Complementi di Programmazione Matlab

Array Multidimensionali

Grafici 2D e 3D

Algoritmi genetici

Array

- Prima di introdurre gli array multidimensionali alcune funzioni che danno informazioni su di essi:

`length(matrice)`: restituisce il numero di elementi della dimensione con più elementi

`ndims(matrice)`: restituisce il numero di dimensioni della matrice

`numel(matrice)`: restituisce il numero degli elementi della matrice

`size(matrice)`: restituisce il numero di elementi di ogni dimensione sottoforma di vettore

Array multidimensionali

- Gli array multidimensionali possono essere creati
  - Estendendo un array esistente
  - Utilizzando le funzioni di libreria
  - Per concatenazione

Array multidimensionali

Estendendo un array esistente

Partiamo dall'array 2x3

```
A=[1 2 3; 4 5 6]
```

Con l'istruzione

```
A(:,:,2) = [7 8 9; 10 11 12]
```

Si aggiunge una ulteriore dimensione all'array

che diviene 2 x 3 x 2

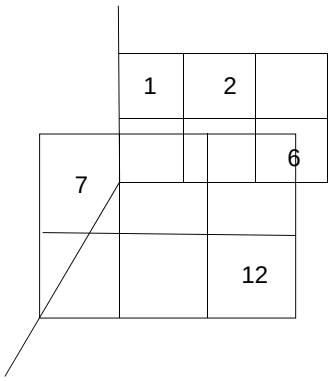
Per aggiungere una ulteriore dimensione con tutti gli elementi dello stesso valore si può usare:

```
A(:,:,:,4) = -10
```

Ottenendo un array 2 x 3 x 2 x 4

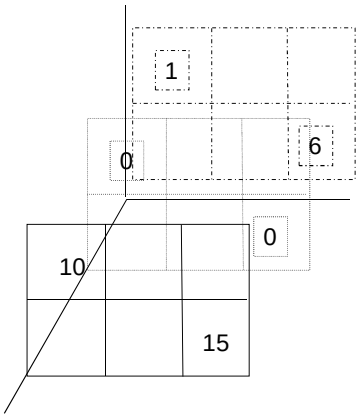
Array multidimensionali

```
A=[1 2 3; 4 5 6]
A(:,:,2) =
      [7 8 9;
       10 11 12]
```



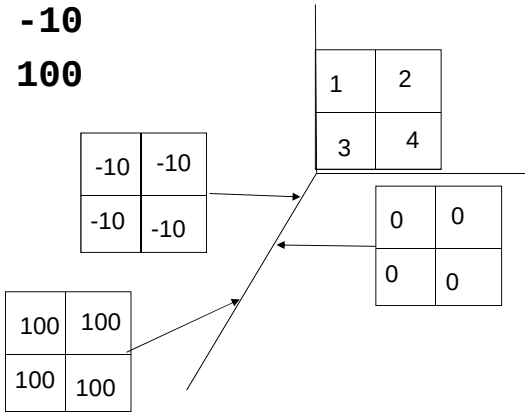
Array multidimensionali

```
A=[1 2 3; 4 5 6]
A(:,:,3) =
    [10 11 12;
     13 14 15]
```



Array multidimensionali

- **B = [1 2; 3 4]**
- **B(:,:,2) = -10**
- **B(:,:,4) = 100**



Array multidimensionali

- **C=[10 11;**
  12 13];
- **C(:,:,1) = 7**

```
C =
     7     7
     7     7
```

- **C=[10 11;**
  12 13];
- **C(:,:,2) = 7**

```
C(:,:,1) =
    10    11
    12    13

C(:,:,2) =
     7     7
     7     7
```

Array multidimensionali

- `C=[10 11;`
  `12 13];`
- `C(:,:,:,2) = 10`

```
C(:,:,1,1) =
    10    11
    12    13
C(:,:,1,2) =
    10    10
    10    10
```

- `size (C)`

```
ans =
     2   2   1   2
```

- `C=[10 11;  12 13];`
- `C(:,:,2,2) = 10`

```
C(:,:,1,1) =
    10    11
    12    13
C(:,:,2,1) =
     0     0
     0     0
C(:,:,1,2) =
     0     0
     0     0
C(:,:,2,2) =
    10    10
    10    10
```

```
size(C)
ans = 2   2   2   2
```


Array multidimensionali

Usando le funzioni di libreria

- **`rand (dim1, dim2, ...,dimN)`**
  - Restituisce un array **`dim1 x dim2 x ... X dimN`** di elementi generati casualmente secondo una uniforme
- **`randn(dim1, dim2, ...,dimN)`**
  - Restituisce un array **`dim1 x dim2 x ... X dimN`** di elementi generati casualmente secondo una Normale standardizzata
- **`repmat(valore,[dim1,dim2, ..., dimN])`**
  - Restituisce un array **`dim1 x dim2 x ... X dimN`** riempita con il valore **`val`**.

