SPIn4D Data Release 1 Exploration

This notebook is provided to familiarize the user with data artifacts produced by the *Critical Early DKIST Science: Spectropolarimetric Inversion in Four Dimensions with Deep Learning (SPIN4D)* project (funded by NSF#2008344).

This notebook contains a series of sections covering:

1. **Running the Notebook**
2. **Project Overview**
3. **Data Description**
4. **Data Access**
5. **Data Exploration** 1. MURaM cubes 1. Stokes profiles
6. **Data Visualization** 1. MURaM cubes 1. Stokes profiles

Running the Notebook

This notebook has the following dependencies:

- os (operating system functions)
- numpy (numerical python library)
- h5py (H5Py file format)
- matplotlib (standard Python plotting library)

Of course, there is a dependency on the **SPIn4D-DR1** data. This notebook uses a 33GB sample of the 13TB SPIN4D-DR1 dataset. Accessing the data is shown in a section below.

In [1]:
```
import os
import h5py
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
%matplotlib inline
```

Project Overview

The SPIn4D project was performed by a group of researchers at the Institute for Astronomy, University of Hawai'i, Manoa; National Solar Observatory; High Altitude Observatory, NSF National Center for Atmospheric Research led by PI Xudong Sun. Simulations of the small-scale dynamo actions that are prevalent in quiet-Sun and plage regions were performed at the NCAR-Wyoming Supercomputing Center (NWSC) using Matthias Rempel's 2014 version of the Max-Planck University-of-Chicago Radiative MHD code (MURaM). For a complete description of the simulation and data read the paper.

Project members

- Institute for Astronomy, University of Hawaiʻi at Manoa
  - KAI E. YANG (杨凯)
  - S. CURT DODDS
  - IAN CUNNYNGHAM
  - JIAYI LIU (刘嘉奕)
  - XUDONG SUN (孙旭东) (PI)
- High Altitude Observatory, NSF National Center for Atmospheric Research
  - MATTHIAS REMPEL
- National Solar Observatory
  - LUCAS A. TARR
  - SARAH A. JAEGGLI
  - THOMAS A. SCHAD
- Department of Information and Computer Sciences, University of Hawaiʻi at Manoa
  - PETER SADOWSKI
  - YANNIK GLASER

Simulation Cases

Six simulation cases with different mean magnetic fields were conducted. Each case covers six solar-hours with output snapshots stored at a 40s cadence. A detailed description of each case is given in the **Data Description** below.

Spatial Dimension

The volume and spatial resolution of the first 5 cases are **25x25x8Mm** with **16x16x12km** spatial resolution, extending from the upper convection zone up to the temperature minimum region. The volume of the sixth case (SPIN4D_SSD_Large) is **50x50x8Mm** with the same **16x16x12km** spatial resolution.

Stokes Profiles

For each case we synthesized Stokes profiles for two sets of Fe I lines at 630 and 1565 nm for snapshots at 12 minute intervals with two opposite magnetic field orientations (due to the inherent LOS ambiguity). The Stokes profile files were created using our modification of Andrés Asensio Ramos' 3d_sir code that is available on Github here.

Data Description

SPIN4D Data Release 1

This notebook accompanies the **SPIN4D-DR1** data release and paper.

This data set was produced by running Matthias Rempel's 2014 version of the Max-Planck University-of-Chicago Radiative MHD code (MURaM) for simulating the solar photosphere under quiet sun conditions. 5 simulation "cases" were run with a 25x25x8Mm grid volume and with 16x16x12km spatial resolution, each with a different initial mean magnetic field.

- SPIN4D_SSD
- SPIN4D_SSD_50G
- SPIN4D_SSD_50G_V
- SPIN4D_SSD_100G
- SPIN4D_SSD_200G

A sixth case was performed at a 50x50x8Mm volume with the same spatial resolution but with a different initial mean magnetic field applied to each quadrant.

- SPIN4D_SSD_Large

Data Access

The data is stored on project storage at the Institute for Astronomy (IfA), University of Hawai'i at Manoa data center in Honolulu, Hawaii. The data is accessible via HTTP (e.g. curl, wget, etc.) via the IfA data transfer node (DTN) at the data center. It is also accessible as a Globus Data Collection named **SPIN4D-DR1**. A new, third way to access the data is via the Pelican Platform *pelican object get* command using the *namespace* **osdf///uhkoa/SPIN4D-DR1/**. Examples of each data access method are shown below.

To stay organized and limit the size of the download we set several variables in Python and the OS environment. We used **DATA_MODEL** to store the "simulation case". In our case we want to explore the *SPIN4D_SSD_100G* case so we set **DATA_MODEL** accordingly to **100G**.

We choose a single timestep, in this case **090047** and set the **DATA_STEP** variable in both Python and the environment.

We create a path in the Jupyter environment to store the downloaded data locally (wherever Jupyter is running). Be sure you can store 33GB of data locally! We store this as **DATA_PATH**.

In [2]:
```
DATA_MODEL='100G'
os.environ["DATA_MODEL"] = DATA_MODEL

DATA_STEP='090047'
os.environ["DATA_STEP"] = DATA_STEP

DATA_PATH='./data/SPIN4D_SSD_{0}'.format(DATA_MODEL)
os.environ["DATA_PATH"] = DATA_PATH

!mkdir -p $DATA_PATH
!ls $DATA_PATH
```

```
stokes-090047-15648.h5  subdomain_11.090047  subdomain_6.090047
stokes-090047-6302.h5   subdomain_2.090047   subdomain_7.090047
subdomain_0.090047      subdomain_3.090047   subdomain_8.090047
subdomain_1.090047      subdomain_4.090047   subdomain_9.090047
subdomain_10.090047     subdomain_5.090047
```

In [3]:
```
!echo "The local path is" $DATA_PATH.
!echo "The simulation model case is" $DATA_MODEL.
!echo "The simulation timestep is" $DATA_STEP.
```

```
The local path is ./data/SPIN4D_SSD_100G.
The simulation model case is 100G.
The simulation timestep is 090047.
```

Download the SPIN4D-DR1 File Manifest

We will use wget to download the file manifest for **SPIN4D-DR1**. Equivalently, you could use curl if you prefer.

The first row lists the column attributes.

- **run** identifies the case and subdirectory.
- **step** is the timestep
- **file_type** is either *MURaM* or *SIR*.
- **is_flipped** indicates the orientation of the magnetic field along the Z (LOS) axis (*stokes only*).
- **file_name** is the name of the file.

First we will download the manifest "csv" file. The manifest lists each file in the dataset with some metadata about each file.

In [4]:
```
!wget -c -q -nv --show-progress --progress=bar:force http://dtn-itc.ifa.hawaii.edu/spin4d/DR1/sp
```

In [5]:
```
!grep $DATA_STEP spin4d-dr1-manifest.csv
```

```
SPIN4D_SSD_100G,090047,MURaM,-,subdomain_0.090047
SPIN4D_SSD_100G,090047,MURaM,-,subdomain_10.090047
SPIN4D_SSD_100G,090047,MURaM,-,subdomain_1.090047
SPIN4D_SSD_100G,090047,MURaM,-,subdomain_11.090047
SPIN4D_SSD_100G,090047,MURaM,-,subdomain_2.090047
SPIN4D_SSD_100G,090047,MURaM,-,subdomain_3.090047
SPIN4D_SSD_100G,090047,MURaM,-,subdomain_4.090047
SPIN4D_SSD_100G,090047,MURaM,-,subdomain_5.090047
SPIN4D_SSD_100G,090047,MURaM,-,subdomain_6.090047
SPIN4D_SSD_100G,090047,MURaM,-,subdomain_7.090047
SPIN4D_SSD_100G,090047,MURaM,-,subdomain_8.090047
SPIN4D_SSD_100G,090047,MURaM,-,subdomain_9.090047
SPIN4D_SSD_100G,090047,SIR,N,stokes-090047-15648.h5
SPIN4D_SSD_100G,090047,SIR,N,stokes-090047-6302.h5
```

Custom Manifest

We will use this command and slice the last column to create a text file named 'my-manifest' containing the file names we will use in this notebook to explore the kinds of data in the data release.

In [6]:
```
!grep $DATA_STEP spin4d-dr1-manifest.csv|awk -F, '{print $5}' >my-manifest
```

HTTP Access Method

In the following cells we will use the HTTP access method. We use the wget command (available on Linux systems) to use HTTP to download each file from the **SPIN4D-DR1** data repository in Hawaii.

We have tested this in several settings. In Hawaii with a 1Gbps network connection the 33GB download takes us ~24 minutes, about ~7.5 minutes per stokes file and ~48sec per subdomain file.

