Temi d'esame per il corso IN400, A.A. 2020-2021
Modulo MATLAB

Simone Cacace

Tema 1: Life, il gioco della vita

Consideriamo una variante a due specie dell'automa cellulare sviluppato dal matematico inglese John Conway alla fine degli anni sessanta.
Sia M una matrice di dimensione $N \times N$, i cui elementi possono assumere solo i valori $0, 1, 2$, dove 0 corrisponde all'assenza di vita, mentre 1 e 2 indicano la presenza di due specie diverse di esseri viventi.
Nello stato iniziale la matrice contiene una distribuzione casuale delle due specie.
Per ogni $i, j = 1, ..., N$ denotiamo con $I_{i,j}$ l'insieme dei primi vicini dell'elemento $M_{i,j}$, cioe'

$$I_{i,j} = \{M_{i+1,j}\,,\, M_{i-1,j}\,,\, M_{i,j+1}\,,\, M_{i,j-1}\,,\, M_{i+1,j+1}\,,\, M_{i+1,j-1}\,,\, M_{i-1,j+1}\,,\, M_{i-1,j-1}\}\,,$$

adottando condizioni periodiche per i vicini degli elementi sul bordo:

$$M_{i,0} = M_{i,N}\,, \qquad M_{i,N+1} = M_{i,1}\,, \qquad M_{0,j} = M_{N,j} \qquad M_{N+1,j} = M_{1,j}\,.$$

Indichiamo con $\#(I_{i,j})$, $\#_1(I_{i,j})$ e $\#_2(I_{i,j})$ rispettivamente il numero totale dei vicini vivi, il numero dei vicini vivi di tipo 1 ed il numero dei vicini vivi di tipo 2. Il sistema evolve utilizzando le seguenti semplici regole:

- nascita:
  un elemento morto $M_{i,j} = 0$ nasce se è circondato esattamente da 3 elementi vivi di uno stesso tipo, diventando vivo
  di tipo 1 se $\#_1(I_{i,j}) = \#(I_{i,j}) = 3$,
  di tipo 2 se $\#_2(I_{i,j}) = \#(I_{i,j}) = 3$.

- morte o sopravvivenza:
  un elemento vivo $M_{i,j} \neq 0$ con $\#(I_{i,j}) \neq 2$ e $\#(I_{i,j}) \neq 3$ muore, altrimenti sopravvive solo se è
  di tipo 1 con $\#_1(I_{i,j}) = 2$ oppure $\#_1(I_{i,j}) = 3$,
  di tipo 2 con $\#_2(I_{i,j}) = 2$ oppure $\#_2(I_{i,j}) = 3$.

Realizzare uno script MATLAB che calcoli continuamente l'evoluzione della matrice M. Ogni elemento di M va aggiornato sulla base dell'iterazione precedente (e non dei valori appena calcolati!). Utilizzare a tal fine una matrice d'appoggio e reiterare con un opportuno *swap*.

Creare altre due matrici A_1 e A_2 (inizialmente nulle), aggiungendo 1 (ad ogni iterazione) in corrispondenza degli elementi non nulli della matrice M corrente, rispettivamente di tipo 1 o di tipo 2. La matrici A_1 e A_2 forniscono quindi il numero delle volte che una determinata posizione viene visitata dalle due specie di esseri viventi.

Creare una interfaccia utente che, tramite opportuni *uicontrol*, permetta di

- impostare la dimensione N delle matrici
- variare la distribuzione iniziale delle due popolazioni
- avviare/fermare/resettare la simulazione
- visualizzare le matrici M, A_1 e A_2 tramite diversi stili grafici (immagini, curve di livello, superfici...)

Tema 2: Equazione del calore

Sia $Q = (0,1)^2 \subset \mathbb{R}^2$ il quadrato unitario nel piano, ∂Q il suo bordo. Siano inoltre u, $f : Q \times [0,+\infty) \to \mathbb{R}$ e $g : Q \to \mathbb{R}$ con $g_{|\partial Q} = 0$.
Consideriamo il seguente problema di Cauchy per l'equazione del calore con sorgente e condizioni al bordo di tipo Dirichlet:

$$\begin{cases} \dfrac{\partial u}{\partial t}(x,t) = \Delta u(x,t) + f(x,t) & (x,t) \in Q \times (0,+\infty), \\ u(x,0) = g(x) & x \in Q, \\ u(x,t) = 0 & (x,t) \in \partial Q \times (0,+\infty). \end{cases}$$

