**SIVIC Software Development Tutorial**

HMTRC Workshop - March 23-24, 2017

Department of Radiology and Biomedical Imaging, UCSF

Supported by NIBIB P41EB013598

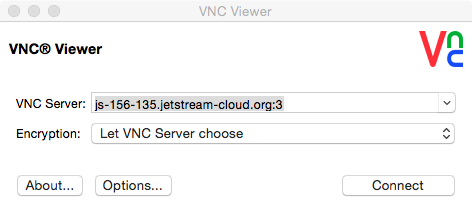
#### NSF Jetstream Development Environment Supported by XSEDE Educational Grant TG-SEE170001

**Goal:** The purpose of this tutorial is to introduce you the SIVIC C++ software framework (svk = SIVIC Kit). The tutorial will cover building the SIVIC package from source and extending the svk by developing a new kinetic model and using the model to fit an example data set.

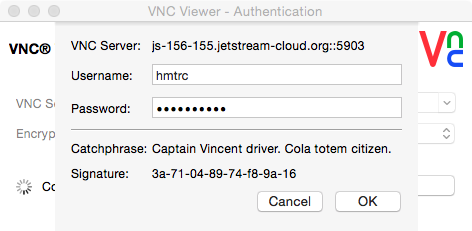
**Introduction:** The SIVIC C++ framework (svk) is an extensible software framework for developing MRS algorithms and applications. In this example you will use the framework to create a new kinetic model that can be run from the svk\_met\_kinetics command line tool in order to fit a simulated dynamic HP 13C data set.

**Setup:** The build environment for this tutorial has been preconfigured on XSEDE Linux VMs for this workshop so there is no need to set up your software environment. Simply use a VNC client on your laptop to log into the remote VM. Login information will be provided at the workshop.

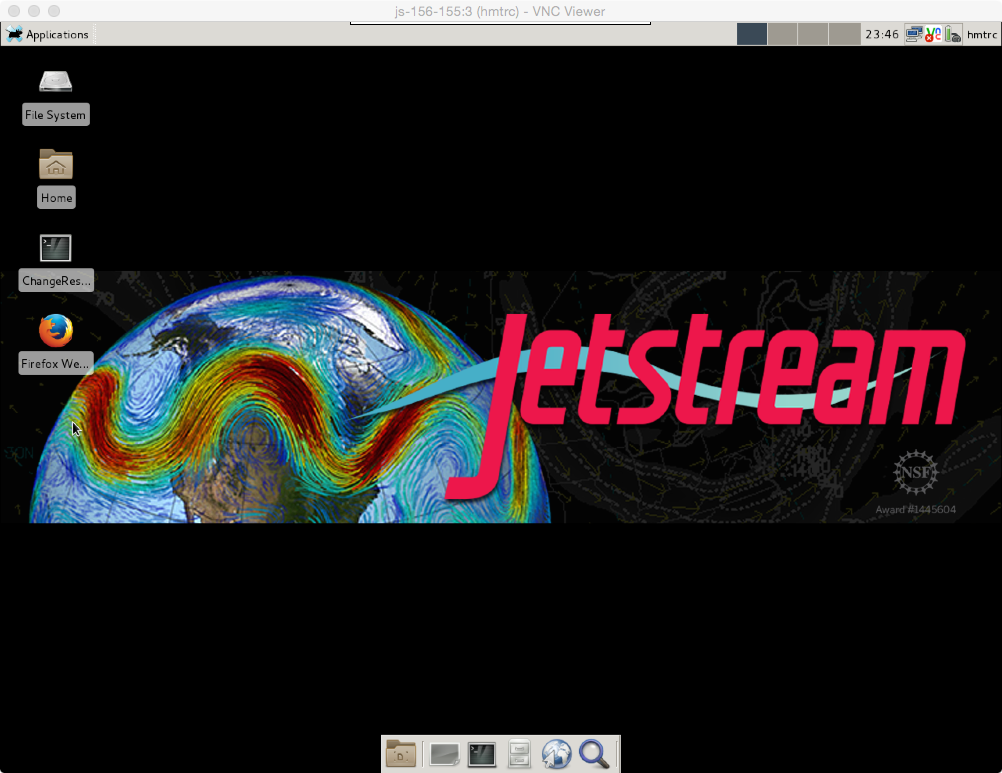
* Download and install VNC Viewer: <https://www.realvnc.com/download/viewer/>
* Open VNC Viewer and enter the URL of your Jetstream Linux VM in the “VNC Server” box., then click “Connect”. The URL and login info will be provided at the worksop.



