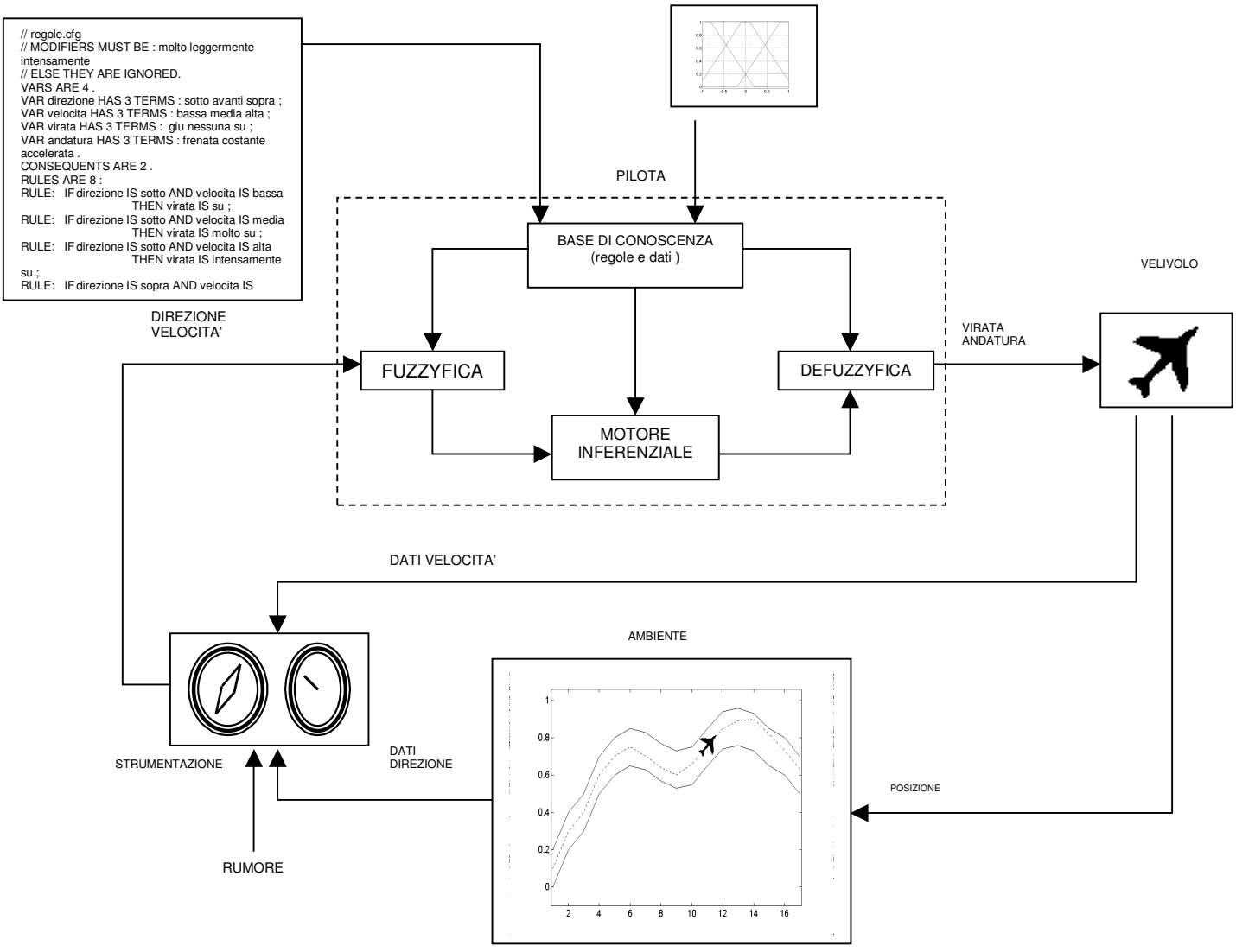
PILOTA AUTOMATICO
DI UN JET ANTIRADAR
BASATO SU CONTROLLO FUZZY

Mario Cimino e Giovanni D'Alessandro

Pisa, 2001



Indice

1. ANALISI DEL PROBLEMA E OBIETTIVI DI PROGETTO

1.1 Il dominio del sistema

Vogliamo implementare un motore inferenziale generico, funzionante con degli insiemi fuzzzy di qualsiasi forma e dimensione, dotato anche di alcuni modificatori.

Opereremo quindi con insiemi fuzzy discreti, poiche' nel caso continuo occorre specializzare i fuzzyset, se si vogliono ottenere formule algebriche per alcune grandezze che coinvolgono infiniti punti: es. gli estremi superiori, gli integrali.

Vogliamo poter definire un numero qualsiasi di regole MISO, con qualsiasi numero di antecedenti, con una sintassi del tipo "IF <variabile linguistica 1> IS <valore linguistco 1> AND … <variabile linguistica N > IS <valore linguistico N> THEN <variabile linguistica> IS <valore linguistco>", e per ogni valore linguistico poter specificare a piacere la regola semantica.

Infine, vogliamo adoperare il controllore fuzzy come pilota automatico di un aereo che vola a bassa quota per non essere rilevato dai radar, per avere una immediata valutazione delle sue prestazioni; introdurremo anche del rumore sulle misurazioni.

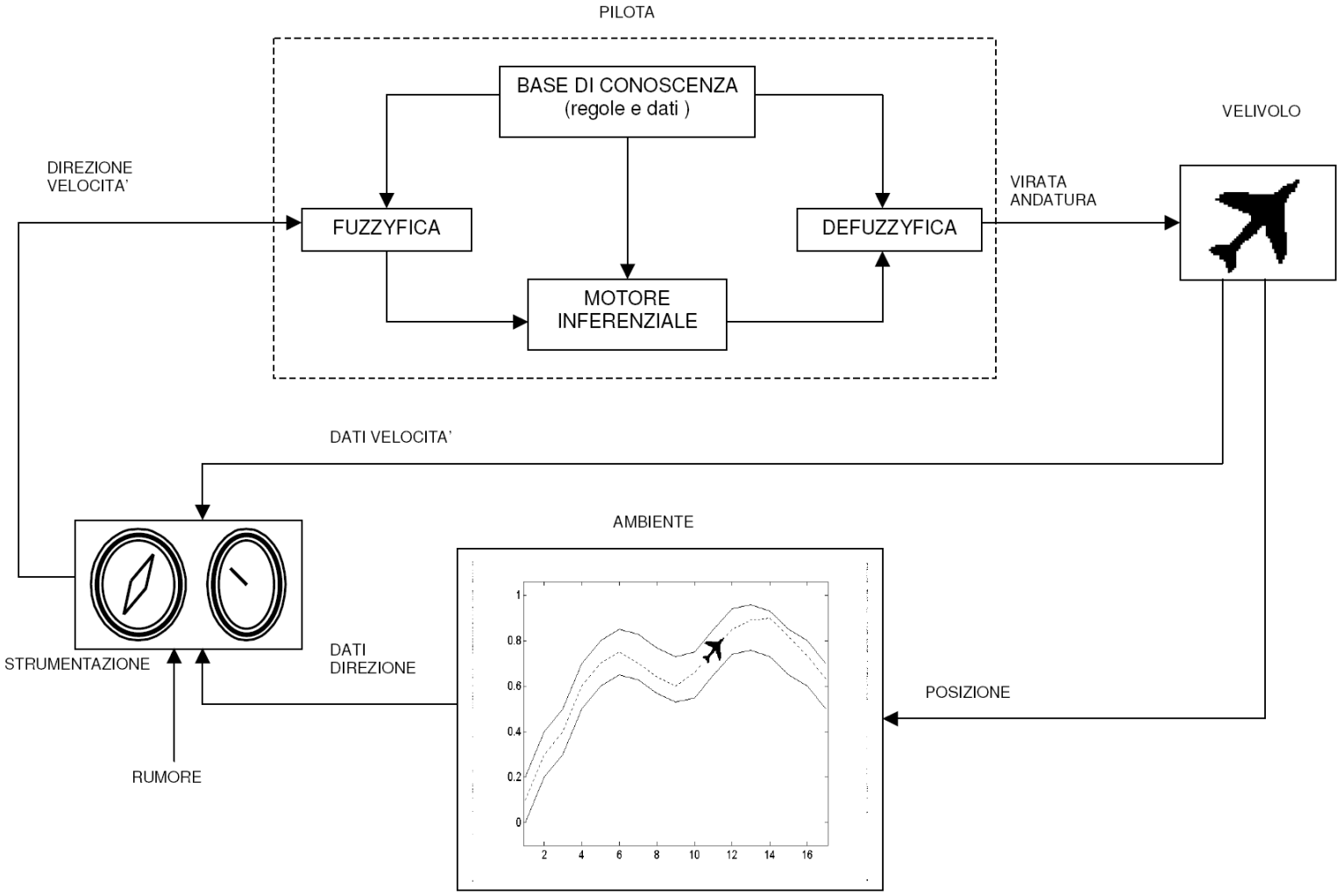
1.2 Il processo di sviluppo

Vogliamo un codice c++ orientato agli oggetti, in cui ogni entita' a se' sia raggruppata in una classe, incapsulata in un modulo, e realizzata in modo da garantire una riusabilita' del codice, una buona leggibilita' e modificabilita'. Daremo quindi importanza al processo di sviluppo piuttosto che al prodotto, laddove i criteri suddetti potessero produrre un lieve peggioramento della velocita' di esecuzione.

A tale proposito cercheremo, se possibile, di usare le funzioni inline, tenendo presente che "Scrivere la definizione di una funzione membro nella dichiarazione di una classe rende i programmi meno leggibili e meno modificabili." *(pag. 215,* [Domenici-Frosini, 1996] *);* oppure il passaggio dei parametri di ingresso per riferimento (mediante attributo const), per una chiamata di funzione piu' snella.

2. PROGETTAZIONE

SCHEMA A BLOCCHI DEL SISTEMA COMPLESSIVO



2.1 Modello dell'ambiente in cui opera il controllore

Lo schema a blocchi del sistema complessivo e' mostrato a pagina precedente.

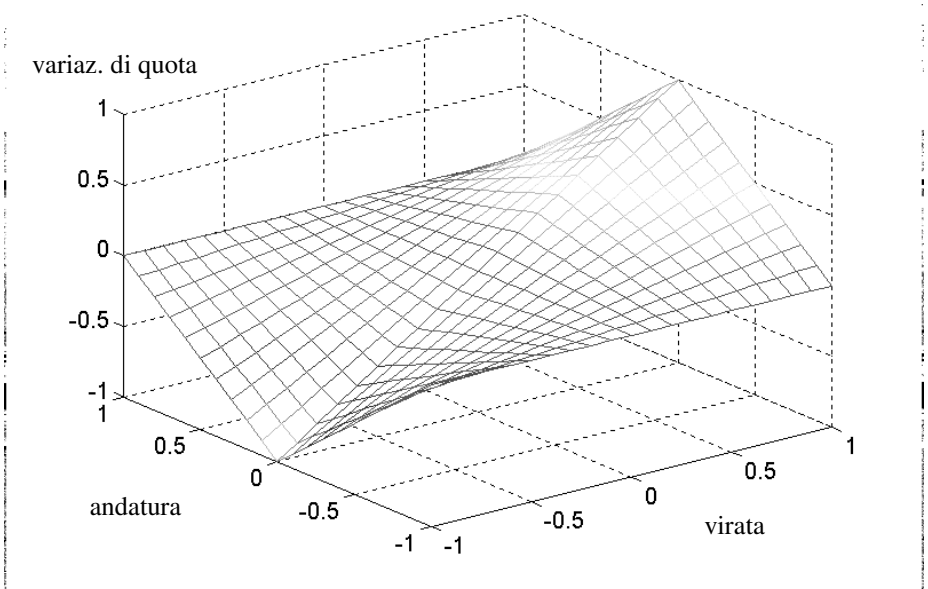
Il controllore fuzzy (pilota automatico) riceve dalla strumentazione di bordo la direzione e la velocita' dell'aereo; quindi effettua sul velivolo delle correzioni su andatura e virata.