Array multidimensionali

Per concatenazione

- **`cat(dim,Matr1, Matr2, ...)`**
  - Aggiunge gli array **`Matr1`**, **`Matr2`**, ... alla dimensione **`dim`**.

Esempio

```
A = [3 4]
B = [5 6]
x = cat(3, A ,B)
y = cat(3,A',B')
```


Array: indicizzazione lineare

- Gli array possono essere anche indirizzati con un solo indice (indicizzazione lineare)
  - Gli array sono immagazzinati linearmente in memoria colonna dopo colonna (4 x 3)

```
A=[ 1  2  3;
    4  5  6;          (2,1)=2
    7  8  9;
    10 11 12]         (4,2)=8
```

  - Viene immagazzinato come vettore lineare:

```
1  4  7  10   2  5  8  11   3  6  9  12
```

  - Quindi l'elemento (i,j) di un array d1xd2 si può indicizzare mediante la posizione nel vettore lineare usando la formula:

```
(i,j) <-> (j-1) * d1 + i
```

Array: indicizzazione lineare

– Lo stesso discorso vale per le matrici n-d
- Gli array sono immagazzinate linearmente in memoria colonna dopo colonna; pagina dopo pagina
- La posizione (s_1, s_2, …, s_n) in un array $d_1 x d_2 x \ldots x d_n$ è data dalla seguente formula:

$$\sum_{i=2}^{n} (s_i - 1) \cdot \prod_{j=1}^{i-1} d_j + s_1$$

Array: indicizzazione lineare

```
A = [1 2; 3 4];
A = cat(3,A,-A);
```

```
1 2
3 4
-1 -2
-3 -4
```

– Viene immagazzinata come vettore lineare:

```
1 3  2 4  -1 -3  -2 -4
Pos:(2,1,2)
Matr: 2 x 2 x 2
(1–1)*2 + (2-1)*2*2 + 2 = 6
```

Grafici

- MatLab mette a disposizione due famiglie di comandi per la creazione di
  - grafici di funzioni bidimensionali (2D)
  - grafici di funzioni tridimensionali (3D)
    - Il grafico può essere visualizzato per
      - Punti
      - A fili (mesh)
      - Superficie (surf)

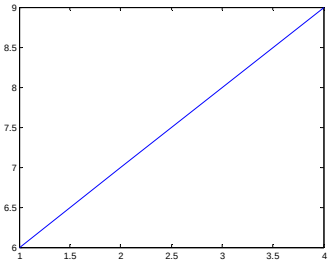
L'Help in linea fornisce ulteriori dettagli. Questa è solo una breve Introduzione.

Grafici 2D

- Si usa la funzione plot
  - **`plot(y)`**
    - Produce un grafico dei punti del vettore ***y*** che sono posti in ordinata, l'ascissa è data dalla posizione
  - **`plot(x,y)`**
    - Produce un grafico dei punti in *y* in funzione dei punti in *x*
  - **`plot(x1,y1, x2, y2)`**
    - produce grafici multipli fornendo più <u>coppie</u> di vettori di coordinate

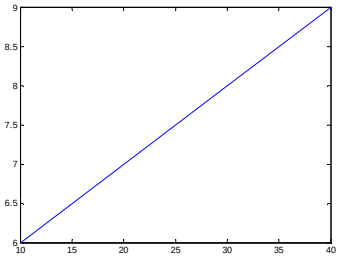
Grafici 2D

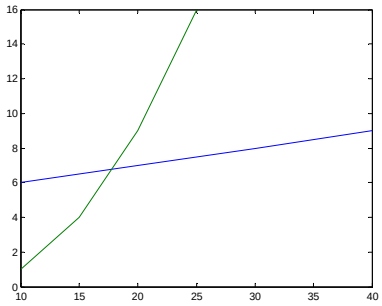
plot(Y1)



```
X1=[10 20 30 40];
Y1=[6 7 8 9];
X2=[10,15 20 ,25]
Y2=[1 4 9 16];
```

plot(X1,Y1)





plot(X1,Y1,X2,Y2)

Grafici 2D

- Il comando **`plot(X,Y,`*`tipoLinea`*`)`** permette di specificare il parametro *`tipoLinea`* (sequenza di caratteri speciali fra apici):
  - Colore della curva

    cyan **m**agenta **y**ellow **r**ed **g**reen **b**lu **w**hite blac**k**
  - Stile della curva

    - (continua) - - (tratteggiata) **:** (a puntini) **-.** (punti e linea)
  - Marcatori per i punti

