

SIMULAZIONE DELL'EVOLUZIONE DI UN'EPIDEmia

Costanza Leopardi, Bianca Facchini, Valentina Moruzzi

A.A. 2023/2024
Aprile 2024

1 Introduzione

Il progetto consiste nella modellizzazione dell'evoluzione di un'epidemia mediante il modello SIR. I dati vengono letti in input mediante il file di configurazione contenente i parametri del modello β e γ , i valori iniziali di suscettibili (S), infetti (I) e rimossi (R) e la durata in giorni della simulazione (T).

Successivamente il programma è stato implementato con un altro modello epidemiologico (SIRS) che introduce la probabilità di reinfettarsi espressa dal parametro α . Per la rappresentazione grafica dell'andamento delle variabili è stata utilizzata, invece, la libreria SFML.

Per agevolare il lavoro di gruppo è stato utilizzato Github, di cui si riporta il link

2 Modelli epidemiologici

2.1 Modello SIR

Il modello SIR è un modello matematico utilizzato per studiare l'evoluzione di una malattia infettiva nella popolazione. Secondo questo modello la popolazione viene divisa in tre categorie:

- **Suscettibili (S):** individui che possono contrarre la malattia,
- **Infetti (I):** individui che contraggono la malattia,
- **Rimossi (R):** individui che hanno contratto la malattia e sono o guariti o deceduti.

La loro evoluzione temporale si basa sulle seguenti equazioni differenziali:

$$\frac{dS}{dt} = -\beta \cdot \frac{S}{N} I \quad (1)$$

$$\frac{dI}{dt} = \beta \cdot \frac{S}{N} I - \gamma \cdot I \quad (2)$$

$$\frac{dR}{dt} = \gamma \cdot I \quad (3)$$

Dove N rappresenta il numero della popolazione ed è costante:

$$N = S + I + R \quad (4)$$

Il parametro $\beta \in [0,1]$ rappresenta il tasso di contagio, mentre $\gamma \in [0,1]$ è il tasso di mortalità.

Nel modello implementato questi due parametri sono considerati anch'essi costanti mentre nella realtà variano nel tempo e dipendono da vari fattori ambientali, tra cui, ad esempio, le misure cautelative messe in atto durante la pandemia.

L'epidemia ha in inizio, risultando quindi in espansione, se l'indice $R_0 = \frac{\beta}{\gamma}$ risulta maggiore di 1.

2.2 Modello SIRS

Il modello SIRS è un'estensione del modello SIR in cui un individuo, una volta passato da infetto a rimosso, può tornare ad essere nuovamente suscettibile. Chiamata $\alpha \in [0,1]$ la percentuale di rimossi che perde l'immunità e torna ad essere suscettibile, le equazioni del modello diventano:

$$\frac{dS}{dt} = -\beta \cdot \frac{S}{N} I + \alpha \cdot R \quad (5)$$

$$\frac{dI}{dt} = \beta \cdot \frac{S}{N} I - \gamma \cdot I \quad (6)$$

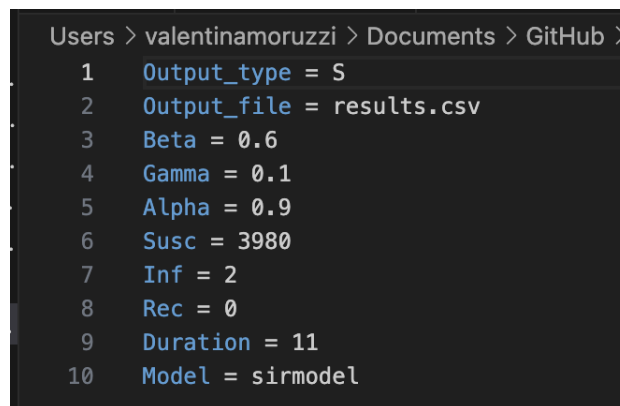
$$\frac{dR}{dt} = \gamma \cdot I - \alpha \cdot R \quad (7)$$

Anche in questo caso i parametri α , β e γ sono stati considerati costanti. Inoltre è utile notare che per α che tende a 0 il modello SIRS si riconduce al modello SIR.

