Esercizi di riepilogo

Esercizio

Scrivere uno *script* Matlab che crea il vettore dei primi 50 numeri dispari

$$\begin{bmatrix} 1 & 3 & 5 & 7 & \dots & 99 \end{bmatrix}.$$

Esercizio

Scrivere uno *script* Matlab che crea la matrice

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 2 & 0 & 0 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 & 0 \\ 4 & 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 5 & 5 & 0 \\ 6 & 6 & 6 & 6 & 6 & 6 \end{bmatrix},$$

utilizzando due cicli `for` uno dentro l'altro (*annidati*).

Esercizio: crescita lineare vs. esponenziale

Esercizio

In due allevamenti di animali, la popolazione di animali cresce in questo modo:

- Allevamento A: inizialmente, sono presenti 100 animali. Ogni mese, vengono aggiunti 10 animali.
- Allevamento B: inizialmente, sono presenti 8 animali. Ogni mese, vengono aggiunti $\frac{1}{2}B$ animali, dove B è il numero di animali presenti (ad es. ogni coppia di animali produce un figlio).

Quanti animali sono presenti nei due allevamenti dopo un anno (12 mesi)?

- Scrivere uno *script* Matlab che simula l'andamento delle due popolazioni. Salvare in due vettori A e B i valori della popolazione nei due allevamenti nel momento iniziale ($A(1) = 100, B(1) = 8$) e dopo ogni mese: $A = [A(1), A(2), \ldots, A(13)]$, $B = [B(1), B(2), \ldots, B(13)]$.

Esercizio: simulazione di modelli probabilistici

Esercizio

Scrivere uno *script* che simula per 10000 volte il lancio di due dadi (utilizzando ogni volta il comando `randi(6)` che restituisce un numero casuale da 1 a 6), e tiene traccia in un vettore di quante volte capita ognuno dei possibili risultati (tra 2 e 12). Quale numero esce più spesso?

Esercizio

Giochi al seguente gioco: vengono tirati 3 dadi; se il numero 6 non compare su nessuno dei dadi perdi 1€; se il numero 6 compare una volta vinci 1€, se compare due volte 2€, se compare tre volte 3€.
Scrivi uno script Matlab (utilizzando `for`,`if`) che simula 10000 partite del gioco. In media, si guadagna o si perde a giocare?

Modello di Lotka–Leslie

Il modello di Lotka–Leslie simula l'andamento di una popolazione di animali lungo diversi anni.

- Ogni anno, è presente un certo numero $x(1)$ di animali di età 1 (o meno — neonati/cuccioli), un certo numero $x(2)$ di animali di età 2 (o meno), ... un certo numero $x(5)$ di animali di età 5 o meno.
- Di tutti gli animali di età 1, una frazione m_1 muore durante l'anno, e una frazione $1 - m_1$ sopravvive raggiungendo la fascia di età successiva. Di tutti gli animali di età 2, una frazione m_2 muore e una frazione $1 - m_2$ sopravvive, e così via fino a m_4. Raggiunta età 5, tutti gli animali rimanenti muoiono.
- Ogni anno, gli animali di età k generano un numero di figli medio n_k.

Per esempio, $m = [0.8, 0.2, 0.2, 0.5]$, $n = [0, 4, 3.5, 2.5, 1]$.

Partendo da $x = [1000, 0, 0, 0, 0]$, quale sarà la popolazione dopo 10 anni?

Funzioni

In alcuni script che abbiamo creato, compaiono alcuni parametri che possiamo impostare a valori numerici diversi; per esempio, tra i nostri primi script c'era

```
% crea una matrice identita'

n = 6
I = zeros(n, n);
for k = 1:n
    I(k,k) = 1;
end
```

Avevamo detto che piuttosto che scrivere 3 volte il numero 6, è meglio assegnare una volta sola all'inizio la dimensione ad una variabile `n`, in modo da poterla cambiare più facilmente.

Avevamo anche detto che esiste già l'istruzione Matlab `I = eye(6)`, che restituisce la stessa matrice.

Funzioni

Ora vediamo come creare nuovi comandi sul modello di `I = eye(6)`.
Una funzione è una forma speciale di script che ha il ruolo di utilizzare uno o più valori forniti dall'utente (input) per produrre uno o più risultati (output).