Andatura e virata influenzeranno la posizione del jet al prossimo passo. Se la virata e' nulla, il jet prosegue lungo la direzione della traiettoria precedente, arrivando ad un punto, che chiameremo inerziale.

Piu' la virata e' positiva (negativa), piu' l'aereo effettuera' un cambiamento della propria quota in alto (basso), rispetto al punto inerziale.

Ma piu' la andatura e' grande in modulo, e meno la virata avra' effetto, perche' un aereo che accelera o decelera non riesce contemporaneamente a virare molto, per motivi legati all'inerzia.

Fig. effetto (normalizzato) della virata e della andatura sulla quota del jet



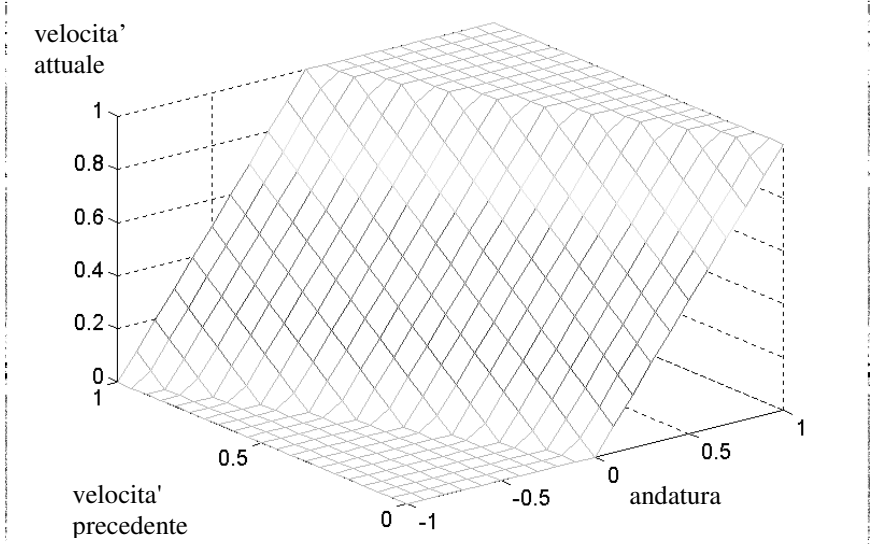
Il velivolo e l'ambiente danno successivamente alla strumentazione i dati per il calcolo della nuova velocita' e nuova direzione indicate, con del rumore additivo.

La nuova velocita' idicata dal sensore e' pari alla precedente piu' l'andatura (variazione di velocita'), fornita dal controllore fuzzy:

velocita' = velocita' + andatura + rumore

il valore risultante viene eventualmente riportato tra –1 ed 1.

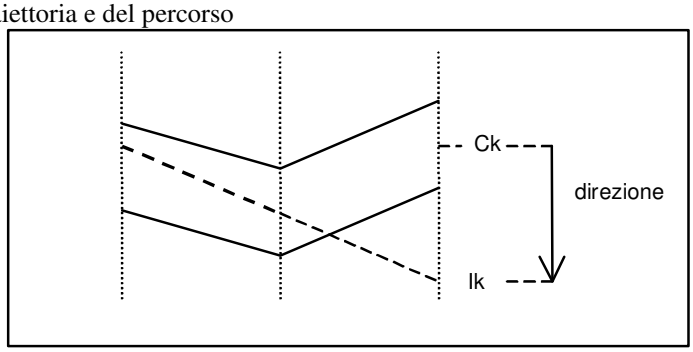
Fig. variazione della velocita' in funzione dell'andatura (senza rumore)



La nuova direzione indicata dal sensore viene intesa come la differenza tra il punto inerziale (Ik) ed il punto centrale della fascia di navigazione (Ck), normalizzando sulla larghezza della fascia.

Quindi se direzione e' maggiore 1 il jet sta puntando in direzione fuori fascia in alto (verrebbe scoperto dal radar) ; se e' minore di –1 sta puntando al suolo (si schianterebbe). Altrimenti si avvicinera' a queti estremi quanto piu' e' prossima in modulo ad uno. Se la direzione e' zero, il jet si manterra' esattamente sul centro della fascia (ottimo). Il motore inferenziale provvede quindi a prendere le opportune azioni in base a questi due indicatori. Osservando il file di regole nell'esempio, si puo' notare che esse tendono a far rimanere il jet sulla linea mediana della fascia. Se il jet dovesse uscire 'fuori', a causa del rumore introdotto, o di una eccessiva spigolosita' del percorso, verra' bloccato e riposizionato al centro fascia con velocita' nulla e direzione centrale, per poi proseguire.

Fig. andamento della direzione indicata dal sensore, in funzione della traiettoria e del percorso



2.1 Identificazione degli oggetti

La classe FuzzySet implementa in maniera naturale le proprieta' e le operazioni degli insiemi fuzzy, e contiene la parte principale del motore inferenziale, piu' le interfacce di fuzzificazione e defuzzificazione; la classe Ruleset invece realizza la aggregazione di regole fuzzy, e contiene la base di conoscenza, cioe' le regole ed i dati, gia' tradotti in termini di insiemi fuzzy; la classe VarLing raggruppa le proprieta' di una variabile linguistica, principalmente la regola semantica, associando ai vari termini linguistici i relativi fuzzyset.

Infine, la classe Percorso realizza il modello del percorso e del jet.

A differenza delle classi Ruleset e VarLing, che sono generiche, e permettono rispettivamente di definire un numero qualsiasi di antecedenti e di conseguenti nelle regole, o un numero qualsiasi di variabili linguistiche e di termini linguistici, questa classe elabora solo quattro variabili di stato, cioe' i primi due antecedenti ed i primi due conseguenti delle regole, avendo uno scopo principalmente dimostrativo.

2.2 Identificazione dei servizi

Per la classe FuzzySet abbiamo realizzato tre modificatori (dilata, intensifica e concentra); poi delle funzioni di I/O da file che permettono di visualizzare con Matlab 4 la funzione di appartenenza e di definirla con una semplice descrizione testuale; la funzione di fuzzificazione e' triangolare con un supporto configurabile; la defuzzificazione e' basata sul centroide; la ricerca di un elemento e' realizzata con algoritmo binario; infine, abbiamo la composizione, la congiunzione, e l'implicazione (di Mamdani) tra insiemi fuzzy.

Fig. dilata() intensifica() e concentra() di un insieme fuzzy

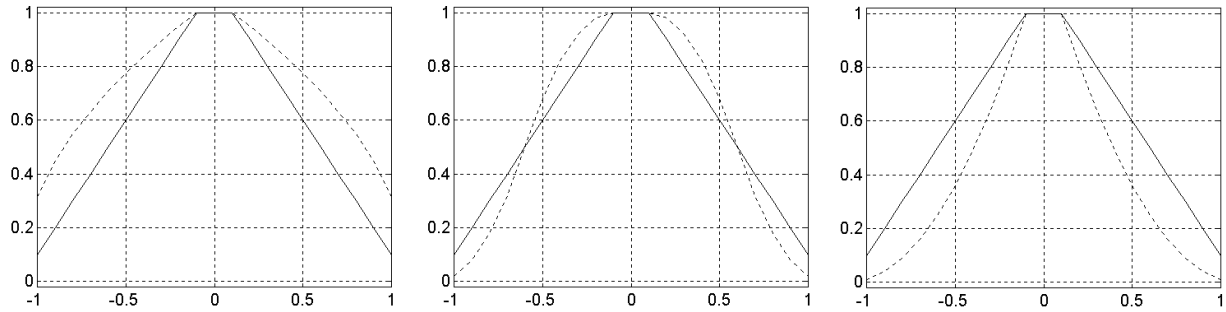
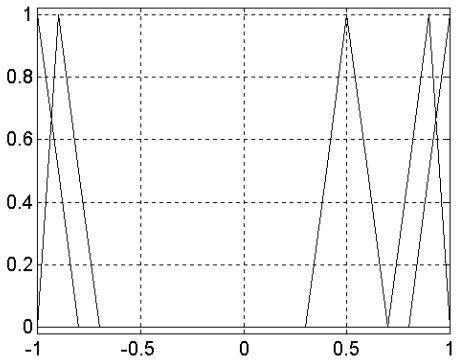


Fig. descrizione di un insieme fuzzy: file ‘andcosta.m’

```
% andatura costante ( visualizzare con Matlab )

dimU = 21;
elem = [ -1 -.9 -.8 -.7 -.6 -.5 -.4 -.3 -.2 -.1   0  .1  .2  .3  .4  .5  .6  .7  .8  .9   1 ];
grap = [ .1  .2  .3  .4  .5  .6  .7  .8  .9   1   1   1  .9  .8  .7  .6  .5  .4  .3  .2  .1 ];
clf;
plot(elem,grap,'k');
axis([min(elem) max(elem) -.02 1.02]);
grid on;
```

Fig.- Esempi di fuzzificazione



L’aggregazione e’ realizzata nella classe Ruleset dalla funzione “inferenzaFITA”; nella stessa classe c’e’ anche una funzione di interfaccia I/O con il file di regole, e con le variabili linguistiche (“caricaRuleSet”).

La classe VarLing contiene un servizio per i modificatori (“hedge”), uno di I/O con file di regole (“caricaVarLing”) ed uno per la regola semantica (“mean”).

La classe Percorso, contiene la traettoria da seguire, i sensori del velivolo (velocita’ e direzione) gli attuatori (virata, intesa come beccheggio, e andatura intesa come variazione di velocita’ impressa) ad ogni step. Abbiamo un servizio per sapere se il velivolo e’ arrivato a destinazione (“arrivo”), uno che lo riposiziona sul centro del tracciato (“posiziona”) ed in direzione parallela al profilo del percorso, se al passo precedente l’aereo e’ uscito fuori dalla fascia (“!dentropercorso”), funzioni di I/O da file (“stampa_su_file”,”leggi_da_file”) funzioni per la lettura dei sensori (“leggiSens”) dall’ambiente e la restituzione degli attuatori sull’ambiente (“modifAtt”); infine, un servizio che esegue il passo (“dostep”) aggiornando i sensori.

