Laboratorio di Informatica T

Funzioni come Argomento
e Funzioni Anonime

Oscillatore di Van Der Pol (di Nuovo)

Consideriamo ancora l'oscillatore (discretizzato) di Van Der Pol:

$$x^{(t+1)} = x^{(t)} + hy^{(t)}$$
$$y^{(t+1)} = y^{(t)} + h\left(\mu(1 - x^{(t)2})y^{(t)} - x^{(t)}\right)$$

...Ed il suo codice di simulazione:

```
X = [];
T = 1:1000; % Istanti di tempo
xc = [x0, y0]; % Stato iniziale
for t = T
    X(t, :) = xc; % Salvo lo stato corrente
    xc = f(xc, h, mu); % Genero lo stato futuro
end
```

- Lo abbiamo usato in tutti gli esercizi seguenti, e non è buona cosa

Funzioni come Argomenti

Per essere "puliti", dovremmo definire una funzione

Per prima cosa, incapsuliamo il codice in una nuova funzione:

```
function X = simulate(...)
  X = [];
  T = 1:1000; % Istanti di tempo
  xc = [x0, y0]; % Stato iniziale
  for t = T
      X(t, :) = xc; % Salvo lo stato corrente
      xc = f(xc, h, mu); % Genero lo stato futuro
  end
end
```

Funzioni come Argomenti

Per essere "puliti", dovremmo definire una funzione

Quindi, identifichiamo i parametri:

```
function X = simulate(..., T, ...)
  X = [];
  xc = [x0, y0]; % Stato iniziale
  for t = T
      X(t, :) = xc; % Salvo lo stato corrente
      xc = f(xc, h, mu); % Genero lo stato futuro
  end
end
```

- È bene che `T` sia un parametro...
- ...Così possiamo definirlo al momento della chiamata

Funzioni come Argomenti

Per essere "puliti", dovremmo definire una funzione

Quindi, identifichiamo i parametri:

```
function X = simulate(..., T, X0)
  X = [];
  xc = X0;
  for t = T
      X(t, :) = xc; % Salvo lo stato corrente
      xc = f(xc, h, mu); % Genero lo stato futuro
  end
end
```

- Lo stesso vale per lo stato iniziale...
- ...E ci conviene rappresentarlo con un vettore `X0`...
- ...Così funziona qualunque sia il numero di variabili di stato

Funzioni come Argomenti

Per essere "puliti", dovremmo definire una funzione

Quindi, identifichiamo i parametri:

```
function X = simulate(f, T, X0)
  X = [];
  xc = X0;
  for t = T
      X(t, :) = xc; % Salvo lo stato corrente
      xc = f(xc, h, mu); % Genero lo stato futuro
  end
end
```

- Il terzo parametro è la funzione di transizione da utilizzare
- Matlab permette di memorizzare una funzione in una variabile…
- …E di invocarla, utilizzando il nome della variabile

Funzioni come Argomenti

- Dobbiamo eliminare la dipendenza da `h` e `mu`...
- ...Perché sono specifici dell'oscillatore di Van Der Pol

Possiamo includerli nella definizione della funzione

Dentro `simulate` avremo:

```
function X = simulate(f, T, X0)
  X = [];
  xc = X0;
  for t = T
      X(t, :) = xc;
      xc = f(xc, t); % Passo solo lo stato
  end
end
```

- Passo anche il tempo `t` (è utile per alcuni sistemi)

Funzioni come Argomenti

- Dobbiamo eliminare la dipendenza da `h` e `mu`...
- ...Perché sono specifici dell'oscillatore di Van Der Pol

Possiamo includerli nella definizione della funzione

A parte, potremmo per esempio definire:

```
function xf = vdp(xc, t)
    h = 0.1; % Hard-coded nella funzione
    mu = 1; % Hard-coded nella funzione
    x = X(1);
    y = X(2);
    Xf(1) = x + h * y;
    Xf(2) = y + h * (mu * (1 - x^2)*y - x);
end
```

Funzioni come Argomenti

Nel complesso, potremmo avere:

```
function es_vdp()
    T = 1:300;
    X0 = [0.5, 0.5];
    X = simulate(@vdp, T, X0); % Funzione come valore
end

function xf = vdp(xc, t)
    ...
end
```

Quando Matlab incontra un nome di funzione:

- Se non c'è `@`, cerca di invocarla...
- ...Ma se c'è `@`, allora passa la funzione come valore

Funzioni come Argomenti

C'è ancora qualche punto debole:

```
function es_vdp()
    ...
end

function xf = vdp(xc, t)
    h = 0.1;
    mu = 1;
    ...
end
```

- `t` è un parametro in `vdp` solo per compatibilità con `simulate`...
- ...La funzione non lo usa! Questo riduce la leggibilità
- `h` e `mu` sono hard-coded nella funzione `vdp`...
- ...Se vogliamo provare valori diversi, dobbiamo ridefinire la funzione!