Per trasformare il nostro script più sopra in una funzione dobbiamo fare tre cose:

- Mettere all'inizio del file la riga `function I = identita(n)` (rimpiazzando `n`=6) in cui specifichiamo quali variabili sono l'input e l'output della funzione.
- Salvare il nostro file dandogli lo stesso nome della funzione, `identita`.
- Dalla finestra dei comandi (quella con >>), chiamare la funzione specificando il valore degli input e (opzionalmente) una variabile in cui restituire il risultato.

```
function I = identita(n)
 % crea una matrice identita'
I = zeros(n, n);
for k = 1:n
    I(k,k) = 1;
end
```

```
>> identita(6);
>> A = identita(1)
A =
     1
>> k = 3;
>> J = identita(k+3);
```

Variabili all'interno delle funzioni

Una funzione ha uno "spazio di lavoro" privato che contiene variabili diverse rispetto a quelle della riga dei comandi.

Variabili all'interno della funzione (ad es. `k`) non sono visibili all'esterno, e non vanno a cambiare il valore di eventuali variabili esterne che hanno lo stesso nome.

Similmente, una funzione non "vede" altre variabili esistenti; interagisce con l'esterno unicamente tramite le variabili di input e output.

Più nel dettaglio, quando chiamiamo una funzione con (ad esempio) `>> J = identita(k+3)`, Matlab fa le seguenti cose:

- Assegna alle variabili di input (`n` in questo caso) il valore specificato nella linea di comando (il valore di `k+3`, cioè 6, in questo caso).
- Esegue tutto il codice contenuto nel file della funzione.
- Restituisce all'esterno il valore delle variabili di output (`I` in questo caso), dove può venire visualizzato a schermo, o assegnato ad altre variabili della linea di comando (`J` in questo caso).

Esempio più complicato

Questa funzione restituisce due variabili:

```
function [minimo, massimo] = ordina(a, b)
if a > b
    minimo = b;
    massimo = a;
else
    minimo = a;
    massimo = b;
end
```

Cosa viene visualizzato qui?

```
a = 5;
b = 6;
c = 8;
[c, d] = ordina(c+a, a);
a, b, c, d %scrive a schermo il valore di queste 4 variabili
```

Esercizi

Esercizio

Scrivere una `function A = crea_matrice(n)` che crea una matrice $n \times n$ costruita come nella nostra prima slides (la riga k-esima contiene k volte il numero k, e poi tutti zeri).

Esercizio

Scrivere una `function S = sommaquadrati(n)` che calcola $1^2 + 2^2 + 3^2 + \cdots + (n-1)^2 + n^2$.

Verificarne il funzionamento calcolando per esempio `sommaquadrati(4)`.

Esercizio

Scrivere una `function M = media(v)` che calcola la media degli elementi di un vettore v.

Esercizio

Scrivere una `function grafico(a,b,c,d)` che prende in input quattro numeri e traccia il grafico di

$$y = \frac{at + b}{ct + d}$$

in $[-2, 2]$.

La funzione in questo caso non ha nessun valore di ritorno, ma se includiamo il comando `plot(x, y)` al suo interno avrà comunque l'effetto di aprire una finestra con il grafico.

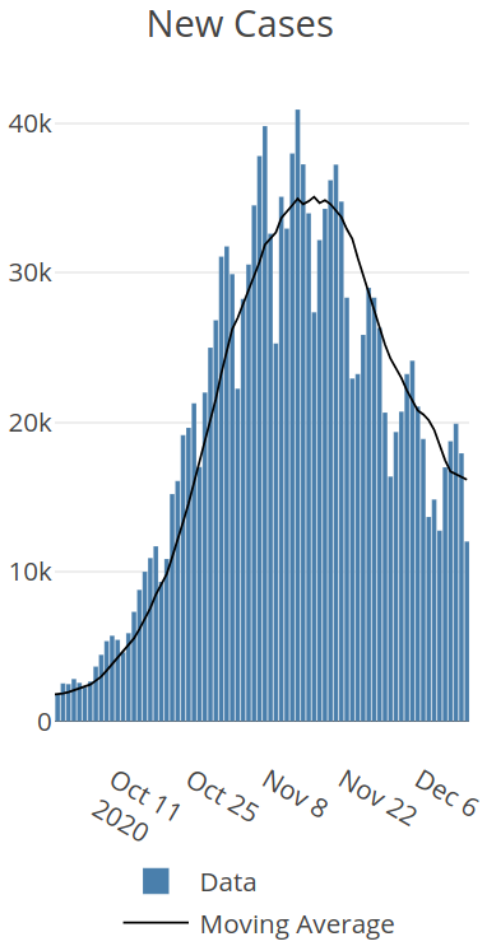
Che disegno ottenete con `grafico(1,0,0,1)`? Con `grafico(1,0,1,0)`? Con `grafico(1,2,3,4)`?

