Materials and methods

1. Simulation domain

The simulation domain is a two-dimensional (*x, y*) and micro-scale space (~10^{-6} m) defined by the user. In it is solved the diffusion of soluble components considered, which are the substrates and products of the microbial activity. The simulation domain is divided in three different zones: the aggregate (biofilm or granule), the boundary layer and the bulk liquid (**Figure 1.1**).

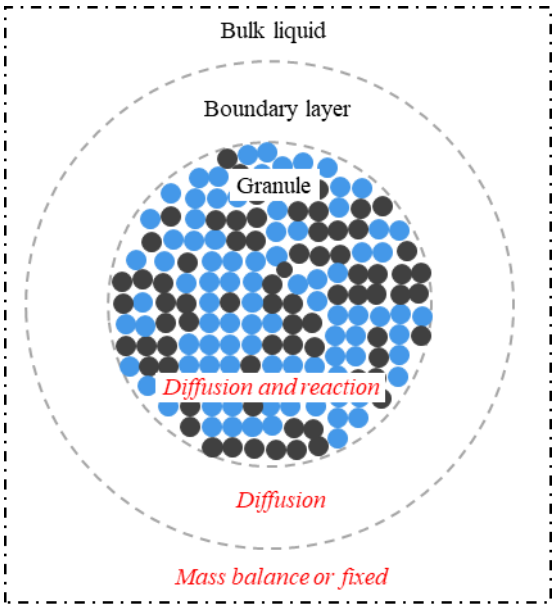

Figure 1.1. Zones of the simulation domain

In granular region (**Figure 1.1**) is where the microbial activity occurs. The microbes are represented like circles with a specific radius. When they grow, they push each other increasing the height of the biofilm or the radius of granule. Biofilm and granule increase with the microbial growth until a maximum height and radius defined by the user. Diffusion of soluble components occurs through the aggregate at the same time that they are consumed or produced by the microorganisms. The boundary layer is the surrounding space of the biofilm and granule defined to model the gradient of concentrations between the bulk liquid and the surface of the microbial aggregate. Only the diffusion of the soluble components is resolved in this space. At the outside of the boundary layer, identified as bulk liquid, it is considered that the gradient of concentration of all soluble species is negligible. The concentrations are homogenous in space, therefore there is not space discretization and the concentrations in the bulk liquid are only function of time. These can be fixed by the user or also can be calculated fixing a specific hydraulic retention time (HRT) and assuming that the activity of the aggregate modelled is representative of the whole reactor.

In microbial aggregate and boundary layer zones, diffusion equation must be solved. For this, the space is discretized in two coordinates (*x* and *y*). In an *Excel* sheet read by MATLAB function, the length of the domain over each *x* and *y* coordinates are defined. Also, the number of nodes in *x* and *y* (N_x, N_y), and their sizes (h_x, h_y). Each of the microbe that forms the biofilm or granule is an individual with its own kinetic parameters and specific mass and position in the aggregate. As mentioned above, each microbe acts like a small reactor that grows function of the local environmental condition of the node where is its centre (**Figure 1.2**).

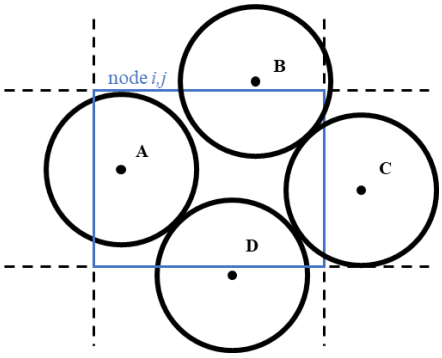

Figure 1.2. Hypothetical position of microbes in a node. Only "microbe A" belongs to node *i, j* because its centre is inside the node

2. Diffusion-reaction of soluble components

The soluble components diffuse in the water and trough the granule where the reaction occurs. To describe it, the Fick´s second law equation is integrated over the time (*t*) and in the two dimensions (*x* and *y*) for each of the soluble components considered. To it, a term of reaction is added function of the position in space and time (*R(x,y,t)*) (equation 2.1).

$$\frac{\partial}{\partial t}\phi(x,y,t) = \mathbb{D}\cdot\nabla^2_{xy}\phi(x,y,t) + R(x,y,t) \qquad [2.1]$$

Where $\phi(x,y,t)$ refers to the concentration of a soluble component in a position of the simulation domain and in a time step and $\mathbb{D}$ refers to the effective coefficient of diffusion. For simplicity, the notation used from now on is presented in equation 2.2.

$$\phi^n_{i,j} = \phi(x_i, y_j, t_n) \qquad [2.2]$$

To solve the equation 2.1, the implicit Crank-Nicholson method, which is unconditionally stable, is used to discretize in time the diffusion term. For the reaction term, an explicit forward Euler formula is used (equation 2.3). This can be done, as the reaction process has a much slower time scale than the diffusion [1, 2].

$$\frac{\phi^{n+1}_{i,j} - \phi^n_{i,j}}{h_t} = \mathbb{D}\cdot\frac{1}{2}\left[\nabla^2\phi^{n+1}_{i,j} + \nabla^2\phi^n_{i,j}\right] + R(\phi^n_{i,j}) \quad n \in 1\ldots N_t \qquad [2.3]$$

Where h_t refers to the time step and N_t to the total number of time steps. To approximate the Laplacian of equation 2.3 (∇^2) in a two-dimensional space (x,y), this is discretized using the finite-difference method (equation 2.4), where *h* is the grid size ($h = \Delta x = \Delta y$).

$$\nabla^2\phi^n_{i,j} = \frac{\phi^n_{i-1,j}+\phi^n_{i+1,j}+\phi^n_{i,j-1}+\phi^n_{i,j+1}-4\phi^n_{i,j}}{h^2} \qquad [2.4]$$

Equation 2.4 is also known as discrete Laplacian. Due to diffusion-reaction equation is solved using the convolution method, discrete Laplacian is given as the following kernel (equation 2.5), and Laplacian approximation is re-written in matrix from (equation 2.6), where the convolution of $[L]$ and $\phi^n_{i,j}$ is denoted using the symbol *.

$$[L] = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix} \qquad [2.5]$$

$$\nabla^2 \phi_{i,j}^n = \frac{1}{h^2}([L] * [\phi^n]) \quad [2.6]$$

Where $[\phi^n]$ is the concentration of soluble compounds ($\phi_{i,j}^n$) defined in matrix form (equation 2.7).

$$[\phi^n] = \begin{pmatrix} \phi_{1,1}^n & \phi_{1,2}^n & \cdots & \phi_{1,N_y}^n \\ \phi_{2,1}^n & \phi_{2,2}^n & & \vdots \\ \vdots & & \ddots & \vdots \\ \phi_{Nx,1}^n & \cdots & \cdots & \phi_{N_x,N_y}^n \end{pmatrix} \in M_{N_x \times N_y} \quad [2.7]$$

For discrete, 2-dimensional variables (A and B), equation 2.8 defines the convolution of A and B (i.e., A∗B). Convolution satisfies the distributive property, multiplicative identity and associative property with scalar multiplication. These properties are essential to rearrange the diffusion-reaction equation properly.

$$A[x] * B[x] = \sum_{k=-\infty}^{+\infty} A[x] \cdot B[x-k] \quad [2.8]$$

Then diffusion-reaction equation (equation 2.1) can be re-written in form of matrixes system (equation 2.9), where α is a constant defined for each soluble component (equation 1210) and $[I_k]$ is the so-called *identity kernel* or *do-nothing convolution kernel* (equation 2.11).

$$([I_k] - \alpha \cdot [L]) * [\phi^{n+1}] = ([I_k] + \alpha \cdot [L]) * [\phi^n] + R([\phi^n]) \cdot h_t \quad [2.9]$$

$$\alpha = \frac{\mathbb{D} \cdot h_t}{2 \cdot h^2} \quad [2.10]$$

$$[I_k] = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad [2.11]$$

2.1. Boundary conditions of simulation domain

To solve equation 2.9 in all the simulation domain, it is necessary to define the boundary conditions of the problem. In this case, a unique boundary condition type is defined – the concentrations at outside of the granule are defined by the conditions of the bulk liquid (**Figure 2.1**). This is modelled implementing a Dirichlet boundary condition, in which is imposed the concentration of the bulk liquid (γ) at the extreme of the boundary layer (equation 2.12) and bulk liquid.

$$\phi_{-1,j}^n = \phi_{N_x+1,j}^n = \phi_{i,-1}^n = \phi_{i,N_y+1}^n = \gamma \quad [2.12]$$

To implement Dirichlet boundary condition in this system, first it is defined which region of the simulation domain is considered bulk liquid (where no diffusion nor reaction occurs, and its concentration is established by mass balance of reactor, see section 2.3), and which one is considered *diffusion region* (comprising boundary layer and granular region). Then, concentration matrix $[\phi^n]$ is modified properly (equation 2.13) – those nodes in which are considered *diffusion region* have the corresponding concentration value ($[\phi^n]$), and that ones in which are considered *no-diffusion region* have the boundary value (i.e., concentration of bulk liquid, γ).

$$[\phi^n]^\gamma = diffR \circ [\phi^n] + \neg diffR \cdot \gamma \quad [2.13]$$

Where $diffR$ is a logical matrix with 1 (or true) in those nodes of *diffusion region*, and 0 (or false) in those nodes of *no-diffusion* region; $\neg diffR$ is the inverse of $diffR$. Then, the diffusion-reaction equation including the boundary conditions imposed over the simulation domain is written in equation 2.14.

$$([I_k] - \alpha \cdot [L]) * [\phi^{n+1}] = ([I_k] + \alpha \cdot [L]) * [\phi^n]^\gamma + R([\phi^n]) \cdot h_t \quad [2.14]$$

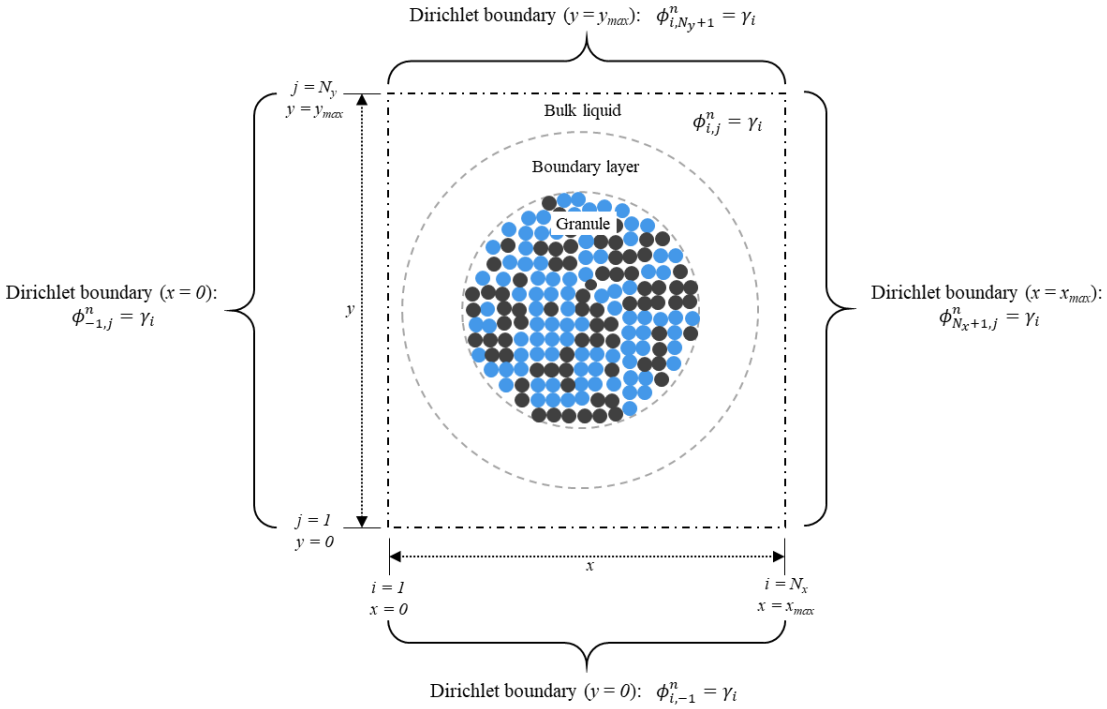

Figure 2.1. Considered boundary conditions

2.2. The term of reaction

The matrix of reaction components $R([\phi^n])$must be calculated and added to equation 2.14. The reaction of the soluble components in this system is considered function of the acid-base reaction (section 2.2.1) and the microbial activity of the aggregate (section 2.2.2).

