

# Primer on use of the SWIM IPS

D. B. Batchelor

3/18/2011

This document describes the absolute basics of how to get access to the SWIM Integrated Plasma Simulator (IPS) and to run some very simple demo simulations. Two example simulations are provided – a “Hello World”, which requires no input files and does not require the plasma state, and a “model simulation” that behaves very much like a real simulation but does no physics. Both have been run on frankin at NERSC and on stix at PPPL. A step-by-step outline is provided in tables in the Appendix.

## Getting, building and setting up the SWIM IPS

The IPS software is maintained in a Subversion (SVN) repository at <https://cswim.org/svn/cswim>, hosted on the ORNL SciCompForge service. To access it you must register as a SWIM member, which can be done at our web site at <http://cswim.org>. Once this is completed, and you have been approved by our repository gatekeeper, David Bernholdt ([bernholdtde@ornl.gov](mailto:bernholdtde@ornl.gov)), you will be able to check out the IPS. Your computer will need to have an up to date version of Subversion if you want to check it out locally. SVN is already installed on all of the computers on which the project supports build systems, so you don't need to do anything. Supported systems include: franklin and hopper at NERSC, stix at PPPL, jaguar at ORNL, ITER at Tech-X.

The SWIM repository contains a number of top level directories, but for purposes of this primer, and in fact for most user purposes, the only part of the IPS that needs to be checked out is */ips/trunk/*. A command line that will do this is:

```
svn co https://<YourUserName>@cswim.org/svn/cswim/ips/trunk <YourIPS_Identifier>.
```

You can name your working copy whatever you like and put it wherever you like. It is sometimes useful to have more than one copy present.

To simplify the Makefile, which must support a number of computers, a make configuration file has been written for each supported platform. These files are found in the */config/* directory at the top level of the IPS trunk, i.e. in */cswim/ips/trunk/config/*. Before trying to build it is necessary to copy or establish a symbolic link from the appropriate file to *makeconfig.local*, e.g.

```
makeconfig.local → makeconfig.franklin
```

Also a number of environment settings and module loads are needed. This is easily done by sourcing the *swim.bashrc.<platform>* file for your machine, located at the top level of the IPS tree (e.g. *swim.bashrc.franklin*). It is probably most convenient to do this in the user *.bashrc* file by first exporting an environment variable *IPS\_ROOT* that points to the top of your IPS tree, and then sourcing *\$IPS\_ROOT/swim.bashrc.<platform>*.

The `IPS_ROOT` environment variable is also useful in other contexts. The command lines and batch scripts used in these examples make reference to the `IPS_ROOT` variable, so the user should indeed define it. It is a convenience and not strictly necessary but it does simplify the batch scripts. Henceforth in this document we will refer to the top level of the IPS tree as `IPS_ROOT`.

With these things done it is only necessary to issue “make” at the command line, and when that is completed to do “make install”. The “make” builds the plasma state software, and builds any fortran codes that reside in the `/components/` directory of the IPS. It can take an hour or so to do a top level make.

Note that the IPS does not attempt to build the physics fortran codes that implement the various physics components, nor do these codes reside in the IPS `/components/` directory. The physics codes are maintained and built by the individual code authors. The physics executables are collected in a separate, non-version-controlled directory called `PHYS_BIN`. That does not concern us here since the example simulations do not use the physics codes. However a number of components do have small, helper fortran codes that implement the interface between the plasma state software and the physics codes proper. Also the “model” components discussed in this primer are implemented with small fortran codes. These codes do reside in the `/components/` directory and are built by the IPS “make”.

The “make install” creates a `/bin/` directory in `IPS_ROOT` and moves all of the Python and fortran executables that constitute the IPS to there. This includes the basic IPS Python module, “`ips.py`”.