La soluzione $u(x,t)$ rappresenta la temperatura nel punto x e al tempo t di una piastra quadrata, sottoposta ad una sorgente di calore $f(x,t)$, inizialmente a temperatura g e il cui bordo è mantenuto costantemente alla temperatura 0.
Introduciamo su $Q \times [0,+\infty)$ una griglia uniforme G con $I \times I$ nodi in spazio e infiniti nodi in tempo:

$$G = \{(x_{i,j}, t_n) \in Q \times [0,+\infty), \quad i,j = 1,...,I, \quad n \geq 0\},$$

dove

$$x_{i,j} = ((i-1)\Delta x, (j-1)\Delta x)\ , \qquad \Delta x = \frac{1}{I-1},$$

$$t_n = n\Delta t\,, \qquad n \geq 0, \quad \Delta t < \frac{1}{2}\Delta x^2.$$

Per ogni $i,j = 1,...,I$ e $n \geq 0$, denotiamo con $u^n_{i,j}$, $f^n_{i,j}$ e $g_{i,j}$ rispettivamente le valutazioni delle funzioni u, f e g sulla griglia:

$$u^n_{i,j} = u(x_{i,j}, t_n), \qquad f^n_{i,j} = f(x_{i,j}, t_n), \qquad g_{i,j} = g(x_{i,j}).$$

Possiamo allora approssimare il problema di Cauchy tramite il seguente schema numerico su G: per ogni $i, j = 1, ..., I$ e $n \geq 0$

$$\begin{cases} \dfrac{u_{i,j}^{n+1} - u_{i,j}^n}{\Delta t} = \dfrac{u_{i+1,j}^n + u_{i-1,j}^n - 4u_{i,j}^n + u_{i,j+1}^n + u_{i,j-1}^n}{\Delta x^2} + f_{i,j}^n, \quad 2 \leq i, j \leq I-1, \\ u_{i,j}^0 = g_{i,j}, \\ u_{i,1}^{n+1} = 0, \quad u_{i,I}^{n+1} = 0, \quad u_{1,j}^{n+1} = 0, \quad u_{I,j}^{n+1} = 0. \end{cases}$$

Esplicitando rispetto a $u_{i,j}^{n+1}$ (svolgere i calcoli!), lo schema assume la forma (attenzione: le condizioni al bordo sono incluse in S)

$$\begin{cases} u_{i,j}^{n+1} = S(u_{i,j}^n, u_{i+1,j}^n, u_{i-1,j}^n, u_{i,j+1}^n, u_{i,j-1}^n, f_{i,j}^n), \\ u_{i,j}^0 = g_{i,j}. \end{cases}$$

Per ogni $n \geq 0$ si può quindi calcolare la soluzione u^{n+1} utilizzando la soluzione al passo precedente u^n.

Realizzare uno script MATLAB che calcoli continuamente la soluzione dello schema. Utilizzare a tal fine due matrici di dimensioni $I \times I$ per memorizzare la soluzione al passo nuovo e quella al passo precedente, reiterando con un opportuno *swap*.

Creare una interfaccia utente che, tramite opportuni *uicontrol*, permetta di

- impostare la dimensione I delle matrici
- variare il dato iniziale g scegliendo da un menù
- avviare/fermare/resettare la simulazione
- attivare/disattivare/variare interattivamente la sorgente f (ad esempio spostando il punto in cui è concentrata)
- visualizzare la soluzione del problema tramite diversi stili grafici (immagini, curve di livello, superfici...)

Tema 3: Equazione delle onde

Sia $Q = (0,1)^2 \subset \mathbb{R}^2$ il quadrato unitario nel piano. Denotiamo con Γ_D il suo lato sinistro e con Γ_N l'unione degli altri suoi tre lati. Siano inoltre u, $f : Q \times [0, +\infty) \to \mathbb{R}$ e $\beta > 0$.
Consideriamo il seguente problema di Cauchy per l'equazione delle onde con

attrito viscoso, sorgente e condizioni al bordo di tipo misto Dirichlet-Neumann:

$$\begin{cases} \dfrac{\partial^2 u}{\partial t^2}(x,t) = \Delta u(x,t) - \beta \dfrac{\partial u}{\partial t}(x,t) + f(x,t) & (x,t) \in Q \times (0,+\infty), \\ u(x,0) = 0 & x \in Q, \\ \dfrac{\partial u}{\partial t}(x,0) = 0 & x \in Q, \\ u(x,t) = 0 & (x,t) \in \Gamma_D \times (0,+\infty), \\ \dfrac{\partial u}{\partial n}(x,t) = 0 & (x,t) \in \Gamma_N \times (0,+\infty). \end{cases}$$