Funzioni Anonime

Possiamo risolvere i due problemi con una funzione anonima

Si tratta di un modo alternativo per definire una funzione:

```
@(<p1>, <p2>, ...) <espressone>
```

Il costrutto restituisce una funzione senza nome:

- `<p1>`, `<p2>`... sono i parametri formali
- La funzione restituisce il valore di `<espressione>`

Si può memorizzare la funzione in una variabile, e.g.:

```
f = @(x) x^2;
```

- Inserisce nella variabile `f`...
- ...Una funzione che, quando invocata, restituisce `x^2`

Funzioni Anonime e Funzioni Normali

Tipicamente, useremo una funzione anonima in questo modo:

```
function es_vdp()
    T = 1:300;
    X0 = [0.5, 0.5];
    X = simulate(@vdp, T, X0);
end

function xf = vdp(xc, t)
    h = 0.1;
    mu = 1;
    ...
end
```

Funzioni Anonime e Funzioni Normali

Tipicamente, useremo una funzione anonima in questo modo:

```
function es_vdp()
    h = 0.1; % Spostato
    mu = 1; % Spostato
    T = 1:300;
    X0 = [0.5, 0.5];
    X = simulate(@vdp, T, X0);
end

function xf = vdp(xc, h, mu) % Parametri cambiati
    ...
end
```

- Ridefiniamo **`vdp`** in modo che riceva solo i parametri appropriati

Funzioni Anonime e Funzioni Normali

Tipicamente, useremo una funzione anonima in questo modo:

```
function es_vdp()
    h = 0.1; % Spostato
    mu = 1; % Spostato
    T = 1:300;
    X0 = [0.5, 0.5];
    f = @(x, t) ...; % Parametri per "simulate"
    X = simulate(@vdp, T, X0);
end

function xf = vdp(xc, h, mu)
    ...
end
```

- Definiamo una funz. an. con l'interfaccia richiesta da **`simulate`**...

Funzioni Anonime e Funzioni Normali

Tipicamente, useremo una funzione anonima in questo modo:

```
function es_vdp()
    h = 0.1; % Spostato
    mu = 1; % Spostato
    T = 1:300;
    X0 = [0.5, 0.5];
    f = @(x, t) vdp(x, h, mu); % Chiama solo vdp!
    X = simulate(@vdp, T, X0);
end

function xf = vdp(xc, h, mu)
    ...
end
```

- ...E che non faccia altro che chiamare **vdp**

Funzioni Anonime e Funzioni Normali

Tipicamente, useremo una funzione anonima in questo modo:

```
function es_vdp()
    h = 0.1; % Spostato
    mu = 1; % Spostato
    T = 1:300;
    X0 = [0.5, 0.5];
    f = @(x, t) vdp(x, h, mu);
    X = simulate(f, T, X0); % f passata a simulate
end

function xf = vdp(xc, h, mu)
    ...
end
```

- La funzione è memorizzata nella variabile **`f`**, passata a **`simulate`**

Funzioni Anonime e Funzioni Normali

Ci sono ancora due cose da notare:

```
function es_vdp()
    h = 0.1;
    mu = 1;
    T = 1:300;
    X0 = [0.5, 0.5];
    f = @(x, t) vdp(x, h, mu);
    X = simulate(f, T, X0);
end
```

Per le funzioni memorizzate dentro variabili non serve `@`

- Matlab le invoca solo se ci sono le parentesi
  - E.g. `f([0.5, 0.5], 1)` nel nostro caso
- Altrimenti, le passa come valore

Funzioni Anonime e Funzioni Normali

Ci sono ancora due cose da notare:

```
function es_vdp()
    h = 0.1; % <-- Questo!
    mu = 1; % <-- Questo!
    T = 1:300;
    X0 = [0.5, 0.5];
    f = @(x, t) vdp(x, h, mu);
    X = simulate(f, T, X0);
end
```

Le funzioni anonime hanno accesso all'ambiente del chiamante:

- Quando `f` viene eseguita, invoca `vdp(x, h, mu)`
- `x` è uno dei parametri di `f`, ma `h` e `mu` no
- Matlab li cerca nell'ambiente in cui `f` è definita!