* Enter the provided username and password:



* This will log you into the Linux desktop on the Jetstream VM:



* Click on the terminal icon at the bottom of the desktop to open a console window (arrow above).
* cd to the “workshop” directory in your home directory. There is a README with commands used in this tutorial.

cd workshop

cat README

* If you want to change the screen resolution you can do so with the xrandr command (see item 1 in the README file).

**Download and build the SIVIC package form source:** Get the sivic project from GITHUB and build the package.

* From the ~/workshop directory clone the “xsede” branch of the SIVIC GIT source code repository:

cd ~/workshop

git clone -b xsede https://github.com/SIVICLab/sivic.git

* Build the software in the “build” directory (this will take about 1 minute):

cd build/

make -f ../sivic/Makefile.ctest configure\_dist

make -j 10

* *Note, the binary applications are created under the build/applications directory, e.g. build/applications/sivic\_app/Linux\_x86\_65/sivic), however your $PATH has been set to include this so you can just type the command names*.
* Run the svk\_met\_kinetics application to confirm that the build worked. You should see the help message from the application:

svk\_met\_kinetics

Version 0.9.55

svk\_met\_kinetics   --i1 name --i2 name [ --i3 name ]

                 [ --mask name ] -o root [ -t output\_data\_type ]

                 [ --param num --lb value --ub value]

                 [ --model type ] [ -h ]

   --i1               name   Name of dynamic pyr signal file

   --i2               name   Name of dynamic lac signal file

   --i3               name   Name of dynamic urea signal file

   --mask             name   Name of mask file

   -o                 root   Root Name of outputfile.  Will write:

                                        root\_pyr\_fit.dcm

                                        root\_lac\_fit.dcm

                                        root\_urea\_fit.dcm

   --model            type   Model to fit data to:

                                 1 = 2 Site Exchange(default)

                                 2 = 2 Site Exchange Perf

                                 3 = 2 Site IM

                                 4 = 2 Site IM\_PYR

   -t                 type   Output data type:

                                 3 = UCSF IDF

                                 5 = DICOM\_MRI

                                 6 = DICOM\_ENHANCED\_MRI (default)

   --tr               tr    TR,  time in seconds between kinetic samples

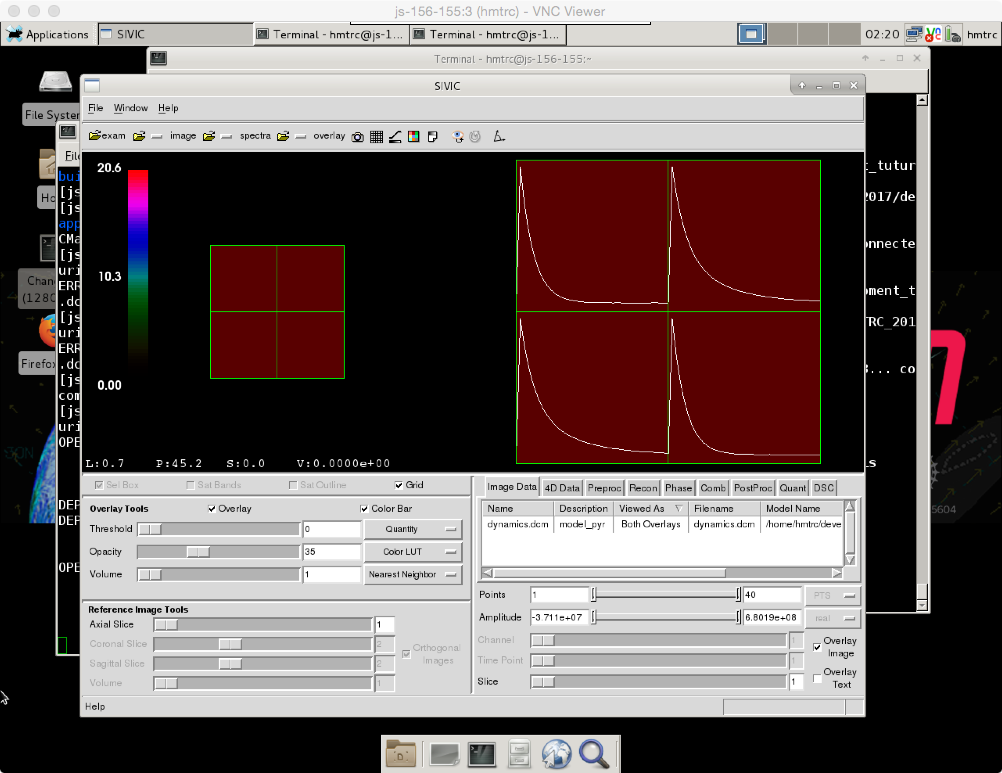
**Look at Simulated Dynamic Data:** The simulated data represents a single exponentially or bi-exponentially decaying signal in 4 voxels. You will create a new model representing bi-exponential decay and fit the decaying edge of the signals to the model:

The output will be a set of 3D parameter maps representing the modeled signal and fitted parameter maps.

* Open the simulated sample dynamic data in the SIVIC GUI.

cd ~/workshop/data

sivic --id input/dynamics.dcm



Bi-exp decay

Single exp decay

**Create a new model:** Kinetic models are a subclass of svkKineticModelCostFunction. New models are defining a new model class and defining 6 virtual methods. Copy another model and modify:

virtual InitOutputDescriptionVector()

virtual InitNumberOfSignals()

virtual GetNumberOfParameters()

virtual InitParamBounds()

virtual InitParamScaleFactors()

virtual GetKineticModel()

* cd to the source code directory and make a copy of another kinetic model class and name it svkBiExponentialCostFunction.h:

cd ~/workshop/sivic/libs/src

cp svk2SiteIMCostFunction.h svkBiexponentialCostFunction.h

* Open the class in an editor (vi, etc.) and modify.

vi svkBiexponentialCostFunction.h

* Modify the new class to define your new biexponential model. Edit the following virtual methods as described. Alternatively, a copy of the new model from has been provided for you (~/workshop/data/src/svkBiexponential.h) to copy :
  + InitOutputDescriptionVector(): Set string descriptions of inputs/output
  + InitNumberOfSignals(): 1 input signal
  + GetNumberOfParameters(): 4 (tmax, k1, k2, fraction)
  + InitParamBounds(): find tmax and fix it, set ranges
  + InitParamScaleFactors(): scale by Tr to point space
  + GetKineticModel(): write the model function.

. . .

this->GetModelSignal(PYR)[t] = this->GetSignalAtTime(PYR, Tmax) \* (

fraction \* exp( -1 \* k1 \* (t-Tmax))

+ (1. - fraction) \* exp( -1 \* k2 \* (t-Tmax))

) ;

* Register the new model in the numerical fitting class (svkMRSKinetics.h). Add an entry in the ModelType enum ( BI\_EXPONENTIAL = 5):

vi svkMRSKinetics.h

- LAST\_MODEL = TWO\_SITE\_IM\_PYR

+ BI\_EXPONENTIAL,

+ LAST\_MODEL = BI\_EXPONENTIAL

* Add a case in svkMRSKinetics so that this model can get instantiated:

vi svkMRSKinetics.cc

#include <svk2SiteIMPyrCostFunction.h>

+#include <svkBiexponentialCostFunction.h>

} else if ( this->modelType == svkMRSKinetics::TWO\_SITE\_IM\_PYR) {

costFunction = svk2SiteIMPyrCostFunction::New();

+ } else if ( this->modelType == svkMRSKinetics::BI\_EXPONENTIAL) {

+ costFunction = svkBiexponentialCostFunction::New();

}

* Add an option for calling the new model in the svk\_met\_kinetics class:

cd ../../applications/cmd\_line/src

vi svk\_met\_kinetics.cc

usemsg += " 4 = 2 Site IM\_PYR \n";

+ usemsg += " 5 = Biexponential decay \n";

usemsg += " -t type Output data type: \n";

modelType = svkMRSKinetics::TWO\_SITE\_IM\_PYR;

+ } else if ( modelTypeInt == 5 ) {

+ modelType = svkMRSKinetics::BI\_EXPONENTIAL;

}

**Your Done!**

**Now you can rebuild the svk\_met\_kinetics application and try it.**

**Rebuild the numerical fitting application (svk\_met\_kinetics)with the new model and test it:**

* This time just rebuild the kinetic modeling application:

cd ~/workshop/build

make -j 10 svk\_met\_kinetics

* Run it and verify the new usage message:

svk\_met\_kinetics

* the output should show the new –model option 5:

--model type Model to fit data to:

1 = 2 Site Exchange(default)

