Multiscale Multicellular Spatiotemporal Modeling of Viral Infection and Immune Response

A Modular, Extensible Agent-Based Framework

Try the nanoHUB tool at
https://nanohub.org/tools/cc3dcovid19

T.J. Sego, Ph.D.

Biocomplexity Institute

Indiana University Department of Intelligent Systems Engineering

Support: NIH R01 GM122424, U01 GM111243, U24 EB028887, NSF 1720625

About Me

- Postdoctoral Fellow
  - Biocomplexity Institute, Prof. James Glazier
  - Intelligent Systems Engineering
  - Lead developer: CompuCell3D
- Ph.D. (Mech. Eng.): Purdue University, August 2019
  - Engineering Design Research Lab, Prof. Andres Tovar
- Research
  - Theoretical/computational modeling in cellular/tissue dynamics and tissue engineering applications
  - Bone biomechanics and implant design
  - Biofabrication processes and bioreactor design
  - Immunology and viral infection

compucell3d.org

Multiscale Multicellular Modeling of Viral Infection and Immune Response

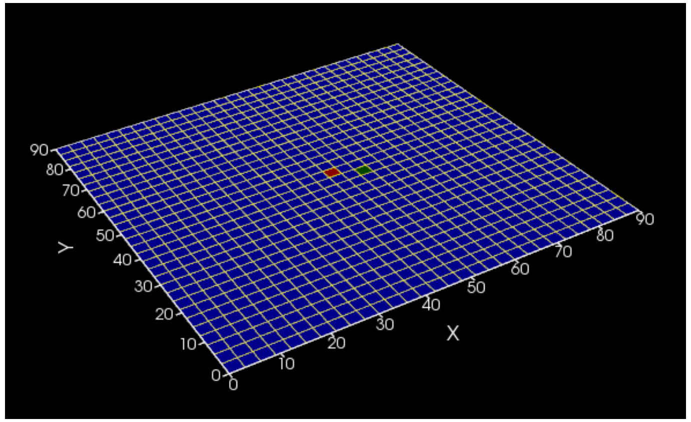
A modular framework for multiscale, multicellular, spatiotemporal modeling of acute primary viral infection and immune response in epithelial tissues and its application to drug therapy timing and effectiveness

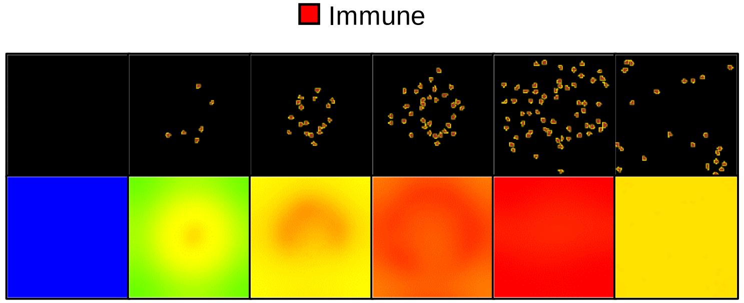
T.J. Sego, Josua O. Aponte-Serrano, Juliano Ferrari Gianlupi, Samuel R. Heaps, Kira Breithaupt, Lutz Brusch, Jessica Crawshaw, James M. Osborne, Ellen M. Quardokus, Richard K. Plemper, James A. Glazier

doi: https://doi.org/10.1101/2020.04.27.064139

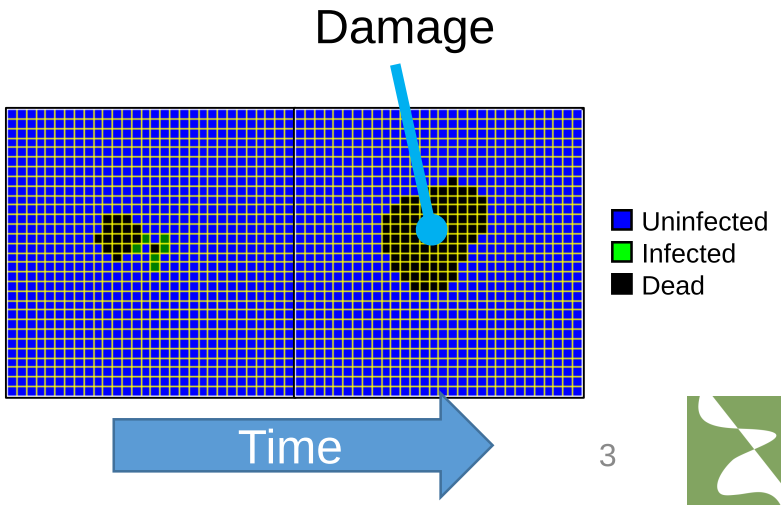
Accepted, PLoS Comp. Bio.

Watch/Star/Fork this project on GitHub:
https://github.com/covid-tissue-models/covid-tissue-response-models