    + o * x ^ v < > diamond **s**quare **p**entagram **h**exagram

Grafici 2D

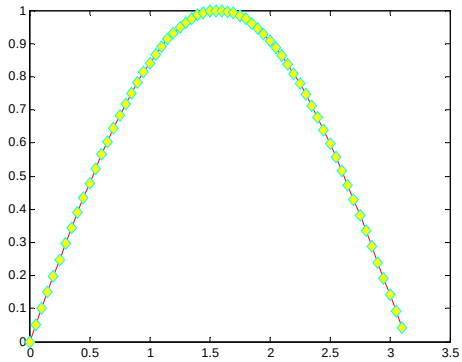
– Possono essere specificati anche altre caratteristiche mediante la notazione

`caratteristica, valore`

- **`'LineWidth',dim`**
  - larghezza linea grafico in punti
- **`'MarkerEdgeColor',colore`**
  - Colore marcatore
- **`'MarkerFaceColor', colore`**
  - Colore riempimento del marcatore
- **`'MarkerSize', dim`**
  - Grandezza in punti del marcatore

Grafici 2D

```
x = 0:0.05:pi
y = sin(x)
plot(x,y,'r-d','LineWidth', 1,'MarkerEdgeColor', ...
   'c','MarkerFaceColor', 'y','MarkerSize', 8)
```



Grafici 2D

- Ogni comando **`plot`** cancella il grafico precedente
  - Il comando **`hold on`** permette di creare un nuovo grafico che si sovrappone al precedente
- Esiste la possibilità di usare l'editor dei grafici

Grafici 2D

- È possibile definire l'intervallo degli assi in modo da cambiare quello automatico:
  - **`axis([`*`xmin xmax ymin ymax`*`])`**
- Per intitolare assi e grafici si usano i seguenti comandi:
  - **`xlabel(`*`'stringa titolo asse X'`*`)`**
  - **`ylabel(`*`'stringa titolo asse Y'`*`)`**
  - **`title(`*`'stringa titolo grafico'`*`)`**
  - **`text(`*`xpos,ypos,'posizione e testo legenda'`*`)`**

Grafici 2D

In alternativa alla funzione **`plot()`** esiste

`ezplot(fun1, …, funN, xMin,Xmax, yMin, yMax)`

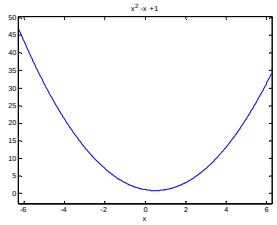
- Disegna il grafico delle funzioni fun1, funN, usando gli intervalli [xMin , …,xMax] e [yMin, …, yMax]
- Solo la prima funzione è obbligatoria
- Se non specificato l'intervallo delle x varia fra -2π e 2 π
- Se non specificato l'intervallo delle y si adatta al massimo e al minimo della funzione

Grafici 2D

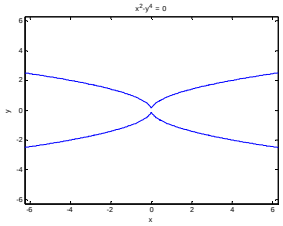
Passaggio delle funzioni per `ezplot`

- La funzione o le funzioni argomento di `ezplot` possono essere passate in diversi modi
  - Implicita, la funzione è compresa fra apici ('):
    - » **`ezplot('x^2 -x +1')`**
  - Richiamando una funzione mediante il puntatore alla funzione
  - Usando una funzione anonima

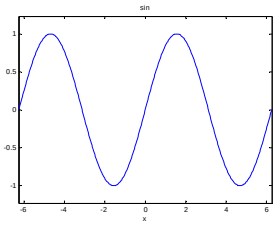
Grafica 2D



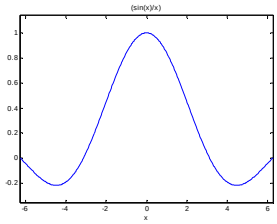
ezplot('x^2 -x +1')



ezplot('x^2-y^4')



ezplot(@sin)



ezplot(@(x)(sin(x)/x))

Grafici 3D

plot3(X,Y,Z)

- disegna la funzione a partire dai vettori X, Y, Z (uno per dimensione) come per i grafici 2D
  - » in pratica unisce punto per punto le coordinate fornite con i 3 vettori

contour(X,Y,Z)

- crea un grafico a curve di livello

contourf(X,Y,Z)

- crea un grafico a curve di livello piene

mesh(X,Y,Z)

- crea un grafico a griglia (wired)

surf(X,Y,Z)

- crea un grafico di superficie

surfl(X,Y,Z)