3 Design del programma

La logica del programma è composta da una translation unit principale, chiamata *main.cpp*, dentro la quale vengono richiamate tutte le funzionalità. Tutti gli header file *.hpp* sono contenuti nella repository *include*, mentre i file sorgente *.cpp* in quella denominata *src*. Per la lettura dei dati in input è stata scelta come modalità l'utilizzo di un file di configurazione (*sirmodel.cfg*), come si può vedere in **Figura 1**, nel quale vengono inseriti:

- Tipologia dell'output: è possibile scegliere se stampare i dati dell'evoluzione dell'epidemia sullo standard output (opzione **S**), se scriverli in un file CSV (opzione **F**), o se rappresentarli graficamente (opzione **P**);
- Nome del file di output, nel caso si scelga l'opzione **F**;
- Valori dei parametri α , β e γ ,
- Valori iniziali di S , I e R ,
- La durata T ,
- Il modello da utilizzare: si può scegliere se usare il modello SIR (opzione *sirmodel*) o SIRS (opzione *sirmodelextended*)



```
Users > valentinamoruzzi > Documents > GitHub >  
1  Output_type = S  
2  Output_file = results.csv  
3  Beta = 0.6  
4  Gamma = 0.1  
5  Alpha = 0.9  
6  Susc = 3980  
7  Inf = 2  
8  Rec = 0  
9  Duration = 11  
10 Model = sirmodel
```

Figura 1.: Esempio file di configurazione.

L'ordine di queste informazioni è fondamentale, non deve essere cambiato. La gestione dei dati di input è stata implementata mediante la classe **sirmanage** e i rispettivi file *sirmanage.hpp* e *sirmanage.cpp*. La prima operazione consiste nella istanziatura di un oggetto di tipo *sirmanage* a cui viene passato il nome del file di configurazione (default *sirmodel.cfg*). Successivamente, mediante il metodo pubblico `read_row_fromfile()` vengono letti tutti i parametri del modello.

Una volta letti i dati di input, si procede alla definizione dello stato iniziale del modello mediante l'istanziamento di un oggetto della classe **sirdata** a cui vengono passati, in ordine, il valore dei suscettibili, degli infetti e dei rimossi appena letti. Nella classe sirdata, l'inizializzazione dei parametri avviene mediante i vari metodi **set** (**set_susc()**, **set_inf()**, **set_rec()**) nei quali viene anche controllata la correttezza formale dei parametri lanciando, all'occorrenza, opportune eccezioni. Nella classe sirdata sono, ovviamente, disponibili anche i metodi **get** dei parametri S , R e I .

L'implementazione dei modelli è realizzata mediante le classi **sirmodel** e **sirmodelextended** e i rispettivi .hpp e .cpp. La classe sirmodelextended, derivata dalla classe sirmodel, implementa il modello SIRS, estensione del modello SIR. In fase di creazione, un opportuno costruttore provvederà ad inizializzare i parametri α , β e γ .

Il metodo principale della classe sirmodel è **generate_data()** che implementa le formule del modello secondo le equazioni (1), (2) e (3). Il metodo restituisce un vettore di oggetti di tipo sirdata, ognuno dei quali rappresenta lo stato del modello in un certo istante temporale.

La classe sirmodelextended, oltre ad ereditare metodi e proprietà della classe genitore, definisce un attributo di classe proprio α , come descritto in (5) e (7), che rappresenta il parametro aggiuntivo del modello. Inoltre, c'è la sovrascrittura (overriding) del metodo generate_data() che sarà adeguato al nuovo modello.

La visualizzazione dei risultati è stata affidata alla classe **sirprint** con rispettivi .hpp e .cpp. Nella classe sono presenti due costruttori che accettano in input l'istanza di sirmanage per la gestione del tipo di output e una istanza delle classi sirmodel e sirmodelextended a seconda del modello scelto. Nella classe sono presenti tre metodi:

- **print_tostdout** per la stampa su standard output;
- **print_tofile** per la stampa su file CSV;
- **plot** per la rappresentazione dei dati mediante grafico.

In quest'ultimo caso si è fatto ricorso all'uso delle librerie grafiche SFML che è necessario installare sul proprio computer.

Infine nella repository *Test*, si trova una seconda *translation unit* *sirmodel.test.cpp* la quale, come suggerisce il nome, contiene i test che verificano la correttezza del calcolo del programma e si basa su *doctest.h*.