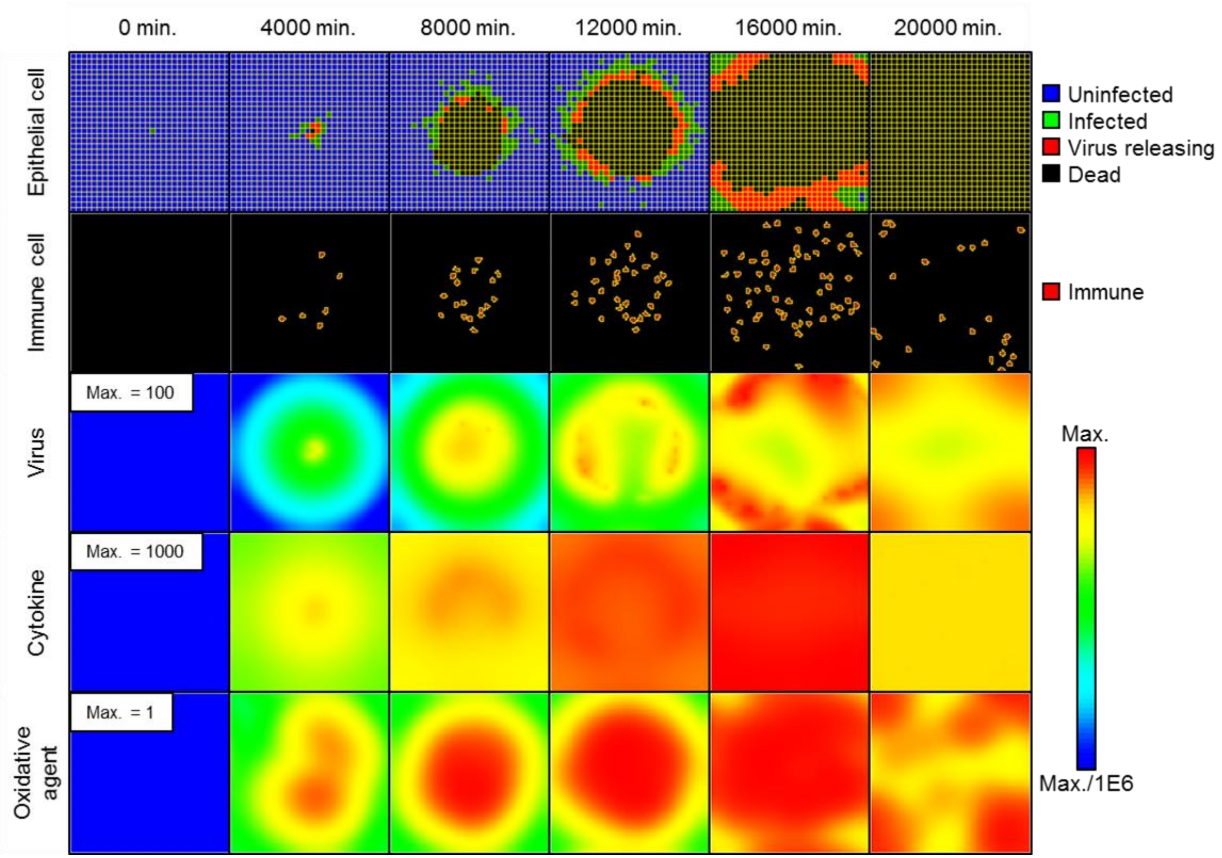


Try the nanoHUB tool at
https://nanohub.org/tools/cc3dcovid19



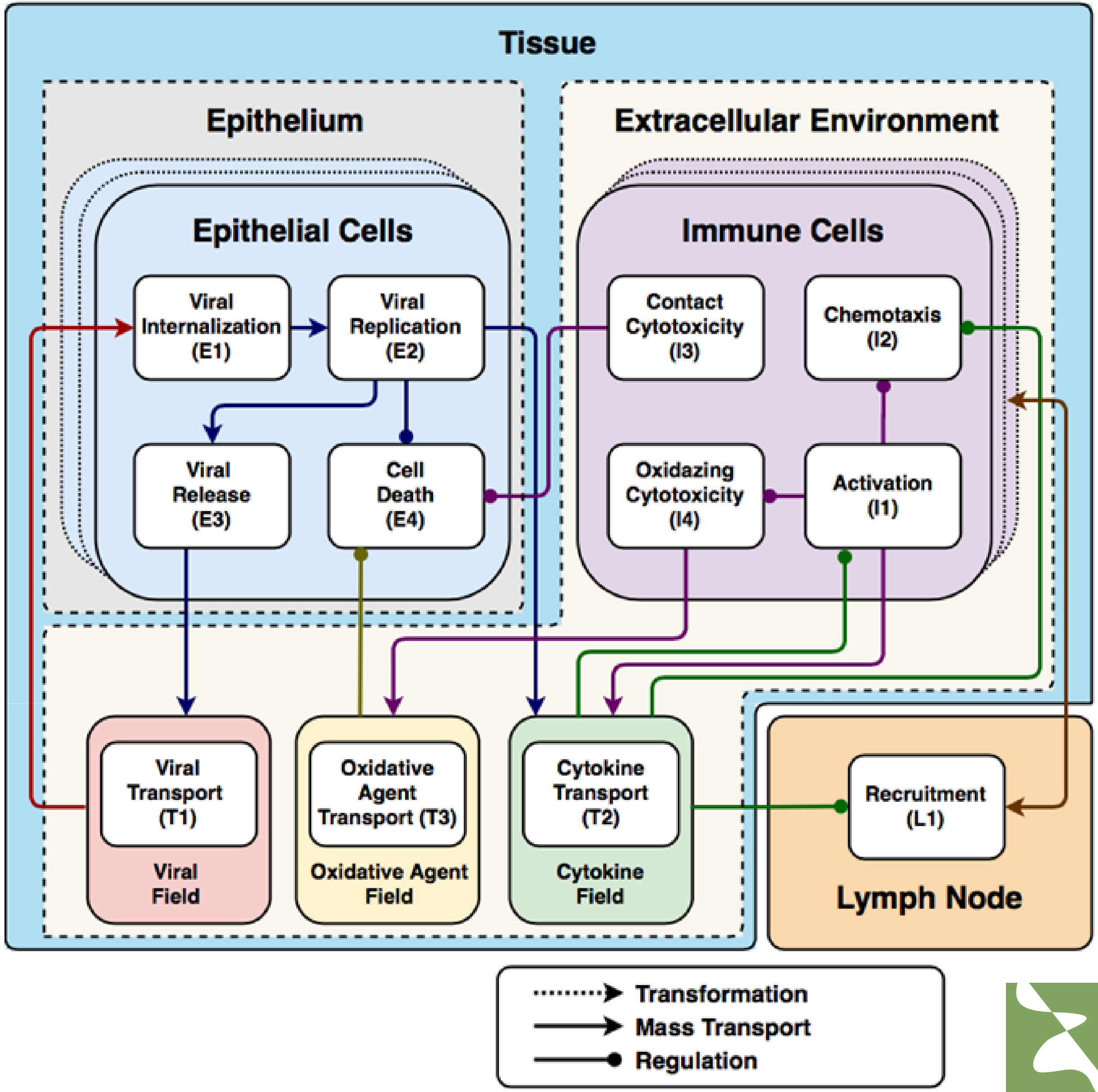
Premise: Primary Acute Local Infection and Innate Response in a Planar Milieu

- Infection in a small quasi-2D patch of susceptible tissue
- Assume primary infection
  - no pre-existing adaptive immune response
  - no specific antibodies, memory T-cells or targeted B cells
- Assume acute infection
  - consider a short time where the immune system either clears the virus, the virus spreads over the entire tissue patch, or something in between



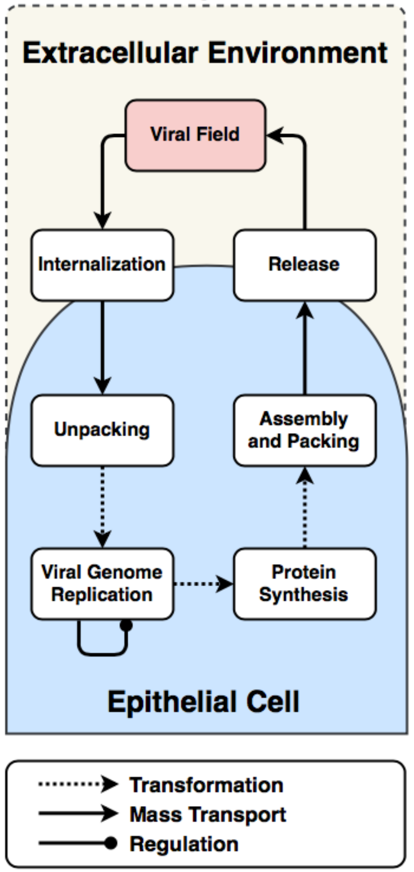
Overview of Model Components

- Two cell classes
  - Epithelial cell: the susceptible cells
  - Immune cell: the infection fighters
- Three diffusive fields
  - (Extracellular) Viral Field: extracellular virus transport
  - Cytokine Field: local and global signaling
  - Oxidative Agent Field: epithelial cell killing by immune cells
- Lymph node
  - Compartmental model
  - Regulates local immune cell population