In [7]:
```
if True:
  !rm -f $DATA_PATH/*
```

In [8]:
```
with open('my-manifest', 'r') as file:
  for line in file:
    fname = line.strip()
    wget_cmd = 'wget -c -q -nv --show-progress --progress=bar:force'
    url = 'http://dtn-itc.ifa.hawaii.edu/spin4d/DR1/SPIN4D_SSD_%s/%s' %(DATA_MODEL, fname)
    http_cmd = '(cd %s;%s %s)' %(DATA_PATH, wget_cmd, url)
    print(http_cmd)
    os.system(http_cmd)
```

```
(cd ./data/SPIN4D_SSD_100G;wget -c -q -nv --show-progress --progress=bar:force http://dtn-itc.if
a.hawaii.edu/spin4d/DR1/SPIN4D_SSD_100G/subdomain_0.090047)
subdomain_0.090047  100%[===================>]   1.12G  27.6MB/s    in 40s
subdomain_10.090047   0%[                    ]       0  --.-KB/s
(cd ./data/SPIN4D_SSD_100G;wget -c -q -nv --show-progress --progress=bar:force http://dtn-itc.if
a.hawaii.edu/spin4d/DR1/SPIN4D_SSD_100G/subdomain_10.090047)
subdomain_10.090047 100%[===================>]   1.12G  29.9MB/s    in 38s
subdomain_1.090047    0%[                    ]       0  --.-KB/s
(cd ./data/SPIN4D_SSD_100G;wget -c -q -nv --show-progress --progress=bar:force http://dtn-itc.if
a.hawaii.edu/spin4d/DR1/SPIN4D_SSD_100G/subdomain_1.090047)
subdomain_1.090047  100%[===================>]   1.12G  29.5MB/s    in 38s
(cd ./data/SPIN4D_SSD_100G;wget -c -q -nv --show-progress --progress=bar:force http://dtn-itc.if
a.hawaii.edu/spin4d/DR1/SPIN4D_SSD_100G/subdomain_11.090047)
subdomain_11.090047 100%[===================>]   1.12G  27.1MB/s    in 42s
subdomain_2.090047    0%[                    ]       0  --.-KB/s
(cd ./data/SPIN4D_SSD_100G;wget -c -q -nv --show-progress --progress=bar:force http://dtn-itc.if
a.hawaii.edu/spin4d/DR1/SPIN4D_SSD_100G/subdomain_2.090047)
subdomain_2.090047  100%[===================>]   1.12G  29.6MB/s    in 39s
subdomain_3.090047    0%[                    ]       0  --.-KB/s
(cd ./data/SPIN4D_SSD_100G;wget -c -q -nv --show-progress --progress=bar:force http://dtn-itc.if
a.hawaii.edu/spin4d/DR1/SPIN4D_SSD_100G/subdomain_3.090047)
subdomain_3.090047  100%[===================>]   1.12G  27.7MB/s    in 42s
(cd ./data/SPIN4D_SSD_100G;wget -c -q -nv --show-progress --progress=bar:force http://dtn-itc.if
a.hawaii.edu/spin4d/DR1/SPIN4D_SSD_100G/subdomain_4.090047)
subdomain_4.090047  100%[===================>]   1.12G  29.4MB/s    in 40s
subdomain_5.090047    0%[                    ]       0  --.-KB/s
(cd ./data/SPIN4D_SSD_100G;wget -c -q -nv --show-progress --progress=bar:force http://dtn-itc.if
a.hawaii.edu/spin4d/DR1/SPIN4D_SSD_100G/subdomain_5.090047)
subdomain_5.090047  100%[===================>]   1.12G  28.4MB/s    in 38s
subdomain_6.090047    0%[                    ]       0  --.-KB/s
(cd ./data/SPIN4D_SSD_100G;wget -c -q -nv --show-progress --progress=bar:force http://dtn-itc.if
a.hawaii.edu/spin4d/DR1/SPIN4D_SSD_100G/subdomain_6.090047)
```

```
subdomain_6.090047  100%[===================>]   1.12G  28.9MB/s    in 40s
subdomain_7.090047    0%[                    ]       0  --.-KB/s
(cd ./data/SPIN4D_SSD_100G;wget -c -q -nv --show-progress --progress=bar:force http://dtn-itc.if
a.hawaii.edu/spin4d/DR1/SPIN4D_SSD_100G/subdomain_7.090047)
subdomain_7.090047  100%[===================>]   1.12G  30.4MB/s    in 39s
subdomain_8.090047    0%[                    ]       0  --.-KB/s
(cd ./data/SPIN4D_SSD_100G;wget -c -q -nv --show-progress --progress=bar:force http://dtn-itc.if
a.hawaii.edu/spin4d/DR1/SPIN4D_SSD_100G/subdomain_8.090047)
subdomain_8.090047  100%[===================>]   1.12G  29.2MB/s    in 39s
subdomain_9.090047    0%[                    ]       0  --.-KB/s
(cd ./data/SPIN4D_SSD_100G;wget -c -q -nv --show-progress --progress=bar:force http://dtn-itc.if
a.hawaii.edu/spin4d/DR1/SPIN4D_SSD_100G/subdomain_9.090047)
subdomain_9.090047  100%[===================>]   1.12G  28.9MB/s    in 42s
(cd ./data/SPIN4D_SSD_100G;wget -c -q -nv --show-progress --progress=bar:force http://dtn-itc.if
a.hawaii.edu/spin4d/DR1/SPIN4D_SSD_100G/stokes-090047-15648.h5)
stokes-090047-15648 100%[===================>]   9.00G  30.2MB/s    in 5m 37s
stokes-090047-6302.   0%[                    ]       0  --.-KB/s
(cd ./data/SPIN4D_SSD_100G;wget -c -q -nv --show-progress --progress=bar:force http://dtn-itc.if
a.hawaii.edu/spin4d/DR1/SPIN4D_SSD_100G/stokes-090047-6302.h5)
stokes-090047-6302. 100%[===================>]   9.70G  29.9MB/s    in 5m 38s
```

Check the Download Results

Let's check the results of our download by inspecting the local directory in which we stored the downloaded files.

In [9]:

```
!echo "Stokes profiles:"
!ls -l $DATA_PATH/stokes*

!echo "MURaM cubes:"
!ls -l $DATA_PATH/subdomain*

!echo "Total disk space used:"
!du -h $DATA_PATH
```

```
Stokes profiles:
-rw-r--r-- 1 root root  9663830576 Aug 18  2022 ./data/SPIN4D_SSD_100G/stokes-090047-15648.h5
-rw-r--r-- 1 root root 10418792832 Aug 18  2022 ./data/SPIN4D_SSD_100G/stokes-090047-6302.h5
MURaM cubes:
-rw-r--r-- 1 root root 1207959552 Jul 30  2022 ./data/SPIN4D_SSD_100G/subdomain_0.090047
-rw-r--r-- 1 root root 1207959552 Jul 30  2022 ./data/SPIN4D_SSD_100G/subdomain_1.090047
-rw-r--r-- 1 root root 1207959552 Jul 30  2022 ./data/SPIN4D_SSD_100G/subdomain_10.090047
-rw-r--r-- 1 root root 1207959552 Jul 30  2022 ./data/SPIN4D_SSD_100G/subdomain_11.090047
-rw-r--r-- 1 root root 1207959552 Jul 30  2022 ./data/SPIN4D_SSD_100G/subdomain_2.090047
-rw-r--r-- 1 root root 1207959552 Jul 30  2022 ./data/SPIN4D_SSD_100G/subdomain_3.090047
-rw-r--r-- 1 root root 1207959552 Jul 30  2022 ./data/SPIN4D_SSD_100G/subdomain_4.090047
-rw-r--r-- 1 root root 1207959552 Jul 30  2022 ./data/SPIN4D_SSD_100G/subdomain_5.090047
-rw-r--r-- 1 root root 1207959552 Jul 30  2022 ./data/SPIN4D_SSD_100G/subdomain_6.090047
-rw-r--r-- 1 root root 1207959552 Jul 30  2022 ./data/SPIN4D_SSD_100G/subdomain_7.090047
-rw-r--r-- 1 root root 1207959552 Jul 30  2022 ./data/SPIN4D_SSD_100G/subdomain_8.090047
-rw-r--r-- 1 root root 1207959552 Jul 30  2022 ./data/SPIN4D_SSD_100G/subdomain_9.090047
Total disk space used:
33G     ./data/SPIN4D_SSD_100G
```

Globus Access Method

Globus File Transfer is a supported access method. The Globus "data collection" name is **SPIN4D-DR1**. To use Globus File Transfer to download files from the **SPIN4D-DR1** Globus Collection in Hawaii follow these steps:

1. Email the **SPIN4D-DR1** data manager requesting access. Provide your email address and intended use (for reference).
2. You will receive an email when your authorization is approved.
3. To download data, browse to this Globus File Transfer URL.

Learn more about using the interactive browser-based Globus File Transfer: https://docs.globus.org/guides/tutorials/manage-files/transfer-files/.

Learn about installing the Globus Transfer command line utility: https://docs.globus.org/cli/reference/transfer/.

Open Science Data Federation Access Method

The Open Science Data Federation (OSDF) is a content distribution network (CDN) for science data. The OSDF is a network of geodistributed data caches accessed using the Pelican Platform software. Datasets are provided by **data-origins** and automatically copied to the closest **data-cache** to the requestor.

Let's give it a try!

Here are the steps we will perform next.

1. Download the binary distribution of Pelican for linux
2. Install the Pelican client
3. Use the Pelican client to download a test file.
4. Use Pelican to download **SPIN4D-DR1** data files.

Download Pelican

For convenience we use the binary download method. There are other ways to obtain Pelican that may be better suited for production. You can explore the different downloads here.

In [49]:
```
!wget https://github.com/PelicanPlatform/pelican/releases/latest/download/pelican_Linux_x86_64.ta
!ls -l pelican_Linux_x86_64.tar.*
```

```
--2024-10-02 05:07:00--  https://github.com/PelicanPlatform/pelican/releases/latest/download/peli
can_Linux_x86_64.tar.gz
Resolving github.com (github.com)... 140.82.116.4
Connecting to github.com (github.com)|140.82.116.4|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://github.com/PelicanPlatform/pelican/releases/download/v7.10.5/pelican_Linux_x86_
64.tar.gz [following]
--2024-10-02 05:07:00--  https://github.com/PelicanPlatform/pelican/releases/download/v7.10.5/pel
ican_Linux_x86_64.tar.gz
Reusing existing connection to github.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://objects.githubusercontent.com/github-production-release-asset-2e65be/652665253/
02ceff17-9092-4034-bc42-321425cd6846?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=releaseass
etproduction%2F20241002%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20241002T050152Z&X-Amz-Expires
=300&X-Amz-Signature=7127221a77cc23c7ce0227d72e40d8933591ce32a3da55db2c5e391f97faf9a8&X-Amz-Signe
dHeaders=host&response-content-disposition=attachment%3B%20filename%3Dpelican_Linux_x86_64.tar.gz
&response-content-type=application%2Foctet-stream [following]
--2024-10-02 05:07:00--  https://objects.githubusercontent.com/github-production-release-asset-2e
65be/652665253/02ceff17-9092-4034-bc42-321425cd6846?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Creden
tial=releaseassetproduction%2F20241002%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20241002T050152
Z&X-Amz-Expires=300&X-Amz-Signature=7127221a77cc23c7ce0227d72e40d8933591ce32a3da55db2c5e391f97faf
9a8&X-Amz-SignedHeaders=host&response-content-disposition=attachment%3B%20filename%3Dpelican_Linu
x_x86_64.tar.gz&response-content-type=application%2Foctet-stream
Resolving objects.githubusercontent.com (objects.githubusercontent.com)... 185.199.109.133, 185.1
99.110.133, 185.199.111.133, ...
Connecting to objects.githubusercontent.com (objects.githubusercontent.com)|185.199.109.133|:44
3... connected.
HTTP request sent, awaiting response... 200 OK
Length: 16056804 (15M) [application/octet-stream]
Saving to: ‘pelican_Linux_x86_64.tar.gz’

pelican_Linux_x86_6 100%[===================>]  15.31M   597KB/s    in 28s

2024-10-02 05:07:29 (567 KB/s) - ‘pelican_Linux_x86_64.tar.gz’ saved [16056804/16056804]

-rw-r--r-- 1 root root 16056804 Sep 19 14:03 pelican_Linux_x86_64.tar.gz
```

Extract Files

In [11]:
```
!tar -zxvf pelican_Linux_x86_64.tar.gz
```

```
pelican-7.10.5/LICENSE
pelican-7.10.5/README.md
pelican-7.10.5/pelican
```

Download a Test File

In [12]:
```
# Prepare the path to the pelican binary
pelican_path = os.popen("find . -type d -name 'pelican-*'").read().split('\n')[0]
pelican_path = os.path.join(pelican_path,'pelican')
print("Path to the pelican command:\n  " + pelican_path)

pelican_cmd = pelican_path + ' object get pelican://osg-htc.org/ospool/uc-shared/public/OSG-Staf
print("\nCommand to download a test file using pelican:\n  " + pelican_cmd)
os.system(pelican_cmd)
```

```
Path to the pelican command:
  ./pelican-7.10.5/pelican

Command to download a test file using pelican:
  ./pelican-7.10.5/pelican object get pelican://osg-htc.org/ospool/uc-shared/public/OSG-Staff/val
idation/test.txt .
```

Out[12]: 0

In [13]:
```
ls -l test.txt
```

```
-rw-r--r-- 1 1000 1000 14 Oct  1 04:24 test.txt
```

Download SPIN4D-DR1 Data

We prepared a list of the **SPIN4D-DR1** files used used by this notebook in a text file named **my-manifest**.

Here are the steps we will take:

1. List the contents of *my-manifest*
2. For each file in the manifest run *pelican object get* to copy the file to our local data directory.

In [14]:
```
print("Here are the files that we need:")
with open('my-manifest', 'r') as file:
  for line in file:
    print('  '+line.strip())
```

```
Here are the files that we need:
  subdomain_0.090047
  subdomain_10.090047
  subdomain_1.090047
  subdomain_11.090047
  subdomain_2.090047
  subdomain_3.090047
  subdomain_4.090047
  subdomain_5.090047
  subdomain_6.090047
  subdomain_7.090047
  subdomain_8.090047
  subdomain_9.090047
  stokes-090047-15648.h5
  stokes-090047-6302.h5
```

Now we will use Pelican to download our files!