2.2.1. The acid-base reaction

A sub-model is added to IbM to comprehensively describe the pH dynamics of the system. The influent enters in the reactor at a fix pH with a buffer of anions and cations. Then, in the system and due to the microbial activity, the pH varies function of the space and time ($\mathrm{pH}^n_{i,j}$). As the acid-base reactions have a fast kinetics, they can be considered instantaneous and it is possible to use and algebraic algorithm to approximate it each time step of the integration [3, 4].

To describe the acid-base reactions taking place, first it is needed to define the chemical species considered and their forms (Table 2.1). The chemical species are the states of the model: the soluble components that the user is interested to integrate (solution of equation 2.14, $[\phi^{n+1}]$). And some of these chemical species have different chemical forms in water function of the acid-base reactions associated to them. For example, acetic acid in solution tends to deprotonate and therefore is always in equilibrium with its deprotonated form, acetate. The concentration of these two forms is function of the pH of the liquid as the neutrality of charges is maintained in the liquid.

Each of the forms included in the model is associated to its charge and its Gibbs energy value. This allows to directly calculate the acid-base constants for all the chemical species in the solution and to compute the balance of charges in the bulk liquid (equation 2.15). Because the equilibrium of charges occurs at a fast time scale, this balance is assumed to remain neutral [5]. Therefore, it is used to calculate the pH and the concentration of all the forms of the states, finding the root of equation 2.15 using an implicit Newton-Raphson scheme [6].

$$0 = [H^+] + \sum_{k=1}^{k=K} z_k \cdot [C_k] \quad [2.15]$$

Where K is the total number of all the chemical forms present in the bulk liquid, C_k is the concentration of each of the forms, and z_k is its charge.

Table 2.1. Structure of the chemical forms of considered components in the model

State	Chemical Forms(1)				
Name	$-H_2O$	$+H_2O$	$-H^+$	$-2H^+$	$-3H^+$
Acetic acid	–	CH_3COOH	CH_3COO^-	–	–
Inorganic carbon	CO_2	H_2CO_3	HCO_3^-	CO_3^{2-}	–

(1) $-H_2O$: *Non hydrated form*
$+H_2O$: *Hydrated form*
$-H^+$: *First deprotonation*
$-2H^+$: *Second deprotonation*
$-3H^+$: *Third deprotonation*

To find the root of the equation 2.15, it is needed to write the concentrations of the forms function the concentration of protons and the states of the model (ϕ^{n+1}), calculated through equation 2.14. The equations for each of the chemical forms are present in equations 2.16 – 2.21.

Not hydrated form concentration: $[Not\ Hyd.] = k_d \cdot \frac{\phi_{i,j}^n \cdot [H^+]^3}{\theta}$ [2.16]

Hydrated and fully protonated form concentration: $[Hyd.] = \frac{\phi_{i,j}^n \cdot [H^+]^3}{\theta}$ [2.17]

First deprotonated form concentration: $[1^{st} dep.] = K_{a_1} \cdot \frac{\phi_{i,j}^n \cdot [H^+]^2}{\theta}$ [2.18]

Second deprotonated form concentration: $[2^{nd} dep.] = K_{a_1} \cdot K_{a_2} \cdot \frac{\phi_{i,j}^n \cdot [H^+]}{\theta}$ [2.19]

Third deprotonated form concentration: $[3^{er} dep.] = K_{a_1} \cdot K_{a_2} \cdot K_{a_3} \cdot \frac{\phi_{i,j}^n}{\theta}$ [2.20]

$$\theta = (1 + k_d) \cdot [H^+]^3 + k_{a_1} \cdot [H^+]^2 + k_{a_1} \cdot k_{a_2} \cdot [H^+] + k_{a_1} \cdot k_{a_2} \cdot k_{a_3} \quad [2.21]$$

Where k_d, k_{a1}, k_{a2} and k_{a3} are the hydration and acid-base equilibrium constants calculated using the Gibbs energy values of formation of the substrate(s) and product(s).

2.2.2. Microbial activity (Monod kinetics)

The kinetics of microbes that grow in the aggregate are calculated function of the local conditions of the node where the cell is located inside the simulation domain. The kinetics parameters associated to each of the microbial species considered are included in the *Excel* file that feeds the model. But also, the *Excel* file needs to indicate the stoichiometry chosen for the anabolic (*Ana*), catabolic (*Cat*) and decay (*Dec*) processes for each of the microbial species. Equations 2.22 – 2.24 show examples of stoichiometries for anabolism, catabolism and decay where C_s and N_s refer to

carbon and nitrogen sources, *eD* and *eA* to electron donor and acceptor and *X* to biomass considering an average formula of $C_1H_{1.8}O_{0.5}N_{0.2}$.

$$Ana \ = \ 1 \cdot C_s + 0.2 \cdot N_s \ + \cdots \rightarrow \cdots + \ 1 \cdot \ X \qquad [2.22]$$

$$Cat \ = \ \alpha \cdot \ eD \ + \ \beta \cdot \ eA \ \rightarrow \cdots \qquad [2.23]$$

$$Dec \ = \ 1 \cdot \ X + \cdots \rightarrow \ 1 \cdot C_s + 0.2 \cdot N_s \qquad [2.24]$$

The stoichiometry of the overall metabolism (*Met*) is calculated function of the anabolic and catabolic stoichiometries and the growth yield (Y_{XS}, equation 2.25) [7, 8].

$$Met \ = \ \frac{1}{Y_{XS}} \cdot Cat \ + \ Ana \qquad [2.25]$$

Therefore, if the cell grows, is going to do it function of the metabolic stoichiometry calculated per equation 2.22. But if the conditions are not favourable, then cell might start to decay function of the stoichiometry fixed per equation 2.24.

The value of Y_{XS} can be defined initially by the user (through bibliographic review or experimental measure), but also Y_{XS} can be estimated assuming that the Gibbs energy values of the anabolic and catabolic equations plus an energy dissipation term (ΔG_{Dis}) give a fair approximation (equation 2.26). Because the Gibbs energies for the catabolic and anabolic reactions are updated in the system with the local concentrations and temperature surrounding the microbes, it is possible to approximate changes in the stoichiometry of the metabolism function the environmental conditions [5, 9].

$$Y_{XS} = \frac{\Delta G_{Cat}}{\Delta G_{Ana} + \Delta G_{Dis}} \qquad [2.26]$$

Where ΔG_{Cat} refers to the Gibbs energy of the catabolic reaction, ΔG_{Ana} to the Gibbs energy of the anabolic reaction and ΔG_{Dis} to the energy assumed dissipated in the anabolic process. This energy is assumed only function of the carbon source used [7] (equation 2.27).

$$\Delta G_{Dis} = \ 200 \ + \ 18 \cdot (6 \ - \ NoC)^{1.8} \ + \ exp[\{(-0.2 - \gamma)^2\}^{0.16} \cdot (3.6 + 0.4 \cdot NoC)] \qquad [2.27]$$

Where NoC refers to the number of elements of carbon that the C-source has and γ refers to the degree of reduction of the carbon source. When the carbon source is inorganic carbon, a value of 986 kJ/C-mol are assumed for energy dissipation if there is not occurrence of reversed electron transfer and 3500 kJ/C-mol if it occurs [10].

To compute the growth of bacteria, the modified version of Monod equation (including maintenance or decay term, equation 2.28) is employed.

$$\mu_m^n = \ \mu_m^{max} \cdot \frac{\phi_{i,j}^n}{K_{S,m} + \phi_{i,j}^n} - b_m^{max} \qquad [2.28]$$

Where μ_m^{max} is the maximum specific growth rate of bacterium *m*, $K_{S,m}$ is the half-velocity constant or Monod constant (the value of ϕ that μ/μ^{max}= 0.5) and b_m^{max} is the maintenance or decay rate. Equation 2.28 would change function of the Monod or inhibition terms that are considered for the species *m*.

It is known that μ_m^{max} and b_m^{max} of bacteria are influenced by temperature and pH. The main expressions to considerate the influence of temperature and pH are the Arrhenius equation (equation 2.29) and bell-shape equation (equation 2.30), respectively.

$$\mu_m^{max}(T) = A \cdot \exp\left(-\frac{E}{T}\right) \quad [2.29]$$

$$\mu_m^{max}(pH) = \frac{\mu_m^{max}\,(pH_{op})}{1 + \left(\frac{10^{-pK1}}{10^{-pH}}\right) + \left(\frac{10^{-pH}}{10^{-pK2}}\right)} \quad [2.30]$$

Where A is the pre-exponential factor, E is the Arrhenius constant (E_a/R), T is the temperature in Kelvins, and $pK1$ and $pK2$ are the pH in which μ_m^{max} is half of the value at optimal pH. The same expressions are used to calculate the influence of temperature and pH on b_m^{max}.

With the metabolic stoichiometry and the kinetic parameters (μ^{max}, K_S and b^{max}) for each of the microbial species present in the reactor, it is possible to calculate the growth rate (μ, equation 2.28) of a microbe *m* and then, through the stoichiometry and the mass of the microbe, the generation/consumption term for the biomass, substrate uptakes and product generation. Equation 2.31 is the derivative of the mass of a microbe which is integrated in time using a forward Euler scheme; and equation 2.32 computes the reaction term for a specific soluble component *s* due to the activity of the microbe *m*.

$$\frac{dX_m}{dt} = \mu_m^n \cdot X_m^n \quad [2.31]$$

$$R_m^n = \frac{\mu_m^n \cdot \delta_{s,m}}{V_{xy}} \quad [2.32]$$

Where X_m refers to the mass in moles of the cell *m*, $\delta_{s,m}$ to the stoichiometric coefficient of the substrate S for the cell *m* and V_{xy} to the volume of one node of the simulation domain. In non-favourable conditions, the cell might not be able to harvest enough substrate or energy to grow or maintain. In that case, equation 2.28 returns a negative value which means that the cell is going to lose mass and shrink function of the decay stoichiometry assigned (equation 2.24) and with the kinetics of equation 2.33.

$$b_m^n = b_m^{max} - \mu_m^{max} \cdot \frac{\phi_{i,j}^n}{K_{S,m} + \phi_{i,j}^n} \quad [2.33]$$

The calculation of the reaction term for the soluble components is function of the position in the simulation domain (one reaction term per node for each soluble component). Therefore, equation 2.34 is used to consider all microorganisms that are contributing to the reaction term of a specific soluble component in the *node i,j*.

$$R_{i,j}^n = \sum_{m=1}^{m=M_{i,j}} R_m^n \quad [2.34]$$

Where $M_{i,j}$ refers to all microorganism that are in the *node i,j.* The reaction terms R_{ij}^n calculated by equation 2.34 for all the nodes of the simulation domain are included in the equation 2.14 to calculate the concentration of each of the soluble components in all the simulation domain.