Fig. descrizione del percorso: file ‘percorso.m’

```
% percorso.m  ( visualizzare con Matlab )
height = 0.2;
maxstep = 17;
step = [1:1:maxstep];
down = [ 0 0.2 0.3 0.5 0.6 0.65 0.63 0.57 0.53 0.55 0.65 0.74 0.76 0.73 0.65 0.6 0.5 ];
top = down + height;
traett = [ 0.1 0.3 0.4 0.6 0.700868 0.750868 0.702436 0.642436 0.601785 0.660265 0.760265 0.850265 0.89245
0.898076 0.818076 0.732231 0.632757 ];
clf;
plot(step,down,'k',step,top,'k',step,traett,':k');
axis([min(step)-.1 max(step)+.1  -0.1  max(top)+.1]);
```

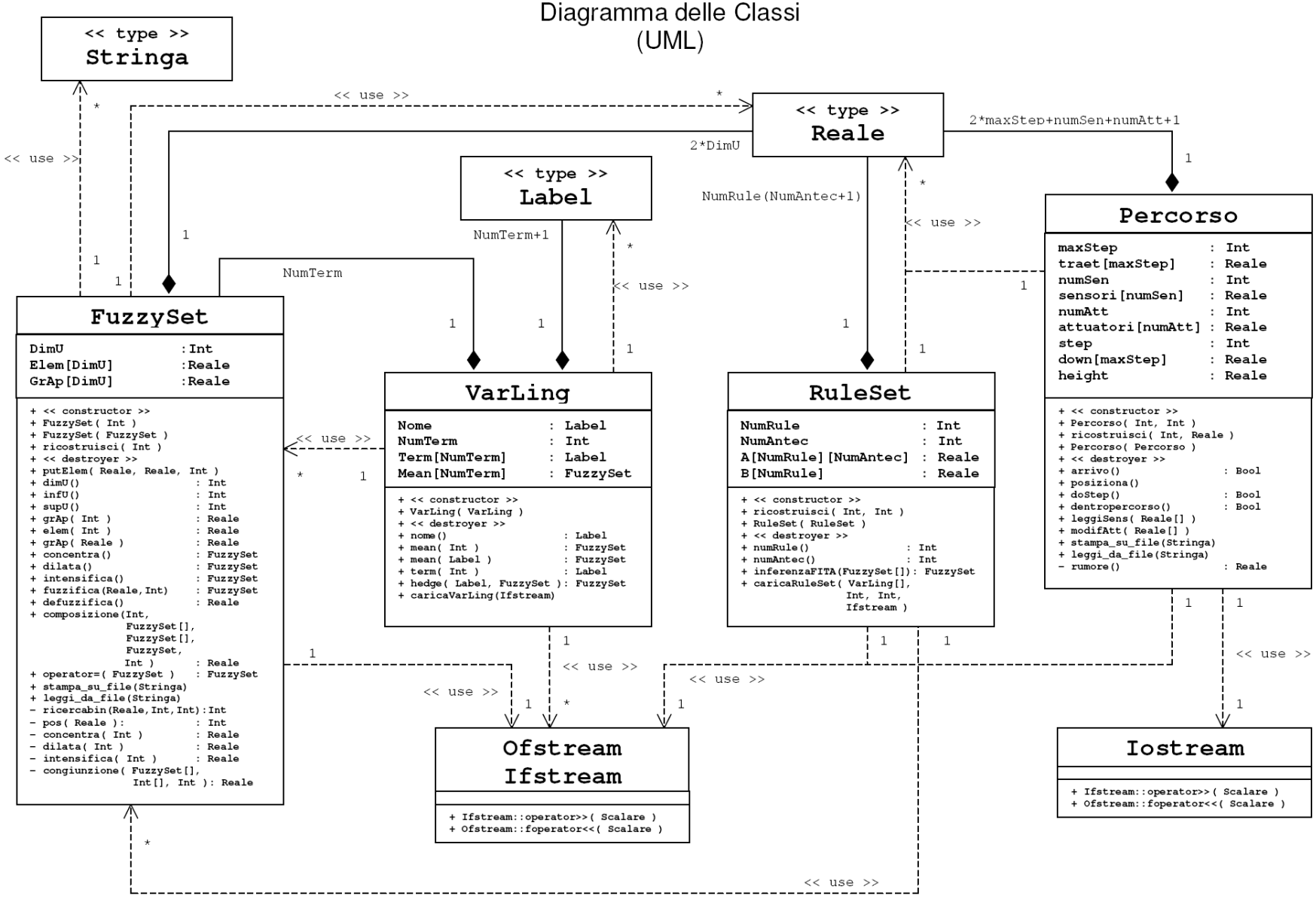
Fig. alcune righe della base di conoscenza: file ‘regole.cfg’

```
// MODIFIERS MUST BE : molto leggermente intensamente ELSE THEY ARE IGNORED.
VARS ARE 4 .
VAR direzione HAS 3 TERMS : sotto avanti sopra ;

...

VAR andatura HAS 3 TERMS : frenata costante accelerata .
CONSEQUENTS ARE 2 .
RULES ARE 8 :
RULE:   IF direzione IS sotto AND velocita IS bassa THEN virata IS su ;
RULE:   IF direzione IS sotto AND velocita IS media THEN virata IS molto su ;
RULE:   IF direzione IS sotto AND velocita IS alta  THEN virata IS intensamente su ;
RULE:   IF direzione IS sopra AND velocita IS bassa THEN virata IS giu ;
RULE:   IF direzione IS sopra AND velocita IS media THEN virata IS molto giu ;
...
```

I nomi usati dovrebbero essere abbastanza esplicativi, e la struttura delle classi si puo’ vedere nel diagramma della classi a pagina precedente, scritto in Unified Modeling Language (UML), la notazione usata dalle principali aziende di software mondiali.

E’ inevitabile che sin da adesso i servizi offerti dalle classi contengano dei dettagli implementativi, poiche’ non abbiamo rappresentato le varie fasi intermedie di sviluppo. Di queste funzioni parleremo nel prossimo capitolo.

Diagramma delle Classi
(UML)



3. SVILUPPO DEL CODICE IN C++

3.1 Il problema degli array di oggetti classe

Proseguiamo descrivendo le funzioni membro legate all' implementazione.

Il C++ non permette l'uso di costruttori diversi da quello di default per array di oggetti classe. Se ad esempio si scrive " *miooggetto miaclasse[10]; ",* verra' eseguito il costruttore standard miaclasse(); non e' permesso *"miooggetto miaclasse(n,dim)[10];".*

La soluzione da noi proposta consiste nel dichiarare l'array con il costruttore standard e di ricostruire subito dopo l'oggetto con una funzione "ricostruisci".

3.2 I costruttori di copia

Poiche' la parte dati di tutti gli oggetti contiene dei puntatori a memoria libera, e' stato necessario ridefinire i costruttori di copia.

3.3 Il problema dei for infiniti: i metaindici

Nell'algoritmo di composizione, e' necessario ripetere delle elaborazioni per tutte le ennuple possibili di variabili base. Cioe' occore realizzare la funzione seguente:

$$z = \sup f(x_1, x_2, \ldots, x_M)$$

algoritmicamente parlando si avrebbero tanti for, in numero variabile:

```
z = 0;
for ( int i1 = 0; i1 < i1max, i1 ++ )
  for ( int i2 = 0; i2 < i2max, i2 ++ )
      …
          for ( int iM = 0; iM < iMmax, iM ++ )
              z = max ( z, f(xi1, x i2,…, x iM )
```

la soluzione da noi proposta e' basata sul concetto di 'metaindice' ossia un indice di indici, $i = [i_1, i_2, \ldots, i_M]$; qundi cui si ha:

```
i = 0; z = 0;
do z = max ( z, f(xi1, x i2,…, x iM )
while scegli(i);
```

la “scegli” realizza la scelta di tutte le possibili ennuple di indici; cio’ viene realizzato considerando i_1, i_2, …, i_M come un ‘numero’, per cui ogni cifra i_k viene incrementata solo se non e’ ad i_{kmax}, e tutte le cifre a destra sono i_{k+1max} i_{k+2max} … i_{Mmax} e successivamente ponendo a zero tutte queste cifre.

Quindi

$[0\ 0\ \dots\ 0\ 0]$ $[0\ 0\ \dots\ 0\ 1]$ … $[0\ 0\ \dots\ 0\ i_{Mmax}]$

$[0\ 0\ \dots\ 1\ 0]$ $[0\ 0\ \dots\ 1\ 1]$ … $[0\ 0\ \dots\ 1\ i_{Mmax}]$

$[0\ 0\ \dots\ 2\ 0]$ …

…

$[0\ 0\ \dots\ i_{M-1max}\ i_{Mmax}]$

$[0\ 0\ \dots\ 1\ 0\ 0]$ …

…

$[i_{1max}\ i_{2max}\ \dots\ i_{M-1max}\ i_{Mmax}]$.

3.4 L’approsimazione dal continuo al discreto

I dati reali sono soggetti al rumore, quindi possono assumere valori non contenuti nell’universo (discreto) delle variabili base; in tal caso il valore da fuzzificare e’ quello piu’ vicino, appartenente all’universo. E’ la ‘ricercabin’ che opera questa discretizzazione.

3.5 La suddivisione in moduli ed il main

Seguendo i principi di incapsulamento, ogni classe ha un suo modulo ed una sua interfaccia.

Il main contiene due funzioni per leggere dei parametri dal file di regole.

Nella pagina seguente rappresentiamo il Diagramma dei Moduli, in UML, dove compare anche l’interazione dell’utente con l’applicazione.