In [15]:
```
OSDF_PATH='SPIN4D_SSD_{0}'.format(DATA_MODEL)
with open('my-manifest', 'r') as file:
  for line in file:
    fname = line.strip()
    pelican_cmd = pelican_path + ' object get osdf:///uhkoa/SPIN4D-DR1/' + os.path.join(OSDF_PATI
    print(pelican_cmd)
    os.system(pelican_cmd)
```

```
./pelican-7.10.5/pelican object get osdf:///uhkoa/SPIN4D-DR1/SPIN4D_SSD_100G/subdomain_0.090047
./data/SPIN4D_SSD_100G
./pelican-7.10.5/pelican object get osdf:///uhkoa/SPIN4D-DR1/SPIN4D_SSD_100G/subdomain_10.090047
./data/SPIN4D_SSD_100G
./pelican-7.10.5/pelican object get osdf:///uhkoa/SPIN4D-DR1/SPIN4D_SSD_100G/subdomain_1.090047
./data/SPIN4D_SSD_100G
./pelican-7.10.5/pelican object get osdf:///uhkoa/SPIN4D-DR1/SPIN4D_SSD_100G/subdomain_11.090047
./data/SPIN4D_SSD_100G
./pelican-7.10.5/pelican object get osdf:///uhkoa/SPIN4D-DR1/SPIN4D_SSD_100G/subdomain_2.090047
./data/SPIN4D_SSD_100G
./pelican-7.10.5/pelican object get osdf:///uhkoa/SPIN4D-DR1/SPIN4D_SSD_100G/subdomain_3.090047
./data/SPIN4D_SSD_100G
./pelican-7.10.5/pelican object get osdf:///uhkoa/SPIN4D-DR1/SPIN4D_SSD_100G/subdomain_4.090047
./data/SPIN4D_SSD_100G
./pelican-7.10.5/pelican object get osdf:///uhkoa/SPIN4D-DR1/SPIN4D_SSD_100G/subdomain_5.090047
./data/SPIN4D_SSD_100G
./pelican-7.10.5/pelican object get osdf:///uhkoa/SPIN4D-DR1/SPIN4D_SSD_100G/subdomain_6.090047
./data/SPIN4D_SSD_100G
./pelican-7.10.5/pelican object get osdf:///uhkoa/SPIN4D-DR1/SPIN4D_SSD_100G/subdomain_7.090047
./data/SPIN4D_SSD_100G
./pelican-7.10.5/pelican object get osdf:///uhkoa/SPIN4D-DR1/SPIN4D_SSD_100G/subdomain_8.090047
./data/SPIN4D_SSD_100G
./pelican-7.10.5/pelican object get osdf:///uhkoa/SPIN4D-DR1/SPIN4D_SSD_100G/subdomain_9.090047
./data/SPIN4D_SSD_100G
./pelican-7.10.5/pelican object get osdf:///uhkoa/SPIN4D-DR1/SPIN4D_SSD_100G/stokes-090047-15648.
h5 ./data/SPIN4D_SSD_100G
./pelican-7.10.5/pelican object get osdf:///uhkoa/SPIN4D-DR1/SPIN4D_SSD_100G/stokes-090047-6302.h
5 ./data/SPIN4D_SSD_100G
```

Data Exploration

SPIN-4D MURaM cube snapshots

Each of "case" is a series of snapshots in time of the state of 12 physical parameters of the solar atmosphere equation of state. The **SPIN4D-DR1** dataset selects snapshots 12 minutes apart so that convective cells in the solar atmosphere are uncorrelated in each successive snapshot.

In this notebook we will use the 100G initialization case and focus on timestep 090047.

Let's list the 12 files that contain the data for case SPIN4D_SSD_100G, timestep 090047.

In [16]:
```
!ls -l $DATA_PATH/subdomain_*.$DATA_STEP
```

```
-rw-r--r-- 1 root root 1207959552 Jul 30  2022 ./data/SPIN4D_SSD_100G/subdomain_0.090047
-rw-r--r-- 1 root root 1207959552 Jul 30  2022 ./data/SPIN4D_SSD_100G/subdomain_1.090047
-rw-r--r-- 1 root root 1207959552 Jul 30  2022 ./data/SPIN4D_SSD_100G/subdomain_10.090047
-rw-r--r-- 1 root root 1207959552 Jul 30  2022 ./data/SPIN4D_SSD_100G/subdomain_11.090047
-rw-r--r-- 1 root root 1207959552 Jul 30  2022 ./data/SPIN4D_SSD_100G/subdomain_2.090047
-rw-r--r-- 1 root root 1207959552 Jul 30  2022 ./data/SPIN4D_SSD_100G/subdomain_3.090047
-rw-r--r-- 1 root root 1207959552 Jul 30  2022 ./data/SPIN4D_SSD_100G/subdomain_4.090047
-rw-r--r-- 1 root root 1207959552 Jul 30  2022 ./data/SPIN4D_SSD_100G/subdomain_5.090047
-rw-r--r-- 1 root root 1207959552 Jul 30  2022 ./data/SPIN4D_SSD_100G/subdomain_6.090047
-rw-r--r-- 1 root root 1207959552 Jul 30  2022 ./data/SPIN4D_SSD_100G/subdomain_7.090047
-rw-r--r-- 1 root root 1207959552 Jul 30  2022 ./data/SPIN4D_SSD_100G/subdomain_8.090047
-rw-r--r-- 1 root root 1207959552 Jul 30  2022 ./data/SPIN4D_SSD_100G/subdomain_9.090047
```

Synthetic Stokes Profiles

For each simulation timestep snapshot the **SIR** program was used to synthesize line of sight **Stokes Profiles** for the absorption lines at 6302A and 15648A. These absorption lines are specific to an iron electron's energy state which in turn determines the wavelength band. Our data set includes synthetic Stokes profiles for two lines, FeI 6302 and 15648 (units are Angstroms). Each (Y,X) pixel of the MURaM cube (along Z, the line of sight) produced a Stokes vector that was saved in HDF5 format.

In [17]:
```
!ls -l $DATA_PATH/stokes-$DATA_STEP*
```

```
-rw-r--r-- 1 root root  9663830576 Aug 18  2022 ./data/SPIN4D_SSD_100G/stokes-090047-15648.h5
-rw-r--r-- 1 root root 10418792832 Aug 18  2022 ./data/SPIN4D_SSD_100G/stokes-090047-6302.h5
```

Let's open a stokes file and look inside ...

In [18]:
```
stokesPath = os.path.join(DATA_PATH,'stokes-%s-6302.h5' % (DATA_STEP))
print(stokesPath)
stokes = h5py.File(stokesPath,'r')
```

```
./data/SPIN4D_SSD_100G/stokes-090047-6302.h5
```

In [19]:
```
print(stokes.keys())
```

```
<KeysViewHDF5 ['I', 'Q', 'U', 'V', 'lambda']>
```

In [20]:
```
print(stokes.attrs.keys())
base_wl = stokes.attrs['lambda_zeropoint']
print(base_wl)
```

```
<KeysViewHDF5 ['lambda_zeropoint']>
6301508.0
```

The stokes profile HDF5 file

Each Stokes parameter is stored as a 3D matrix (Y,X,WL) with wavelength sampled in the 3rd dimension.

In [21]:
```
si = np.array(stokes['I'])
sq = np.array(stokes['Q'])
su = np.array(stokes['U'])
```

```
sv = np.array(stokes['V'])
print(si.shape)
```

```
(1536, 1536, 276)
```

The wavelengths can be "decoded" by referencing the "lambda" object.

In [22]:
```
wl = np.array(stokes['lambda'])
print(wl.shape)
```

```
(276,)
```

In [23]:
```
stokes.close()
```

Let's have a look at the "lambda" vector ...

In [24]:
```
for w in range(10):
  print(w, wl[w])
```

```
0 -655.9
1 -646.95526
2 -638.01056
3 -629.0658
4 -620.1211
5 -611.1764
6 -602.2316
7 -593.2869
8 -584.34216
9 -575.39746
```

In [25]:
```
for w in range(-10,0):
  print(w, wl[w])
```

```
-10 1723.3973
-9 1732.3422
-8 1741.2867
-7 1750.2316
-6 1759.1761
-5 1768.121
-4 1777.0658
-3 1786.0104
-2 1794.9552
-1 1803.8998
```

Converting to Angstroms

Wait a minute! Why are these negative numbers?

The lambda values are in Angstrom units but they are offsets from a reference wavelength. We can get the reference wavelength in Angstroms from the file name. In this case it's 6302 so we can add this to the lambda values to get Angstroms.

In [26]:
```
for w in range(10):
  print(w, "%.2f" %(base_wl+wl[w]))
print("...")
for w in range(10):
  print(w, "%.2f" %(base_wl+wl[-(10-w)]))
```

```
0 6300852.10
1 6300861.04
2 6300869.99
3 6300878.93
4 6300887.88
5 6300896.82
6 6300905.77
7 6300914.71
8 6300923.66
9 6300932.60
...
0 6303231.40
1 6303240.34
2 6303249.29
3 6303258.23
4 6303267.18
5 6303276.12
6 6303285.07
7 6303294.01
8 6303302.96
9 6303311.90
```

The wavelength dimension

The wavelength dimension varies with the absorption line:

- 6302A line: 276
- 15648A line: 256

We will truncate the wavelength dimension and use the same dimension (256) for both lines.

In [27]:
```
wavelen=100
si = si[:,:,10:266]
sq = sq[:,:,10:266]
su = su[:,:,10:266]
sv = sv[:,:,10:266]
wave = wl[10:266] + base_wl
wave *= 1e-3
```

Dimensionality of the data

One can treat the Stokes profiles as a 3D volume with 2 spatial dimensions, wavelength in the z dimension and 4 channels:

- 6302: shape (1536, 1536, 256, 4)
- 15648: shape (1536, 1536, 256, 4)

In [28]:
```
X = np.empty((1536, 1536, 256, 4))
X[:,:,:,0] = si
X[:,:,:,1] = sq
X[:,:,:,2] = su
```

```
X[:,:,:,3] = sv
X.shape
```

Out[28]: (1536, 1536, 256, 4)

Data Visualization

Visualizing MURam Cubes

Simulation MURaM cube snapshots

For a given simulation run there are ~20 snapshots of the MURaM cube parameter values. The MURaM simulation grid is 1536 x 1536 x 128 or 1536 "pixels" square sampled at 128 levels in the (simulated) solar atmosphere.

MURaM cube dimensionality

The MURaM cube axes correspond to solar (X, Y, Z) where:

- Solar Y = North to South
- Solar X = East to West
- Solar Z = Center to Surface

Each snapshot consists of 12 files. Each file stores 1 of the 12 parameters. Each file contains a 3D numpy array with shape: (1536, 1536, 128). The data is stored in the file in (X, Y, Z) order so we need to swap axes after reading in the data. Each snapshot is uniquely identified by a sequence number suffix, e.g. **090047**.