Modules of the Viral Lifecycle

- **Viral Internalization**: how virus gets into a cell
  - Virus is taken from the environment and transferred into a cell
  - Binding to receptors determines rate of internalization vs. extracellular viral concentration
- **Viral Replication**: how virus replicates inside a cell
  - Four basic stages of replication: Unpacking, Genome Replication, Protein Synthesis, and Assembly and Packing
  - Exponential amplification phase: Genome Replication
- **Viral Release**: how virus is released into the environment
  - Virus is taken from the cell and transferred into the environment
  - Rate of release is proportional to internal amount of Assembled and Packaged genomic material



$$\frac{dU}{dt} = Uptake - r_u U$$

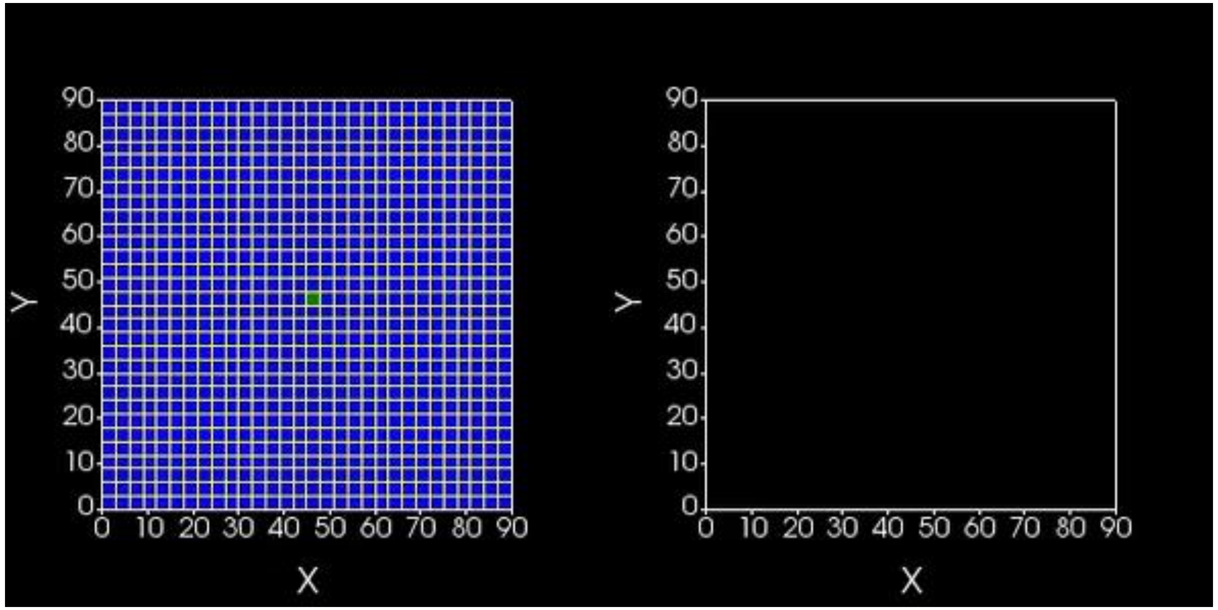
$$\frac{dR}{dt} = r_u U + r_{max} R \frac{r_{half}}{R + r_{half}} - r_t R$$

$$\frac{dP}{dt} = r_t R - r_p P$$

$$\frac{dA}{dt} = r_p P - Release$$

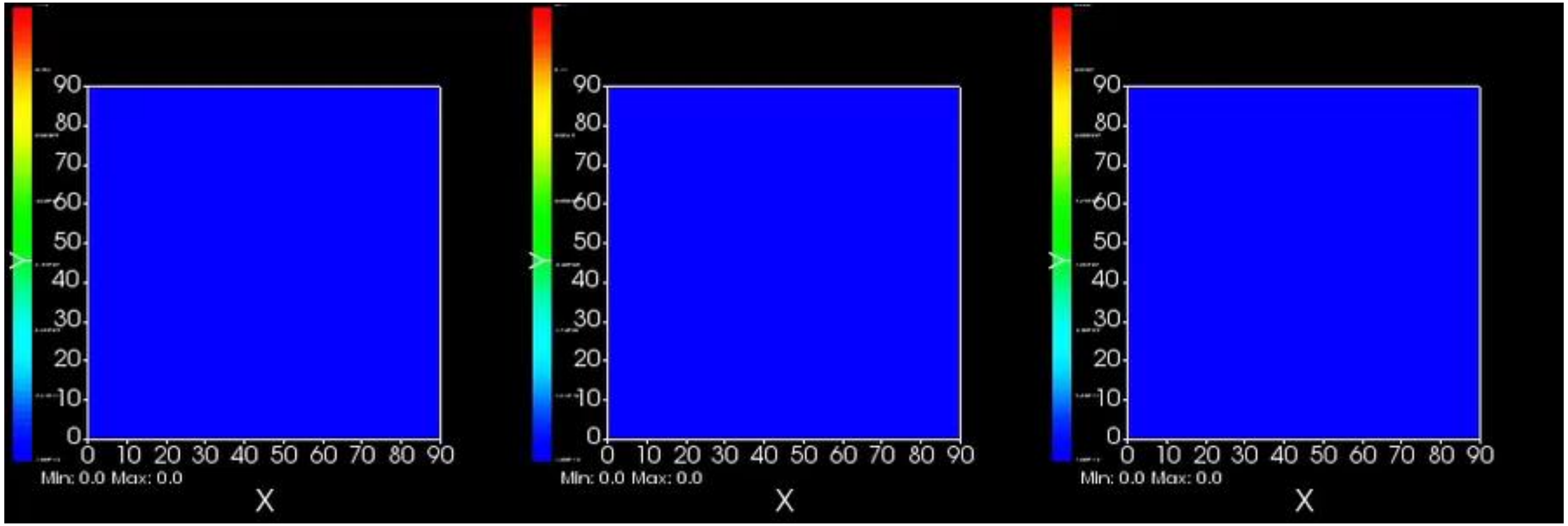
Epithelial Immune

Uninfected
Infected
Virus releasing
Dead



Immune

Virus Cytokine Oxidative Agent



Simulate Therapy with RNA-Synthesis Blocker

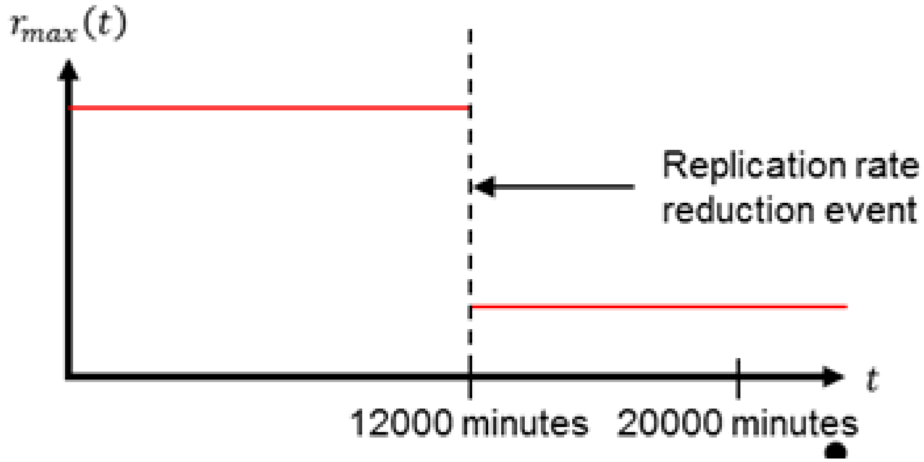
Drugs like Remdesivir inhibit RNA synthesis, the one exponential step in viral replication

Issues:

- Effectiveness decreases rapidly as the time of first treatment increases
- Optimal treatment: lowest effective dose

Easy to model and simulate

- Treatment corresponds to reducing replication rate in viral replication model
- Treatment can be applied at various times after initial infection in simulation



Example simulated therapy. r_{max} is the replication rate of all cells in simulation time.

