passo precedente, mentre il suo secondo elemento diventa la somma i + j degli elementi del vettore del passo precedente; ad esempio, da [FIB(0), FIB(1)] passiamo a [FIB(1), FIB(0) + FIB(1)] = [FIB(1), FIB(2)].

La sequenza dei numeri di Fibonacci richiesti può quindi essere letta come prima colonna della matrice che si ottiene.

Come si vede da questo esempio l'implementazione di un procedimento iterativo abbastanza semplice non è certamente immediata e ci obbliga a lavorare con vettori e matrici, anche in un caso in cui potremmo risolvere la questione senza farvi ricorso; anche se con questo tipo di programmazione si possono affrontare con successo parecchi problemi (vedi, ad esempio, Bacchelli et al., 1992, Barozzi e Cappuccio, 1993, Manara e Perotti, 1992), non si può certamente pensare

di farne un uso sistematico nella didattica della Matematica nella scuola superiore.

3. La programmazione "mista" di Derive 5 e Derive 6

Le principali novità introdotte nella versione 5 sono i costrutti Prog e Loop; il primo supera la limitazione che non consentiva l'esecuzione sequenziale di istruzioni, mentre il secondo permette un utilizzo "tradizionale" dei cicli, senza doverli convertire in opportune iterazioni funzionali.

La sintassi di Prog è

Prog(arg_1, arg_2, …, arg_n)

e il suo effetto è l'esecuzione sequenziale di arg_1, arg_2, …, arg_n; viene restituito il risultato di arg_n. Le strutture Prog possono essere anche annidate.

La sintassi di Loop è formalmente simile

Loop(arg_1, arg_2, ..., arg_n)

La struttura di controllo Loop esegue gli argomenti arg_1, arg_2, ..., arg_n sequenzialmente, finché non incontra una condizione di uscita (in generale espressa tramite un costrutto condizionale IF) contenente il comando RETURN espressione oppure EXIT.
Quando il ciclo finisce, viene restituito il valore dell'espressione che segue RETURN.
Osserviamo che Loop non prescrive la posizione al suo interno della condizione di uscita; quindi possiamo utilizzare sia un pre-test (come il While del Pascal) che un post-test (come il Repeat).

Esempio 1
Vogliamo scrivere un programma per calcolare la somma dei numeri da 1 a 50; si tratta di un esercizio puramente accademico, in quanto la funzione di librerie SUM ci permette di ottenere immediatamente la somma desiderata; ci serve comunque come primo esempio delle caratteristiche della programmazione in Derive.
Scriviamo il listato in forma indentata; otteniamo:

```
Prog
  i := 1
  sum := 0
  Loop
    sum := sum + i
    i := i + 1
    If i > 50
      RETURN sum
```

Sarebbe molto comodo avere un editor interno a Derive che ci consentisse di scrivere il listato in questa maniera e di immetterlo in Derive per l'esecuzione; purtroppo questo non avviene.
Dobbiamo riscrivere il tutto in forma lineare, utilizzando le parentesi in modo coerente con la sintassi dei vari comandi; questa è una grossa limitazione dell'ambiente di programmazione di Derive. Il listato precedente diventa:

```
#1:     PROG(i := 1, sum := 0, LOOP(sum := sum + i, i := i + 1, IF(i > 50, RETURN sum)))
```

Non dobbiamo fermarci a questo punto; la logica di Derive è quella di lavorare definendo ed utilizzando funzioni; possiamo allora introdurre la funzione SOMMA(n), che a un numero n assegnato dall'utente faccia corrispondere la somma degli interi da 1 a n.
Con semplici modifiche al programma possiamo scrivere:

```
#2:     SOMMA(n, i, sum) := PROG(i := 1, sum := 0, LOOP(sum := sum + i, i := i + 1, IF(i >n, RETURN sum)))
```

Per calcolarla è sufficiente scriverla indicando il primo argomento

```
#4:     SOMMA(50)
#5:             1275
```

Esempio 2

In questo esempio vediamo alcune funzioni che possono aiutarci nell'introduzione del concetto di integrale definito di una funzione continua in [a, b] (vedi, ad esempio, Barozzi, 1998).

Come primo passo dobbiamo "spiegare" a *Derive* che con f indichiamo una funzione di una variabile; questo viene realizzato nella riga 1, con la definizione "a vuoto".

Seguono poi i listati delle funzioni che calcolano un'approssimazione dell'area del trapezoide di f(x) con la formula dei rettangoli; si utilizza una suddivisione di [a, b] in n sottointervalli; in AREA_1 l'altezza di ciascun rettangolo è valutata calcolando f nel primo estremo, in AREA_2 nel secondo estremo.

```
#1:     f(x) :=

AREA_1(a, b, n, h, s) :=
 Prog
  s := 0
  h := (b - a)/n
  Loop
   If a ≤ b
    RETURN s
   s := s + f(a)·h
   a := a + h

AREA_2(a, b, n, h, s) :=
 Prog
  s := 0
  h := (b - a)/n
  Loop
   If a ≤ b
    RETURN s
   s := s + f(a + h)·h
   a := a + h
```

La funzione AREA_G utilizza invece in ogni sottointervallo un punto scelto a caso; essa si rivela quindi utile per mostrare il concetto di somma di Riemann.

```
AREA_G(a, b, n, h, s) :=
 Prog
  s := 0
  h := (b - a)/n

  Loop
   If a ≤ b
    RETURN s
   s := s + f(a + h·RANDOM(1))·h
   a := a + h
```

La funzione AREA_G, a differenza delle due precedenti fornisce un risultato casuale:

```
#1:     AREA_G(1, 2, 10)
#2:             0.6865026023
#3:             0.6845723281
#4:             0.6900543001
```

Fino a questo punto le applicazioni che abbiamo visto non si discostano molto da quelle che si potrebbero fare con un qualunque linguaggio di programmazione; come già detto è a questo punto che possiamo combinare la programmazione con l'ambiente dei comandi per proseguire nello studio di vari problemi; realizziamo così una integrazione maggiore (e più facile) tra la programmazione e lo studio di argomenti matematici.

Per dare un'idea dei possibili sviluppi, affrontiamo alcuni problemi, indicando la domanda che ci poniamo e una possibile risposta.

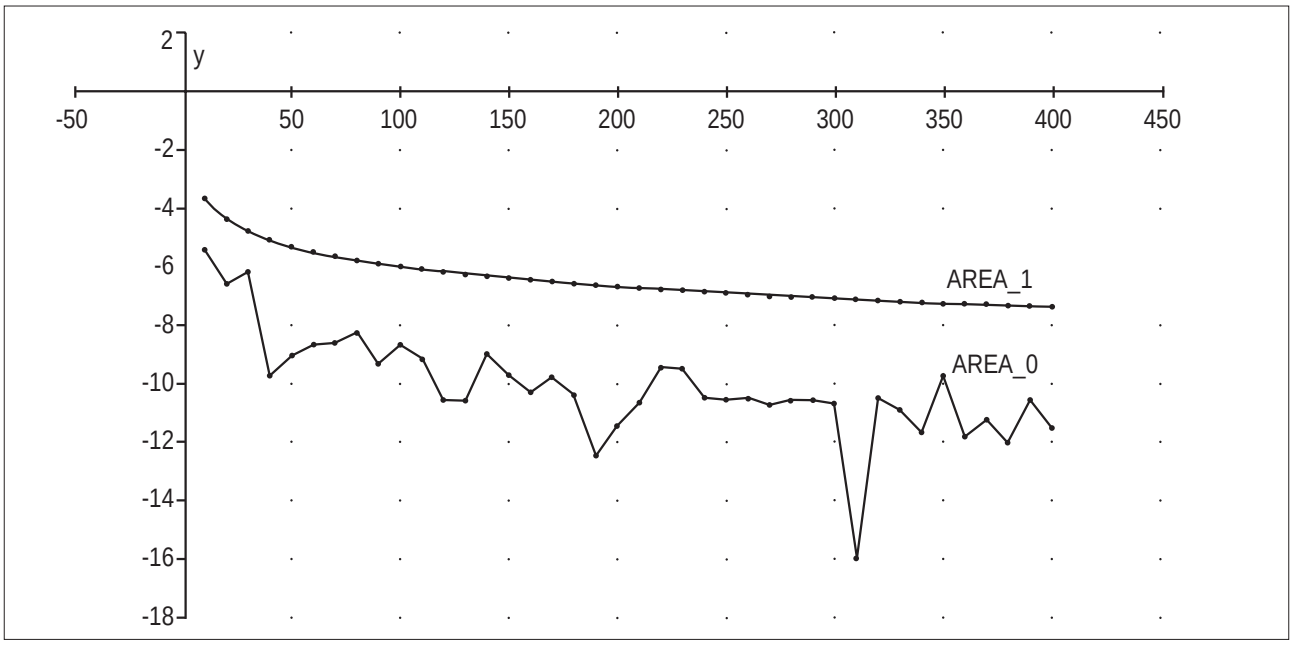
D: Vorrei fare alcuni esperimenti numerici, calcolando ad esempio, 100 volte la funzione AREA_G; devo ripetere manualmente il calcolo oppure posso fare più velocemente?
R: No, basta utilizzare il comando VECTOR; per simulare 100 valutazioni si introduce e si valuta il seguente vettore:
VECTOR(AREA_G(1, 2, 10), k, 1, 100)

D: Voglio vedere che cosa succede quando aumento il numero di sottointervalli, ad esempio da 10 a 400 a passi di 10.
R: Anche in questo caso basta utilizzare VECTOR.
VECTOR(AREA_G(1, 2, k), k, 10, 400, 10)

D: Il risultato dell'integrale è log2; come posso avere un'idea dell'errore (o meglio, del suo modulo), quando aumenta il numero delle suddivisioni?
R: Calcoliamo
VECTOR(ABS(AREA_G(1, 2, k) - LOG(2)), k, 10, 400, 10)
La valutazione di questo vettore fornisce (riportiamo solo le prime e le ultime componenti):
[0.003810382581, -0.003011925016, -0.002178114194, 0.000106627631, ..., 6.516967948·10^(-6), 9.226224087·10^(-6), 7.175589547·10^(-6)]

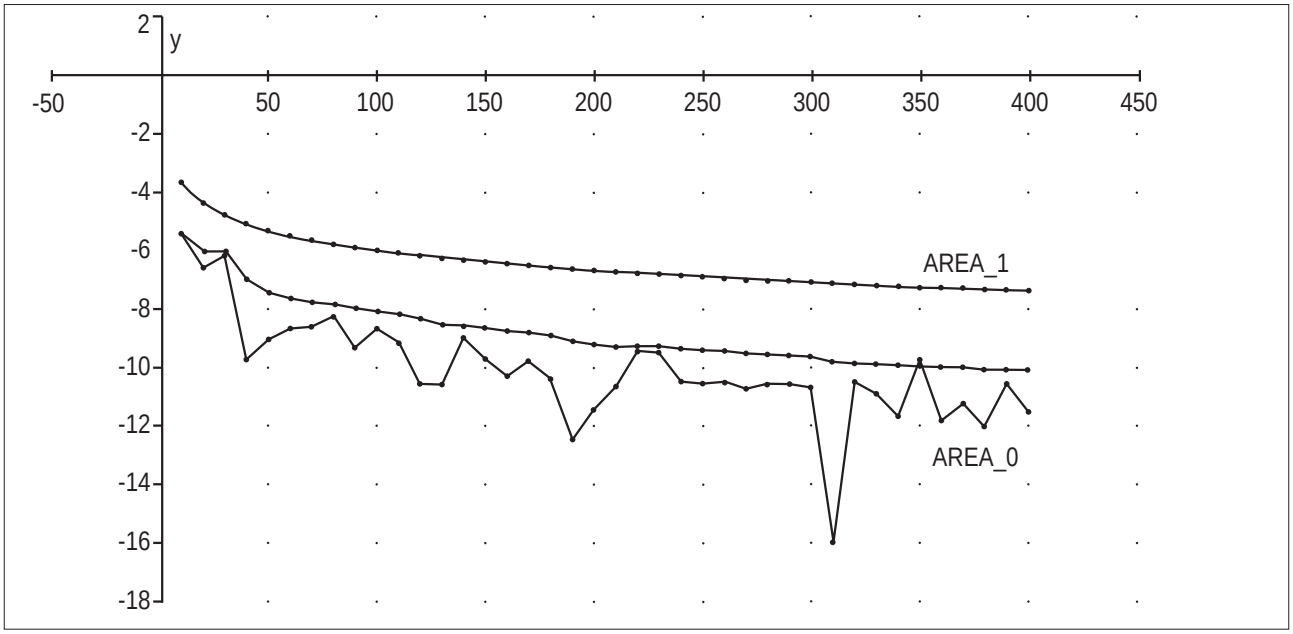
D: Il risultato precedente è scarsamente leggibile; come posso migliorarlo?
R: Conviene calcolare non il modulo dell'errore, ma il logaritmo di tale modulo.
VECTOR(LN(ABS(AREA_G(1, 2, k) - LOG(2))), k, 10, 400, 10)

D: Vorrei visualizzare il comportamento del logaritmo del modulo dell'errore e confrontarlo con quello della funzione AREA_1. Come posso fare?
R: Per tracciare un grafico abbiamo bisogno delle coppie [k, LN(ABS(AREA_G(1, 2, k) - LOG(2)))]; la valutazione della funzione
VECTOR([k, LN(ABS(AREA_G(1, 2, k) - LOG(2)))], k, 10, 400, 10)
fornisce una matrice (ogni riga della matrice rappresenta un punto del grafico); se calcoliamo la quantità analoga per la funzione AREA_1, otteniamo il grafico della figura 1.



D: La funzione AREA_G presenta ampie oscillazioni; vorrei tracciare il grafico della sua media.
R: Si tratta di trovare la media dei valori di AREA_G calcolati fino a un certo punto; per fare questo, innanzitutto salviamo i valori di AREA_G in un vettore
vv := VECTOR([k, AVERAGE(VECTOR(LOG(ABS(AREA_G(1, 2, h) - LOG(2))), h, 1, k))], k, 10, 400, 10)
poi calcoliamo il vettore delle medie

vv_media := VECTOR([k·10, AVERAGE(VECTOR(vv SUB h SUB 2, h, 1, k))], k, 1, 40)
La visualizzazione di questa quantità è riportata nella figura 2.



Bibliografia

Accascina G., Berneschi P. (1998a), *Introduzione all'uso di DERIVE in geometria*, Quaderni di didattica della matematica, n.2, IRRSAE Lazio, 1998

Accascina G., Berneschi P. (1998b), *Uso di DERIVE per esplorare le trasformazioni geometriche*, Quaderni di didattica della matematica, n.3, IRRSAE Lazio, 1998

Bacchelli B., Lorenzi A,. Perotti A. (1992), *Analisi Matematica con Derive*, McGraw-Hill Italia, Milano

Barozzi G.C. (1998), *Primo corso di Analisi Matematica*, Zanichelli, Bologna

Barozzi G.C. (1990), *Derive, un sistema di calcolo simbolico al servizio della didattica*, La matematica e la sua didattica, vol. IV n. 2

Barozzi G.C., Cappuccio S. (1993), *Matematica I. Schede di lavoro guidato. Schemi delle lezioni*, Pitagora, Bologna

Boieri P. (1994), *Il computer nella didattica dell'algebra*, in "L'algebra tra tradizione e rinnovamento - Seminario di formazione per Docenti - Viareggio, settembre 1994", Min. Pubblica Istruz. e U.M.I. - Liceo Vallisneri – Lucca

Boieri P. (2003), *Derive. Laboratorio informatico per la matematica*, Loescher, Torino.

Cappuccio S. (1995), *Nuove caratteristiche di Derive (Versione 3)*, La matematica e la sua didattica, n. 3/1995.

Cappuccio S. (1998), *Uno sguardo a Derive per Windows*, La matematica e la sua didattica, n. 1/1998.

Cappuccio S. (2001), *Derive Versione 5*, La matematica e la sua didattica, n. 3/2001.

Manara M., Perotti A. (1992), *Algebra lineare e Geometria con Derive*, McGraw-Hill, Milano.

Rich A.D., Stoutemyer D.R. (1994), *Inside the Derive Computer Algebra System*, The International *Derive* Journal, vol. 1, n. 1, 3-17

The new features in *Derive 6*

di B. Kutzler
ACDCA (Austrian Center for Didactics of Computer Algebra)

Since the days that computers emerged as fast and diligent number-crunching devices, mathematicians have searched for ways of getting them to perform increasingly sophisticated tasks. This has led to the latest development in technology, computer algebra systems, a sort of mathematical expert system which is necessarily changing the way we do mathematics.

Derive is a very mature computer algebra system for PCs, its roots go back to the late Seventies. It was the first computer algebra system used in mathematics education and is the most widely used computer algebra software for teaching and learning mathematics. Many ministries and school authorities throughout Europe have adopted *Derive* as the primary computer tool for mathematics education.

Derive 6 is the newest version with a lot of features which support teachers and students in a mathematics class. We give an overview of the major new features of *Derive 6*.

1) Display the steps in the simplification of an expression with optional display of transformation rules

#2: $\frac{d}{dt}(5 \cdot v \cdot t^3)$

$\frac{d}{dx}(a \cdot F(x)) \Rightarrow a \cdot \frac{d}{dx} F(x)$

#3: $5 \cdot v \cdot \frac{d}{dt} t^3$

Derive 6 offers a very powerful feature which allows you to step through a simplification and see the transformation rules the program applies. It is called the Display Steps feature.

When using this feature, *Derive* performs only one step of the simplification and displays the transformation rule used for this step. The rule is contained in a text box and is displayed in blue to distinguish it from normal text that is black by default.

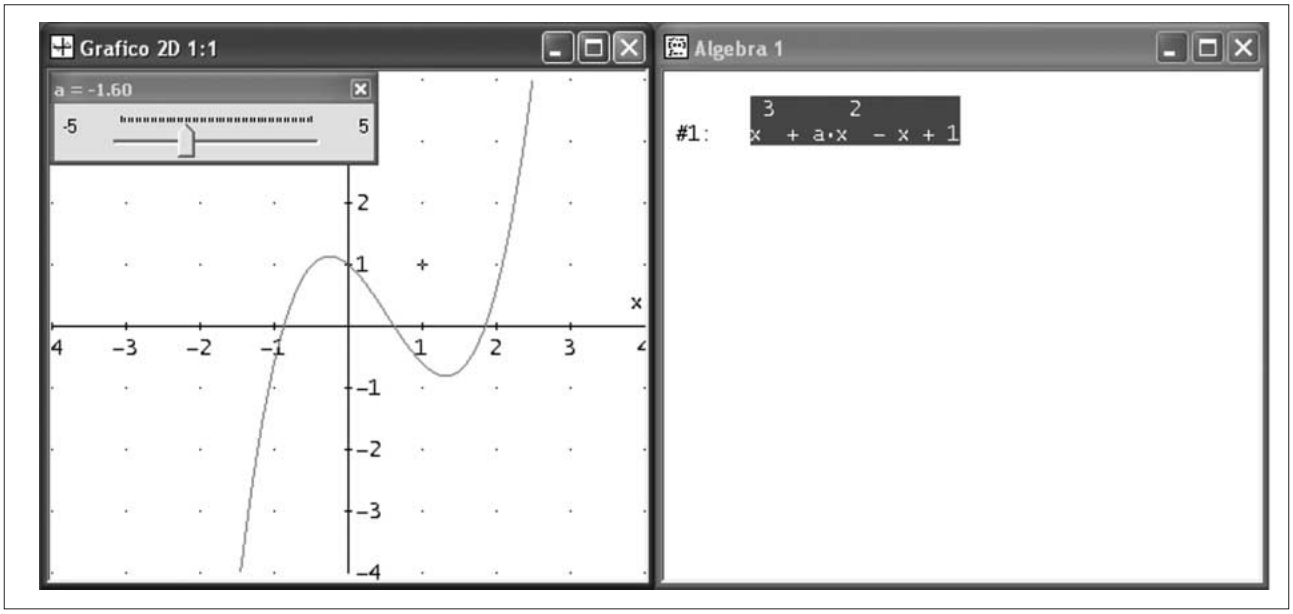
This Display Step feature is an ongoing research and development project. About 2,000 transformation rules are implemented in *Derive 6* covering differentiation, integration, summation, products, elementary and special function simplification, and equation and inequality simplification. You can try stepping through any problem, but some or all of the intermediate steps may not be displayed (yet). More transformation rules will be implemented in the future. If you want to keep up and obtain more of this feature, regularly look for free upgrades to version 6.x at **http://www.derive.it** or **http://www.derive-europe.com**.