2.3. The update of the Dirichlet boundary condition

The modelled granule is considered that growing inside of a reactor where the conditions in the bulk liquid are changing due to the activity of the whole biomass in it. Considering the activity calculated in the granule, representative of the activity of the whole reactor, its average activity is

used to integrate the concentration of the soluble components in the bulk liquid of the reactor (S) through a mass balance (equation 2.35).

$$\frac{dS}{dt} = \frac{1}{HRT} \cdot \left(S_{inf} - S\right) + R \qquad [2.35]$$

Where HRT refers to the hydraulic time fixed in the reactor, $S_{inf,i}$ to the concentration in the influent and R to the reaction term considered in reactor. R is calculated assuming the average of the reaction terms calculated for the granule is representative of the whole reactor activity (equation 2.36).

$$R = \frac{\sum_{i=1,j=1}^{i=N_x,j=N_y} {R_{i,j}}^n}{N_x \cdot N_y} \qquad [2.36]$$

The concentrations in the influent and the hydraulic retention time are defined by the user. Nevertheless, this last one, can also be dynamically imposed by imposing fixed a concentration of a specific substrate in the reactor (equation 2.37).

$$HRT = \frac{(S_{inf} - S)}{R_{r,S}} \qquad [2.37]$$

The new concentration *S* calculated with the integration of equation 2.35, and using an explicit forward Euler scheme, updates the Dirichlet value (γ) included in equation 2.14.

2.4. Domain definition – *diffusion* and *no-diffusion regions*

As mentioned, the *diffusion region*, where diffusion-reaction equation is solved (equation 2.14), comprises the *granule region* and *boundary layer*. The *granule region* is basically the grid cells that hold bacteria, and *boundary layer* is the grid cells in the immediate vicinity of grid cells with bacteria. The thickness of *boundary layer* is selected by the user.

The distinction of the *diffusion region* starts with the detection of which nodes are potentially in this region, creating a preliminary diffusion region (**Figure 2.2**). For that, the minimum and maximum position of bacteria in granule in coordinate *x* and *y*, and boundary layer thickness are considered. The computational cost is reduced by using this preliminary diffusion region instead of the entire simulation domain.

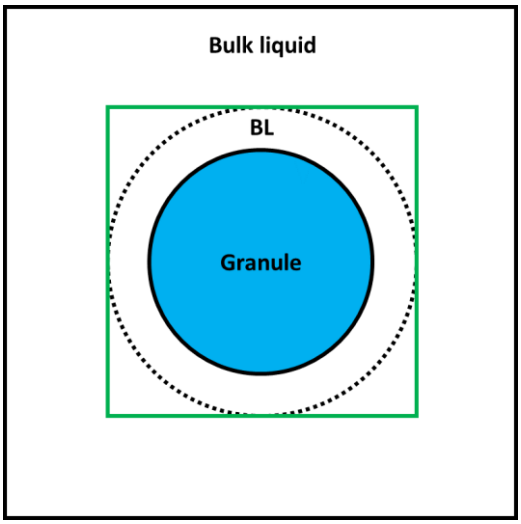

Figure 2.2. Representation of preliminary diffusion region (green square).
Legend: BL – boundary layer.

Then, grid cells that hold bacteria (at least one bacterium) are determined and included in diffusion region. Thus, it is defined the *granule region*. To recognise which grid cells are belonging to boundary layer region (and including them in diffusion region), the boundary grid cells with bacteria (i.e., the outermost grid cells of granule region) must be sought. An efficient way to find them is through the convolution of granule region matrix and *edge detection kernel* (equation 2.38).

$$[K]_{ED} = \begin{pmatrix} -1/8 & -1/8 & -1/8 \\ -1/8 & 1 & -1/8 \\ -1/8 & -1/8 & -1/8 \end{pmatrix} \quad [2.38]$$

Once found the boundary grid cells with bacteria, it is time to check whether the neighbouring grid cells ($g_{i,j}$) belong to boundary layer region and, thus, the diffusion region. By dividing the boundary layer thickness by grid size (*h*), the extent of the neighbouring grid cells with possibility to belong the boundary layer region is obtained (**Figure 2.3**).

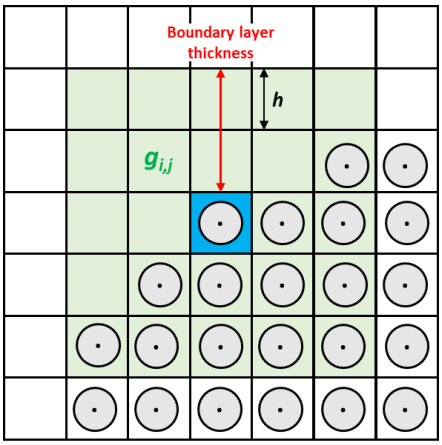

Figure 2.3. Neighbouring grid cells (green squares, $g_{i,j}$) of a specific boundary grid cell with bacteria (blue square) with potential to be included in *diffusion region*.

For each boundary grid cell with bacteria, the neighbouring grid cells with potential are selected and evaluated whether they are actually in boundary layer region or not. If they are in it, they are included in *diffusion region*.

Finally, it is defined a *focus region* in which includes all diffusion nodes, has at least one bulk layer node at each side of diffusion nodes, and has an odd number of nodes in *x* and *y* coordinates (for efficient solution of diffusion-reaction equation [11]). The *focus region* establishes the region of the simulation domain where diffusion-reaction equation is solved, and the pseudo-steady state is checked. The flowchart of *diffusion region* determination is presented in **Figure 2.4**.

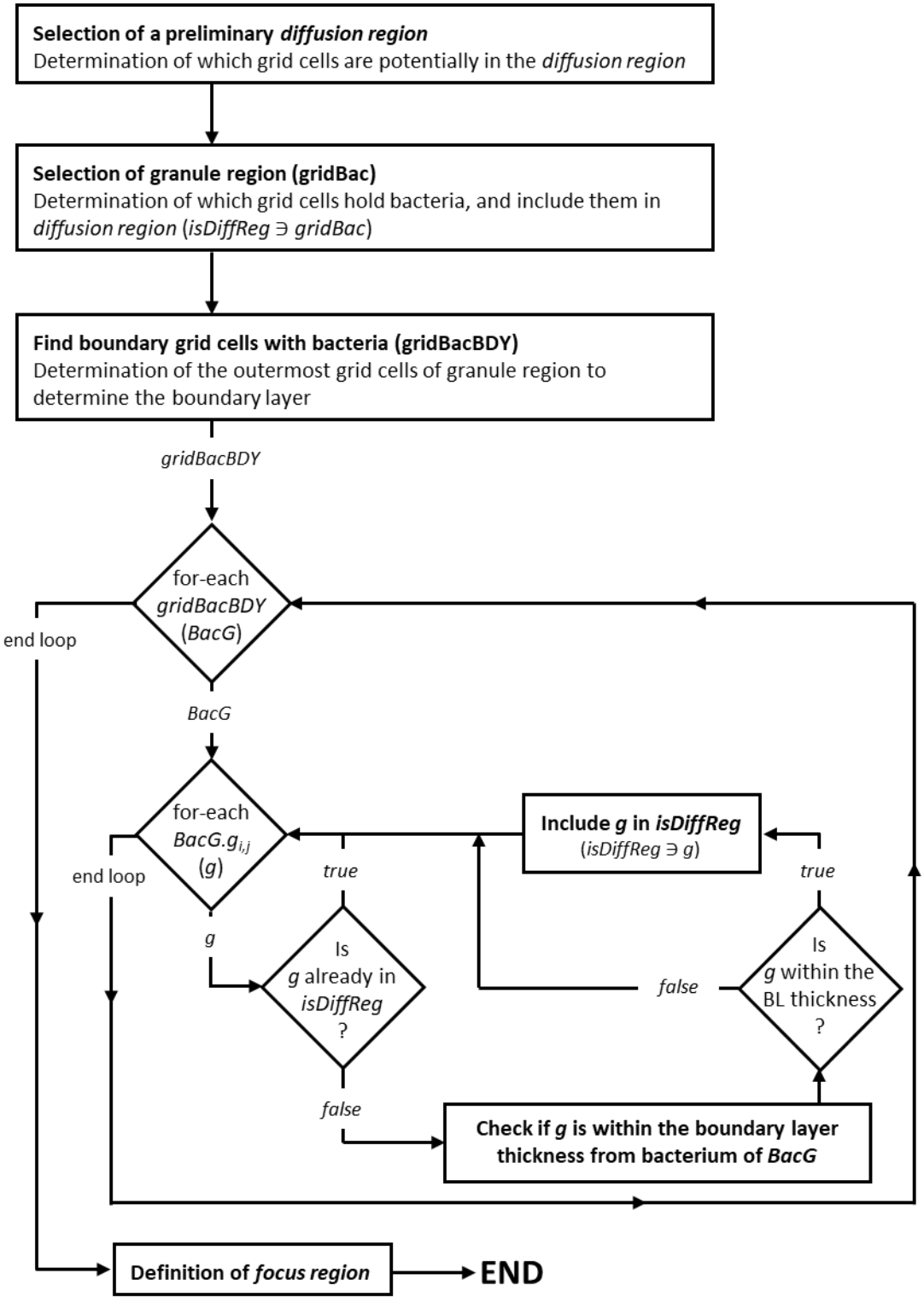

Figure 2.4. Scheme of *diffusion region* determination

3. Multigrid method

Noted that the system that is solved (equation 2.14) has the form

$$A\mathrm{u} = f. \qquad [3.1]$$

Where *A* and *f* corresponds to the left-hand and right-hand side of equation 2.14, respectively; and **u** corresponds to the exact solution that we seek (i.e., $[\phi^{n+1}]$). In this case, multigrid method (MG) is used to solve the diffusion-reaction equation (equation 2.1).

The classic iterative (or relaxation) methods are the core of multigrid methods. Although iterative methods are easier to implement than most direct methods and possess a strong smoothing property, they are only very effective at eliminating the high-frequency (or oscillatory) components of the error, but not for low-frequency (or smooth) components. That is, most basic relaxation schemes suffer in the presence of smooth components of error. By transferring the problem unto coarser grid, a smooth error looks *more oscillatory* and therefore, relaxation will be more effective (i.e., method is more able to smooth lower frequencies). By transferring the problem unto coarser and coarser grids, more and more of the low frequencies can be smoothed. This is done with a hierarchy of grids (**Figure 3.1**).

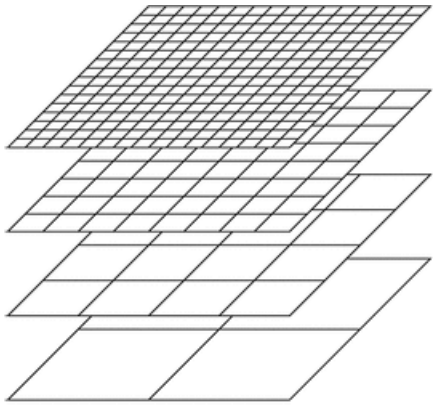

Figure 3.1. Hierarchy system of grids. Grid spacing is halved for each level.

In addition to the improving efficiency, relaxation on a coarse grid is less expensive because there are fewer unknowns to be updated. Also, the coarse gird will have an improved converge rate, due to the convergence factor behaves like $1 - O(h^2)$.