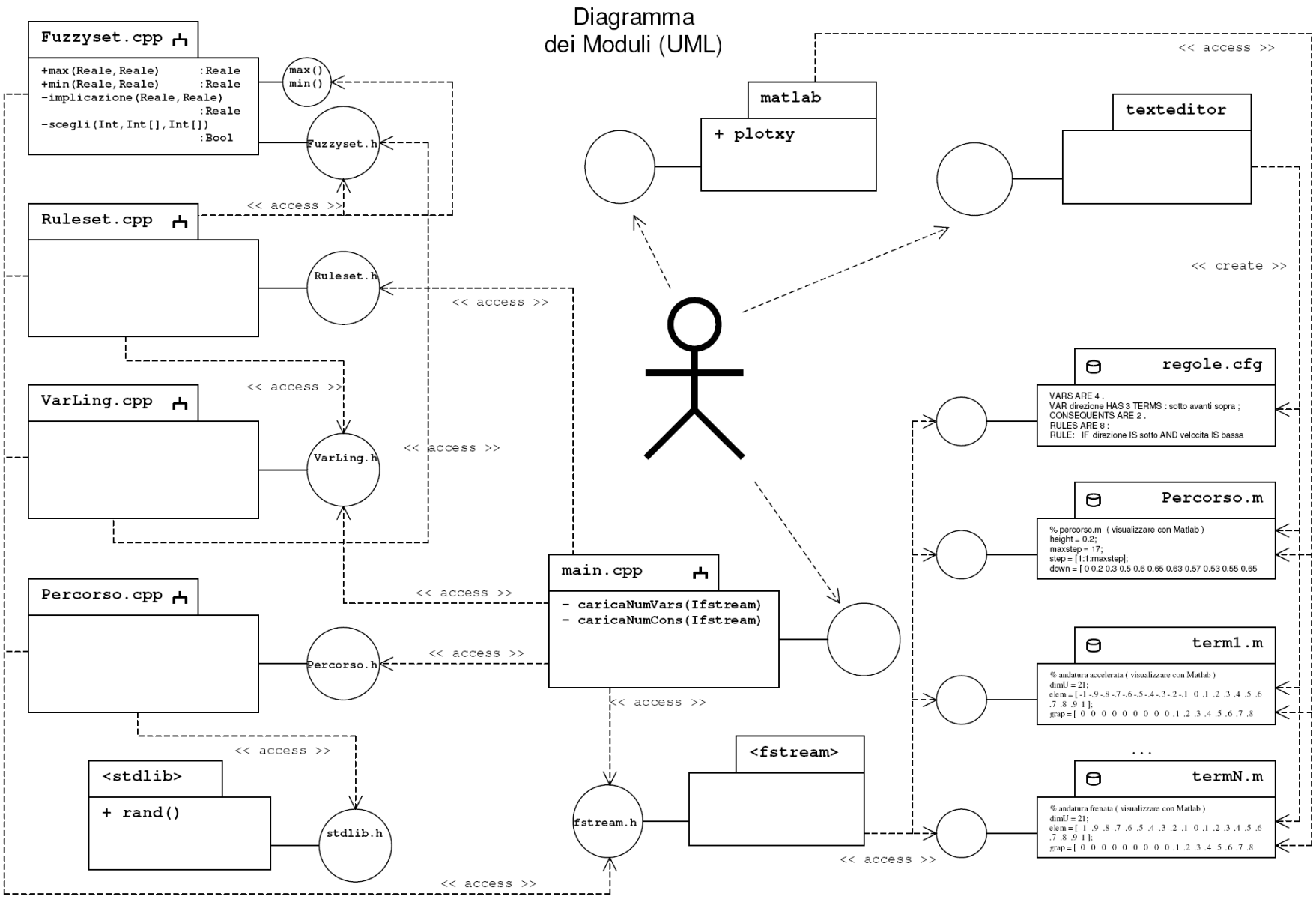


Diagramma
dei Moduli (UML)
Fuzzyset.cpp
+max(Reale,Reale) :Reale
+min(Reale,Reale) :Reale
-implicazione(Reale,Reale) :Reale
-scegli(Int,Int[],Int[]) :Bool
max()
min()
Fuzzyset.h
Ruleset.cpp
Ruleset.h
VarLing.cpp
VarLing.h
Percorso.cpp
Percorso.h
<stdlib>
+ rand()
stdlib.h
main.cpp
- caricaNumVars(Ifstream)
- caricaNumCons(Ifstream)
<fstream>
fstream.h
matlab
+ plotxy
texteditor
regole.cfg
VARS ARE 4 .
VAR direzione HAS 3 TERMS : sotto avanti sopra ;
CONSEQUENTS ARE 2 .
RULES ARE 8 :
RULE: IF direzione IS sotto AND velocita IS bassa
Percorso.m
% percorso.m (visualizzare con Matlab)
height = 0.2;
maxstep = 17;
step = [1:1:maxstep];
down = [0 0.2 0.3 0.5 0.6 0.65 0.63 0.57 0.53 0.55 0.65
term1.m
% andatura accelerata (visualizzare con Matlab)
dimU = 21;
elem = [-1 -.9 -.8 -.7 -.6 -.5 -.4 -.3 -.2 -.1 0 .1 .2 .3 .4 .5 .6 .7 .8 .9 1];
grap = [0 0 0 0 0 0 0 0 0 0 0 .1 .2 .3 .4 .5 .6 .7 .8
...
termN.m
% andatura frenata (visualizzare con Matlab)
dimU = 21;
elem = [-1 -.9 -.8 -.7 -.6 -.5 -.4 -.3 -.2 -.1 0 .1 .2 .3 .4 .5 .6 .7 .8 .9 1];
grap = [0 0 0 0 0 0 0 0 0 0 0 .1 .2 .3 .4 .5 .6 .7 .8
<< access >>
<< create >>

