User Manual

Under development by Vyacheslav S. Lukin
Last modification by V.S. Lukin on July 3rd, 2013.

A user account with a username '***myhifi***' has just been created for you on the SourceRepo repository website. Now, you'd like to check out a copy of the code. Maybe to study the source code, maybe to build your own physics application on top of the existing framework, or maybe to directly use one of the existing physics application modules.

Lets get started.

1. Change Password on SourceRepo.

Log into SourceRepo and change the password you have been given together with your '***myhifi***' username by going to 'https://hifi.sourcerepo.com/login/project_user' in your preferred Web browser.

2. Download the code to your computer

1) First, you need to download the source code from SourceRepo via SVN. If you don't already have SVN installed on your system (or your version is ridiculously outdated), you can easily download and install a new version of SVN from http://subversion.tigris.org/. It is open-source freeware.
2) Now that SVN is installed, you can familiarize yourself with basic commands by typing 'svn help' or studying the complete manual available at http://svnbook.red-bean.com/ . To help with basic SVN commands, included at the end of this document is a quick refernce SVN card.
3) Fine, you already know SVN. You are ready to download the code. It is time to decide on the name for your SVN controlled repository, where the code will be downloaded to and where you can continue to get updated framework solver versions for the duration of your project. Suppose, you have limited imagination and you decided on '***hifi_svn***', to be located in your root directory. Now, to download the code, at the prompt type:

for 2D code, aka SEL:

$ svn co --username *myhifi* http://hifi.sourcerepo.com/hifi/SEL/trunk ~/*hifi_svn*

for 3D code:

$ svn co --username *myhifi* http://hifi.sourcerepo.com/hifi/HiFi/trunk ~/*hifi_svn*

4) The system will request your password. Provide your new password.
5) You may be asked if you want to store ssh keys. It is advised to do so; otherwise, you maybe asked your password several times whenever you want to update your SVN code repository.
6) A new directory '~/hifi_svn' should now have been created and the main source code directories, together with input decks, a README file, and a separate post-processing code

directory, should have been added to the repository inside '~/hifi_svn'.

3. What is it you have just downloaded?

1) In your new 'hifi_svn' directory, you will find a *.txt copy of the User Agreement you completed in order to be granted access to the code.
2) You will find one or more directories named \code3D_*.* (or \code_*.* for the SEL repository), where the numbers stand for the main code version. These are the directories where the existing HiFi (SEL) physics modules and input parameter files are located. It is strongly advised to use the latest version available.
3) You will find one or more directories named \solver_*.*, where the numbers stand for the PETSc version that the solver files in this directory are written for. The \solver_*.* directory contains all of the core HiFi solver modules, all of which are physics-blind, and constitute the HiFi framework itself.
4) You will find a directory 'post3D' (or 'post' for the SEL repository) that contains the separate post-processing code used to convert HiFi output into data files readable by the VisIt visualization software (see below for more details).
5) You will also find a directory called 'draw', that contains input files for 1D & 2D basic visualization package XDRAW. Due to extensive capabilities of the freely available VisIt package, XDRAW for all but most basic applications is now obsolete. (It is much faster, yet much more limited in its capabilities.)

4. Do you want to see the actual code?

As mentioned above, the HiFi (SEL) framework solver modules are all in the \solver_*.* directory, while the physics modules and everything that pertains to any specific HiFi application is in \code3D_*.* (\code_*.*). At present, the latest available solver version is \solver_3.2. Upon entering the \code3D_*.* (\code_*.*) directory, you see the following:

README – a file with a brief description of the code structure, input decks, and some basic instructions on how to compile and run the HiFi code. You should study it if you are planning to use the HiFi (SEL) code framework;

hifi.in (sel.in) & beltrami.in – input deck files, see README for more details;

makefile_* – makefiles for several HPC machines the code has been compiled on;

go_* – batch submission scripts for several HPC machines the code has been run on;

physics_templ.f – the template for constructing your own physics application module;

all other *.f files – existing physics application modules;

In a \solver_*.* directory, you see the following:

*.[fF] – Fortran files comprising the framework solver library;
makefile_* – makefiles for the HPC machines the library has been compiled on;

5. How to install the necessary libraries?

    1) General instructions on how to install the libraries necessary to compile the framework code (that may or may not be useful on your particular machine) can be found at: http://www.psicenter.org/wiki/index.php/HiFi/SEL_Libraries. These are: MPI, PETSc, HDF5, and netCDF. (It should be sufficient to compile parallel HDF5 without compiling the serial version. Also, unless given specific instructions to the contrary, you can ignore the SLEPc library altogether.) For MPI, HDF5 and netCDF you should ignore the specific version numbers of the libraries given in the instructions and go with the latest stable versions available. (The specific instructions provided for compiling these libraries may also be outdated.) For PETSc, you should compile the latest version for which the corresponding \solver_*.* directory is available.

