Simulazione dell'evoluzione di un'epidemia

Elena Calzolaio, Elena Mariotti, Giovanni Pedrelli

30/08/2024

1 Introduzione

Lo scopo del presente progetto è quello di simulare l'andamento di un'epidemia e della sua diffusione in una popolazione. Per fare ciò è stata necessaria l'implementazione di un modello matematico: il modello SIR (Susceptible, Infectious, Removed). Tale modello discende dagli studi degli scienziati W. O. Kermack e A. G. McKendrick (1927).

Successivamente, il modello SEIR (Susceptible, Exposed, Infectious, Removed) fu creato per includere una fase di Esposizione. Il modello SEIRD (Susceptible, Exposed, Infectious, Recovered, Deceased) è un'estensione di questi modelli, includendo una fase per rappresentare i guariti e i deceduti. Sinteticamente, il modello SEIRD è un modello evolutivo compartimentale, composto cioè da 5 compartimenti S, E, I, R, D che insieme formano una popolazione N considerata costante ad ogni istante t per cui $N(t) = S(t) + E(t) + I(t) + R(t) + D(t) = const.$

Lo stato di un individuo della popolazione può evolvere in una sola direzione, da S a E, da E a I, da I a R o D, come mostrato in Fig. (1).

L'andamento dei compartimenti è definito nel modo seguente, come indicato in [2]:

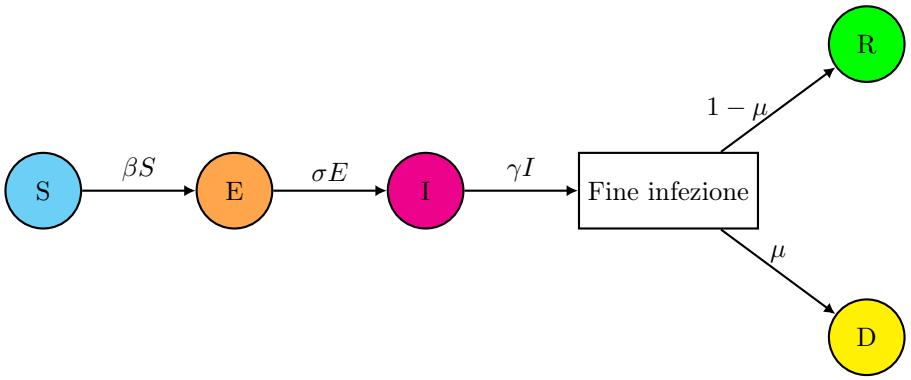

Figura 1: Struttura compartimentale del modello SEIRD.

$$\begin{cases} \dfrac{dS(t)}{dt} = -\dfrac{\beta}{N} S(t) I(t), \\ \dfrac{dE(t)}{dt} = \dfrac{\beta}{N} S(t) I(t) - \sigma E(t), \\ \dfrac{dI(t)}{dt} = \sigma E(t) - \gamma I(t), \\ \dfrac{dR(t)}{dt} = (1-\mu)\gamma I(t), \\ \dfrac{dD(t)}{dt} = \mu \gamma I(t). \end{cases} \tag{1}$$

Nell'Eq. 1 compaiono altri parametri tipici del modello considerato:

- β probabilità di infezione,
- σ inverso del tempo di incubazione, tasso di latenza,
- γ tasso di risoluzione dell'infezione, ossia inverso del tempo medio di risoluzione dell'infezione,
- μ tasso di mortalità.

Il rapporto $R_0 = \frac{E\sigma}{I\gamma}$ è detto tasso di riproduttività di base e stabilisce il valore di soglia tra il contenimento e lo scatenamento dell'epidemia. Se $R_0 > 1$ l'epidemia si scatena, se invece $R_0 < 1$ l'epidemia non sviluppa.

2 Compilazione ed esecuzione

Le *translation units* coinvolte in questo progetto sono `main.cpp`, `population.cpp`, `population.hpp`, `test.cpp` e `graph.cpp` (quest'ultimo da compilarsi col framework ROOT CERN).