3.1. Definition of error (e), residual (r) and residual equation

From now, it is use **u** to denote the exact solution of the system $A\mathrm{u} = f$, and **v** to denote an approximation to the exact solution. Considering that this system has a unique solution, there are two important measures of **v** as an approximation of **u** – the *error* (equation 3.2) and the *residual* (equation 3.3).

$$e = \mathrm{u} - \mathrm{v} \qquad [3.2]$$

$$r = f - A\mathrm{v} \qquad [3.3]$$

The error (or algebraic error) is as inaccessible as the exact solution itself. In contrast, the residual is a computable measure of how well **v** approximates **u**. Basically, the residual computes the amount that the approximation **v** fails to satisfy the original problem $A\mathrm{u} = f$. Combining the equations 3.1, 3.2 and 3.3, the *residual equation* is obtained (equation 3.4).

$$Ae = r \qquad [3.4]$$

This relationship between the error and the residual is extremely important in multigrid methods, because it says that the error satisfies the same set of equation as the unknown **u** when f is replaced by the residual r. Therefore, relaxation on the original equation $A\mathrm{u} = f$ with an arbitrary initial guess **v** is equivalent to relaxing on the residual equation $Ae = r$ with the specific initial guess $e = 0$.

The residual equation involves a great advantage to solve the original system. Suppose that an approximation **v** has been obtained by some method. As described above, it is easy to compute the residual r (equation 3.3). To improve the approximation **v**, we might only solve the *residual equation* for e and then compute a new approximation using the definition of the error

$$\mathrm{u} = \mathrm{v} + e. \qquad [3.5]$$

We call this relationship the *correction equation* or *residual correction*.

3.2. Elements of multigrid (V-Cycle method)

There are many variations of multigrid method, but all of them are based on the same strategy, the so-called *coarse grid correction*. This strategy combines the idea of using the residual equation to relax on the error, and the concept of using coarser grids to generate a better initial guess (in this case, an approximation of the error) to improve an iterative method (known as *nested iteration*). Considering a grid with grid size h (Ω^h) and the corresponding linear system of equations

$$A^h\mathrm{u}^h = f^h, \qquad [3.6]$$

the *coarse grid correction* can be presented by the following procedure:

- Smooth (pre-smoothing) $A^h\mathrm{u}^h = f^h$ on Ω^h. This step gives an approximation $\mathbf{v}^h$ of the solution. Compute the residual r^h using the equation 3.3.
- Project (restrict) the residual to Ω^{2h}. The result is called $R(r^h)$.
- Solve $A^{2h}e^{2h} = R\left(r^h\right)$ on Ω^{2h}. With this step, one obtains an approximation e^{2h} of the error.
- Project (interpolate/prolongate) e^{2h} to Ω^h. The result is denoted by $P(e^{2h})$.
- Update the approximation of the solution on Ω^h using the equation 3.5. After the update, one can apply once more some iteration (post-smoothing).

A recursive application of this procedure, of using the *coarse grid correction* for solving the coarse grid equation, leads to the multigrid method.

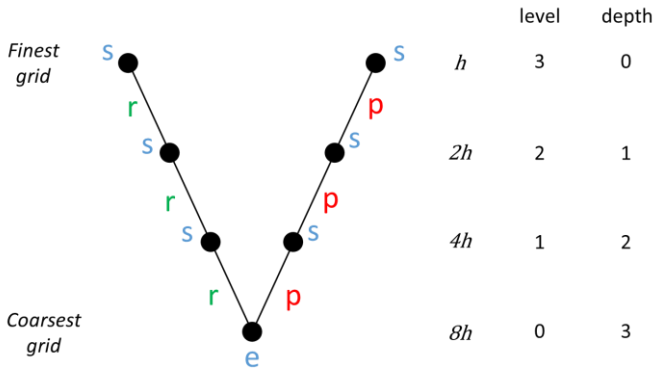

Figure 3.2. Multigrid V-cycle, **s** – smoothing, **r** – restriction, **p** – prolongation, **e** – exact solver.

As mentioned, there are several variations of multigrid methods, with different speed of solving a single iteration and its rate of convergence. In this case, based on the rate of solving, V-cycle method is chosen over the others (F-cycle, W-cycle, or full multigrid cycle). Multigrid itself is an iterative method and the V-cycle is repeated several times to find a sufficiently accurate approximation of the solution. **Figure 3.2** shows the schedule for the grids in the order in which they are visited. Because of the pattern in this diagram, this algorithm is called the V-cycle.

To describe the V-cycle method, it is assumed that there are $l+1$ grids ($l \geq 0$), where the finest grid has the grid spacing *h* and the grid spacing increase by the factor 2 for each coarser grid. Let $L = 2^l$. Then, V-cycle method heeds the following procedure:

- Apply the smoother ν_1 times (pre-smoothing) to $A^h \mathrm{u}^h = f^h$ with an initial guess $\mathbf{v}^h$ (arbitrary or not).
- Compute r^h and project it (restriction) to Ω^{2h}. The result is called r^{2h}.
    - Apply the smoother ν_1 times (pre-smoothing) to $A^{2h}e^{2h} = r^{2h}$ with the initial guess $e^{2h} = 0$.
    - Compute $\tilde{r}^{2h}$ and project it (restriction) to Ω^{4h}. The result is called r^{4h}.

        ⋮

        - Solve $A^{Lh}e^{Lh} = r^{Lh}$.
        - Project (interpolate) the solution (e^{Lh}) to Ω^{Lh-2}. The result is called $\tilde{e}^{Lh-2}$.

        ⋮

    - Correct $e^{2h} \coloneqq e^{2h} + \tilde{e}^{2h}$.
    - Apply the smoother ν_2 times (post-smoothing) to $A^{2h}e^{2h} = r^{2h}$ with the initial guess e^{2h}.
    - Project (interpolate) the solution (e^{2h}) to Ω^h. The result is called e^h.
- Correct $v^h \coloneqq v^h + e^h$.
- Apply the smoother ν_2 times (post-smoothing) to $A^h u^h = b^h$ with the initial guess v^h.

3.2.1. Smoothing – Jacobi method

In multigrid methods, an iterative method is used to solve the system of equations, either the original problem ($A\mathrm{u} = f$) or the residual equation ($Ae = r$), by finding successive approximation for its solution. The application of only a few iterations is called *smoothing*, and the iterative method itself *smoother*. In this case, the classic iterative *Jacobi method* is chosen as smoother, because it is the simplest and easiest method for solving our system of equations. Moreover, Jacobi method is easier to implement in parallel than other iterative method (as Gauss-Seidel method).

Let $A\mathrm{u} = f$ be a square system of *n* linear equations, where:

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{pmatrix}, \quad \mathrm{u} = \begin{pmatrix} \mathrm{u}_1 \\ \mathrm{u}_2 \\ \vdots \\ \mathrm{u}_n \end{pmatrix}, \quad f = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{pmatrix}. \qquad [3.7]$$

Then *A* is decomposed into a diagonal component (*D*), a lower triangular part (*L*) and an upper triangular part (*U*) such that $A = D + (L + U)$, where:

$$D = \begin{pmatrix} a_{1,1} & 0 & \cdots & 0 \\ 0 & a_{2,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{n,n} \end{pmatrix}, \quad L + U = \begin{pmatrix} 0 & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & 0 & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & 0 \end{pmatrix}. \qquad [3.8]$$

Applying this to the set system of linear equations we find that the solution is obtained iteratively via

$$\mathrm{u}^{(k+1)} = D^{-1}\big(f - (L+U)\cdot \mathrm{u}^{(k)}\big), \qquad [3.9]$$

where $\mathrm{u}^{(k)}$ is the *k*th approximation or iteration of **u** and $\mathrm{u}^{(k+1)}$ is the next or $k+1$ iteration of **u**. The element-based formula of equation 2.9 is

$$\mathrm{u}_i^{(k+1)} = \frac{1}{a_{i,i}}\left(f_i - \sum_{j\neq i} a_{i,j}\cdot \mathrm{u}_j^{(k)}\right), \qquad [3.10]$$

where $a_{i,i}$ denotes the entries in the matrix *A*, $\mathrm{u}_j^{(k)}$ and $\mathrm{u}_i^{(k+1)}$ are respectively the entries in the approximation vector $\mathrm{u}^{(k)}$ and $\mathrm{u}^{(k+1)}$, and f_i is the entries of the right-side vector f.

The standard convergence condition is when the spectral radius (ρ) of the iteration matrix is less than 1:

$$\rho\big(D^{-1}(L+U)\big) < 1. \qquad [3.11]$$

In order to construct a multigrid method, one needs mechanisms that transfer information in an appropriate way between the grids, either from coarser grid to finer grid or vice versa. The transfer of information between the grids (restriction and prolongation) are described in further detail in the following sections.

3.2.2. Prolongation

Prolongation (or interpolation) is the transfer from the coarse grid to the fine grid. It exists several methods to prolongate the information, but for most multigrid purposes the simplest method is used – the *linear interpolation*. Because of our system is a two-dimensional domain, bilinear interpolation scheme is used. From now, the interpolation operator will be denoted I_{2h}^{h}. This operator takes coarse-grid matrices (v^{2h}) and produces fine-grid matrices (v^{h}) according to

$$I_{2h}^{h} * \mathrm{v}^{2h} = \mathrm{v}^{h}, \qquad [3.12]$$

where the components of v^{h} are given by:

- $\mathrm{v}_{2i-1,2j-1}^{h} = \mathrm{v}_{i,j}^{2h}$
- $\mathrm{v}_{2i,2j-1}^{h} = \frac{1}{2}\big(\mathrm{v}_{i,j}^{2h} + \mathrm{v}_{i+1,j}^{2h}\big)$
- $\mathrm{v}_{2i-1,2j}^{h} = \frac{1}{2}\big(\mathrm{v}_{i,j}^{2h} + \mathrm{v}_{i,j+1}^{2h}\big)$
- $\mathrm{v}_{2i,2j}^{h} = \frac{1}{4}\big(\mathrm{v}_{i,j}^{2h} + \mathrm{v}_{i,j+1}^{2h} + \mathrm{v}_{i+1,j}^{2h} + \mathrm{v}_{i+1,j+1}^{2h}\big)$ [3.13]

Written in stencil notation, these equations yield

$$I_{2h}^{h} = \frac{1}{4}\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}. \qquad [3.14]$$

The colour coding of the equations 3.13 takes the same colour notation as **Figure 3.3**.

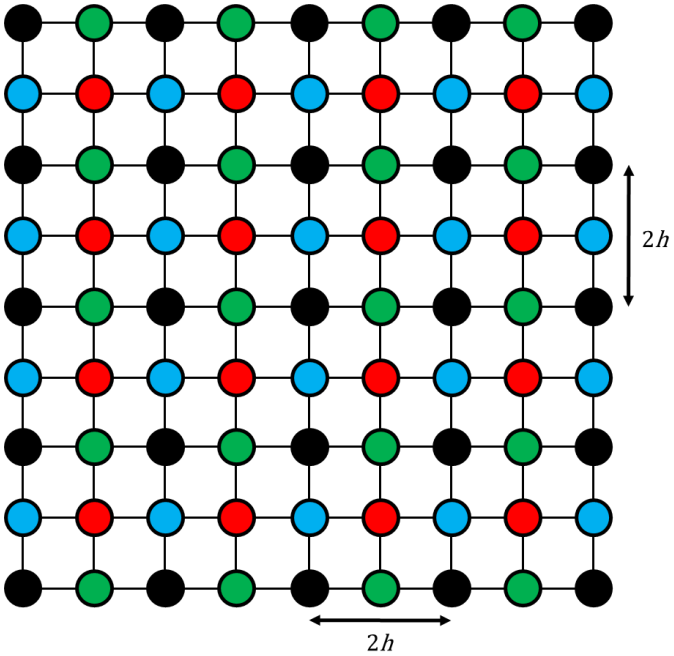

Figure 3.3. Fine grid in two dimensions. The grid points that exist on both grids (coarse and fine) are marked in black. The *new* grid points of finer grid are marked in colours (same colour notation as equations 3.13).

