Examples Based on ParaCells

1 The yeast autophagy model

1.1 Model description

For cell-centered hybrid models, we tested ParaCells with a computational model of molecular regulation and population dynamics for yeast autophagy (Jin and Lei, 2016). The model describes a multi-cell system, including a set of ODEs describing autophagy regulations in molecular level of yeast cells, and the population dynamics based on stochastic process. The population dynamics is presented by individual cell death and division in each cell cycle. The equations for autophagy regulations in a single cell are given below (refer Jin and Lei (2016) for detail of the model equations):

$$\begin{aligned}
\frac{d[aa^{(i)}]}{dt} &= k_{\text{in}}^{(i)}[aa_{\text{out}}]/v - k_{\text{out}}^{(i)}[aa^{(i)}] - \delta^{(i)}[aa^{(i)}] + f_{la}^{(i)}([\text{Atg8}_\tau^{(i)}])[aa_{\text{pro}}^{(i)}] \\
&\quad - f_p^{(i)}([aa]^{(i)})[aa^{(i)}] \qquad (1)\\
\frac{d[\text{TOR}^{(i)}]}{dt} &= f_m^{(i)}([aa^{(i)}], [\text{TOR}^{(i)}])(1 - [\text{TOR}^{(i)}]) \\
&\quad - g_m^{(i)}([\text{Atg1}^{(i)}])[\text{TOR}^{(i)}] \qquad (2)\\
\frac{d[\text{Atg1}^{(i)}]}{dt} &= f_u^{(i)}(1 - [\text{Atg1}^{(i)}]) - g_u^{(i)}([\text{TOR}^{(i)}])[\text{Atg1}^{(i)}] \qquad (3)\\
\frac{d[\text{Atg8}^{(i)}]}{dt} &= f_l^{(i)}([\text{Atg1}^{(i)}])(1 - [\text{Atg8}^{(i)}]) - g_l^{(i)}[\text{Atg8}^{(i)}] \qquad (4)\\
\frac{d[aa_{\text{pro}}^{(i)}]}{dt} &= f_p^{(i)}([aa^{(i)}]) - f_{la}^{(i)}([\text{Atg8}_\tau^{(i)}])[aa_{\text{pro}}^{(i)}] \qquad (5)\\
\frac{d[aa_{\text{out}}]}{dt} &= vN(t)\alpha(t) + \sum_{i=1}^{N(t)}(vk_{\text{out}}^{(i)}[aa^{(i)}] - k_{\text{in}}^{(i)}[aa_{\text{out}}]), \qquad (6)
\end{aligned}$$

where

$$\begin{aligned}
f_{la}^{(i)}([\text{Atg8}_\tau]) &= m_1^{(i)} + M_1^{(i)}\frac{[\text{Atg8}_\tau]^{n_1}}{K_1^{(i)n_1} + [\text{Atg8}_\tau]^{n_1}}\\
f_p^{(i)}([aa]) &= k_p^{(i)} + K_p^{(i)}\frac{[aa]}{K_6^{(i)} + [aa]}\\
f_m^{(i)}([aa], [\text{TOR}]) &= M_2^{(i)}\frac{[aa]^{n_2}}{K_2^{(i)}([\text{TOR}])^{n_2} + [aa]^{n_2}}\\
K_2^{(i)}([\text{TOR}]) &= m_3^{(i)} + M_3^{(i)}\frac{[\text{TOR}]^{n_3}}{K_3^{(i)n_3} + [\text{TOR}]^{n_3}}
\end{aligned}$$

$$
\begin{aligned}
g_m^{(i)}([\text{Atg1}]) &= m_4^{(i)} + M_4^{(i)} \frac{[\text{Atg1}]^{n_4}}{K_4^{(i)n_4} + [\text{Atg1}]^{n_4}} \\
g_u^{(i)}([\text{TOR}]) &= k_u^{(i)} \frac{[\text{TOR}]^{n_5}}{K_5^{(i)n_5} + [\text{TOR}]^{n_5}} \\
f_l^{(i)}([\text{Atg1}]) &= k_l^{(i)}[\text{Atg1}].
\end{aligned}
$$