The IPS provides a very convenient way to access the most recently generated monitor file as the simulation progresses. On each system a web accessible directory exist in the SWIM project area in which these files are collected. By establishing a browser bookmark to this directory the user can download the latest monitor file to his local computer for visualization with a single click. At NERSC this directory is `/project/projectdirs/m876/www` and at PPPL it is `/p/swim/w3_html`. The user should create a subdirectory there where his monitor files will be collected and should point to that directory in his simulation configuration files as discussed below.

So, now you are ready to run an IPS simulation.

## **Running an IPS simulation**

To run an IPS simulation it is necessary to provide a simulation configuration file, and the needed input files for the physics components. References to more detailed information on configuration files are given at the end, but these details are not needed for these demo simulations. Suffice it for now to realize that the simulation configuration file specifies such things as: simulation/run identifiers, file naming conventions, what files constitute the plasma state data, what components to use in the simulation, and configuration information for individual components such as names of required input/output files, paths to binaries, number or processors required for executable, any other configuration data needed by a given component. Simulations can be launched from the command line in an interactive session, but more usually for a serious simulation one submits a batch script.

All the configuration files, batch scripts, and full simulation results for both of the example simulations can be found in the SWIM data tree on the respective platforms. The data tree is a SWIM-public area where simulation input data can be stored. It allows multiple users to access the same data and have reasonable assurance that they are indeed using the same versions. On franklin the data tree root is:

```
/project/projectdirs/m876/data/
```

and on stix it is:

```
/p/swim1/data/
```

In adapting a configuration file from another user (the recommended way to start a configuration file) there are always a few minor customizations required. For example to adapt the config files for these simulations on franklin a user (other than batchelor) would need to modify the following parameters:

```
IPS_ROOT = /global/homes/u/u2115/ips_2_1_11
SIM_ROOT =
/project/projectdirs/m876/dbb/IPS_examples/examples_franklin/hello_world/${SIM_NAME}
USER_W3_DIR = /project/projectdirs/m876/www/dbb
USER_W3_BASEURL = http://portal.nersc.gov/project/m876/dbb
```

UNIX environment variables are not available inside the simulation configuration file therefore it is necessary to specify `IPS_ROOT` in each config file. The configuration variable `SIM_ROOT` defines the directory in which all of the simulation data is placed. It needs to be in a user writeable location. It is usually convenient to distinguish the `SIM_ROOT` by including the simulation name. In the present example this is done by variable substitution with the config variable `SIM_NAME`.

The IPS framework is the same on all computers. However in moving a simulation from one computer to another there are a number of local parameters that the framework needs to know. In order to minimize the complexity, and hence the opportunities for error in migrating simulation configuration files, this data is placed machine specific configuration files that the user need not change, or even reference. These files, such as *franklin.conf* and *stix.conf*, are found at the top level of the IPS directory, `IPS_ROOT`. However there are a few things in the simulation configuration file that are specific to both the computer and to the user, and which the user therefore needs to change for different computer platforms – specifically `USER_W3_DIR` and `USER_W3_BASEURL`. The user should change `USER_W3_DIR` to point to his own `www` subdirectory as discussed above. The framework also needs the URL to that directory, `USER_W3_BASEURL`. At franklin the base URL is:

```
http://portal.nersc.gov/project/m876/<user\_directory>
```

and on stix it is:

```
http://w3.pppl.gov/swim/<user\_directory>.
```

The user would probably also change:

```
USER = Batchelor          # Optional, if missing the unix username is used
```

There is a great deal of flexibility in where the inputs and physics executables are located, where the job is launched relative to the configuration files, and where the outputs go and what they are called. But we will not give those details here.

## Hello World

Hello\_world is (almost) the simplest possible IPS run, exercising two components DRIVER → *hello\_driver.py* and WORKER → *hello\_worker.py*. “Almost” because it would be possible to have an IPS simulation with only a driver. These components reside in */ips/trunk/components/drivers/hello/*, although there is no need to look at them to run the simulation. The files to run hello world simulation can be found on franklin at

```
/project/projectdirs/m876/data/IPS_examples/franklin_examples/hello_world/
```

and on stix at

```
/p/swim1/data/IPS_examples/stix_examples/hello_world
```

To run the hello world simulation the user should copy the files [*hello\_world.config* and *run*] to a location in his own area where he wants to run the simulation. The config file *hello\_world.config* should be customized according the to comments above. The file *run* contains the command line needed to run the simulation.