3.2.3. Restriction

The second class of intergrid transfer operations comprises moving information from a fine grid to a coarse grid. This operation is called *restriction*. Henceforth, the restriction operator is denoted by I_h^{2h}. The simplest restriction is the *injection*, but this type of restriction does not lead to an efficient method, because ignores part of the fine grid and, as consequence, a loss of information is produced. A most reliable restriction operator (because this uses all nodes on the fine grid) is the *full weighting restriction*, in which is defined by

$$I_h^{2h} * \mathrm{v}^h = \mathrm{v}^{2h}, \qquad [3.15]$$

where the components of v^{2h} are given by:

$$\mathrm{v}_{i,j}^{2h} = \frac{1}{16}\left(4\mathrm{v}_{i,j}^h + 2\mathrm{v}_{i+1,j}^h + 2\mathrm{v}_{i+1,j}^h + 2\mathrm{v}_{i-1,j}^h + 2\mathrm{v}_{i,j+1}^h + 2\mathrm{v}_{i,j-1}^h + \mathrm{v}_{i+1,j+1}^h + \mathrm{v}_{i+1,j-1}^h + \mathrm{v}_{i-1,j+1}^h + \mathrm{v}_{i-1,j-1}^h\right). \qquad [3.16]$$

Then, written in stencil notation, the restriction operator is

$$I_h^{2h} = \frac{1}{16}\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}. \qquad [3.17]$$

As **Figure 3.4** shows, the values of the coarse grid matrix are weighted averages of values at neighbouring fine-grid points.

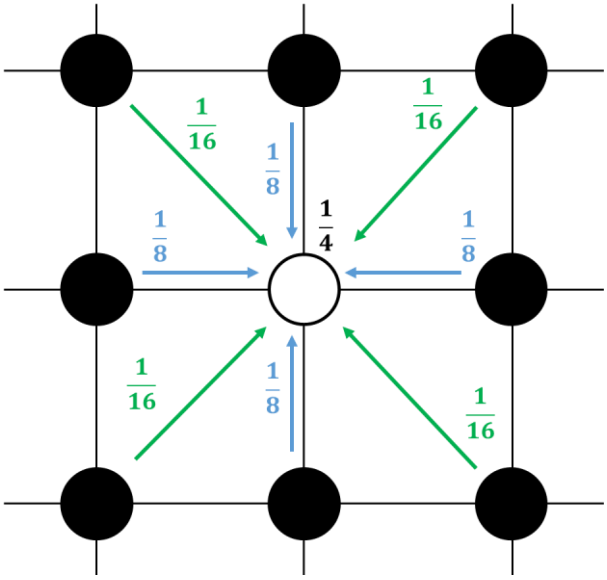

Figure 3.4. Full weighted restriction. The grid points of the coarser grid ($v_{i,j}^{2h}$) is marked in white. The previous grid points of finer grid are marked in black.

3.2.4. Coarse grid definition

In this section is defined the coarse grid matrices (from A^{2h} to A^{Lh}). These matrices are a Ω^{Lh} version of the fine grid matrix A^h. There are two main approaches to define the coarse grid matrix (A^{lh}) – (*1*) defining A^{Lh} by applying finite difference method to the differential operator on Ω^{Lh}, or (*2*) defining A^{Lh} by *Galerkin property* (two-level method application– equation 3.18; multigrid method application – equation 3.19).

$$A^{2h} \coloneqq I_h^{2h} A^h I_{2h}^h \qquad [3.18]$$

$$A^{(2^{l+1})h} \coloneqq I_{(2^l)h}^{(2^{l+1})h} A^{(2^l)h} I_{(2^{l+1})h}^{(2^l)} \qquad [3.19]$$

Here, Galerkin property is used to justify the definition of *Laplacian kernel* in coarse grids. To simplify the notation, the Galerkin definition for two-level method is used (equation 3.18). First, all operators on the right-hand side of equation 2.18 are represented in their matrixial form:

$$I_h^{2h} = \frac{1}{4}\begin{pmatrix} 1 & 2 & 1 & & & & & & \\ & & 1 & 2 & 1 & & & & \\ & & & & 1 & 2 & 1 & & \\ & & & & & & & \ddots & \\ & & & & & & & 1 & 2 & 1 \end{pmatrix} \qquad [3.20]$$

$$I_h^{2h} = \frac{1}{h^2}\begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \vdots & \vdots & \vdots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix} \qquad [3.21]$$

$$I_{2h}^{h} = \frac{1}{2}\begin{pmatrix} 1 & & & & & \\ 2 & & & & & \\ 1 & 1 & & & & \\ & 2 & & & & \\ & 1 & 1 & & & \\ & & 2 & & & \\ & & 1 & \ddots & & \\ & & & \ddots & & \\ & & & & 1 & \\ & & & & 2 & \\ & & & & 1 & \end{pmatrix} \qquad [3.22]$$

Then,

$$A^{2h} = \frac{1}{4}\begin{pmatrix} 1 & 2 & 1 & & & & & & \\ & & 1 & 2 & 1 & & & & \\ & & & & 1 & 2 & 1 & & \\ & & & & & & \ddots & & \\ & & & & & & & 1 & 2 & 1 \end{pmatrix} \frac{1}{h^2}\begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \vdots & \vdots & \vdots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix}$$

$$\times \frac{1}{2}\begin{pmatrix} 1 & & & & & \\ 2 & & & & & \\ 1 & 1 & & & & \\ & 2 & & & & \\ & 1 & 1 & & & \\ & & 2 & & & \\ & & 1 & \ddots & & \\ & & & \ddots & & \\ & & & & 1 & \\ & & & & 2 & \\ & & & & 1 & \end{pmatrix}$$

$$= \frac{1}{8h^2}\begin{pmatrix} 1 & 2 & 1 & & & & & & \\ & & 1 & 2 & 1 & & & & \\ & & & & 1 & 2 & 1 & & \\ & & & & & & \ddots & & \\ & & & & & & & 1 & 2 & 1 \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 & & & \\ 2 & -1 & 0 & & & \\ 0 & 0 & 0 & & & \\ -1 & 2 & -1 & & & \\ & & & \ddots & & \\ & & & & -1 & 2 & -1 \\ & & & & 0 & 0 & 0 \\ & & & & 0 & -1 & 2 \\ & & & & 0 & 0 & 0 \end{pmatrix}$$

$$= \frac{1}{8h^2}\begin{pmatrix} 4 & -2 & & & \\ -2 & 4 & -2 & & \\ & \vdots & \vdots & \vdots & \\ & & -2 & 4 & -2 \\ & & & -2 & 4 \end{pmatrix}$$

$$= \frac{1}{4h^2}\begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \vdots & \vdots & \vdots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix} \qquad [3.23]$$

Note that the coarse matrix (A^{2h}) is the same as fine matrix (A^{h}) but replacing h by $2h$. Thus, the matrix defined by the *Galerkin property* and the matrix obtained by discretizing the differential operator on the coarse grid Ω^{2h} coincide, and for any Ω^{Lh} domain will coincide if the grid spacing Lh is increased twice of its corresponding fine grid.

Remember that the left-hand side of our system $A\mathrm{u} = f$ is defined as

$$A \coloneqq ([I_k] - \alpha \cdot [L]), \qquad [3.24]$$

where $\alpha = \frac{\mathbb{D} \cdot h_t}{2 \cdot h^2}$. $[I_k]$ is the *identity kernel* and $[L]$ is the *Laplacian kernel*. Therefore, α is the only term that we should modify to define all coarse grid matrices (from A^{2h} to A^{Lh}). In this case, instead of modifying α, it is added a new term on left-hand side – the *depth* term (δ). Assuming a multigrid method with 4 levels (like **Figure 3.2**), the finest grid (with grid size h and level 3) has *depth* 0, and the coarsest grid (with grid size $8h$ and level 0) has *depth* 3. Every time that we restrict information to a coarser grid, we go "deeper" (δ+1). In contrast, when we prolongate information to a finer gird, we go to the surface (δ-1). To connect *depth* with the grid size, it is used the following definition:

$$A^{Lh} \coloneqq \left([I_k] - \left(\frac{1}{2}\right)^{2 \cdot \delta} \alpha \cdot [L] \right). \qquad [3.25]$$

For a detailed description of multigrid method, see Briggs W. *et al.* [12].

4. Microbial growth

4.1. Division and inactivity/death

Microorganisms will change their mass function of their kinetics. For each individual, its mass is integrated in time using a forward Euler scheme on equation 2.31 which allows to describe the dynamics of each of the cell growth (or decay). The model assumes that once cells achieve a maximum mass (M_{max}) or size (density is considered constant), they should divide. In this case, a new cell is formed (cell *m+1*, equation 4.1) with an initial mass that is a random percentage (A is a stochastic parameter with a value between 0 and 1) of the total mass of the mother cell. The mass of the mother cell is updated with the mass remained after the division (equation 4.2).

$$X_{m+1} = \mathrm{A} \cdot X_m \qquad [4.1]$$

$$X_m = (1 - \mathrm{A}) \cdot X_m \qquad [4.2]$$

At the same time, cells may shrink if they do not have favourable conditions and equation 2.28 returns negative values. When cells reach a minimum mass (M_{min}) or size fixed which is considered negligible, they become inactive and only can increase their mass if conditions are favourable again (and to be active again) (**Figure 4.1**). If needed, those bacteria that reach M_{min} can be removed from the system instead of becoming inactive.

Once the mass of each cell is known, it is possible to estimate how much space a cell is occupying calculating their radius (r_m) assuming they have the shape of a perfect circle and a specific density (ρ_m) (equation 4.3) [13].

$$r_m = \left(\frac{X_m}{\rho_m} \cdot \frac{3}{4\pi} \right)^{1/3} \qquad [4.3]$$

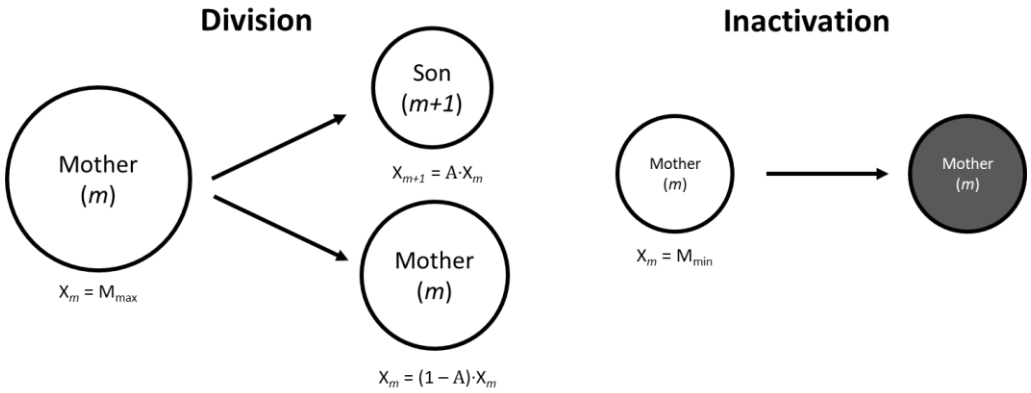

Figure 4.1. Microbial division and inactivation. A is any stochastic value between 0 – 1. M_{max} and M_{min} refer to the maximum and minimum mass that microbe can reach, respectively.

