

Procedure to set up saving data products

The [DASDataProducts](#) github repository contains the python modules for saving files of data products calculated from the stream of live data that is concurrently being saved. To start the set up for running the script to save data product files, first an individual with access to the repository should pull the latest version of the repo to the system files. In the repository, first check the file 'param.py' in the SourceCode directory to make sure the parameters in the file are set to the user's desired values. If the virtual treble package has not been installed on the remote system, open the terminal and use the wget and pip install commands to download from the Terra15 site:

Download wheel (This example uses python 3.8):

For python 3.6 use "cp36" before _v3.whl

For python 3.7 use "cp37" before _v3.whl

For python 3.8 use "cp38" before _v3.whl

```
wget terra15.com.au/download/latestlinuxapi_cp38_v3.whl --content-disposition
```

```
pip install acq_server-3.16.3-cp38-cp38-linux_x86_64.whl
```

Or for version 4 (newer), this example uses python 3.9:

```
wget --content-disposition https://terra15.com.au/download/latestlinuxapi_cp39_v4.whl
```

```
conda create -n terra15py39 python=3.9
```

```
conda activate terra15py39
```

```
pip install treble-4.10.10-cp39-cp39-linux_x86_64.whl
```

After the Terra15 Treble server package has been installed, the DASDataProducts modules can use the Treble API to make calls to the virtual server. The example parameter file (param.py) located in the SourceCode folder of the repository should be checked and edited to make sure the correct data product parameters are chosen. Next, the steps on the next page should be followed to set up the cron file that will repeatedly call the script to calculate and save the data products to files. The files should then start saving to the file location specified in the parameter file.

Using Cronjob

Cron is a background program for scheduling job submissions at specific times. Since data needs to be continuously fetched from the data stream every so often to calculate data products, cron jobs can be scheduled to run at intervals like every few minutes. This doc goes over the steps to start and check if cron is running and set up cron jobs to save data product files. (Note: any changes or edits to the parameter file should be done before scheduling jobs, and make sure the path to save the data product files to is correct in the parameter file)

Checking Python Script

Open the terminal for the Linux system by clicking on the windows start button and searching for “terminal”.

Click on the app.

Navigate into the “DASDataProducts” folder by typing

cd

then a space, followed by the path to the folder. The path to the folder is the same location as the folder that would be followed using the File Explorer.

For example, the location in this example system is in the user’s (“saman”) Documents, so the command is

cd Documents/DASDataProducts

The location name on the left of the blinking cursor should now display the location of the folder and its name.

```
/mnt/c/Users/saman$ cd Documents/DASDataProducts/  
/mnt/c/Users/saman/Documents/DASDataProducts$ |
```

Navigate into the SourceCode folder by typing

cd SourceCode

The files in the directory can be listed with the command

ls -l

Make sure the file save_data_prod.py is executable by typing the command

chmod +x save_data_prod.py

To make sure the `save_data_prod.py` script can be called by cron, the line endings must be changed in the file from Windows format to Unix format. First the package `dos2unix` must be installed by typing

`sudo apt install dos2unix`

Then, type the administrator password and hit enter.

```
[sudo] password for spaulus:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  dos2unix
0 upgraded, 1 newly installed, 0 to remove and 29 not upgraded.
Need to get 374 kB of archives.
After this operation, 1342 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu focal/universe amd64 dos2unix amd64 7.4.0-2 [374 kB]
Fetched 374 kB in 1s (480 kB/s)
debconf: unable to initialize frontend: Dialog
debconf: (Dialog frontend will not work on a dumb terminal, an emacs shell buffer, or without
a controlling terminal.)
debconf: falling back to frontend: Readline
Selecting previously unselected package dos2unix.
(Reading database ... 38345 files and directories currently installed.)
Preparing to unpack .../dos2unix_7.4.0-2_amd64.deb ...
Unpacking dos2unix (7.4.0-2) ...
Setting up dos2unix (7.4.0-2) ...
```