Un Pattern

Useremo molto spesso le funzioni anonime in questo modo:

```
function principale()
    ...
    f = @(x, t) ftrans(x, ...)
    X = simulate(f, ...)
    ...
end

function xf = ftrans(x, ...)
    ...
end
```

- Così **`ftrans`** ha solo i suoi parametri "naturali"...
- La funzione anonima **`f`** la "avvolge" (fa da wrapper)...
- ...Ed espone l'interfaccia richiesta da **`simulate`**

Laboratorio di Informatica T

Traiettoria di un Sistema Dinamico

Traiettoria di un Sistema Dinamico

- Se un sistema dinamico ha uno stato non-scalare...
- ...Se ne può disegnare la traiettoria

Si chiama traiettoria di un sistema dinamico la curva parametrica definita dai valori che il suo stato assume nel tempo

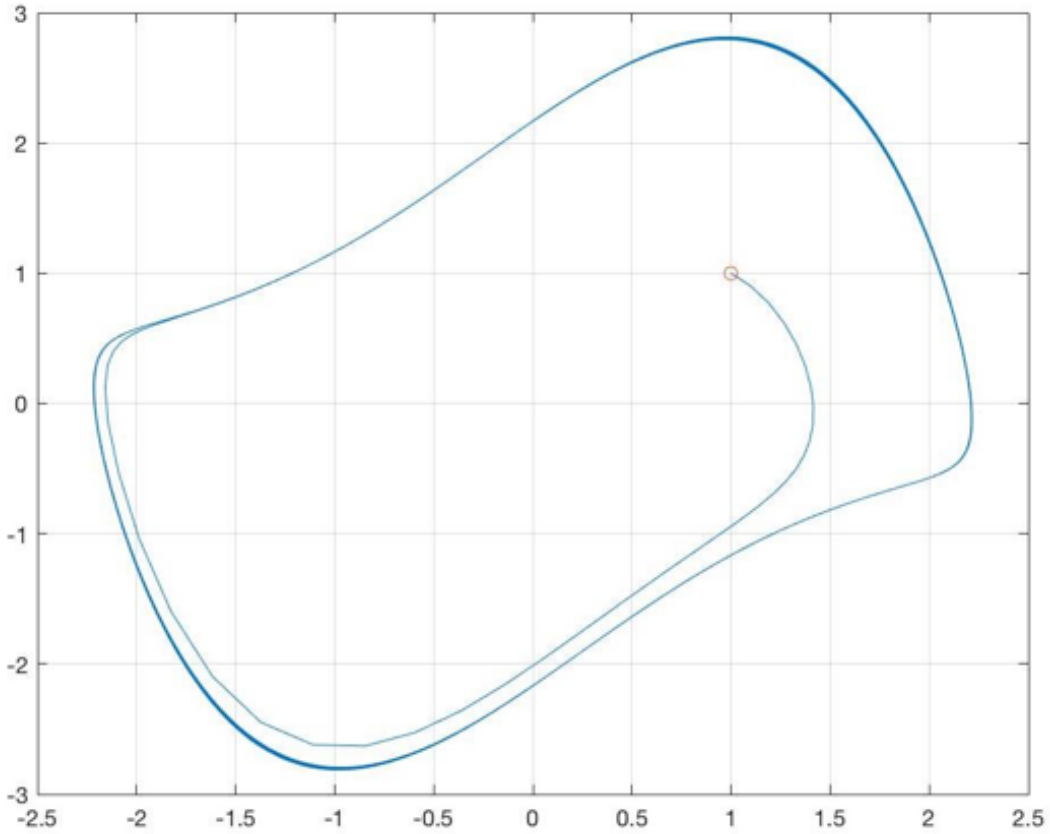
Per l'oscillatore di Van Der Pol abbiamo:

$$x^{(t+1)} = x^{(t)} + hy^{(t)}$$
$$y^{(t+1)} = y^{(t)} + h\left(\mu(1 - x^{(t)2})y^{(t)} - x^{(t)}\right)$$