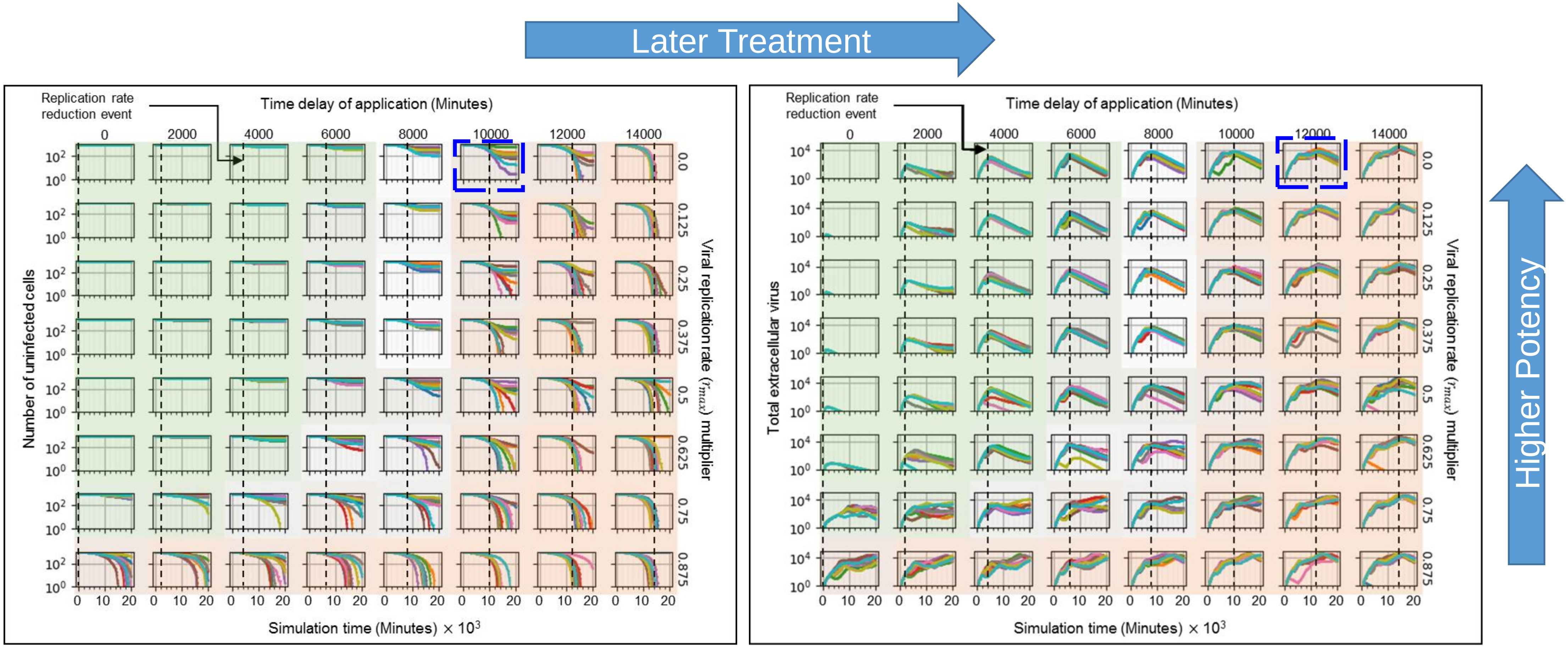
$$\frac{dU}{dt} = Uptake - r_u U$$

$$\frac{dR}{dt} = r_u U + \boxed{r_{max} R \frac{r_{half}}{R + r_{half}}} - r_t R$$

$$\frac{dP}{dt} = r_t R - r_p P$$

$$\frac{dA}{dt} = r_p P - Release$$

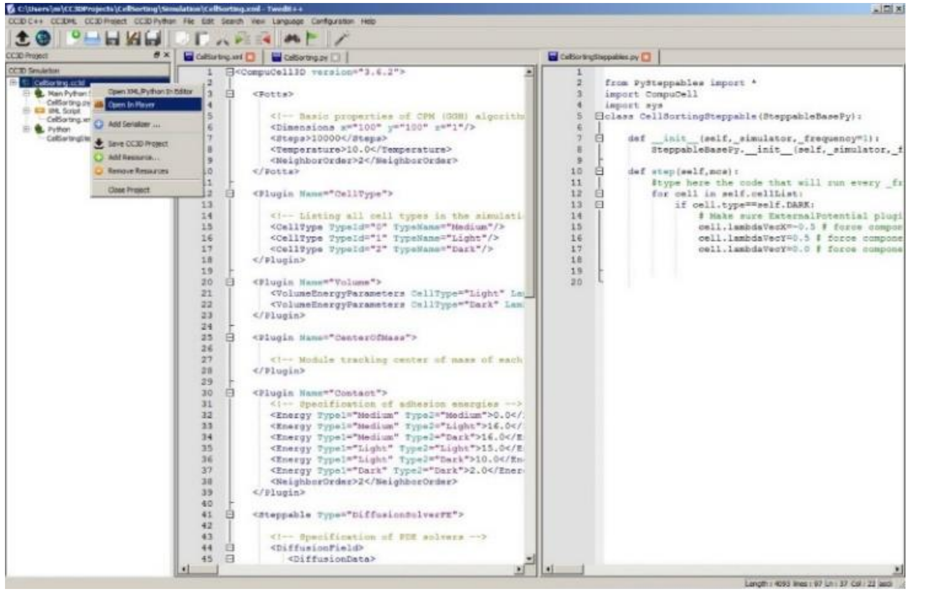
Time vs Potency Tradeoffs for an RNA-Synthesis Blocker

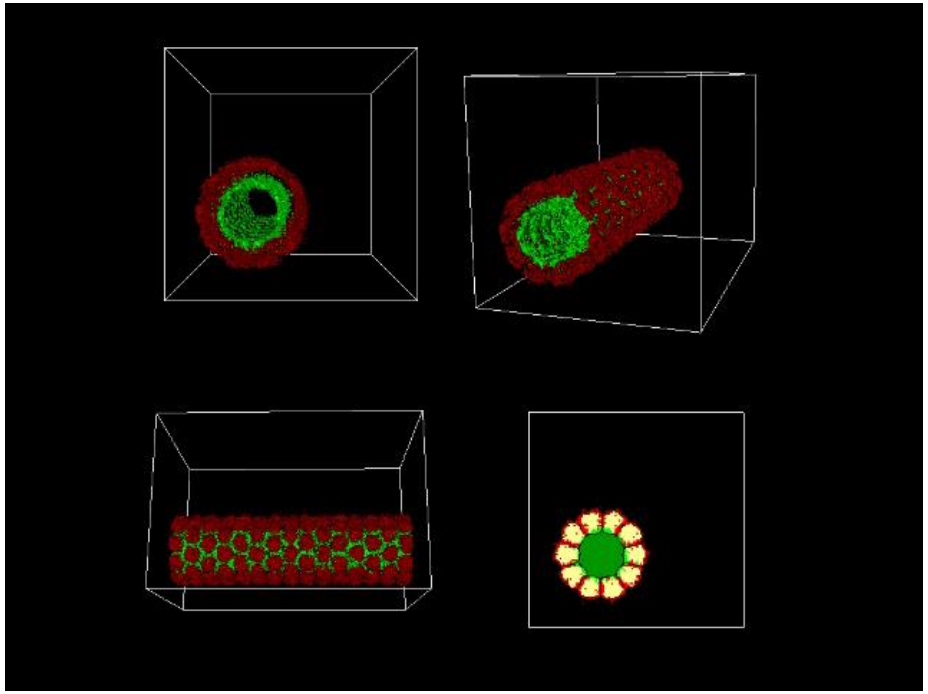


Green: virus controlled and most cells left uninfected
Red: most cells infected, virus not controlled
In between: high stochasticity, uncertain outcome