After the package is successfully installed, type the command

`dos2unix save_data_prod.py`

The terminal should state the file is being converted with the following message:

```
dos2unix: converting file save_data_prod.py to Unix format...
```

Finding file paths

To determine the full paths to the files needed for the cron job command, type

`realpath save_data_prod.py`

Then type “`realpath` “ and the name of the parameter file that will be used, in this case named `param.py`. Copy those paths for adding into the crontab file next.

```
spaulus@DESKTOP-OPQB687:/mnt/c/Users/saman/Documents/DASDataProducts/SourceCode$ realpath save_data_prod.py
/mnt/c/Users/saman/Documents/DASDataProducts/SourceCode/save_data_prod.py
spaulus@DESKTOP-OPQB687:/mnt/c/Users/saman/Documents/DASDataProducts/SourceCode$ realpath param.py
/mnt/c/Users/saman/Documents/DASDataProducts/SourceCode/param.py
```

Creating and editing crontab file

To create the crontab file that will contain the time to submit the jobs and what commands to run, type the command

crontab -e

If a crontab has not already been created, the user must select an editor. Select “/bin/nano” by entering the corresponding number.

```
spaulus@DESKTOP-0PQB687:/mnt/c/Users/saman$ crontab -e
no crontab for spaulus - using an empty one

Select an editor. To change later, run 'select-editor'.
 1. /bin/nano      <---- easiest
 2. /usr/bin/vim.basic
 3. /usr/bin/vim.tiny
 4. /bin/ed
Choose 1-4 [1]: 1
```

The crontab file will then open. The crontab file heading explains the setup of the cron command.

```
GNU nano 4.8 /tmp/crontab.d0t0u8/crontab
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow  command
```

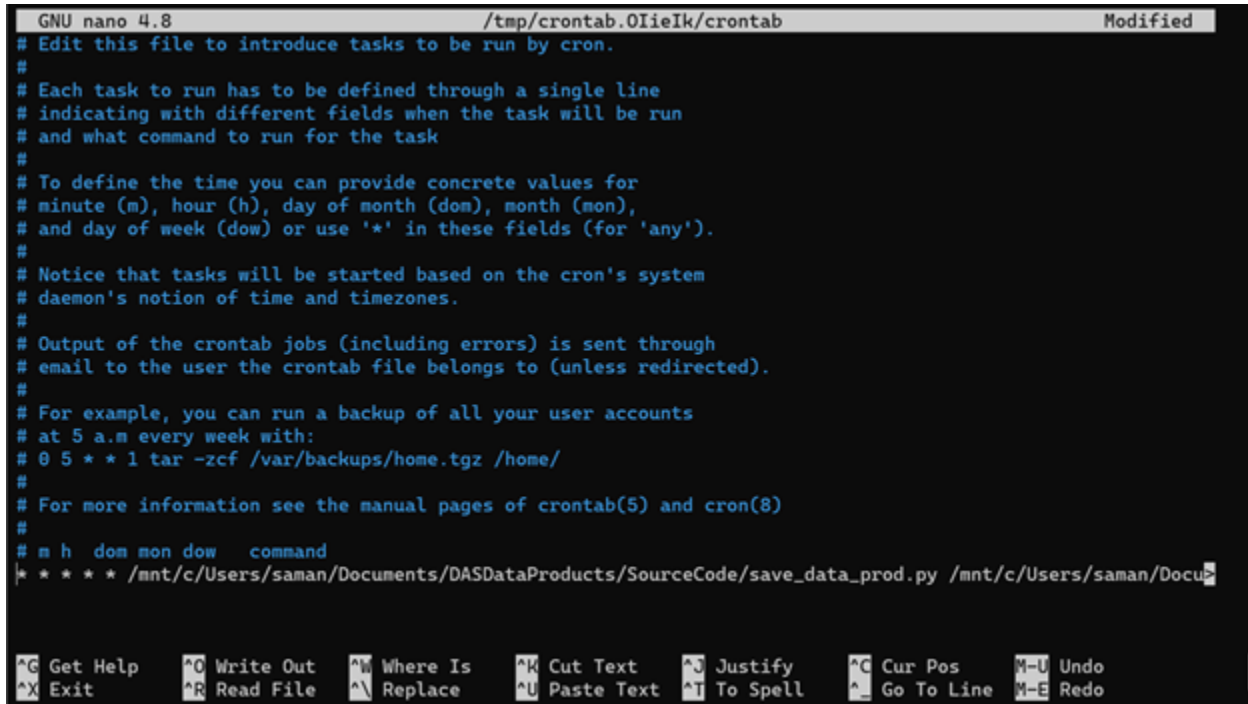
To type the command that will call the python script every minute, move the cursor to a line under the last line of the header, then type

```
* * * * *
```