- Una volta simulato l'andamento dello stato...
- ...Dobbiamo solo disegnare y rispetto ad x

Traiettoria di un Sistema Dinamico

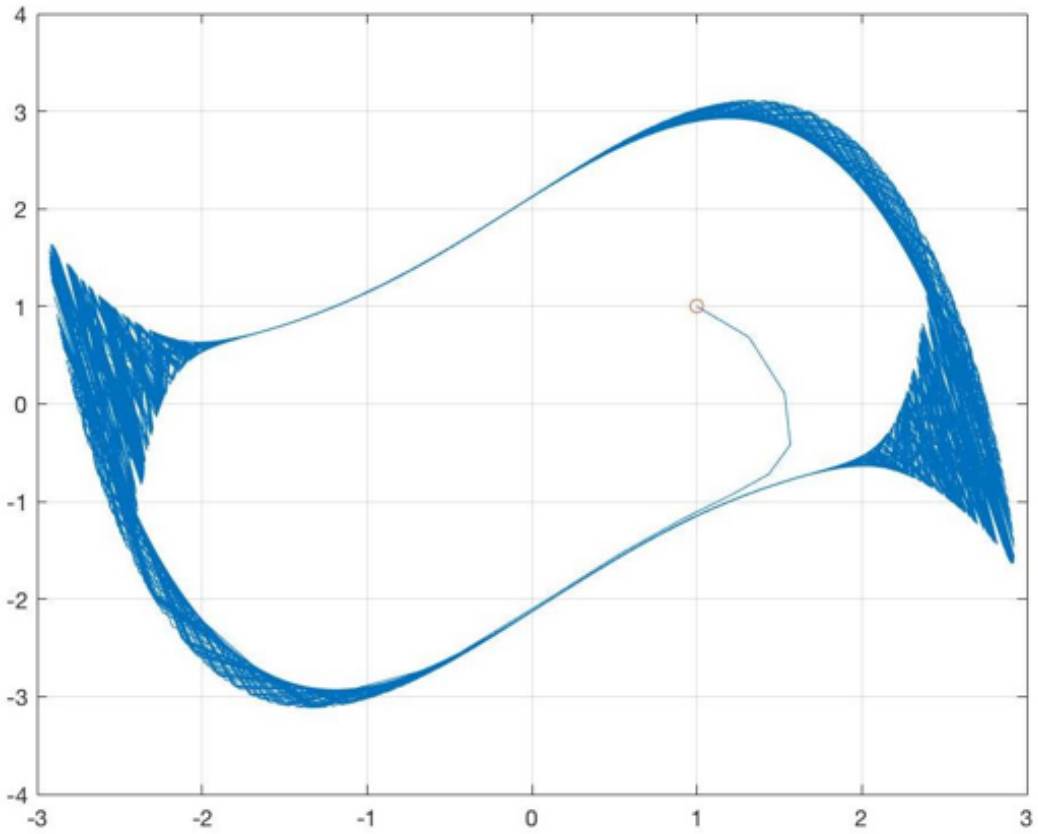
Per `h` sufficientemente piccolo, avremo qualcosa del genere:



- Si nota bene il ciclo limite

Traiettoria di un Sistema Dinamico

Se `h` cresce, il comportamento può diventare caotico:



- È "quasi periodico", ma non si assesta mai su un vero ciclo

Laboratorio di Informatica T

Esercizio: Oscillatore di Van Der Pol Forzato

Oscillatore di Van Der Pol Forzato

Consideriamo una variante dell'oscillatore di Van Der Pol

In particolare, vediamo la versione discretizza a "forzata":

$$x^{(t+1)} = x^{(t)} + hy^{(t)}$$
$$y^{(t+1)} = y^{(t)} + h\left(\mu(1 - x^{(t)2})y^{(t)} - x^{(t)} - A\sin(\omega ht)\right)$$

Il termine $A\sin(\omega ht)$ si dice guida (driver):

- A è l'ampiezza dell'oscillazione guida
- ω è la sua frequenza
- t è il tempo
- h compare perché il tempo è discretizzato