Framework Deployment

- CompuCell3D: a widely used software environment for virtual tissue modeling
  - Open source
  - PDE solver suite, ODE solver (libRoadrunner)
  - Real-time GUI-based interactive simulations
  - Code editor supporting easy model specification
  - HPC deployment (*e.g.*, Carbonate at IU)
- Modular model specification using XML and Python

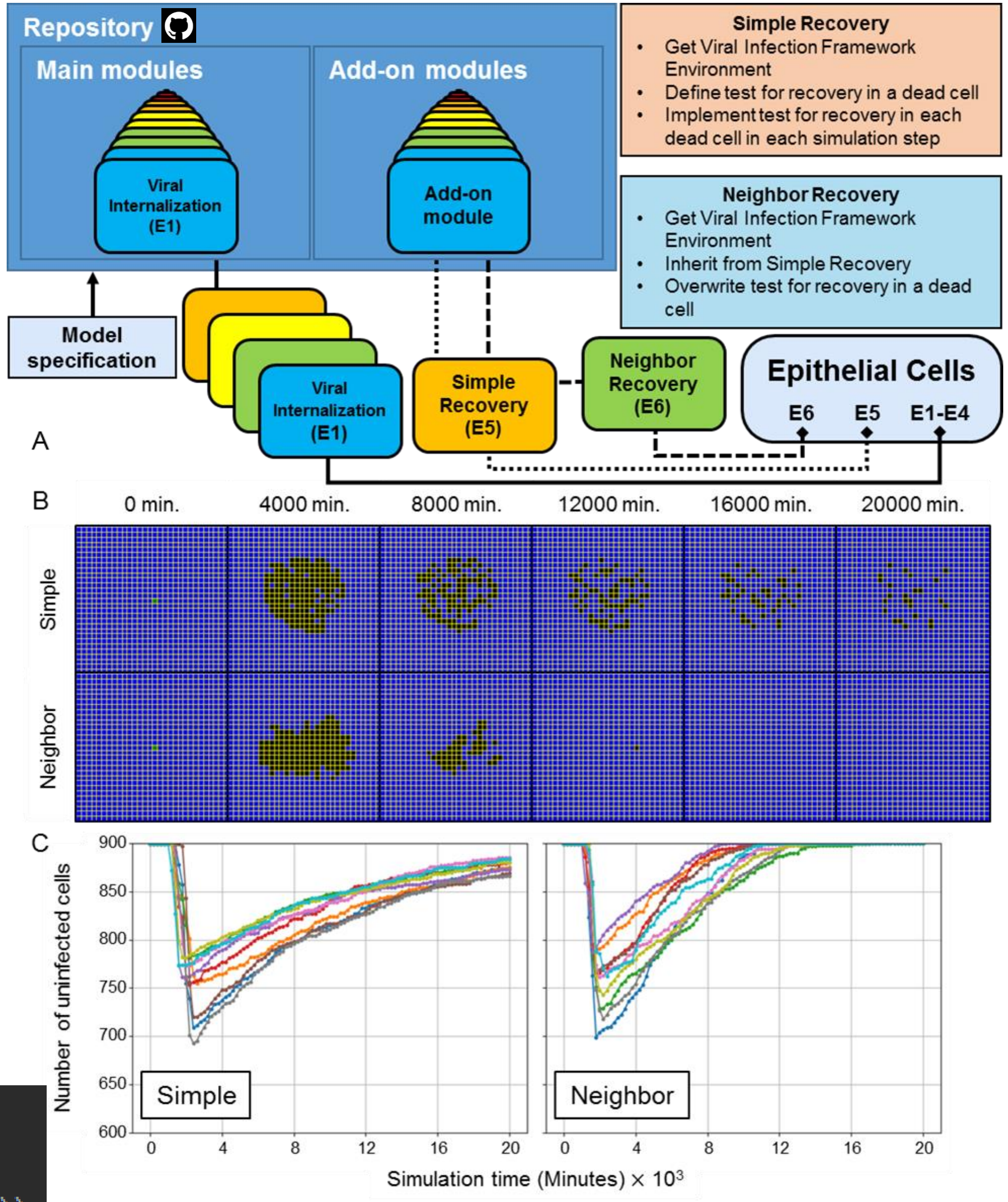




Collaborative, Concurrent Model Development

- Simulation framework is designed with *interchangeable*, *shareable*, and *extensible* model modules (architecture like the Python programming language)
- Simulation specification: load a set of model modules
- Built-in support for seamlessly downloading, adding, using and uploading add-on model modules
- Architecture prevents collision during concurrent development
- Framework and library are maintained on GitHub: collaborative public development

```
from Models.RecoveryNeighbor.RecoverySteppables import NeighborRecoveryDataSteppable

CompuCellSetup.register_steppable(steppable=NeighborRecoveryDataSteppable(frequency=1))

CompuCellSetup.run()
```



Group A recovery model: RecoverySimple
Dead cells resurrect with a fixed probability

(Add-on library) (Model Module) (Model Specification)

Models 〉 RecoverySimple 〉 RecoverySteppables.py

```python
from .RecoveryInputs import *
rec_steppable_key = "sprec_steppable"


class SimpleRecoverySteppable(SteppableBasePy):
    """
    Implements simple recovery
    """
    def __init__(self, frequency=1):
        super().__init__(self, frequency)
        self.num_recovered = 0

        self.rec_steppable_key = rec_steppable_key

    def start(self):
        # Post reference to self
        self.shared_steppable_vars[self.rec_steppable_key] = self

    def step(self, mcs):
        [self.recover_cell(cell) for cell in self.cell_list_by_type(self.DYING) if self.cell_recovers(cell)]

    def cell_recovers(self, _cell) -> bool:
        """
        Recovery criterion
        :param _cell: dead cell to test for recovery
        :return: True if cell recovers
        """
        return random.random() < recovery_rate * s_to_mcs

    def recover_cell(self, _cell):
        """
        Implement recovery
        :param _cell: dead cell to recover
        :return: None
        """
        _cell.type = self.UNINFECTED
        _cell.dict[ViralInfectionVTMLib.vrl_key] = False
        self.num_recovered += 1
```

Import model parameters defined in a different script

Get CompuCell3D's model specification features

Test for recovery at each step

Define a recovery criterion

Define what recovery means

Group B recovery model: RecoveryNeighbor
Extend recovery model by Group A with
neighbor-dependent recovery probability

