ChemChaste: Installation and running guide

November 2020

Abstract

ChemChaste is a software suite produced to simulate the complex chemistry of systems of cells coupled with their environment. Rather than focusing solely on the cells themselves, ChemChaste simulates the environment as a reaction-diffusion system which then coupled this to a population of cells. To do this a hybrid discrete-continuous model has been implemented within the ChemChaste source code and an interface method for running tailored experiments has been provided. The interface has been developed to be user friendly with minimal code knowledge required. This file documents the steps needed to acquire the Chaste base code onto which ChemChaste builds and also set up and run the first ChemChaste tests.

Contents

1 Introduction

ChemChaste has been developed with the aim of modelling the complex chemical interactions of cell populations and their environments. These models are performed through solving a reaction-diffusion (Pde) system and coupling the solution to an off lattice cell based simulation. The Pdes are solved using a finite element method solving for the steady state of the spatial reaction systems at the nodes of a mesh. The spatial reaction systems are defined on a nodal basis alongside defining the chemical diffusion coefficients nodally. Therein subdomains may be defined by the user by controlling these nodal definitions. The nodal results are then interpolated across the space into a finer pixelated surface. The surface pixels may house a point-like particle representing the location of a centre of a cell which acts as a discrete agent.

The cells are defined on a separate simulation layer interfacing with the Pde mesh at the locations of the cellular agents. Each cell runs their own cell cycle models and subcellular reaction networks (SRN) and couples to the Pde pixel through localised reaction and transport processes. The cell cycle models provide dynamism to the population of cells wherein maximum and minimum threshold values of the cell concentration values. When the internal cells concentrations go above their maximum threshold the cell divides into two daughter cells. This cell division is performed through introducing a new cell object at a random orientation from the parent, both forming the two daughter cells. The parent cell concentrations are divided between the two daughter cells and properties and reactions defined in the parent are passed on. Similarly, when the cell concentrations fall below the minimum thresholds the cell is marked for apoptosis and is removed from the population. These cell dynamics build upon the cell-based methods provided in Chaste with the addition of the domain Pdes coupling.

ChemChaste builds upon the methods developed in the Chaste software system. Chaste provides the simulation backbone to ChemChaste utilising finite element solvers and cell-based simulation techniques. ChemChaste adds chemical methods to Chaste and develops the Pde simulations to work for a general sized Pde system and couple to the cell-based methods.

To install and run ChemChaste one first needs to install and be able to run simulation in Chaste. To do this it is recommended that the Docker container systems and the Docker image of Chaste be installed alongside Paraview for visualisation. Interfacing with Docker and the Chaste image is done through terminal commands however for this using Visual Studio Code with the Docker plugin is recommended. ChemChaste is added as a user project into the Chaste hierarchy and a file based front end with a python control script is provided. Instructions on installing, Chaste, ChemChaste and running ChemChaste experiments are provided below. It should be noted that while Chaste is built to run on HPC machines these instructions are primarily concerned with running on a single machine.

2 Acquiring and installing Chaste

Docker is a container system which allows software to be rapidly deployed to work alongside varying operating systems. It provides a quick and reliable method to access the functions and capability of the Chaste software suite and therein specifically ChemChaste. Platform specific instructions to install Docker may be found on the Docker website *(https://hub.docker.com/search?offering=community&type=edition)*.

The Chaste Docker image may be found on the Chaste GitHub page *(https://github.com/Chaste/chaste-docker)*. Install instructions and the steps to get a container running are provided on the GitHub page alongside the structure of the Chaste image. Depending on the machine's native operating system, Docker configuration settings may need changing to ensure enough RAM and cores are made available to the image. Details on this are provided within the GitHub page and the links therein. Python scripts to create and build a user project are provided within the Docker container and a user project will be updated with the ChemChaste codes.

For running ChemChaste, it is recommended that symbolic links between the Docker volume directories and the native system directories are recommended. Specifically linking to the "**/home/chaste/lib**", "**/home/chaste/projects**", and "**/home/chaste/testoutput**" directories. For example, the run command for a Chaste Docker image linking to directories within a directory called "**ChemChasteLocal**" is;

```
Docker run -it --rm -v chaste_data:/home/chaste -v
~/Documents/ChemChasteLocal/projects:/home/chaste/projects -v
~/Documents/ChemChasteLocal/lib:/home/chaste/lib -v
~/Documents/ChemChasteLocal/testoutput:/home/chaste/testoutput chaste/release
```

Linking the lib and projects directories provides an easy to use interface between the users file system and the Docker container. This linking allows code and experiments to be written on the familiar local system and then later added to the Chaste container using a native file explorer. An all-in-one solution for writing and running experiments in ChemChaste is to use Visual Studio Code with the Docker plug-in.

3 Running Chaste within Visual Studio Code

The Chaste may be used within the Docker container by copying files written locally to the symbolically linked directories. The code executables and test files may then be run through the terminal in which the Docker container was "run". Alternatively, the cross platform Visual Studio Code (VSC) code editor may be connected to the Docker container. This allows the code to be edited and run in the same place. For this the Docker extension for VSC is needed and a shell instance will be attached to the Chaste container.

The VSC Docker extension may be found and installed from the Visual Studio marketplace (*https://marketplace.visualstudio.com/items?itemName=ms-azuretools.vscode-docker*). The Docker logo will appear on the VSC explorer ribbon and, within the Docker extension, under the "containers" drop down menu the Docker containers running on the system will be shown. By "right-clicking" on the active Chaste container and selecting "**Attach Shell**" a terminal will be initialised from which the Chaste commands may be executed. To edit the code within the container, navigate to the symbolically linked directories using the VSC file explorer and select the project codes at will.

4 Running a Chaste test

Chaste uses the ctest system to run simulations in a controlled and reproducible way. A variety of test scripts are provided within the Chaste trunk code which may be run to verify the install. Running a test in Chaste follows a prescribed order of 4 instructions which may be executed in the container terminal (either the terminal where "**Docker run**" command was executed or the VSC terminal):

1. **cd /lib** Navigate to the library directory where the executable codes are found.
2. **cmake /src** Sets up the make system for compiling and linking code files, this command is needed when new files are added to the Chaste project for example. Sets up the executables for the test code headers.
3. **make -j4 TestName** This is the standard make command for compiling the code in the "**TestName**" executable. For example, replacing "**TestName**" with "**TestHello**" will run a "hello-world" script from a project directory. The -j4 option is a make argument for permitting the code to run on 4 cores of a multicore processor. The number of cores, "**4**", may be replaced at will or the argument, "**-j4**" may be omitted entirely. Running the code in parallel may be desirable for some of the computationally intensive steps. However not all tests may run in parallel but the option will be disabled within the test itself.
4. **ctest -V -R TestName** This command executes the "**TestName**" test script using the ctest framework.

5 Building a new project in Chaste

In the Chaste software system the user written code is contained in a user project separate from the trunk code. Therefore a user needs to build a project which will integrate with the trunk code during the cmake call. Python scripts have been provided in the Docker Chaste container to assist with creating and building a new project. To create a new project;

1. **cd /scripts** navigate to the directory containing the python scripts
2. **./new_project.sh NewProjectName** This script sets up a new directory "**NewProjectName**" in the Projects folder and adds the necessary cmake commands. Into this new directory the new_project script clones the template_project from the Chaste GitHub page then calls the setup_project script. A sequence of commands are given in the terminal for the user to specify which components of the Chaste trunk code are necessary for the new project. This will set up the correct dependencies and speeds up compile time if some dependencies are not used. If the user set up the symbolic links when running the Docker container then new files will be added to the linked projects file on the local system.
3. **./build_project.sh NewProjectName c** Builds the user project, setting up and compiling the project source code and then runs the ctest command to ensure the files have been set up correctly. From here it will be possible to run the specific tests as indicated in earlier sections.

6 Writing a new test in a Chaste Project

Simulations in Chaste are written in test files. After building a project a new test may be composed through adding a new "*.hpp*" header file with the filename beginning with "Test". This file is added within the new project directory; "**projects>NewProject>test>TestNewSimulation.hpp**". The contents of the "**TestNewSimulation.hpp**" file should follow the template described in the Chaste files on the Wiki (*https://chaste.cs.ox.ac.uk/trac/wiki/UserTutorials*). Additionally, for the build and make structure to correctly integrate the new test with the Chaste trunk code the name of the new test,
"**TestNewSimulation.hpp**" here, needs adding to the
"**projects>NewProject>test>ContinuousTestPack.txt**" file. Run the test just like any other test, starting with cmake as a new test file has been added to the Chaste system.

For the user to create their own Chaste-based simulations, the best place to start is the tutorials on the Chaste Wiki, linked above. However as a practical strategy, it can be suggested to create a new project, copy, rename, and paste an existing Test file under the "test" folder under that project, edit the Test file according to the simulation needs and then compile and run it as any other Test. For example, to simulate an Ode system, the "**TestSolvingOdesTutorial.hpp**" can be copied, renamed and pasted under the "test" folder of the project. Then the test may be edited to suit the Ode needs.

7 Visualising cell-based test results using Paraview

Chaste writes the user test output results into the "**testoutput**" directory under a subdirectory specified in the tests themselves. The open-source software Paraview is recommended for visualising domain Pde and/or cell based results. The steps for visualising both the Domain Pde and cell based results may be performed separately or together for a combined simulation. All the data needed for Paraview, whether Pde or Cell-based or combined, will be found in the test directory in testoutput.

7.1 Visualising Domain Pde results

Chaste outputs Pde results as a series of *.vtu* files. There is a series of files for each individual Pde state variable in the system with each file appended with the solver timestep. The series of files are opened within Paraview and by selecting the "**Apply**" option will display the domain within the layout window. Pde results are visualised by selecting the values of the Pde variable as the display data, instead of the default "**Solid Color**" option, and selecting "**Rescale to data range over all timesteps**" from the toolbar. The animation may be controlled through the options in the toolbar. Display options may be found in the "**Properties**" window and tabs therein. Further information may be acquired through plotting the results over a slice of the domain or averaged over the whole domain.

Projecting the data to a single slice may be useful for investigating phenomena such as travelling waves in the Pde system. To apply the projection highlight the Pde *.vtu* series and right click to access the "**Add Filter>Data Analysis>Plot Over Line**". The new line filter will be displayed in the pipeline browser and by selecting the "**Apply**" option a line chart will be displayed. The line over which the data slice will be taken will be displayed over the domain and the position and direction may be manipulated. The line slice may be manipulated either graphically by moving the head or tail in the domain window or by changing the line parameters in the properties window.

System wide information may be gathered by considering the Pde values averaged over the whole domain. For example, system wide oscillations and spatial variance may be evident in the average value traces. To apply this averaging filter, highlight the Pde *.vtu* series "**Add Filter>Data Analysis>Plot Data Over Time**" and select "**Apply**". From the properties window trace statistics may be selected, including "**Average**", "**Quartiles**", and "**Ranges**".

The line charts and data filters are separate entities in Paraview. Therefore multiple line or time filter data may be plotted on a single line chart. For this select the desired line chart to display the data on, highlight each filter in the pipeline browser and select "**Apply**" on each filter. Perusing the "**Properties**" window of the filters when applied to the line chart will reveal the chart display options which may be modified. When the display has been finalised the animations and line chart screenshots may be saved under the "**File**" tab.

7.2 Visualising Node based cell results

The cell-based package of Chaste contains a range of simulation methods for investigating cell based models. ChemChaste utilises the Node based cell population routines in Chaste. Here the centre of the cell specifies nodes in a honeycomb mesh with interactions between the cells are defined by a cutoff region rather than by direct connectivities. When new cells are added through cell division, a new node is added to the mesh within the cutoff region at a random orientation around the parent cell. The surrounding cells move to accommodate the new cell addition. Movement in the mesh topology is specified through a force class which is written in the test script. The tests output the cell locations and any data writers in the test script. This data may be analysed through defining Glyph objects in Paraview.

Cell based tests output the simulation data in **results.pvd** and **results.vtu** files. The *.pvd* file contains the information necessary to reconstruct the potential data for the cells in the mesh while the *.vtu* file series contains the internal cell data. Therefore both sets of data are required to visualise the cell processes, so are imported into Paraview and are viewable in the "**Pipeline Browser**". For the case of the node based cell simulation the data files require the application of filters.

