Simulazione di una pandemia con un modello SMRV

Giulio Pastorello, Federico Gnudi

Novembre 2023

1 Il Modello

Per descrivere l'andamento della pandemia si è scelto di usare un modello SIR al quale è stata aggiunta un quarta equazione per rappresentare i vaccinati.
Le equazioni che descrivono l'evoluzione della pandemia sono quindi:

- $S(t)$: "Susceptible". Descrive il numero delle persone non malate che non hanno mai contratto l'infezione.
- $I(t)$: "Infected". Il numero di persone che ad un dato tempo t sono malate.
- $R(t)$: "Removed". Una volta che una persona è stata contagiata, dopo un certo intervallo di tempo viene *rimossa*, ovvero guarisce o muore. Si assume che, una volta guariti, i soggetti diventino immuni. L'unica possibilità di transizione da uno stato all'altro è quindi: $S \to I \to R$.
- $V(t)$: "Vaccinated". Questa equazione è un'aggiunta al modello SIR "standard", che prevede solo le prime tre equazioni. Anche in questo caso si fa l'assunzione che una persona vaccinata non possa più ammalarsi.

Il vincolo fondamentale del modello è che la popolazione totale sia costante:

$$S(t) + I(t) + R(t) + V(t) = N \tag{1}$$

Questa è un'approssimazione ragionevole se si fa l'assunzione che il numero totale di persone che muoiono a causa del virus sia trascurabile rispetto alla popolazione totale.

1.1 Parametri

Il modello descrive l'andamento della pandemia attraverso alcuni parametri:

- $\beta \in [0, 1]$: questo è interpretabile come la *contagiosità* del virus, ovvero la probabilità di trasmissione a seguito di un contatto tra un suscettibile e un infetto.
- $\gamma \in [0, 1]$: è l'inverso della durata media di un'infezione, che corrisponde alla probabilità di guarigione o morte.
- $\eta \in \mathbb{N}, \eta < N$: il numero di persone non vaccinabili, per motivi medici o personali.
- $\mu \in [0, 1]$: rappresenta la velocità delle vaccinazioni.
- $\xi \in [0, 1]$: rappresenta l'efficacia delle vaccinazioni. Si usa come fattore moltiplicativo.

1.2 Condizioni Iniziali

Per studiare la diffusione del virus vengono anche usati come condizioni iniziali i seguenti termini:

- $I_0 \in \mathbb{N}, I_0 < N$: malati al tempo $t = 0$.
- $V_0 \in \mathbb{N}, V_0 < N$: vaccinati al tempo $t = 0$.
- $R_0 \in \mathbb{N}, R_0 < N$: rimossi al tempo $t = 0$.

Ovviamente si avrà che il numero di suscettibili dall'inizio della simulazione ($t = 0$) S_0 sarà:

$$S_0 = N - I_0 - V_0 - R_0$$

.

1.3 Equazioni

Usando i parametri descritti sopra, le equazioni differenziali che descrivono la diffusione della pandemia sono quindi quattro:

$$\frac{dS}{dt}(t) = -\beta \frac{S(t)}{N} I(t) - \frac{dV}{dt}(t) \tag{2}$$

$$\frac{dI}{dt}(t) = \beta \frac{S(t)}{N} I(t) - \gamma I(t) \tag{3}$$

$$\frac{dR}{dt}(t) = \gamma I(t) \tag{4}$$

$$\frac{dV}{dt}(t) = \mu \frac{V(t)}{\xi} \left(1 - \frac{V(t)}{\xi(N-\eta)}\right) \tag{5}$$

Una simulazione basata su dati rappresentativi dell'Emilia-Romagna realizzata con Wolfram Mathematica è mostrata in figura (1). Si è usato: popolazione totale $N = 4459000$, $\beta = 0.05$, $\gamma = 0.02$, $I_0 = 530502$, $R_0 = 320000$, $V_0 = 600437$, non vaccinabili $\eta = 251042$, velocità vaccinazioni $\mu = 0.05$, efficacia vaccino $\xi = 0.83$.