- crea un grafico di superficie con ombreggiatura

Grafici 3D

Funzioni collegate

`linspace(ini,fin,punti)`

– crea un vettore di punti elementi equispaziati a partire da ini fino a fin

`meshgrid(matX, matY)`

– crea due matrici X ed Y replicando i valori di matX e matY di dimensione **`dim(matY) x dim(matX)`**

`meshgrid(matX)`

– crea due vettori **`dim(matX) x dim(matX)`** X ed Y con gli stessi valori di matX

Grafici 3D

Esempio uso meshgrid

```
matX = [1 2 3]

matX =

   1   2   3

>> matY = [10 20 30 40]

matY =

  10   20   30   40
```

```
>> [X Y] = meshgrid(matX, matY)

X =

   1   2   3
   1   2   3
   1   2   3
   1   2   3

Y =

  10  10  10
  20  20  20
  30  30  30
  40  40  40
```

Grafici 3D

Uso di meshgrid

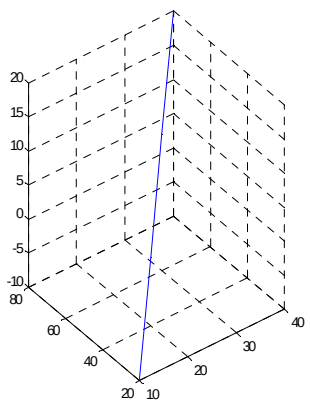
```
>> matX = -1:.5:1

matX =

   -1.0000   -0.5000         0    0.5000    1.0000

>> meshgrid(matX)

ans =

   -1.0000   -0.5000         0    0.5000    1.0000
   -1.0000   -0.5000         0    0.5000    1.0000
   -1.0000   -0.5000         0    0.5000    1.0000
   -1.0000   -0.5000         0    0.5000    1.0000
   -1.0000   -0.5000         0    0.5000    1.0000
```


Grafici 3D

```
X=[10 20 30 40];
Y=[20 40 60 80];
Z=[-10 0 10 20];
plot3(X,Y,Z);
```

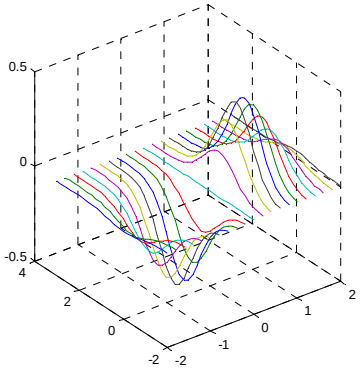
- Per poter evidenziare la griglia si devono far seguire i comandi precedenti da:

```
axis square;
grid on;
```



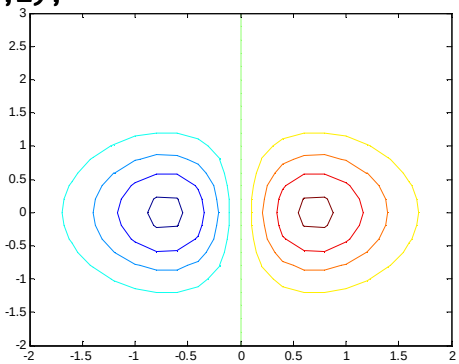
Grafici 3D

```
[X,Y] = meshgrid(-2:.2:2,-2:.2:3);
Z = X.*exp(-X.^2-Y.^2);
plot3(X,Y,Z); axis square; grid on;
```



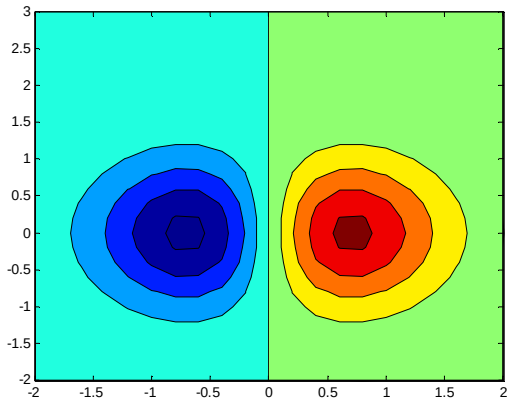
Grafici 3D

```
[X,Y] = meshgrid(-2:.2:2,-2:.2:3);
Z = X.*exp(-X.^2-Y.^2);
contour(X,Y,Z);
```



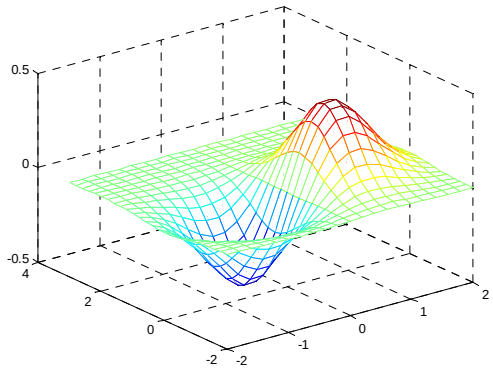
Grafici 3D

```
[X,Y] = meshgrid(-2:.2:2,-2:.2:3);
Z = X.*exp(-X.^2-Y.^2);
contourf(X,Y,Z);
```