La soluzione $u(x,t)$ rappresenta l'elevazione nel punto x e al tempo t di una membrana elastica fissata al lato Γ_D del quadrato e libera altrove, soggetta ad una forza di attrito viscoso (con coefficiente β) e ad una forza esterna f.
Introduciamo su $Q \times [0,+\infty)$ una griglia uniforme G con $I \times I$ nodi in spazio e infiniti nodi in tempo:

$$G = \{(x_{i,j}, t_n) \in Q \times [0,+\infty), \quad i,j = 1,...,I, \quad n \geq 0\},$$

dove

$$x_{i,j} = ((i-1)\Delta x, (j-1)\Delta x)\,, \qquad \Delta x = \frac{1}{I-1},$$
$$t_n = n\Delta t\,, \qquad n \geq 0, \quad \Delta t < \Delta x.$$

Per ogni $i,j = 1,...,I$ e $n \geq 0$, denotiamo con $u^n_{i,j}$, $f^n_{i,j}$ rispettivamente le valutazioni delle funzioni u, f sulla griglia:

$$u^n_{i,j} = u(x_{i,j}, t_n), \qquad f^n_{i,j} = f(x_{i,j}, t_n).$$

Possiamo allora approssimare il problema di Cauchy tramite il seguente schema numerico su G: per ogni $i,j = 1,...,I$ e $n \geq 1$

$$\begin{cases} \dfrac{u^{n+1}_{i,j} - 2u^n_{i,j} + u^{n-1}_{i,j}}{\Delta t^2} = \dfrac{u^n_{i+1,j} + u^n_{i-1,j} - 4u^n_{i,j} + u^n_{i,j+1} + u^n_{i,j-1}}{\Delta x^2} - \beta \dfrac{u^{n+1}_{i,j} - u^n_{i,j}}{\Delta t} + f^n_{i,j}, \quad 2 \leq i,j \leq I-1, \\ u^0_{i,j} = 0, \\ u^1_{i,j} = 0, \\ u^{n+1}_{1,j} = 0, \\ u^{n+1}_{i,1} = u^{n+1}_{i,2}, \quad u^{n+1}_{i,I} = u^{n+1}_{i,I-1}, \quad u^{n+1}_{I,j} = u^{n+1}_{I-1,j}. \end{cases}$$

Esplicitando rispetto a $u_{i,j}^{n+1}$ (svolgere i calcoli!), lo schema assume la forma (attenzione: le condizioni al bordo sono incluse in S)

$$\begin{cases} u_{i,j}^{n+1} = S(u_{i,j}^{n-1}, u_{i,j}^{n}, u_{i+1,j}^{n}, u_{i-1,j}^{n}, u_{i,j+1}^{n}, u_{i,j-1}^{n}, f_{i,j}^{n}), \\ u_{i,j}^{0} = 0, \\ u_{i,j}^{1} = 0. \end{cases}$$

Per ogni $n \geq 1$ si può quindi calcolare la soluzione u^{n+1} utilizzando la soluzione ai due passi precedenti u^n e u^{n-1}.

Realizzare uno script MATLAB che calcoli continuamente la soluzione dello schema. Utilizzare a tal fine tre matrici di dimensioni $I \times I$ per memorizzare la soluzione al passo nuovo e quelle ai due passi precedente, reiterando con un opportuno *swap*.

Creare una interfaccia utente che, tramite opportuni *uicontrol*, permetta di

- impostare la dimensione I delle matrici
- variare il coefficiente di attrito β usando una slidebar
- avviare/fermare/resettare la simulazione
- attivare/disattivare/variare interattivamente la sorgente f (ad esempio spostando un punto in cui è concentrata)
- visualizzare la soluzione del problema tramite diversi stili grafici (immagini, curve di livello, superfici...)