In addition, a cell can either divide or die in each cell cycle, so that the population dynamics is described with an equation below:

$$
\begin{cases}
N(t+\Delta t) &= N(t) - \sum_{i=1}^{N(t)} I_{\text{death}}^{(i)}(t) + \sum_{i=1}^{N(t)} I_{\text{division}}^{(i)}(t), \\
N(0) &= N_0
\end{cases} \tag{7}
$$

The cell fate decision is represented as a random process:

$$
I_{\text{death}}^{(i)}(t) = \begin{cases} 1, & \text{if } \xi^{(i)}(t) \in [0, r_{\text{death}}^{(i)}(t)\Delta t], \\ 0, & \text{otherwise,} \end{cases} \tag{8}
$$

and

$$
I_{\text{division}}^{(i)}(t) = \begin{cases} 1, & \text{if } t = nT \ (n \in N), \text{and } \eta^{(i)}(t) \in [0, r_{\text{division}}^{(i)}(t)], \\ 0, & \text{otherwise,} \end{cases} \tag{9}
$$

where the possibilities of cell fate decision are affected by the status of extracellular environment as shown below:

$$
r_{\text{death}}^{(i)}(t) = R_{\text{dea}} \frac{K_{\text{dea}}^2}{K_{\text{dea}}^2 + [aa^{(i)}(t)]^2}, \tag{10}
$$

$$
r_{\text{division}}^{(i)}(t) = R_{\text{div}} \frac{[aa^{(i)}(t)]^2}{K_{div}^2 + [aa^{(i)}(t)]^2}. \tag{11}
$$

Moreover, we extend the original models with position information in the cellular definition, and update them in each step:

$$
\text{Pos}_x^i(t+\Delta t) = \text{Pos}_x^i(t) + 2(\lambda^{(i)}(t) - 0.5) D_{\text{move}} \Delta t, \tag{12}
$$

$$
\text{Pos}_y^i(t+\Delta t) = \text{Pos}_y^i(t) + 2(\mu^{(i)}(t) - 0.5) D_{\text{move}} \Delta t. \tag{13}
$$

where the $\lambda^{(i)}(t), \mu^{(i)}(t)$ are random variables with uniform distribution in $[0, 1]$, and D_{move} is a step size.

Here, we applied ParaCells of this model with crosstalk between molecular and population dynamics. We simulated the cell fate decision with periodically changes in the nutrition concentration input flux. As a result, we can monitor the nutrient distribution and population dynamics of the multi-cellular system when affected by starvation and autophagy.

1.2 Implementation

We simulated the hybrid model with initial state varied from 100 to 500 cells, each cell is assigned with given parameters. In the simulation, molecular level model in each cell executes independently, including interactions between cells and extra-cellular environment. Cells may die or divide in each cell cycle, so that we can track the system population dynamics.

In ParaCells, we used the *Custom Functions* to optimize the performance. Moreover, we implemented complex operations such as death and proliferation based on ParaCells pre-defined data structures and algorithms. Here, we implemented molecular level model that describes single cell kinetics with *Custom Functions*. Each cell is assigned with a unique set of parameters. During simulation, we continuously update and check the status of each cell, and determine whether it should perform death or proliferation at the end of each cell cycle.

We begin the simulation with the initialization *Custom Function*, by which a unique set of parameters is generated randomly for each cell. ParaCells provide built-in parallel random number generators, and users can use this feature to introduce diversity among individuals. By accessing the methods provided by *Environment* object, we can generate with uniform or poisson distribution random numbers.