In [29]:
```
!ls -l $DATA_PATH | grep $DATA_STEP | grep subdomain
```

```
-rw-r--r-- 1 root root  1207959552 Jul 30  2022 subdomain_0.090047
-rw-r--r-- 1 root root  1207959552 Jul 30  2022 subdomain_1.090047
-rw-r--r-- 1 root root  1207959552 Jul 30  2022 subdomain_10.090047
-rw-r--r-- 1 root root  1207959552 Jul 30  2022 subdomain_11.090047
-rw-r--r-- 1 root root  1207959552 Jul 30  2022 subdomain_2.090047
-rw-r--r-- 1 root root  1207959552 Jul 30  2022 subdomain_3.090047
-rw-r--r-- 1 root root  1207959552 Jul 30  2022 subdomain_4.090047
-rw-r--r-- 1 root root  1207959552 Jul 30  2022 subdomain_5.090047
-rw-r--r-- 1 root root  1207959552 Jul 30  2022 subdomain_6.090047
-rw-r--r-- 1 root root  1207959552 Jul 30  2022 subdomain_7.090047
-rw-r--r-- 1 root root  1207959552 Jul 30  2022 subdomain_8.090047
-rw-r--r-- 1 root root  1207959552 Jul 30  2022 subdomain_9.090047
```

MURaM cube parameters

The following table is a key to mapping the numeric "subdomain_" file names to recognizable parameters of physics equations.

Abbrev	File Name	Parameter
rho	subdomain_0	density

Abbrev	File Name	Parameter
vx	subdomain_1	velocity_x
vy	subdomain_2	velocity_y
vz	subdomain_3	velocity_z
eint	subdomain_4	internal electron pressure
Bx	subdomain_5	magnetic field_x
By	subdomain_6	magnetic field_y
Bz	subdomain_7	magnetic field_z
T	subdomain_8	temperature
P	subdomain_9	pressure
ne	subdomain_10	number of electrons
tau500	subdomain_11	opacity at 500nm

In [30]:
```
cubeParam = {
  'rho': 'subdomain_0',
  'vx': 'subdomain_1',
  'vy': 'subdomain_2',
  'vz': 'subdomain_3',
  'eint': 'subdomain_4',
  'Bx': 'subdomain_5',
  'By': 'subdomain_6',
  'Bz': 'subdomain_7',
  'T': 'subdomain_8',
  'P': 'subdomain_9',
  'ne': 'subdomain_10',
  'tau': 'subdomain_11',
}
```

In [31]:
```
cube = {}
```

We provide a helper function to read and reshape a MURaM cube from a file.

In [32]:
```
def read_cube(param_key, model, sequence):
  paramPath = os.path.join("./data",'SPIN4D_SSD_{0}/'.format(model),cubeParam[param_key]+'.'+sequ
  #print('get {0} from {1}'.format(param_key, paramPath))
  tmp = np.fromfile(paramPath, dtype = np.float32)
  tmpa = tmp.reshape((1536,1536,128))
  return tmpa.transpose(1, 0, 2)
```

Visualize slices of the Sun's atmosphere

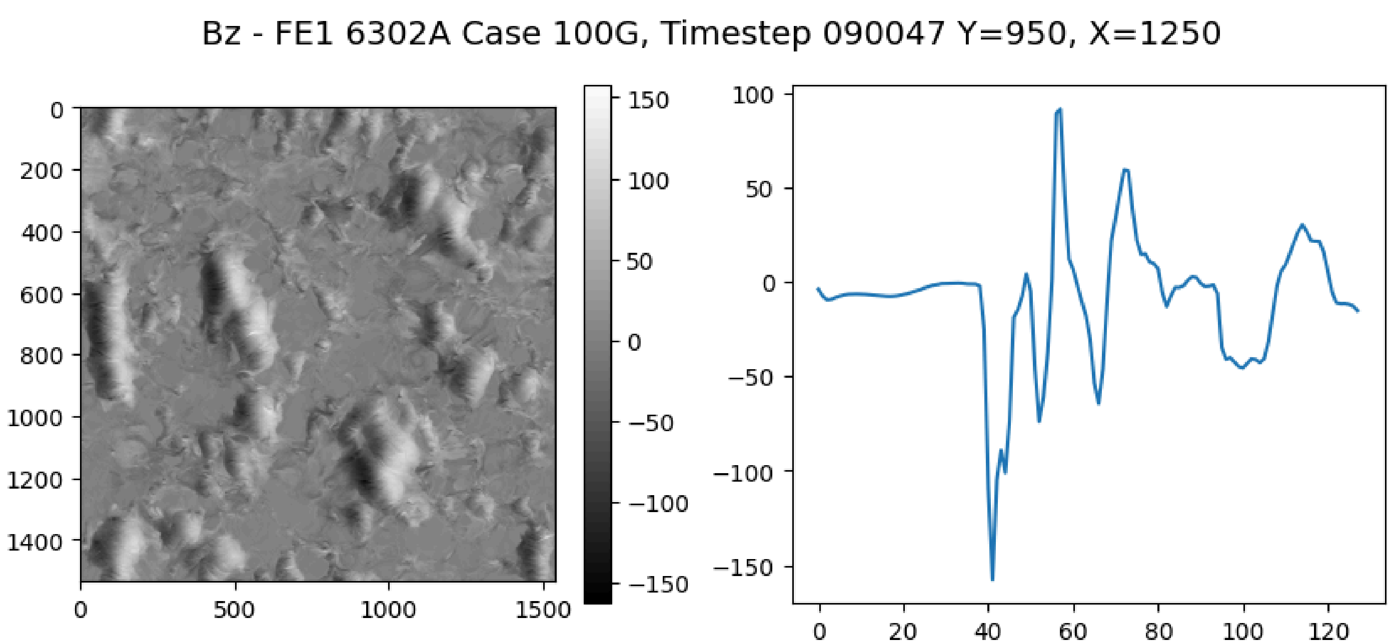
We can visualize a cross sectional slice at half the depth of the cube grid (z = 64) and plot a single line of sight looking from the center of the Sun to Earth (the surface of the Sun is at the right)

In [33]:
```
position = (950,1250,100)
```

Magnetic field strength

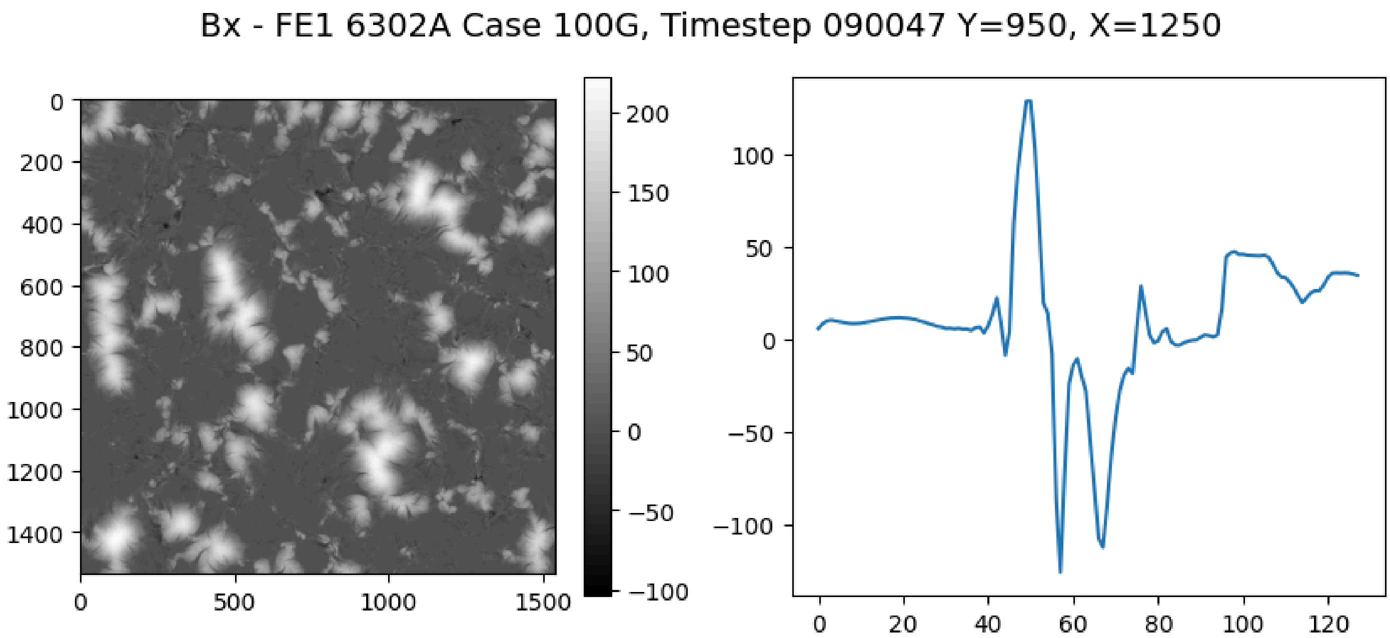
The magnetic field is described by 3 parameters: (Bx, By, Bz).

In [34]:
```
cube['Bz'] = read_cube('Bz', DATA_MODEL, DATA_STEP)
cube['Bz'].shape


title='Bz - FE1 6302A Case {2}, Timestep {3} Y={0}, X={1}'.format(position[0],position[1],DATA_M

plt.rcParams['figure.figsize'] = [10, 4]
plt.rcParams['figure.dpi'] = 100
fig, axs = plt.subplots(1, 2)
fig.suptitle(title, fontsize=14)

plt.subplot(1, 2, 1)
plt.imshow(cube['Bz'][:,:,position[2]],cmap="gray")
plt.colorbar()

plt.subplot(1, 2, 2)
plt.plot(cube['Bz'][position[0],position[1],:])
cube['Bz'] = [] # release memory
```

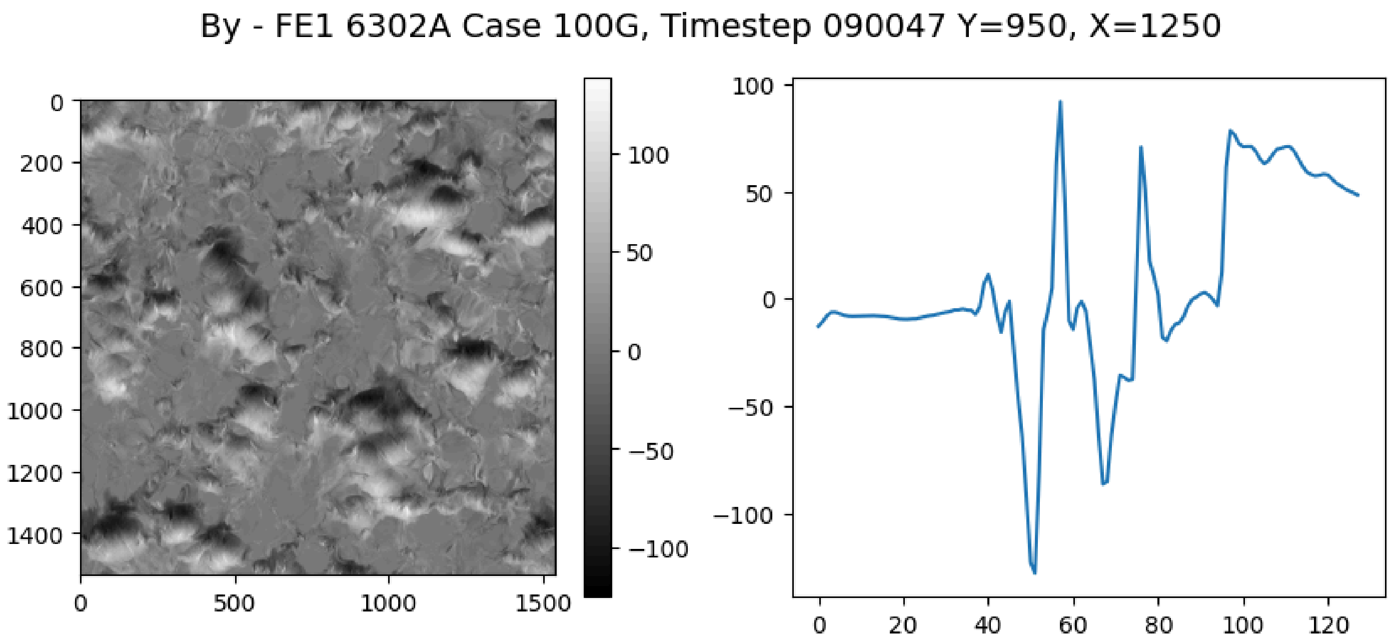


In [35]:
```
cube['Bx'] = read_cube('Bx', DATA_MODEL, DATA_STEP)
cube['Bx'].shape

title='Bx - FE1 6302A Case {2}, Timestep {3} Y={0}, X={1}'.format(position[0],position[1],DATA_M

plt.rcParams['figure.figsize'] = [10, 4]
plt.rcParams['figure.dpi'] = 100
fig, axs = plt.subplots(1, 2)
fig.suptitle(title, fontsize=14)

plt.subplot(1, 2, 1)
plt.imshow(cube['Bx'][:,:,position[2]],cmap="gray")
plt.colorbar()
```

```
plt.subplot(1, 2, 2)
plt.plot(cube['Bx'][position[0],position[1],:])
cube['Bx'] = []
```