Tema 4: Attrazione magnetica

Consideriamo la dinamica planare di una pallina metallica (puntiforme) di massa m, soggetta ad una forza di attrito viscoso (con coefficiente $\beta > 0$) e alla forza di tipo magnetico generata da K magneti uguali (anch'essi puntiformi), posti al di sotto del piano a distanza $d = 0.25$ e in posizione orizzontale

$$X_k = (\cos(2(k-1)\pi/K), \sin(2(k-1)\pi/K)), \qquad k = 1, ..., K\,.$$

Indicando con $x(t) = (x_1(t), x_2(t))$ la traiettoria della pallina in funzione del tempo e con $\dot{x}(t)$, $\ddot{x}(t)$ rispettivamente la sua velocità e la sua accelerazione, allora le equazioni del moto sono date da

$$m\ddot{x}(t) = -\sum_{k=1}^{K} \frac{x(t) - X_k}{(|x(t) - X_k|^2 + d^2)^{\frac{3}{2}}} - \beta\dot{x}(t)\,,$$

cui aggiungiamo una condizione iniziale sulla posizione e sulla velocità

$$x(0) = x_0 = (x_{0,1}, x_{0,2})\,, \qquad \dot{x}(0) = v_0 = (v_{0,1}, v_{0,2})\,.$$

Introducendo la variabile ausiliaria $v(t) = \dot{x}(t)$ si ottiene il seguente problema di Cauchy per un sistema di equazioni differenziali ordinarie del primo ordine:

$$\begin{cases} \dot{x}(t) = v(t) \\ \dot{v}(t) = -\dfrac{1}{m}\displaystyle\sum_{k=1}^{K} \frac{x(t) - X_k}{(|x(t) - X_k|^2 + d^2)^{\frac{3}{2}}} - \frac{\beta}{m} v(t) \\ x(0) = x_0 \\ \\ v(0) = v_0 \end{cases} .$$

Definiamo un passo temporale $\Delta t > 0$ e denotiamo con $x^n = x(t_n)$, $v^n = v(t_n)$ rispettivamente la posizione e la velocità al tempo discreto $t_n = n\Delta t$.
Possiamo allora approssimare il problema di Cauchy tramite il seguente schema numerico: per $n \geq 0$

$$\begin{cases} v^{n+1} = v^n - \frac{\Delta t}{m} \left(\displaystyle\sum_{k=1}^{K} \frac{x^n - X_k}{(|x^n - X_k|^2 + d^2)^{\frac{3}{2}}} + \beta v^n \right) \\ \\ x^{n+1} = x^n + \Delta t\, v^{n+1} \\ \\ x^0 = x_0 \\ \\ v^0 = v_0 \end{cases} .$$

Per ogni $n \geq 1$ si può quindi calcolare la soluzione (x^{n+1}, v^{n+1}) utilizzando la soluzione al passo precedente (x^n, v^n).

Realizzare uno script MATLAB che calcoli la soluzione dello schema per un generico punto iniziale $x_0 = (x_{0,1}, x_{0,2})$ con velocità iniziale $v_0 = (0, 0)$. Utilizzare a tal fine dei vettori per memorizzare la soluzione al passo nuovo e quella al passo precedente, reiterando con un opportuno *swap*. Al variare del dato iniziale, la pallina si fermerà per tempi lunghi su uno dei K magneti. Per terminare il calcolo fissare un numero massimo di iterazioni $n_{\max}$, una tolleranza $\varepsilon > 0$ e utilizzare la seguente condizione

$$n > n_{\max} \quad \text{or} \quad \min_{k=1,\ldots,K} \left\{ |x^{n+1} - X_k| \right\} + |v^{n+1}| < \varepsilon\,,$$

memorizzando in uscita sia il numero n^* di iterazioni effettuate, sia l'indice k^* del magnete che realizza il minimo nell'espressione precedente.

Considerare inoltre una griglia uniforme G sul quadrato $Q_l = [-\frac{l}{2}, \frac{l}{2}]^2$ di lato l con $I \times I$ nodi:

$$G = \{x_{i,j} \in Q_l, \quad i, j = 1, ..., I\}\,,$$

dove

$$x_{i,j} = (-\frac{l}{2} + (i-1)\Delta x, -\frac{l}{2} + (j-1)\Delta x)\,, \qquad \Delta x = \frac{l}{I-1}.$$

Per *ogni* $i, j = 1, ..., I$ calcolare, fino a convergenza (utilizzando il criterio di arresto precedente), la soluzione dello schema con punto iniziale $x_0 = x_{i,j}$ e velocita' iniziale $v_0 = (0, 0)$ e creare le due matrici A e B di dimensioni $I \times I$ come segue:

$$A_{ij} = n^*, \qquad B_{ij} = \begin{cases} 0 & \text{se } n^* > n_{\max}, \\ k^* & \text{altrimenti.} \end{cases}$$

La matrice A rappresenta quindi la velocità di convergenza di ciascun punto iniziale della griglia G verso uno dei magneti, mentre la matrice B rappresenta i bacini di attrazione di ogni magnete.