Il valore della guida non dipende dallo stato, ma solo dal tempo

Oscillatore di Van Der Pol Forzato

Partendo dal file `es_forced_vdp.m` nello start-kit:

- Provate ad eseguirlo (contiene il codice per l'oscillatore non forzato)
- Aggiungete la funzione di transizione per l'oscillatore forzato:

```
function Xf = forced_vdp(X, t, omega, A, h, mu)
```

- Aggiungete il codice di simulazione per l'oscillatore forzato

Quindi:

- Disegnate (su un'unica figura) i due andamenti di x nel tempo
  - Cosa notate?
- Disegnate le due traiettorie su due figure distinte
  - Cosa notate?

Laboratorio di Informatica T

Esercizio: Crescita Logistica

Esercizio: Crescita Logistica

Sia data una popolazione che segue il modello logistico

$$^{(t+1)} = r\,^{(t)}\left(1 - \frac{^{(t)}}{N}\right)$$

Dove:

- r indica il tasso di crescita (deve valere $r > 1$)
- N indica un valore di popolazione…
- …che, se raggiunto, ne causa il collasso immediato

Vogliamo studiare il comportamento del modello mediante simulazione

Esercizio: Crescita Logistica

A partire dal file `es_logi.m` nello start-kit:

Definite la funzione di transizione:

```
function xf = logi(xc, r, k)
```

- Che calcoli lo stato futuro **`xf`**
- ...A partire da quello corrente **`xc`**, e dai valori di **`r`** e **`k`**

Definite il codice di simulazione nella funzione principale:

- Lo stato iniziale è definito nel codice (e così l'intervallo di simulazione)
- Valori di partenza per **`r`** e **`k`** sono definiti nel codice
- Si disegni l'andamento dello stato nel tempo
- Si esplori cosa succede per vari valori di r, da 1.1 a 4.0

Laboratorio di Informatica T

Esercizio: Un Modello Preda-Predatore

Esercizio: Un Modello Preda-Predatore

I modelli preda-predatore:

- Rappresentano la co-evoluzione di due popolazioni antagoniste
- E.g. prede-predatori, ma anche reazioni chimiche che si ostacolano

Tipicamente ci sono due elementi nel vettore dello stato:

- Il numero di "prede" H
- Il numero di "predatori" P

Simulando il sistema:

- Si può studiare il comportamento delle due popolazioni
- Si può verificare se vi sia un equilibrio stabile

Esercizio: Un Modello Preda-Predatore

Consideriamo questo modello preda-predatore:

$$H^{(t+1)} = \overbrace{r\left(1 - \frac{H^{(t)}}{k}\right) H^{(t)}}^{\text{crescita logistica}} - \overbrace{s\, H^{(t)} P^{(t)}}^{\text{prede eliminate}}$$

$$P^{(t+1)} = \underbrace{u\, P^{(t)}}_{\text{calo in assenza di prede}} + v\underbrace{\left(s\, H^{(t)} P^{(t)}\right)}_{\text{prede eliminate}}$$

- k è la massima popolazione di prede sostenibile
- s è la frazione di $H^{(t)}$ che un predatore può "mangiare"
- $u < 1$ è il ritmo di scomparsa dei predatori in assenza di prede
- v è il "bonus riproduttivo" per ogni preda "mangiata"

Esercizio: Un Modello Preda-Predatore

Nello start-kit, partite dal file `es_logi_pp.m`

Implementare la funzione (ausiliaria) di transizione:

```
function xf = logi_pp(xc, r, k, s, u, v)
```

- HP: `xc(1)` sia il numero di prede e `xc(2)` quello di predatori
- I valori di r, k, s, u, v sono già definiti nel codice

Definite il codice di simulazione nella funzione principale:

- Lo stato iniziale è riportato nel codice
- Disegnate l'andamento delle due popolazioni nel tempo
- Disegnate la traiettoria nello spazio degli stati

Esercizio: Un Modello Preda-Predatore

Provate a fare delle variazioni!

Aumentate e diminuite (solo) s:

- Cosa vi aspettate che succeda? Cosa succede?

Aumentate e diminuite (solo) r:

- Cosa vi aspettate che succeda? Cosa succede?

Aumentate e diminuite (solo) u:

- Cosa vi aspettate che succeda? Cosa succede?

Riuscite a spiegare intuitivamente le ragioni?