Models 〉 RecoveryNeighbor 〉 RecoverySteppables.py

```python
from .RecoveryInputs import *
rec_steppable_key = "nbrec_steppable"

# Inherit from Simple Recovery model
sys.path.append(os.environ["ViralInfectionVTM"])
from Models.RecoverySimple.RecoverySteppables import SimpleRecoverySteppable, SimpleRecoveryDataSteppable


class NeighborRecoverySteppable(SimpleRecoverySteppable):
    """
    Implements neighbor-dependent recovery
    """
    def __init__(self, frequency=1):
        super().__init__(frequency)

        # Particularize SimpleRecoverySteppable
        self.rec_steppable_key = rec_steppable_key

    def cell_recovers(self, _cell) -> bool:
        """
        Recovery criterion
        :param _cell: dead cell to test for recovery
        :return: True if cell recovers
        """
        ca = sum([a for n, a in self.get_cell_neighbor_data_list(_cell) if n is not None and n.type == self.UNINFECTED])
        return random.random() < ca * recovery_rate * s_to_mcs
```

Import model parameters defined in a different script

Import recovery model by Group A

Copy the recovery model by Group A

Overwrite recovery criterion

Main simulation script

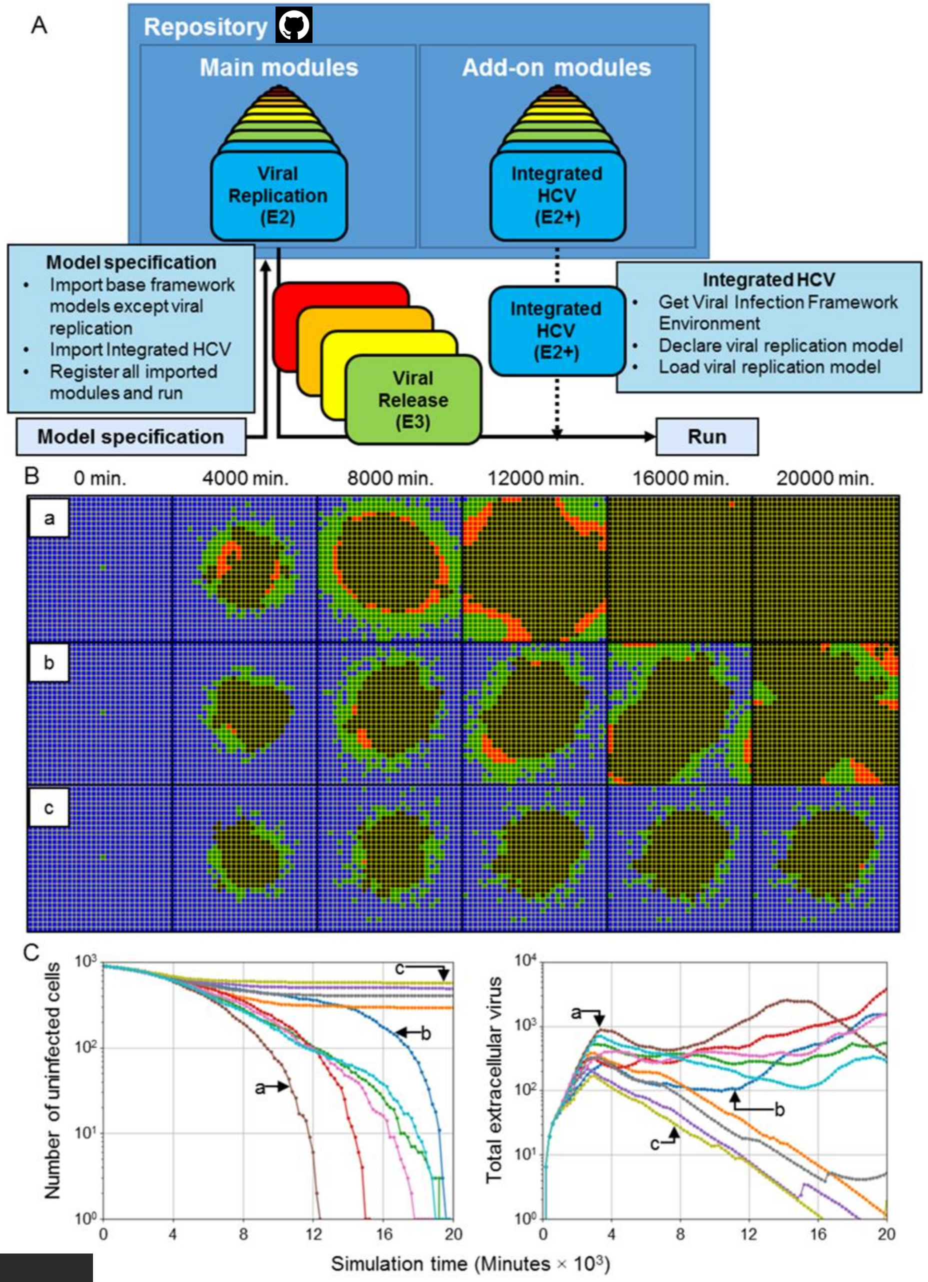
```python
from ViralInfectionVTMSteppables import oxidationAgentModelSteppable

CompuCellSetup.register_steppable(steppable=oxidationAgentModelSteppable(frequency=1))

from Models.RecoveryNeighbor.RecoverySteppables import NeighborRecoverySteppable

CompuCellSetup.register_steppable(steppable=NeighborRecoverySteppable(frequency=1))
```

Load from
main framework

Load from
add-on library

Building a Better Simulation Framework Together

- Continuous development of framework to better support community development
- Default framework is particular to SARS-CoV-2, but supports modeling other viruses



<u>Integrated Compartmental HCV subcellular model</u>
(Dahari, Ribeiro, Rice, Perelson, J Virol, 2007)

$$\frac{dR_P^{cyt}}{dt} = k_2 T_c + k_{Pout} R_P - k_I R_{ibo} R_P^{cyt} - k_{Pin} R_P^{cyt} - \mu_P^{cyt} R_P^{cyt} + n_{HCV} r_u U$$

$$\frac{dT_c}{dt} = k_I R_{ibo} R_P^{cyt} - k_2 T_c - \mu_{Tc} T_c$$

$$\frac{dP^{cyt}}{dt} = k_2 T_c - k_c P^{cyt}$$

$$\frac{dE^{cyt}}{dt} = k_c P^{cyt} - k_{Ein} E^{cyt} - \mu_E^{cyt} E^{cyt}$$

$$\frac{dR_P}{dt} = -k_3 R_P E + k_{4p} R_{Ids} + k_{Pin} R_P^{cyt} - (k_{Pout} + \mu_P) R_P$$

$$\frac{dR_{ds}}{dt} = k_{4m} R_{Ip} + k_{4p} R_{Ids} - k_5 R_{ds} E - \mu_{ds} R_{ds}$$

$$\frac{dE}{dt} = k_{Ein} E^{cyt} + k_{4m} R_{Ip} + k_{4p} R_{Ids} - k_3 R_P E - k_5 R_{ds} E - \mu_E E$$

