Multi-scale Hybrid Method Algorithm

Yuan Yin (Rebecca)

March 18, 2021

Aim: Find $\theta(I)$ s.t. $\theta(I) \propto \left[\sum_{j\geq 1}^{I} V_j T_j\right] / T_{ave} V_{ave}$

Step 1: Assume the W-H subsystem is const.

W.l.o.g. $(T, T^*, V) = (\bar{T}, \bar{T}^*, \bar{V})$ ← Non-viral free Steady State. (i.e. $\bar{V} \neq 0$)

The within-host system in the nested model reads

$$\begin{cases} \dot{T} = \Lambda_c - kTV - \mu_c T, \\ \dot{T}^* = kTV - (\mu_c + \delta_c)T^*, \\ \dot{V} = pT^* - cV, \end{cases} \quad (2)$$

$$\Rightarrow \begin{cases} \Lambda_c = \bar{T}(\mu_c + k\bar{V}) & \text{— ①} \\ k\bar{T}\bar{V} = (\mu_c + \delta_c)\bar{T}^* & \text{— ②} \\ p\bar{T}^* = c\bar{V} & \text{— ③} \end{cases}$$

③ $\Rightarrow \bar{T}^* = \frac{c}{p}\bar{V}$ — ④

② $\Rightarrow \bar{T} = \frac{(\mu_c + \delta_c)\bar{T}^*}{k\bar{V}} = \frac{(\mu_c + \delta_c)c}{pk}$

① $\Rightarrow \Lambda_c = \frac{(\mu_c + \delta_c)c}{pk}(\mu_c + k\bar{V})$

$\Rightarrow \bar{V} = \left\{\frac{\Lambda_c p k}{c(\mu_c + \delta_c)} - \mu_c\right\} / k$

④ $\Rightarrow \bar{T}^* = \left\{\frac{\Lambda_c p k}{c(\mu_c + \delta_c)} - \mu_c\right\} \frac{c}{pk}$

Hence, we firstly assume that $(T, T^*, V) =$

$$\left(\frac{(\mu_c + \delta_c)c}{pk}, \left\{\frac{\Lambda_c p k}{c(\mu_c + \delta_c)} - \mu_c\right\}\frac{c}{pk}, \left\{\frac{\Lambda_c p k}{c(\mu_c + \delta_c)} - \mu_c\right\}/k\right).$$

However: If our with-host system is const. : $T_j = T_{ave}$ & $V_j = V_{ave}$ $\forall j$

$\Rightarrow \theta(I) \propto \left[\sum_{j\geq 1}^{I} T_{ave} V_{ave}\right] / T_{ave} V_{ave} = I(t)$.

i.e. $\theta(I) \propto I(t)$. ⇐ This always happens when we treat the W-H system constant. !?

This has been verified in my numerical simulation

1 Within-Host Subsystem

The Within-Host subsystem is governed by the following Ordinary Differential Equations (Eqs (1) − (3)):

$$\frac{dT(t)}{dt} = \Lambda_c - kTV - \mu_c T, \tag{1}$$

$$\frac{dT^{\star}(t)}{dt} = kTV - (\mu_c + \delta_c)\, T^{\star}, \tag{2}$$

$$\frac{dV(t)}{dt} = pT^{\star} - cV, \tag{3}$$

By assuming that the within-host subsystem is at a fast timescale and setting Eqs (1) − (3) to 0 gives the non-virus-free equilibrium:

$$\left(\bar{T}, \bar{T^{\star}}, \bar{V}\right) = \left(\frac{(\mu_c + \delta_c)\, c}{pk}, \frac{\Lambda_c}{\mu_c + \delta_c} - \frac{c\mu_c}{pk}, \frac{\Lambda_c p}{c\,(\mu_c + \delta_c)} - \frac{\mu_c}{k}\right). \tag{4}$$

Algorithm 1: CONST_WH

output: the non-virus-free steady state solution, $\left(\bar{T}, \bar{T^{\star}}, \bar{V}\right)$

Initialize $p, k, c, \mu_c, \delta_c, \Lambda_c$

$\bar{T} \leftarrow \frac{(\mu_c+\delta_c)c}{pk}$
$\bar{T^{\star}} \leftarrow \frac{\Lambda_c}{\mu_c+\delta_c} - \frac{c\mu_c}{pk}$
$\bar{V} \leftarrow \frac{\Lambda_c p}{c(\mu_c+\delta_c)} - \frac{\mu_c}{k}$