Laboratorio di Informatica T

Esercizio: Random Surfer

Esercizio: Random Surfer

Consideriamo un semplice modello per un utente web

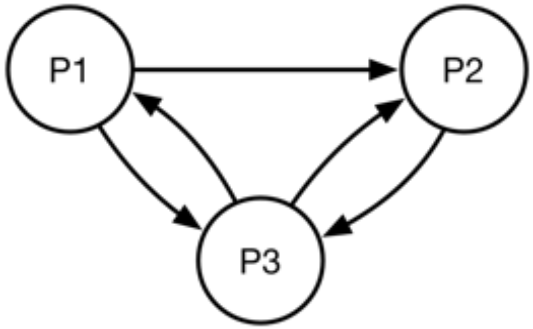
Il nostro utente si comporta così:

- Parte da una pagina scelta uniformemente a caso
- Ad ogni passo:
  - Con probabilità $1 - p$ l'utente clicca su un link a caso
  - Con probabilità p l'utente si stanca...
  - ...E ricomincia da una pagina a caso
- Se non ci sono link in uscita, l'utente resta fermo

Esercizio: Random Surfer

Consideriamo un semplice modello per un utente web

Possiamo descrivere la rete come un grafo, e.g.:



- I nodi (i.e. le palline) rappresentano pagine
- Gli archi (i.e.) le freccette rappresentano i link
- Supponiamo di avere n pagine
- Chiamiamo $P_{i,j}$ la probabilità di cliccare sul link da j a i
  - Quindi, $P_{i,j} = 1/\#\text{link in uscita da } j$

Esercizio: Random Surfer

Proviamo a tradurlo in equazioni

Per lo stato, facciamo una scelta un po' strana:

- Una variabile x_i per ogni pagina, con $x_i \in [0, 1]$
- x_i rappresenta la probabilità che l'utente sia sulla pagina i

All'inizio, tutte le pagine sono equiprobabili:

- Quindi, per tutte le pagine $x_i = 1/n$

Esercizio: Random Surfer

Proviamo a tradurlo in equazioni

Calcolare lo stato futuro vuol dire:

- A partire delle probabilità $x_i^{(t)}$ di essere sulla pagina i…
- …Calcolare le probabilità future $x_i^{(t+1)}$

Formalmente, la probabilità $x_i^{(t)}$ di essere su i al passo $t+1$:

$$x_i^{(t+1)} = \underbrace{p}_{\text{mi stanco}} \overbrace{\frac{1}{n}}^{\text{reset}} + \underbrace{(1-p)}_{\text{non mi stanco}} \sum_{j=1}^{n} \overbrace{x_j^{(t)}}^{\text{sono su } j} \underbrace{P_{i,j}}_{\text{link per } i}$$

- È data dalla probabilità di finire su i dopo essersi stancati…
- …Più la probabilità di finire su i da un'altra pagina

Esercizio: Random Surfer

Se ci fate caso, la funzione di transizione è lineare

Per i sistemi dinamici lineari, è comodo usare la notazione matriciale:

$$x^{(t+1)} = p\frac{1}{n} + (1-p)Px^{(t)}$$

- $x^{(t+1)}$ è il vettore (colonna) delle probabilità al tempo $t+1$
- $x^{(t)}$ è il vettore (colonna) delle probabilità al tempo t
- P è la matrice con le probabilità di transizione $P_{i,j}$

Vogliamo definire un simulatore per il modello appena visto

Esercizio: Random Surfer

A partire dal file `es_surfer.m` nello start-kit:

- Definite la funzione di transizione:

```
function xf = browse(xc, P, p, n)
```

  - Che implementi la transizione:

$$x^{(t+1)} = p\frac{1}{n} + (1-p)Px^{(t)}$$

  - La matrice `P` è già fornita nel file
- Definite il codice di simulazione nella funzione principale
- Disegnate l'andamento dello stato (x_0, x_1, x_2) nel tempo
  - Usata un'unica figura
  - Quali sono alla fine le pagine su cui è più probabile trovarsi?