The Display Step feature is a powerful and useful feature for mathematics teaching and learning. Possible uses of this feature are:

1. A user wants to know how *Derive* simplified an expression, i.e. wants to look into the “black box”.
2. A user wants to study the subtleties of simplification.
3. A student starts studying a topic by observing an “expert” on examples the student chooses.
4. A student deepens understanding by seeing the steps an “expert” makes and recognizing the rules the expert used. (Here the display of rules must be switched off.)

We see the major goal in using computer algebra systems such as *Derive* for teaching mathematics in a reduction of the handicraft parts of the subject. *Derive* supports this goal by automating calculations. With the Display Steps feature it also supports more traditional approaches. We consider this ideal for an evolutional transition from what we have to what we want.

2) Make plots dynamic: Animate expression plots with slider bars



To study, for example, the influence of parameter *a* on the shape of the graph of you can insert a slider bar for *a* before plotting the graph. Then you can control the value of *a* (within the bounds specified when introducing the slider bar) with the mouse and immediately see the change of the graph.

3) Optionally format the background of a plot window with a bitmap graphic file format

This is a very powerful pedagogical feature. Teachers can prepare background pictures with curves and then let the students find a function whose graph matches the curve(s).

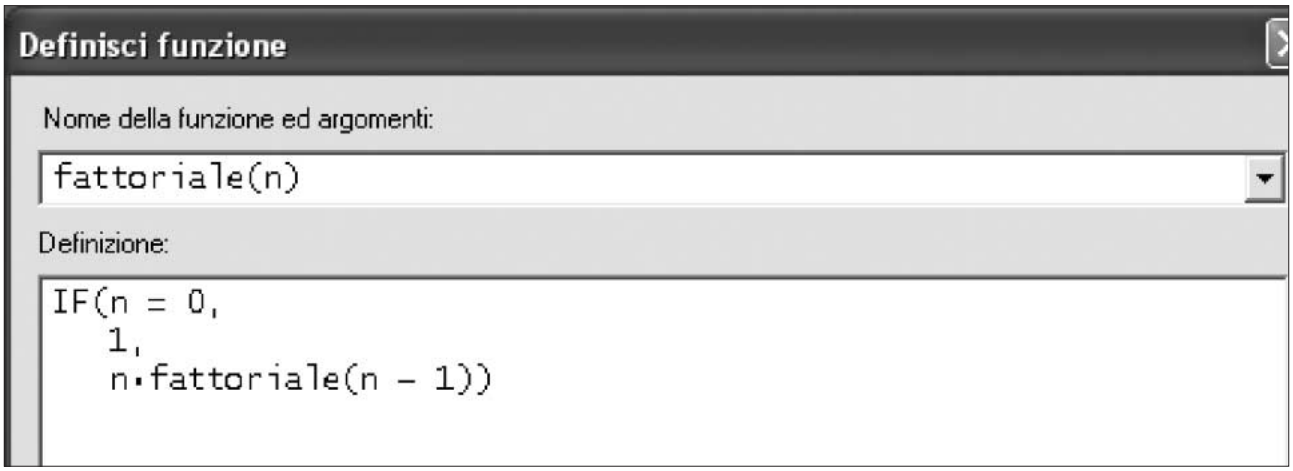
4) Let plots be labelled with defining equations

This option uses the existing plot window annotation feature to provide a descriptive label in the same color used by the plot itself. The label is initially placed in the top left hand corner of the plot window. Subsequent labels are positioned below the previous label. Like all plot window annotations, the plot label can be edited and/or moved by the user as desired.

5) Customize menus and toolbars

Teachers can now tailor *Derive* according to the needs in the classroom. For example, one can generate a “calculus-free” *Derive* by removing the five calculus buttons (for limits, derivatives, integrals, sums, and products) and the calculus menu.

6) Function definitions and variable values may now be edited with multi-line edit boxes



One can now use the **Author>Function Definition** command to display a dialog box where definitions that span more than one line can easily be edited. A vertical scrollbar is displayed when there are more definition lines than display lines. One can use the up/down arrow keys to move between lines and press the (Alt)+(Enter) key while editing to create new lines.
The **Edit>Expression** command can now be used to edit function definitions, variable values, and domains using the **Function Definition**, **Variable Value** and **Variable Domain** multi-line edit dialogs.

7) Communicate with TI CAS calculators

A handheld provides a maximum of mobility and robustness. You can use it in virtually any environment: an office, a lecture room, a classroom, at home, a car, a train, a bus, etc.

Derive, on the other hand, provides speed, a high resolution color screen, mouse support, and connection to a printer. *Derive* also offers more mathematics than the handhelds and a powerful worksheet concept for producing mathematical documents comprising expressions, text, graphs, and OLE objects.

Interconnectivity is a concept that allows you to combine the best of these two worlds into a powerful mathematics and science teaching and learning environment, as is shown in the book “Interconnectivity – Derive 6 and Voyage200/TI-92+/TI-89 in Teaching Mathematics”.

Using a combination of tools has pedagogical value in itself: *Derive 6* and the handheld use two different but still sufficiently similar languages. It is advantageous for students to learn more than one mathematics software language, because this will prepare them better for their professional life.

Another important advantage of using two different tools is that students learn to choose the tool that is most appropriate for a given task.

There are many other new features. Following is a list of most of them:

- Mathematical characters are now entered and displayed using the new *Derive* Monospace Unicode encoded font.
- The *Derive* Unicode font is scaleable and the interface appropriately provides options for selecting font size, style and color.
- An enhanced text box now supports Unicode characters and html link hot spots.
- The state variable settings are now saved in dfw files rather than in the Derive6.ini file.
- A table of contents tab makes it easy to navigate through the on-line help for *Derive*.
- Expression entry using multiple entry lines is now available.
- Parentheses matching is offered in most places where Greek and Math toolbar symbols are allowed (e.g. expres sions, function definitions and variables, plot annotations, etc.)
- The style of a line segment may be selected for connected point plots.
- Optionally turn off mesh lines in 3D plots.
- Small, medium, or large points may now be specified.
- Rotate the 3D-plot “box” with the mouse.
- Many algorithmic improvements and extensions make the mathematics of *Derive* even more powerful.

Find out more about *Derive 6* by trying the free 30 day trial license. It can be downloaded from **http://www.derive.it** or **http://www.derive-europe.com**.

Dopo Pascal, *Derive*?

di Laura Gobetti
Liceo Scientifico Giordano Bruno - Torino

Un paio di anni fa, nel Dipartimento di Matematica del Liceo Scientifico Giordano Bruno di Torino, è nato un confronto sulla necessità di sostituire, nei corsi sperimentali, il linguaggio di programmazione Turbo Pascal. Il nostro disagio nel proporre un linguaggio "antico", la scarsa motivazione verso il suo apprendimento da parte dei ragazzi e le effettive difficoltà formali e concettuali rendevano l'insegnamento dell'informatica sempre più difficile e conseguentemente sacrificato. Il desiderio di sostituire tale linguaggio non era realizzabile per la mancanza, tra i prodotti a nostra conoscenza, di uno che ci soddisfacesse completamente.
Nel settembre 2002 abbiamo deciso di contattare il prof. Boieri, che ci ha proposto di imparare a programmare usando la *versione 5 di Derive*, di cui noi conoscevamo già la versione 4 come software.
L'idea ha convinto tutti, soprattutto per la possibilità di unire, in un unico pacchetto, la parte di programmazione all'utilizzo di *Derive* come software di manipolazione algebrica e grafica di oggetti matematici. Così a novembre abbiamo iniziato il nostro corso su *Derive* tenuto dal prof. Boieri.
Nelle prime ore di corso abbiamo affrontato alcune caratteristiche basilari di *Derive* come software, dopo di che siamo partiti con la programmazione vera e propria.
Nella prima lezione abbiamo visto che parecchi problemi sui cicli enumerativi potevano essere risolti usando la semplice istruzione *vector*; nelle lezioni successive abbiamo via via affrontato le strutture vere e proprie di programmazione: *prog, if, loop* e abbiamo risolto problemi di programmazione con queste.
All'inizio dell'anno scolastico 2003/2004 abbiamo deciso di mettere in pratica in alcune classi pilota quanto imparato nell'anno precedente, abbandonando il Turbo Pascal in favore di *Derive* in alcune classi di biennio e di triennio.
Con i colleghi del dipartimento di matematica abbiamo stabilito, nel biennio, di affrontare le conoscenze fondamentali del software *Derive* e di iniziare a lavorare su semplici programmi con *vector, prog, if*. Nel triennio, negli anni futuri, verrà introdotta la struttura *loop* e verranno affrontati programmi più complessi. Le classi di triennio che quest'anno hanno iniziato a lavorare con *Derive* hanno invece dovuto affrontare tutte le strutture basilari della programmazione, perché tale linguaggio non era stato utilizzato negli anni precedenti.
L'esperienza che presento è il lavoro svolto durante il corrente anno scolastico nelle mie classi 3ª e 4ª P.N.I e nasce dalla collaborazione con i colleghi del dipartimento che come me hanno deciso di iniziare a programmare usando *Derive*.
Per ogni comando è stata proposta ai ragazzi una scheda con otto-dieci esercizi: io ne presenterò qui solo due o tre a titolo esemplificativo.
Dei primi esercizi scriverò sia quanto i ragazzi devono digitare, sia ciò che appare sulla finestra di ALGEBRA di *Derive*.

1. I vettori e il loro utilizzo

Per iniziare a far vedere ai ragazzi l'economicità di *Derive* rispetto al Turbo Pascal che loro conoscevano, ho introdotto per primo il comando *vector* con cui si potevano risolvere in una riga problemi la cui sintassi in Turbo Pascal era ben più pesante.
Per prima cosa ho introdotto la sintassi del comando *vector* e quindi ho proposto loro alcuni esercizi da risolvere utilizzandola.

ESERCIZIO 1

Fai scrivere al computer i quadrati dei primi 10 numeri naturali.

VECTOR (x^2, x, 1, 10)

1: VECTOR(x^2, x, 1, 10)
2: [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]

ESERCIZIO 2

Fai scrivere al computer la tavola pitagorica.

VECTOR(VECTOR(i*j, j, 1, 10), i, 1, 10)

1: VECTOR(VECTOR(i·j, j, 1, 10), i, 1, 10)

2:

1	2	3	4	5	6	7	8	9	10
2	4	6	8	10	12	14	16	18	20
3	6	9	12	15	18	21	24	27	30
4	8	12	16	20	24	28	32	36	40
5	10	15	20	25	30	35	40	45	50
6	12	18	24	30	36	42	48	54	60
7	14	21	28	35	42	49	56	63	70
8	16	24	32	40	48	56	64	72	80
9	18	27	36	45	54	63	72	81	90
10	20	30	40	50	60	70	80	90	100

2. Le strutture fondamentali della Programmazione in *Derive*

Dopo un paio di lezioni su *vector* ho quindi iniziato la programmazione vera e propria.

Un programma nel linguaggio *Derive* è sempre una funzione. La struttura di controllo fondamentale per la programmazione è *prog.*

Avendo inizialmente a disposizione solo l'istruzione *prog* gli esercizi risultano banali e in genere facilmente risolvibili con funzioni predefinite di *Derive*, ma non potendo presentare ai ragazzi due comandi contemporaneamente non avevo altra scelta. Ho comunicato molto onestamente tale situazione alla classe che ha compreso la necessità didattica e ha svolto seriamente gli esercizi senza contestazioni.

Anche qui ho dapprima introdotto la sintassi del comando e poi ho proposto alcuni esercizi da svolgere con questo.

F_1(x1, x2, …, xk) :=prog(arg_1, arg_2, …, arg_n)

- *x1, x2, …, xn sono le variabili che compaiono nel programma*
- *arg_1, …, arg_n vengono eseguiti in sequenza e il programma ritorna il risultato di arg_n*

ESERCIZIO 1

Costruire un programma che, letti a e b, fornisca il quadrato della loro somma.

F(a, b, c):= PROG(c := a+b, c^2)

```
        F(a, b, c) :=
          Prog
#1:         c := a + b
            c^2
#2:   F(2, 5)
#3:                    49
```

ESERCIZIO 2

Scrivere un programma che fornisca la media dei voti assegnati ad un alunno nelle quattro verifiche di matematica del quadrimestre.

media(x, y, z, w, s, m) :=PROG(s := x + y + z + w, m := s/4)

```
        media(x, y, z, w, s, m) :=
          Prog
#1:       s := x + y + z + w
            m := s/4
#2:     media(5, 7, 7, 6)
#3:                      6.25
```

Ho quindi introdotto l'istruzione *if* utilizzata per risolvere problemi di alternativa.

IF(condizione, se_vera, se_falsa, se_indecidibile)

- *se "condizione" è vera viene eseguito l'argomento se_vera;*
- *se "condizione" è falsa viene eseguito l'argomento se_falsa;*
- *se Derive non può stabilire la verità o la falsità di "condizione" allora viene eseguito l'argomento se_indecidibile.*

All'interno di PROG( ) gli argomenti se_falsa e se_indecidibile possono essere omessi; se la condizione è falsa viene eseguito l'argomento successivo di PROG.

ESERCIZIO 1

Letto x da tastiera, calcolare 1/(x+1).

F2 (x, impossibile) := PROG(IF(x = -1, impossibile, 1/(x+1)))

```
            F2 (x, impossibile) :=
               Prog
#1 :              If x = -1
                        impossibile
                        1/(x+1)
#2:         F2 (4)
#3:                                   1
                                      5

#4:         F2 (-1)
#5:                                   impossibile
```

ESERCIZIO 2

Letti tre numeri in ordine crescente, scrivere se sono elementi successivi di una progressione aritmetica.

F6 (x, y, z, si, no) := PROG (IF (y-x = z-y, si, no))

```
          F6 (x, y, z , si , no) :=
             Prog
#1:             If  y – x = z – y
                     si
                     no
#2  :     F6 (3 , 7 , 11)
#3  :                                 si
#4  :     F6 (3 , 5, 11)
#5  :                                 no
```

Per ultima ho introdotto l'istruzione *loop*

LOOP(arg_1,arg_2, ... , arg_n)

- *La struttura di controllo LOOP() esegue gli argomenti arg_1, arg_2, ..., arg_n sequenzialmente , finché non incontra il comando RETURN oppure EXIT.*
- *La sintassi di **RETURN** è: RETURN espressione*
- *In generale, la condizione di uscita viene espressa tramite un costrutto condizionale IF.*
- *Quando il ciclo finisce, la funzione in cui è inserito restituisce il valore dell'espressione che segue RETURN.*

ESERCIZIO 1

Calcolare il MCD tra due numeri interi relativi utilizzando l'algoritmo di Euclide.

eucl(x, y, z):=PROG(LOOP(IF(y = 0, RETURN ABS(x)), z := MOD(x, y), x := y, y := z))

```
          eucl(x, y, z) :=
            Prog
              Loop
```

```
                If y = 0
#1:               RETURN ABS(x)
                z := MOD(x, y)
                x := y
                y := z
#2:         eucl(30, 12)
#3:                                   6
```

ESERCIZIO 2
Costruire un programma che indichi in quale posizione si trova il massimo tra gli elementi di un vettore v.
Posiz_max(v, a) := PROG(a := 0, LOOP(a := a+1, IF(MAX(v) = FIRST(v), RETURN a), v := REST(v)))

```
        posiz_max(v, a) :=
          Prog
            a := 0
            Loop
#1:           a := a + 1
              If  MAX(v) = FIRST(v)
                  RETURN a
              v := REST(v)
```

3. Applicazioni alla probabilità
Quando avevo finito di presentare le istruzioni fondamentali della programmazione con *Derive*, nella classe quarta stavo trattando il calcolo delle probabilità secondo la definizione classica.
Mi sono chiesta se potessi applicare *Derive* alla probabilità e la risposta è stata la scheda seguente. Nella parte riguardante gli esercizi, viene anche qui proposto il testo dell'esercizio, la soluzione che lo studente fornisce e, in alcuni casi, il calcolo della funzione che è stata scritta.

Scheda di lavoro

DERIVE e i numeri casuali

Derive possiede alcune funzioni di probabilità che permettono di effettuare al computer i calcoli che durante le lezioni abbiamo finora eseguito manualmente:
z! è la funzione fattoriale: z! è il prodotto dei primi z numeri naturali.
COMB(z, w) è il numero di combinazioni di z oggetti presi a gruppi di w: COMB(z, w) restituisce z!/(w!(z - w)!).

Adesso, però, non lavoriamo con queste funzioni e ci poniamo una domanda diversa: *se invece di calcolare le probabilità teoricamente, effettuassimo veramente le prove, che risultati otterremmo?*
L'elaboratore ci permette di compiere un lavoro che manualmente sarebbe troppo faticoso.
Per creare questi programmi dobbiamo prima conoscere la funzione RANDOM(n).

RANDOM(n) richiama il generatore di numeri pseudocasuali. Consideriamo una sola delle sue definizioni:
se n>1, RANDOM(n) restituisce un numero naturale casuale nell'intervallo [0, n). Ad esempio, ogni volta che si richiama RANDOM(6), viene restituito uno dei 6 numeri interi equiprobabili dell'insieme {0, 1, 2, 3, 4, 5}.

ESERCIZI

1. Che comando scriveresti per simulare il lancio di una moneta?
#1: RANDOM(2)

2. Lanciando 100 volte in aria una moneta, quante volte esce testa? Costruisci un programma che faccia eseguire la prova all'elaboratore.

```
MONETA(N, X, TESTA) :=
    Prog
```

```
                N := 0
                TESTA := 0
                 Loop
#2:                N := N + 1
                   X := RANDOM(2)
                   If X = 0
                      TESTA := TESTA + 1
                   If N > 99
                      RETURN TESTA
   #3:                                          51
   #4:          MONETA( )
   #5:                                          58
```

3. Facciamo eseguire all'elaboratore per 100 volte l'esercizio 2, quindi facciamo la media dei valori trovati. Che risultato otteniamo?

```
#6:       AVERAGE(VECTOR(MONETA( ), i, 1, 100))
#7:                                          50.62
```

4. Che comando scriveresti per simulare il lancio di un dado?

```
#8:  RANDOM(6) + 1
```

5. Lanciando 100 volte in aria un dado, quante volte esce il numero 6? Costruisci un programma che faccia eseguire la prova all'elaboratore.

```
      DADO(N, SEI, X) :=
          Prog
            N := 0
            SEI := 0
            Loop
#9:            N := N + 1
               X := RANDOM(6) + 1
               If X = 6
                 SEI := SEI + 1
               If N > 99
                  RETURN SEI
#10:                                  14
   #11:  DADO( )
#12:                                  11
```

6. Facciamo eseguire all'elaboratore per 100 volte l'esercizio 5, quindi facciamo la media dei valori trovati. Che risultato otteniamo?

```
#13:  AVERAGE(VECTOR(DADO( ), i, 1, 100))
   #14:                                  14.42
```

7. Hai a disposizione una scacchiera 10 ? 10. Ogni casella è individuata da una coppia ordinata di numeri interi: il primo individua la colonna, il secondo la riga. In una di queste celle il computer nasconde un alieno. Costruisci un programma che permetta di trovare l'alieno.

```
          TANA(a, b, OK) :=
           Prog
#15:         a := RANDOM(10) + 1
             b := RANDOM(10) + 1
            OK

      ALIENO(x, y, d, trovato) :=
         Prog
```

```
        d := ‹((a - x)^2 + (b - y)^2)
#16:    If d = 0
          trovato
          d
  #17:  ALIENO(2, 3)
  #18:                          √53
  #19:  ALIENO(9, 5)
  #20:                           4
  #21:  ALIENO(9, 9)
  #22:                           8
  #23:  ALIENO(9, 1)
  #24:                       trovato
```

8. Costruisci un programma che ti dica quante buste, da una figurina, devi acquistare per riempire un album di n figurine.

```
FIGUR(n, i, a, c, k) :=
 Prog
  a := VECTOR(0, i, 1, n)
  c := 0
  Loop
   If MIN(a) = 1
     RETURN c
   k := RANDOM(n) + 1
   If a↓k = 0
     a↓k := 1
   c := c + 1
#26:  FIGUR(10)
#27:                    22
```

Conclusione

Dopo i primi mesi di questa esperienza, credo di poter affermare che *Derive* sia un buon sostituto di Pascal.