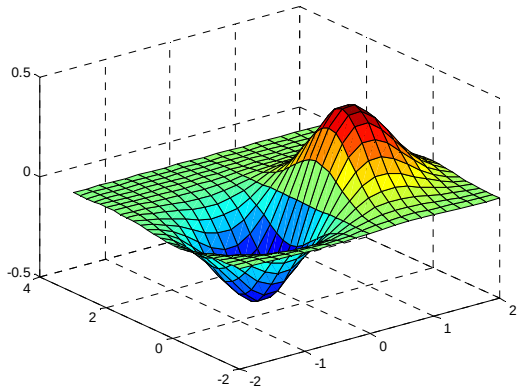
Grafici 3D

```
[X,Y] = meshgrid(-2:.2:2,-2:.2:3);
Z = X.*exp(-X.^2-Y.^2);
mesh(X,Y,Z);
```



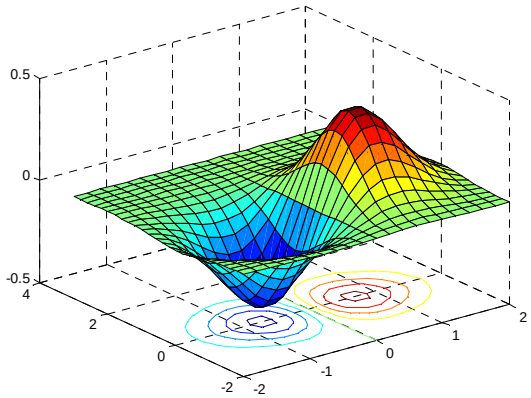
Grafici 3D

```
[X,Y] = meshgrid(-2:.2:2,-2:.2:3);
Z = X.*exp(-X.^2-Y.^2);
surf(X,Y,Z);
```



Grafici 3D

```
[X,Y] = meshgrid(-2:.2:2,-2:.2:3);
Z = X.*exp(-X.^2-Y.^2);
surfc(X,Y,Z);
```



Grafici 3D

Definiamo le seguenti matrici

```
[X Y] = meshgrid(-8:.5:8);
R = sqrt(X.^2 + Y.^2) + eps;
Z = sin(R) ./ R;
```

Disegnamo la figura in diversi modi

1. contour(X,Y,Z);
2. contourf(X,Y,Z);
3. plot3(X,Y,Z);
4. mesh(X,Y,Z);
5. surf(X,Y,Z);
6. surfl(X,Y,Z);

Richiamando di seguito l'istruzione
`shading interp`
il grafico è disegnato con sfumature di colore invece che a griglia

Algoritmi Genetici

Bibliografia (minima)

- J. H. Holland, algoritmi genetici, le Scienze 289 SET 92
  - introduce al problema
- R. L. Riolo La sopravvivenza dei bit piu' adatti, 289 SET 92
  - da qui ho tratto l'esempio
- D. Whitley, A Genetic Algorithm Tutorial
  - reperibile in Internet (introduzione teorica)

Algoritmi genetici

- Metodologia per risolvere i problemi ispirata ai meccanismi dell’evoluzione
- Utilizzata per risolvere problemi di:
  - Ottimizzazione
  - Creazione automatica di programmi
  - Apprendimento delle macchine
  - Previsioni dei mercati
  - …

Algoritmi Genetici

Quando si usano:

- Problemi con uno spazio delle soluzioni troppo grande per
  - Ricerca esaustiva delle soluzioni
  - Ricerca casuale delle soluzioni
- Metodo debole
  - Il procedimento non è fortemente legato al problema da risolvere

Quando non si usano:

- Esiste una soluzione specifica ed ottimizzata per il problema

Algoritmi Genetici

- Componenti
  - Popolazione di *cromosomi*;
  - Regola per selezionare i cromosomi;
  - Regola per generare nuovi cromosomi;
  - Regola per creare delle *mutazioni*.