La compilazione ed esecuzione del programma avvengono attraverso `CMake`, scrivendo nel terminale:

```
cmake -DCMAKE_BUILD_TYPE=Debug -S . -Bbuild
```

```
cmake --build build
```

```
cmake --build build --target test
```

```
build/population
```

Per eseguire unicamente il test occorre invece scrivere:

```
build/population.t
```

Se nel proprio computer sono installati il framework ROOT CERN e un server grafico è possibile eseguire i seguenti comandi per ottenere la rappresentazione grafica dell'andamento dell'epidemia:

```
root .
```

```
.X graph.cpp
```

È stato utilizzato il version controller `Git` e una repository `GitHub` remota per la gestione del codice.

3 Considerazioni sull'implementazione

3.1 Struttura del file `main.cpp`

All'interno del file `main.cpp` vengono richiesti in input tutti i parametri del modello necessari per la simulazione dell'epidemia. L'utente può scegliere se inserirli *1. Da file di configurazione* o *2. Da terminale*. Abbiamo implementato questa funzionalità attraverso uno statement `switch` che si articola in 2 casi determinati da un `enum` che associa ai numeri interi 1 e 2 le stringhe `FILE_INPUT` e `INTERACTIVE` e un caso `default` che gestisce eventuali errori di inserimento. Digitando sul terminale il numero, si sceglie l'opzione corrispondente.

Il *file di configurazione* è chiamato `ConfigFile.txt` e al suo interno l'utente può inserire i valori dei parametri del modello secondo le regole di immissione riportate. I valori numerici vanno inseriti al posto di quelli di default, prima del cancelletto. Tutto quello che verrà inserito dopo il cancelletto verrà ignorato. La lettura del file di configurazione avviene grazie alla funzione `readVariablesFromFile` che prende come argomento una stringa che corrisponde al nome del file dal quale si intende acquisire le variabili (`ConfigFile.txt`) e restituisce un elemento della `struct Parameters` definita nel `main` per raccogliere i valori dati in input. Questa funzione utilizza un ciclo `while` che ha come argomento la funzione `getline()` e restituisce un vettore contenente i valori in input scritti sul file di configurazione. Esempio di file di configurazione:

```
0.6   # beta, probabilita' di contagio (tra 0 e 1 inclusi)
0.3   # gamma, inverso del tempo medio di risoluzione dell'infezione (tra 0 e 1 inclusi
    )
10000 # dimensione della popolazione (intero positivo)
1000  # durata dell'epidemia in giorni (intero positivo)
35    # numero iniziale di infetti (intero positivo)
3     # tempo di incubazione in giorni (intero positivo)
0.3   # mu, tasso di mortalita' (tra 0 e 1 inclusi)
```

Se vengono inseriti in input da file di configurazione caratteri di tipo `char` o `string`, la funzione `stod`, presente in `readVariablesFromFile`, lancia un errore.

La correttezza dell'*inserimento da terminale* viene verificata nei primi due casi attraverso un ciclo `while` in cui viene proposta una domanda più precisa nel caso di inserimento di valori non adeguati. Per gli altri parametri viene utilizzato un loop `do while`. In entrambi i casi viene utilizzata la funzione `isNotInteger` che verifica che il numero dato in input sia effettivamente un intero e non un `double`. Essa restituisce un valore booleano, che è la condizione che se negata continua a far girare il ciclo.

Successivamente il programma restituisce su terminale il numero di persone appartenente a ogni categoria del modello SEIRD per ciascun giorno, calcolato attraverso le equazioni differenziali fornite. Inizialmente un messaggio comunica dopo quanti giorni l'epidemia è entrata nella sua fase di contrazione o se non è ancora terminata la sua fase di espansione. Infine, un messaggio comunica se l'epidemia si è conclusa entro i T giorni o è ancora in corso.