2 = 2 Site Exchange Perf

3 = 2 Site IM

4 = 2 Site IM\_PYR

5 = Biexponential fit

* Run it with the sample data:

cd ~/workshop/data

svk\_met\_kinetics --i1 input/dynamics.dcm --model 5 -t 6 --tr 3 -o model5 --mask ./input/fitting\_mask.dcm

* Visualize the results. The display below is the fraction map (model5\_fraction.dcm) showing single exponential voxels (f ~1, red) and double exponential voxels (f ~.3, green):

sivic --id input/dynamics.dcm --id model5\_pyr\_fit.dcm --id model5\_pyr\_residual.dcm -i ./input/fitting\_mask.dcm -i model5\_Tmax.dcm –I model5\_k1.dcm -i model5\_k2.dcm -i model5\_fraction.dcm -i model5\_rss.dcm

Color Overlay WL icon. Adjust color

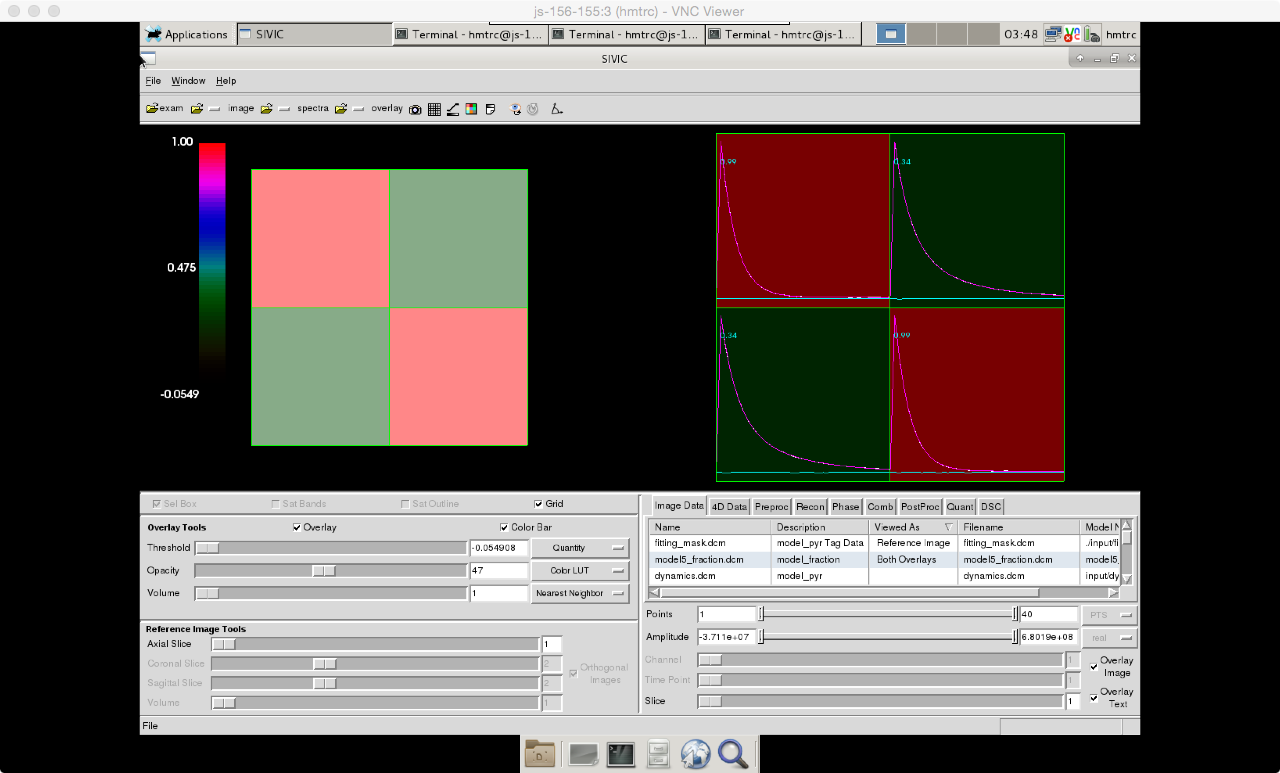
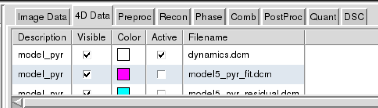


Image Data Tab: Select maps to display as overlays (right click, set as overlay)



4D Data Tab: Select dynamic traces to display on right.