Algoritmi Genetici

Cromosomi

- possibili soluzioni del problema
- In pratica sono i punti di uno spazio (discreto) di soluzioni opportunamente codificati
  - La codifica è di solito binaria
  - Occorre creare una corrispondenza fra la possibile soluzione e la codifica binaria
    - » Problema delle codifiche vuote
  - La lunghezza della codifica influenza la precisione della soluzione

Algoritmi Genetici

Funzione di valutazione

- Procedimento che assegna ad una soluzione un valore
  - Permette il confronto fra soluzioni
  - Deve essere veloce

Algoritmi Genetici

Funzionamento

- A partire da un insieme di possibili soluzioni si creano delle nuove soluzioni
  - Mescolando quelle esistenti per crearne delle nuove
    - » Privilegiando le migliori => funzione di fitness
  - Modificando casualmente alcune delle soluzioni create

Algoritmi Genetici

Algoritmo Genetico Fondamentale (CGA)

1. Generare la popolazione iniziale di N cromosomi di L bit
2. Calcolare la fitness F(x) di ogni cromosoma x
   - F(x)=f(x)/E(f(x)) f(x): valore della funzione di adattamento per x
3. Generare N nuovi figli come segue:
   a) Selezionare un paio di cromosomi padri (estrazione con rigenerazione)
   b) Tagliare con probabilità $\mathbf{p_c}$ (probabilità crossover) in un punto casuale i due genitori e generare i due nuovi figli
   c) Modificare con probabilità $\mathbf{p_m}$ (probabilità di mutazione) ciascun bit dei due nuovi figli
4. Rimpiazzare la popolazione con i nuovi N figli
5. Ripetere i passi da 2 a 5
   - Di solito si ripete dalle 50 alle 500 volte

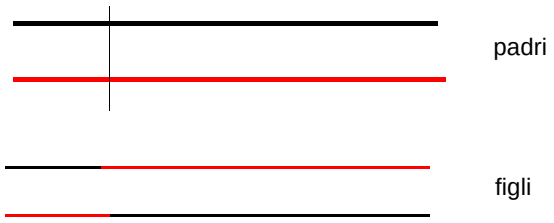
Generazione della popolazione Intermedia

Esempio: fitness

```
a 1001   f(a)=5  F(a)=5/4,5 = 1.1111
b 1110   f(b)=3  F(b)=3/4,5 = 0.6667
c 1011   f(c)=2  F(c)=2/4,5 = 0.4444
d 1100   f(d)=8  F(d)=8/4,5 = 1.7778
           E(f())=4,5
```

I valori di f() sono arbitrari

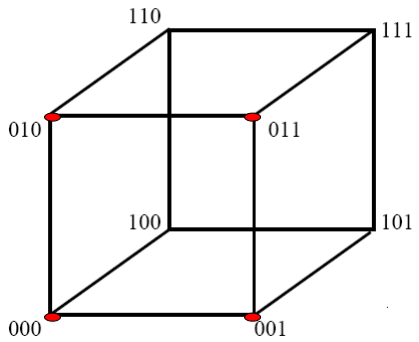
Esempio: crossing over



Algoritmi Genetici

- Codifica binaria
  - per ogni posizione 3 possibilità:
    - 0
    - 1
    - * (indifferente se è 0 oppure 1)
      - Le posizioni * definiscono uno schema (*schemata*) di un cromosoma
        - » Il numero di posizioni diverse da * in uno schema definisce il **grado** dello schema
  - Stringa lunga L
    - 3^L possibili piani di ricerca
    - Ogni punto appartiene a 3^L -1 possibili piani di ricerca

Algoritmi Genetici



- Lo schema **`0**`**
  - ordine 1
  - rappresenta i punti in rosso
  - le sue **istanze** sono
    - 000
    - 001
    - 010
    - 011

Algoritmi Genetici

Un'idea del funzionamento

Parallelismo implicito:
- Il grado di parallelismo dipende dal numero diverso di schemi iniziali
- Non dal numero di cromosomi iniziali

Ogni cromosoma della popolazione iniziale appartiene ad uno schema

- Alcuni cromosomi possono appartenere allo stesso schema
- La valutazione della fitness di un cromosoma è una stima della fitness media dello **schema** cui appartiene

La riproduzione (crossover+mutazione) porta a selezionare alcuni schemi piuttosto che altri

- Gli schemi migliori hanno una possibilità più alta di essere selezionati
- L'evoluzione porta a selezionare più campioni dagli schemi migliori migliorando l'accuratezza

Algoritmi Genetici

Schema della funzione che determina l'ottimo

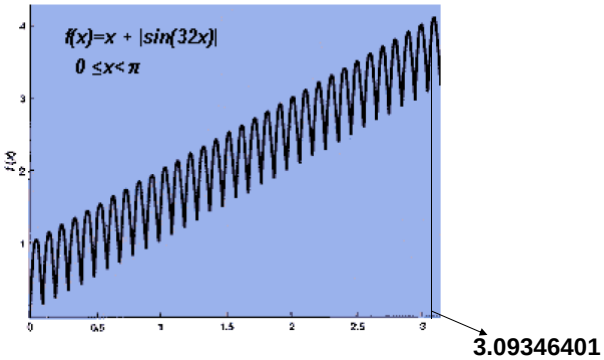
```
function [it, x, y]=CGA(L, P, G)
   popolazione =(rand(P,L) < 0.5);
   %vettore di numeri fra 0 ed 1
   for i=1:G
       %calcolo fitness
       %salvataggio migliore in it
       %scelta popolazione intermedia
       %crossover
       %mutazione
   end
end
```