The rest of *Custom Functions* implement molecular level dynamics in a single cell. They are launched several times in one cell cycle, and all data should be synchronized between launches. We can store these data to *Custom Object* provided by ParaCells for both usability and performance. *Custom Object* are designed for storing flexible data structure and operations in *Custom Functions*. These data can only be accessed in *Custom Functions*. To specify the data structure in *Custom Object*, we need to define a class inherited from a template class *ParaCellsObject*. A required function `Cell::proliferate()` is implemented for the division behavior. In the implementation of this model, the code defining *Custom Object* is presented below:

```
// code snippet
class C: public ParaCellsObject
{
private:
// Define concentrations and model parameters
float aa, TOR, Atg1, Atg8, aapro;
float kin;
float kout;
...

public:
// Object constructor
__device__ C(Environment *env)
{
// Init concentrations
```

```
aa = 0;
TOR = 0;
Atg1 = 0;
Atg8 = 0;
aapro = 1;
memset(Atg8_delay, 0, sizeof(Atg8_delay));

// Init random parameters
kin = generateParameter(_kin, env);
kout = generateParameter(_kout, env);
delta = generateParameter(_delta, env);
...

// Other initialization operations
// ...
}

// Describes the behaviour when the cell containing this object
// undergoes proliferation process
__device__ ParaCellsObject *proliferate(Environment *env)
{
C *rtn = new C(env);
rtn->init(aa, TOR, Atg1, Atg8, aapro);

return rtn;
}

// Generate parameter value from a reference(original) value
__device__ float generateParameter(float originalValue,
Environment *env)
{
float random = env->getUniformRandom() * 0.3 - 0.15;
return originalValue * (1.0 + random);
}

// Simulate one step of a cell
__device__ void oneStep(Cell *cell, Environment *env)
{
// Operations
// ...
}

// Other member functions
// ...
};
```

We implement the molecular level dynamics in the *Custom Functions*. Initially, the *Custom Object* should be firstly created and assigned for individual cell. We put these logics in the initialization *Custom Function*:

```
// code snippet
__device__ void init(Cell *cell, Environment *env)
{
// Create Custom Object defined earlier
C *obj = new C(env);

// Assign the Custom Object into the current cell
cell->setCustomObject(obj);

// Other initialization operations
// ...
}
```

The *Custom Object* is obtained before the model calculation, and we use it to get and update the status of individual cells. A simplified code snippet is shown below:

```
// code snippet
__device__ void oneStep(Cell *cell, Environment *env)
{
// Get "Custom Object" belongs to the current cell
C *obj = (C *)cell->getCustomObject();

for (int i = 0; i < STEP_NUM/10; i++)
{
// Invoke the function inside "Custom Object" to simulate one step
obj->oneStep(cell, env);
}

// Other operations
// ...
}
```

ParaCells provides intuitive pre-defined functions to simulate multi-cellular systems, such as cell death and proliferation. In *Custom Functions*, operations related to cell fate decision could be implemented by calling the corresponding function on a single cell. For instance, in order to remove a cell from the system, users can apply the pre-defined function below:

```
// code snippet
cell->die();
```

Similarly, to perform cell division, we only need to call the function `proliferate` and specify cell attributes of two daughter cells[1].

```
// code snippet
cell->proliferate();
```

Then, we need to register and make *Custom Functions* callable. The overall code in a ParaCells project is split into sequential part and *Custom Functions*. The project is launched from the main entrance function in the sequential part and only registered *Custom Functions* can be invoked from the sequential part. Hence, we need to register the initialization and helper *Custom Functions*. The rest of the *Custom Functions* do not need to be registered, because they are called by other *Custom Functions*, instead of from the sequential part.

```
// code snippet
__device__ CustomFunction customFunctions[2] = { init, oneStep };
```

In addition to the molecular level dynamics, population models and environment controls describe the system dynamics at a higher level. These higher level operations are mostly independent from the single cell, so that the logic should be implemented outside the *Custom Functions*, and work as sequential code in ParaCells project. A simplified version of sequential code is presented below. In the models, interactions between single cell and population levels are realized by exchanges of amino acids between cell and extracellular environment. Hence, in the implementation we defined two environment attributes for the amino acids concentration and feedback from all cells. The concentration attribute is continuously updated at population level whenever the cells status are updated, while the feedback attribute is updated after each cells cycle, following which they are used for affecting cells fate decision in the next cells cycle.