Of course, as is often the case with multiple inter-dependent libraries, it will likely take some effort to properly compile and link HiFi with all of the libraries on your particular machine. Therefore, if you have access to one of the externally supported HPC machines, we strongly recommend installing and running HiFi on one of these. The libraries listed above are now in common use and have likely already been installed there. In particular, HiFi has already been compiled, tested and is running in production mode on multiple machines at US DOE, US DOD, NSF HPC and NASA computing centers.

    2) You will also want to install the VisIt visualization package freely available at https://wci.llnl.gov/codes/visit/ . They have pre-compiled binaries available for most platforms.

    3) For modeling in complex geometries, you may want to get access to the Cubit grid generation package (http://cubit.sandia.gov/). Unfortunately, unlike everything else used by the HiFi (SEL) framework, Cubit Tool Suite is not freely available, but instead requires a registration fee (at present, it is $300 for 5 years) and it may take you 2-3 months to get access to the package from its developers, the Sandia Nat'l Laboratory.

6. To be continued...

Subversion Quick Reference Card

$Rev: 28 $

Subversion is a version control system that is a replacement for CVS. It has most of CVS's features. Generally, Subversion's interface to a particular feature is similar to CVS's, except where there's a compelling reason to do otherwise.

Quick start

```
svnadmin create /var/svnroot
svn import LocalDir file:///var/svnroot/ProjectName
svn checkout file:///var/svnroot/ProjectName
cd ProjectName
svn help [command]
```

Subversion URLs

`file://`	Direct repository access (on local disk).
`http://`	Access via WebDAV protocol to Subversion-aware Apache server.
`https://`	Same as `http://`, but with SSL encryption.
`svn://`	Access via custom protocol to an `svnserve` server.
`svn+ssh://`	Same as `svn://`, but through an SSH tunnel.

svn subcommands

`add`	Adds files and directories.
`blame`	(`praise`, `annotate`, `ann`) Shows author and revision information in-line for the specified files or URLs.
`cat`	Outputs the contents of the specified files or URLs.
`checkout`	(`co`) Checks out a working copy from a repository.
`cleanup`	Recursively clean up the working copy.
`commit`	(`ci`) Send changes from your working copy to the repository.
`copy`	(`cp`) Copy a file or directory in a working copy or in the repository.
`delete`	(`del`, `remove`, `rm`) Delete an item from a working copy or the repository.
`diff`	(`di`) Display the differences between two paths.
`export`	Exports a clean directory tree.
`help`	Prints help text.
`import`	Recursively commit a copy of local dir into a repository.
`info`	Print information about PATHs.
`list`	(`ls`) List directory entries in the repository.
`log`	Displays commit log messages.

`merge`	Apply the differences between two sources to a working copy path.
`mkdir`	Create a new directory under version control.
`move`	(`mv`, `rename`, `ren`) Move a file or directory.
`propdel`	(`pdel`, `pd`) Remove a property from an item.
`propedit`	(`pedit`, `pe`) Edit the property of one or more items under version control.
`propget`	(`pget`, `pg`) Prints the value of a property.
`proplist`	(`plist`, `pl`) Lists all properties.
`propset`	(`pset`, `ps`) Sets a property on files, directories, or revisions.
`resolved`	Remove conflicted state on working copy files or directories.
`revert`	Undo all local edits.
`status`	(`stat`, `st`) Print the status of working copy files and directories.
`switch`	(`sw`) Update working copy to a different URL.
`update`	(`up`) Updates your working copy.

svn switches

`--config-dir` *DIR*
: Read configuration from *DIR* instead of `~/.subversion`.

`--diff-cmd` *CMD*
: Use external program *CMD* for generating `diff` output instead of internal diff engine.

`--diff3-cmd` *CMD*
: Use external program *CMD* for merging files.

`--dry-run`
: Run the command without changing anything.

`--editor-cmd` *CMD*
: Use external program *CMD* for editing files.

`--encoding` *ENC*
: Instructs Subversion to use encoding *ENC* to store log messages.

`--extensions` *ARG* `(-x)`
: Additional arguments for external diff, eg.: `svn --diff-cmd diff -x --normal diff main.c`.

`--file` *FILE* `(-F)`
: Use the contents of file *FILE* as an argument for a given subcommand.

`--force`
: Forces a particular subcommand to run.

`--force-log`
: Forces a suspicious parameter passed to the `--message` options to be accepted as valid.

`--help` (`-h` or `-?`)
: Prints help for a given command or general help text.

`--ignore-ancestry`
: Ignore ancestry when calculating differences (rely on path contents alone).

`--incremental`
: Prints output in a format suitable for concatenation.

`--message (-m)` *MSG*
: Specify a commit message *MSG*.

`--no-auth-cache`
: Prevents caching of authentication information.

`--no-auto-props`
: Disable `auto-props`, overriding the `enable-auto-props` directive in the config file.

`--no-diff-deleted`
: Prevents Subversion from printing differences for deleted files.

`--no-ignore`
: Shows files in the status listing that would normally be omitted since they match a pattern in the `svn:ignore` property.

`--non-interactive`
: Prevents prompting for authentication information.

`--non-recursive (-N)`
: Stops a subcommand from recursing into subdirectories.