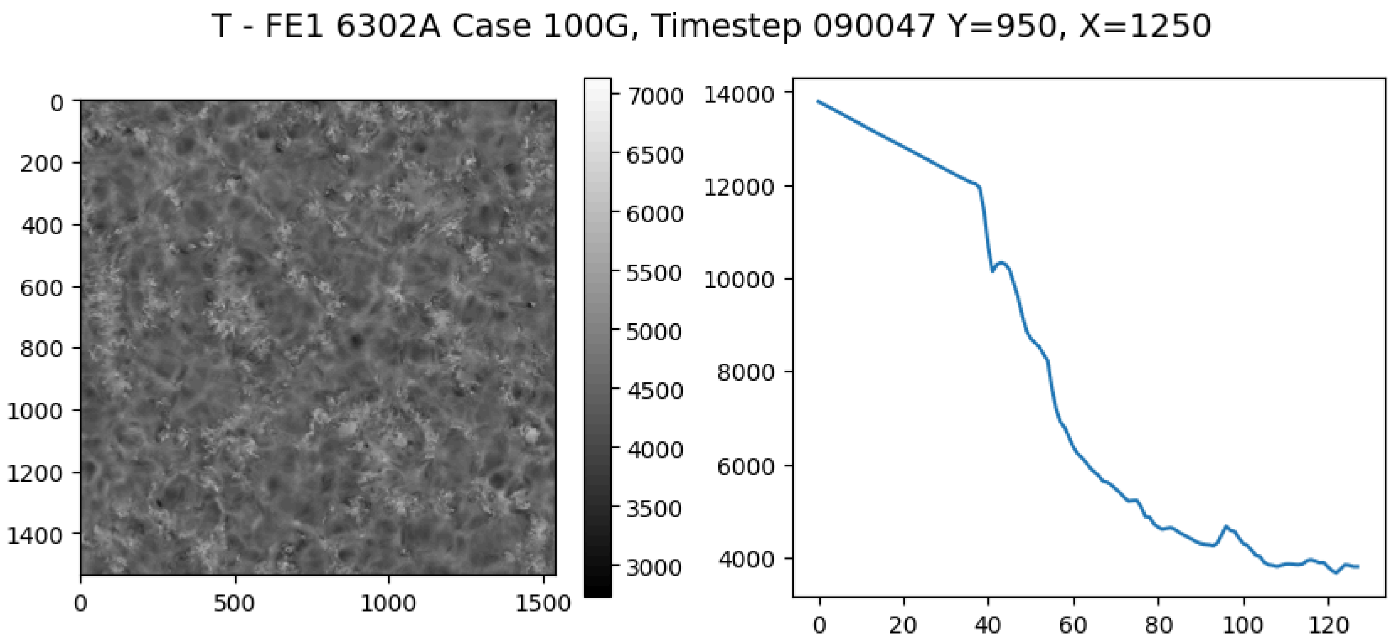


In [36]:
```
cube['By'] = read_cube('By', DATA_MODEL, DATA_STEP)
cube['By'].shape

title='By - FE1 6302A Case {2}, Timestep {3} Y={0}, X={1}'.format(position[0],position[1],DATA_M(

plt.rcParams['figure.figsize'] = [10, 4]
plt.rcParams['figure.dpi'] = 100 # 200 e.g. is really fine, but slower
fig, axs = plt.subplots(1, 2)
fig.suptitle(title, fontsize=14)

plt.subplot(1, 2, 1)
plt.imshow(cube['By'][:,:,position[2]],cmap="gray")
plt.colorbar()
plt.subplot(1, 2, 2)
plt.plot(cube['By'][position[0],position[1],:])
cube['By'] = []
```

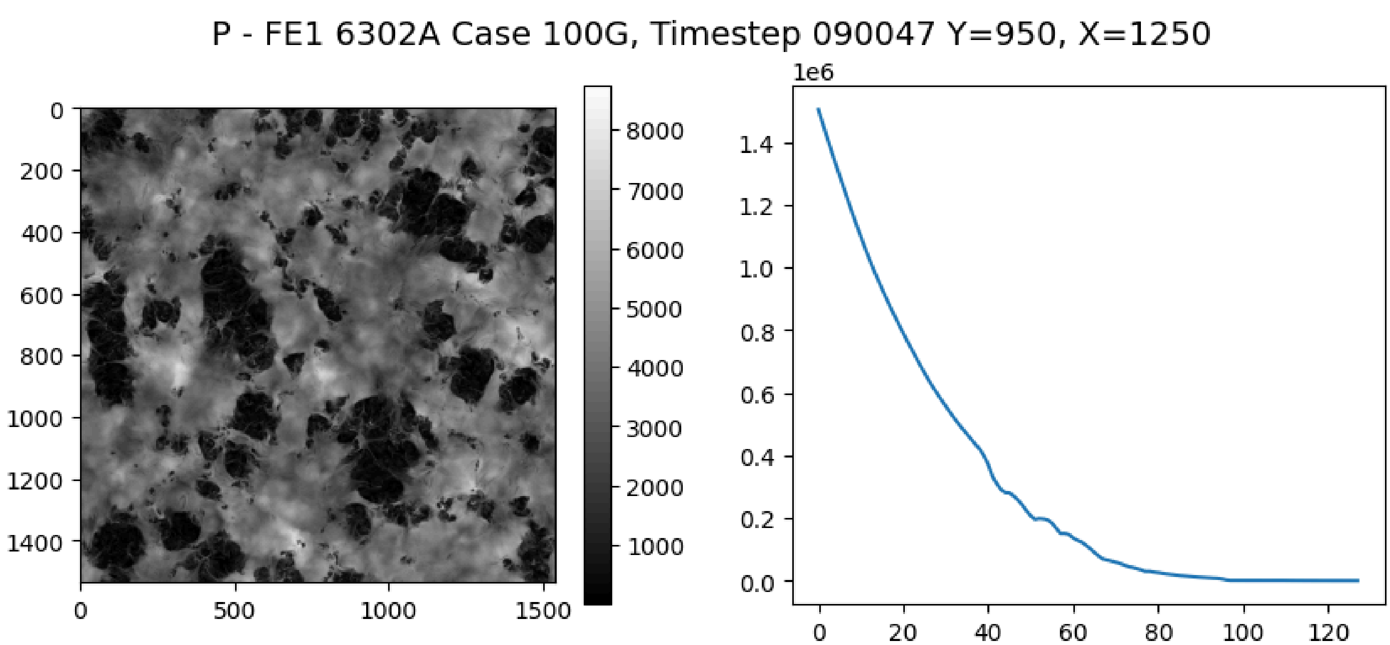


Temperature, Pressure and Density

In [37]:
```
cube['T'] = read_cube('T', DATA_MODEL, DATA_STEP)
cube['T'].shape

title='T - FE1 6302A Case {2}, Timestep {3} Y={0}, X={1}'.format(position[0],position[1],DATA_MO

plt.rcParams['figure.figsize'] = [10, 4]
plt.rcParams['figure.dpi'] = 100 # 200 e.g. is really fine, but slower
fig, axs = plt.subplots(1, 2)
fig.suptitle(title, fontsize=14)

plt.subplot(1, 2, 1)
plt.imshow(cube['T'][:,:,position[2]],cmap="gray")
plt.colorbar()
plt.subplot(1, 2, 2)
plt.plot(cube['T'][position[0],position[1],:])
cube['T'] = []
```

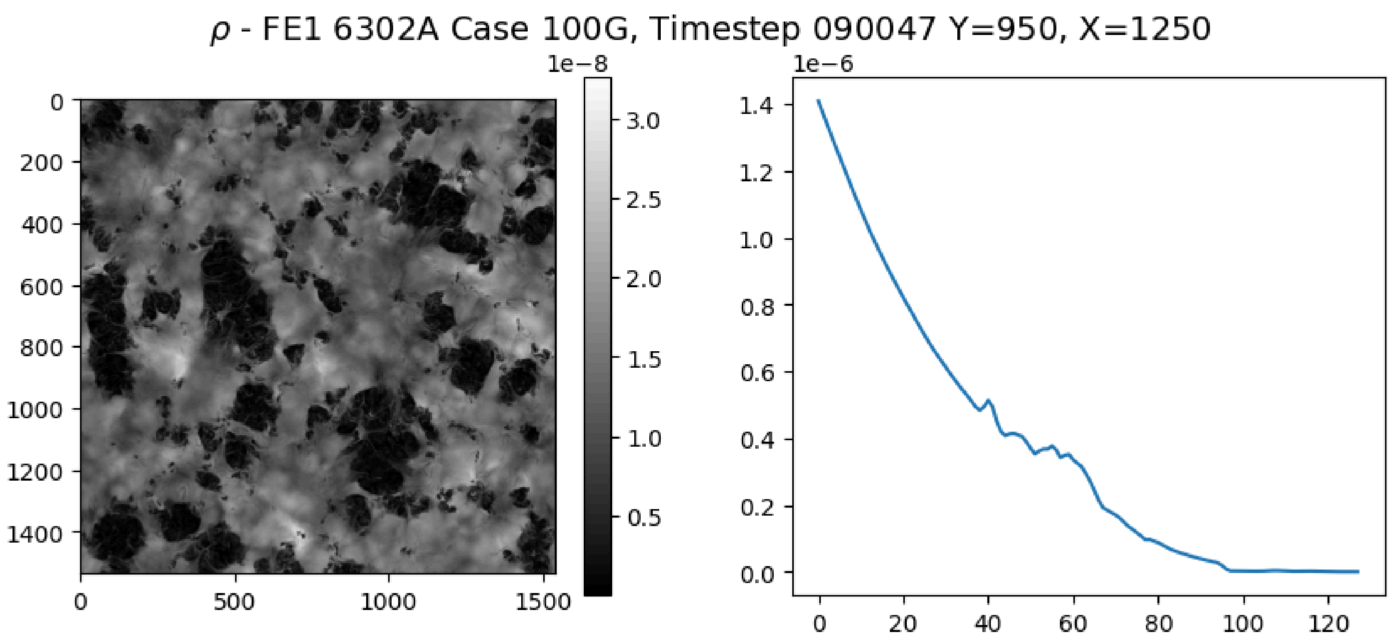


In [38]:
```python
cube['P'] = read_cube('P', DATA_MODEL, DATA_STEP)
cube['P'].shape

title='P - FE1 6302A Case {2}, Timestep {3} Y={0}, X={1}'.format(position[0],position[1],DATA_MO

plt.rcParams['figure.figsize'] = [10, 4]
plt.rcParams['figure.dpi'] = 100 # 200 e.g. is really fine, but slower
fig, axs = plt.subplots(1, 2)
fig.suptitle(title, fontsize=14)

plt.subplot(1, 2, 1)
plt.imshow(cube['P'][:,:,position[2]],cmap="gray")
plt.colorbar()
plt.subplot(1, 2, 2)
plt.plot(cube['P'][position[0],position[1],:])
cube['P'] = []
```