Finally, to implement the periodic changes from environment nutrient supply, we defined the environment attribute `alpha_t`. By changing the its value between cells cycles, every individual cell and the overall population can perform different dynamics. As a result, we are able to track and analyze the results of long time dynamics after simulation. These procedures are shown below.

```
// code snippet
void work()
{
CellEnv cellEnv(MAXIMUM_CELL_NUM, 1, 2);

cellEnv.addCellAttribute("accumulator");
cellEnv.addEnvironmentAttribute("aaout", 0);
cellEnv.addEnvironmentAttribute("accumulator", 0);

cellEnv.addCells(INITIAL_CELL_NUM, 0);
```

[1]For *Custom Object*, the behavior of object division has been described in the definition of *Custom Object* classes.

```
for (int i = 0; i < CYCLE_NUM; i++)
{
// Modify the value of parameter(alpha_t)
if (i % 3 == 0)
{
if ((i / 3) % 2 == 0) alpha_t = 0.125;
else alpha_t = 0;
}

for (int j = 0; j <= 10; j++)
{
// Simulate molecular-level dynamics
cellEnv.updateAllCellsWithoutFateDecision(1);

// Environment controls, relying on parameter(alpha_t)
// and feedbacks(from single cells)
updateAaout(cellEnv);
}

// Determine cell proliferation
cellEnv.updateAllCells(2);

// Data collecting operations
// ...
}

// Other operations
// ...
}
```

2 The DNA damage response model

2.1 Model description

We tested ParaCells with an single-cell based model of how programmed-cell-death 5 (PDCD5) interacts with the P53-MDM2 oscillation in regulating cell fate decision in response to DNA damage (Zhuge *et al.*, 2016). The model consists of a set of ODEs describing chemicals reactions in a single cell. The equations for a single cell are given below (refer Zhuge *et al.* (2016) for detail of the model equations):

$$v_{\mathrm{ATM}}(t) = \frac{v_0 e^{t/12}}{(1+e^{t/12})(1+e^{(t-t_c)/24})} \tag{14}$$

$$
\begin{aligned}
P(t) &= 0.1 + \frac{(P_0 - 0.1)e^{(t-300)/100}}{(1+e^{(t-300)/100})(1+e^{(t-t_c)/100})} && (15)\\
\frac{d[\mathrm{p53}]}{dt} &= v_{\mathrm{p53}}([\mathrm{Mdm2}^{395\mathrm{P}}_{\mathrm{cyt}}], [\mathrm{ATM}^*]) \\
&\quad - d_{\mathrm{p53}}([\mathrm{Mdm2}_{\mathrm{nuc}}], P(t))[\mathrm{p53}] && (16)\\
\frac{d[\mathrm{p53}]}{dt} &= v_{\mathrm{Mdm2}}([\mathrm{p53}]) - k_{\mathrm{in}}[\mathrm{Mdm2}_{\mathrm{cyt}}] + k_{\mathrm{out}}[\mathrm{Mdm2}_{\mathrm{nuc}}] \\
&\quad - k_p([\mathrm{ATM}^*])[\mathrm{Mdm2}_{\mathrm{cyt}}] + k_q[\mathrm{Mdm2}^{395\mathrm{P}}_{\mathrm{cyt}}] \\
&\quad - d_{\mathrm{Mdm2}}[\mathrm{Mdm2}_{\mathrm{cyt}}] && (17)\\
\frac{d[\mathrm{Mdm2}_{\mathrm{nuc}}]}{dt} &= k_{\mathrm{in}}[\mathrm{Mdm2}_{\mathrm{cyt}}] - k_{\mathrm{out}}[\mathrm{Mdm2}_{\mathrm{nuc}}] \\
&\quad - f(t) d_{\mathrm{Mdm2}}[\mathrm{Mdm2}_{\mathrm{nuc}}] && (18)\\
\frac{d[\mathrm{Mdm2}^{395\mathrm{P}}_{\mathrm{cyt}}]}{dt} &= k_p([\mathrm{ATM}^*])[\mathrm{Mdm2}_{\mathrm{cyt}}] - k_q[\mathrm{Mdm2}^{395\mathrm{P}}_{\mathrm{cyt}}] \\
&\quad - g_0 d_{\mathrm{Mdm2}}[\mathrm{Mdm2}^{395\mathrm{P}}_{\mathrm{cyt}}] && (19)\\
\frac{d[\mathrm{ATM}^*]}{dt} &= v_{\mathrm{ATM}}(t) - d_{\mathrm{ATM}}([\mathrm{p53}])[\mathrm{ATM}^*], && (20)
\end{aligned}
$$