3.6 Listato di tutto il codice

```
////////////////////////////////////////////////////////////////////////////
// fuzzyset.h

#ifndef FUZZYSET_H
#define FUZZYSET_H

typedef char*    stringa;
typedef float    reale;

class FuzzySet
{
private:
    int          DimU;
    reale*       Elem;
    reale*       GrAp;

    int          ricercabin( const reale& elem, int inf, int sup ) const;
    int          pos( const reale& elem ) const;
    inline reale concentra( const int i ) const;
    reale        dilata( const int i ) const;
    reale        intensifica( const int i ) const;
    reale        congiunzione( const FuzzySet A[], const int i[],
                               const int m ) const;
public:
    inline       FuzzySet();
                 FuzzySet( const int dimU );
                 FuzzySet( const FuzzySet& fset );
    inline       ~FuzzySet();
    inline       putElem( const reale& elem, const reale& grAp, int i );
    void         ricostruisci ( const int dimU );
    inline int   dimU() const;
    inline reale infU() const;
    inline reale supU() const;
    inline reale grAp( const int i ) const;
    reale        elem( int i ) const;
    reale        grAp( const reale& elem ) const;
    FuzzySet     concentra() const;
    FuzzySet     dilata() const;
    FuzzySet     intensifica() const;
    FuzzySet     fuzzifica( const reale& x, const int emisupporto ) const;
    reale        defuzzifica() const;
    reale        composizione( const int m, const FuzzySet Ap[],
                               const FuzzySet A[], const reale& By ) const;
    FuzzySet&    operator=( const FuzzySet& fset );
    void         stampa_su_file( const stringa& nomefile ) const;
    void         leggi_da_file( const stringa& nomefile );

};

inline reale FuzzySet::concentra( const int i ) const
{   return GrAp[i] * GrAp[i];   }

inline FuzzySet::FuzzySet()
{   DimU = 0; Elem = 0; GrAp = 0;   }

inline FuzzySet::~FuzzySet()
{   delete[] Elem; delete[] GrAp;   }

inline  FuzzySet::putElem( const reale& elem, const reale& grAp, int i )
```

```
{   Elem[i] = elem; GrAp[i] = grAp; }

inline int FuzzySet::dimU() const
{   return DimU;}

inline reale FuzzySet::infU() const
{   return Elem[0]; }

inline reale FuzzySet::supU() const
{   return Elem[DimU-1];}

inline reale FuzzySet::grAp( const int i ) const
{   return GrAp[i]; }

#endif


////////////////////////////////////////////////////////////////////////////
// fuzzyset.cpp

#include <fstream.h>
#include <stdlib.h>
#include <math.h>
#include "fuzzyset.h"

int FuzzySet::ricercabin( const reale& elem, int inf, int sup ) const
{   if ( inf > sup )
    {   if ( sup < 0 ) return inf;
        if ( inf >= DimU ) return sup;
        if ( ( Elem[inf]-elem ) <= ( elem - Elem[sup] ) )
            return inf;
        return sup;
    }
    if ( inf == sup )
        return inf;
    int med = ( inf + sup ) / 2;
    if ( Elem[med] > elem )
        return ricercabin( elem, inf, med-1 );
    if ( Elem[med] < elem )
        return ricercabin( elem, med+1, sup );
    else
        return med;
}

int FuzzySet::pos( const reale& elem ) const
{   return ricercabin( elem, 0, DimU-1 );
}

reale FuzzySet::dilata( const int i ) const
{   return reale ( sqrt( GrAp[i] ) );   }

reale FuzzySet::intensifica( const int i ) const
{   if  ( GrAp[i] <= 0.5 )
        return 2 * GrAp[i] * GrAp[i];
    return 1 - 2 * ( 1 - GrAp[i] ) * ( 1 - GrAp[i] );
}

reale max( const reale& a, const reale& b )
{   if ( a >= b ) return a;
    return b;
}
```

```
reale min( const reale& a, const reale& b )
{   if ( a <= b ) return a;
    return b;
}

reale implicazione( const reale& Ax, const reale& Bx )
{   return min( Ax,Bx );}               // Mamdani
//  return max( min( Ax, Bx ), 1 - Ax );// Zadeh

reale FuzzySet::congiunzione( const FuzzySet A[], const int i[],
                              const int m ) const
{   reale cong = 1;
    for ( int h = 0; h < m; h++ )
        cong = min ( cong, A[h].GrAp[i[h]] );
    return cong;
}

FuzzySet::FuzzySet( const int dimU )
{   DimU = dimU;
    Elem = new reale[DimU];
    GrAp = new reale[DimU];
}

FuzzySet::FuzzySet( const FuzzySet& fset )
{   DimU = fset.DimU;
    Elem = new reale[DimU];
    GrAp = new reale[DimU];
    for ( int i = 0; i < DimU; i++ )
    {   Elem[i] = fset.Elem[i];
        GrAp[i] = fset.GrAp[i];
    }
}

void FuzzySet::ricostruisci ( const int dimU )
{   if ( DimU == 0)
    {   DimU = dimU;
        Elem = new reale[ DimU ];
        GrAp = new reale[ DimU ];
    }
    else
    {   cerr << "tentativo di ricostuire un insieme non vuoto." << endl;
        exit(1);
    }
}

reale FuzzySet::elem( int i ) const
{   if ( i < 0 || i >= DimU )
    {   cerr << "indice fuori dall'universo." << endl;
        exit(2);
    }
    return Elem[i];
}

reale FuzzySet::grAp( const reale& elem ) const
{   return GrAp[ pos( elem ) ]; }

FuzzySet FuzzySet::concentra() const
{   FuzzySet result( DimU );
    for ( int i = 0; i < DimU; i++ )
    {   result.Elem[i] = Elem[i];
        result.GrAp[i] = concentra(i);
    }
    return result;
```

```
}

FuzzySet FuzzySet::dilata() const
{   FuzzySet result( DimU );
    for ( int i = 0; i < DimU; i++ )
    {   result.Elem[i] = Elem[i];
        result.GrAp[i] = dilata(i);
    }
    return result;
}

FuzzySet FuzzySet::intensifica() const
{   FuzzySet result( DimU );
    for ( int i = 0; i < DimU; i++ )
    {   result.Elem[i] = Elem[i];
        result.GrAp[i] = intensifica(i);
    }
    return result;
}

bool scegli( const int m, int i[], const int imax[] )
{
    for ( int h = 0; ( i[h] == imax[h] ) && ( h < m ) ; h++ ) {}
    if ( h == m )
        return false;
    i[h]++;
    for ( int k = h-1; k >= 0; k-- )
        i[k] = 0;
    return true;
}

reale FuzzySet::composizione( const int m, const FuzzySet Ap[],
                              const FuzzySet A[], const reale& By ) const
{   int* i = new int[m];
    for ( int h = 0; h < m; h++ ) i[h] = 0;
    int* imax = new int[m];
    for ( h = 0; h < m; h++ ) imax[h] = A[h].DimU-1;
    reale sup = 0;
    do
        sup = max( sup, min( congiunzione( Ap, i, m ),
                             implicazione( congiunzione( A, i, m ), By ) ) );

    while ( scegli( m, i, imax ) );

    delete[] i;
    delete[] imax;
    return sup;
}

FuzzySet FuzzySet::fuzzifica( const reale& x, const int emisupporto ) const
{   int ix = pos(x);
    int sx = ix - emisupporto;
    if ( sx < 0 ) sx = 0;
    int dx = ix + emisupporto;
    if ( dx >= DimU ) dx = DimU - 1;
        GrAp[ix] = 1;
    for ( int i = 0; i < sx; i++ )
        GrAp[i] = 0;
    for ( i = sx; i < ix; i++ )
        GrAp[i] = ( Elem[i] - Elem[sx] ) / ( x - Elem[sx] );
    for ( i = ix + 1; i <= dx; i++ )
        GrAp[i] = ( Elem[dx] - Elem[i] ) / ( Elem[dx] - x );
    for ( i = dx + 1; i < DimU; i++ )
```

```
        GrAp[i] = 0;
    return (*this);
}

reale FuzzySet::defuzzifica() const
{   reale area = 0;
    reale areapesata = 0;
    for ( int i = 0; i < DimU; i++ )
    {   areapesata += GrAp[i] * Elem[i];
        area += GrAp[i] ;
    }
    if ( area == 0 ) area = 1;
    return areapesata / area ;
}

FuzzySet& FuzzySet::operator=( const FuzzySet& fset )
{   if ( this == &fset )
        return *this;
    else
    if ( DimU != fset.DimU )
    {   cerr << "Insiemi di diverse dimensioni." << endl;
        exit(3);
    }
    else
    for ( int i = 0; i < DimU; i++ )
    {   Elem[i] = fset.Elem[i];
        GrAp[i] = fset.GrAp[i];
    }
    return (*this);
}

void FuzzySet::stampa_su_file( const stringa& nomefile ) const
{   ofstream out( nomefile );
        if ( !out )
        {   cerr << "errore di apertura file." << endl;
            exit(4);
        }
        out << "% visualizzare con Matlab 4\n";
        out << "dimU = " << DimU << ";\n";
        out << "elem = [ ";
        for ( int i = 0; i < DimU; i++ )
            out << Elem[i] << ' ';
        out << "];" << endl << "grap = [ ";
        for ( i = 0; i < DimU; i++ )
            out << GrAp[i] << ' ';
        out << "];" << endl;
        out <<
        "clf;\n"
        "plot(elem,grap,'k');\n"
        "axis([min(elem) max(elem) -.02 1.02]);\n"
        "grid on;\n";
    out.close();
}

void FuzzySet::leggi_da_file( const stringa& nomefile )
{   charleggi;
    int     leggiNum;
    ifstream in( nomefile );
        if ( !in )
        {   cerr << "errore di apertura file." << endl;
            exit(4);
        }
        do in >> leggi; while ( leggi != '=');
```

```
        in >> leggiNum;
        ricostruisci( leggiNum );
        do in >> leggi; while ( leggi != '[');
        for ( int i = 0; i < DimU; i++ )
            in >> Elem[i];
        do in >> leggi; while ( leggi != '[');
        for ( i = 0; i < DimU; i++ )
            in >> GrAp[i];
        in >> leggi;
        if ( leggi != ']' )
        {   cerr << "errore nel numero di elementi" << endl;
            exit(5);
        }
    in.close();
}
```

```
////////////////////////////////////////////////////////////////////////////
// varling.h

#ifndef VARLING_H
#define VARLING_H

#include "fuzzyset.h"

const MAXLTERM = 128;
typedef char*   label;

class VarLing
{
private:
    label           Nome;
    int             NumTerm;
    label*          Term;
    FuzzySet*       Mean;
public:
    inline          VarLing();
                    ~VarLing();
                    VarLing( const VarLing& VL );
    inline label    nome();
    inline FuzzySet mean( const int i) const;
    FuzzySet        mean( const label& L ) const;
    inline label    term( const int i ) const;
    FuzzySet        hedge( const label& M, const FuzzySet& fset );
    void            caricaVarLing( ifstream& in );
};

inline label VarLing::term( const int i ) const
{   return Term[i]; }

inline VarLing::VarLing()
{   Nome = '\0'; NumTerm = 0; Term = 0; Mean = 0;    }

inline label VarLing::nome()
{   return Nome;}

inline FuzzySet VarLing::mean( const int i) const
{   return Mean[i]; }

#endif
```

```
////////////////////////////////////////////////////////////////////////////
// varling.cpp

#include <string.h>
#include <stdlib.h>
#include <fstream.h>
#include <math.h>

#include "varling.h"

VarLing::~VarLing()
{   delete[] Nome;
    for ( int i = 0; i < NumTerm; i++ )
        delete Term[i];
    delete[] Term;
    delete[] Mean;
}

VarLing::VarLing( const VarLing& VL )
{   Nome = new char[ strlen( VL.Nome ) ];
    strcpy( Nome, VL.Nome );
    NumTerm = VL.NumTerm;
    Term = new label[ NumTerm ];
    Mean = new FuzzySet[ NumTerm ];
    for ( int i = 0; i < NumTerm; i++ )
    {   Term[i] = new char[ strlen( VL.Term[i] )];
        strcpy( Term[i], VL.Term[i] );
        Mean[i].ricostruisci( VL.Mean[i].dimU() );
        Mean[i] = VL.Mean[i];
    }
}

FuzzySet VarLing::mean( const label& L ) const
{   for ( int i = 0; i < NumTerm  && strcmp( L, Term[i] ); i++ ) {}
    if  ( i == NumTerm )
    {   cerr << "valore linguistico " << L << " sconosciuto." << endl;
        exit(6);
    }
    return Mean[i];
}

FuzzySet VarLing::hedge( const label& M, const FuzzySet& fset )
{   if ( !strcmp("molto", M ) )
        return fset.concentra();
    if ( !strcmp("leggermente", M ) )
        return fset.dilata();
    if ( !strcmp("intensamente", M ) )
        return fset.intensifica();
    return fset;
}

void VarLing::caricaVarLing( ifstream& in )
{   if (!in)
    {   cerr << "errore di apertura file" << endl;
        exit(4);
    }
    label leggiTerm = new char[ MAXLTERM ];
    in >> leggiTerm;                        // VAR
    if ( strcmp ("VAR", leggiTerm ) )
    {   cerr << "errore di sintassi: non trovo \'VAR\'." << endl;
        exit(7);
```

```
    }
    in >> leggiTerm;                          // nome della var. ling.
    Nome = new char[ strlen(leggiTerm) ];
    strcpy( Nome, leggiTerm );
    in >> leggiTerm;                          // HAS
    if ( strcmp("HAS",leggiTerm) )
    {   cerr << "errore di sintassi: non trovo \'HAS\'." << endl;
        exit(8);
    }
    in >> NumTerm;                                // numero
    in >> leggiTerm; in >> leggiTerm;             // TERMS :
    Term = new label[ NumTerm ];
    Mean = new FuzzySet[ NumTerm ];
    for ( int i = 0; i < NumTerm; i++ )
    {   in >> leggiTerm;                      // termine linguistico
        Term[i] = new char[ strlen(leggiTerm)]; // nome del termine
        strcpy( Term[i], leggiTerm );
        strncpy( leggiTerm, Nome, 3 );            // nome del file composto
        leggiTerm[3] = '\0';                  // da 3 caratt della var
        strncat( leggiTerm, Term[i], 5 );         // e 5 del termine
        strcat( leggiTerm, ".m" );
        Mean[i].leggi_da_file( leggiTerm );
    }
    in >> leggiTerm;
}
```

```
////////////////////////////////////////////////////////////////////////////
// ruleset.h

#ifndef RULESET_H
#define RULESET_H

#include "fuzzyset.h"
#include "varling.h"

const MAXLUNLABEL = 128;
typedef VarLing* VarLingVett;

class RuleSet
{
private:
    int             NumRule;
    int             NumAntec;
    FuzzySet*       A;
    FuzzySet*       B;
public:
    inline          RuleSet();
    void            ricostruisci( const int numRule, const int numAntec );
                    RuleSet( const RuleSet& Rset );
    inline          ~RuleSet();
    inline int      numRule() const;
    inline int      numAntec() const;
    FuzzySet        inferenzaFITA( FuzzySet Ap[] ) const;
    void            caricaRuleSet(  const VarLingVett& VLV,
                                    const int dimVett,
                                    const int numantec, ifstream& in );

};

inline RuleSet::RuleSet()
{   NumRule = 0;NumAntec = 0;   A = 0;  B = 0;  }

inline RuleSet::~RuleSet()
{   delete[] A; delete[] B; }

inline int  RuleSet::numRule() const
{   return NumRule;     }

inline int RuleSet::numAntec() const
{   return NumAntec;}

#endif
```

```
////////////////////////////////////////////////////////////////////////////
// ruleset.cpp

#include <fstream.h>
#include <string.h>
#include <stdlib.h>

#include "ruleset.h"

void RuleSet::ricostruisci( const int numRule, const int numAntec )
{   if ( NumRule == 0 )
    {   NumRule = numRule;
        NumAntec = numAntec;
        A = new FuzzySet[ NumRule * NumAntec ];
        B = new FuzzySet[ NumRule ];
    }
    else
    {   cerr << "tentativo di ricostruire una ruleset non vuota." << endl;
        exit(9);
    }
}

RuleSet::RuleSet( const RuleSet& Rset )
{   NumRule = Rset.NumRule;
    NumAntec = Rset.NumAntec;
    A = new FuzzySet[ NumRule * NumAntec ];
    for ( int i = 0; i < NumRule; i++ )
        for ( int j = 0; j < NumAntec; j++ )
        {   A[i*NumAntec+j].ricostruisci( Rset.A[i*NumAntec+j].dimU() );
            A[i*NumAntec+j] = Rset.A[i*NumAntec+j];
        }
    B = new FuzzySet[ NumRule ];
    for ( i = 0; i < NumRule; i++ )