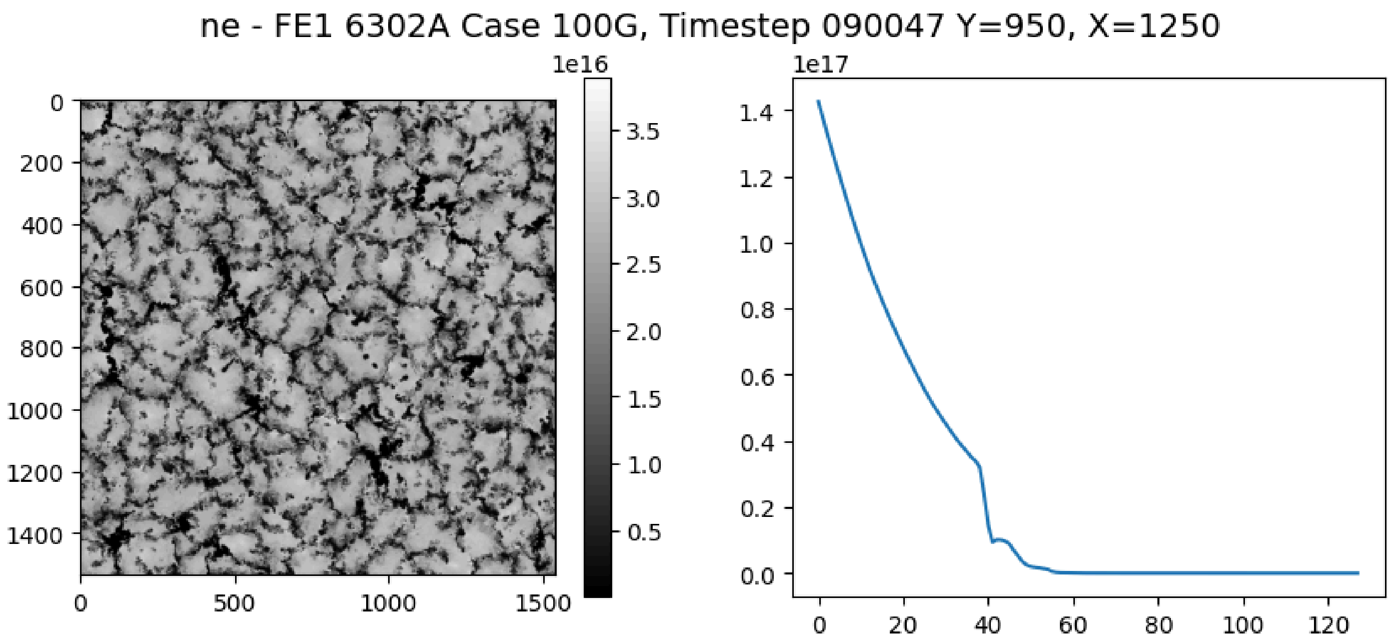
In [39]:
```python
cube['rho'] = read_cube('rho', DATA_MODEL, DATA_STEP)
cube['rho'].shape
```

```
title=r'$\rho$ - FE1 6302A Case {2}, Timestep {3} Y={0}, X={1}'.format(position[0],position[1],DA

plt.rcParams['figure.figsize'] = [10, 4]
plt.rcParams['figure.dpi'] = 100 # 200 e.g. is really fine, but slower
fig, axs = plt.subplots(1, 2)
fig.suptitle(title, fontsize=14)

plt.subplot(1, 2, 1)
plt.imshow(cube['rho'][:,:,position[2]],cmap="gray")
plt.colorbar()
plt.subplot(1, 2, 2)
plt.plot(cube['rho'][position[0],position[1],:])
cube['rho'] = []
```



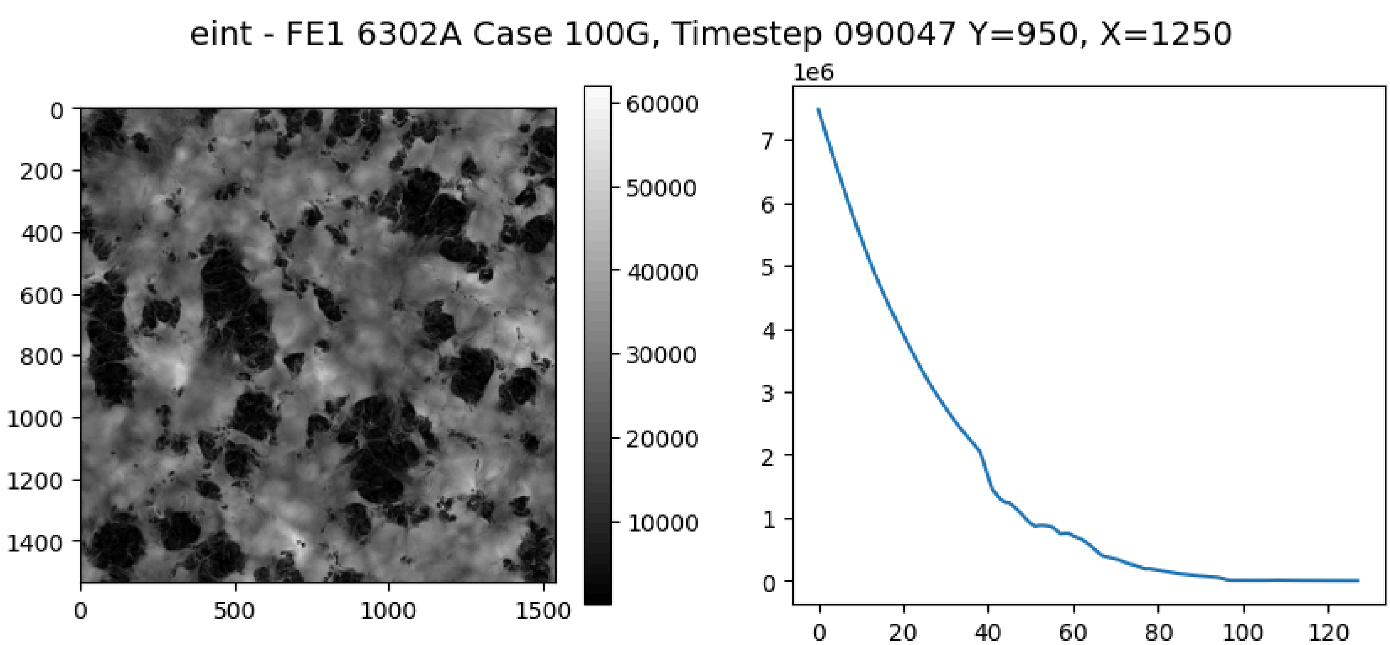
Number of electrons, internal electron pressure

In [40]:
```
cube['ne'] = read_cube('ne', DATA_MODEL, DATA_STEP)
cube['ne'].shape

title='ne - FE1 6302A Case {2}, Timestep {3} Y={0}, X={1}'.format(position[0],position[1],DATA_M(

plt.rcParams['figure.figsize'] = [10, 4]
plt.rcParams['figure.dpi'] = 100 # 200 e.g. is really fine, but slower
fig, axs = plt.subplots(1, 2)
fig.suptitle(title, fontsize=14)

plt.subplot(1, 2, 1)
plt.imshow(cube['ne'][:,:,40],cmap="gray")
plt.colorbar()
plt.subplot(1, 2, 2)
plt.plot(cube['ne'][position[0],position[1],:])
cube['ne'] = []
```



In [41]:

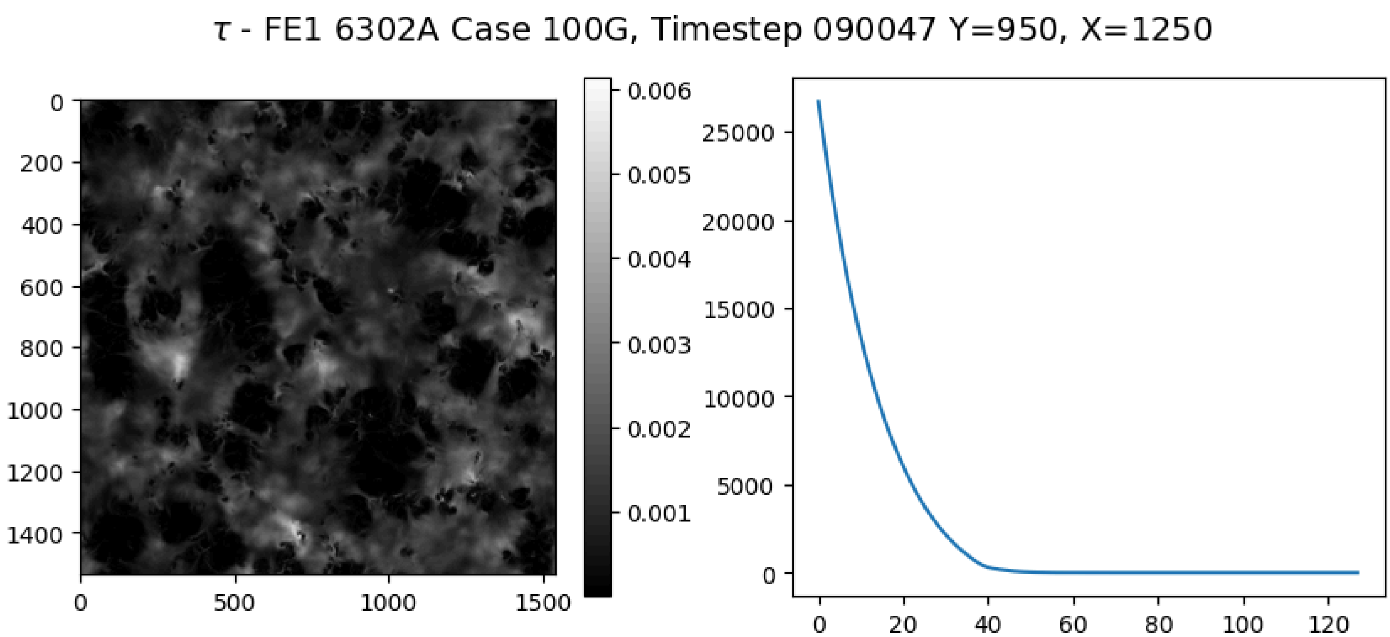
```
cube['eint'] = read_cube('eint', DATA_MODEL, DATA_STEP)
cube['eint'].shape

title='eint - FE1 6302A Case {2}, Timestep {3} Y={0}, X={1}'.format(position[0],position[1],DATA_

plt.rcParams['figure.figsize'] = [10, 4]
plt.rcParams['figure.dpi'] = 100 # 200 e.g. is really fine, but slower
fig, axs = plt.subplots(1, 2)
fig.suptitle(title, fontsize=14)

plt.subplot(1, 2, 1)
plt.imshow(cube['eint'][:,:,position[2]],cmap="gray")
plt.colorbar()
plt.subplot(1, 2, 2)
plt.plot(cube['eint'][position[0],position[1],:])
cube['eint'] = []
```