$$\frac{dR_{Ip}}{dt} = k_3 R_P E - k_{4m} R_{Ip} - \mu_{Ip} R_{Ip}$$

$$\frac{dR_{Ids}}{dt} = k_5 R_{ds} E - k_{4p} R_{Ids} - \mu_{Ids} R_{Ids}$$

<u>Revised viral replication model</u>

$$\frac{dU}{dt} = Uptake - r_u U$$

$$n_{HCV} R = R_P^{cyt}$$

$$\frac{dP}{dt} = r_t R - r_p P$$

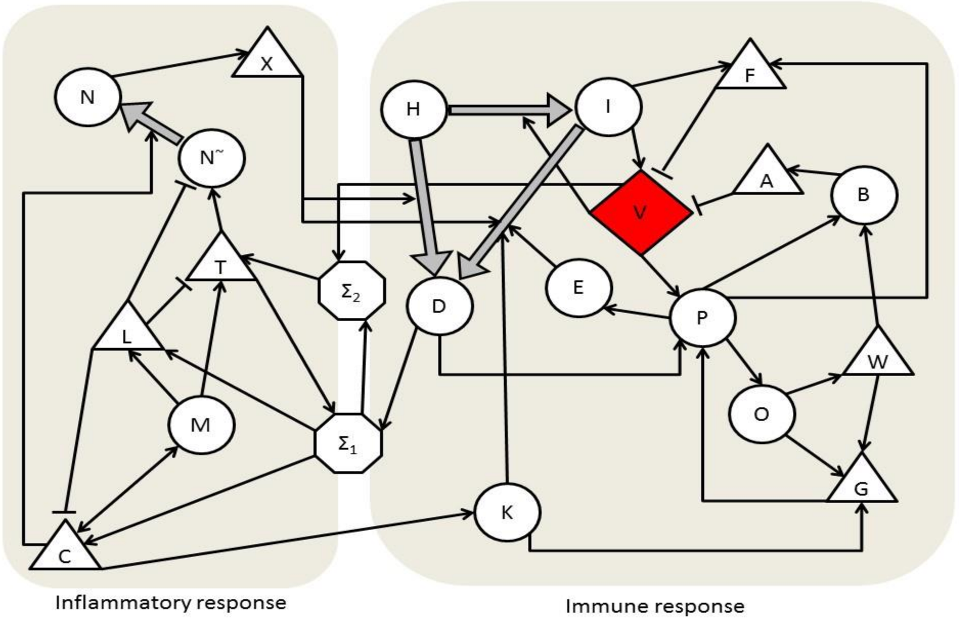
$$\frac{dA}{dt} = r_p P - Release$$

```python
class HCVIntegrator(ViralInfectionVTMSteppableBasePy):
    def __init__(self, frequency=1):
        super().__init__(frequency)
        self.set_viral_replication_model(hcv_viral_replication_model_string)
```

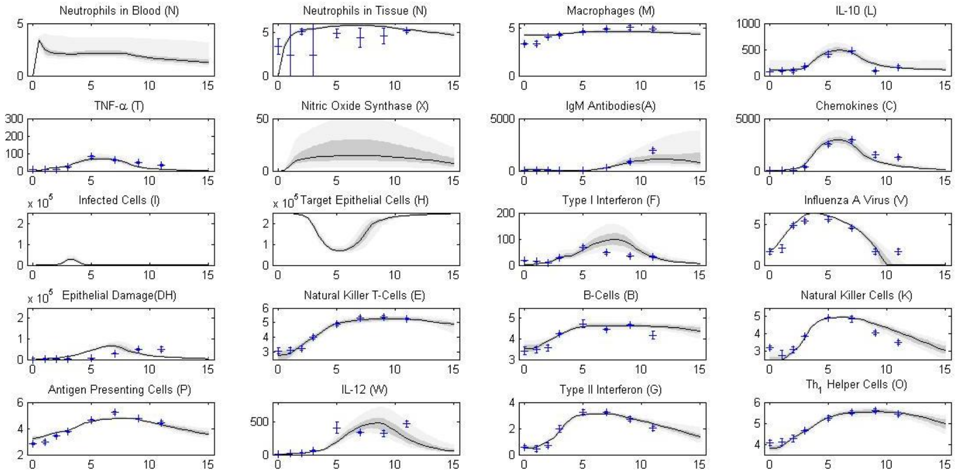
Extending the Framework: Enhanced Immune Response Modeling

- Collaboration with Profs. Ericka Mochan (Carlow U.) and G. Bard Ermentrout (U. Pittsburgh)
- Approach: generate a spatial model analogue of their calibrated ODE model of influenza and immune (innate and adaptive) response

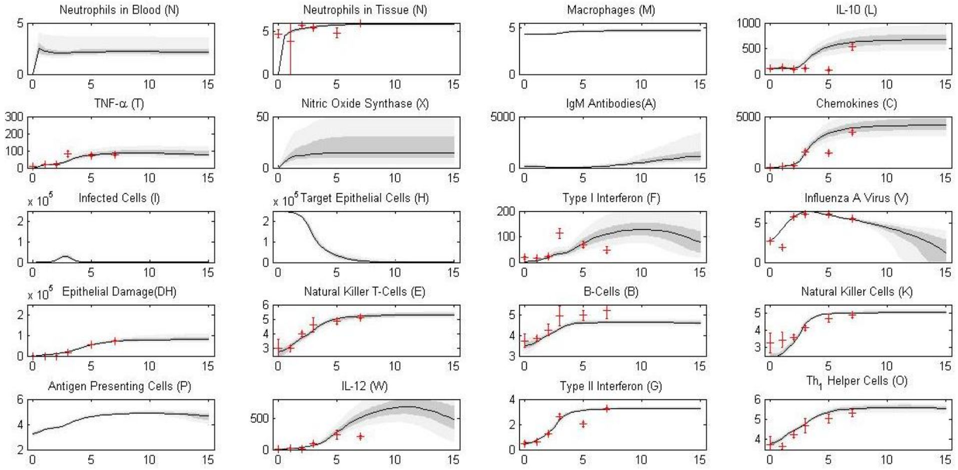
Label	Variable
TNF	T
IL-10	L
Chemokines	C
Macrophages	M
Blood neutrophils	Ñ
Tissue neutrophils	N
Reactive oxygen species	X
Target epithelial cells	H
Infected epithelial cells	I
Damaged epithelial cells	D_H
Virus	V
Type I interferon	F
Type II interferon	G
Natural killer cells	K
Antigen presenting cells	P
B cells	B
CD8+T cells	E
IL-12	W
CD4+T cells	O
Antibodies	A