where

$$
\begin{aligned}
v_{\mathrm{p53}}([\mathrm{Mdm2}^{395\mathrm{P}}_{\mathrm{cyt}}], [\mathrm{ATM}^*]) &= \bar{v}_{\mathrm{p53}} \left((1-\rho_0) + \rho_0 \frac{[\mathrm{ATM}^*]^{s_0}}{K_0^{s_0} + [\mathrm{ATM}^*]^{s_0}} \right) \\
&\quad \times \left((1-\rho_1) + \rho_1 \frac{[\mathrm{Mdm2}^{395\mathrm{P}}_{\mathrm{cyt}}]^{s_1}}{K_1^{s_1} + [\mathrm{Mdm2}^{395\mathrm{P}}_{\mathrm{cyt}}]^{s_1}} \right) \\
d_{\mathrm{p53}}([\mathrm{Mdm2}_{\mathrm{nuc}}], P(t)) &= \bar{d}_{\mathrm{p53}} \left((1-\rho_2) + \rho_2 \frac{[\mathrm{Mdm2}_{\mathrm{nuc}}]^{s2}}{K_2(P(t))^{s_2} + [\mathrm{Mdm2}_{\mathrm{nuc}}]^{s_2}} \right) \\
K_2(p) &= \bar{K}_2 \left((1-r_1) + r_1 \frac{(\alpha_1 P)^{m_1}}{1 + (\alpha_1 P)^{m_1}} \right) \\
v_{\mathrm{Mdm2}}([\mathrm{p53}]) &= \bar{v}_{\mathrm{Mdm2}} \left((1-\rho_3) + \rho_3 \frac{[\mathrm{p53}]^{s_3}}{K_3^{s_3} + [\mathrm{p53}]^{s_3}} \right) \\
k_p([\mathrm{ATM}^*]) &= \bar{k}_p \left((1-\rho_4) + \rho_4 \frac{[\mathrm{ATM}^*]^{s_4}}{K_4^{s_4} + [\mathrm{ATM}^*]^{s_4}} \right) \\
f(t) &= 1 + r_2 P(t) \\
d_{\mathrm{ATM}}([\mathrm{p53}]) &= \bar{d}_{\mathrm{ATM}} \left((1-\rho_5) + \rho_5 \frac{[\mathrm{p53}]^{s_5}}{K_5^{s_5} + [\mathrm{p53}]^{s_5}} \right).
\end{aligned}
$$

Here, we used ParaCells to study a group of many cells, assuming the P53 dynamics of each cell is described by the same set of ODEs, but the parameter values can be different. As a result, we can examine how different cells with different parameter yield different P53 dynamics upon DNA damage.

2.2 Implementation

We applied ParaCells to simulate the single-cell model for a large number of independent runs, each with different value of parameter P_0 for the PDCD5 level in order to track the P53 dynamics under different PDCD5 level.

In ParaCells, since the above model equations were applied to a single cell, we mapped each instance of simulations to one *Cell* in ParaCells, so that model equations can be implemented in *Custom Functions*. These functions are automatically applied to every single cell in the system during simulation. In the model, we set the only changeable parameter P_0 as the attribute in a cell. Thus, different PDCD5 level in each cell is controlled by varying the cell attribute P_0.