- I ragazzi lavorano più volentieri con *Derive* di quanto non facessero con Pascal.
- È chiaro che la difficoltà di scoprire l'algoritmo risolutivo di un problema rimane immutata, ma la traduzione nel linguaggio di programmazione è ora più semplice.
- Con il Pascal i ragazzi dovevano ricordarsi di dichiarare inizialmente tutte le variabili specificandone il tipo, con *Derive* è sufficiente elencarle negli argomenti della funzione facendo attenzione a nominare per prime quelle che devono essere introdotte da tastiera.
- Una difficoltà di *Derive* è l'uso delle parentesi nella scrittura del programma: ogni comando deve essere seguito dall'apertura di una parentesi che sarà chiusa quando saranno state date tutte le sue istruzioni; una difficoltà analoga del Pascal era quella dell'uso di begin…end.
- Il problema più grande di *Derive* è la sua memoria eccezionale per cui anche quando si cancella una funzione sbagliata non viene eliminata dalla memoria:
    1. è utile abituare i ragazzi a scrivere il nome della funzione seguito da un numero in modo da poterlo cambiare ad ogni eventuale correzione;
    2. a volte, anche con l'accorgimento precedente, dopo che un programma è stato ripetutamente corretto, *Derive* ci dà strane risposte: conviene far salvare ai ragazzi la parte che ritengono più corretta, uscire dalla sessione di lavoro e riaprirne un'altra in cui continuare il lavoro ripartendo dal file salvato.
- Il tipo di esercizi che si può affrontare in classe dipende anche da quali sono le funzioni predefinite di *Derive* che i ragazzi conoscono: io ho scelto di fornirne poche come prerequisito per iniziare a programmare e di introdurne man mano che si procede nel lavoro di programmazione.

Algoritmi iterativi

di Sebastiano Cappuccio
Istituto Tecnico Aeronautico "F. Baracca" – Forlì

Molto è stato scritto sulle potenzialità didattiche dei *Computer Algebra Systems* e non abbiamo la pretesa di dire qualcosa di nuovo; tuttavia forse vale la pena elencare qui di seguito alcune delle possibili strategie di utilizzo:

- come mezzo di visualizzazione,
- come strumento per favorire la scoperta guidata,
- come aiuto per gli studenti più incerti nel calcolo algebrico,
- come supporto nel *problem solving*, permettendo di concentrarsi sull'impostazione del modello matematico senza doversi troppo preoccupare del calcolo,
- ultimo, ma non meno importante, i CAS possono essere usati dall'insegnante anche come espediente per rendere più gradito lo studio della matematica da parte dei suoi studenti, sfruttando il loro atteggiamento di solito molto favorevole nei confronti della tecnologia.

Ovviamente non c'è rosa senza spine: i dubbiosi sull'utilità dell'uso della tecnologia nell'insegnamento della matematica potrebbero avanzare alcune critiche sull'uso dei CAS; ne cito alcune tra quelle che ho ascoltato in questi anni.

1. I Cas possono dare assuefazione? Che fare se poi gli studenti si rendono dipendenti dai CAS? L'uso sistematico di un CAS potrebbe far perdere molte abilità di calcolo manuale, incentivando la naturale pigrizia dello studente e facendo preferire nei problemi le soluzioni più "muscolari" e puramente orientate al calcolo rispetto a soluzioni più ragionate ed efficienti.
2. La presenza di un CAS potrebbe far perdere all'insegnante il suo ruolo di "giudice supremo" e di unico mediatore tra la materia e lo studente: può rendersi necessaria una ridefinizione del ruolo dell'insegnante.
3. L'uso di un CAS richiede all'insegnante una certa conoscenza dello strumento e ciò a sua volta richiede un notevole investimento di tempo e di fatiche che spesso non è ripagato dai risultati e che soprattutto ben difficilmente potrà avere un riconoscimento concreto a livello di stipendio e di carriera.
4. L'uso regolare di un CAS può "liberare" molto tempo nel lavoro in classe. L'esistenza di questi strumenti può permettere di ridurre molto del tempo oggi sprecato… pardon, impiegato nel puro e semplice addestramento al calcolo, il più delle volte fine a se stesso, invece che nell'apprendimento di concetti o nell'allenamento alla soluzione di problemi. Come impiegare il tempo guadagnato? La presenza dei CAS costringe a ridefinire obiettivi e curricoli scolastici e soprattutto, cosa ben più difficile, a modificare la prassi didattica codificata nei libri di testo e tramandata tra gli insegnanti di generazione in generazione…
5. Inoltre, l'apprendimento dell'uso dei CAS richiede agli studenti uno sforzo maggiore: per loro si tratta di una difficoltà in più; il loro uso richiede una conoscenza approfondita di ciò che si sta facendo. Paradossalmente l'uso di un CAS è molto meno meccanico di certi algoritmi di calcolo che vengono invece applicati senza sapere ciò che si fa, ma solo applicando certe regole ("Questo si fa.", "Questo non si deve fare, sennò l'insegnante, per qualche suo imperscrutabile motivo, si arrabbia."…).

Questo lavoro riporta, con qualche modifica, una esperienza di qualche anno fa in una classe quinta di Istituto Tecnico Aeronautico e può essere pensato come un esempio di breve corso di approfondimento nell'ambito dell'insegnamento dell'Analisi matematica in una scuola secondaria superiore.
In particolare ci si soffermerà sul contributo che un *Computer Algebra Systems* può dare nel presentare, sia pure in forma molto semplificata, un argomento tutt'altro che facile rendendolo tollerabile anche a studenti del tutto "normali".

Antefatto

Hollywood ci ha da tempo abituati a film che prendono lo spunto da argomenti scientifici: al di là dei film di fantascienza veri e propri (a dire il vero spesso con molta *fanta* ma poca *scienza*), sono numerosi i film con riferimenti alla biologia, all'informatica, all'ecologia e alla fisica (o almeno ai suoi effetti). Pochissimi sono i riferimenti alla matematica, evidentemente meno spettacolare di altre scienze.
Qualche anno fa però ebbe un grande successo il film "Jurassic Park", che contiene un breve ma esplicito riferimento a certe parti della matematica.
Nella scena dell'elicottero che viaggia verso la fatale isola, si svolge un dialogo tra il matematico Malcolm, interpretato dall'attore Jeff Goldblum, e il proprietario dell'isola, il miliardario John Hamond (interpretato da Richard Attenborough) e i protagonisti del film:

Malcolm: *E così voi due... scavate dinosauri.*
- Beh, ci proviamo.

Hamond: Ci dovete fare l'abitudine al professor Malcolm: soffre di un deplorevole eccesso di personalità, specialmente per un matematico.

M. *Caosologo, caosologo per l'esattezza. Lui non crede nella teoria del caos, specie per quanto riguarda il suo piccolo progetto scientifico.*

H. Che sciocchezze: non ha mai dato spiegazioni convincenti sulle sue preoccupazioni... Sicuramente sono un mucchio di masticature numeriche alla moda.

M. *Voi conoscete la teoria del caos?*
- No.

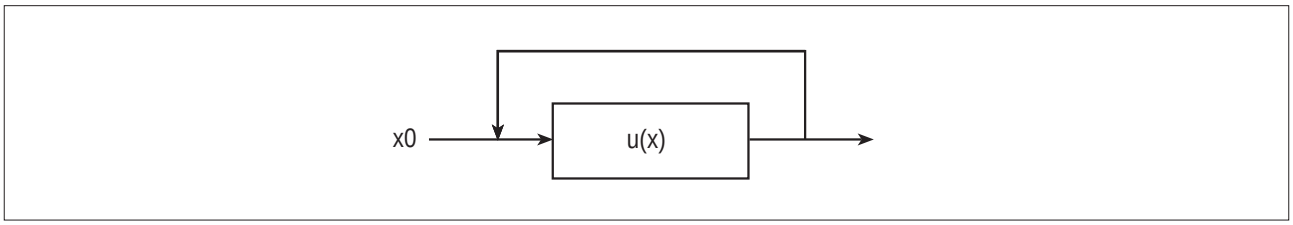
M. *No? Equazioni non lineari? Attrattori strani? Professoressa Sattler, mi rifiuto di credere che lei non familiarizzi con il concetto di attrazione ...*

Alcuni miei studenti, incuriositi, mi chiesero qualche spiegazione sulle parole del Prof. Malcolm. E così ebbe inizio il lavoro che, con qualche modifica, presento qui di seguito.

Prologo

In quasi tutte le attività che ora vedremo, la funzione **ITERATES** di *Derive* gioca un ruolo essenziale.
Ricordiamo al Lettore la sua sintassi:

ITERATES (u, *x*, *x*0) restituisce un vettore così generato: la prima componente è il valore iniziale (o "di innesco") ***x*0**, la seconda è il valore della funzione **u** in corrispondenza ad esso, il terzo è il valore di **u** in corrispondenza del precedente valore e così via.
Il ciclo ha termine appena si ottiene un valore uguale ad uno precedentemente ottenuto.



In pratica viene generato il vettore **[*x*0, u(*x*0), u(u(*x*0)), u(u(u(*x*0))), …]**.
Il "ciclo" ha termine, come si è detto, quando una componente risulta uguale ad una delle precedenti.
Si tenga presente che, per ridurre il rischio di un "loop infinito", conviene sempre calcolare la funzione **ITERATES** in modalità approssimata e che il concetto di "uguaglianza" è relativo alla precisione, cioè al numero di cifre decimali impostate.
Sempre per questo motivo può succedere che il vettore contenga come ultime componenti valori tutti (apparentemente) uguali.
La funzione **ITERATES (u, *x*, *x*0, n)** produce lo stesso risultato, ma terminando in ogni caso l'iterazione dopo **n** cicli.

Infine la funzione **ITERATES (u, *x*, *x*0, n)** produce solo l'ultimo risultato dell'iterazione e non l'intero vettore.

Atto primo, scena prima: algoritmo di bisezione

Gli studenti di questa classe conoscevano fin dalla seconda l'algoritmo di bisezione per determinare in modo approssimato gli zeri di una funzione (a dire in vero lo avevano realizzato partendo da un giochino, "indovina il numero", in Pascal). Ecco come questo algoritmo è stato riproposto con *Derive*.
L'algoritmo è molto semplice: con l'ipotesi che la funzione sia continua in un intervallo che contiene un unico zero della funzione, si divide l'intervallo in due parti (per comodità, uguali) e si individua in quale dei due semi-intervalli è contenuto lo zero. Per questo scopo sarà sufficiente vedere se la funzione negli estremi dell'intervallo assume o no segno discorde.
Poi si ripete il procedimento nel semi-intervallo individuato e così via fino a che l'intervallo non diventa minore di una prefissata tolleranza.
Ecco il relativo "programma" in *Derive*:

```
#1:   Algoritmo di bisezione - Sebastiano Cappuccio
#2:   F(X) :=
      test(a, b) :=
        If F(a)·F(b) < 0
#3:       "Il metodo di bisezione è applicabile"
          "Il metodo di bisezione non è applicabile"
#4:   m := (ELEMENT(q, 1) + ELEMENT(q, 2)) / 2
      passo(q) :=
        If F(m) ≠ 0
          If F(m)·F(ELEMENT(q, 1)) < 0
#5:         [ELEMENT(q, 1), m]
            [m, ELEMENT(q, 2)]
          [m, m]
#6:   bisect(a, b) := ITERATES(passo(q), q, [a, b])
```

Infine l'esecuzione; si noti l'istruzione che setta il numero di cifre visualizzato:

```
#6:   F(x) := x^2 - 2
#7:   test(0, 3)
#8:                    Il metodo di bisezione è applicabile
#9:   PrecisionDigits := 6
#10:  bisect(0, 3)
```

Dopo una dozzina di iterazioni si ottiene una buona approssimazione dello zero compreso nell'intervallo specificato: 1.41421.

Atto secondo, scena prima: il punto fisso

Anche in questo caso si tratta di algoritmo certamente ben noto a tutti: si tratta di individuare uno zero della funzione $f(x) := x - F(x)$, ovvero di determinare una soluzione approssimata dell'equazione $x = F(x)$.
Data la funzione $f(x) := x - F(x)$, a partire dal valore iniziale x_0 (detto anche *seme* o *valore di innesco*) viene generata la successione:

$$x_n := \begin{cases} F(x_{n-1}), & se \quad n > 1 \\ x_0 & altrimenti \end{cases}$$

In altre parole:

x_0
$x_1 = F(x_0)$
$x_2 = F(x_1) = F(F(x_0))$
$x_3 = F(x_2) = F(F(F(x_0)))$
. . . . . .

Se allora a è uno zero della funzione $f(x)$ e a è *punto fisso* di $F(x)$.
L'implementazione dell'algoritmo con *Derive* è immediata:

```
#1:   PFISSO di Sebastiano Cappuccio
#2:   F(x) :=
#3:   P_FISSO(a) := ITERATES(F(x), x, a)
```

Interessante e suggestiva è la sua interpretazione geometrica: in sostanza si tratta di determinare una approssimazione

delle coordinate del punto di intersezione (se esiste) tra il grafico della retta $y = x$ e quello della funzione $y = F(x)$.
Vediamo una possibile realizzazione con *Derive*. Iniziamo con il preparare lo scenario tracciando il grafico della retta e della funzione $F(x)$; scegliamo l'esempio più classico perché più "fotogenico", la funzione cos$x$:

```
#4:   PrecisionDigits := 6
#5:   x
#6:   F(x) := COS(x)
#7:   P_FISSO(1)
```

Dopo aver invocato la funzione P_FISSO con il valore di innesco 1, assegniamo il vettore ottenuto, che qui non riportiamo per non sprecare spazio, alla variabile *v*.
Passiamo ora alla costruzione della "ragnatela":

```
#9: innesco := [ ELEMENT(v, 1)          0          ]
               [ ELEMENT(v, 1)  F(ELEMENT(v, 1)) ]

#10:grafico := VECTOR([[  ELEMENT(v,k)    F(ELEMENT(v, k))   ]
                       [F(ELEMENT(v,k))   F(ELEMENT(v, k))   ]
                       [F(ELEMENT(v,k)) F(ELEMENT(v, k + 1)) ]]
                                        , k, 1, DIMENSION(v) - 1)

#11:La seguente matrice premette di disegnare la 'ragnatela'
                                    ; rappresentata da GRAFICO
#12: INNESCO rappresenta il lato iniziale.
#13:[grafico, innesco]
#14:Selezionare Approssima.
#15:Selezionare Plot, senza dimenticare di selezionare
                      l'opzione di visualizzazione che
                         permette di collegare i punti.
```

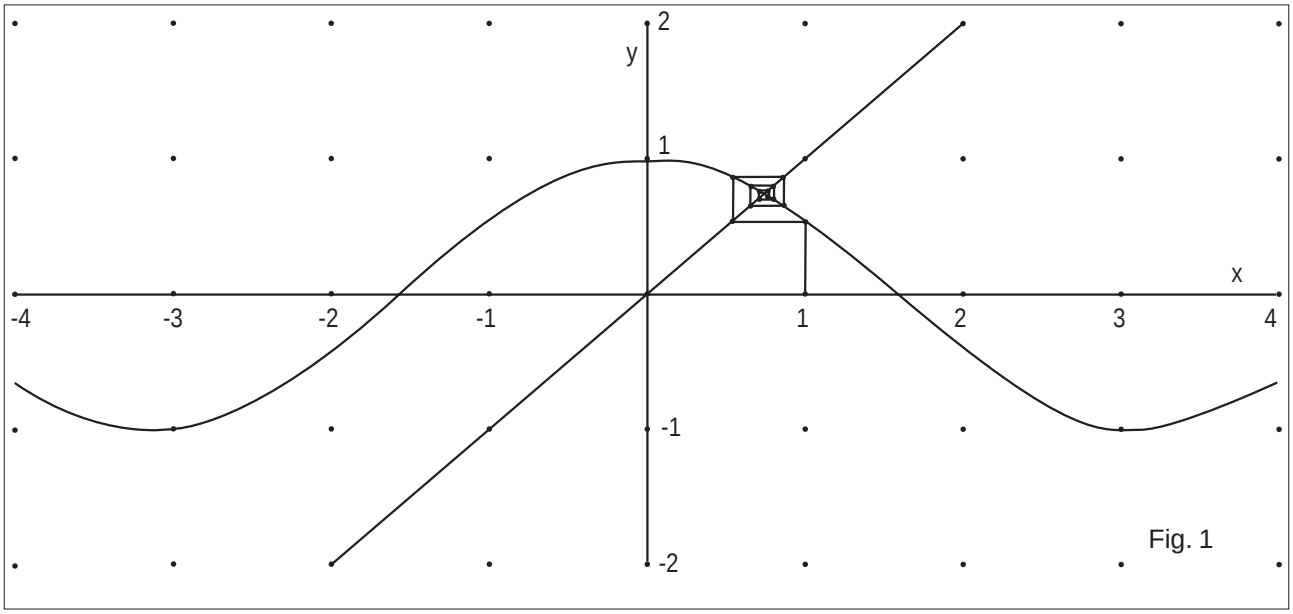


Fig. 1

In figura 1 appare il risultato ottenuto con *Derive*: si noti che il punto fisso $\alpha \approx 0.739085$ sembra attirare verso di sé i

valori della successione; si comporta cioè come un *attrattore*.
A questo punto può essere interessante andare ad applicare questo procedimento a vari casi e, avendo un po' di tempo a disposizione, lasciare liberi gli studenti di porsi domande cercando di formulare alcune congetture. Sarà poi compito dell'insegnante, oltre a offrire lo stimolo, validare o meno tali congetture.
Ecco alcuni risultati che possono emergere da questa esplorazione:

- La scelta del valore di innesco è indifferente? Se un punto fisso è un attrattore, lo è qualsiasi sia il punto di innesco? La risposta è no, almeno in generale. L'insieme dei valori di innesco che generano una successione convergente si chiama *bacino di attrazione* del punto fisso.
- La soluzione dell'equazione è sempre un attrattore? Cioè la successione è sempre convergente, se c'è l'intersezione tra $y = f(x)$ e $y = x$? La risposta è no: in certi casi può essere un repulsore (successione divergente).
  In certi casi poi la successione è sì convergente e addirittura monotona, ma con una convergenza lentissima, tanto da mettere a dura prova il nostro programma: ad esempio, $x = sinx$, con $x_0 = 2$.
- Esiste un criterio di convergenza della successione, cioè una condizione affinché il punto fisso a sia un attrattore? La risposta è sì: fatte opportune ipotesi [(1)], condizione sufficiente perché la successione x_n sia convergente ad a è che sia $|F'(x)| < 1$ per ogni x appartenente a un intorno di a, e ciò qualunque sia il valore di innesco x_0 appartenente ad $[a, b]$.

La dimostrazione del teorema non è difficilissima e oltre che per induzione, può essere fatta usando il Teorema del Valor Medio. Ma è giunto il momento di cambiare scena.

Atto secondo, scena seconda: il bacino di attrazione
Un modo per evidenziare il bacino di attrazione di un punto fisso è quello di iterare per un "ragionevole numero di volte" la funzione $F(x)$ in gioco.

Il grafico della funzione $cosx$ iterata 10 volte (Fig.2) suggerisce che $x - cosx$ abbia l'unico zero $x = 0.7390851332$ e che il suo bacino di attrazione sia l'intera retta reale.