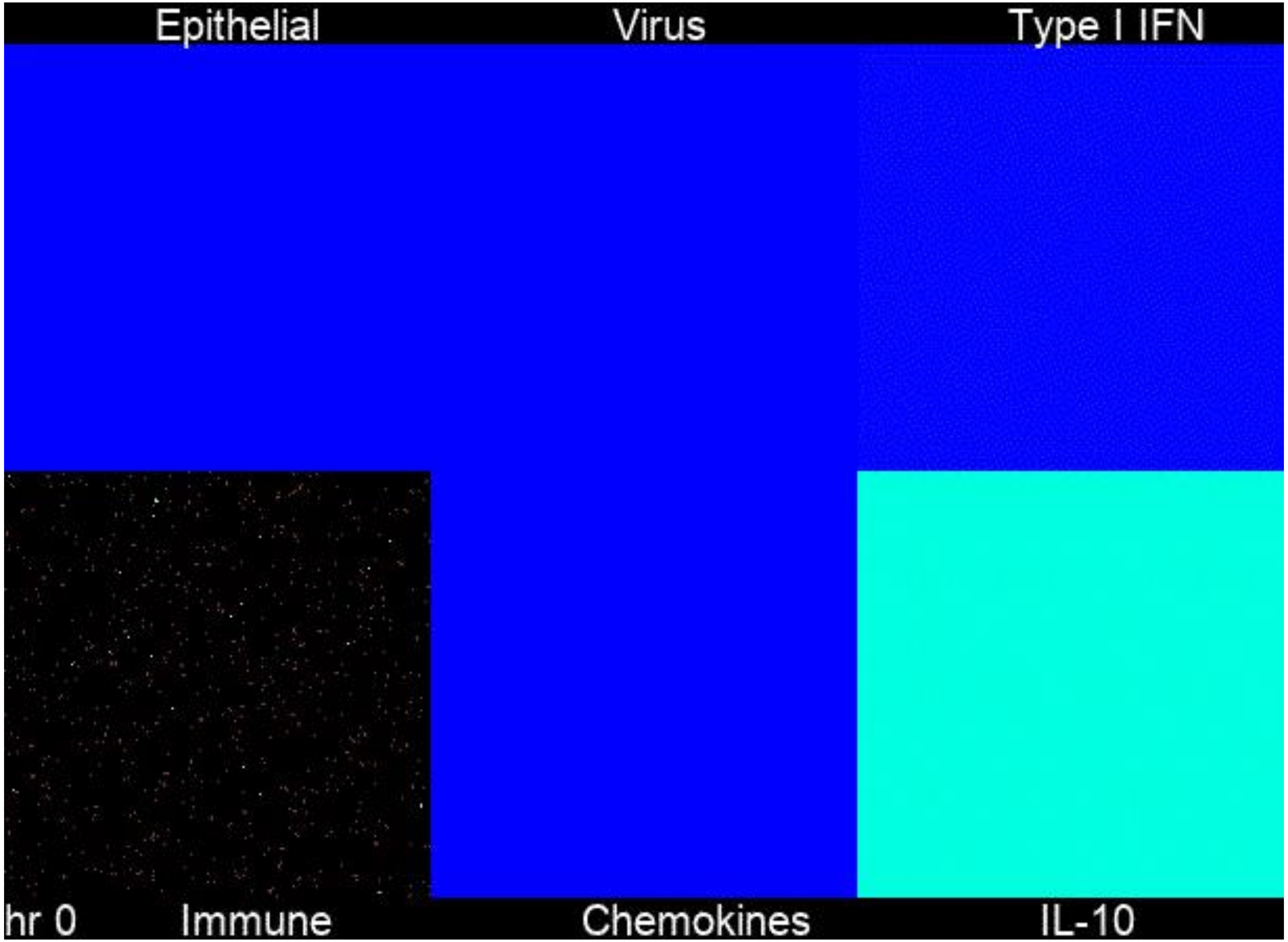
Nonlethal scenario



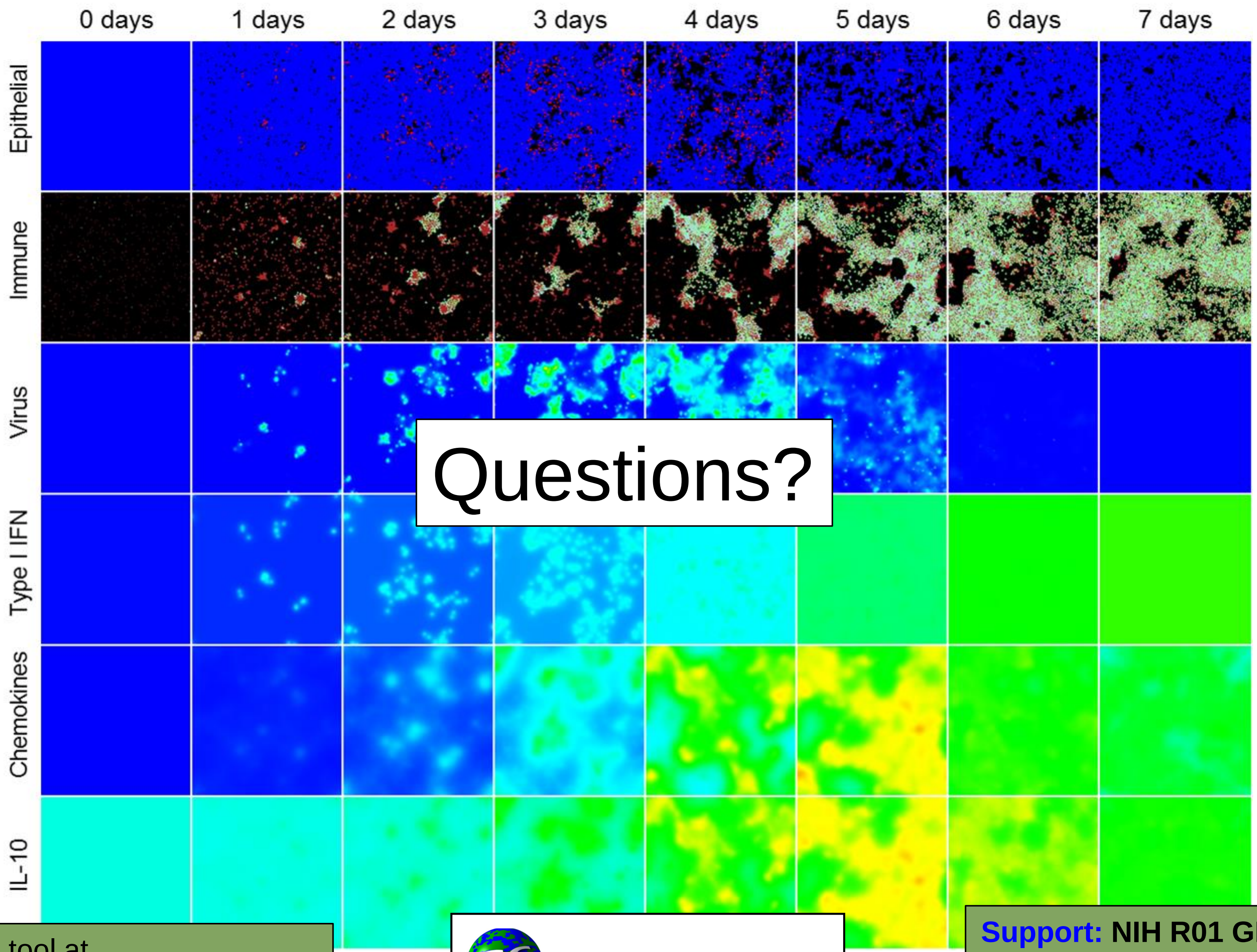
Lethal scenario



Price, Mochan et. al, J Theor. Biol., 2015


Epithelial
Virus
Type I IFN
hr 0
Immune
Chemokines
IL-10

tjsego@iu.edu



Questions?

Try the nanoHUB tool at
https://nanohub.org/tools/cc3dcovid19

compucell3d.org

Support: **NIH R01 GM122424, U01 GM111243, U24 EB028887, NSF 1720625**