    {   B[i].ricostruisci( Rset.B[i].dimU() );
        B[i] = Rset.B[i];
    }
}

extern reale max( const reale& a, const reale& b );
extern reale min( const reale& a, const reale& b );

FuzzySet RuleSet::inferenzaFITA( FuzzySet Ap[] ) const
{
    FuzzySet result( B[0].dimU() );
    reale aggregazione;
    for ( int j = 0; j < result.dimU(); j++ )
    {   aggregazione = 0;
        for ( int i = 0; i < NumRule; i++ )
            aggregazione =
            max ( aggregazione,
                  result.composizione( NumAntec, Ap, &A[i* NumAntec],
                                       B[i].grAp(j) ) );
        result.putElem( B[0].elem(j), aggregazione, j );
    }
    return result;
}

void RuleSet::caricaRuleSet( const VarLingVett& VLV, const int dimVett,
                             const int numantec,ifstream& in )
{   int leggiNum;
```

```
    label leggiVarLing = new char[ MAXLUNLABEL ];
    label leggiModif = new char[ MAXLUNLABEL ];
    label leggiTerm = new char[ MAXLUNLABEL ];
    do  { in >> leggiVarLing; }
    while ( strcmp( "ARE", leggiVarLing ) );// ARE
    in >> leggiNum;
    ricostruisci( leggiNum, numantec );

    in >> leggiVarLing;                          // :
    for ( int i = 0; i < NumRule; i++ )
    {   in >> leggiVarLing;                      // RULE:
        for ( int j = 0; j < NumAntec+1; j++ )
        {   in >> leggiVarLing;                  // IF oppure AND o THEN
            in >> leggiVarLing;                  // termine
            in >> leggiTerm;                // IS
            in >> leggiTerm;                // termine
            if ((!strcmp( leggiTerm, "molto" )) ||
                (!strcmp( leggiTerm, "leggermente" )) ||
                (!strcmp( leggiTerm, "intensamente" )) )
            { strcpy(leggiModif,leggiTerm);
              in >> leggiTerm;                   // termine
            }
            else
            strcpy(leggiModif,"nessuno");

            for ( int k = 0; (strcmp( leggiVarLing, VLV[k].nome() )) &&
                    ( k < dimVett ) ; k++) {}
            if  ( k == dimVett )
            {   cerr << "variabile linguistica sconosciuta" << endl;
                exit(10);
            }
            if (j < NumAntec )
            {   A[i*NumAntec+j].ricostruisci( VLV[k].mean(leggiTerm).dimU() );
                A[i*NumAntec+j]=VLV[k].hedge(leggiModif,VLV[k].mean(leggiTerm));
            }
            else
            {   B[i].ricostruisci( VLV[k].mean(leggiTerm).dimU() );
                B[i] = VLV[k].hedge(leggiModif, VLV[k].mean(leggiTerm));
            }
        }
        in >> leggiVarLing;                      // ; oppure .
    }
}
```

```
////////////////////////////////////////////////////////////////////////////
// percorso.h

#ifndef PERCORSO_H
#define PERCORSO_H

typedef float   reale;
typedef char*   stringa;
typedef reale*  realeVett;

class Percorso
{
private:
    int             maxStep;
    realeVett       traet;
    int             numSen;
    realeVett       sensori;
    int             numAtt;
    realeVett       attuatori;
    int             step;
    realeVett       down;
    reale           height;
    inline  reale   rumore();
public:
                    Percorso();
                    Percorso( const int nAtt, const int nSen );
    void            ricostruisci( const int mStep,  const reale& h );
                    Percorso( Percorso& Per );
    inline          ~Percorso();
    inline bool     arrivo();
    void            posiziona();
    bool            doStep();
    bool            dentropercorso();
    void            leggiSens( reale crispAntec[] );
    void            modifAtt( const reale crispCons[] );
    void            stampa_su_file( const stringa& nomefile );
    void            leggi_da_file( const stringa& nomefile );
};

inline reale Percorso::rumore()
{   return (reale(rand())/reale(RAND_MAX)-reale(0.5) )*reale(0.8);  }

inline Percorso::~Percorso()
{   delete[] traet;         delete[] sensori;
    delete[] attuatori;     delete[] down;
}

inline bool Percorso::arrivo()
{   return ( step == ( maxStep - 1 ) ); }

#endif
```

```
////////////////////////////////////////////////////////////////////////////
// percorso.cpp

#include <fstream.h>
#include <stdlib.h>
#include <math.h>

#include "percorso.h"

bool Percorso::dentropercorso()
{   return ( ( traet[step] > down[step] &&
             ( traet[step] < down[step] + height) ) ) ; }

Percorso::Percorso()
{   maxStep = 0;    traet = 0;      numSen = 0;     sensori = 0;
    numAtt = 0;     attuatori = 0;  step = 0;       down = 0;
    height = 0;
}

Percorso::Percorso( const int nAtt, const int nSen )
{   maxStep     = 0;
    traet       = 0;
    numSen      = nSen;
    sensori     = new reale[numSen];
    numAtt      = nAtt;
    attuatori   = new reale[numAtt];
    step        = 0;
    down        = 0;
    height      = 0;
}

void Percorso::ricostruisci( const int mStep,  const reale& h )
{   if ( maxStep != 0 )
    {   cerr << "tentativo di ricostruire un percorso non vuoto." << endl;
        exit(12);
    }
    maxStep     = mStep;
    traet       = new reale[ maxStep];
    down        = new reale[ maxStep];
    height      = h;

}

Percorso::Percorso( Percorso& Per )
{   maxStep     = Per.maxStep;
    traet       = new reale[ maxStep];
    numSen      = Per.numSen;
    sensori     = new reale[ numSen];
    numAtt      = Per.numAtt;
    attuatori   = new reale[ numAtt];
    step        = Per.step;
    down        = new reale[ maxStep];
    height      = Per.height;
    for ( int i = 0; i < maxStep; i++ )
    {   traet[i]    = Per.traet[i];
        down[i]     = Per.down[i];
    }
    for ( i = 0; i < numSen; i++ )
        sensori[i] = Per.sensori[i];
    for ( i = 0; i < numAtt; i++ )
```

```
        attuatori[i] = Per.attuatori[i];
}

void Percorso::posiziona()
{   traet[step] = down[step] + height/2;
    reale& direzione = sensori[0];
    reale& velocita  = sensori[1];
    direzione = 0;
    velocita = 0;
}

bool Percorso::doStep()
{   reale&  direzione = sensori[0];
    reale&  velocita  = sensori[1];
    reale   virata    = attuatori[0];
    reale   andatura  = attuatori[1];
    int&s         = step;

    cout << s+1 << "\t" << direzione << "\t" << velocita
         << "\t" << virata << "\t" << andatura << "\t\n" ;
    cout.flush();

    if ( arrivo() )
        return false;

    if ( !dentropercorso() )
    {   step++;
        posiziona();
        return true;
    }

    step++;
    if ( velocita == 0 )
        traet[s] = traet[s-1] + down[s] - down[s-1];
    else
        traet[s] = 2*traet[s-1] - traet[s-2] +
                    virata*(1-reale(fabs(andatura)))*height;

    velocita =  velocita + andatura + rumore();
    if ( velocita > 1 ) velocita = 1;
    if ( velocita < 0 ) velocita = 0;

    direzione = reale( (2*traet[s]-traet[s-1]-down[s+1]-height/2)*2/height
                + rumore() );
    if ( direzione < -1 ) direzione = -1;
    if ( direzione > 1 ) direzione = 1;

    return ( s < maxStep-1 );
}


void Percorso::leggiSens( reale crispAntec[] )
{   for ( int i = 0; i < numSen; i++ )
        crispAntec[i] = sensori[i];
}

void Percorso::modifAtt( const reale crispCons[] )
{   for ( int i = 0; i < numAtt; i++ )
        attuatori[i] = crispCons[i];
}

void Percorso::stampa_su_file( const stringa& nomefile )
{   ofstream out( nomefile );
```

```
        if ( !out )
        {   cerr << "errore di apertura file." << endl;
            exit(4);
        }
        out << "% percorso.m  ( visualizzare con Matlab )\n"
        "height = " << height << ";\n"
        "maxstep = " << maxStep << ";\n"
        "step = [1:1:maxstep];\n"
        "down = [ ";
        for ( int i = 0; i < maxStep; i++ )
            out << down[i] << ' ';
        out << "];\n"
        "top = down + height;\n"
        "traett = [ ";
        for ( i = 0; i < maxStep; i++ )
            out << traet[i] << ' ';
        out << "];\n"
        "clf;\n"
        "plot(step,down,'k',step,top,'k',step,traett,':k');\n"
        "axis([min(step)-.1 max(step)+.1  -0.1  max(top)+.1]);\n";
    out.close();
}

void Percorso::leggi_da_file( const stringa& nomefile )
{   charleggi;
    int     mStep;
    reale   h;
    ifstream in( nomefile );
        if ( !in )
        {   cerr << "errore di apertura file." << endl;
            exit(4);
        }
        do in >> leggi; while ( leggi != '=');
        in >> h;
        do in >> leggi; while ( leggi != '=');
        in >> mStep;
        ricostruisci( mStep, h );
        do in >> leggi; while ( leggi != ']');
        do in >> leggi; while ( leggi != '[');
        for ( int i = 0; i < maxStep; i++ )
            in >> down[i];
    in.close();
}
```

```
////////////////////////////////////////////////////////////////////////////
// main.cpp

#include <stdlib.h>
#include <fstream.h>
#include <string.h>

#include "varling.h"
#include "ruleset.h"
#include "percorso.h"

const stringa       fileregole = "regole.cfg";
const stringa       filepercorso = "percorso.m";
const int           emisupporto = 1;
typedef VarLing*    VarLingVett;


int caricaNumVars( ifstream& in )
{   int leggiNum;
    label leggiLabel = new char[ MAXLUNLABEL ];
    do  in >> leggiLabel;
    while ( strcmp( "VARS", leggiLabel ) );         // VARS
    in >> leggiLabel;                               // ARE
    in >> leggiNum;                                 // numVars
    if ( leggiNum <= 0 )
    {   cerr << "errore di sintassi su \'VARS ARE ...\'." << endl;
        exit(10);
    }
    in >> leggiLabel;                               // .
    return leggiNum;
}

int caricaNumCons( ifstream& in)
{   int leggiNum;
    label leggiLabel = new char[ MAXLUNLABEL ];
    do  in >> leggiLabel;
    while ( strcmp( "CONSEQUENTS", leggiLabel ) );  // CONSEQUENTS
    in >> leggiLabel;                               // ARE
    in >> leggiNum;
    if ( leggiNum <= 0 )
    {   cerr << "errore di sintassi su \'CONSEQUENTS ARE ...\'." << endl;
        exit(11);
    }
    in >> leggiLabel;                               // .
    return leggiNum;
}


void main()
{   ifstream in( fileregole );
        int numVars = caricaNumVars( in );

        VarLingVett Vars = new VarLing[ numVars ];
        for ( int i = 0; i < numVars; i++ )
            Vars[i].caricaVarLing( in );

        int numCons = caricaNumCons(in);
        int numAntec = numVars - numCons;
        RuleSet* RSet = new RuleSet[ numCons ];
        for ( i = 0; i < numCons; i++ )
```

```
            RSet[i].caricaRuleSet( Vars, numVars, numAntec, in );

    in.close();

    Percorso P( numAntec, numCons );
    P.leggi_da_file( filepercorso );

    FuzzySet* FuzzyAntec = new FuzzySet[numAntec];
    for ( i = 0; i < numAntec; i++ )
    {   FuzzyAntec[i].ricostruisci( Vars[i].mean(0).dimU() );
        FuzzyAntec[i] = Vars[i].mean(0);
    }

    FuzzySet* FuzzyCons = new FuzzySet[numCons];
    for ( i = numAntec; i < numVars; i++ )
    {   FuzzyCons[i-numAntec].ricostruisci( Vars[i].mean(0).dimU() );
        FuzzyCons[i-numAntec] = Vars[i].mean(0);
    }

    reale*    CrispAntec = new reale[numAntec];
    reale*    CrispCons = new reale[numCons];

    P.posiziona();

    cout << "\tdirez\tveloc\tvirat\tandat\n" ;
    cout.precision(2);
    cout.setf(ios::fixed);

    do
    {   P.leggiSens( CrispAntec );
        for ( int i = 0; i < numAntec; i++)
            FuzzyAntec[i]=FuzzyAntec[i].fuzzifica(CrispAntec[i],emisupporto);

        for ( i = 0; i < numCons; i++ )
        {   FuzzyCons[i] = RSet[i].inferenzaFITA( FuzzyAntec );
            CrispCons[i] = FuzzyCons[i].defuzzifica();
        }

        P.modifAtt( CrispCons );
    }
    while ( P.doStep() );

    P.stampa_su_file( filepercorso );
}
```

```
////////////////////////////////////////////////////////////////////////////
// files della directory di progetto
 Il volume nell'unità D è DATI
 Directory di D:\c++\jet.src

DIRSOTTO M             329  13/06/01   0.28 dirsotto.m
ANDACCEL M             331  13/06/01   0.17 andaccel.m
ANDCOSTA M             339  13/06/01   0.24 andcosta.m
ANDFRENA M             330  13/06/01   0.25 andfrena.m
DIRAVANT M             330  13/06/01   0.27 diravant.m
DIRSOPRA M             327  13/06/01   0.26 dirsopra.m
LEGGIMI  TXT           786  13/06/01  22.04 leggimi.txt
FUZZYSET CPP         5.939  13/06/01   2.02 fuzzyset.cpp
FUZZYSET H           2.111  11/06/01  16.14 fuzzyset.h
JET      DSP         4.757  13/06/01  21.44 jet.dsp
JET      DSW           531  13/06/01  21.35 jet.dsw
JET      NCB        74.752  13/06/01  21.45 jet.ncb
JET      OPT        48.640  13/06/01  21.45 jet.opt
JET      PLG         1.438  13/06/01  21.45 jet.plg
MAIN     CPP         2.677  13/06/01  13.25 main.cpp
PERCORSO CPP         4.040  13/06/01   2.35 percorso.cpp
PERCORSO H           1.224  13/06/01   2.08 percorso.h
PERCORSO M             467  13/06/01  21.41 percorso.m
REGOLE   CFG         1.764  12/06/01  15.31 regole.cfg
RULESET  CPP         3.064  13/06/01   2.02 ruleset.cpp
RULESET  H           1.003  12/06/01  17.42 ruleset.h
VARLING  CPP         2.426  11/06/01  15.51 varling.cpp
VARLING  H             932  11/06/01  15.17 varling.h
VELALTA  M             326  13/06/01   0.31 velalta.m
VELBASSA M             325  13/06/01   0.30 velbassa.m
VELMEDIA M             328  13/06/01   0.30 velmedia.m
VIRGIU   M             324  13/06/01   0.29 virgiu.m
VIRNESSU M             328  13/06/01   0.28 virnessu.m
VIRSU    M             323  13/06/01   0.26 virsu.m
DIR1     TXT             0  13/06/01  22.09 dir1.txt
        30 file        160.491 byte
         2 dir       124.067.840 byte disponibili

////////////////////////////////////////////////////////////////////////////
// files della directory di esecuzione
 Il volume nell'unità D è DATI
 Directory di D:\c++\jet.exe

DIRAVANT M             330  13/06/01   0.27 diravant.m
ANDCOSTA M             339  13/06/01   0.24 andcosta.m
ANDFRENA M             330  13/06/01   0.25 andfrena.m
ANDACCEL M             331  13/06/01   0.17 andaccel.m
DIRSOPRA M             327  13/06/01   0.26 dirsopra.m
DIRSOTTO M             329  13/06/01   0.28 dirsotto.m
PERCORSO M             467  13/06/01  22.00 percorso.m
REGOLE   CFG         1.764  12/06/01  15.31 regole.cfg
VELALTA  M             326  13/06/01   0.31 velalta.m
VELBASSA M             325  13/06/01   0.30 velbassa.m
VELMEDIA M             328  13/06/01   0.30 velmedia.m
VIRGIU   M             324  13/06/01   0.29 virgiu.m
VIRNESSU M             328  13/06/01   0.28 virnessu.m
VIRSU    M             323  13/06/01   0.26 virsu.m
JET      EXE       253.993  13/06/01  21.39 jet.exe
LEGGIMI  TXT           786  13/06/01  22.04 leggimi.txt
DIR      TXT             0  13/06/01  22.08 dir.txt
```

```
      17 file         260.950 byte
       2 dir        124.076.032 byte disponibili
```