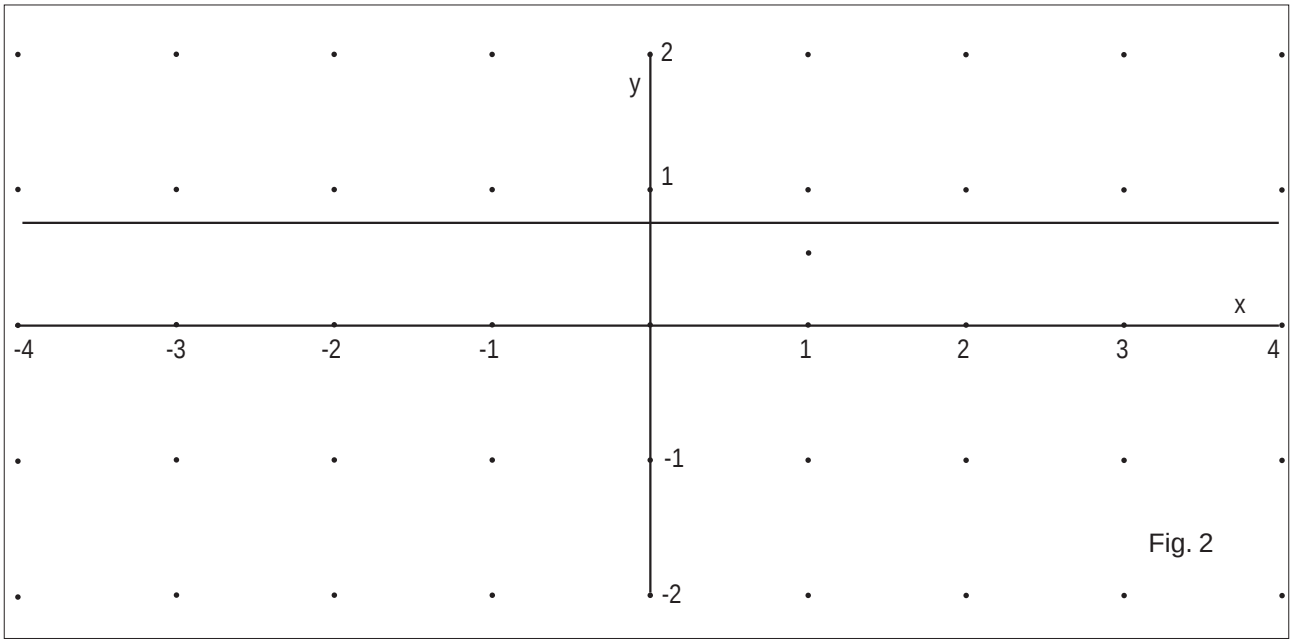


Fig. 2

Il grafico della funzione (2-x^3)/4 iterata 5 volte (Fig.3) suggerisce il bacino di attrazione dello zero $x = 0.4734658077$:

Analogamente il grafico della funzione $atan(2x)$ iterato 10 volte (Fig.4) suggerisce il bacino di attrazione di ciascuno dei due zeri di $x - atan(2x)$:

[(1)] α sia un punto fisso di $F(x)$. La funzione F sia definita, continua, derivabile nell'intervallo $[a, b]$ e tale che $F(x)$ appartenga a $[a, b]$ per ogni x dell'intervallo.

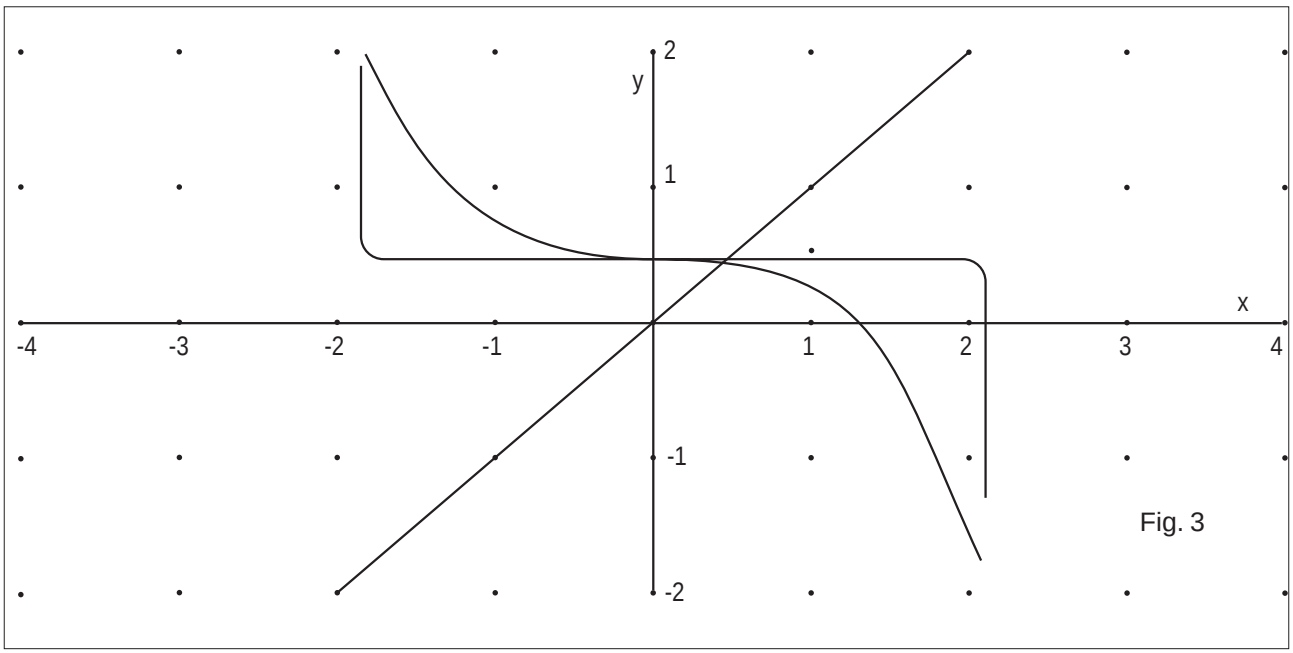


Fig. 3

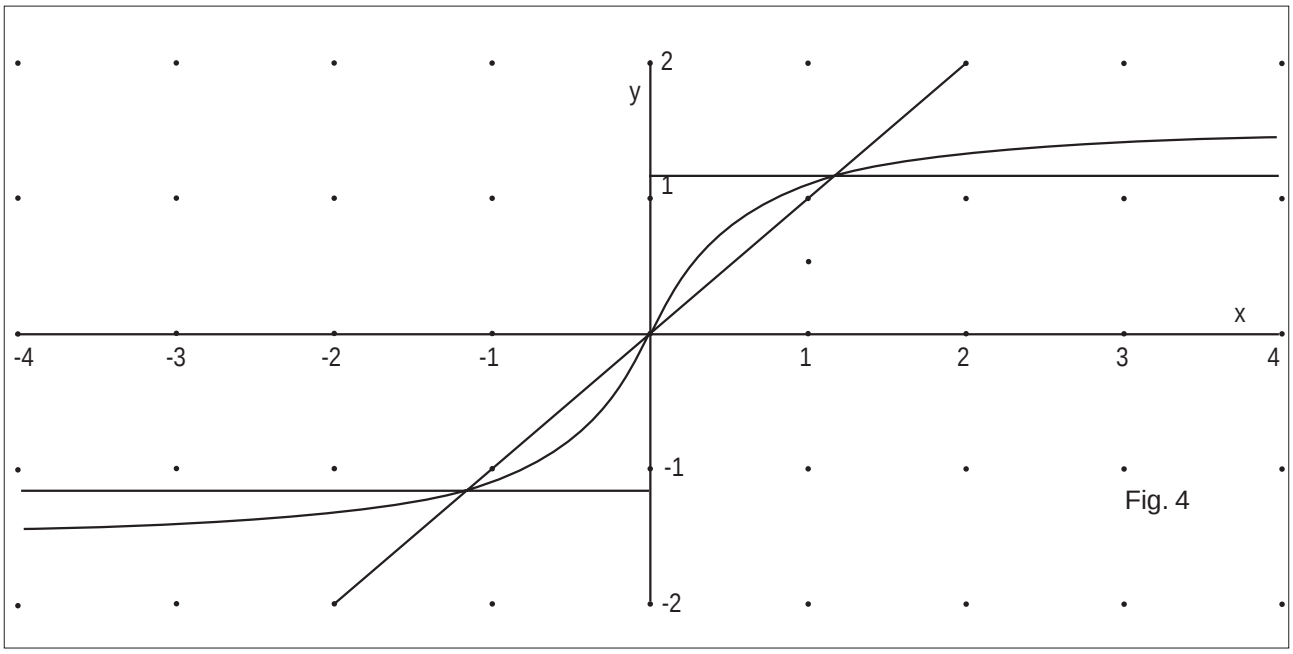


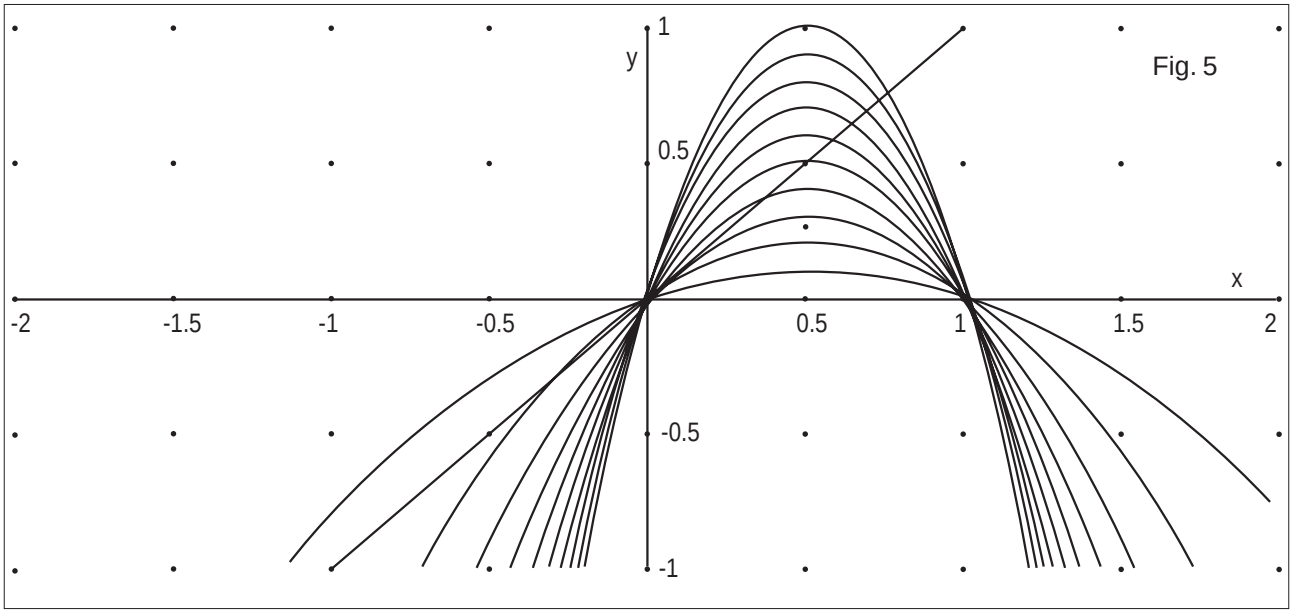
Fig. 4

Atto secondo, scena terza: la visualizzazione delle successioni ovvero la prima comparsa del Caos

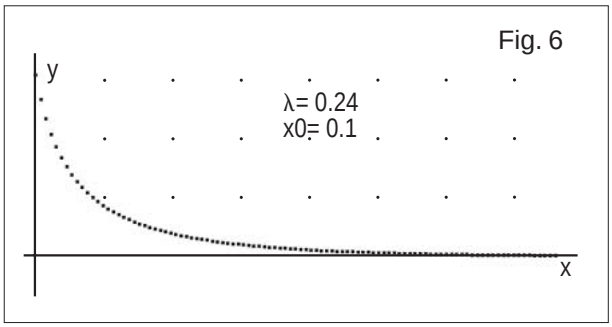
Un modo per visualizzare il comportamento di una successione è quello di rappresentare un numero "ragionevolmente grande" di valori di x_n trasformando la successione in un vettore di punti con ascissa $1, 2, \ldots n$ e ordinata rispettivamente $x_1, x_2, \ldots x_n$.

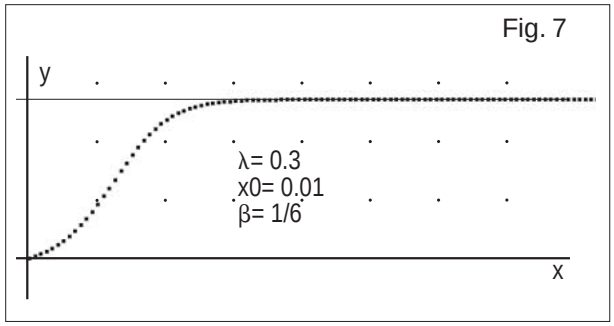
```
#1:                          Biforcazione - di S. Cappuccio
#2:   F(x) := 4·λ·x·(1 - x)
#3:   λ :=
#4:   fisso(x0, n) := ITERATES(F(x), x, x0, n)
#5:   v := fisso(x0, n)
#6:   succ := VECTOR([k, ELEMENT(v, k + 1)], k, 0,
                                   DIMENSION(v) - 1)
```

Questo tipo di rappresentazione è particolarmente utile in un caso interessante.
Consideriamo una funzione $F(x)$ molto semplice: $F(x) := 4\lambda x(1 - x)$ e studiamone il comportamento in funzione del parametro λ, con $0 < \lambda < 1$.
Ovviamente di tratta di una famiglia di parabole con la concavità "verso il basso" che intersecano l'asse delle ascisse nei punti di ascissa 0 e 1 e hanno quindi vertice nel loro punto di ascissa 1/2.
La situazione in esame sarà quella descritta in figura 5 nella quale si mostra l'intersezione tra la retta $y = x$ e la famiglia di parabole:

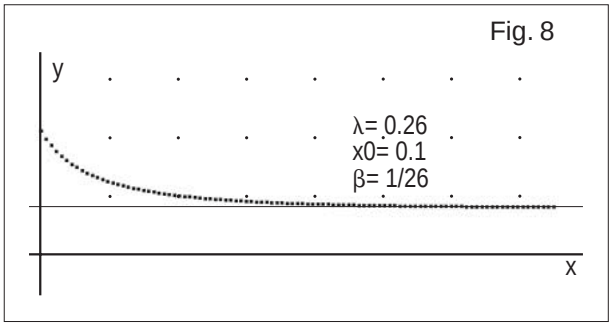


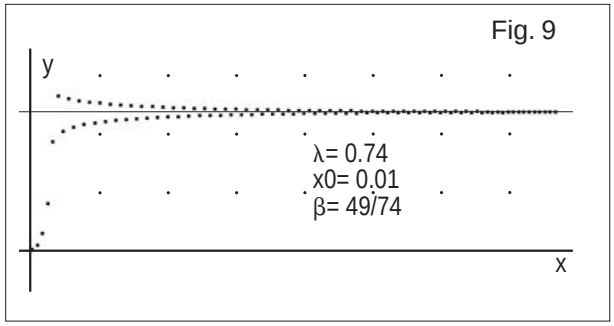
È facile calcolare le soluzioni dell'equazione $x = F(x)$: $\alpha = 0$, $\beta = (4\lambda - 1) / 4\lambda$
È possibile dimostrare che α è un attrattore per $0 < \lambda < 1/4$; in figura 6 viene mostrato un esempio; il valore del parametro e il valore iniziale sono indicati direttamente in figura.



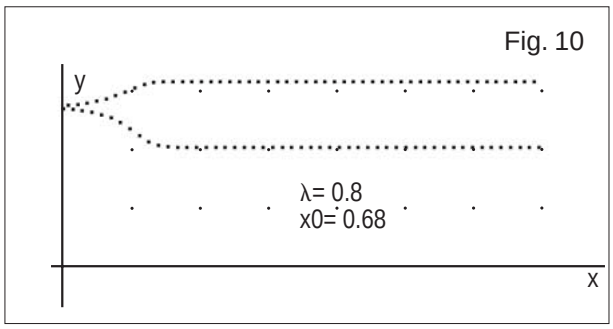


Ancora, si dimostra che con $1/4 < \lambda < 3/4$ l'attrattore sarà β; ecco tre esempi: si tratta, rispettivamente, di una successione monotona crescente (Fig.7), monotona decrescente (Fig.8) e, per così dire, alternata (Fig.9). In ogni caso comunque si ha la convergenza a β.

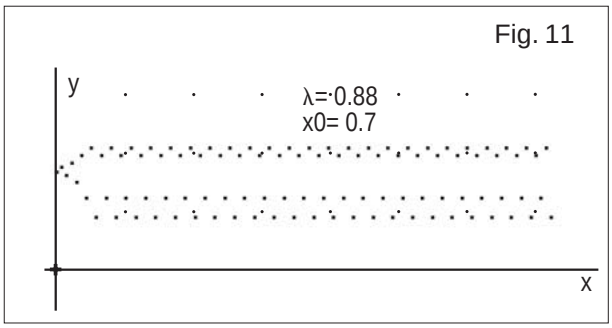


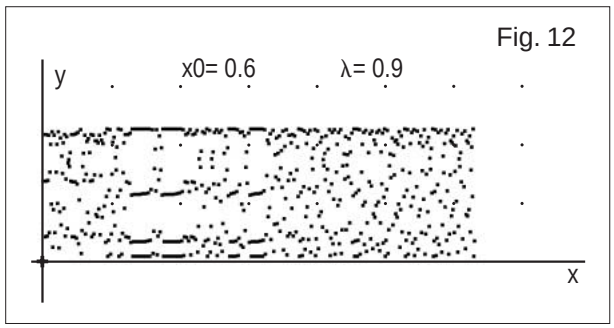


Per ciò che abbiamo detto, valori di λ maggiori di 3/4 daranno origine a successioni non convergenti a β, ma la visualizzazione (Fig.10) ci riserva qualche sorpresa:



È come se la successione convergesse (se ci è consentita l'espressione) alternativamente a due valori diversi β_1 e β_2. La coppia (β_1, β_2) viene chiamata in gergo tecnico *attrattore di periodo 2* o *ciclo 2*.
Se aumentiamo il valore di λ abbiamo un *ciclo* 4 (Fig.11):





Con un valore di λ prossimo a 1 si generano situazioni caotiche (Fig. 12):

Atto terzo, scena prima: algoritmo di Newton
Non stiamo qui a descrivere in dettaglio l'algoritmo di Newton-Raphson, noto a tutti. Limitiamoci a osservare che non sempre il metodo del punto fisso è efficace: ad esempio per la pur banale equazione $x^2 - 1 = 0$, se usiamo il metodo del punto fisso traformando l'equazione in $x = 1/x$, avremo la successione: x_0, $1/x_0$, x_0, $1/x_0$, … ovviamente oscillante qualunque sia il valore iniziale (diverso da zero).
Alla stessa equazione può invece essere applicato l'algoritmo di Newton. *Derive* possiede già una funzione predefinita per l'algoritmo di Newton, ma per ora ne useremo una "fatta in casa".
Eccone una possibile implementazione:

```
#1:   Funzione NEWTON(b) - S. Cappuccio
#2:   Newton(b) fornisce una successione di valori che
                                  approssimano uno zero
#3:   di F(x) partendo dal valore iniziale b con il
                                   metodo di Newton.
#4:   F(x) :=
#5:   NEWTON(b) := ITERATES(x - F(x) / (d/dx)^1 F(x), x, b)
#6:   F(x) := x^2 - 1
#7:   NEWTON(3)
#8:   [3, 1.666666666, 1.133333333, 1.007843137,
                              1.000030518, 1, 1]
#9:   NEWTON(-2)
#10:  [-2, -1.25, -1.025, -1.000304878, -1, -1]
```

Gli esempi visti suggeriscono che i bacini di attrazione dei due zeri della funzione, 1 e –1 sono, rispettivamente, il semiasse delle ascisse positive e quello delle ascisse negative.
Si noti anche la notevole "velocità di convergenza" della successione.
Anche in questo caso possiamo facilmente visualizzare l'interpretazione grafica dell'algoritmo con la funzione GRAFICO:

```
#11:  GRAFICO(b) := VECTOR([[  ELEMENT(NEWTON(b), k)
                               ELEMENT(NEWTON(b), k)
                            ELEMENT(NEWTON(b), k + 1)
                                        0
                            F(ELEMENT(NEWTON(b), k))  ], k, 1,
                                        0
                            DIMENSION(NEWTON(b)) - 1  ])
```

ottenendo il grafico riportato in Fig.13:

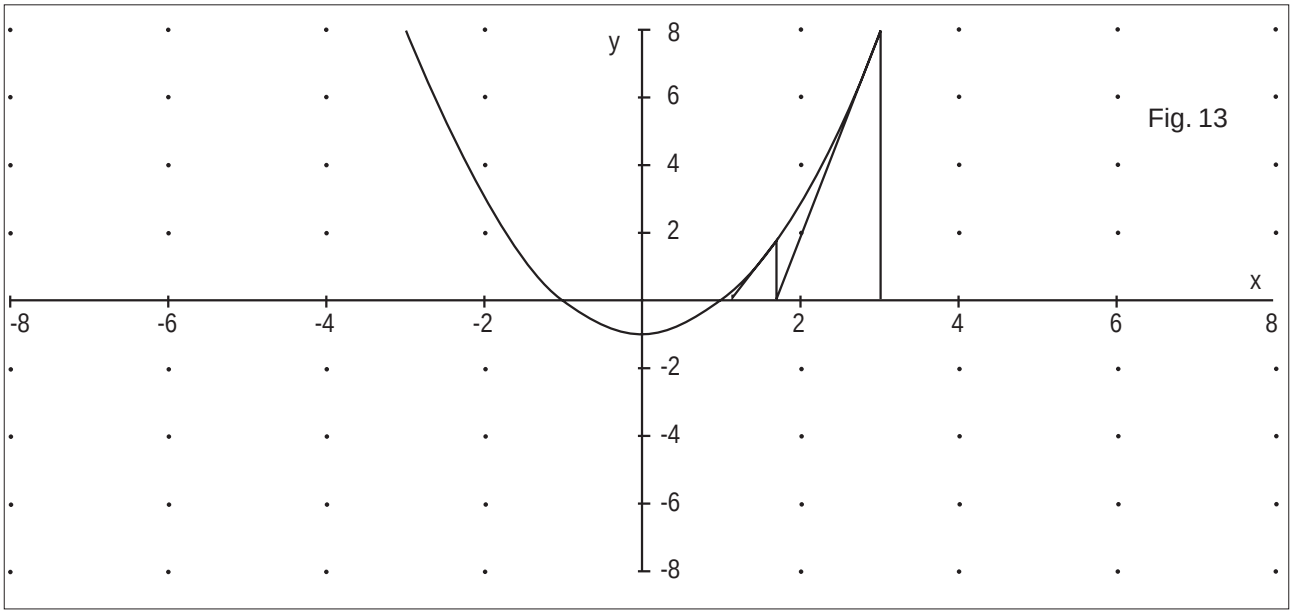


Fig. 13

Fin qui, nulla di particolarmente eccitante. Nel 1879 però sir Arthur Cayley ebbe la geniale intuizione di applicare l'algoritmo anche al campo complesso, cercando di individuare i bacini di attrazione degli zeri delle equazioni $x^2 - 1 = 0$ e $x^3 - 1 = 0$.
Citiamo le sue parole: "*La soluzione del problema è facile ed elegante nel caso delle equazioni quadratiche, ma già le equazioni cubiche presentano difficoltà non lievi*."
Per esplorare questa situazione useremo invece della funzione prima definita, la funzione predefinita **NEWTON(u, *x*, *x*0, n)** che, citiamo le parole del manuale *on line* di *Derive* [(2)], "restituisce un vettore di **n+1** approssimazioni della variabile ***x*** risultanti dall'applicazione di **n** volte del metodo di Newton all'espressione **u(*x*)**, iniziando dal valore ***x*0**. **NEWTON(u, *x*, *x*0)** restituisce un vettore di approssimazioni per la ***x*** fino ad ottenere la convergenza con la precisione corrente".*(vedi schermata a pag. 47)*
La funzione ITERAZIONI(a) attiva l'algoritmo di Newton a partire dal valore iniziale (complesso) *a*, la funzione PERCORSO(a) genera un vettore di punti che rappresentano le successive approssimazioni dello zero della funzione $z^2 - 1$ sul piano complesso, sempre a partire dal valore iniziale *a*. Questo vettore sarà poi visualizzato (attivando l'opzione di collegamento dei punti) sul piano.
In figura 14 appaiono i percorsi a partire da alcuni valori iniziali scelti a casaccio: si noti che tutte le successioni di punti che hanno punto iniziale sul semipiano delle ascisse positive convergono verso 1, mentre tutte le successioni che hanno

[(2)] La funzione è contenuta nella libreria SOLVE.MTH, che nelle ultime versioni di *Derive* viene caricata automaticamente.

```
#1:   Bacini di convergenza di z^2-1  con Newton in campo
                                    complesso (S. Cappuccio)
#2:   z :∈ Complex
#3:   iterazioni(a) := NEWTON(z^2 - 1, z, a)
#4:   percorso(a) := VECTOR([RE((iterazioni(a))_k),
                      IM((iterazioni(a))_k)], k, 1,
                         DIMENSION(iterazioni(a)))
```

punto iniziale sul semipiano delle ascisse negative convergono verso – 1. Se il punto iniziale si trova sull'asse immaginario, la successione… fa la fine dell'asino di Buridano.

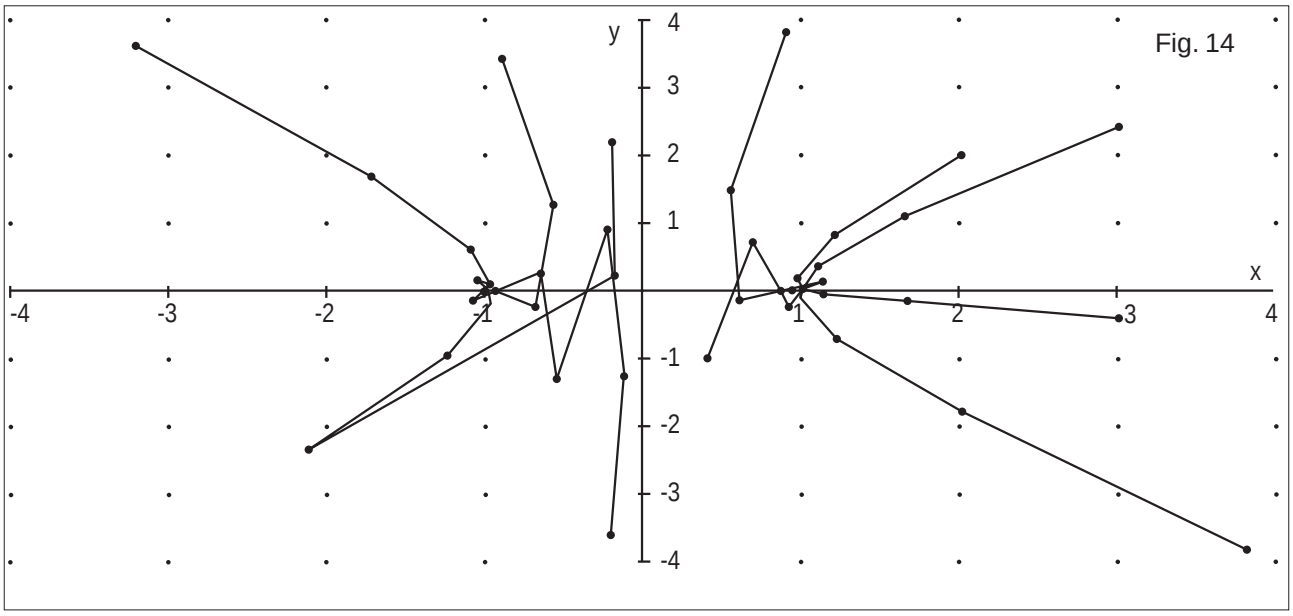


Fig. 14

In questo caso è quindi facile congetturare ciò che Cayley dimostrò, cioè che i bacini di attrazione di 1 e –1 sono, rispettivamente, i semipiani $x > 0$ e $x < 0$.

Nel caso della equazione $z^3 - 1 = 0$ la situazione non sembra molto diversa: viene spontaneo congetturare che la situazione sia quella mostrata in figura 15, nella quale le tre radici cubiche dell'unità si dividono fraternamente la loro zona di influenza sul piano complesso in tre "spicchi" uguali:

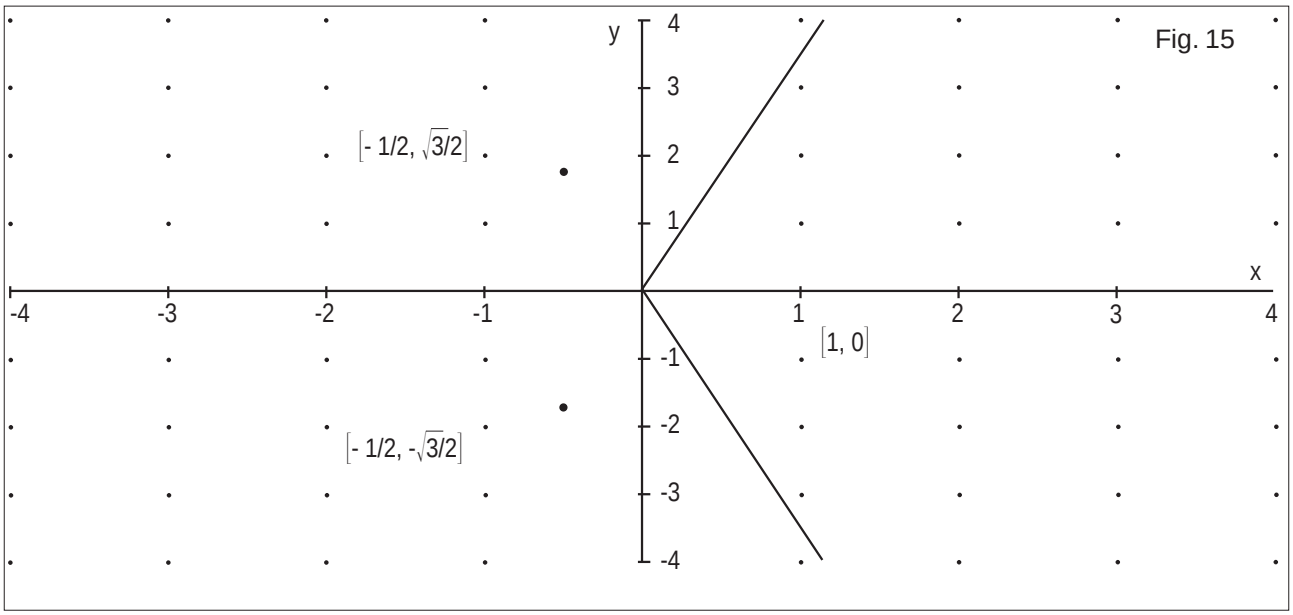


Fig. 15

Useremo le stesse funzioni ITERAZIONI e PERCORSO del caso precedente, modificando solo l'equazione in gioco. Questa volta però le cose vanno in modo meno scontato: mentre in molti casi i percorsi sono quelli prevedibili, in altri (in particolare quelli "vicini" alle frontiere tra i bacini di attrazione previsti) hanno un comportamento del tutto imprevedibile.

Tutto si spiega se ricorriamo a un programma che "accenda" ogni *pixel* dello schermo del computer in relazione al valore a cui converge la successione che ha come punto iniziale quello corrispondente al *pixel* stesso (Fig.16).

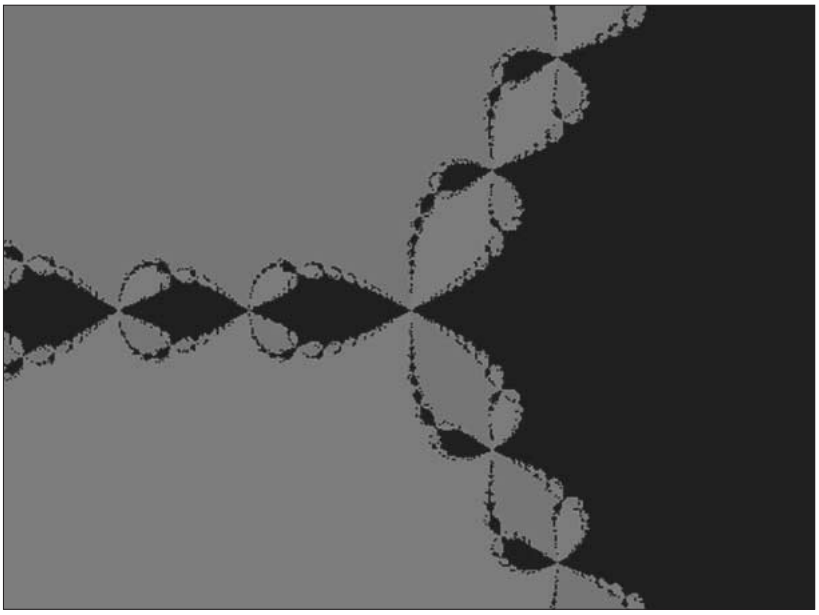


Fig. 16

L'ultima versione di *Derive* ci permette di utilizzare l'immagine come sfondo e quindi di spiegare il motivo dei comportamenti precedenti (Fig.17).

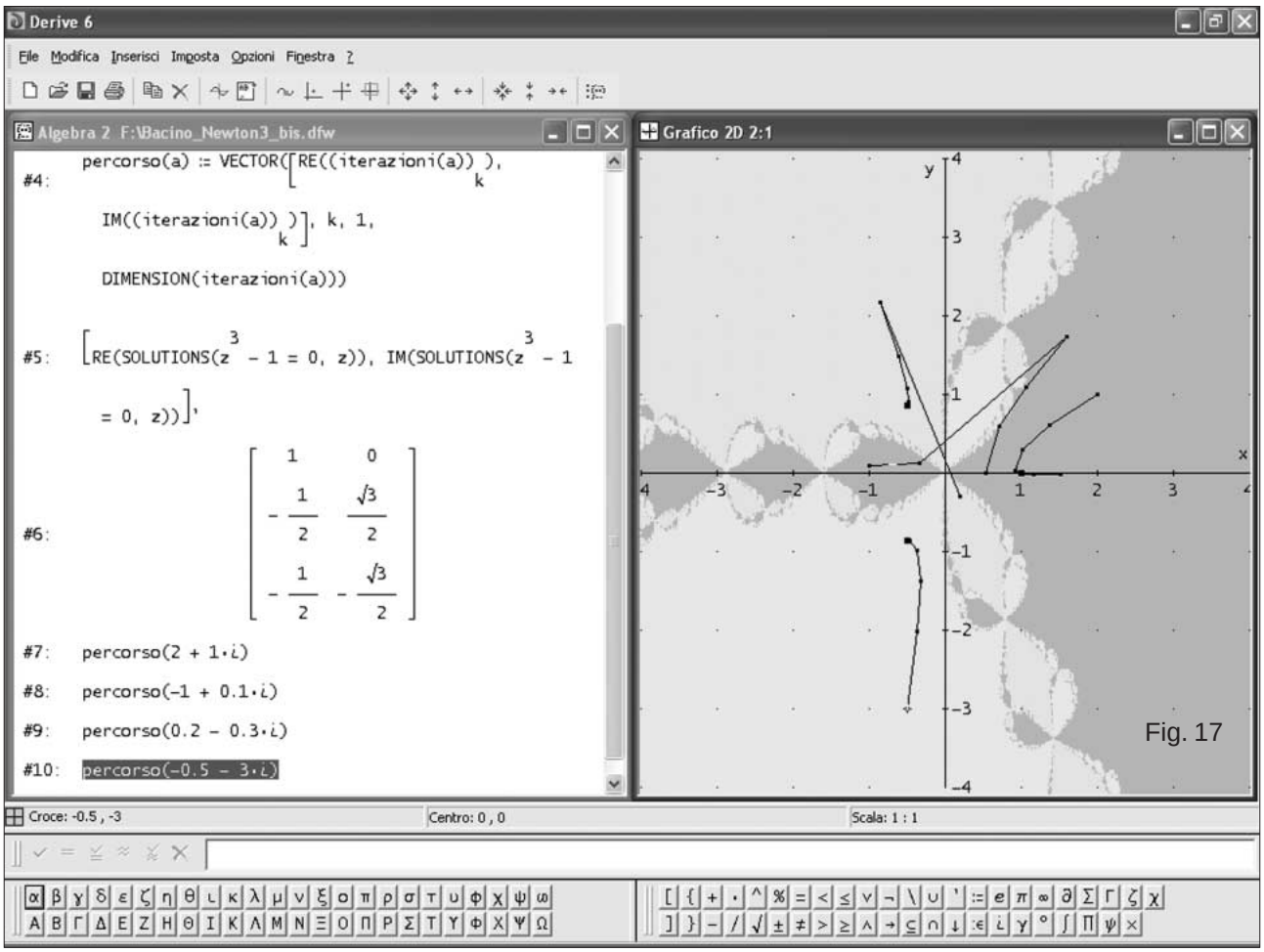


Fig. 17

Si noti che la posizione del punto che rappresenta il valore iniziale della successione, determina da quale degli zeri la successione sarà "attirata".

La figura 16 è un classico esempio di figura *autosimilare*, cioè, detto in parole povere, tale che ingrandendo una sua opportuna parte, otteniamo una figura uguale alla precedente; questa è una delle caratteristiche delle cosiddette figure *frattali*.
Sono disponibili numerosi programmi specificamente progettati per ottenere figure di questo tipo e non è neppure difficile costruirsene uno con un linguaggio di programmazione, tuttavia anche con *Derive* si riescono a realizzare immagini interessanti. Vediamo un paio di esempi.

Atto terzo, scena seconda: la curva di Koch
Altra famosa figura frattale (e classico esempio di ricorsività grafica) è la Curva (o *Trina*) di Koch, presentata nel 1904 dal matematico svedese Helge von Koch: prendiamo un segmento, dividiamolo in tre parti uguali, costruiamo il triangolo equilatero che ha per lato la parte centrale e eliminiamo la parte centrale stessa; ripetiamo lo stesso procedimento su ciascuno dei segmenti della nuova figura, e così via …
Ecco come può essere realizzata in *Derive*; si noti l'idea semplice ma molto ingegnosa dell'Autrice del programma, Maria Koth (Università di Vienna) di esprimere il vertice del triangolo equilatero attraverso la funzione NV(v) che è il vettore normale di v con uguale lunghezza e ruotato a sinistra.

```
#1:  Curva di Koch (Maria Koth)-DNL # 33
#2:  NV(U) := [ 0  -1 ] · U
              [ 1   0 ]
#3:  GENERATORE(X, Y) := [X, (2·X + Y)/3, (X + Y)/2 + √3/6·NV(Y - X), (X + 2·Y)/3]
#4:  successivo(v) := APPEND(VECTOR(GENERATORE(v_i, v_(i+1)), i, DIMENSION(v) - 1),
       [[v_DIMENSION(v)]])
     koch(k) :=
       If k = 0
#5:       [0, 0; 1, 0]
          APPEND(successivo(koch(k - 1)))
```

Ecco di seguito le immagini, con opportune scelte nelle opzioni grafiche, delle curve di Koch rispettivamente di ordine 1, 2, 3 e 4:

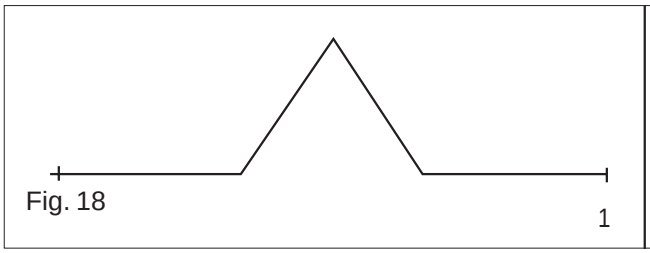


Fig. 18

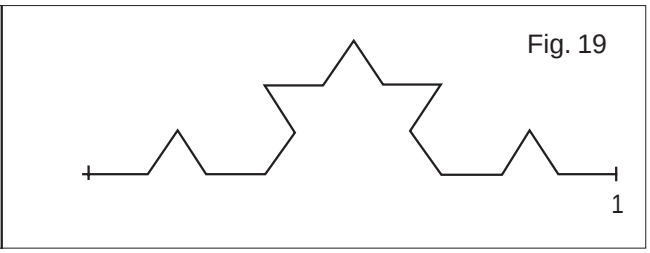


Fig. 19

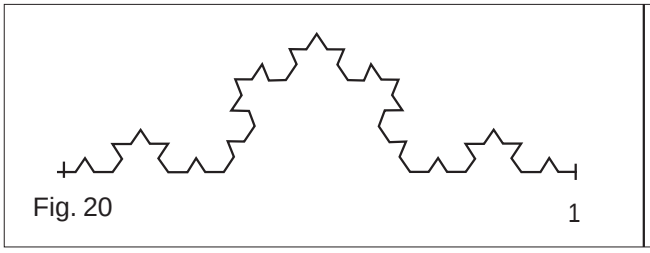


Fig. 20

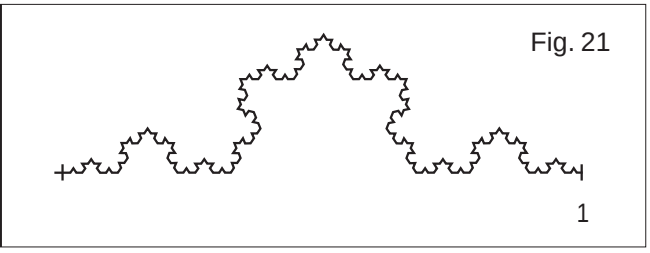


Fig. 21

Si potrebbe dire molto su questa famosa curva; al limite per n tendente a infinito (ove n è il numero delle iterazioni) si ottiene un classico esempio di curva continua ma non differenziabile in tutti i suoi punti. Due problemi interessanti e non difficilissimi da sottoporre agli studenti sono quelli di trovare il perimetro e l'area del "fiocco di neve" (che è generato dallo stesso algoritmo, partendo dai tre lati di un triangolo equilatero invece che da un solo segmento) dopo n iterazioni.