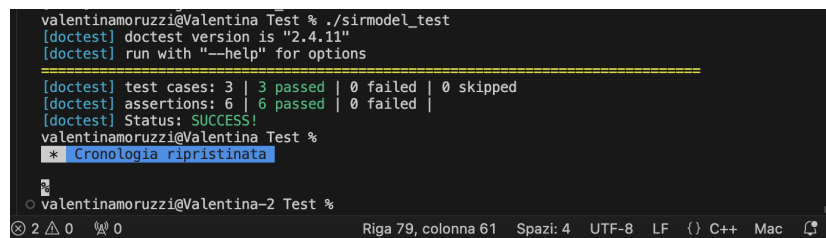
Il primo TEST.CASE mira a testare la classe *sirdata* verificando che passi correttamente i valori che inserisco in input di S,I e R.

Il secondo TEST.CASE testa la classe *sirmodel*, la quale invece prende in input anche i parametri β e γ .

Il terzo TEST.CASE testa la funzione *generate_data* dichiarata in *sirmodel.hpp* e implementata nel rispettivo file sorgente, la quale implementa i calcoli del modello SIR.

Il quarto TEST.CASE testa sempre la funzione *generate_data* ma questa volta implementata in *sirmodelextended.cpp*, la quale, invece, implementa i calcoli del modello SIRS.

Infine il quinto TEST.CASE controlla che il modello SIRS con α che tende a zero si riconduca effettivamente al modello SIR. Una volta eseguiti i test da terminale dovrebbe apparire la seguente schermata:



```
valentinamoruzzi@Valentina Test % ./sirmodel_test
[doctest] doctest version is "2.4.11"
[doctest] run with "--help" for options

=====
[doctest] test cases: 3 | 3 passed | 0 failed | 0 skipped
[doctest] assertions: 6 | 6 passed | 0 failed |
[doctest] Status: SUCCESS!
valentinamoruzzi@Valentina Test %
* Cronologia ripristinata
```

Figura 2.: Schermata da terminale dopo l'esecuzione dei test.

Si noti inoltre che per la verifica dei test è stato effettuato l'overloading dell'operatore "==" per poter scorrere e fare uguaglianze in un vettore di oggetti.

L'output di questo programma può essere scelto tra le seguenti opzioni:

- tabella su standard output (**S**),
- tabella su standard output (**F**) e file csv,
- Grafico realizzato con SFML (**P**).

Come accennato in precedenza e scritto nel file *README*, la tipologia dell'output deve essere dichiarata nel file di configurazione.

Per la parte grafica è stata utilizzata SFML: in primo luogo è stata inizializzata la *window* dentro la quale realizzare il grafico, lo stile scelto è stato *Default*. Per la finestra quindi è stato creato un oggetto *window* inserendo ampiezza e altezza. Secondariamente sono stati settati i parametri del plot e delle singole curve di suscettibili, infetti e rimossi utilizzando il metodo plot

di SFML. Nella repository del progetto è anche presente il font utilizzato (*font.ttf*).

Inoltre la classe *sirprint* deve ereditare la classe astratta *drawable*.

4 Istruzioni

Per la compilazione è stato utilizzato *cmake*, come si può vedere dal file *CMakeLists.txt*. Per avviare la compilazione bisogna accedere alla cartella che contiene il programma e inserire da terminale i seguenti comandi:

```
% cmake .
```

```
% cmake --build .
```

```
% ./SIRproject
```

Per testare invece bisogna mettersi nella repository *Test* dentro la quale si trova il file *sirmodel_test.cpp* e *Doctest.h*. In seguito, sempre da terminale eseguire le seguenti istruzioni:

```
% cmake -- build .
```

```
% ./sirmodel_test
```

In seguito apparirà la schermata riportata in **Figura 2**.

La libreria grafica SFML viene riportata nel repository principale, nell'*include* e nel *src*.