Creare una interfaccia utente che, tramite opportuni *uicontrol*, permetta di

- impostare i parametri della simulazione (m, β, x_0, v_0, $n_{\max}$, ε, l, I)
- avviare il calcolo della traiettoria per il singolo dato iniziale (x_0, v_0)
- avviare il calcolo delle matrici A e B (Attenzione! Può richiedere molto tempo per I, $n_{\max}$ grandi ed ε piccolo)
- visualizzare i magneti e la traiettoria della pallina
- visualizzare le matrici A e B tramite diversi stili grafici (immagini, curve di livello, superfici...)

Tema 5: Metodo di Newton e radici $N-$esime dell'unità

Dato $N \geq 1$, consideriamo le radici N-esime dell'unità, ovvero i numeri complessi della forma

$$Z_n = \cos\left(\frac{2(n-1)\pi}{N}\right) + i \sin\left(\frac{2(n-1)\pi}{N}\right), \qquad n = 1, ..., N\,,$$

soluzioni dell'equazione complessa $z^N = 1$. Possiamo approssimare tali soluzioni applicando il metodo di Newton per la ricerca di zeri alla funzione complessa $F(z) = z^N - 1$, iterando per $k \geq 0$ lo schema

$$z_{k+1} = z_k - \frac{F(z_k)}{F'(z_k)}$$

Dal momento che $F(z)$ ha esattamente N zeri, la convergenza del metodo dipende dal dato iniziale z_0.

Realizzare uno script MATLAB che calcoli le iterate del metodo di Newton per un generico dato iniziale $z_0 \in \mathbb{C}$. Per terminare il calcolo fissare un numero massimo di iterazioni $k_{\max}$, una tolleranza $\varepsilon > 0$ e utilizzare la seguente condizione

$$k > k_{\max} \quad \text{or} \quad \min_{n=1,\dots,N} \{|z_{k+1} - Z_n|\} < \varepsilon \,,$$

memorizzando in uscita sia il numero k^* di iterazioni effettuate, sia l'indice n^* della radice N-esima dell'unità che realizza il minimo nell'espressione precedente.

Considerare inoltre una griglia uniforme G sul quadrato $Q_l = [-\frac{l}{2}, \frac{l}{2}]^2$ di lato l con $I \times I$ nodi:

$$G = \{z_{r,c} \in Q_l, \quad r, c = 1, ..., I\} \,,$$

dove

$$z_{r,c} = (-\frac{l}{2} + (r-1)\Delta z) + i(-\frac{l}{2} + (c-1)\Delta z) \,, \qquad \Delta z = \frac{l}{I-1}.$$

Per *ogni* $r, c = 1, ..., I$ calcolare, fino a convergenza (utilizzando il criterio di arresto precedente), la soluzione dello schema con punto iniziale $z_0 = z_{r,c}$ e creare le due matrici A e B di dimensioni $I \times I$ come segue:

$$A_{rc} = k^*, \qquad B_{rc} = \begin{cases} 0 & \text{se } k^* > k_{\max}, \\ n^* & \text{altrimenti.} \end{cases}$$

La matrice A rappresenta quindi la velocità di convergenza di ciascun punto iniziale della griglia G verso una delle radici N-esime dell'unità, mentre la matrice B rappresenta i bacini di attrazione di ogni radice.

Creare una interfaccia utente che, tramite opportuni *uicontrol*, permetta di

- impostare i parametri della simulazione (z_0, $k_{\max}$, ε, l, I)
- avviare il calcolo dell'approssimazione di Newton per il singolo dato iniziale z_0
- avviare il calcolo delle matrici A e B (Attenzione! Può richiedere molto tempo per I, $k_{\max}$ grandi ed ε piccolo. Scegliere inoltre I pari, in modo da escludere dal calcolo il punto $z_0 = 0$ che annulla $F'(z)$ al denominatore nel metodo di Newton)
- visualizzare le radici N-esime dell'uniità e la traiettoria del metodo di Newton
- visualizzare le matrici A e B tramite diversi stili grafici (immagini, curve di livello, superfici...)