1.4 Curva Logistica

Per rappresentare la progressione della campagna vaccinale, nell'equazione (5) si è scelto di usare un'equazione logistica generalizzata: $f'(x) = f(x)(1 - f(x))$, che ha come soluzione un sigmoide. Quest'approssimazione è ragionevole perché ci si aspettava un inizio più lento dovuto alla scarsità di vaccini e all'organizzazione non ancora rodata, seguito da una crescita veloce della quantità di persone vaccinate, seguito inevitabilmente da una diminuzione dovuta al fatto che le persone da vaccinare sono sempre meno. Il limite dell'equazione logistica è che usando come numero di vaccinati iniziali $V_0 = 0$ la campagna vaccinale resta a zero, ma partendo da un qualsiasi altro numero si arriverà sicuramente a vaccinare tutti i vaccinabili, in un tempo che dipende dalla velocità delle vaccinazioni μ. In figura (2) un andamento possibile della curva dei vaccinati.

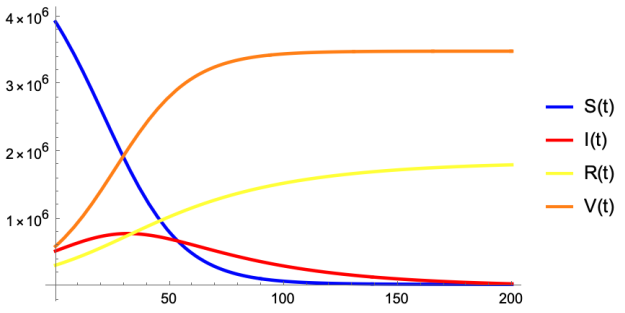

Figure 1: *Comportamento del modello SIRV con dati rappresentativi dell'Emilia-Romagna*

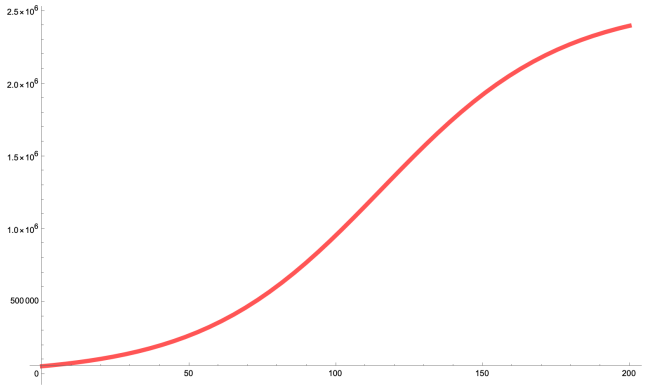

Figure 2: *Curva logistica con* $\nu = 56980, \mu = 0.0274216, \eta = 1418957$*, e popolazione totale della regione Emilia-Romagna.*

2 I Metodi Numerici

Per calcolare i risultati prodotti dal modello si sono scritti due diversi metodi numerici: il *metodo di Eulero*, meno preciso ma anche meno dispendioso a livello di calcoli, e il metodo *Runge-Kutta 4*, che è più complesso ma produce risultati migliori.

2.1 Il metodo di Eulero

Il metodo di Eulero è eseguito dalla funzione `evolve()` in `infection.cpp`. Dato un probelma di Cauchy nella forma

$$\dot{\vec{x}} = f(t, \vec{x}), \qquad \vec{x}(t_0) = \vec{x_0} \tag{6}$$

con $f(t, \vec{x})$ campo vettoriale che descrive il sistema di equazioni, $\vec{x} \in \mathbb{R}^n$ variabile di stato, t variabile indipendente (tempo), e $\vec{x}(t_0) = \vec{x_0}$ condizione iniziale, il *metodo di Eulero* discretizza il sistema nella seguente maniera:

$$\vec{x}(t_{n+1}) = \vec{x}(t_n) + hf(t_n, \vec{x}(t_n)) \tag{7}$$

dove h è lo *step size*, un intervallo di tempo arbitrario che ovviamente determina la precisione dei risultati ottenuti. Nel caso dell'*algoritmo di Eulero*, si dimostra che l'errore totale è dell'ordine di $O(h^2)$.
Nel caso del modello SMRV, il sistema è *autonomo*, ovvero le euqazioni non dipendono esplicitamente dal tempo. L'algoritmo di discretizzazione assume quindi la forma:

$$\vec{x}(t_{n+1}) = \vec{x}(t_n) + hf(\vec{x}(t_n)) \tag{8}$$

Le equazioni risultanti sono i valori che stampiamo:

$$\begin{cases} S(t_{n+1}) = S(t_n) + h\left(-\beta\frac{S(t_n)}{N}M(t_n) - \mu\frac{V(t_n)}{\xi}\left(1 - \frac{V(t_n)}{\xi(N-\eta)}\right)\right) \\ M(t_{n+1}) = M(t_n) + h\left(\beta\frac{S(t_n)}{N}M(t_n) - \gamma M(t_n)\right) \\ R(t_{n+1}) = R(t_n) + h\gamma M(t_n) \\ V(t_{n+1}) = V(t_n) + h\mu\frac{V(t_n)}{\xi}\left(1 - \frac{V(t_n)}{\xi(N-\eta)}\right) \end{cases} \tag{9}$$

dove si è usata la notazione $\Delta X = X(t_{n+1}) - X(t_n)$. Questo permette di ridurre il numero di valori da calcolare per ogni iterazione da 4 a 3.

2.2 Runge-Kutta 4th Order

L'algoritmo alternativo usato per calcolare i valori delle variabili di stato giorno-per-giorno è il Runge-Kutta del quarto ordine, abbreviato spesso come RK o RK4, che è eseguito nella funzione `rk()` in `rk.cpp`. Come da nome, l'errore totale è dell'ordine di $O(h^4)$. Partendo da un'equazione generica nella forma dell' eq. (6) l'approssimazione che viene fatta è la seguente:

$$\vec{x}(t_{n+1}) = \vec{x}(t_n) + \frac{h}{6}(k_1 + k_2 + k_3 + k_4), \qquad \begin{cases} k_1 = f(t_n, \vec{x}(t_n)) \\ k_2 = f(t_n + \frac{h}{2}, \vec{x}(t_n) + h\frac{k_1}{2}) \\ k_3 = f(t_n + \frac{h}{2}, \vec{x}(t_n) + h\frac{k_2}{2}) \\ k_4 = f(t_n + h, \vec{x}(t_n) + hk_3) \end{cases} \tag{10}$$

Nel caso del nostro modello, ricordando che il sistema è autonomo (ovvero il campo è nella forma $f(\vec{x})$), il calcolo dei coefficienti è dato dalle seguenti espressioni:

$$\begin{cases} (V)\ a_1 = \mu\frac{V(t)}{\xi}\left(1 - \frac{V(t)}{\xi(N-\eta)}\right) \\ (S)\ \ b_1 = -\beta\frac{S(t)}{N}M(t) - a_1 \\ (R)\ \ c_1 = \gamma M(t) \\ (M)\ d_1 = -a_1 - b_1 - c_1 \end{cases} \quad \begin{cases} a_2 = \mu\frac{V(t)+a_1h/2}{\xi}\left(1 - \frac{V(t)+a_1h/2}{\xi(N-\eta)}\right) \\ b_2 = -\beta\frac{S(t)+b_1h/2}{N}(M(t) + d_1h/2) - a_2 \\ c_2 = \gamma(M(t) + d_1h/2) \\ d_2 = -a_2 - b_2 - c_2 \end{cases} \quad ... \tag{11}$$

Scegliendo lo step size $h = 1$(giorno), si otterranno i valori delle quattro variabili di ogni giorno. Ad ogni iterazione vengono ri-calcolati i coefficienti $a_i, b_i, c_i, d_i, i \in \{1, 2, 3, 4\}$, che sarebbero dunque 16 valori per ogni giornata. Usando lo stesso ragionamento di prima sul vincolo, si riesce a esprimere uno dei quattro coefficienti in funzione degli altri tre, e quindi i valori da calcolare si riducono a 12 per ogni giornata.