5 Analisi

Per quanto riguarda il modello SIR il grafico segue l'andamento previsto come si può vedere in **Figura 3**:

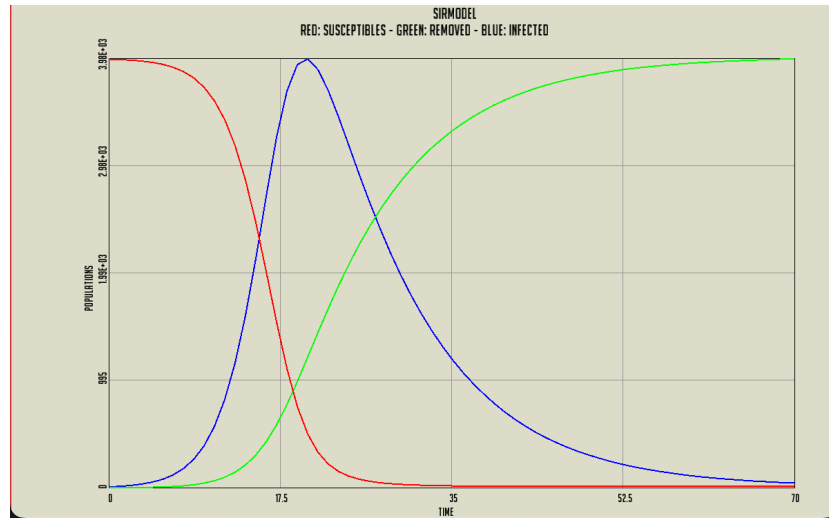


Figura 3.: Grafico del modello SIR realizzato mediante SFML.

Anche il grafico del modello SIRS riproduce l'andamento atteso (**Figura 4**), e per α che tende a 0 ritroviamo il modello SIR come si può vedere in **Figura 5**.

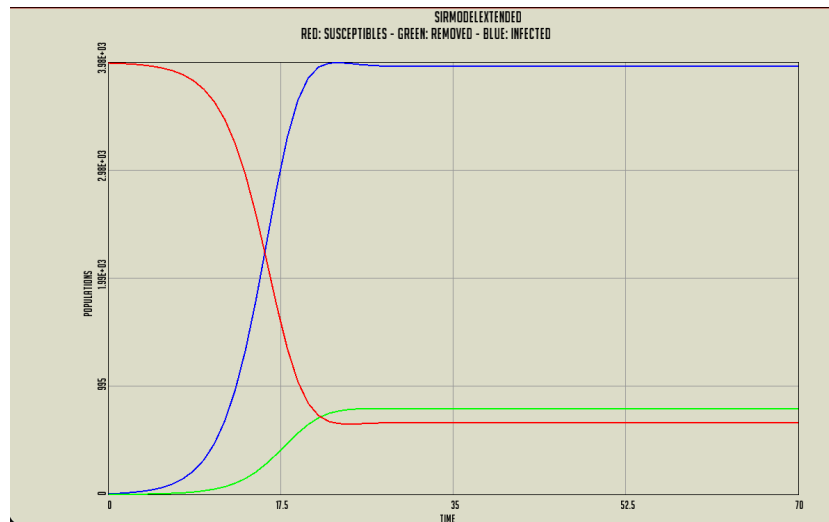


Figura 4.: Grafico del modello SIRS realizzato mediante SFML.

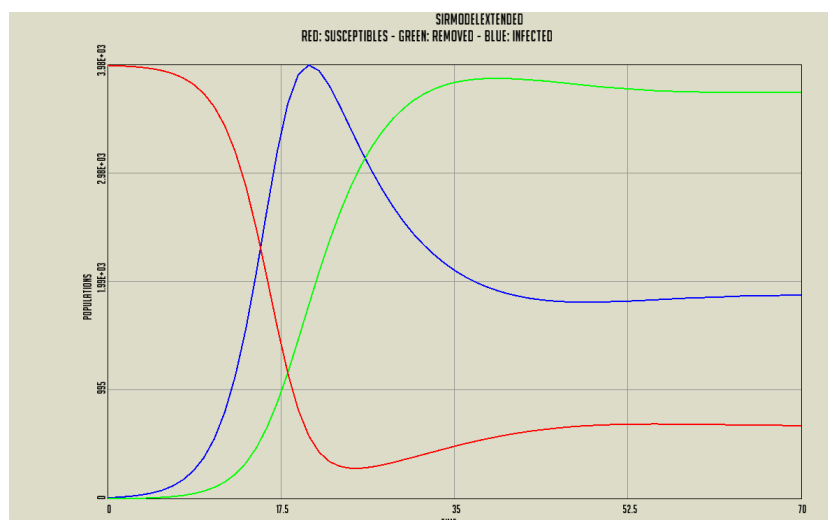


Figura 5.:Grafico del modello SIRS con α che tende a 0 realizzato mediante SFML .