Esercizi

Media mobile (*moving average*): tecnica per ottenere grafici più lisci (e più descrittivi). Invece di fare un plot dei valori $x_1, x_2, x_3, \ldots, x_n$, plottiamo

$$\frac{x_1+x_2+x_3}{3}, \frac{x_2+x_3+x_4}{3}, \frac{x_3+x_4+x_5}{3}, \ldots, \frac{x_{n-2}+x_{n-1}+x_n}{3}.$$

O analogamente con k termini anziché tre per volta.



Esercizio

Scrivere una
`function medie = moving_average(dati, k)`
che restituisce la 'moving average' (a k termini) degli elementi del vettore `dati` (quindi, restituisce un vettore di lunghezza... contenente...)

Grafico da `https://gbocchi-coronavirus-app.herokuapp.com`.

Ricapitoliamo

Funzioni: modo per organizzare diversamente codice

```
function S = somma(a, b, c)


S = a + b + c;
```

Va salvato in un file `somma.m` (stesso nome) e chiamato con (per esempio) `T = somma(1, 1+1, 4-1);`

Quando lo incontra, Matlab:

- Calcola, se necessario, il valore degli input (qui 1,2,3).
- Esegue tutto il codice presente dopo la riga `function`, sostituendo agli input a, b, c i valori forniti.
- Restituisce (in `T`) il valore della variabile `S` alla fine delle istruzioni.

Una `function` interagisce con l'esterno solo tramite input e output. Tutte le sue variabili 'vivono' in uno spazio privato della funzione (anche se hanno lo stesso nome di altre variabili presenti all'esterno!).

Alcune soluzioni

```
% vettore con i primi 50 numeri dispari
n = 50;
v = zeros(1, n);
for k = 1:n
    v(k) = 2*k - 1;
end
```

```
% alternativa:
n = 50;
v = zeros(1, n);
v(1) = 1;
for k = 2:n
    v(k) = v(k-1) + 2;
end
```

```
% alternativamente, in una riga:
v = 1:2:99;
```

```
n = 6;
A = zeros(n, n);
for i = 1:n
    for j = 1:i
        A(i,j) = i;
    end
end
```

```
% simulazione di allevamenti
A = zeros(1, 13);
B = zeros(1, 13);

A(1) = 100;
for n = 1:12
    A(n+1) = A(n) + 100;
end

B(1) = 8;
for n = 1:12
    B(n+1) = B(n) + 1/2*B(n);
    % per evitare frazioni di animali, potete arrotondare
    % 1/2*B(n) verso il basso cosi':
    % B(n+1) = B(n) + floor(1/2*B(n));
end
```

```
function A = crea_matrice(n)
A = zeros(n, n);
for i = 1:n
    for j = 1:i
        A(i,j) = i;
    end
end
```

```
function S = sommaquadrati(n)
S = 0;
for k = 1:n
    S = S + k^2;
end
```

```
>> sommaquadrati(4)
ans =
    30
>> 1+4+9+16
ans =
    30
```

```
function M = media(v)

n = length(v);
S = 0; % calcola prima di tutto la somma
for k = 1:n
    S = S + v(k);
end
M = S / n;
```

```
function grafico(a, b, c, d)

% generiamo punti tra -2 e 2, spaziati di 0.01
x = -2:0.01:2;
n = length(x);
y = zeros(1, n);

for k = 1:n
    y(k) = (a*x(k)+b) / (c*x(k) + d);
end

plot(x, y);
```