Opacity

By convention opacity (tau) is integrated at 500nm

In [42]:
```
cube['tau'] = read_cube('tau', DATA_MODEL, DATA_STEP)
cube['tau'].shape

title=r'$\tau$ - FE1 6302A Case {2}, Timestep {3} Y={0}, X={1}'.format(position[0],position[1],D

plt.rcParams['figure.figsize'] = [10, 4]
plt.rcParams['figure.dpi'] = 100 # 200 e.g. is really fine, but slower
fig, axs = plt.subplots(1, 2)
fig.suptitle(title, fontsize=14)

plt.subplot(1, 2, 1)
plt.imshow(cube['tau'][:,:,position[2]],cmap="gray")
plt.colorbar()
plt.subplot(1, 2, 2)
plt.plot(cube['tau'][position[0],position[1],:])
cube['tau'] = []
```



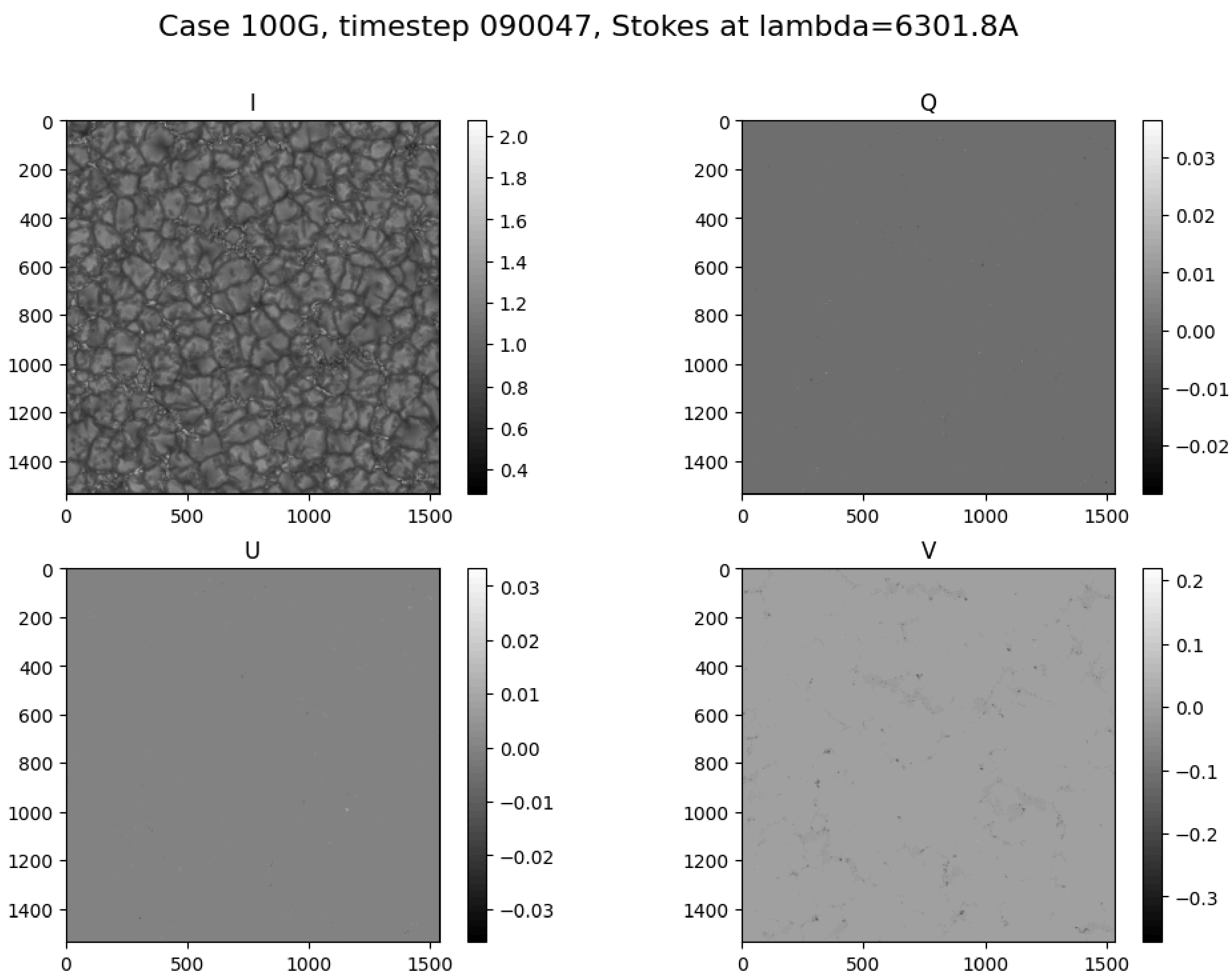
Visualizing Stokes Profiles

We can display I, Q, U, and V as images as if we were looking from Earth toward the center of the simulation cube. We observe the polarized light as it leaves the surface of last scattering. We choose a wavelength index of 200 (from the valid range 0 to 255) to avoid the polarized wavelength bands with absorption. This makes the (Q, U, V) plots featureless (low/no polarization). The X axis is east/west on the surface of the Sun. The Y axis is north/south on the surface of the Sun.

In [43]:
```
colormap="gray"
plt.rcParams['figure.figsize'] = [12, 8]
plt.rcParams['figure.dpi'] = 100 # 200 e.g. is really fine, but slower

fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2)
fig.suptitle('Case %s, timestep %s, Stokes at lambda=%.1fA' %(DATA_MODEL, DATA_STEP, wave[wavele

plt.subplot(2, 2, 1)
```

```
plt.imshow(si[:,:,200],cmap=colormap)
plt.title('I')
plt.colorbar()

plt.subplot(2, 2, 2)
plt.imshow(sq[:,:,200],cmap=colormap)
plt.title('Q')
plt.colorbar()

plt.subplot(2, 2, 3)
plt.imshow(su[:,:,200],cmap=colormap)
plt.title('U')
plt.colorbar()

plt.subplot(2, 2, 4)
plt.imshow(sv[:,:,200],cmap=colormap)
plt.title('V')
plt.colorbar()
```

Out[43]: `<matplotlib.colorbar.Colorbar at 0x7f7c6d7eacb0>`



Quiet Sun and Granules

The Intensity map shows what one might see through a telescope. There are visible granules, convective cells in the Sun's atmosphere.

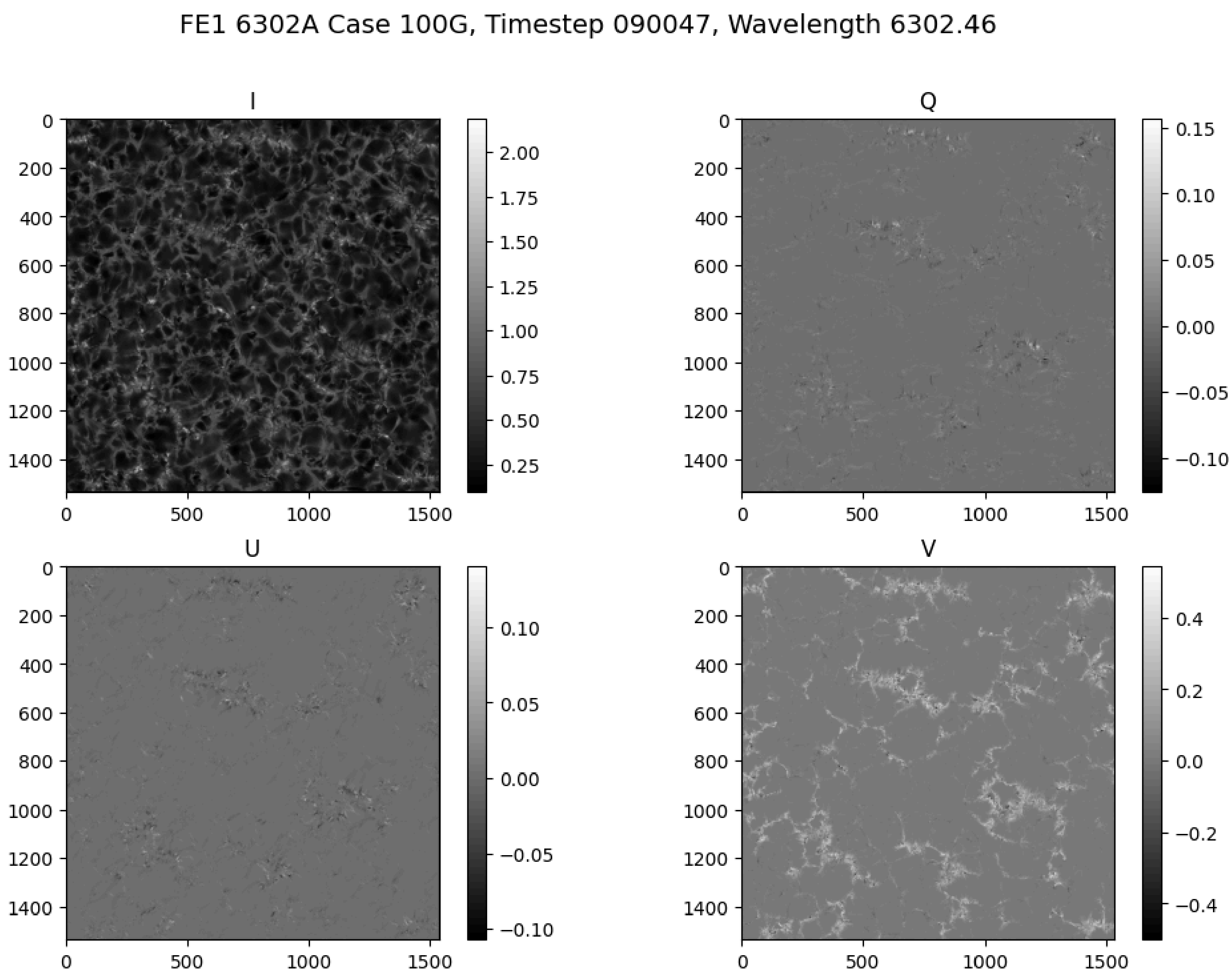
Other features may be observed. The Q, U and V plots above showed effectively no polarized light. Note the intensity scale shown on the colorbar at the right for Q, U and V. The V signal is about 1000x less than I. U and V are about 100,000x less than I! We see that Q, U and V have very low signal at the wavelength that we have plotted (intentionally).

What could we do to plot Q, U and V images that show a stronger signal?

Hint: we selected the wavelength bin 100 above but there are other wavelengths we could choose.