The output below shows the results of the run that was done in the data tree. When the job is launched the IPS framework prints a few lines of messages e.g.:

```
/project/projectdirs/m876/data/IPS_examples/franklin_examples/hello_world $ ./run
```

```
Starting IPS
```

```
Created <class 'hello_driver.HelloDriver'>
```

```
Created <class 'hello_worker.HelloWorker'>
```

Then the driver, followed by the worker, followed again by the driver will print:

```
HelloDriver: beginning step call
```

```
Hello from HelloWorker
```

```
HelloDriver: finished worker call
```

A few log files are produced, leaving the \$SIM\_ROOT directory with something like:

```
-rw-r-x--- 1 u2115 m876 4894 Mar 14 20:30 hello_world.config*  
-rwxr-x--- 1 u2115 m876  107 Mar 14 20:30 run*  
-rw-r--r-- 1 u2115 m876  447 Mar 15 19:50 hello_world_pbs.log  
-rw-r--r-- 1 u2115 m876 6298 Mar 15 19:50 Hello_world_sim.log  
-rw-r--r-- 1 u2115 m876  308 Mar 15 19:50 resource_usage
```

The reader can go to the SWIM portal web page, <http://swim.gat.com:8080/>, and find this run as Portal Run Id 18990. The details page for this run is:

<http://swim.gat.com:8080/detail/?id=18990>. The identical run on stix is Portal Run Id 18993 and the details page is: <http://swim.gat.com:8080/detail/?id=18993>.

## Sequential Model Simulation

This “model” simulation is intended to look almost like a real simulation, short of requiring actual physics codes and input data. Instead typical simulation-like data is generated from simple analytic (physics-less) models for most of the plasma state quantities that are followed by the MONITOR component. The “model” simulation includes time stepping, time varying scalars and profiles, and checkpoint/restart.

The PORTS/components exercised are:

Simulation initialization = INIT → `minimal_state_init.py`

Driver = DRIVER → `generic_driver.py`

Equilibrium and profile advance = EPA → `model_epa_ps_file_init.py`

Ion cyclotron heating = RF\_IC → `model_RF_IC_2_mcmd.py`

Neutral beam heating = NB → `model_NB_2_mcmd.py`

Fusion heating and reaction products = FUS → `model_FUS_2_mcmd.py`

Simulation time history monitoring = MONITOR → `monitor_comp.py`

The driver and monitor components are the full components as used in many of the real simulations. The others are simple models, but models that can be modified by changing the component input data files. The python components and fortran source files that implement the model components reside in the `/ips/trunk/components/` tree. For example `model_RF_IC` can be found in `/components/rf/model_RF_IC/`. Again one need not deal with the components to run this simulation.

As can be seen by looking at the config file the input data resides in the data tree in a subdirectory associated with each component. For example input data for `model_epa_ps_file_init.py` can be found in:

`/project/projectdirs/m876/data/model_epa/ITER/hy040510/t20.0`

The path to the input data, `/model_epa/ITER/hy040510/t20.0/`, can be parsed as follows: data for `model_EPA` component, for tokamak = `ITER`, a hybrid scenario called `hy040510` from a previous PTRANSP run, time slice = `20.0` sec. Other data in the data tree also follows this pattern. It seems to be a clear and convenient way to organize the data and make it generally accessible. We encourage users to add their own simulation input data to the data tree and to follow the same layout.