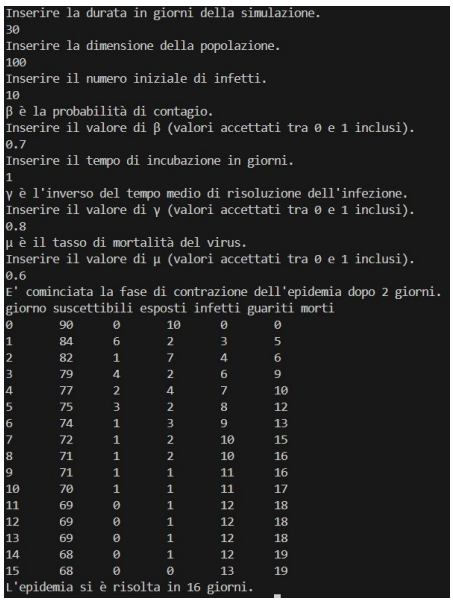

Figura 2: Esempio di una epidemia risolta in 16 giorni e rispettivi parametri della simulazione.

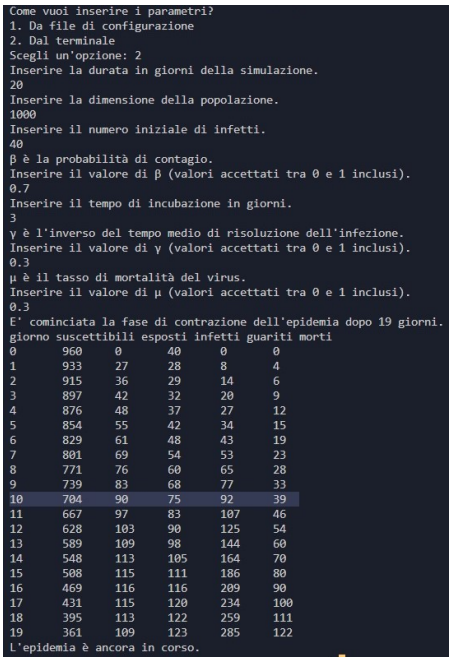

Figura 3: Esempio di una epidemia ancora in corso dopo il numero di giorni inserito dall'utente (20) e rispettivi parametri della simulazione.

3.2 Entità user-defined

```
State
```

Struttura che rappresenta lo stato e l'evoluzione di una popolazione durante una epidemia contenente le variabili S, E, I, R, D (Susceptible, Exposed, Infectious, Recovered, Deceased).

```
Population
```

Classe contenente le variabili rappresentative del modello come attributi privati. Dispone di un costruttore (`Population`) e 5 funzioni membro.

```
Population::next(State& initial_state)
```

Metodo che ha come argomento `initial_state`, elemento della `struct State`, e che restituisce un altro elemento della stessa. Tale metodo utilizza le equazioni differenziali 1 applicate alle variabili relative a un giorno fornendo lo stato del giorno successivo.

```
Population::approximation(State& initial_state, double number)
```

Metodo progettato per fare un'approssimazione dello stato del sistema ad un certo tempo. I parametri sono `State&`, elemento della `struct State`, e `double number` il quale rappresenta la dimensione della popolazione. È necessario introdurla perché i dati stampati su terminale devono essere interi in quanto rappresentativi di

elementi di una popolazione. È stata implementata questa funzione al posto del metodo `std::round()` per garantire la conservazione del numero totale di elementi della popolazione.

```
Population::evolve(State& initial_state)
```

Metodo che evolve lo stato del sistema a partire da uno stato iniziale, restituendo un vettore di elementi della `struct State` che rappresenta l'evoluzione temporale del sistema. Questa funzione itera con un ciclo `for` la funzione `next()` i cui valori di ritorno vengono copiati e approssimati tramite la funzione `approximation()` e inseriti nel vettore `dati` tramite il metodo `push_back()`. Viene generata una copia allo scopo di poter calcolare i dati relativi al giorno successivo con parametri decimali. Questa accortezza permette di evitare che venga ridotta la precisione dei dati attraverso i quali viene calcolata l'evoluzione, cosa che può portare ad una stabilizzazione errata degli output. Invece, se ne viene fatta una copia e con il vettore originario viene eseguito il calcolo, si ha un'evoluzione nel continuo che porta ad una precisione maggiore e ad una convergenza a zero di infetti ed esposti.

```
Population::peak(std::vector<State> const& dati)
```