`--notice-ancestry`
: Pay attention to ancestry when calculating differences.

`--old` *ARG*
: Uses *ARG* as the older target.

`--password` *PASS*
: Password for authentication.

`--quiet (-q)`
: Print only essential information while performing an operation.

`--recursive (-R)`
: Makes a subcommand recurse into subdirectories.

`--relocate` *FROM TO* [*PATH...*]
: Used with the `svn switch` subcommand, changes the location of the repository that your working copy references.

`--revision` *REV* `(-r)`
: Supply a revision *REV* (or range of revisions) for a particular operation.

`--revprop`
: Operates on a revision property instead of a Subversion property specific to a file or directory (requires `--revision` switch).

`--show-updates (-u)`
: Causes the client to display information about which files in your working copy are out-of-date.

`--stop-on-copy`
: Stop harvesting historical information when a copy is encountered.

`--strict`
: Causes Subversion to use strict semantics.

`--targets` *FILE*
: Get the list of files that you wish to operate on from the file *FILE*.

`--username` *NAME*
: Username for authentication.

`--verbose (-v)`
: Verbose mode.

`--version`
: Prints the client version info.

`--xml`
: Prints output in XML format.

svnadmin subcommands

`list-unused-dblogs`	Ask Berkeley DB which log files can be safely deleted.
`create`	Create a new, empty repository.
`dump`	Dump the contents of filesystem to stdout.
`help`	Help.
`hotcopy`	Makes a hot copy of a repository.
`load`	Read a dumpfile-formatted stream from stdin.
`lstxns`	Print the names of all uncommitted transactions.
`recover`	Recovers any lost state in a repository.
`rmtxns`	Deletes transactions from a repository.
`setlog`	Set the log-message on a revision.

svnadmin switches

`--bypass-hooks`
: Bypass the repository hook system.

`--copies`
: Follow copy history when examining a path.

`--in-repos-template` *ARG*
: Specify a template for the repository structure when creating a new repository.

`--incremental`
: Dump a revision only as a diff against the previous revision, instead of the usual fulltext.

`--on-disk-template` *ARG*
: Specify a template to use for the on-disk structure of the repository you want to create.

`--revision` *ARG* `(-r)`
: Specify a particular revision to operate on.

svnlook subcommands

`author`	Prints the author.
`cat`	Print the contents of a file.
`changed`	Print the paths that were changed.
`date`	Print the date stamp.
`diff`	Prints differences of changed files and properties.
`dirs-changed`	Print the directories that were themselves changed.
`help`	Help.
`history`	Print information about the history of a path in the repository.
`info`	Print the author, date stamp, log message size, and log message.
`log`	Print the log message.
`proplist`	Print the names and values of versioned file and directory properties.
`tree`	Print the tree.
`youngest`	Print the youngest revision number.

svnlook switches

`--no-diff-deleted`
: Prevents svnlook from printing differences for deleted files.

`--revision` *REV* `(-r)`
: Specify a particular revision number that you wish to examine.

`--transaction` *TID* `(-t)`
: Specify a particular transaction id *TID* that you wish to examine.

`--show-ids`
: Show the filesystem node revision IDs for each path in the filesystem tree.

File status

`U`	File was updated.
`A`	File was added.
`D`	File was deleted.
`R`	File was replaced.
`G`	File was merged.
`C`	Conflicting changes.
`?`	Resource is not under version control.
`!`	Resource is missing or incomplete (removed by another tool than Subversion).

Special properties

`svn:executable`	Executable file permission.
`svn:mime-type`	MIME type of a file.
`svn:ignore`	List of file patterns which certain Subversion operations will ignore. Full list may be obtained by `svn status --no-ignore`.
`svn:eol-style`	Possible values are: `native`, `CRLF`, `LF`, `CR`.
`svn:externals`	Instructions for Subversion to populate a versioned directory with one or more other checked out Subversion working copies.
`svn:keywords`	List of keywords that will be substituted during commit:

`$Date$`	date of the last modification
`$Rev$`	revision number
`$Author$`	the last user who changed the file
`$URL$`	full URL to the latest version of the file in the repository
`$Id$`	compressed combination of keywords above

Client configuration

File `~/.subversion/config`:

```
[helpers]
editor-cmd = vim

[miscellany]
log-encoding = iso-8859-2
enable-auto-props = yes

[auto-props]
*.sh = svn:executable
*.bat = svn:eol-style=CRLF
*.c = svn:eol-style=native;svn:keywords=Id
```

Autocompletion in `bash`:

```
shopt -s extglob progcomp
.  /usr/share/subversion/.../bash_completion
```

Other sources of information

`http://subversion.tigris.org`
: Home page of the Subversion project.

`http://svnbook.red-bean.com`
: *Version Control with Subversion* — a book on Subversion.

`http://tortoisesvn.tigris.org/`
: TortoiseSVN is a Windows client for Subversion implemented as a windows shell extension.

Compiled by Cezary Sobaniec
http://www.cs.put.poznan.pl/csobaniec/