Atto terzo, scena terza: il triangolo di Sierpinski e il Gioco del Caos

Un'altra famosa figura frattale è il triangolo (o *gerla*) presentato dal matematico polacco Waclaw Sierpinski nel 1915, che viene così ottenuto: da un triangolo (per comodità equilatero) "tagliamo via" il triangolo che ha per vertici i punti medi dei lati, poi iteriamo il procedimento ai triangoli rimanenti.

Ecco come appare il triangolo iniziale, e dopo una, due, tre e quattro iterazioni (Fig.22):

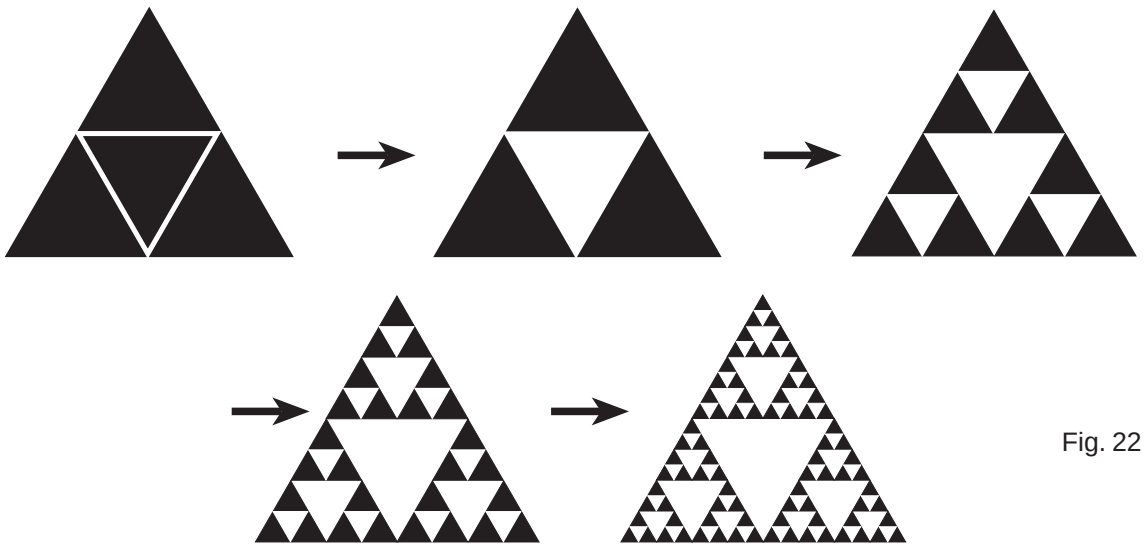


Fig. 22

Immagini come queste possono essere ottenute anche con *Derive* con un procedimento non dissimile da quello usato per la curva di Koch.

La cosa sorprendente è la possibilità di ottenere un risultato analogo con una facile costruzione che potremmo chiamare "probabilistica": il *Gioco del Caos*, presentato da Michael Barnsley nel 1985.

Dato un triangolo e un punto qualsiasi P sul piano, costruiamo il punto medio del segmento da P a uno dei vertici del triangolo, scelto a caso, poi il punto medio tra quello ora trovato e un altro vertice del triangolo, anch'esso scelto a caso, e così via ...

Questa costruzione è facilissima da realizzare in *Derive.*

```
#1:  Costruzione 'probabilistica' del Triangolo di Sierpinski - S. Cappuccio
#2:  p := [ 0   0  ]
          [ 2   0  ]
          [ 1   √3 ]
#3:  midpoint(a, b) := [ ELEMENT(a, 1) + ELEMENT(b, 1)   ELEMENT(a, 2) + ELEMENT(b, 2) ]
                       [ ---------------------------, ---------------------------     ]
                       [              2                             2                  ]
#4:  trian(n, a, b) := ITERATES(midpoint(u, ELEMENT(p, RANDOM(3) + 1)), u, [a, b], n)
#5:  trian(30, 5, 4)
#6:  trian(300, 5, 4)
#7:  trian(3000, 5, 4)
#8:  trian(5000, 5, 4)
```

Ecco ciò che si ottiene con 30, 300, 3000 e 5000 iterazioni (Fig.23 E 24):

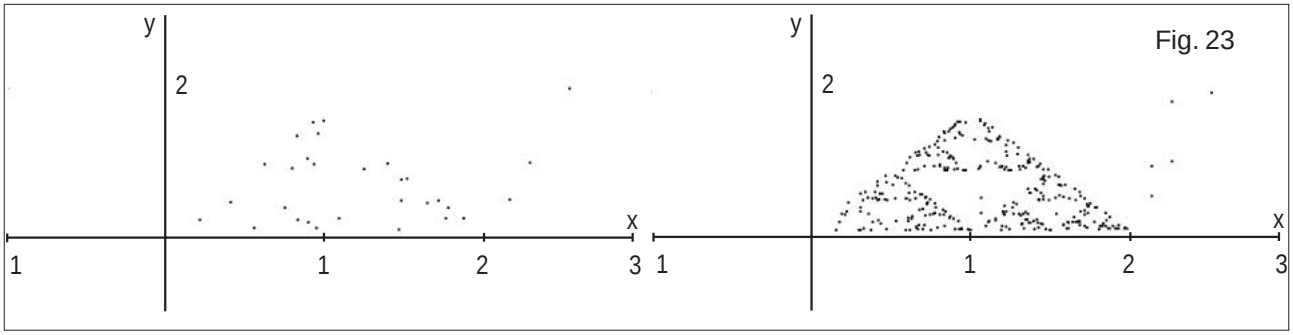


Fig. 23

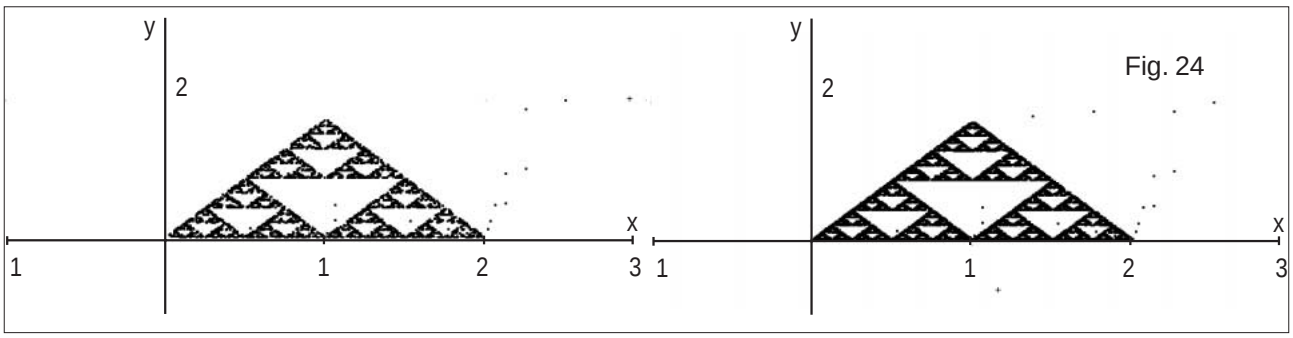
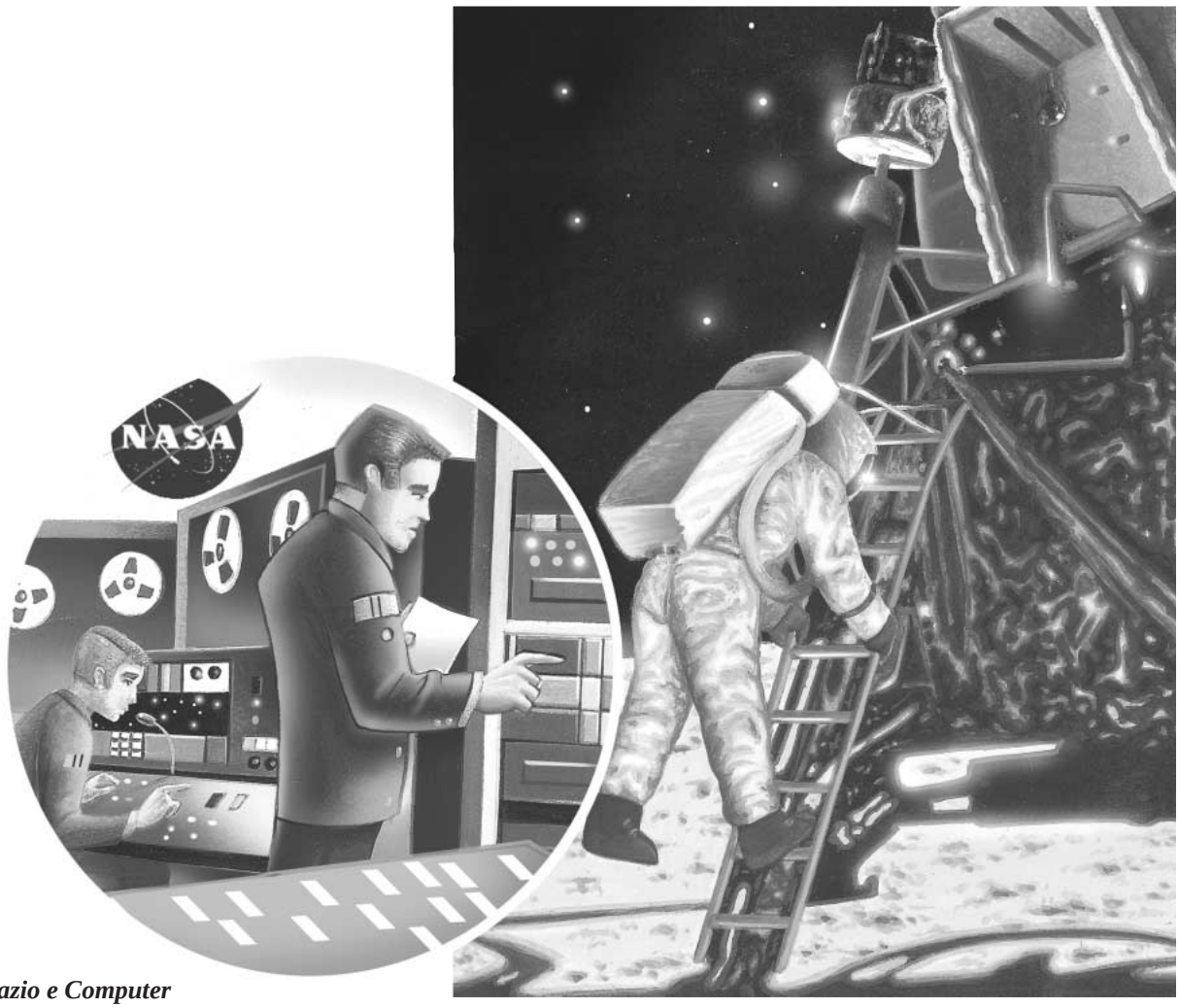


Fig. 24

Epilogo

A questo punto punto si apre una immensa possibilità di sviluppi; in particolare si dovrebbe introdurre il concetto di *dimensione metrica* (Felix Hausdorff, 1919) e soprattutto l'*insieme di Julia* (Gaston Julia, 1918) per poi arrivare all'*insieme di Mandelbrot* (Benoit Mandelbrot, 1978), con i loro universi di meraviglie grafiche che nascono dalla semplicissima famiglia di funzioni quadratiche in C: $z^2 + c$ (c parametro complesso).
Ma questa è un'altra storia ...



Spazio e Computer

Derive ed il progetto Eccellenza

di Aurelia Orlandoni
IRRE Emilia Romagna

Il progetto Eccellenza

Nel 1998 l'IRRSAE (ora IRRE) Emilia Romagna ha promosso un'attività seminariale per i docenti di matematica di scuola superiore nell'ambito del progetto pluriennale "Eccellenza nel triennio delle superiori"; l'iniziativa si è poi sviluppata nel tempo con il coinvolgimento dell'allora IRRSAE Lazio. Il progetto si propone di cercare di evidenziare l'eventuale guadagno formativo legato all'utilizzo di software per la matematica (*Cabri*, *Derive*, *Mathematica*, ...) e di formulare proposte finalizzate a stimolare negli studenti l'interesse all'approfondimento dello studio della matematica.
Fin dal secondo anno gli insegnanti coinvolti (circa 40 all'anno) hanno lavorato in gruppo, utilizzando la posta elettronica per comunicare e preparare le proposte.
Le attività connesse con il progetto sono state curate ad anni alterni dall'IRRE-ER e dall'IRRE LAZIO e, nei primi quattro anni, si sono concluse con un seminario residenziale in cui ogni gruppo ha presentato le soluzioni elaborate nella fase precedente. I materiali prodotti dai docenti sono stati raccolti in volumi, la cui stampa è stata curata dall'IRRE che in quell'anno coordinava l'esperienza. Tre dei quattro volumi prodotti sono anche scaricabili dai rispettivi siti[(1)].
La riflessione sull'esperienza dei primi quattro anni e la quantità di materiali elaborati ha portato, nell'anno 2003, a focalizzare l'attenzione sull'esperienza in classe.
La proposta è stata orientata allo sviluppo di esperienze di eccellenza realizzate con gli studenti e documentate: i problemi proposti nelle varie aree dovevano essere inseriti dai docenti in un percorso didattico e l'esperienza in classe documentata. Sono stati proposti in rete 6 problemi con alcuni suggerimenti per gli insegnanti e le attività sono state coordinate da 12 insegnanti.
Entro Pasqua 2004 saranno disponibili sul sito FARDICONTO i materiali relativi alle esperienze svolte.

Una proposta per il biennio: le frazioni proprie

Un problema che ha suscitato interesse e che si presta a diversi livelli di approfondimento è stato quello proposto e coordinato da E. Crespina e A. Zanni per le attività del 2003:
Si consideri l'insieme delle frazioni proprie aventi denominatore 3, poi quello delle frazioni proprie con denominatore uguale a 4, e così via per i denominatori 5, 6 e 7.
Si esprimano poi questi numeri nella forma decimale.
1. *Cosa si può osservare relativamente ai loro periodi?*
2. *Cosa cambia se i numeri precedenti sono espressi in basi diverse da 10?*
3. *È possibile giustificare o dimostrare le proprietà scoperte?*

Il problema si presta ad attività di esplorazione che tutti gli allievi di un biennio possono affrontare, ma anche ad approfondimenti; inoltre consente una revisione e una sistemazione sulle frazioni e i numeri decimali, ambiti in cui gli studenti di prima superiore spesso operano con difficoltà.
Per rispondere alla prima domanda costruiamo una funzione che restituisca le sequenze di frazioni proposte. Esiste in *Derive* una funzione predefinita **VECTOR(espressione, variabile, vi, vf, passo)** che restituisce la sequenza di valori data da **espressione** al variare di variabile dal valore iniziale **vi** al valore finale **vf** con incremento determinato da **passo**. Il **passo** può essere omesso nel caso sia 1.
Per ottenere tutte le sequenze è sufficiente, dopo avere definito la funzione **frazione (n, d)**, sostituire a *d* i vari denominatori, semplificare l'espressione e approssimarla. Di seguito sono riportati i valori ottenuti utilizzando una precisione di 10 cifre decimali.

#1: $\text{frazione}(n, d) := \text{VECTOR}\left(\frac{n}{d}, n, 1, d - 1\right)$

#2: $\text{frazione}(n, 3)$

#3: $\left[\frac{1}{3}, \frac{2}{3}\right]$

(1) IRRE-ER: http://kidslink.scuole.bo.it/fardiconto e IRRE LAZIO: http://www.irrsae.lazio.it/matema/index.html

#4: [0.3333333333, 0.6666666666]

#5: frazione(n, 4)

#6: $\left[\frac{1}{4}, \frac{1}{2}, \frac{3}{4}\right]$

#7: [0.25, 0.5, 0.75]

#8: frazione(n, 5)

#9: $\left[\frac{1}{5}, \frac{2}{5}, \frac{3}{5}, \frac{4}{5}\right]$

#10: [0.2, 0.4, 0.6, 0.8]

#11: frazione(n, 6)

#12: $\left[\frac{1}{6}, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}, \frac{5}{6}\right]$

#13: [0.1666666666, 0.3333333333, 0.5, 0.6666666666, 0.8333333333]

#14: frazione(n, 7)

#15: $\left[\frac{1}{7}, \frac{2}{7}, \frac{3}{7}, \frac{4}{7}, \frac{5}{7}, \frac{6}{7}\right]$

#16: [0.1428571428, 0.2857142857, 0.4285714285, 0.5714285714, 0.7142857142, 0.8571428571]

Gli studenti possono osservare che *Derive* riduce le frazioni ai minimi termini in modo automatico, e che si ottengono sia numeri decimali finiti che periodici e che il periodo a volte è preceduto da un antiperiodo.
Questa esplorazione consente di formulare la congettura che le frazioni proposte danno origine a:

- numeri decimali limitati quando i soli divisori del denominatore sono 2 e 5,
- numeri periodici semplici se fra i divisori del denominatore non compaiono né 2 né 5,
- numeri periodici con antiperiodo se fra i divisori compaiono anche 2 o 5, ma non solo.

La semplicità con cui si sono ottenuti i risultati consente di ampliare il campo di esplorazione anche a denominatori > 7, in modo da rafforzare la convinzione sulle ipotesi fatte.
Il secondo quesito riguarda i cambiamenti di base. Selezionando **Opzioni Modalità**, si apre una finestra in cui è possibile scegliere sia la base in cui si scrivono i numeri (**Input**) che quella in cui si vogliono i risultati (**Output**). In modo semplice e automatico è possibile trasformare le frazioni precedenti in basi diverse. A titolo di esempio sono riportati i risultati ottenuti trasformando le frazioni proprie di denominatore 6 in base 2, 3, 4, 5.

#17: frazione(n, 6)

base 10

#18: $\left[\frac{1}{6}, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}, \frac{5}{6}\right]$

#19: [0.1666666666, 0.3333333333, 0.5, 0.6666666666, 0.8333333333]

base 2

#20: $\left[\frac{1}{110}, \frac{1}{11}, \frac{1}{10}, \frac{10}{11}, \frac{101}{110}\right]$

#21: [0.001010101010, 0.01010101010, 0.1, 0.1010101010, 0.1101010101]

base 3

#22: $\left[\frac{1}{20}, \frac{1}{10}, \frac{1}{2}, \frac{2}{10}, \frac{12}{20}\right]$

#23: [0.01111111111, 0.1, 0.1111111111, 0.2, 0.2111111111]

```
base 4
#24: [1/12, 1/3, 1/2, 2/3, 11/12]
#25: [0.02222222222, 0.1111111111, 0.2, 0.2222222222, 0.3111111111]
base 5
#26: [1/11, 1/3, 1/2, 2/3, 10/11]
#27: [0.04040404040, 0.1313131313, 0.2222222222, 0.3131313131, 0.4040404040]
base 6
#28: [1/10, 1/3, 1/2, 2/3, 5/10]
#29: [0.1, 0.2, 0.3, 0.4, 0.5]
```

Agli studenti occorreranno diversi tentativi per arrivare a formulare la congettura che la periodicità di un numero non dipende solo dalla frazione di partenza ma dalla base in cui viene scritta e che l'ipotesi precedente deve essere opportunamente modificata se la base in cui si opera è diversa da 10. Analizzando i risultati ottenuti si può formulare la congettura che le frazioni danno origine a:

- numeri decimali limitati quando i soli divisori del denominatore sono gli stessi della base,
- numeri periodici semplici se fra i divisori del denominatore non compare nessuno dei divisori della base, cioè se il denominatore e la base sono primi fra loro,
- numeri periodici con antiperiodo se i divisori del denominatore non sono solo divisori della base.

Possiamo quindi concludere che una frazione dà luogo sempre a numeri periodici di periodo 0 o diverso da 0 e che il periodo può essere preceduto da un antiperiodo.
A questo punto l'insegnante valuterà l'opportunità di passare alla dimostrazione di queste ultime affermazioni[(2)].
A partire da questo problema è immediato aprire una riflessione sulla rappresentazione dei "numeri macchina", argomento non sempre affrontato nella scuola superiore e che, nel momento in cui vengono trattate le strutture algebriche, consente di sottolineare analogie e differenze con quelle degli insiemi N, Z, Q, R[(3)].