3 Il Codice

PERCHè SI è deciso di dividerlo in più *translation units*: `infection.cpp`, `display.cpp` e `main.cpp`. Di seguito si dà una una breve spiegazione del funzionamento e dei metodi implementati.

3.1 Infection

3.1.1 La classe

Per rendere il codice più leggibile si è scelto di raccogliere i metodi riguardanti l'evoluzione della pandemia all'interno del

```
namespace epidemic{.
```

All'interno di `infection.h` si è definita una struct `State`:

```
struct State {
    int S;
    int M;
    int R;
    int V;
  };
```

che descrive i valori delle quattro variabili di stato in un dato giorno. Si è poi scritta la classe `Infection`

```
class Infection {

int m_time_indays;
std::vector<State> m_data;
int const m_N;
```

che ha come membri privati la durata della simulazione, un vettore di stati, che rappresenta l'evoluzione della pandemia, e la popolazione totale.

3.1.2 Evoluzione

Come anticipato sopra, si può scegliere fra due diversi metodi numerici per discretizzare le equazioni del modello. Il primo è il metodo di Eulero, implementato dal metodo `evolve()`:

```
Infection evolve(Infection const &plague, double beta,
double gamma, int no_vax, double vel_vax, double eff_vax) {
```

COME FUNZIONA?
L'altra opzione è il metodo Runge-Kutta in `RK4()`:

```
Infection RK4(Infection const &plague, double beta,
double gamma, int no_vax, double vel_vax, double eff_vax) {
```

COME FUNZIONA?

3.2 Display

In `display.cpp` si sono raggruppati i metodi che, come da nome, danno una rappresentazione dei risultati ottenuti.
Il primo metodo è

```
void print(Infection const &infection){
```

che stampa sul terminale i risultati della simulazione e il valore `int` del giorno, avvalendosi della funzione

```
int count_digit(int n){
```

dichiarata sempre all'interno di `display.cpp`, che serve ad impaginare in modo leggibile i valori stampati. Si è usata inoltre la libreria `termcolor` che permette di colorare quanto stampato su terminale per distinguere meglio i nomi delle variabili.
Oltre a stampare i valori della simulazione su terminale si è voluto scrivere un metodo che facesse degli istogrammi per rappresentare i valori delle variabili giorno-per-giorno. Questa funzione è implementata dal metodo `graph()`:

```
void graph(Infection const &infection) {
```

Per questo si è usata la libreria `matplotplusplus`, che è basata su `gnuplot`. al *run-time* viene protto un file contenente i quattro istogrammi che viene salvato automaticamente in formato .jpg e viene sovrascritto ogni volta che viene fatto girare di nuovo il programma.

4 Compilazione

Per costruire il programma si è deciso di usare `cmake`, nel folder `parte1/` è incluso il `CMakeLists.txt` che permette di costruire il progetto (caricando la libreria di plotting). Per compilare serve fare la build con `cmake .`, poi `cmake --build ..` Una volta fatto questo vengono prodotti due eseguibili: `infection_tty` e `infection.test`. Poiché la libreria `matplotplusplus` dava alcuni problemi di linking su MacOS, si è deciso semplicemente di clonare la repository github della libreria all'interno del folder in cui viene implementata, utilizzando poi il comando `add_subdirectory(matplotplusplus)` su CMake.