On initialization the `model_epa_ps_file_init.py` component takes as input an existing plasma state file and eqdsk file from a previous IPS run, or perhaps from a TRANSP analysis of an experimental shot. It then copies the EPA relevant data to the initial plasma state file and eqdsk file. And on subsequent time steps it applies simple operations to that data to produce time varying profiles. The description of the model operations is supplied in another input namelist file, `model_epa_input.nml`. The config file, batch scripts and results of a sample run can be found on franklin at:

`/project/projectdirs/m876/data/IPS_examples/franklin_examples/seq_model_simulation/`  
and on stix at:

`/p/swim1/data/IPS_examples/stix_examples/seq_model_simulation`

To run the sequential model simulation the user should copy the files [*seq\_model\_sim.config*, *run*, *debug*, *restart\_12\_sec.config*, *restart*] to a location in his own area where he wants to run the simulation. The files should be modified according the comments above to fit the users situation. The file *run* contains the command line needed to run the simulation. However, this simulation requires two processors (one for the framework/python components and one for the fortran executables) and can only be run from the command line in a multi-processor interactive session. The file *debug* is a batch script to run the job in the franklin debug queue. On stix the file *run* is a batch script that can be submitted from portalr5 or directly on stix using “use”. The simulation output data is in the various log files and the simulation directory, *Model\_seq\_1/*, as specified in the SIM\_ROOT variable of the config file. The contents of this simulation output are the same in structure as one would get in a real simulation. Explore it and marvel.

The results in this particular output directory came from running the simulation from 0 to 20.0 sec, then restarting from 12.0 sec and running out to 20.0 sec again. To actually restart the simulation after running it, just submit the batch script *restart*. This script launches IPS with the config file *restart\_12\_sec.config*, which identical to the original config file except for the modifications needed for restart.

The reader can go to the SWIM portal web page, <http://swim.gat.com:8080/>, and find the original 20 sec run as Portal Run Id 18974. The details page for this run is: <http://swim.gat.com:8080/detail/?id=18974>. The restart run from 12 sec is Portal Run Id 18976 and the details page is: <http://swim.gat.com:8080/detail/?id=18976>. Again the output from the simulation is contained in the SIM\_ROOT directory.

The standard out is contained in the files *model\_sim.o7069708* for the original run, and in *model\_sim.o7069792* for the restarted run. The easiest way to see the output is to download the final monitor file to your local computer and plot it using either ELVis or the PCMF.py utility that was described in an email to SWIM on 1/31/11. The monitor file can be downloaded with one click from your USER\_W3\_DIR as described above. Each IPS run will be represented in USER\_W3\_DIR by three files. The most recent monitor file is labeled <RunId#>\_monitor\_file.nc.

Figures 1 and 2 below are screen shots in which ELVis has been used to plot the model simulation monitor file. The first is using the ELVis template file *basic\_time\_traces.xml* the second used template file *thermal\_profiles.xml*. These ELVis template files, and others, are included in the monitor component directory, */components/monitor/monitor\_4/*.