The cells at the cell locations are visualised through defining "**Glyph**" object filters. To apply the filter highlight the **results.pvd** file then "**Add Filter>Common>Glyph**" and "**Apply**". From here display properties may be selected in the "**Properties**" window. For example, to view the cells as a population of spherical objects change the "**Glyph Type**" option to "**Sphere**", "**Glyph Mode**" to "**All Points**" and "**Coloring**" and "**Scale**" options to cell data to show the changing of cell colour size or size. For information for visualising cell based results may be found on the Chaste wiki (*https://chaste.cs.ox.ac.uk/trac/wiki/UserTutorials/VisualizingWithParaview*). To view cell based results coupled to Pde results import both data sets and visualise as described but apply both domain Pde and glyphs onto one animation.

8 Acquiring and running simulations in ChemChaste

ChemChaste builds upon the Chaste trunk code providing the means to simulate chemical reactions in the domain, reactions within cells and transport processes across cell membranes. These code editions are packaged as a user project to be added to the Chaste system. To compile the ChemChaste code; build a new project with the project name "**ChemChaste**" then copy the contents of the ChemChaste directory on the GitHub page into the new project. Run the "**cd /lib**", "**cmake /src**", and "**make -j4 TestChemChaste**" command as previously described. This will verify the install.

8.1 Python scripts

In the GitHub page Python scripts are provided to assist with running multiple simulations and producing the config files. These files need adding to the " **/lib/**" directory of the Chaste system and they may be executed from there. There are two python files that control the simulation: "**RunChemChaste.py**" the main command file and "**ChemChasteDefinitions.py**" an definition import file. The RunChemChaste files contain a list of configuration files which specify the experiment to simulate. Based on the configuration file, a script runs the ChemChaste executable for each experiment as parallel processes. To run the python command script; "**cd /lib**" to navigate to the library and run the python executable "**python RunChemChaste.py**".

9 Tailoring experiments

The individual simulation experiments are detailed in a series of text based configuration files with paths to files detailing the domain topology and reaction systems. The paths to these configuration files themselves are to be written in **RunChemChaste.py**. This file based system provides a user friendly method of implementing reaction systems as chemical equations and domain topologies as graphical maps.

9.1 Configuration files

The parameters and settings required to run a ChemChaste simulation are specified in the configuration, or config, file corresponding to the experiment of interest. The config file contains sections detailing the sections to add in a user's experiment. Note that if a section is omitted it may be provided with default values and overridden by the user in the RunChemChaste python script. For reaction-diffusion only experiments, two sections are required to be defined; the simulation settings and the reaction-diffusion domain information. If a user wishes to couple a cell based model to the reaction-diffusion domain then further sections, cell population topology, and at least one cell object is to be defined.

9.1.1 Simulation settings

The simulation settings are defined in all config files.

- "**output_filename**" provides the path to the directory that the output data from ChemChaste will be written. This path is relative to the "**testoutput**" directory in which the general Chaste test outputs are located.
- "**simulation_end_time**" specifies the maximum timestep for the simulation to run over. The value may be scaled using the "Temporal Shift Scale" filter in Paraview.
- "**number_of_reaction_pdes**" the dimension of the reaction-diffusion Pde to be solved. This will be the number of state variables for the solver to solve for i.e the number of diffusing chemical species in the domain.
- "**spatial_dimensions**" the dimension of the domain space, 2D plane simulations are the standard simulation for Chaste looking at cell monolayers.
- "**FE_element_dimension**" the dimension of the individual mesh elements used by the finite element solver methods for the Pde.

9.1.2 Reaction-diffusion domain

Running simulations with an easy to implement chemistry is one of the aims of ChemChaste. To that end a file based system to describe the topology and diffusive properties of the domain and list the chemical reactions occurring in the system has been provided. In the config file the paths to these files is provided:

- "**domain_file_root**" the absolute path to the directory containing the files specifying the domain.
- "**domain_file**" *.csv* file defining the topology of the domain. Labels are used to define the different subdomains of the system. The labels are then written to the domain_key_file.
- "**domain_key_file**" *.csv* file containing the subdomain labels and their key used in the domain *.csv* file.
- "**ode_file**" *.csv* file for the domain specifying which ode reaction systems are occurring at which location in the domain. This file may be the same as the domain_file.
- "**ode_key_file**" *.csv* key file for the chemical reaction systems in the domain. The keys associate the labels in the ode_file with the name of the reaction system *.txt* file.
- "**diffusion_database**" *.csv* file detailing each diffusive species in the system within each of the subdomains defined in the domain_key_file. The order of the file is; Chemical name, subdomain label, diffusion value constant.
- "**initial_conditions**" The initial values of each chemical species may vary on a subdomain basis. The file order is Chemical name, subdomain label, initial value within the subdomain, and a boolean value. The boolean value is a switch to specify whether to perturb the subdomain value on a nodal basis with the addition of a mean zero uniform random value.
- "**boundary_conditions**". The boundary conditions (BCs) for the reaction-diffusion Pde are specified around the boundary of the domain. The boundary types available are Dirichlet, where the boundary is held at a constant value, or Neumann, which specifies the rate of change of the state at the boundary. The order of the file is; chemical species name, BC type (Dirichlet or Neumann), and the BC value. For an isolated system Neumann BCs with zero value are appropriate.

9.1.3 Cell population topology

ChemChaste has been designed to run simulations that couple a cell population to the reaction-diffusion Pde layer. Therefore a second cell layer with a topology and cell labelling needs defining. This may be done in the same way as producing the domain layer topology or by constructing a cell population using the parameters in the config file.

- "**number_cells_across**" for a 2-dimensional cell layer, the number of cells across the layer.
- "**number_cells_high**" for a 2-dimensional cell layer, the number of cells high in the layer.
- "**node_cutoff_length**" the radius of the region around a node to define the connectivity and nearest neighbours of the off lattice cell population.
- "**cell_mesh_origin**" the location of the origin of the cell mesh when compared to the larger domain mesh.
- "**linear_force_cutoff**" the cutoff length for a linear spring-like force acting between two cells.

9.1.4 Cell object

Simulations in ChemChaste act upon cells with their own internal chemistries and couple with the environment through transport processes. Subsequently, the cell based simulation requires the definition of the cell properties. These properties specify the reactions occurring at and within the cell membrane, the cell cycle model and cell state, and the initial conditions and division conditions of the cell.

- "**cell_file_root**" the absolute path to the directory containing the files specifying the cell properties.
- "**cell_initial_conditions**" the initial concentration values for the chemical species within the cell.
- "**membrane_property**" *.txt* file containing the membrane bound reactions for the cell.
- "**transport_property**" *.txt* file containing the transport processes/reactions occurring across the cell membrane.
- "**srn_file**" *.txt* file containing the reactions occurring within the cell. These for the subcellular reaction network (SRN).
- "**cell_cycle_file**" *.csv* file containing the maximum and minimum thresholds for the cell bound species. When the cellular concentration of a single species falls below the minimum threshold the cell is marked for apoptosis. When a single species concentration goes above the maximum threshold the cell division process is triggered.

9.2 Reaction file interface

The python scripts provided with ChemChaste are the main method of running the ChemChaste simulations. They allow for parameter sweeping and ensembles of experiments to be investigated in an efficient, paralised method without needing to re-write the source code. The source code doesn't need modifying to tailor the simulations to an individual experiment. For this, a file based system has been provided to interface with the ChemChaste executables in a user friendly way. The interface is composed of two sections, specifying the reactions in/on the cell and in the domain.

One motivational aspect of ChemChaste is to present the reaction dynamics in an easy to implement way. For this reactions are given in the standard chemical reaction notation rather than in a mathematical form. The chemical reaction systems are written into .txt files as a list of separate reactions with the reaction kinetic law, the reaction itself, and the kinetic parameters. The reactions fall within four domains; extracellular, membrane bound, transport processes, and subcellular. These domains specify the concentration vectors used in the reaction ode systems and subsequently require separate kinetic laws. The kinetic laws are implemented within the ChemChaste source code and may be called through their identifying names. For the extracellular and subcellular domains currently packaged reaction types are:

- ZerothOrderReaction
- ZerothOrderReversibleReaction
- MassActionReaction
- SpectatorDependentReaction
- MichaelisMentenReaction

For transport reactions where there is a flux across the cell membrane:

- ZerothOrderTransportIntoCell
- ZerothOrderTransportOutOfCell
- ZerothOrderReversibleTransport
- MassActionTransportReaction

For membrane bound reactions where the reaction kinetics depends on the species concentrations either side of the membrane but where there is no species flux across the membrane:

- ZerothOrderCoupledMembrane
- ZerothOrderReversibleMembrane
- MassActionCoupledMembraneReaction

Each reaction specifies the parameters that are required to be provided in the file.

9.3 Further modifying ChemChaste capabilities

9.3.1 Adding new chemical reaction kinetics