Tema 6: Insieme di Mandelbrot e insiemi di Julia

L'insieme di Mandelbrot e gli insiemi di Julia sono forse gli insiemi frattali più popolari, anche tra i non matematici. Per definirli consideriamo la seguente famiglia di mappe quadratiche complesse:

$$F_c(z) = z^2 + c, \qquad c \in \mathbb{C}.$$

Indicando con $F_c^k(z) = F_c \circ F_c \circ ...F_c(z)$ la composizione di F_c con se stessa k volte, l'insieme di Mandelbrot consiste nell'insieme dei numeri complessi c tali che la successione $F_c^k(0)$ (che parte dal punto $z = 0$) è limitata in k:

$$\mathcal{M} = \{c \in \mathbb{C}, \quad \sup_{k \geq 0} |F_c^k(0)| < +\infty\}.$$

Esplicitando la traiettoria che parte dal punto $z = 0$ in una forma ricorsiva, otteniamo per $k \geq 0$ lo schema

$$\begin{cases} z_{k+1} = z_k^2 + c\,, \\ z_0 = 0\,. \end{cases}$$

Si può dimostrare che se esiste k tale che $|z_k| > 2$, allora la successione diverge e pertanto c è esterno ad $\mathcal{M}$.
In maniera analoga, possiamo fissare c e considerare la traiettoria che parte da un generico punto z. In tal caso, l'insieme di Julia $\mathcal{J}_c$ consiste nella frontiera dell'insieme dei punti iniziali z per cui la successione z_k non diverge.

Realizzare uno script MATLAB che calcoli la traiettoria z_k generata dallo schema per generici dati $c, z_0 \in \mathbb{C}$. Per terminare il calcolo fissare un numero massimo di iterazioni $k_{\max}$ e utilizzare la seguente condizione

$$k > k_{\max} \quad \text{or} \quad |z_k| > 2\,,$$

memorizzando in uscita il numero k^* di iterazioni effettuate.

Considerare inoltre due griglie uniformi $G_{\mathcal{M}}$ e $G_{\mathcal{J}}$, rispettivamente sui quadrati $Q_{\mathcal{M}} = [-2.5, 1.5] \times [-2, 2]$ e $Q_{\mathcal{J}} = [-2, 2]^2$ con $I \times I$ nodi:

$$G_{\mathcal{M}} = \{c_{m,n} \in Q_{\mathcal{M}}, \quad m, n = 1, ..., I\}\,,$$

$$c_{m,n} = (-2.5 + (m-1)\Delta z) + i(-2 + (n-1)\Delta z)\,, \qquad \Delta z = \frac{4}{I-1}$$

e

$$G_{\mathcal{J}} = \{z_{m,n} \in Q_{\mathcal{J}}, \quad m, n = 1, ..., I\}\,,$$

$$z_{m,n} = (-2 + (m-1)\Delta z) + i(-2 + (n-1)\Delta z)\,, \qquad \Delta z = \frac{4}{I-1}.$$

Per *ogni* $m, n = 1, ..., I$ calcolare, fino a convergenza (utilizzando il criterio di arresto precedente), la soluzione dello schema con dati $c = c_{m,n}$ e $z_0 = 0$ e

creare la matrice $\mathcal{M}$ di dimensioni $I \times I$ ponendo $\mathcal{M}_{mn} = k^*$.
Analogamente, fissato un generico $c \in \mathbb{C}$, calcolare per ogni *ogni* $m, n = 1, ..., I$ la soluzione dello schema con dati c e $z_0 = z_{m,n}$ e creare la corrispondente matrice $\mathcal{J}_c$.

Creare una interfaccia utente che, tramite opportuni *uicontrol*, permetta di

- impostare i parametri della simulazione ($k_{\max}$, I)
- variare interattivamente il parametro c per il calcolo di $\mathcal{J}_c$
- avviare il calcolo delle matrici $\mathcal{M}$ e $\mathcal{J}_c$ (Attenzione! Può richiedere molto tempo per I e $k_{\max}$ grandi
- visualizzare le matrici $\mathcal{M}$ e $\mathcal{J}_c$ tramite diversi stili grafici (immagini, curve di livello, superfici...)