Figure 1. Time traces

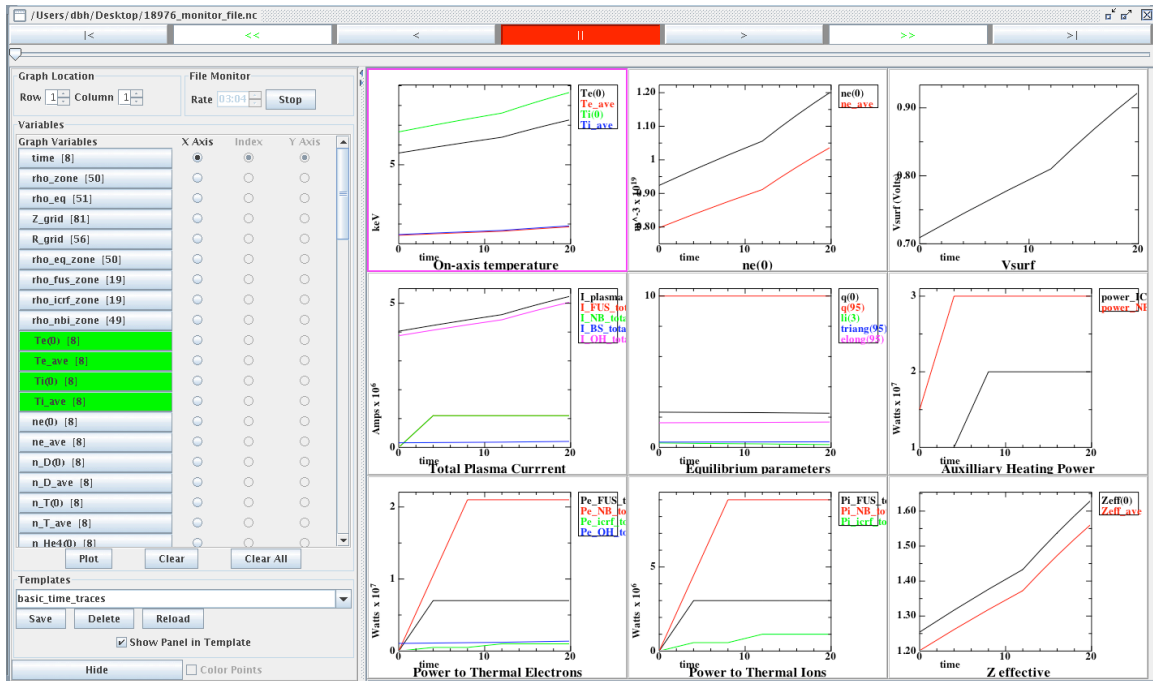
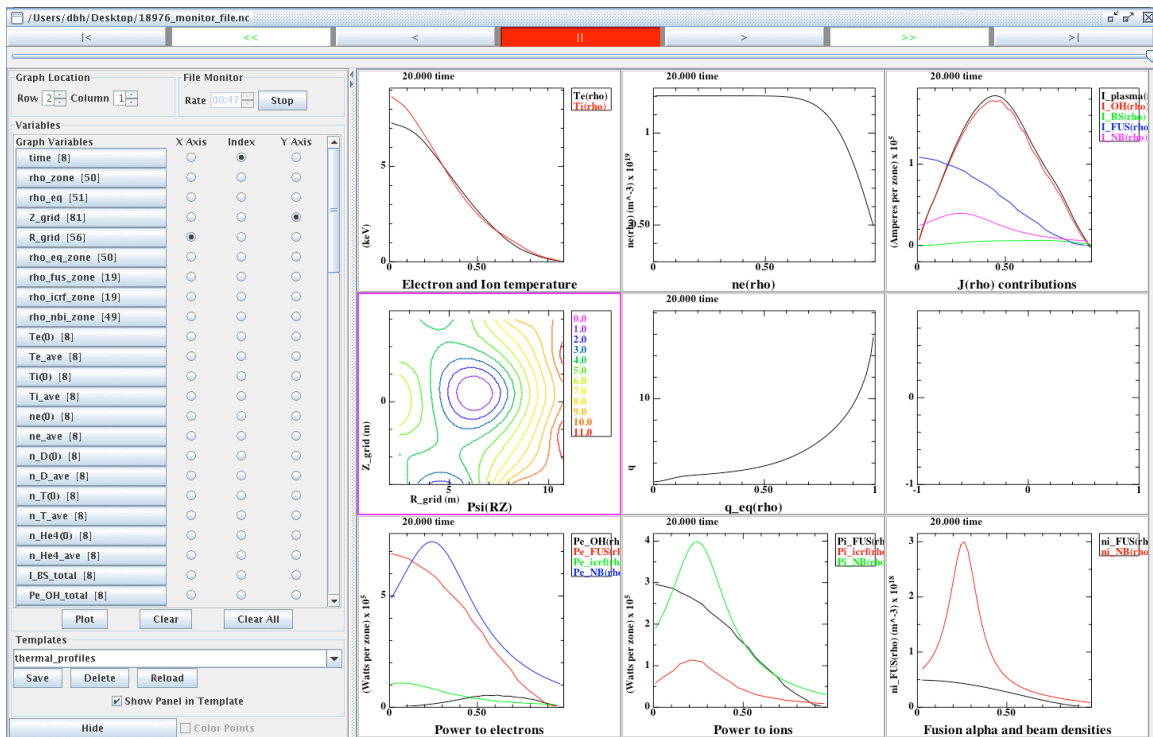


Figure 2. Thermal profiles  $t = 20$  sec



One such plot of profiles is available for each time step. ELVIs can be used to plot the data almost any way you want it, and you can save a template file that retains the layout and styles.