Metodo che seleziona il giorno in cui inizia la fase di contrazione dell'epidemia, ovvero quando $R_0 = \frac{E\sigma}{I\gamma} < 1$, analizzando attraverso l'algoritmo `find_if()` il vettore `dati`. All'interno del `find_if` si è tenuto conto dei giorni di incubazione del virus aggiungendo `offset` al punto di inizio dell'applicazione dell'algoritmo. E' necessario non considerare nel calcolo per il picco dell'epidemia i giorni di incubazione, poichè l'epidemia in quei giorni risulta decrescente per poi aumentare di intensità. Restituisce un intero.

```
Population::print(std::vector<State> const& solution)
```

Metodo che stampa la soluzione dell'evoluzione del sistema. Il parametro in entrata è un vettore contenente elementi della `struct State`. Tale funzione stampa i dati su terminale e sul file `data.dat`.

4 Testing

Per testare il corretto funzionamento del programma è stato utilizzato il testing framework `Doctest`.

Le strategie impiegate nel file di test `test.cpp` per verificare la correttezza del programma consistono nel controllare che:

- la somma dei valori S, E, I, R e D sia sempre uguale a N,
- se $\beta = 0$ non si verifichino nuovi contagi e se $\mu = 1$ non si può sopravvivere cosicché R resti 0,
- se $\gamma = 0$, R e D restino 0 poiché il tasso di latenza è nullo,
- se $\mu = 0$ non si può morire cosicché $D = 0$,
- casi favorevoli e sfavorevoli rispettino le condizioni imposte,
- utilizzando i dati per il COVID nell'Emilia Romagna si evidenzi la presenza di un picco,
- morti e rimossi siano decrescenti.

5 Rappresentazione grafica con framework ROOT

Attraverso il file `data.dat`, contenente i dati in output, il file `graph.cpp` costruisce dei grafici come quello in Fig. 4.

L'esecuzione grafica riporta qualitativamente gli andamenti previsti dalle equazioni differenziali iniziali; abbiamo infatti il numero di suscettibili correttamente sempre in calo e quello dei guariti e dei morti sempre crescente in senso lato. L'andamento degli infetti dipende dal valore di R_0.

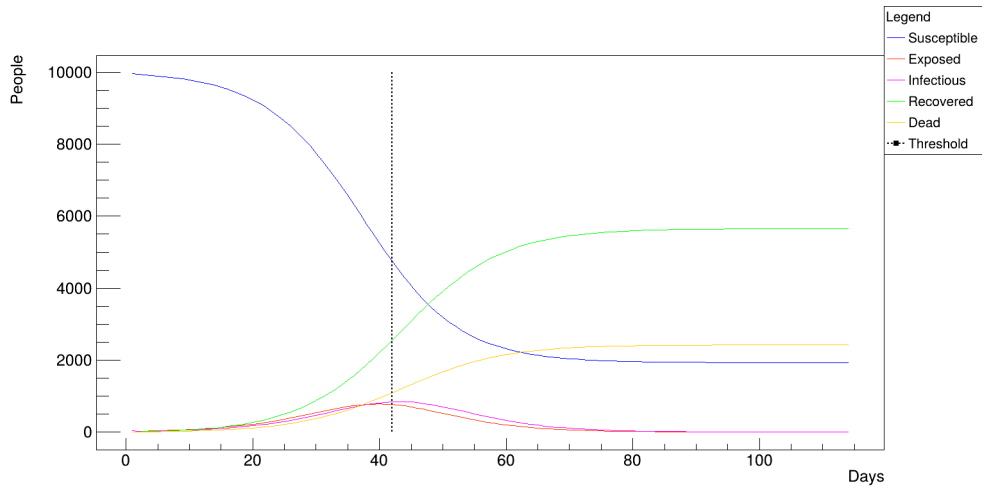

Figura 4: Andamento complessivo dell'epidemia con i parametri di default presenti in `ConfigFile.txt`.

Riferimenti bibliografici

[1] W. O. Kermack and A. G. McKendrick. A contribution to the mathematical theory of epidemics. *Proceedings of the royal society of london. Series A, Containing papers of a mathematical and physical character*, 115(772):700–721, 1927.

[2] B. Mai. Calibration of a seird epidemiological model to analyze covid-19 outbreaks in italy, 2023. URL `http://hdl.handle.net/10589/175285`. Handle Proxy.