4.2. Shoving

When a cell divides, it is randomly assigned a position for the new individual (*m+1*) in the neighbourhood of its mother (*m*). Also, when a microorganism disappears (if applied), a hole is formed on the granule. Therefore, the dynamics of the cell growth implies certain shoving of the individuals inside the granule. The model assumes only certain percentage of overlapping allowed fixed by the user. Therefore, when a cell divides, a shoving algorithm checks the overlapping of the individuals. When the microbial overlap it is bigger of the maximum overlap accepted, cells move.

To compute the shoving of the cells in the aggregate after cell division, first the overlap between microorganisms is checked as per equation 4.5.

$$|\vec{v}| = \sqrt{(x_m - x_{m+1})^2 + (y_m - y_{m+1})^2} \quad [4.4]$$

$$overlap = kDist \cdot (r_m + r_{m+1}) - |\vec{v}| \quad [4.5]$$

Where $|\vec{v}|$ is the norm of the vector that links the centres of both cells *m* and *m+1* and $kDist$ is just a multiplier that allows adjustment of the minimal spacing between bacteria. If the *overlap* value is bigger than the distance allowed by the user, then microorganisms are pushing each other function their mass and their distance (equations 4.6 – 4.11).

$$\vec{p} = \frac{kDist \cdot (r_m + r_{m+1}) - |\vec{v}|}{|\vec{v}|} \quad [4.6]$$

$$a_m = 1 - \frac{X_m}{X_m + X_{m+1}} \; ; \; a_{m+1} = 1 - \frac{X_{m+1}}{X_m + X_{m+1}} \quad [4.7]$$

$$x_{new,m} = x_{old} - (x_{m+1} - x_m) \cdot a_m \cdot \vec{p} \quad [4.8]$$

$$y_{new,m} = y_{old} - (y_{m+1} - y_m) \cdot a_m \cdot \vec{p} \quad [4.9]$$

$$x_{new,m+1} = x_{old} + (x_{m+1} - x_m) \cdot a_{m+1} \cdot \vec{p} \quad [4.10]$$

$$y_{new,m+1} = y_{old} + (y_{m+1} - y_m) \cdot a_{m+1} \cdot \vec{p} \quad [4.11]$$

In this case, *Quadtree* is applied to detect the *overlapping* between bacteria and, subsequently, compute the shoving of these [14].

4.2.1.Quadtree algorithm

Overlap detection and shoving computation between bacteria could be an expensive operation. Let a granule with 1000 bacteria. If we apply a rough overlap algorithm (comparing each pair of bacteria), would require 1x10^6 operations (i.e., time complexity of $O(n^2)$). To improve shoving computation, we should reduce the number of checks that must be made. Two bacteria that are at opposite sites on granule (or far enough) cannot possibly overlap, so there is no need to check for an overlap between them. Here is where *quadtree* comes into play. Applying quadtree in overlap detection and shoving computation the number of checks is reduced significantly, turning the time complexity from $O(n^2)$ to $O(n \log n)$.

Quadtree is a tree data structure in which an internal node is sectioned in four child nodes, and each of those children could potentially be sectioned into four. Basically, quadtrees are used to split a 2D regional space by recursively subdividing it into four regions. In this case, the number of bacteria in each subregion dictates if these regions must be divided or not – if a specific region holds a higher number of bacteria than the maximum number (so-called *capacity*), then this region will be divided into four new regions. **Figure 4.2** shows an example of how quadtree works in a system in which it has a *capacity* of one bacterium (i.e., only one bacterium per region).

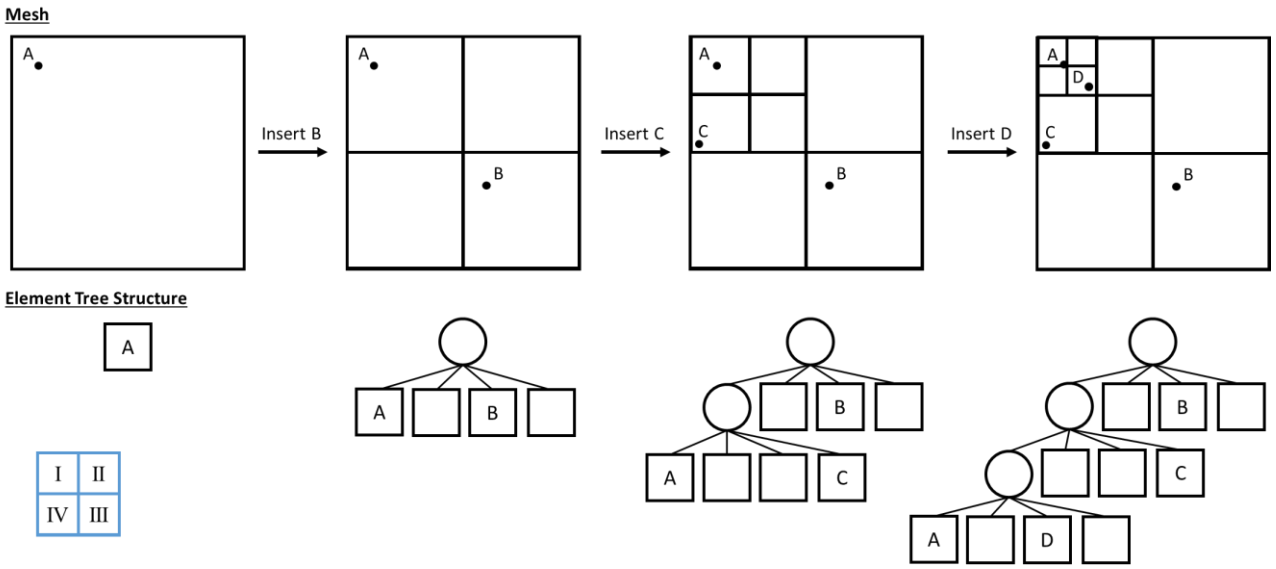

Figure 4.2. Quadtree representation. Each circle represents a bacterium in the system. Subregions are labelled clockwise (see blue grid on bottom-left of figure).

Once understood how quadtree is created (mesh and data structure), it is time to figure out how quadtree plays in shoving algorithm. First, it is necessary to create a new quadtree mesh and tree data structure (**Figure 4.2**). Obviously, every time that a division occurs, quadtree must be updated for the new bacteria and their locations. Afterwards, each bacterium is *inserted* on quadtree, that is, it is set in a specific region (or node).

Before executing the overlap detection, the neighbours of all bacteria are established. For that, it is firstly created a *neighbourhood zone* that surrounds each bacterium (i.e., bacterium is on the centre of neighbourhood zone). As the quadtree is not updated after every shoving step, the

neighbourhood zone is set to 4 times the maximum radius of bacteria. Then, for each bacterium, it is selected those regions (or subregions) which intersects with its *neighbourhood zone* and, subsequently, it is evaluated whether bacteria located in those intersect regions are within the neighbourhood zone or not (**Figure 4.3**).

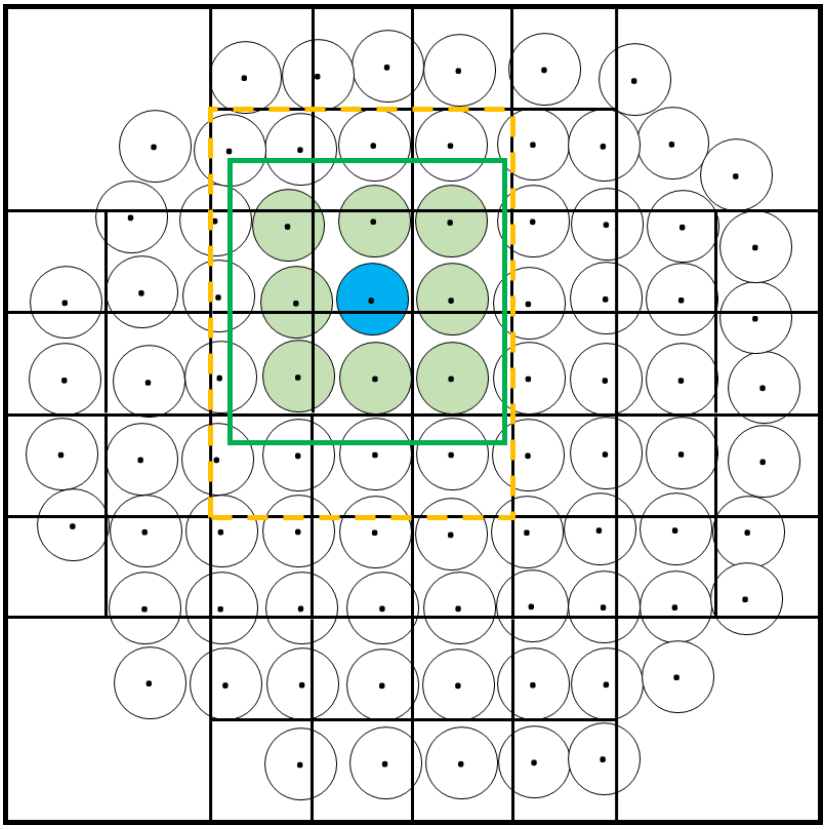

Figure 4.3. Selection of neighbours (green circles) of bacterium *m* (blue circle). The centre of circle (black dot) indicates the exact position of bacteria. In this example, quadtree *capacity* is 4 (i.e., maximum 4 bacteria per region). Green square represents the neighbourhood zone of bacterium *m* (blue circle), and dashed orange square shows all regions that intersect with the neighbourhood zone of bacterium *m*.

Finally, overlap assessment and shoving estimation (if needed) are only performed over the neighbours of bacteria (equations 3.4 – 3.11). Due to bacteria will not move very far from original position, it is assumed that chosen neighbouring bacteria stay constant. An overview of the shoving algorithm is presented in **Figure 4.4**.

4.3. Detachment

Detachment is implemented in this model only in a simplistic way, just to impose a maximum size of granule. The detachment model consists of removing every bacterium which is located outside of an imposed maximum radius of granule due to a shoving step. It is assumed that when granule size is less than maximum size, detachment is not produced due to the low shear forces in the reactor. However, the model is ready to implement a more complex model of detachment.

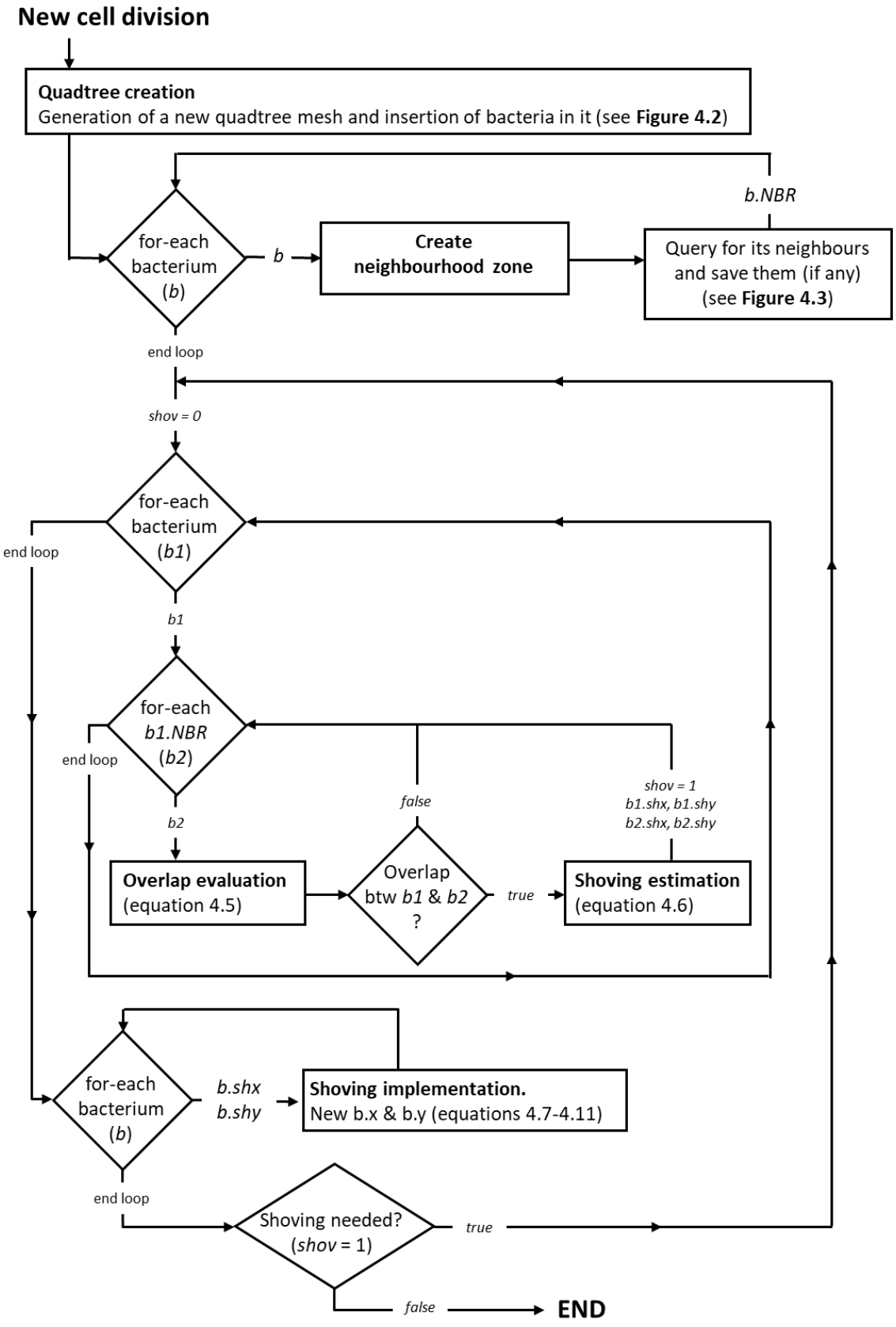

Figure 4.4. Scheme of shoving algorithm. Legend: ***b.shx*** and ***b.shy*** – displacement of bacterium *b* due to shoving of the others; ***b.NBR*** – neighbours of bacterium *b*.

5. Integration

Cells are dividing in a time scale much slower than the diffusion-reaction process (~1 hour versus ~10^{-8} hours). To solve the system, the model takes advantage of this time scale differentiation to separate the processes of solving the diffusion-reaction equation and the cell division and its shoving [13]. The overall scheme of the model is presented in **Figure 5.1**.

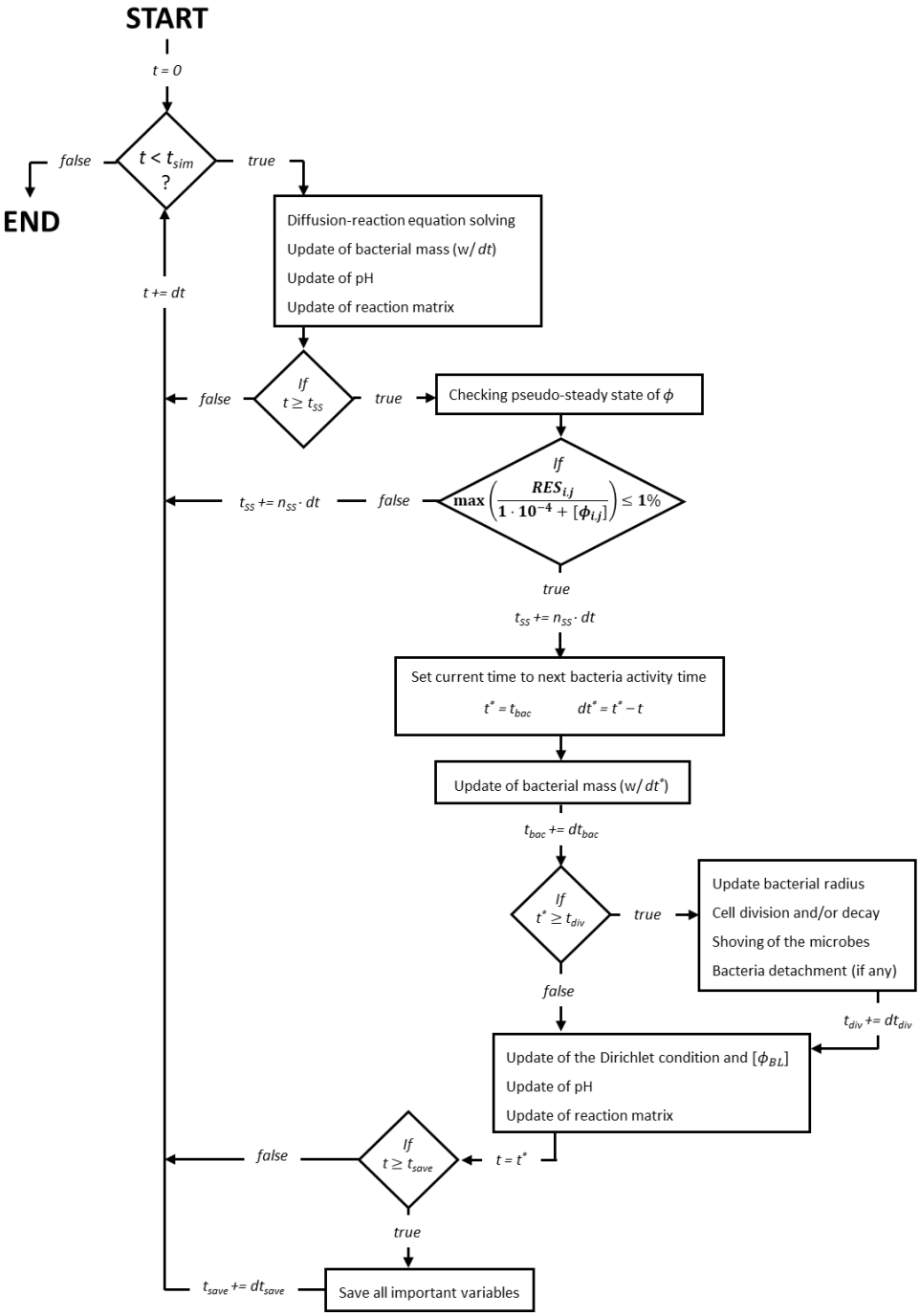

Figure 5.1. Algorithm scheme of the integration process.

First, the diffusion-reaction equation is integrated (with a time step *dt*) until it reaches a pseudo-steady state, in which the difference between the pseudo-steady state and the real steady state is less than a threshold (equation 5.2). To check whether pseudo-steady state is reached or not, only diffusion region is considered. This is because it is assumed that Dirichlet boundary condition and bulk liquid concentration are fixed in this time span, and only change when the microbial community change significantly (due to a cell division, cell inactivation or a substantial variation in microbial mass).

$$[RES] = [L] * [\phi]^{\gamma} + (h^2/\mathbb{D}) \cdot R([\phi]) \qquad [5.1]$$

$$Tol \coloneqq max\left|\frac{RES_{i,j}}{1\cdot 10^{-4}+\phi_{i,j}}\right| \leq 1\%, \quad Tol \coloneqq \begin{cases} max\left|RES_{i,j}\right| \leq 1\% \cdot \phi_{i,j}, \ if\ \phi_{i,j} \gg 1 \cdot 10^{-6} \\ max\left|RES_{i,j}\right| \leq 1 \cdot 10^{-6}, \ if\ \phi_{i,j} \ll 1 \cdot 10^{-6} \end{cases} \qquad [5.2]$$

On certain occasions, the pseudo-steady state might not be achieved with the desire accuracy, i.e., the simulation is not converging. For that, two alternative non-convergent situations are settled down to continue the simulation:

- If the difference between $max\left|RES_{i,j}\right|$ of time *n* and time *n+1* is lower than a specific convergence accuracy threshold, then no longer converging is assumed and the current situation is accepted as the pseudo-steady state (displaying a warning alert in *Command Window* of MATLAB with the actual difference between time *n* and time *n+1*, i.e., $\left|max\left|RES_{i,j}\right|^{n} - max\left|RES_{i,j}\right|^{n+1}\right|$).
- It is established a maximum number of diffusion iterations (*iDiff*). When *iDiff* is higher than 1500, then no longer converging is assumed and the current situation is accepted as the pseudo-steady state (displaying a warning alert in *Command Window* of MATLAB with the actual error, i.e., $max\left|RES_{i,j}\right|$).

The integration continues updating each *n*-iterations the reaction term together with the integration of the microbial growth and finally, updating the pH in all the nodes of the simulation domain. When this pseudo-steady state is reached then, the mass balances of the overall reactor are integrated in a much bigger time step (dt_{bac}), function of the average microbial activity of the aggregate. Also, the biomass growth is integrated in this bigger time step. At the end of this bigger step, the Dirichlet boundary condition, the reaction term and pH are updated, therefore, the diffusion-reaction equation needs to be integrated again to reach a next pseudo-steady state. Each *n* times that the diffusion-reaction equation reaches a pseudo-steady state (dt_{div}), the cell division is checked and if it happens, the algorithm of the microbial shoving is launched.

6. Appendices

6.1. Rearrangement of diffusion-reaction equation in matrixial form

Considering the two-dimensional diffusion-reaction equation

$$\frac{\partial}{\partial t}\phi_{i,j}^{t} = \mathbb{D} \cdot \nabla_{xy}^{2}\phi_{i,j}^{t} + R(\phi_{i,j}^{t}) \qquad [A.1]$$

Where $\phi_{i,j}^{t}$ refers to the concentration of a soluble component in a position of the simulation domain (i, j) and in a time step (t), and $\mathbb{D}$ refers to the effective coefficient of diffusion. Applying the implicit Crank-Nicholson method, which is unconditionally stable, to discretize in time the diffusion term, and explicit forward Euler formula for reaction term

$$\frac{\phi_{i,j}^{n+1} - \phi_{i,j}^{n}}{h_t} = \mathbb{D} \cdot \frac{1}{2}\left[\nabla_{xy}^{2}\phi_{i,j}^{n+1} + \nabla_{xy}^{2}\phi_{i,j}^{n}\right] + R(\phi_{i,j}^{n}) \quad n \in 1 \ldots N_t \qquad [A.2]$$

Using convolution method and the *Laplacian kernel* to compute $\nabla_{xy}^{2}\phi_{i,j}^{n}$ (finite-difference method) and defining constant α as equation A.4,

$$\nabla^{2}\phi_{i,j}^{n} = \frac{1}{h^2}([L] * \phi_{i,j}^{n}) \qquad [A.3]$$

$$\alpha := \frac{\mathbb{D} \cdot h_t}{2 \cdot h^2} \qquad [A.4]$$

diffusion-reaction equation can be rewritten in matrixial form like equation A.5.

$$[\phi^{n+1}] - [\phi^{n}] = \alpha[L] * [\phi^{n+1}] + \alpha[L] * [\phi^{n}] + R([\phi^{n}]) \cdot h_t$$

$$[\phi^{n+1}] - \alpha[L] * [\phi^{n+1}] = [\phi^{n}] + \alpha[L] * [\phi^{n}] + R([\phi^{n}]) \cdot h_t$$

Multiplicative identity $\rightarrow [I_k] * \left[\phi^{n+1}\right] - \alpha[L] * \left[\phi^{n+1}\right] = [I_k] * [\phi^{n}] + \alpha[L] * [\phi^{n}] + R([\phi^{n}]) \cdot h_t$

Distributivity property $\rightarrow ([I_k] - \alpha[L]) * [\phi^{n+1}] = ([I_k] + \alpha[L]) * [\phi^{n}] + R([\phi^{n}]) \cdot h_t$

$$([\boldsymbol{I_k}] - \boldsymbol{\alpha}[\boldsymbol{L}]) * \left[\boldsymbol{\phi^{n+1}}\right] = ([\boldsymbol{I_k}] + \boldsymbol{\alpha}[\boldsymbol{L}]) * [\boldsymbol{\phi^{n}}] + \boldsymbol{R}([\boldsymbol{\phi^{n}}]) \cdot \boldsymbol{h_t} \qquad [A.5]$$

6.2. Deduction of RES equation

Reaction-diffusion equation:

$$\frac{\partial}{\partial t}\phi_{i,j}^{t}, = \mathbb{D} \cdot \nabla_{xy}^{2}\phi_{i,j}^{t} + R(\phi_{i,j}^{t}) \qquad [A.6]$$

Aim $\rightarrow$ Steady state of soluble components ($\phi_{i,j}$), that is, $\frac{\partial}{\partial t}\phi_{i,j}^{t} = 0$

$$0 = \mathbb{D} \cdot \nabla_{xy}^{2}\phi_{i,j} + R(\phi_{i,j}) \qquad [A.7]$$

Applying finite-difference method for space discretization through *Laplacian kernel* $[L]$ and coevolution method – $\nabla^{2}\phi_{i,j}^{n} = \frac{1}{h^2}([L] * [\phi^{n}])$,

$$0 = \mathbb{D} \cdot \frac{1}{h^2} \cdot ([L] * [\phi]) + R([\phi]) \quad [A.8]$$

and considering Dirichlet (or first-type) boundary condition:

$$0 = \mathbb{D} \cdot \frac{1}{h^2} \cdot ([L] * [\phi]^\gamma) + R([\phi]) \quad [A.9]$$

The units of this expression are [mol·m^{-3}·h^{-1}]. We would like the same expression but with concentration units [M]. For this, we multiply every term of equation (or *RES* expression) by characteristic diffusion time ($h^2/\mathbb{D}$). Finally, we pass mol·m^{-3} to M (1 mol·m^{-3} = 1000 mol·L^{-1}).

$$0 = [L] * [\phi]^\gamma + (h^2/\mathbb{D}) \cdot R([\phi]) \quad [A.10]$$

We assume that the system has reached the pseudo-steady state when RES is less than *Abs_tol* (1x10^{-6} M).

$$RES \coloneqq [L] * [\phi]^\gamma + (h^2/\mathbb{D}) \cdot R([\phi]) \leq Tol \quad [A.11]$$

Units

$$[L] * [\phi]^\gamma + (h^2/\mathbb{D}) \cdot R([\phi])$$

$$[ND] \cdot \left[\frac{mol}{m^3}\right] + \left[\frac{m^2}{m^2/h}\right]\left[\frac{mol}{m^3 \cdot h}\right]$$

Tolerance definition

$$Tol \coloneqq \frac{RES_{i,j}}{1 \cdot 10^{-4} + \phi_{i,j}} \leq 1\% \rightarrow Tol \coloneqq RES_{i,j} \leq 1 \cdot 10^{-6} + 1\% \cdot \phi_{i,j}$$

$$Tol \coloneqq \begin{cases} |RES_{i,j}| \leq 1\% \cdot \phi_{i,j} & if\ \phi_{i,j} \gg 1 \cdot 10^{-6} \\ |RES_{i,j}| \leq 1 \cdot 10^{-6} & if\ \phi_{i,j} \ll 1 \cdot 10^{-6} \end{cases}$$

6.3. Deduction of pH equations (equations 1.16 – 1.21)

Assuming the following acid-base system:

$$H_2O + A \overset{K_d}{\rightleftharpoons} B \overset{K_{a1}}{\rightleftharpoons} H^+ + B^- \overset{K_{a2}}{\rightleftharpoons} 2 \cdot H^+ + B^{2-} \overset{K_{a3}}{\rightleftharpoons} 3 \cdot H^+ + B^{3-}$$

where k_d, k_{a1}, k_{a2} and k_{a3} are the hydration and acid-base equilibrium constants calculated using the Gibbs energy values of formation of the substrate(s) and product(s),

$$\cdot \boldsymbol{K_d} = \frac{[A]}{[B]} \qquad \cdot \boldsymbol{K_{a1}} = \frac{[H^+][B^-]}{[B]} \qquad \cdot \boldsymbol{K_{a2}} = \frac{[H^+][B^{2-}]}{[B^-]} \qquad \cdot \boldsymbol{K_{a3}} = \frac{[H^+][B^{3-}]}{[B^{2-}]}$$

and the total concentration of the substrate (C_T) is

$$\cdot [\boldsymbol{C_T}] = \phi_{i,j}^n = [A] + [B] + [B^-] + [B^{2-}] + [B^{3-}]$$

To obtain system of equations to calculate all forms of substrates in function of $[C_T]$ and pH (or $[H^+]$), first it is defined all forms of the substrate ($[A]$, $[B]$, $[B^{2-}]$ and $[B^{3-}]$) in function of $[B^-]$ and $[H^+]$:

· (K_d) + (K_{a1}):

$$\begin{matrix} [A] = K_d \cdot [B] \\ + \\ [B] = \frac{[H^+][B^-]}{K_{a1}} \end{matrix} \Rightarrow [A] = K_d \cdot \frac{[H^+][B^-]}{K_{a1}}$$

$$[B^{2-}] = \frac{K_{a2}\cdot[B^-]}{[H^+]}$$

· (K_{a2}) + (K_{a3}): $+$ $\Rightarrow [B^{3-}] = \frac{K_{a2}\cdot K_{a3}\cdot[B^-]}{[H^+]^2}$

$$[B^{3-}] = \frac{K_{a3}\cdot[B^{2-}]}{[H^+]}$$

$$[C_T] = K_d \cdot \frac{[H^+][B^-]}{K_{a1}} + \frac{[H^+][B^-]}{K_{a1}} + [B^-] + \frac{K_{a2}\cdot[B^-]}{[H^+]} + \frac{K_{a2}\cdot K_{a3}\cdot[B^-]}{[H^+]^2}$$

Then, $[B^-]$ definition in function of $[C_T]$ and $[H^+]$ is obtained by the new expression of $[C_T]$:

$$C_T = K_d \cdot \frac{[H^+][B^-]}{K_{a1}} + \frac{[H^+][B^-]}{K_{a1}} + [B^-] + \frac{K_{a2}\cdot[B^-]}{[H^+]} + \frac{K_{a2}\cdot K_{a3}\cdot[B^-]}{[H^+]^2} \rightarrow$$

$$C_T = \left[(1+K_d)\cdot\frac{[H^+]}{K_{a1}} + 1 + \frac{K_{a2}}{[H^+]} + \frac{K_{a2}\cdot K_{a3}}{[H^+]^2}\right]\cdot[B^-] \rightarrow$$

$$[B^-] = \frac{[C_T]}{(1+K_d)\cdot\frac{[H^+]}{K_{a1}} + 1 + \frac{K_{a2}}{[H^+]} + \frac{K_{a2}\cdot K_{a3}}{[H^+]^2}} \rightarrow$$

$$[B^-] = \frac{K_{a1}\cdot[H^+]^2\cdot[C_T]}{(1+K_d)\cdot[H^+]^3 + K_{a1}\cdot[H^+]^2 + K_{a1}\cdot K_{a2}\cdot[H^+] + K_{a1}\cdot K_{a2}\cdot K_{a3}}$$

Finally, the new expression of $[B^-]$ is applied in the definition of $[A]$, $[B]$, $[B^{2-}]$ and $[B^{3-}]$ obtained previously, yielding:

$$[A] = K_d\frac{[C_T][H^+]^3}{\theta}$$

$$[B] = \frac{[C_T][H^+]^3}{\theta}$$

$$[B^-] = K_{a1}\frac{[C_T][H^+]^2}{\theta}$$

$$[B^{2-}] = K_{a1}K_{a2}\frac{[C_T][H^+]}{\theta}$$

$$[B^{3-}] = K_{a1}K_{a2}K_{a3}\frac{[C_T]}{\theta}$$

where $\theta = (1+K_d)\cdot[H^+]^3 + K_{a1}\cdot[H^+]^2 + K_{a1}\cdot K_{a2}\cdot[H^+] + K_{a1}\cdot K_{a2}\cdot K_{a3}$

7. References

1. Kreft, J.-U., G. Booth, and J.W.T. Wimpenny, *BacSim, a simulator for individual-based modelling of bacterial colony growth.* Microbiology, 1998. **144**(12): p. 3275-3287.
2. Kreft, J.-U., et al., *Individual-based modelling of biofilms.* Microbiology, 2001. **147**(11): p. 2897-2912.
3. Kissel, J.C., P.L. McCarty, and R.L. Street, *Numerical simulation of mixed-culture biofilm.* Journal of Environmental Engineering, 1984. **110**(2): p. 393-411.
4. Batstone, D.J., et al., *Towards a generalized physicochemical framework.* Water Science and Technology, 2012. **66**(6): p. 1147-1161.
5. González-Cabaleiro, R., J.M. Lema, and J. Rodríguez, *Metabolic energy-based modelling explains product yielding in anaerobic mixed culture fermentations.* PLoS ONE, 2015. **10**(5): p. 1-17.
6. Ypma, T.J., *Historical development of the Newton-Raphson method.* Society for Industrial and Applied Mathematics, 1995. **37**(4): p. 531-551.
7. Kleerebezem, R. and M.C.M. Van Loosdrecht, *A generalized method for thermodynamic state analysis of environmental systems.* Critical Reviews in Environmental Science and Technology, 2010. **40**(1): p. 1-54.
8. Heijnen, J.J. and J.P. van Dijken, *In search of a thermodynamic description of biomass yields for chemotrophic growth of microorganisms.* Biotechnology and Bioengineering, 1992. **39**(8): p. 833-858.
9. González-Cabaleiro, R., et al., *Microbial catabolic activities are naturally selected by metabolic energy harvest rate.* ISME Journal, 2015. **9**(12): p. 2630-2641.
10. Tijhuis, L., M.C.M. Van Loosdrecht, and J.J. Heijnen, *A thermodynamically based correlation for maintenance gibbs energy requirements in aerobic and anaerobic chemotrophic growth.* Biotechnology and Bioengineering, 1993. **42**(4): p. 509-519.
11. Pikulin, V.P. and S.I. Pohozaev, *Equations in Mathematical Physics*. Modern Birkhäuser Classics. 2001: Birkhäuser Basel. 207.
12. Briggs, W., V. Henson, and S. McCormick, *A Multigrid Tutorial, 2nd Edition*. 2000.
13. Kreft, J.U., et al., *Individual-based modelling of biofilms.* Microbiology, 2001. **147**(11): p. 2897-2912.
14. Samet, H. *An Overview of Quadtrees, Octrees, and Related Hierarchical Data Structures*. 1988. Berlin, Heidelberg: Springer Berlin Heidelberg.