4 TEST DI VERIFICA

4.1 Prova di esecuzione

Fig. regole semantiche dei vari termini

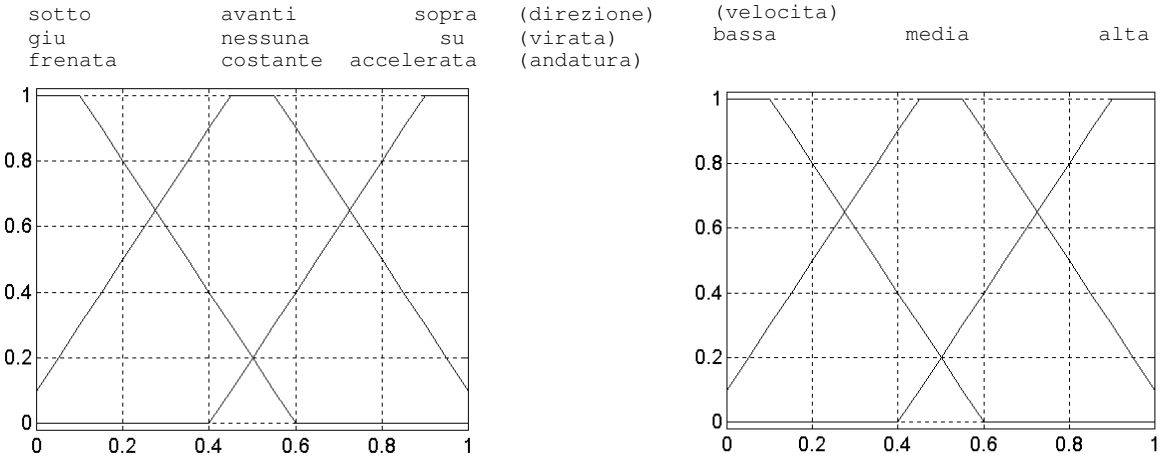
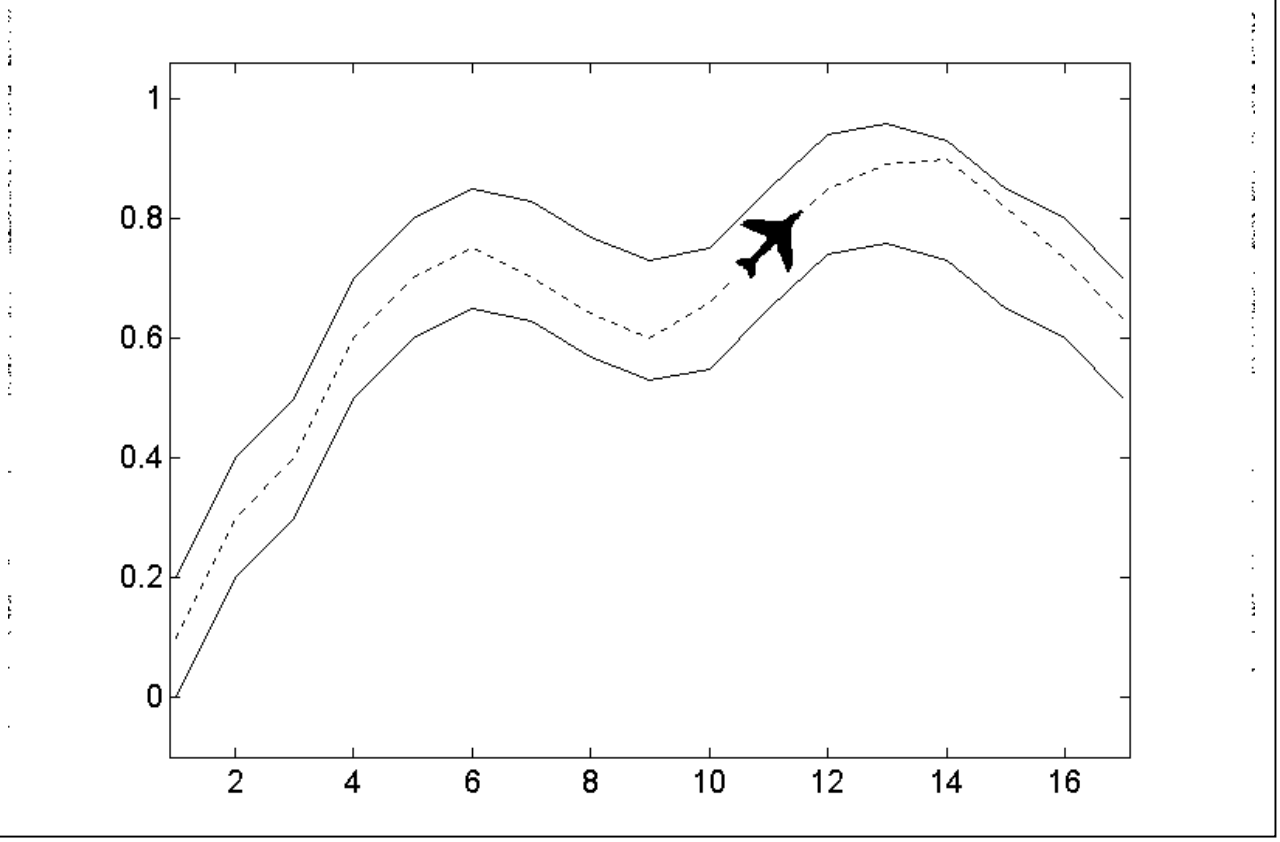


Fig. risultato della guida



```
////////////////////////////////////////////////////////////////////////////
// Output prodotto durante l'esecuzione

    direz   veloc   virat   andat
1   0.00    0.00    0.00    0.39
2   1.00    0.00    -0.49   -0.00
3   -0.75   0.00    0.20    0.05
4   0.98    0.11    -0.47   -0.05
5   1.00    0.00    -0.49   -0.00
6   0.99    0.26    -0.43   -0.15
7   0.73    0.00    -0.20   0.05
8   -0.12   0.21    0.00    0.24
9   -1.00   0.30    0.43    -0.15
10  -1.00   0.00    0.49    -0.00
11  -0.37   0.00    0.03    0.15
12  0.45    0.54    -0.11   -0.22
13  0.96    0.02    -0.49   -0.00
14  0.83    0.00    -0.29   0.02
15  0.13    0.05    0.00    0.32
16  0.36    0.45    -0.08   -0.18
```