then space and the path to the save_data_prod.py file, then space and the path to the python parameter file to use. For example, the command for this example would be

* * * * *

/mnt/c/Users/saman/Documents/DASDataProducts/SourceCode/save_data_prod.py /mnt/c/Users/saman/Documents/DASDataProducts/SourceCode/params.py



Exit the crontab file by pressing ctrl and x at the same time, and type y to save the new file. Then hit enter to accept the file name. The terminal should display “crontab: installing new crontab”. The contents of the file can also be viewed from the terminal by typing the command

crontab -l

If the crontab file needed to be edited, the same file can be accessed by again typing

crontab -e

If the time interval needs to be changed, for example have the script run every five minutes as opposed to every minute, the “* * * * \” should be changed to “*/5 * * * *”.

Starting cron

To check the status of cron, type the command

service crond status

for Linux systems or

service cron status

for Debian systems (Ubuntu). To start cron if it is not running, type

service crond start

or

service cron start

or

/etc/init.d/cron start

If the user does not have permissions (a message may display stating “cron: can't open or create /var/run/crond.pid: Permission denied” or “[fail]”), use the command

sudo service crond start

or

sudo service cron start

or

sudo /etc/init.d/cron start

then type the admin password and enter. Now when checking status, the command to check status should display that cron is running.