The *Custom Functions* defined in this example include two parts. First, we initialized cell attributes by setting P_0 value to a cell. When *Custom Functions* are executed on each cell, a unique cell id ranging from 0 to `cellNum-1` is attached to the cell, and the value P_0 is assigned to cells according to the cell id. Function definition and its operations is shown below:

```
// code snippet
__device__ void initP0(Cell *cell, Environment *env)
{
float P0 = (cell->getCellId() + 1.0f) / 100000.0f;
cell->setAttribute("P0", P0);
}
```

The second part includes implementations of dynamical equations within a single cell. In doing this, the coding processes are similar to those in traditional sequential programming in C++, but add the `__device__` identifier to functions declaring to show that they are *Custom Functions* running on GPU. Moreover, a helper function `work_kernel` is added to the *Custom Functions* to extract necessary information before starting the simulation. The code of this function is shown below. It receives two parameters of type `Cell*` and `Environment*`, respectively, which represent the current simulated object in the multi-cellular system. The helper function retrieves P_0 from the *Cell attribute*, and then launches the dynamical process simulation by calling the model entrance function. As a result, multiple identical simulations with different P_0 can be processed in GPU in parallel executed threads.

```
// code snippet
__device__ float vp53(float Mdm2_395P_cyt, float ATM_star);
__device__ float vCytoC(float killer, float C3, float P);
__device__ void cell_fate_decision(float P0, float t, float dt,
float p53, float &killer, float &CytoC, float &C3, float &arrester,
float &CytoCm);
__device__ void p53_Mdm2_module(float P0, float t, float dt,
float &p53, float &Mdm2_cyt, float &Mdm2_nuc, float &Mdm2_395P_cyt,
float &ATM_star);
...
```

```
__device__ void work_kernel(Cell *cell, Environment *env)
{
// Get parameter P0 from cell attribute
float P0 = cell->getAttribute("P0");

// Time
float t;
float dt;

// Concentrations
float p53, Mdm2_cyt, Mdm2_nuc, Mdm2_395P_cyt, ATM_star;
float killer, CytoC, C3, arrester, CytoCm;

// Init
t = 0;
dt = _dt;
p53 = 0;
...

// Simulate 4800 cell cycles
for (int i = 1; i <= 4800; i++)
{
// Cell fate decision & P53-Mdm2 model
cell_fate_decision(P0, t, dt, p53, killer, CytoC, C3,
arrester, CytoCm);
p53_Mdm2_module(P0, t, dt, p53, Mdm2_cyt, Mdm2_nuc,
Mdm2_395P_cyt, ATM_star);

// Time step
t += dt;
}

cell->setAttribute("p53", p53);
}
```

Similarly, we need to register needed *Custom Functions* in order to make them callable from the sequential part of code:

```
// code snippet
__device__ CustomFunction customFunctions[2] =
{ initP0, work_kernel };
```

The sequential code `work` below shows how the registered *Custom Functions* are called. In the `work` function, we first specified the number of cells according to our needs in the simulation. Here, we focus at the dynamics of the overall system size rather than the detail concentrations of each cell, hence we only

need to add the cell attribute P_0. The initialization *Custom Function* is called to assign the P_0 values, followed by the calling of the helper *Custom Function* – `work_kernel` for actual dynamical process simulation. At the end, the number of cells at that point are recorded.

```
// code snippet
void work()
{
CellEnv cellEnv(MAXIMUM_CELL_NUM, 2, 0);

cellEnv.addCells(MAXIMUM_CELL_NUM);
cellEnv.addCellAttribute("P0");
cellEnv.addCellAttribute("p53");

cellEnv.updateAllCellsWithoutFateDecision(0);
cellEnv.updateAllCellsWithoutFateDecision(1);

// Other operations
// ...
}
```

References

Jin, H. and Lei, J. (2016). A hybrid model of molecular regulation and population dynamics for yeast autophagy. *J Theor Biol*, **402**, 45–53.

Zhuge, C., Sun, X., Chen, Y., and Lei, J. (2016). PDCD5 functions as a regulator of p53 dynamics in the DNA damage response. *J Theor Biol*, **388**, 1–10.