In [44]:
```
wavelen=170
fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2)

title='FE1 6302A Case {1}, Timestep {2}, Wavelength {0:.2f}'.format(wave[wavelen],DATA_MODEL, DA
fig.suptitle(title, fontsize=14)

plt.subplot(2, 2, 1)
plt.imshow(si[:,:,wavelen],cmap=colormap)
plt.title('I')
plt.colorbar()

plt.subplot(2, 2, 2)
plt.imshow(sq[:,:,wavelen],cmap=colormap)
plt.title('Q')
plt.colorbar()

plt.subplot(2, 2, 3)
plt.imshow(su[:,:,wavelen],cmap=colormap)
plt.title('U')
plt.colorbar()

plt.subplot(2, 2, 4)
plt.imshow(sv[:,:,wavelen],cmap=colormap)
plt.title('V')
plt.colorbar()
```

Out[44]:
```
<matplotlib.colorbar.Colorbar at 0x7f7c7c3a8fa0>
```



Butterfly plots

We choose bin 170 because it samples a wavelength where the polarization signal is stronger.

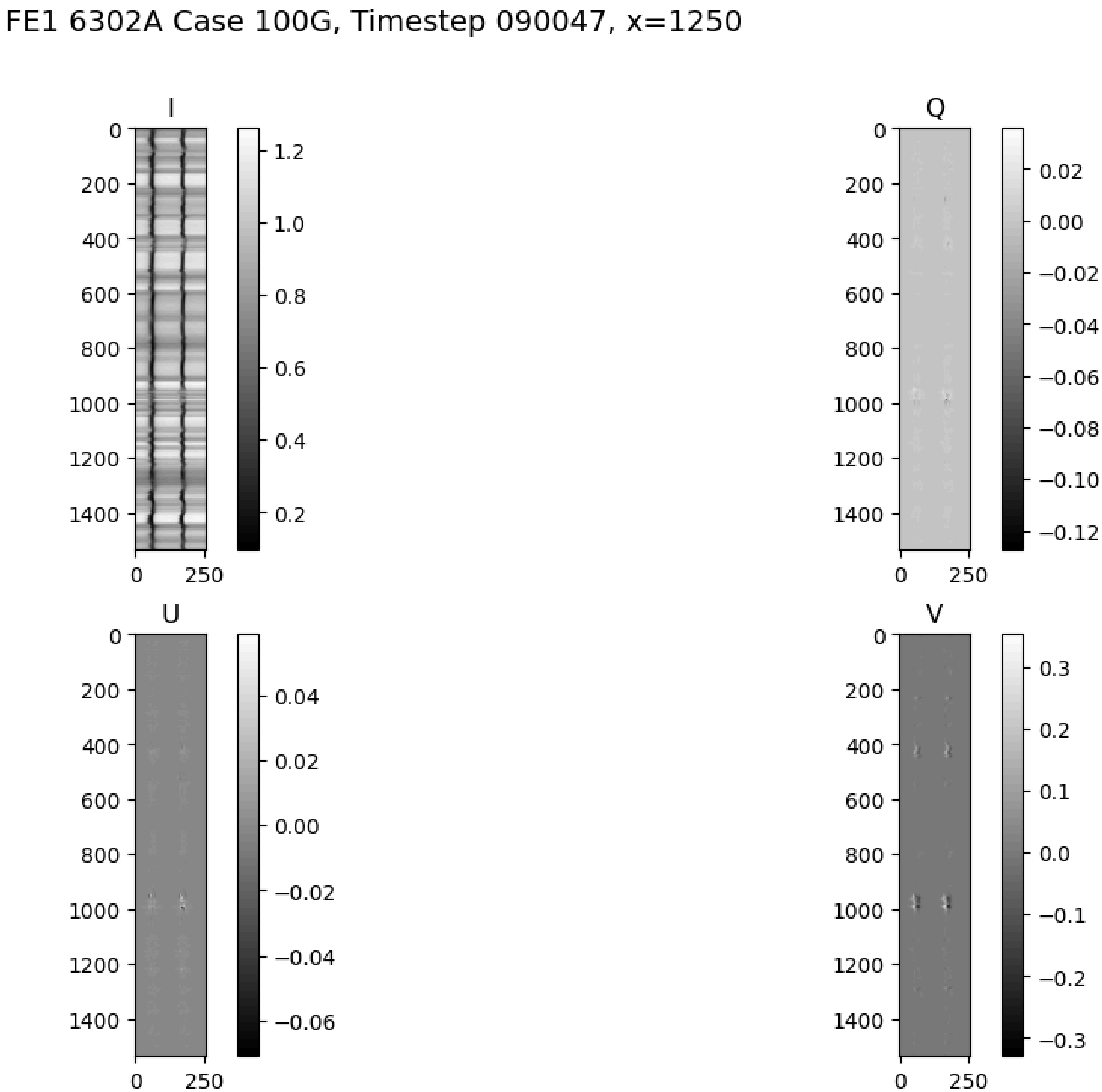
Let's shift our perspective and look along the wavelength dimension along a vertical stripe of the maps above.

In [45]:

```
fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2)

title='FE1 6302A Case {2}, Timestep {3}, x={1}'.format(0,position[1],DATA_MODEL, DATA_STEP)
fig.suptitle(title, fontsize=14)

plt.subplot(2, 2, 1)
plt.imshow(si[:,position[1],:],cmap=colormap)
plt.title('I')
plt.colorbar()

plt.subplot(2, 2, 2)
plt.imshow(sq[:,position[1],:],cmap=colormap)
plt.title('Q')
plt.colorbar()

plt.subplot(2, 2, 3)
plt.imshow(su[:,position[1],:],cmap=colormap)
```

```python
plt.title('U')
plt.colorbar()

plt.subplot(2, 2, 4)
plt.imshow(sv[:,position[1],:],cmap=colormap)
plt.title('V')
plt.colorbar()
```

Out[45]: <matplotlib.colorbar.Colorbar at 0x7f7c6c75bbb0>



Spatial interpretation

Since this is a vertical slice (x=1250) through the spatial map you can see that only certain regions of y (around 400 and 1000) have a strong signal in the Q, U and V plots.

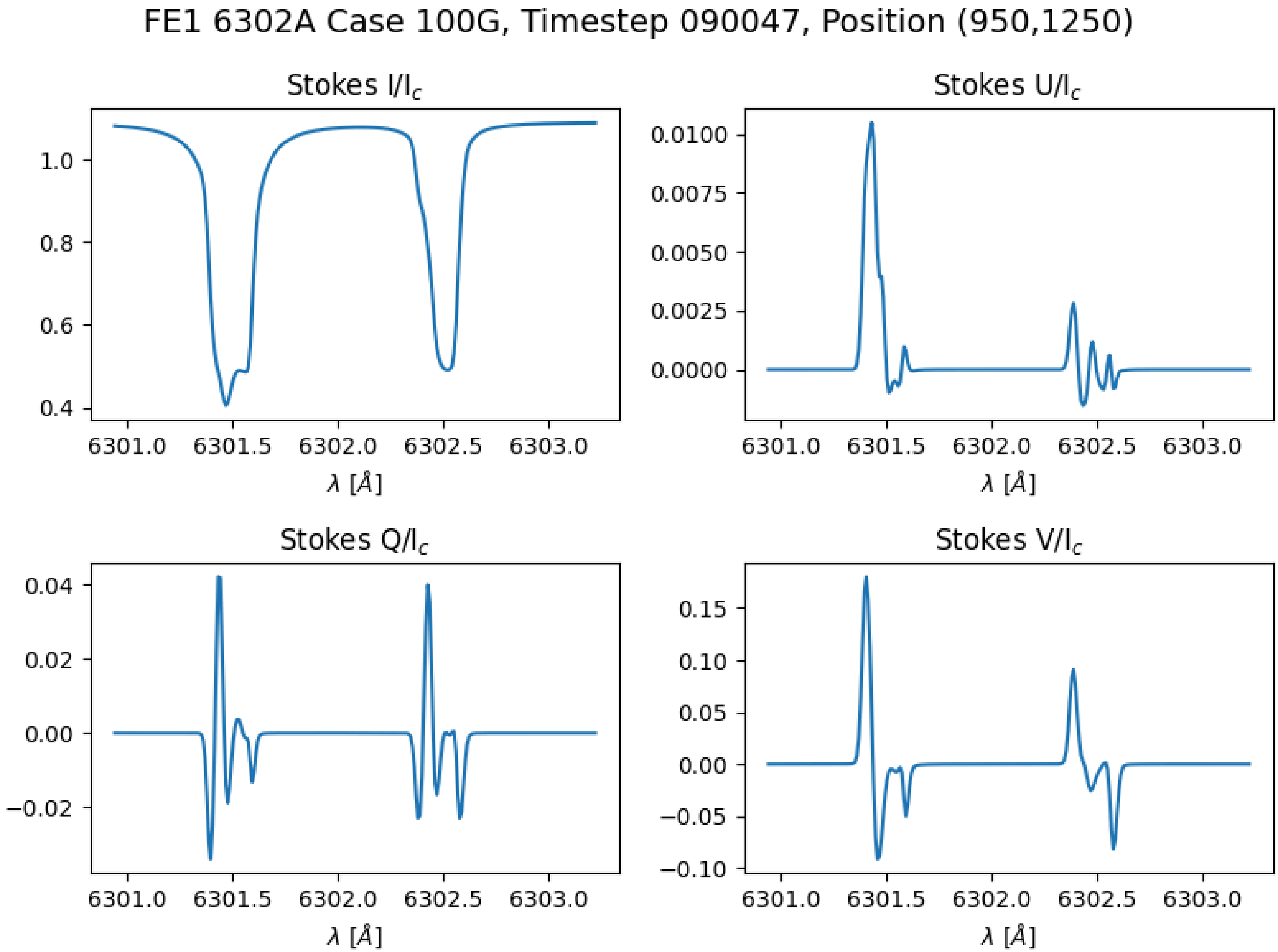
Also, note that the I plot shows the absorption lines around lambda=(60,170) but polarization signals are present only in specific spatial regions.

Stokes Profiles

Now we will take yet another perspective and plot intensity vs wavelength for a single spatial "pixel" or line of sight. This could be done for every 2.3M in each timestep. Using the butterfly plot above we will choose the position (950,1250) where we see a stronger polarization signal.

In [46]:
```
StokesI = si[position[0], position[1]]
StokesQ = sq[position[0],position[1]]
StokesU = su[position[0],position[1]]
StokesV = sv[position[0],position[1]]
```

In [47]:
```
title='FE1 6302A Case {2}, Timestep {3}, Position ({0},{1})'.format(position[0],position[1],DATA_

plt.rcParams['figure.figsize'] = [8, 6]
plt.rcParams['figure.dpi'] = 100
fig, axs = plt.subplots(2, 2)

fig.suptitle(title, fontsize=14)

im = axs[0, 0].plot(wave, StokesI)
axs[0, 0].set(xlabel='$\lambda$ [$\AA$]',title='Stokes I/I$_c$')

im = axs[0, 1].plot(wave, StokesQ)
axs[0, 1].set(xlabel='$\lambda$ [$\AA$]',title='Stokes U/I$_c$')

im = axs[1, 0].plot(wave, StokesU)
axs[1, 0].set(xlabel='$\lambda$ [$\AA$]',title='Stokes Q/I$_c$')

im = axs[1, 1].plot(wave, StokesV)
axs[1, 1].set(xlabel='$\lambda$ [$\AA$]',title='Stokes V/I$_c$')

plt.tight_layout()
plt.show()
```


FE1 6302A Case 100G, Timestep 090047, Position (950,1250)
Stokes I/I$_c$
Stokes U/I$_c$
Stokes Q/I$_c$
Stokes V/I$_c$
λ [Å]
6301.0
6301.5
6302.0
6302.5
6303.0

In [ ]: