PSim Installation and Release Notes

Release 1.0.0

The Polyswift Team

May 09, 2013

Contents

1 Introduction

Welcome to the PSim, the self-consistent field theory code for simulating mesoscale structure of complex block copolymer mixtures. PSim consists of two major components:

- PolySwift++ computational engine
- PSimComposer graphical user interface

PSimComposer provides an interface that allows you to edit and validate your simulation input files, run PolySwift++ simulations, and visualize results using the VisIt-based Visualization pane embedded within PSimComposer. You can also edit PolySwift++ input files in the text editor of your choice, perform calculations with the easy-to-run, command-line-driven PolySwift++ executable, and then run the visualization tool of your choice.

PSim is supported on:

- 64-bit Linux
- Macintosh OS X Mountain Lion (10.8)
- Macintosh OS X Lion (10.7)
- Macintosh OS X Snowleopard (10.6)
- Windows XP and above (32 and 64-bit)

A serial version of PolySwift++ is available for running on single processor workstations. A parallel version of PolySwift++ is provided for multi-core systems that support the Message Passing Interface (MPI).

More information about PSim can be found at the PSim Product Website. Send questions about installing or running PSim to Tech-X Customer Support at support@txcorp.com. For technical questions about the physics, users may wish to email the PolySwift++ discussion list (polyswift-users List). Extensive assistance in the use of PolySwift++ engine or PSim simulations in general is available from Tech-X Professional Services. Please contact Tech-X directly for sales, consulting, and other questions at sales@txcorp.com.

2 Installation

For all operating systems, installation places

- PolySwift++ computational engine
- PSimComposer, the graphical user interface to PolySwift++
- PSim documentation (.pdf files)

2.1 PSim Installation Instructions

Windows PSim Software Installation

The PSim distribution package for Windows, both 32-bit and 64-bit, is a self-extracting executable. Invoke the installer by double clicking on it. The default installation path is:

For 32-bit application on 32-bit Windows:

```
C:\Program Files\Tech-X\PSim 1.0
```

For 32-bit application on 64-bit Windows:

```
C:\Program Files (x86)\Tech-X\PSim 1.0
```

For 64-bit application on 64-bit Windows:

```
C:\Program Files\Tech-X (Win64)\PSim 1.0
```

To open the PSim software, go to the Start Menu, click on the Tech-X folder, click on PSim 1.0, then click on PSimComposer. See *Dialogs for Windows installation.*.

Mac OS X PSim Software Installation

The PSim distribution package for Mac OS X is a .dmg installer. Invoke the installer by double clicking on it. Drag the PSim-1.0.0 folder into your Applications folder (visible in the installer window). From the Application folder, double click on the PSimComposer icon in the PSim-1.0.0 folder. See *Dialog for the Mac installer.*. This default installation path is:

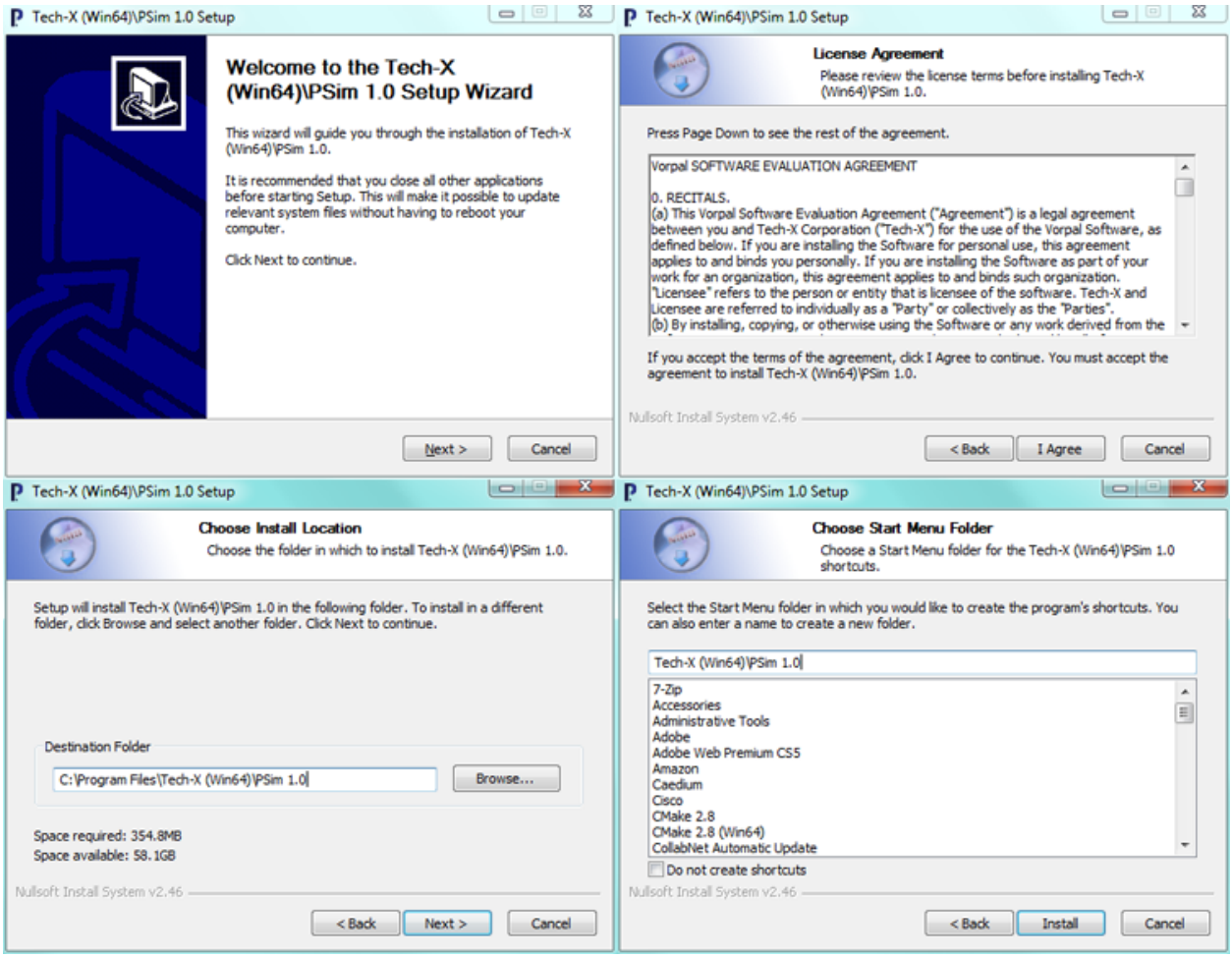

Figure 1: Dialogs for Windows installation.

```
/Applications/PSim-1.0.0
```

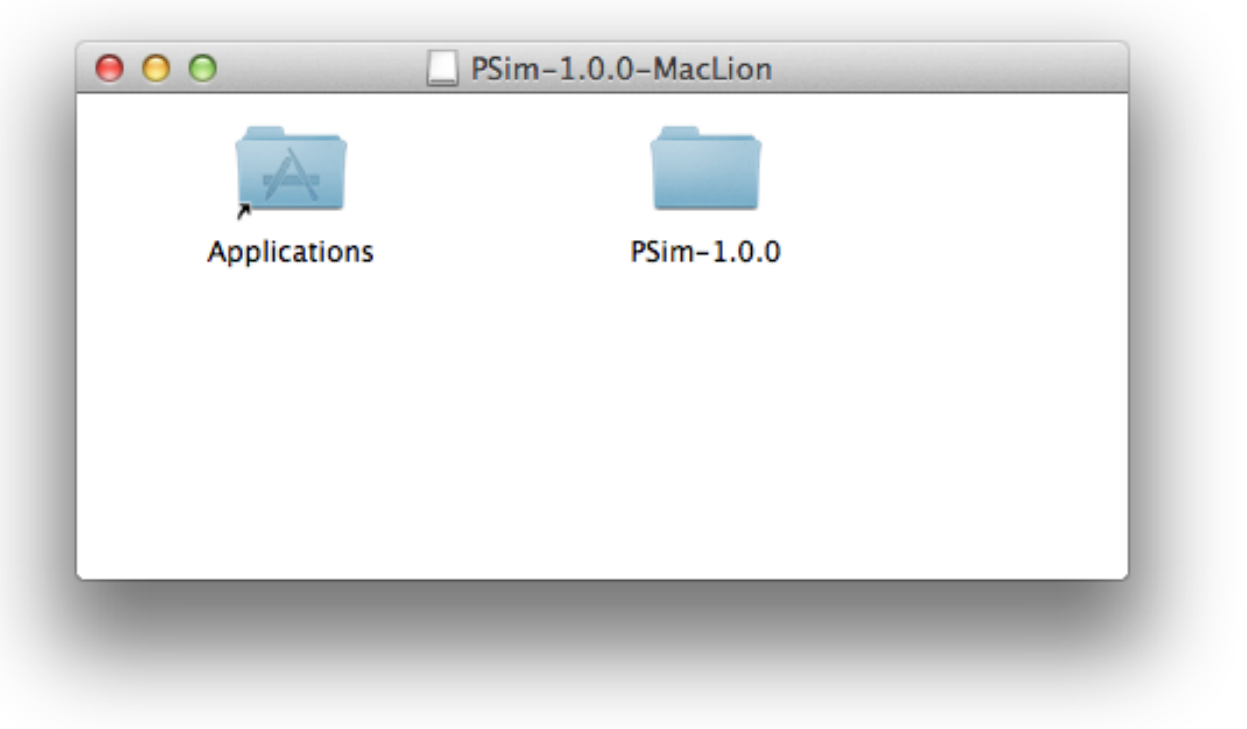

Figure 2: Dialog for the Mac installer.