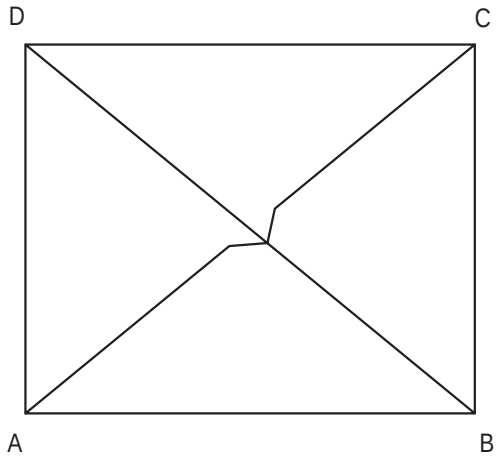
In conclusione sono possibili esplorazioni e approfondimenti molto diversi; infatti fra i contributi pervenuti c'è anche chi ha pensato di "far suonare" i numeri periodici utilizzando una calcolatrice grafico-simbolica, ma per questo rimando alla prossima pubblicazione sul sito FARDICONTO dei materiali relativi alle attività del progetto Eccellenza (vedi nota1).

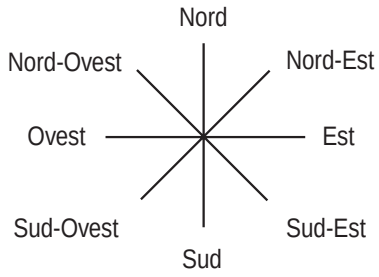
Una proposta per il triennio: la passeggiata di un automa

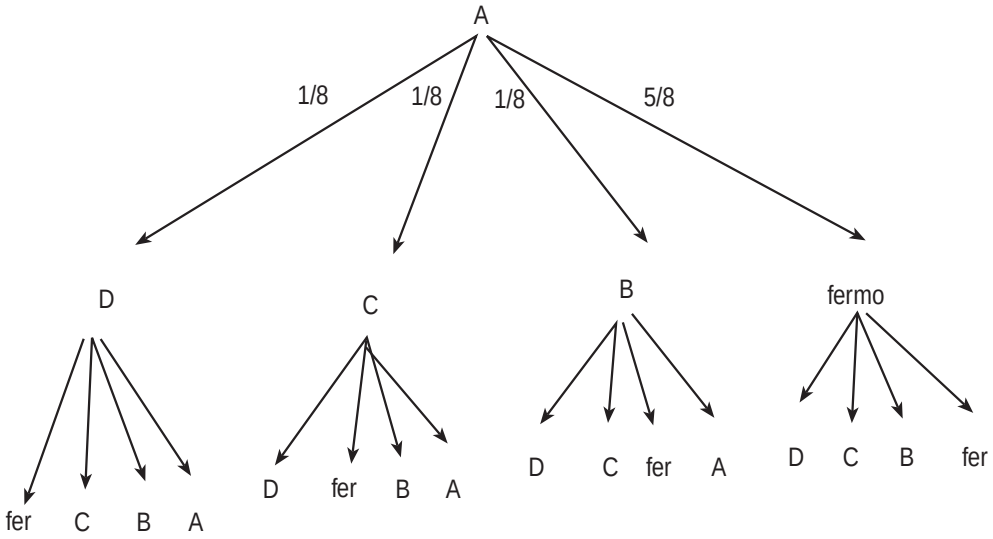
Nel 1999 ero fra i "partecipanti" del progetto Eccellenza e il problema che suscitò in me il maggiore interesse fu il seguente:

Le posizioni di un automa sono rappresentate dai quattro vertici di un quadrato. L'automa può andare da un vertice all'altro muovendosi lungo i lati o le diagonali del quadrato. Per decidere quale direzione prendere da ogni vertice viene lanciato un dado ottagonale le cui facce corrispondono a uno degli otto punti cardinali. A seguito dell'esito del lancio, se la direzione è percorribile, allora l'automa si muove, altrimenti rimane fermo.
Studiare la probabilità che l'automa si trovi nel punto B dopo 1, 2, 3, ... n lanci.
Viene indicato A come punto di partenza, ma in realtà si può osservare che è assolutamente indifferente il punto da cui si inizia. Infatti la rappresentazione della situazione dopo due lanci, fatta con un grafo ad albero risulta essere

(2) Fra i testi che riportano la trattazione dell'argomento con le relative dimostrazioni ricordo: G.C. Barozzi, *Primo Corso di Analisi Matematica*, Zanichelli 1998
(3) Un articolo noto e che offre diversi spunti didattici è quello di P. Boieri citato in bibliografia







Cambiando il punto di partenza la situazione non cambia, quindi:

• dopo un lancio la probabilità di arrivare in B è 1/8

• dopo due lanci sarà $\frac{1}{8}\cdot\frac{1}{8}+\frac{1}{8}\cdot\frac{1}{8}+\frac{1}{8}\cdot\frac{5}{8}+\frac{5}{8}\cdot\frac{1}{8}=\frac{3}{16}$

Possiamo rappresentare la variabile aleatoria relativa alla situazione iniziale: $\begin{Bmatrix} A & B & C & D \\ 1 & 0 & 0 & 0 \end{Bmatrix}$

e chiamare **v** il vettore delle probabilità iniziali: **v** = (1, 0, 0, 0)

Il grafo ad albero non è un modello che si presta ad una rappresentazione semplice e chiara della situazione nel caso di un numero di lanci superiore a due. D'altra parte in ogni passaggio la probabilità di arrivare in un certo stato è strettamente legata solo allo stato precedente. Possiamo allora rappresentare un generico passaggio da una situazione ad un'altra con una matrice T (detta matrice di transizione):

	A	B	C	D
A	5/8	1/8	1/8	1/8
B	1/8	5/8	1/8	1/8
C	1/8	1/8	5/8	1/8
D	1/8	1/8	1/8	5/8

Con *Derive* possiamo verificare che $\mathbf{v}\cdot T^2 = \left(\frac{7}{16},\frac{3}{13},\frac{3}{13},\frac{3}{13}\right)$. Il primo valore rappresenta la probabilità di essere finiti nello stato A, il secondo quello di essere finiti nello stato B,…

Infatti il prodotto fra il vettore **v** e la matrice T riproduce esattamente il calcolo sull'albero.
Possiamo ora utilizzare *Derive* per simulare un numero sempre più grande di lanci, calcolando le potenze successive di T. L'osservazione ci porta ad ipotizzare che la matrice tenda ad una matrice formata di tutti i termini uguali a 1/4.

Se gli studenti sono già abituati all'utilizzo dei grafi ad albero per risolvere problemi di probabilità, saranno portati, nella fase di analisi del problema, a utilizzarli per rappresentare la situazione. Il passaggio alla matrice di transizione risulterà conseguente e, quindi, la trattazione successiva con *Derive*. In questo caso *Derive* è un supporto di grande effi-

#1: v := [1, 0, 0, 0]

#2: $t := \begin{bmatrix} \frac{5}{8} & \frac{1}{8} & \frac{1}{8} & \frac{1}{8} \\ \frac{1}{8} & \frac{5}{8} & \frac{1}{8} & \frac{1}{8} \\ \frac{1}{8} & \frac{1}{8} & \frac{5}{8} & \frac{1}{8} \\ \frac{1}{8} & \frac{1}{8} & \frac{1}{8} & \frac{5}{8} \end{bmatrix}$

#3: v*t rappresenta la situazione dopo un lancio

#4: $\begin{bmatrix} \frac{5}{8}, & \frac{1}{8}, & \frac{1}{8}, & \frac{1}{8} \end{bmatrix}$

#5: calcoliamo ora alcune potenze di t per simulare l'andamento dopo n lanci

#6: t^2

#7: $\begin{bmatrix} \frac{7}{16} & \frac{3}{16} & \frac{3}{16} & \frac{3}{16} \\ \frac{3}{16} & \frac{7}{16} & \frac{3}{16} & \frac{3}{16} \\ \frac{3}{16} & \frac{3}{16} & \frac{7}{16} & \frac{3}{16} \\ \frac{3}{16} & \frac{3}{16} & \frac{3}{16} & \frac{7}{16} \end{bmatrix}$

#8: t^{10}

#9:
$$\begin{bmatrix} 0.2507324218 & 0.2497558593 & 0.2497558593 & 0.2497558593 \\ 0.2497558593 & 0.2507324218 & 0.2497558593 & 0.2497558593 \\ 0.2497558593 & 0.2497558593 & 0.2507324218 & 0.2497558593 \\ 0.2497558593 & 0.2497558593 & 0.2497558593 & 0.2507324218 \end{bmatrix}$$

#10: t^{50}

#11:
$$\begin{bmatrix} 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \end{bmatrix}$$

#12: t^{100}

#13:
$$\begin{bmatrix} 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \end{bmatrix}$$

#14: t^{500}

#15:
$$\begin{bmatrix} 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \end{bmatrix}$$

cacia, non solo per calcolare facilmente prodotti fra matrici, ma, soprattutto per formulare congetture e verificarne la validità.
La situazione presentata può consentire di introdurre l'algebra lineare a partire da un problema di probabilità.
Si può inoltre operare un approfondimento teorico sulle catene di Markov, osservando che la matrice in esame risulta regolare (a partire da una qualche potenza i termini sono tutti positivi) e ha un punto fisso: il vettore **w** è punto fisso per la matrice T, se **w** = **w***T. E' possibile verificare questo fatto risolvendo il sistema **w** = **w***T, dove **w** = (x, y, z, t). Si ottiene un sistema omogeneo che ha ∞^1 soluzioni date da t(1, 1, 1, 1). Quindi il vettore **w** = $\left(\frac{1}{4},\frac{1}{4},\frac{1}{4},\frac{1}{4}\right)$ risulta essere punto fisso per la matrice di transizione. Anche in questa fase di approfondimento *Derive* può essere utilizzato in modo significativo.

Nel corso del seminario conclusivo dei lavori un insegnante propose una soluzione elementare del problema che non necessitava di alcun software e che viene riportata di seguito.
L'analisi del problema evidenzia che la struttura del grafo e le probabilità assegnate ai movimenti elementari dell'automa sono perfettamente simmetriche rispetto ai quattro vertici. Si intuisce quindi che quando *n* è "molto grande" (e l'automa, per dirla in altre parole, ha dimenticato da dove è partito) la probabilità di trovarsi in B è pari alla probabilità di trovarsi in uno qualsiasi degli altri vertici ed è pertanto uguale a 1/4. Se si indica con $\Pr\left\{A \xrightarrow{n} B\right\}$ la probabilità p_n che l'automa, partendo da A, si trovi in B dopo n passi, utilizzando la probabilità condizionata si ottiene la probabilità cercata, espressa mediante la formula ricorsiva:

$$\begin{cases} p_0 = 0 \\ p_n = \frac{1}{8} + \frac{1}{2} \cdot p_{n-1} \quad n \geq 1 \end{cases}$$

L'esame dei primi termini della successione suggerisce la forma chiusa: $p_n = \frac{2^n - 1}{2^{n+2}}$
che si può dimostrare utilizzando il principio d'induzione.
L'ultima formula conferma l'intuizione iniziale: per $n \mapsto \infty$ la probabilità di trovarsi in B tende a 1/4.
La soluzione è senza dubbio più elegante e si potrebbe essere portati a concludere che, in questo caso, non vi è alcun guadagno formativo nell'utilizzo del software. Occorre però sottolineare che pochi studenti sarebbero in grado di affrontare, autonomamente, questo procedimento, e che una fase di esplorazione iniziale con *Derive* consentirebbe, anche in questo caso, di comprendere meglio la soluzione elementare che l'insegnante potrebbe presentare nella sintesi finale.

BIBLIOGRAFIA

G. Accascina, G. Margiotta, G. Olivieri (a cura di), *Problem solving e calcolatore* (materiali relativi alle attività del progetto Eccellenza dell'anno 1999), Franco Angeli, Milano 2001

G.C. Barozzi, *Primo Corso di Analisi Matematica*, Zanichelli, Bologna 1998

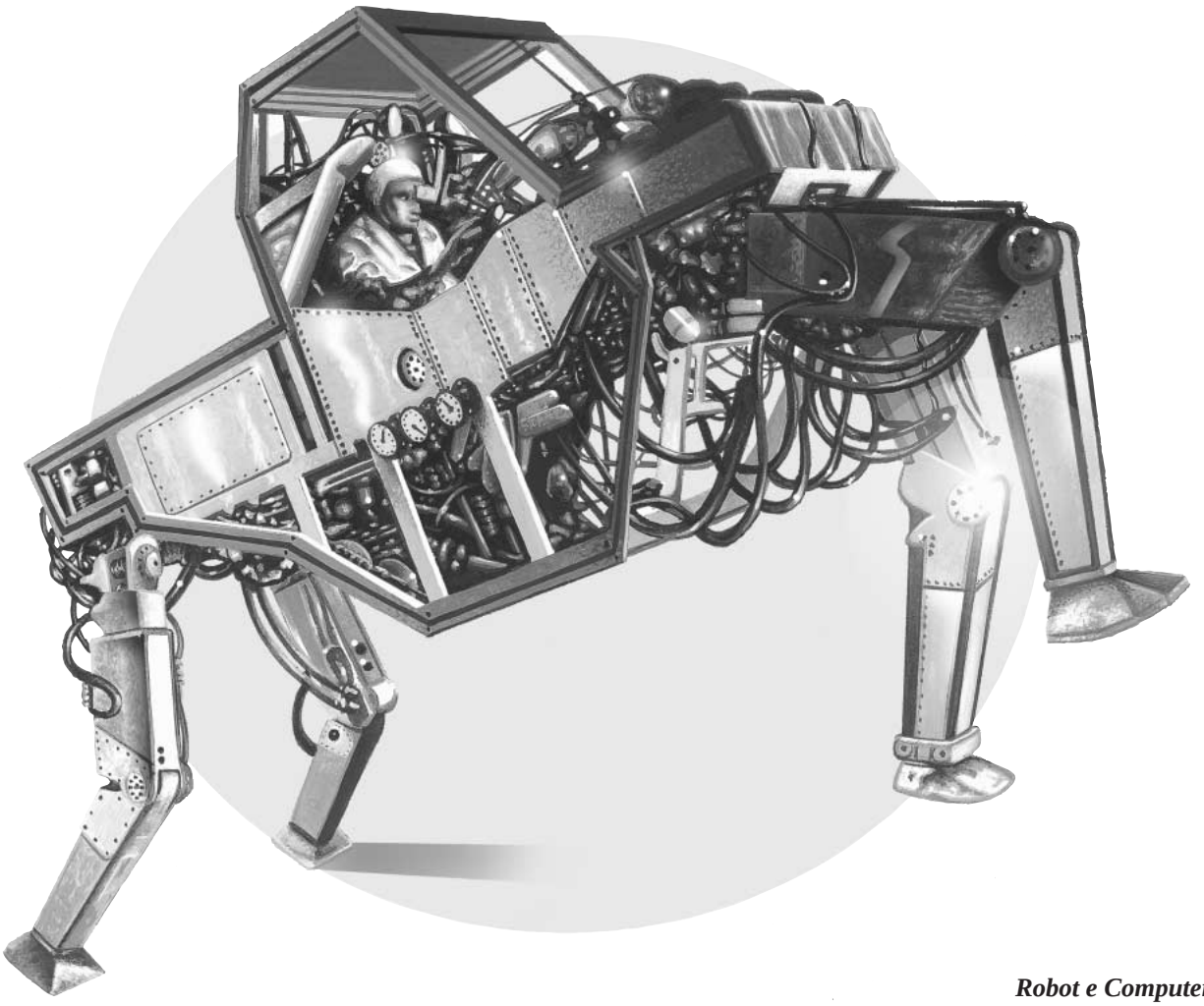
P. Boieri, *Rappresentazione dei numeri e operazioni in virgola mobile: un'applicazione del calcolatore nell'insegnamento della matematica*, Periodico di Matematiche (organo della Mathesis), n.4 , 1986

G. Margiotta (a cura di), *Matematica e software didattici* (materiali relativi alle attività del progetto Eccellenza dell'anno 1998), IRRSAE-Emilia Romagna, Bologna 1999

A. Orlandoni (a cura di), *Matematica e software didattici* (materiali relativi alle attività del progetto Eccellenza dell'anno 2000), IRRE-Emilia Romagna, Bologna 2002

V. Villani, *Cominciamo da zero*, Pitagora, Bologna 2003

AA. VV., *Matematica 2003*: Materiali per un nuovo curricolo di matematica con suggerimenti per attività e prove di verifica (scuola superiore), di prossima pubblicazione all'indirizzo: http://www.dm.unibo.it/umi/italiano/Didattica/didattica.html



Robot e Computer

L'approccio a *Derive* dei futuri docenti di Scuola Secondaria Superiore

di Giuseppe Accascina
Università "La Sapienza" Roma

Fin dal primo anno di istituzione della Scuola di Specializzazione all'Insegnamento Secondario (S.S.I.S.) del Lazio mi è stato assegnato il compito di svolgere i laboratori di introduzione all'uso di Cabri e Derive durante i quali ogni specializzando utilizza effettivamente un calcolatore.

In questi corsi di laboratorio mi sono prefisso lo scopo di far discutere gli specializzandi su quali possano essere i vantaggi e gli svantaggi dell'uso di programmi di calcolo simbolico e di geometria dinamica e quali possano essere le possibili strategie didattiche per rendere proficuo il loro uso.
La maggior parte degli specializzandi è laureata in Matematica e non ha mai fatto uso né di Derive né di Cabri.
Ho già descritto[(1)] le loro reazioni all'uso di Cabri. Espongo qui le reazioni all'uso di Derive.
I laboratori non prevedono alcun esame. Ciò favorisce la discussione e l'esplicitazione di dubbi.
Agli specializzandi chiedo solamente, oltre ovviamente ad un impegno durante i laboratori, di scrivere una pagina di commenti ad ogni seduta di laboratorio e di inviarmela prima della seduta successiva.

Mi pare che le reazioni degli specializzandi durante i laboratori e nei loro commenti scritti possano essere di qualche interesse per i docenti partecipanti al Convegno. Credo infatti che le reazioni degli specializzandi non siano poi troppo dissimili dalle reazioni dei loro futuri studenti.
Non sono ovviamente in grado di dimostrare ciò.
In ogni caso è molto probabile che, se certe ingenuità, certi errori, sono fatti da futuri docenti, laureati in matematica, a maggior ragione saranno fatti dagli studenti di Scuola Secondaria Superiore.

Nel seminario descriverò le reazioni degli specializzandi alle mie "provocazioni" sui seguenti argomenti:
1. I numeri di Fermat
2. I numeri di Mersenne
3. "Grafici di funzioni" contro "Studio di funzioni"
4. Quale tipo di equazione per le curve piane?
5. Disegnare un topolino
6. Ellissi ed ovali
7. Quale tipo di equazione per le superfici?