Fig. Tabella delle transizioni di stato realizzate dalle regole

VIRATA	DIREZIONE→	SOTTO	AVANTI	SOPRA
VELOCITA →	BASSA	**SU**	**- NESS**	**GIU**
	MEDIA	**+SU**	**NESS**	**+GIU**
	ALTA	**++SU**	**-**	**++GIU**

ANDATURA	DIREZIONE →	SOTTO	AVANTI	SOPRA
VELOCITA →	BASSA	**COST**	**ACC**	**COST**
	MEDIA	**FREN**	**-**	**FREN**
	ALTA	**+FREN**	**COST**	**+FREN**

```
////////////////////////////////////////////////////////////////////////////
// regole.cfg
// MODIFIERS MUST BE : molto leggermente intensamente
// ELSE THEY ARE IGNORED.
VARS ARE 4 .
VAR direzione HAS 3 TERMS : sotto avanti sopra ;
VAR velocita HAS 3 TERMS : bassa media alta ;
VAR virata HAS 3 TERMS :  giu nessuna su ;
VAR andatura HAS 3 TERMS : frenata costante accelerata .

CONSEQUENTS ARE 2 .
RULES ARE 8 :

RULE:   IF  direzione IS sotto AND velocita IS bassa
        THEN virata IS su ;

RULE:   IF  direzione IS sotto AND velocita IS media
        THEN virata IS molto su ;

RULE:   IF  direzione IS sotto AND velocita IS alta
        THEN virata IS intensamente su ;

RULE:   IF  direzione IS sopra AND velocita IS bassa
        THEN virata IS giu ;

RULE:   IF  direzione IS sopra AND velocita IS media
        THEN virata IS molto giu ;

RULE:   IF  direzione IS sopra AND velocita IS alta
        THEN virata IS intensamente giu ;

RULE:   IF  direzione IS avanti AND velocita IS bassa
        THEN virata IS leggermente nessuna ;

RULE:   IF  direzione IS avanti AND velocita IS media
        THEN virata IS nessuna .

RULES ARE 8 :

RULE:   IF  direzione IS sotto AND velocita IS bassa
        THEN andatura IS costante ;

RULE:   IF  direzione IS sotto AND velocita IS media
        THEN andatura IS frenata ;

RULE:   IF  direzione IS sotto AND velocita IS alta
        THEN andatura IS molto frenata ;

RULE:   IF  direzione IS sopra AND velocita IS bassa
        THEN andatura IS costante ;

RULE:   IF  direzione IS sopra AND velocita IS media
        THEN andatura IS frenata ;

RULE:   IF  direzione IS sopra AND velocita IS alta
        THEN andatura IS molto frenata ;

RULE:   IF  direzione IS avanti AND velocita IS bassa
        THEN andatura IS accelerata ;
```

RULE: IF direzione IS avanti AND velocita IS alta
THEN andatura IS costante .

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% file andaccel.m
% andatura accelerata ( visualizzare con Matlab )

dimU = 21;
elem = [ -1 -.9 -.8 -.7 -.6 -.5 -.4 -.3 -.2 -.1   0  .1  .2  .3  .4  .5  .6  .7  .8  .9  1 ];
grap = [  0   0   0   0   0   0   0   0   0  .1  .2  .3  .4  .5  .6  .7  .8  .9   1   1  1 ];
clf;
plot(elem,grap,'k');
axis([min(elem) max(elem) -.02 1.02]);
grid on;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% file andcosta.m
% andatura costante ( visualizzare con Matlab )

dimU = 21;
elem = [ -1 -.9 -.8 -.7 -.6 -.5 -.4 -.3 -.2 -.1   0  .1  .2  .3  .4  .5  .6  .7  .8  .9   1 ];
grap = [  .1   .2   .3   .4  .5  .6  .7  .8  .9   1   1   1  .9  .8  .7  .6  .5   .4   .3   .2   .1 ];
clf;
plot(elem,grap,'k');
axis([min(elem) max(elem) -.02 1.02]);
grid on;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% file andfrena.m
% andatura frenata ( visualizzare con Matlab )

dimU = 21;
elem = [ -1 -.9 -.8 -.7 -.6 -.5 -.4 -.3 -.2 -.1  0   .1  .2  .3  .4  .5  .6  .7  .8  .9   1 ];
grap = [  1   1   1  .9  .8  .7  .6  .5  .4  .3  .2  .1   0   0   0   0   0   0   0   0   0 ];
clf;
plot(elem,grap,'k');
axis([min(elem) max(elem) -.02 1.02]);
grid on;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% file diravant.m
% direzione avanti ( visualizzare con Matlab )

dimU = 21;
elem = [ -1 -.9 -.8 -.7 -.6 -.5 -.4 -.3 -.2 -.1   0  .1  .2  .3  .4  .5  .6  .7  .8  .9   1 ];
grap = [ .1  .2  .3  .4  .5  .6  .7  .8  .9   1   1   1  .9  .8  .7  .6  .5  .4  .3  .2  .1 ];
clf;
plot(elem,grap,'k');
axis([min(elem) max(elem) -.02 1.02]);
grid on;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% file dirsopra.m
% direzione sopra ( visualizzare con Matlab )

dimU = 21;
elem = [ -1 -.9 -.8 -.7 -.6 -.5 -.4 -.3 -.2 -.1   0  .1  .2  .3  .4  .5  .6  .7  .8  .9  1 ];
grap = [  0   0   0   0   0   0   0   0   0  .1  .2  .3  .4  .5  .6  .7  .8  .9   1   1  1 ];
clf;
plot(elem,grap,'k');
axis([min(elem) max(elem) -.02 1.02]);
grid on;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% file dirsotto.m
% direzione sotto ( visualizzare con Matlab )

dimU = 21;
elem = [ -1 -.9 -.8 -.7 -.6 -.5 -.4 -.3 -.2 -.1  0   .1  .2  .3  .4  .5  .6  .7  .8  .9   1 ];
grap = [  1   1   1  .9  .8  .7  .6  .5  .4  .3  .2  .1   0   0   0   0   0   0   0   0   0 ];
clf;
plot(elem,grap,'k');
axis([min(elem) max(elem) -.02 1.02]);
grid on;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% file velalta.m
% velocita alta ( visualizzare con Matlab )

dimU = 21;
elem = [ 0 .05 .10 .15 .20 .25 .30 .35 .40 .45 .50 .55 .60 .65 .70 .75 .80 .85 .90 .95  1 ];
grap = [  0   0   0   0   0   0   0   0   0  .1  .2  .3  .4  .5  .6  .7  .8  .9   1   1  1 ];

clf;
plot(elem,grap,'k');
axis([min(elem) max(elem) -.02 1.02]);
grid on;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% file velbassa.m
% velocita bassa ( visualizzare con Matlab )

dimU = 21;
elem = [ 0 .05 .10 .15 .20 .25 .30 .35 .40 .45 .50 .55 .60 .65 .70 .75 .80 .85 .90 .95  1 ];
grap = [ 1   1   1  .9  .8  .7  .6  .5  .4  .3  .2  .1   0   0   0   0   0   0   0   0   0 ];
clf;
plot(elem,grap,'k');
axis([min(elem) max(elem) -.02 1.02]);
grid on;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% file velmedia.m
% velocita media ( visualizzare con Matlab )

dimU = 21;
elem = [ 0 .05 .10 .15 .20 .25 .30 .35 .40 .45 .50 .55 .60 .65 .70 .75 .80 .85 .90 .95  1 ];
grap = [ .1  .2  .3  .4  .5  .6  .7  .8  .9   1   1   1  .9  .8  .7  .6  .5  .4  .3  .2  .1 ];

clf;
plot(elem,grap,'k');
axis([min(elem) max(elem) -.02 1.02]);
grid on;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% file virgiu.m
% virata giu ( visualizzare con Matlab )

dimU = 21;
elem = [ -1 -.9 -.8 -.7 -.6 -.5 -.4 -.3 -.2 -.1  0   .1  .2  .3  .4  .5  .6  .7  .8  .9   1 ];
grap = [  1   1   1  .9  .8  .7  .6  .5  .4  .3  .2  .1   0   0   0   0   0   0   0   0   0 ];
clf;
plot(elem,grap,'k');
axis([min(elem) max(elem) -.02 1.02]);
grid on;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% file virnessu.m
% virata nessuna ( visualizzare con Matlab )

dimU = 21;
elem = [ -1 -.9 -.8 -.7 -.6 -.5 -.4 -.3 -.2 -.1   0  .1  .2  .3  .4  .5  .6  .7  .8  .9   1 ];
grap = [ .1  .2  .3  .4  .5  .6  .7  .8  .9   1   1   1  .9  .8  .7  .6  .5  .4  .3  .2  .1 ];
clf;
plot(elem,grap,'k');
axis([min(elem) max(elem) -.02 1.02]);
grid on;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% file virsu.m
% virata su ( visualizzare con Matlab )

dimU = 21;
elem = [ -1 -.9 -.8 -.7 -.6 -.5 -.4 -.3 -.2 -.1   0  .1  .2  .3  .4  .5  .6  .7  .8  .9  1 ];
grap = [  0   0   0   0   0   0   0   0   0  .1  .2  .3  .4  .5  .6  .7  .8  .9   1   1  1 ];

clf;
plot(elem,grap,'k');
axis([min(elem) max(elem) -.02 1.02]);
grid on;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% file percorso.m  ( visualizzare con Matlab )
height = 0.2;
maxstep = 17;
step = [1:1:maxstep];
down = [ 0 0.2 0.3 0.5 0.6 0.65 0.63 0.57 0.53 0.55 0.65 0.74 0.76 0.73 0.65 0.6
0.5 ];
top = down + height;
traett = [ 0.1 0.3 0.4 0.6 0.709502 0.759502 0.736684 0.676684 0.616684 0.629502
0.729502 0.819502 0.892581 0.867804 0.787804 0.707803 0.614307 ];
clf;
plot(step,down,'k',step,top,'k',step,traett,':k');
axis([min(step)-.1 max(step)+.1  -0.1  max(top)+.1]);
```

5 BIBLIOGRAFIA

[Lazzerini, 2001] Beatrice Lazerini:: *Introduzione agli Insiemi Fuzzy e alla Logica Fuzzy*, Dip. Ing. Informazione – Pisa.

[Appunti, 2001] Appunti del Corso di Ingegneria della Conoscenza e Sistemi Esperti, Pisa 2001.

[Dorbolò-Frosini-Lazzerini, 2000] Daniela Dorbolò, Graziano Frosini, Beatrice Lazzerini: *Programmazione a oggetti con riferimento al C++,* FrancoAngeli – Milano 2000.

[Domenici-Frosini, 1996] Andrea Domenici, Graziano Frosini: *Introduzione alla programmazione ed elementi di strutture dati con il linguaggio C++*, FrancoAngeli – Milano 1996.

[Stroustrup, 2000] Bjarne Stroustrup: *C++, linguaggio, libreria standard, principi di programmazione*, Addison-Wesley – Milano 2000.

[Domenici, 1998] Andrea Domenici: *Appunti per le lezioni di Ingegneria del Software*, Pisa 1998.

[UML v1.3, 1999] AA.VV.: *UML Notation Guide, 1999*