L lunghezza in bit dei cromosomi
P numero di cromosomi (popolazione)
G numero di generazioni

Algoritmi Genetici

- Determinare il massimo della funzione

$$f(x)=x+|\sin(32x)| \quad 0 \le x \le \pi$$



Algoritmi Genetici

```
function [it, x, y]=CGA(L, P, G, pSc, pC, pMz)
% L lunghezza codifica
% P numerosita' popolazione
% G durata simulazione (generazioni)
% pSc probabilta' scelta
% pC  probabilita' crossover
% pMz probabilità di mutazione per bit
%determinare massimo per x+|sin(32x)|
%con 0<=x<=pi
%da Riolo, R. L. (1992). Survival of the fittest bits.
% Sci. Am. July, 1992: 114-116

   popolazione =(rand(P,L) < 0.5);
   it=[];
   for i=1:G
       %calcolo fitness
       F = fitness(popolazione,L);

       %salvataggio migliore in it
       it = [it; decodifica(popolazione(find(max(F)==F,1),:),L) ];

       %scelta popolazione intermedia
       intermedia=scelta(popolazione,F,pSc);

       %crossover
       popolazione=crossover(intermedia,pC);

       %mutazione
       popolazione=muta(popolazione,pMz);
   end
   x = it(G,1);
   y = it(G,2);
end
```

Algoritmi Genetici

– Codifica dei dati
- Numero di bit L=16
  - 2^{16}=65536 possibili valori per x
- Ogni cromosoma (codifica) rappresenta un punto fra 0 e π separato dal successivo di $\pi/2^{16}$.
  - 0…0 equivale ad x=0 $\pi/2^{16} = 0$
  - 1…1 equivale ad x=$(2^{16}-1)\ \pi/2^{16} = \pi - \pi/2^{16}$
    - » Il punto π non possiamo rappresentarlo
  - In generale i punti sono dati da:

$$p\ \pi/2^{L} \quad \text{con} \quad p=0,\ldots,2^{L}-1$$

Decodifica dei cromosomi

- x = base2dec(num2str(char(cod+double('0'))),2)
- 0000 0001 ….. 1111
- Cod=1001
  - cod+double(‘0’) 49 48 48 49
  - char([49 48 48 49]) ‘1’ ‘0’ ‘0’ ‘1’
  - Num2str([‘1’ ‘0’ ‘0’ ‘1’]) = ‘1001’ {E’ una stringa!}
  - Base2dec(‘1001’,2) => 9

Decodifica dei Cromosomi

```
function [x,y]=decodifica(cod, L)
% dato un  cromosoma restituisce il valore di x ed y
% se richiamata con un solo argomento in uscita
% un vettore con x ed y

   %trasforma le codifiche in numero
   x = base2dec(num2str(char(cod+double('0'))),2);
   %valuta la funzione nei punti
   x = x *pi/2^L;
   y = x + abs(sin(32*x));

   if nargout == 1
       x=[x y];
   end
end
```

`num2str(char(cod+double('0')))`: Trasforma un vettore di numeri 0 ed 1 in una stringa di caratteri ‘0’ ed ‘1’

Fitness

```
function ft=fitness(P, L)
%calcola la fitness per ogni cromosoma

   [pNum,fz] = decodifica(P,L);
   %calcola la media e determina F
   ft=fz/mean(fz);
end
```

Selezione

```
popolazione(find(max(F)==F,1))
```

- `F valori di fitness`
- `find(max(F)==F,1)`
  - `La posizione del primo elemento con fitness massima`

Algoritmi Genetici

Scelta popolazione intermedia

- Usa un algoritmo di tipo torneo
  - Si scelgono a caso due cromosomi
  - Con probabilità **pSc** si sceglie quello con fitness più alta per essere aggiunto alla popolazione intermedia
- **pSc** è un valore elevato (0,75 ad esempio)
  - Sopravvivono i migliori ma i più deboli non vengo scartati del tutto
  - Se pSc=0,5 non c'è selezione
- In questa versione (quella originale) nella popolazione intermedia ci possono essere più copie degli stessi individui