Linux Software Installation

The PSim distribution package for Linux is a gzipped tarball. Unpack the gzipped tarball into the directory in which you wish to install PSim. A typical location would be

```
/usr/local/PSim
```

The unzip and untar command is

```
$ tar -zxvf PSim-1.0.0-Linux64.tar.gz
```

To run PSimComposer, execute the command

```
$ cd PSim-1.0.0-Linux64
$ ./PSimComposer.sh
```

from within the installation directory.

2.2 Included Software

Upon completing the installation process, the following software will be installed on your computer.

PSimComposer

PSimComposer is a graphical user interface for

- Creating and editing PolySwift++ input files
- Executing PolySwift++ engine
- Analyzing PolySwift++ generated data
- Visualizing PolySwift++ generated data
- Viewing the documentation.

The PSimComposer editor and validator have built-in functions and graphical components that help you to create well-formed input files. Example input files, ranging in complexity from beginning to advanced, are included with PSimComposer. New PolySwift++ users can use these examples as templates.

The PSimComposer Run pane invokes PolySwift++ with user definable settings for number of steps, number of data dumps, and restart file, if any. It also allows selection of serial versus parallel PolySwift++.

PSimComposer now allows selection of analysis programs, either supplied with PolySwift++ or user written.

The visualization in PSimComposer is provided by the graphical analysis tool VisIt. VisIt is embedded within PSimComposer. Data generated by PolySwift++ or by analysis programs automatically appears in the Visualization pane.

All documentation can be seen from within PSimComposer, fully cross-referenced.

PolySwift++

PolySwift++, the computational engine, runs both as a serial and parallel code for multi-processor / multi-core systems that support MPI. PolySwift++ now comes in the specialized PSim packages. PolySwift++ is embedded within PSimComposer.

Python

Python is an open-source, interpreted scripting language managed by the Python Software Foundation. For more information about Python, see: http://www.python.org/

PSim uses Python to process input files, allowing users to set up simulations with math functions, variable substitutions, and macros.

PolySwift++ uses its own embedded version of the Python interpreter to pre-process input files and execute any Python code in an input file. Python is also included within PSimComposer.

MPI

The Message Passing Interface (MPI) is an application programming interface (API) for communicating between processes that execute in parallel. There are many implementations. The Linux and Mac versions come with the OpenMPI (http://www.open-mpi.org/) implementation of MPI. The Windows versions come with the Microsoft MPI implementation (http://www.microsoft.com/). The appropriate MPI implemention is embedded within PSimComposer.

2.3 PSim Documentation

All documentation is accessible from within the PSimComposer interface, as well as online at the PSim Support Website.

3 Release Notes

The release notes describe new features for the PSim computational engine (PolySwfit++) and the PSimComposer graphical user interface.

3.1 New and Updated Features

PSim Computational Engine (PolySwift++)

In the PSim 1.0.0 release we introduce the new PSim packages providing users with affordable simulation solutions targeted at their application area.

- **PSimBase (Base PolySwift++ Simulation package)**
  - PSimBase is a flexible, multiplatform, numerical Self-Consisient Field Theory (SCFT) simulation tool for running computationally intensive block copolymer mesoscale problems.
- **PSimPlus (PolySwift++ Simulation package for Confinement/Nanocomposites)**
  - PSimPlus solves the numerical SCFT equations for block copolymer mixtures in arbitrary confinement and with movable nanoparticle inclusions
- **Computational features, including**
  - Multiblock, multicomponent mixtures
  - Spectral filtering for removing topological defects
  - Zone annealing
  - Arbitrary confinment, with interactions
  - Hybrid-SCFT algorithm for nanocomposites

PSimComposer GUI Features

The PSimComposer graphical user interface (GUI) allows the user to edit simulation input files, run simulations in either serial or parallel (thereby utilizing multiple cores or even computational nodes that do not share memory), and visualize results.

- Complete GUI - Easy interface for managing licenses - Dialog for saving images
- Help - Access to full hierarchical documenation
- Setup stage - Opening existing runs - Creating runs from template examples - Key parameters easily adjustable in examples - Full editor for input files - Search capability for input files
- Run stage - Editing of run parameters - Control serial/parallel run modes - File browser - Log views - View of standard engine output
- Visualize stage - View driven by structure of data file - History view - Control over labels, color and rendering - Time slider - Save images to file - Visualize while run is being performed
- Analyze stage - Running python files on data (user/Tech-X supplied)