2 Between-Host Subsystem

The deterministic Between-Host subsystem is governed by the following Ordinary Differential Equations (Eqs (5) – (6)):

$$\frac{dS(t)}{dt} = \Lambda - \beta\left(V\right)SI - \mu S, \tag{5}$$

$$\frac{dI(t)}{dt} = \beta\left(V\right)SI - \mu I. \tag{6}$$

For simplicity, we assume $\beta\left(V\right) = \beta_0 \times \bar{V}$, where $\bar{V}$ denotes the nontrivial equilibrium viral load, and β_0 is some arbitrary constant.

Algorithm 2 simulates Eqs (1) – (3) through Continuous Time Markov Chain (CTMC).

Algorithm 2: CTMC_BH

output: t (a real time array),
$S[t]$,
$I[t]$,
$infect\_timeArray$.

Initialize
$\mu, \Lambda, \beta_0, t_{end}, S[\ ], I[\ ], t[\ ], i := 1, S_0(:= S[1]), I_0(:= I[1]), t_0(:= t[1])$.

$\beta \leftarrow \beta_0 * \bar{V}$

while *While t[i]* $\leq t_{end}$ **do**
    / calculate the time stepping size */*
    $\lambda \leftarrow \beta S[i]I[i] + \mu\,(S[i] + I[i]) + \Lambda$
    $\Delta t \leftarrow$***TIMESTEPSIZE*** (λ)
    $t[i+1] \leftarrow t[i] + \Delta t$

    / decide which event will happen at current time, $t(i)$ */*
    $[S, I,$ *infect_Appear*, *infect_Die*$] \leftarrow$ ***EVENT***$(\lambda, \beta, \mu, S, I, i)$

    / record the time stamp if the new infection happens */*
    if *infect_Appear == 'Yes'* **then**
        infect_timeArray$(i) \leftarrow t[i]$
    else
        infect_timeArray$(i) \leftarrow$ *nan*   */* nan: not a number*/*
    end

    / if an existing infectious individual dies, randomly select an entry in 'infect_timeArray' corresponded to an infectious person, and set it to be 'nan' */*
    if *infect_Die == 'Yes'* **then**
        infect_indIndex $\leftarrow$ *[indicies of infectious people]*
        rand_die $\leftarrow$ *a random index drawn from infect_indIndex*
        infect_timeArray(rand_die) $\leftarrow$ *nan*
    end

    $i \leftarrow i + 1$
end

/ make sure that the final time, t_f, doesn't exceed t_{end} */*
$t_f \leftarrow t\,[i]$
if $t_f > t_{end}$ **then**
    $t \leftarrow$ *t(1 : (i-1))* */* delete the last entry in the array*/*
    $S \leftarrow$ *t(1 : (i-1))*
    $I \leftarrow$ *t(1 : (i-1))*
end

Note that **Algorithm 2** calls another two algorithms, 'TIMESTEPSIZE' and 'EVENT' (see **Algorithms 3 - 4**).

Algorithm 3: TIMESTEPSIZE

input : λ
output: Δt

/* Draw a random number from the standard uniform distribution the open interval $(0, 1)$*/
$u_1 \leftarrow$ rand
$\Delta t \leftarrow \frac{-log(u_1)}{\lambda}$

Algorithm 4: EVENT

input : $\lambda, \beta, \mu, S, I, i$
output: S,
I,
$infect\_Appear$,
$infect\_Die$.

Initialize $infect\_Appear = infect\_Die := 0$

/* Draw a random number from the standard uniform distribution the open interval $(0, 1)$*/

$u_2 \leftarrow$ rand

/* calculate the possibilities of one susceptible enters the population, gets infectious, dies, and one infectious dies respectively */

$p_1 \leftarrow \frac{\Lambda}{\lambda}$
$p_2 \leftarrow \frac{\beta S I}{\lambda}$
$p_3 \leftarrow \frac{\mu S}{\lambda}$
$p_4 \leftarrow \frac{\mu I}{\lambda}$