Algoritmi Genetici

```
function inter=scelta(P,F,pSc)
%sceglie la popolazione per la riproduzione
%metodo del torneo di Riolo

   disUnif=linspace(0,1,length(P)+1); %cumulata Uniforme
   inpos=[];
   for i=1:length(P)
       x = rand;
       y = rand;
       pX = find(x < disUnif, 1) - 1;
       pY = find(y < disUnif, 1) - 1;
       r = rand;
       if (r < pSc) %migliore
           if F(pX) > F(pY)
               inpos=[inpos; pX];
           else
               inpos=[inpos; pY];
           end
       else
           if F(pX) > F(pY)
               inpos=[inpos; pY];
           else
               inpos=[inpos; pX];
           end
       end
   end
   inter = P(inpos,:); %prende righe selezionate
end
```


Algoritmi Genetici

Crossover

- Si scelgono coppie di popolazione e si applica il crossover con probabilità **pC** (ad esempio 0,75)
  - Quindi non su tutte le coppie si applica il crossover
  - Sulla coppia cui si deve applicare si sceglie un punto a caso e si scambiano le parti
- Se la fase precedente ha scelto casualmente in questa basta prendere le coppie di valori

- 100011 100 011 101010
- 101010 111001 010101
- 111001
- 010101
- TUTTI DISPARI PARI

- pTaglio = 3

Algoritmi Genetici

```
function popN=crossover(inter,pC)
%forma la nuova popolazione

  pari=inter(2:2:size(inter,1),:); %cromosomi pari
  dispari=inter(1:2:size(inter,1),:); %cromosomi dispari
  popN=[];
  unifPos = linspace(0,1,size(inter,2)+1); %lunghezza dei cromosomi

  for i=1:size(pari,1)
      if rand < pC %crossover
          pTaglio = find(rand <unifPos, 1) - 1; %punto taglio
          prePari = pari(i,1:pTaglio);
          postPari =pari(i,(pTaglio+1):size(pari,2));
          preDisp = dispari(i,1:pTaglio);
          postDisp =dispari(i,(pTaglio+1):size(dispari,2));
          popN=[popN; [prePari postDisp]; [preDisp postPari]];
      else
          popN=[popN; pari(i,:); dispari(i,:)];
      end
  end

end
```

Algoritmi Genetici

Mutazione

- Ad ogni bit della nuova popolazione si applica una mutazione con una probabilità pMz
- pMz è un valore basso (ad esempio 0,005)

Algoritmi Genetici

```
function popMt = muta(pop,pMt)
%muta la popolazione in accordo alla distribuzione pMt

  mutaz = rand(size(pop,1), size(pop,2)) < pMt;

  popMt = mod(pop+mutaz,2);
end
```

- 0 1 1 0 0 1
- 1 0 0 1 0 1

popolazione

- 0 0 0 1 0 0
- 1 0 0 0 0 0

mutaz

- 0 1 1 1 0 1
- 2 0 0 1 0 1

pop+mutaz

- 0 1 1 1 0 1
- 0 0 0 1 0 1

mod(pop+mutaz,2)

Funzioni manipolazione stringhe

Funzioni usate nella risoluzione (CGA.m)

`base2dec(`*`stringa,base`*`)`

- Trasforma il valore in *`stringa`* che rappresenta un numero in base *`base`* in base 10

`num2str(`*`vettoreDiNumeri`*`)`

- Trasforma *`vettoreDiNumeri`* in una stringa di caratteri contenente i numeri espressi come sequenze di caratteri
- Usando **`num2str(`*`vettoreDiNumeri,format`*`)`** è possibile formattare i numeri come in C

`find(`*`vt,pos`*`)`

- Restituisce i primi `pos` elementi del vettore di elementi restituito da *`find(vt)`*

Funzioni manipolazione stringhe

`double(car)`

- Trasforma un carattere (***`car`***) nel suo valore nella rappresentazione interna (ASCII)
- I caratteri sono dei numeri da 0 a 127 secondo la codifica ASCII

`char(vettoreDiNumeri)`

- Converte un *`vettoreDiNumeri`* i cui elementi rappresentano codifiche di caratteri in una stringa di caratteri
- Sono rappresentabili solo i caratteri da 32 (spazio) a 126 (~)
- **`char(1:126)`** fornisce l'elenco di tutti i cartteri ASCII

Funzioni manipolazione stringhe

```
> y=num2str([36, 48, 90])

      y = 36  48  90

> y=num2str([36, 48, 90],'%2d')

   y =364890

> z=char([36, 48, 90])

  z = $0Z
```

```
> base2dec('110',2)

   ans = 6

> find([2 3 5 6 12] > 4,2)

    ans =  3     4

> find([2 3 5 6 12] > 4,4)

    ans =  3     4     5
```