Stopping cronjob

To stop the cron program, type

service crond stop

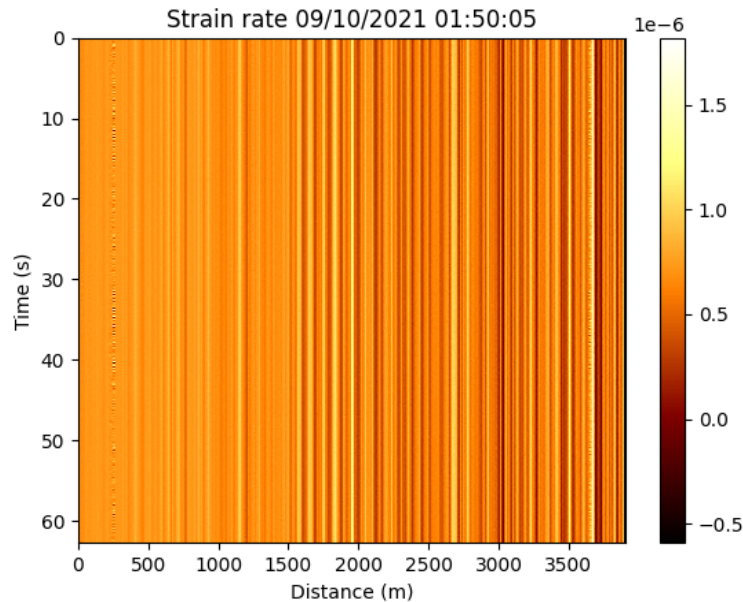
or

service cron stop

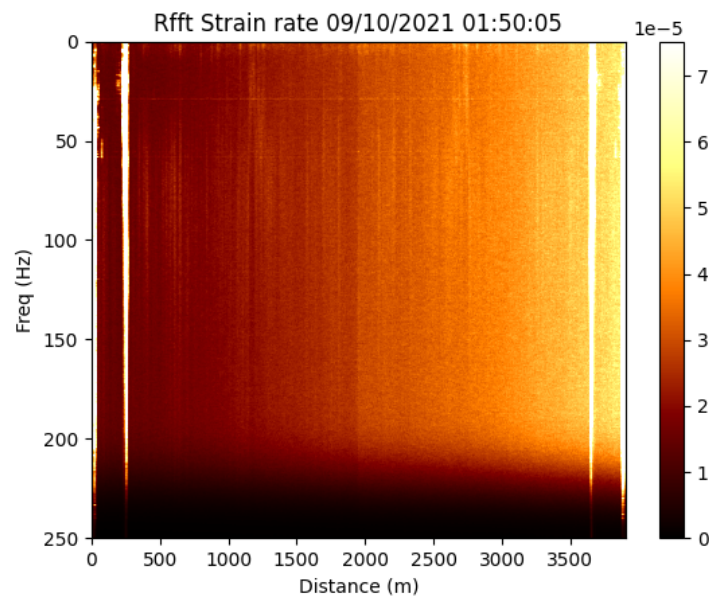
If permissions are denied, add “sudo “ to the beginning of the command and enter the admin password. The specific command to run the save_data_prod.py script can also be commented out in the crontab file by editing it and adding “#” to the beginning of the line.

The saved files of data products are about 5% the memory size of the original dataset's size.

Using a few of the early data files that have been recorded in Alaska, some visualizations of the data products have been calculated to see what may be expected when data products are saved from the stream of data on the server in Alaska.

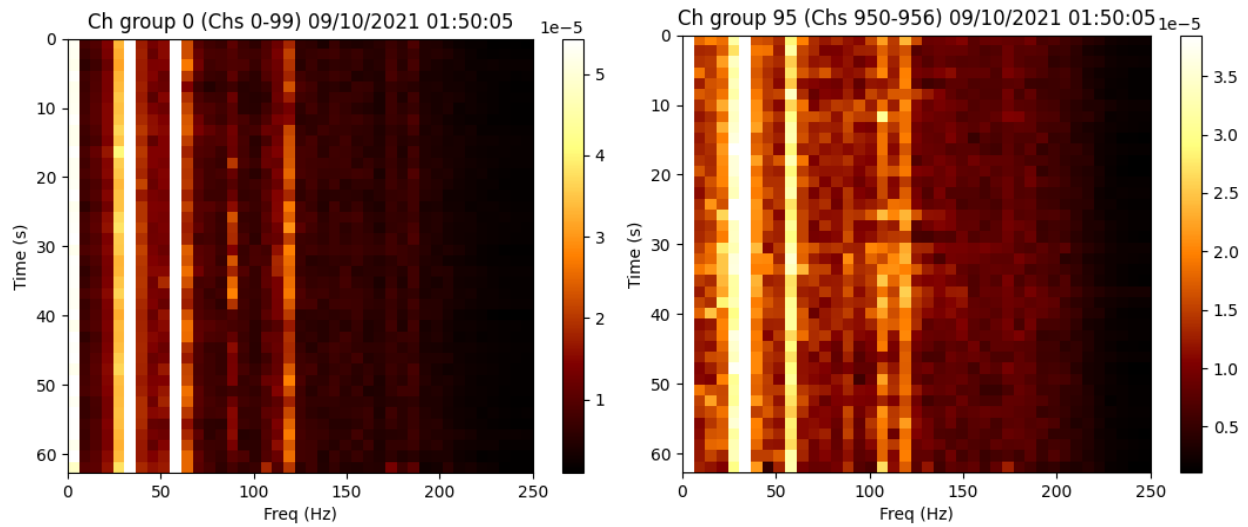


This is the 95th percentile plot of the original strain rate data contained in the file at 9/10/21 1:50:05 UTC (5:50 pm AKDT).