[(1)] **G. Accascina** *Laboratorio di introduzione all'uso di Cabri. I commenti degli specializzandi* in La Formazione degli insegnanti: approccio didattico con le nuove tecnologie Atti del I Convegno Nazionale delle Scuole di Specializzazione, Indirizzo fisico – matematico – informatico (a cura di O.Robutti, M.Mosca), Ghisetti e Corvi, Torino, 2003

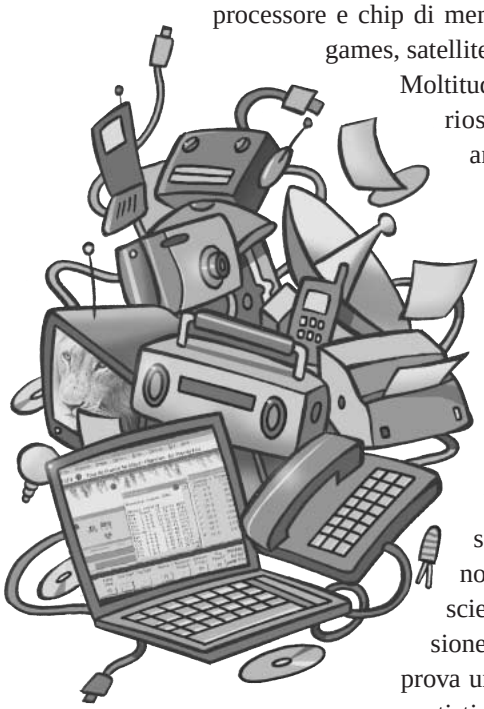
Dall'Abaco al Computer
(un poster per la didattica)

di Anna Maria Arpinati
IRRE Emila Romagna

Sull'onda del discreto successo riscosso nel 2003 dalla pubblicazione e distribuzione del poster *Storia della Matematica* (cm200xcm50*)*, nel 2004 IRRE Emilia Romagna ha predisposto la redazione di un nuovo cartellone, delle stesse dimensioni, dedicato ad una breve e sintetica storia delle idee e delle invenzioni che hanno portato alla messa a punto dei moderni elaboratori.
Titolo del nuovo poster, dedicato prevalentemente ai ragazzi della scuola secondaria: *Dall'Abaco al Computer*, distribuito in occasione del Convegno Nazionale *L'insegnamento dell'algebra (e non solo) nell'era dei computer* (Bologna, 15 marzo 2004).
Dovendo sintetizzare nello spazio di un poster una storia così complessa, che non è solo un'avventura "elettronica" come molti ragazzi credono, il gruppo di coordinamento che ha messo a punto il progetto poteva scegliere diverse strade: ha privilegiato quella che permetteva di conseguire i seguenti obiettivi:
di mostrare come alle spalle dell'invenzione del moderno elaboratore stiano tantissime idee che hanno impiegato duemila anni a collocarsi in un sistema;
di fare capire come il prodotto finale è una specie di puzzle, che ha messo insieme risultati prodotti in ambiti disciplinari diversi, in tempi e luoghi differenti; spesso in campi "di altissima specializzazione", ma talvolta dovuti anche al sogno, apparentemente visionario, di un singolo;
di fare salire alla superficie il ruolo sotterraneo, ma decisivo dell'evoluzione delle "idee" e degli strumenti matematici e del supporto cognitivo diretto e indiretto da essi fornito;
di rendere evidente che a cambiare le società umane, le loro abitudini e capacità sono gli "oggetti", sempre strettamente intrecciati però con i pensieri che li hanno creati.

E se ai giovani la storia non piace...

Chi legge con cura il poster *Dall'abaco al computer* e si annota mentalmente le tante (ma per motivi evidenti di spazio non tutte) invenzioni lì descritte, non può non sentirsi spinto a riflettere o a fare ricerche più approfondite su: lampadina, telefono, radio, macchina da scrivere, pantelegrafo (chi sa di cosa si tratta?), telegrafo, valvola, diodo e triodo, macchina fotografica, registratore magnetico, robot, circuito stampato e circuito integrato, laser, microprocessore e chip di memoria, fotocopiatrice, stampante, hard disk, transistor, consolle e video games, satellite, modem, floppy disk, cd, dvd, e tanti altri o altre "diavolerie".



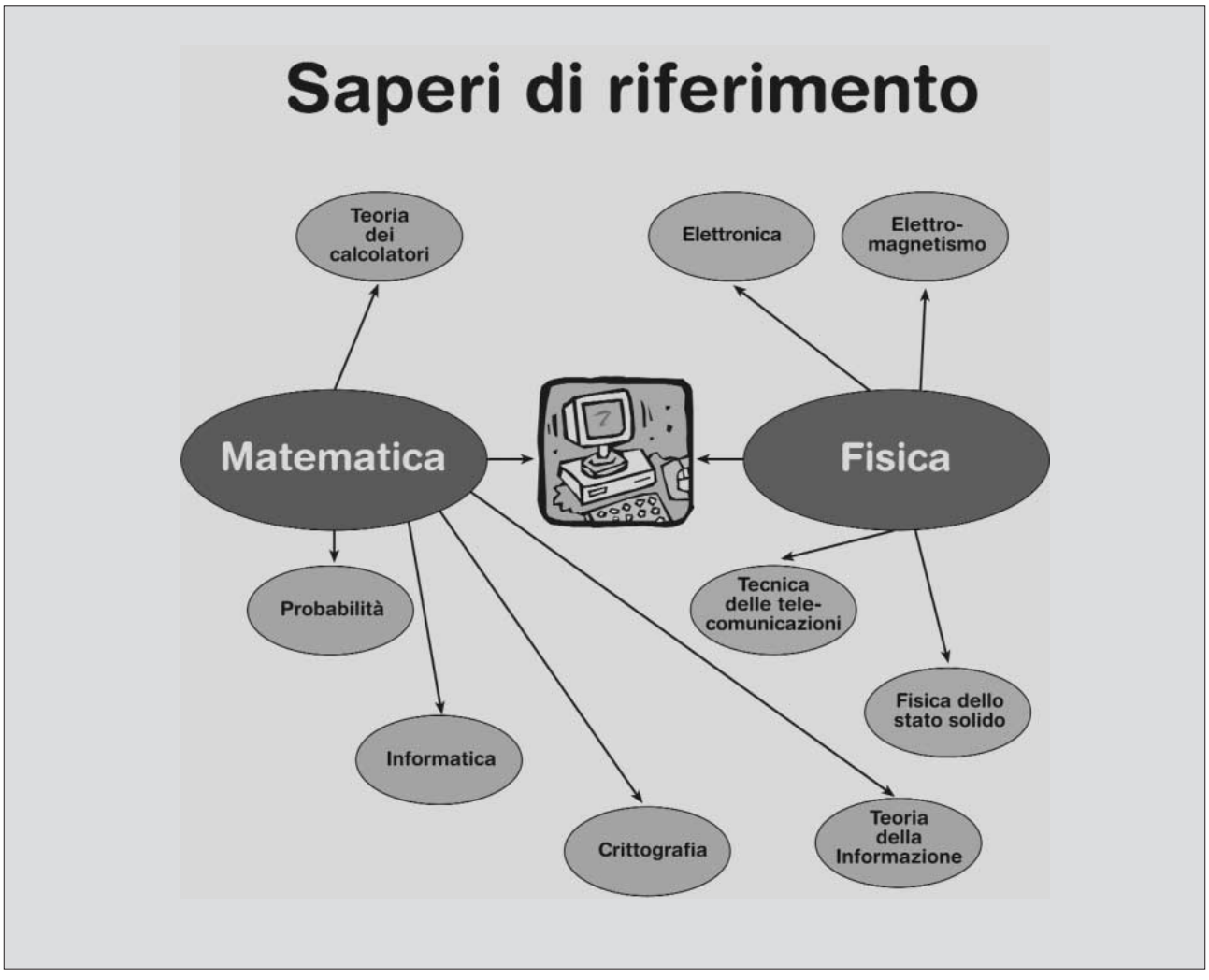
Moltitudine di oggetti, antichi e moderni, arcaici e attualissimi, risaputi o misteriosi. Strumenti della nostra vita quotidiana o reperti museografici, tutti ancora in qualche modo "vivi" nel ventre ronzante della macchina con cui noi tutti ormai quotidianamente dialoghiamo.
Si dice che ai giovani la storia non interessa, che essi vivono integralmente nel presente. Magari è proprio educativa una piccola provocazione: proporre un oggetto mitico del loro presente nella sua dimensione storica.
Non solo dietro alle grandi imprese sportive o musicali (altro ipertrofico "presente" dei nostri meno che ventenni), ma anche nel backstage della tecnologia dimorano idee, personaggi, scoperte e invenzioni.
Forse non è male indurre i nostri ragazzi e le nostre ragazze, a riflettere sulla quantità di lavoro presente in tutto ciò, sulle storie umane che stanno alla base dei successi conseguiti e dei fallimenti patiti; una scoperta scientifica o la stesura di una teoria matematica spesso comporta una tensione emotiva ed un coinvolgimento intellettuale non dissimile da quello che prova un poeta o un pittore, quando vuole realizzare e spiegare ad altri la sua opera artistica (*bibl.1*). Si pensi solo alla diatriba fra Meucci e Bell relativa all'inven-

zione del telefono. Forse tutto è utile al progresso umano: se infatti, dopo infiniti dibattiti è consolante sapere che l'invenzione è stata attribuita ufficialmente all'italiano Meucci (non stimato nel suo paese e sprovvisto di finanziamenti), è una fortuna che l'americano Bell abbia potuto fondare negli USA (più attenti ai problemi della ricerca scientifica) i laboratori Bell, che sono stati una fucina continua di idee e di nuove invenzioni per tutto il ventesimo secolo.

E se valesse la pena studiare le discipline ...

Il poster in questione può fornire un'altra piccola provocazione, per studenti annoiati, pedagogisti rampanti o ministri arrendevoli; con esso si possono forse "comunicare" efficacemente alcune *discipline*. In certe scuole i programmi di studio sono introdotti da una lunga sfilza dei vantaggi pratici che derivano dallo studio di… e della… Dalla cascata delle professioni per cui è vantaggioso essersi formato in … e per … Insomma, pare che, come succedeva nei racconti della nostra infanzia, per far ingoiare le amare discipline sia necessario metterci un po' di "zucchero". Molto più semplicemente le discipline sono frutto di processi storici, sono nate quasi autonomamente, quando ci si accorgeva che per rispondere a certi "perché" della mente umana, era necessario approfondire solo determinati aspetti di un problema, sperimentare altri percorsi di indagine. Chi meglio degli inventori e pensatori presentati nel poster *Dall'abaco al computer* può mostrare come le idee nuove nascano a volte come frutto di una rigorosa ricerca, a volte come miracolo di inventiva? Allo stesso modo è evidente che alcune scoperte sono nate in campi ben recintati e altre sono il risultato di geniali "salti di steccato".

Non è quindi difficile prendere spunti dal cartellone per far vedere ai nostri studenti come le discipline più antiche che sono servite allo sviluppo delle idee che hanno portato ai moderni elaboratori, siano la *matematica* e la *fisica*, con il supporto poi di altri saperi che nascevano dai "salti di steccato" sopra citati (vedi figura).



E se una certa professione avesse più visibilità...

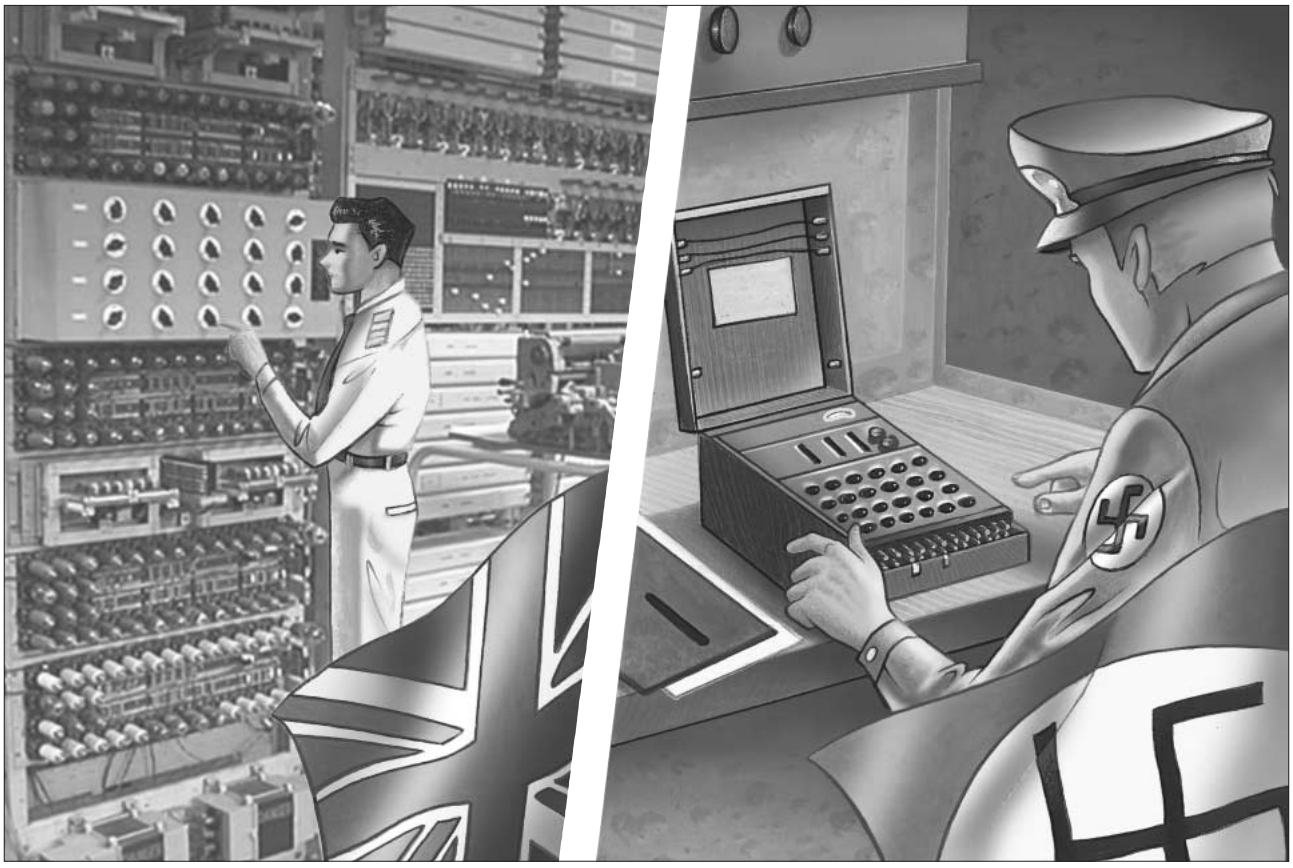
Esistono solo insegnanti di matematica o anche *matematici*? Temo che la totalità dei nostri studenti e studentesse non saprebbe rispondere. Ebbene, il poster *Dall'abaco al computer*, appeso alle pareti dell'aula o del laboratorio, può servire proprio come documento del fatto che, almeno fino a ieri, sono veramente esistiti dei *matematici* e che essi non sono personaggi di fantascienza. Dal cartellone provengono numerosi suggerimenti per affrontare nell'attualità problemi

matematici, *fingendo* (un po' come nei giochi di ruolo) di non conoscerne la soluzione.
Ecco alcuni possibili avvii di ricerca:
- Quali matematici, e per che cosa, furono coinvolti nel ventesimo secolo nell'avventura del calcolatore? (Per dare una risposta vedere ad esempio *bibl. 2*)
- Nel corso dei secoli quante idee "matematiche" furono realizzate, senza nessuna premonizione del loro impiego futuro?
- Dove avevano la possibilità marchingegni matematici, come la *pascalina* o la *macchina di Leibniz,* di risolvere problemi?
- Come si è passati dalla occupazione di trattare numeri a quella di trattare informazioni?
- Chi è stato in grado di trasferire alla macchina, sempre sfruttando la matematica, la capacità umana di *prevedere* e *pianificare* ?
- Quando le tecnologie del tempo furono un limite alle intuizioni matematiche apparentemente utopiche?
- Che ruolo hanno avuto e hanno tuttora i linguaggi di programmazione?
- Quale ruolo ha avuto la possibilità di una interfaccia grafica?

E sicuramente moltissime altri progetti più brillanti potranno scaturire dai docenti e dai loro studenti, lanciati alla ricerca della "professione" di matematico.

E se facessimo qualche concessione interdisciplinare...

Se si osserva poi il poster *Dall'abaco al computer* con l'occhio di chi si interessa della scienza e della tecnologia come fenomeni sociali, ci si trova in un ambiente fertile di informazioni e di spunti per ulteriori approfondimenti. Il computer è strettamente connesso, ad esempio, con le previsioni meteorologiche ed il calcolo delle maree, con l'economia e con tutti i problemi connessi con la riservatezza dei codici, con la sanità e l'informazione. Ancora, a chi è venuto in mente che c'è stato e c'è ancora uno stretto legame tra matematica e scienza della guerra (*bibl.3*)? Non è solo storia recente, vedi guerra del Golfo o invasione dell'Iraq, programmate e condotte con modelli matematici. Nel 1945 si assistette (si fa



per dire) al duello segreto tra la macchina codificatrice *Enigma* dell'esercito tedesco ed il supercomputer *Colossus* costruito dagli inglesi. La decifrazione dei messaggi in codice tedeschi permise alla Gran Bretagna di resistere per mesi, nonostante l'inferiorità militare. Fu una vicenda non solo militare, ma anche matematica e biografica (si potrebbe su questo tema proporre una tesina per l'esame di stato un po' fuori dagli schemi normali).
Dalla vicenda di *Enigma* e *Colossus*, ci si può riallacciare ad una delle prime battute del film *A beautiful mind* ,vincitore di quattro premi Oscar nel 2002. La battuta, che più o meno suona così (*si potrebbe affermare che la Seconda Guerra*

Mondiale l'hanno vinta i matematici) serve per ricordare l'enorme regalo che l'Europa ha fatto (anni 1930 – 1945) agli Stati Uniti d'America, favorendo una migrazione di cervelli matematici e fisici dal vecchio al nuovo continente, a causa delle leggi razziali presenti in Germania e in Italia.
Episodio certamente non estraneo al fatto che oggi, all'inizio del terzo millennio, agli Stati Uniti d'America va riconosciuta la prima posizione a livello mondiale nel campo della ricerca. Se il vecchio continente avesse fatto in passato una politica diversa e, soprattutto, la facesse diversa anche ai giorni nostri, le cose potrebbero forse stare in modo differente.
E' buona cosa discutere con i ragazzi anche di questi problemi? Pensiamo di sì.

E se facessimo un po' di orientamento...

In tempi di disaffezione verso gli studi scientifici (le iscrizioni sono in calo), ma di grandi successi tecnologici (medicina, farmacologia, telecomunicazioni, spazio, elettronica, nanotecnologie) è ancora difficile nel nostro paese presentare come appetibile la carriera scientifica.
Quella che si chiama *la ricerca di base* (che nessuno sa essere *dietro* alle tecnologie che ingombrano o semplificano la nostra esistenza quotidiana) in Italia non trova finanziamenti, non ha "posti" da offrire, né ribalte su cui proporsi.
Il poster aiuterà forse a non perdersi d'animo quei molti giovani, dotati di viva intelligenza, che non hanno come unica meta del loro futuro, quella di divenire "campione di calcio", o "velina" di una trasmissione televisiva, o "grande fratello".
Chissà mai che qualche studente o studentessa, alle soglie dell'università non venga attratto dai nomi, dalle scoperte e dalle invenzioni del poster, e magari non sogni qualche avveniristica innovazione. Potrebbe scegliere allora di iscriversi a corsi che permettono il raggiungimento di *lauree forti* (ingegneria, chimica, fisica, matematica, …). Il percorso di studio sarà più faticoso, richiederà capacità, preparazione, tenacia; presenterà continue scommesse da vincere.
Nel dicembre scorso un quotidiano portava la notizia che all'università di Malaga un giovane affetto da sindrome di Down stava per conseguire una laurea in psicopedagogia (*bibl.4*), laurea appartenente al cosiddetto filone delle *lauree deboli*.
Forse è proprio quando l'ostacolo è più difficile da superare(*lauree forti*), che ci sentiamo motivati a farlo? Forse, in definitiva, è questo il messaggio più importante del poster *Dall'abaco al computer* (una storia vera): il vero successo è in fondo al labirinto.

Bibliografia

1. G. Zanarini, *Ludwig Boltzmann, una passione scientifica*, dalla collana "Tessere", editore CUEN, 1996
2. Autori Vari, *Grandi matematici del Novecento*, lettera matematica pristem , n°50-51, Centro Eleusi-Università Bocconi, Springer Editore, 2004
3. U. Bottazzini, *In guerra con i numeri*, Il Sole-24Ore, 25.01.2004, suppl. "Domenica", pag. 34
4. M.Vignolo, *Ho la sindrome di Down e mi laureo*, Corriere della Sera, 20.12.2003, pag. 19

World Wide Web

Per avere informazioni sul poster, contattare Marisa Cresci di IRRE Emilia Romagna (cresci@irreer.it), *oppure Media Direct di Bassano del Grappa* (info@campustore.it)

Algebra e Computer

Non è ammissibile
che studiosi e scienziati,
anziché elaborare
e confrontare nuove teorie,
perdano le proprie ore
come schiavi
nelle fatiche del calcolo,
che potrebbe
essere affidato
a chiunque,
se si potessero
usare delle macchine...

Gottfried Wilhelm
von Leibniz

I.R.R.E. Emilia Romagna - Sezione Scuola Media

Supplemento al n. 6 Novembre-Dicembre 2003, di INNOVAZIONE EDUCATIVA bollettino bimestrale dell'Istituto Regionale di Ricerca Educativa dell'Emilia-Romagna. Registrazione Trib. Bo n. 4845 del 24 - 10 - 1980. Direttore resp. Franco Frabboni, Direttore edit. Arnaldo Luisi