Laboratorio di Informatica T

Esercizio: Previsioni del Tempo

Esercizio: Previsioni del Tempo

Vogliamo utilizzare un sistema dinamico per prevedere il tempo

Supponiamo che (sommariamente) valgano le regole seguenti:

- Se un giorno c'è bel tempo
  - Il giorno successivo sarà bello con il 90% di probabilità
  - Il giorno successivo sarà brutto con il 10% di probabilità
- Se un giorno c'è brutto tempo
  - Il giorno successivo sarà bello con il 50% di probabilità
  - Il giorno successivo sarà brutto con il 50% di probabilità

Esercizio: Previsioni del Tempo

Per lo stato, ragioniamo di nuovo in termini di probabilità:

- $x_g^{(t)}$ = probabilità che il tempo sia bello a t
- $x_b^{(t)}$ = probabilità che il tempo sia brutto a t

Per le transizioni, avremo:

$$x_g^{(t+1)} = x_g^{(t)} P_{g,g} + x_b^{(t)} P_{g,b}$$
$$x_b^{(t+1)} = x_b^{(t)} P_{b,b} + x_b^{(t)} P_{b,g}$$

- $P_{g,b}$ è la probabilità che il tempo sia brutto a t e bello a $t+1$, etc.
- Il sistema è di nuovo lineare. In forma matriciale possiamo scrivere:

$$x^{(t+1)} = P x^{(t)}$$

Esercizio: Previsioni del Tempo

Si parta dal file `es_weather.m` nello start-kit

Si definisca la funzione di transizione:

```
function xf = day(xc, P)
```

- Dove **`xc`** è lo stato corrente $(x_g^{(t)}, x_b^{(t)})$
- La matrice di transizione **`P`** è nota

Si sviluppi il codice di simulazione nella funzione principale:

- Si disegni l'andamento delle due probabilità (bel/brutto tempo)
  - Nel codice, si assume inizialmente che il primo giorno sia brutto
  - In un secondo momento, provate l'alternativa opposta

Laboratorio di Informatica T

Esercizio: Trovare Elemento in un Vettore

Esercizio: Trovare Elementi in un Vettore

Nel file di funzione `es_findidx.m`, definire la funzione ausiliaria:

```
function I = my_findidx(X, V)
```

- Con `X` scalare e `V` vettore, che restituisca nel vettore `I`...
- ...Gli indici degli elementi che in `V` sono uguali a `X`

Per esempio:

```
V = [1, 2, 4, 2];
my_findidx(2, V)  % Denota [2, 4]
```

- Si implementi la funzione utilizzando un ciclo `for`

Esercizio: Trovare Elementi in un Vettore

Si verifichi la correttezza nella funzione principale `es_findidx`:

- Si utilizzino dei vettori di numeri interi casuali, ottenuti con `randi`

Come ottenere lo stesso risultato con le funzioni predefinite?

- Si determini un metodo e lo si sfrutti per fare un confronto
- Suggerimento: vi ricordate della funzione `find`?

Laboratorio di Informatica T

Esercizio: Prodotto Matriciale

Esercizio: Prodotto Matriciale

Nel file di funzione **`es_mprod.m`**, si definisca la funzione ausiliaria:

Si definisca una funzione:

```
function Z = my_mprod(A, B)
```

- Che calcoli il prodotto matriciale di **`A`** e **`B`**

Si verifichi la correttezza:

- Nella funzione principale
- Utilizzando matrici di numeri casuali (attenzione alle dimensioni!)
- Confrontandosi con l'operatore di prodotto in Matlab

Suggerimento: ogni prodotto riga/colonna è un prodotto scalare!

Laboratorio di Informatica T

Esercizio: Scorrimento Circolare

Esercizio: Scorrimento Circolare

Nel file di funzione `es_shift.m`, definite la funzione ausiliaria:

```
function W = my_shift(V)
```

- Deve restituire un vettore `W`, identico a `V`...
- ...Ma con gli elementi spostati a sx di una posizione...
- ...E in modo circolare: il primo elemento diventa l'ultimo

Per esempio:

```
my_shift([1, 2, 3, 4])  % Denota [2, 3, 4, 1]
```

Esercizio: Scorrimento Circolare

Verificate la correttezza nella funzione principale `es_swap`:

- Utilizzate vettori casuali o inseriti a mano
- Controllate il risultato visivamente

Suggerimento:

- Si può implementare la funzione scambiando gli elementi consecutivi
- ...Ossia scambiando `W(ii)` con `W(ii+1)` in sequenza

Privacy Policy