/* update S and I accordingly */

if $u_2 \in [0, p_1)$ **then**
    $S(i+1) \leftarrow S(i) + 1$
    $I(i+1) \leftarrow I(i)$
else if $u_2 \in [p_1, p_1 + p_2)$ **then**
    $S(i+1) \leftarrow S(i) - 1$
    $I(i+1) \leftarrow I(i) + 1$
    $infect\_Appear \leftarrow 1$
else if $u_2 \in [p_1 + p_2, p_1 + p_2 + p_3)$ **then**
    $S(i+1) \leftarrow S(i) - 1$
    $I(i+1) \leftarrow I(i)$
else
    $S(i+1) \leftarrow S(i)$
    $I(i+1) \leftarrow I(i) - 1$
    $infect\_Die \leftarrow 1$
end

3 Hybrid Multi-scale System

We develop an algorithm (**Algorithm 5**) which aims to figure out $\Theta(I)$ s.t.

$$\Theta(I) \propto \frac{\left(\sum_{j=1}^{I} V_j T_j\right)}{T_{ave} V_{ave}}. \tag{7}$$

Algorithm 5: THETA_I

Initialise the array of interest, Theta_I[ ].

/* get between-host and within-host subsystems' information first */
$[t, S, I, infect\_timeArray] \leftarrow$ **CTMC_BH**
$\left[\bar{T}, \bar{T}^{\star}, \bar{V}\right] \leftarrow$ **CONST_WH**

/* stepping through time */
for $t_i \leftarrow t_{ini}$ *to* t_{final} **do**
    /* record the infectious time-length of each infectious person */
    $infect\_length \leftarrow t_i - infect\_timeArray$ /* array subtraction */

    /* initialisation for E.q. (7) */
    $sum\_LHS, sum\_T, sum\_V, num\_infectious := 0$

    /* formulate E.q. (7) */
    for $infect\_ppl_i \leftarrow 1$ *to* $length\,(infect\_length)$ **do**
        /* The entries in 'infect_length' are either non-negative numbers, negative numbers, or 'nan' (not a number). Only the non-negative entries correspond to a valid infectious time period */
        if $infect\_length[infect\_ppl_i] \geq 0$ **then**
            $num\_infectious \leftarrow num\_infectious + 1$
            $V_j \leftarrow \bar{V}$
            $T_j \leftarrow \bar{T}$
            $sum\_LHS \leftarrow sum\_LHS + V_j T_j$
            $sum\_T \leftarrow sum\_T + T_j$
            $sum\_V \leftarrow sum\_V + V_j$
        end
    end

    $T_{ave} \leftarrow \frac{sum\_T}{num\_infectious}$
    $V_{ave} \leftarrow \frac{sum\_V}{num\_infectious}$
end

$Theta\_I\,[t_i] \leftarrow \frac{sum\_LHS}{T_{ave} V_{ave}}$

---

4 Result (of a trial run)

For within-host subsystem:

- $p = 10^1$;
- $c = 10^0$;
- $k = 10^{-7}$;
- $\Lambda_c = 10^{10}$;
- $\mu_c = 10^{-1}$;
- $\delta_c = 10^1$.

For between-host subsystem:

- $\mu = 10^{-5}$;
- $\Lambda = 10^{-5}$;
- $\beta_0 = 10^{-13}$;
- $S_0 = 990$;
- $I_0 = 10$;
- $t_{end} = 20$.

Running **Algorithm 5** 100 times gives Figure 1:

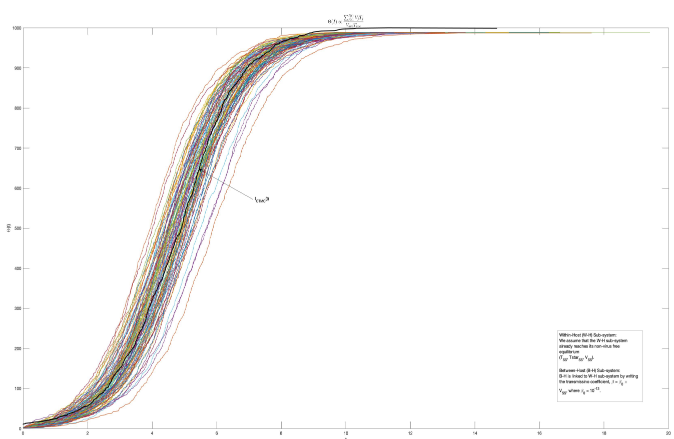

Figure 1: $\Theta(I)$ *V.S.* t