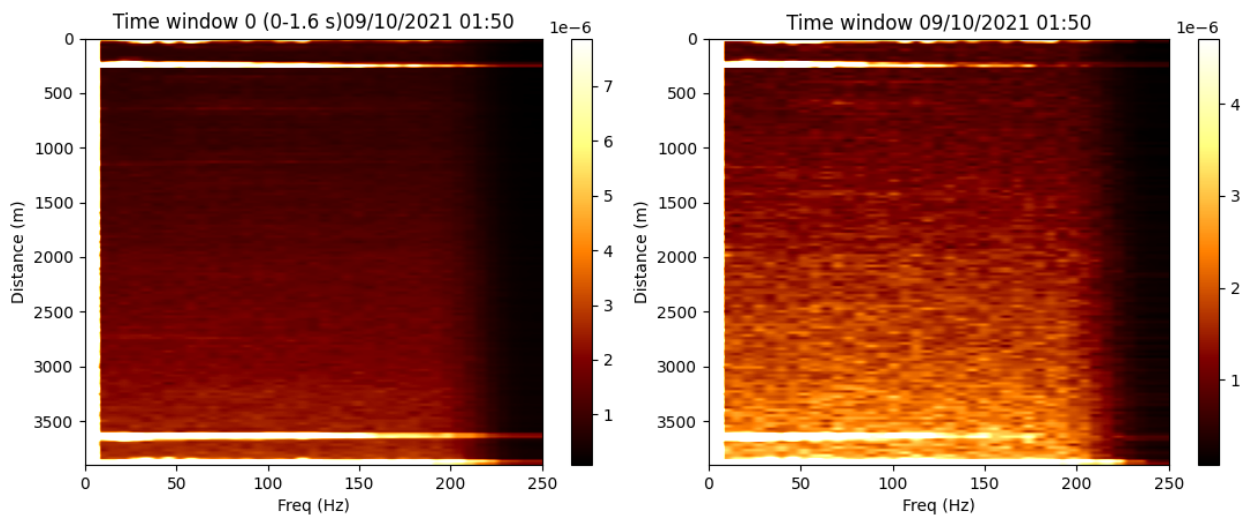


The real discrete Fourier transform of the original data. The script in the repository calculates a spectral tensor by slicing smaller subsections of the original data and performing a Fourier transform and stores them in corresponding time and channel sections in the tensor. This reduces the time and memory that is used to store the

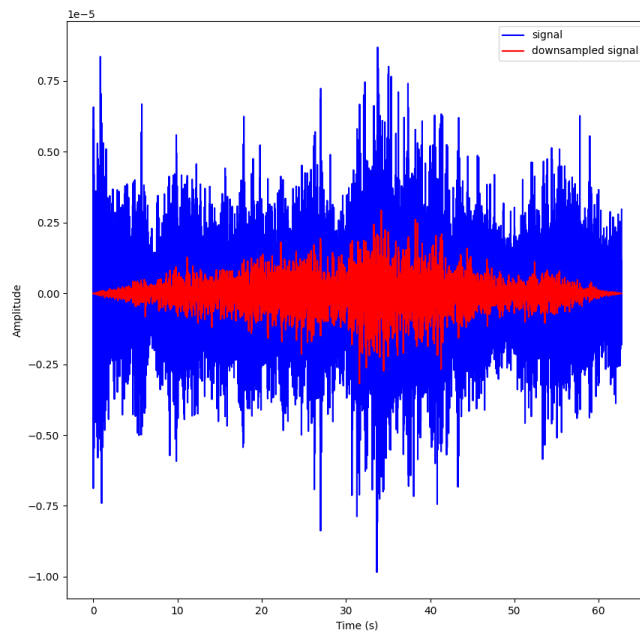
Fourier transform and slices of the tensor at certain time intervals or channels can be taken to see how the data looks over time or distance.



Slices taken from the spectral tensor at channels 0-99 and 950-956.



Slices taken from the spectral tensor at times 1:50:05-1:50:06 and 1:50:51-1:50:52.



Also saved are lowpass and downsampled signals from the original data, where the downsampling rate over time can be specified in the parameter file and the downsampling over space is by a factor of 10.