On stix the original 20 sec run is Portal Run Id 19000. The details page for this run is: <http://swim.gat.com:8080/detail/?id=19000>. The restart run from 12 sec is Portal Run Id 19002 and the details page is: <http://swim.gat.com:8080/detail/?id=19002>. The output from the simulation is contained in the SIM\_ROOT directory:

*/p/swim1/data/IPS\_examples/stix\_examples/seq\_model\_simulation/Model\_seq\_1.*

## **Additional information**

Additional information is available at the SWIM website <http://cswim.org>. In particular under the Guides section there is a draft “Integrated Plasma Simulator User’s Guide” by Samantha Foley *et al* which gives some information about the IPS framework itself and more information for users. Also perhaps useful are presentations by Elwasif, Foley, and Batchelor from SWIM project meetings, which are available in the Guides section and/or the presentations presented at the all hands meetings. Also a more comprehensive set of documentation is being developed in the SVN repository at:

<https://cswim.org/svn/cswim/ips/trunk/doc/>

This material is under construction but does have useable information.

Finally it should be commented that the “model” components used for these examples have uses beyond their didactic applications. A circumstance in which one needs to externally specify certain plasma state data, for example for testing purposes, could use these components. The model\_EPA component, which can extract equilibrium and profile data from previous simulations or from TRANSP analysis of experiments, has proved useful for scaling and benchmarking studies of source modeling codes such as RF or neutral beam heating.

## Appendix

### Setting up the SWIM IPS on NERSC Franklin

1	Get a NERSC user account	<a href="mailto:accounts@nersc.gov">accounts@nersc.gov</a>
2	Get added to NERSC SWIM project (m876)	batchelordb@ornl.gov
3	Register as SWIM member	<a href="http://cswim.org">http://cswim.org</a>
4	Check out IPS from svn repository	cswim.org/svn/cswim/ips/trunk → IPS_ROOT
5	Copy or link makeconfig file	IPS_ROOT/config/makeconfig.local → makeconfig.franklin
6	Add SWIM to .bashrc.ext	Export IPS_ROOT=<path> Source IPS_ROOT/swim.bashrc.franklin
7	make and make install	
8	Add user subdirectory to www	/project/projectdirs/m876/www/<user>

### Run Hello World

1	Pick location for simulation run directory	mkdir <path>/example_runs
2	Copy config file and run command to example_runs	/project/projectdirs/m876 /data/IPS_examples/franklin_examples/ hello_world/
3	Customize config file	IPS_ROOT, SIM_ROOT, USER_W3_DIR, USER_W3_BASEURL, USER
4	Run	./run
5	Look at run on swim portal	<a href="http://swim.gat.com:8080/">http://swim.gat.com:8080/</a>

### Run Model Simulation

1	Copy config file, batch scripts and run command to example_runs	/project/projectdirs/m876 /data/IPS_examples/franklin_examples/ sequential_model_simulation/
3	Customize config file	IPS_ROOT, SIM_ROOT, USER_W3_DIR, USER_W3_BASEURL, USER
4	Submit to debug queue	qsub debug
5	Look at run on swim portal	<a href="http://swim.gat.com:8080/">http://swim.gat.com:8080/</a>
6	Download monitor file	<a href="http://portal.nersc.gov/project/m876">http://portal.nersc.gov/project/m876</a> /<user>/
	Plot monitor file	ELVis or PCMF.py