ISSN: 2454-1907
DOI: https://doi.org/10.29121/ijetmr.v2.i1.2015.28

IJETMR
International Journal of Engineering Technologies and Management Research
A knowledge Repository

FUTURE FOR SCIENTIFIC COMPUTING USING PYTHON

Rakesh Kumar*[1]
[*1]Software Engineer at DigiCollect GIS, Bangalore, INDIA

Abstract:
Computational science (scientific computing or scientific computation) is concerned with constructing mathematical models as well as quantitative analysis techniques and using computers to analyze as well as solve scientific problems. In practical use, it is basically the application of computer simulation as well as other forms of computation from numerical analysis and theoretical computer science to problems in different scientific disciplines. The scientific computing approach is to gain understanding, basically through the analysis of mathematical models implemented on computers. Python is frequently used for high-performance scientific applications and widely used in academia as well as scientific projects because it is easy to write and performs well. Due to its high performance nature, scientific computing in Python often utilizes external libraries like NumPy, SciPy and Matplotlib etc.

Keywords:
IPython, Matplotlib, NumPy, Python, Pandas, Scientific Computing, Sympy, SciPy.

Cite This Article: Rakesh Kumar, "FUTURE FOR SCIENTIFIC COMPUTING USING PYTHON." *International Journal of Engineering Technologies and Management Research,* Vol. 2, No. 1(2015): 30-41. DOI: https://doi.org/10.29121/ijetmr.v2.i1.2015.28.

1. INTRODUCTION

Science has basically been divided into theoretical and experimental disciplines, but during the last several decades computing has emerged as an important part of science. Scientific computing is closely based to theory, but it has many characteristics in common with experimental work. It is therefore often viewed as a new third branch of science. In most fields of science, computational work is a very important complement to both theory as well as experiments, and nowadays a large majority of both theoretical and experimental papers involve some numerical calculations, computer modeling or simulations. In theoretical and experimental sciences there are good established codes of conducts for how methods as well as results are published and made available to other scientists. For example, in theoretical sciences, derivations, proofs as well as results are published in full detail and in experimental sciences, the methods used as well as results are published. In computational sciences there are not yet any well established guidelines for how source code as well as generated data should be handled. For example, it is relatively rare that source code used in simulations for published papers are provided to readers, in contrast to the open nature of theoretical and experimental work. It is not uncommon that source code for simulation software is withheld as well as considered a competitive advantage. So, this problem has recently started to attract increasing attention, as well as a number of editorials in high-profile journals have called for increased openness in computational sciences. Some journals, including Science, have even started to demand of authors to provide the source code for simulation software used in publications to readers upon request.

Replication (An author of a scientific paper that involves numerical calculations should be able to rerun the simulations as well as replicate the results upon request and other scientist should also be able to perform the same calculations as well as obtain the same results, given the information about the methods used in a publication.) and reproducibility (The results obtained from numerical simulations should be reproducible with an independent implementation of the method.) are two of the cornerstones in the scientific method. In short summary, a sound scientific result should be reproducible, as well as a sound scientific study should be replicable.

Ensuring replicability as well as reproducibility of scientific simulations is a complicated problem, but there are some good tools to help with this such as Revision Control System (RCS) software (git, mercurial (hg) and subversion (svn)) as well as online repositories for source code, that available as both public and private repositories (GitHub, Bit bucket and Privately hosted repositories).

Python is a modern, general-purpose, object-oriented, high-level programming language that includes some general characteristics like clean and simple language, expressive language, dynamically typed, automatic memory management and interpreted. The main advantage is ease of programming, minimizing the time required to develop, debug, maintain the code, modular, object-oriented programming, good system for packaging, re-use of code, documentation tightly integrated with the code, large standard library, and a large collection of add-on packages.

Python has a strong position in scientific computing because it contains extensive ecosystem of scientific libraries as well as environments like numpy, scipy, matplotlib etc. Python supports for parallel processing with processes and threads, Interprocess communication as well as GPU computing and suitable for use on high-performance computing clusters.

***Reason for using python in scientific computing*:**

- Clear, readable syntax through whitespace indentation
- Interactive python console
- Strong introspection capabilities
- Full modularity, supporting hierarchical packages
- Exception-based error handling
- Dynamic data types & automatic memory management

***Reason for using python over matlab*:**

- Python packages **can do similarly everything Matlab can do for signal processing**
- Python is free and open source but Matlab is a closed-source commercial product (algorithms are proprietary)
- Python is a **general-purpose language** but Matlab is purely for scientific computing
- Python is **high level, easy to learn, and** so many **cool features**

- The **demand for Python programmers is increasing**
- Matlab is quite **expensive**, that means that code that is written in Matlab can only be used by people with sufficient funds to buy a license
- Python contains great libraries, and yet more external libraries are being developed by Python programmers, scientists, mathematicians, and engineers
- Similar every programming language other than Matlab, Python uses zero-based indexing
- Python code tends to be more compact as well as more readable than Matlab code
- OOP in Python is simple and elegant, that offers flexibility but Matlab's OOP scheme is complex and confusing
- Python offers a wider set of choices in graphics packages and toolsets as well as easily integrates better with other languages

2. METHODOLOGY

SciPy (pronounced "Sigh Pie") Stack is an open source Python library that contains modules for optimization, integration, interpolation, FFT, special functions, linear algebra, signal and image processing and used by scientists, analysts, and engineers doing scientific computing. SciPy library is distributed under the BSD license, and its development is sponsored as well as supported by an open community of developers, it is also supported by Numfocus that is a community foundation for supporting reproducible as well as accessible science.

***The SciPy Stack*:**

a. Core packages

- *NumPy* : NumPy is the basic package for scientific computing in Python that provides multi-dimensional arrays and matrices, broadcasting functions, tools for integrating C/C++, Fortran code, mathematical, logical, shape manipulation, sorting, selecting, I/O, useful linear algebra, Fourier transform, random number capabilities, basic statistical operations and much more. NumPy can be used as an efficient multi-dimensional container of generic data and it is licensed under the BSD license.

- *SciPy library* : SciPy library is one of the core packages used by scientists, analysts, and engineers that provides many user-friendly and efficient numerical routines such as routines for numerical integration, optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing. SciPy builds on the NumPy array object as well as is part of the NumPy stack that includes tools like Matplotlib, pandas and SymPy. The SciPy library is currently distributed under the BSD license, as well as its development is supported and sponsored by an open community of developers. It is also supported by Numfocus that is a community foundation for supporting reproducible and accessible science.

- *Matplotlib* : Matplotlib is a 2D plotting library for the Python programming language that produces publication quality figures in a variety of hardcopy formats as well as interactive environments across platforms. Matplotlib tries to make easy things easy as well as hard things possible and provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits such as wxPython, Qt, or GTK+. Matplotlib was mainly written by John D. Hunter, has an active development community, and is distributed under a BSD-style license.

- *Pandas* : Pandas is a software library written for the Python programming language that providing high-performance, data manipulation, easy-to-use data structures and data analysis tools for the Python programming language as well as it is free software released under the three-clause BSD license.

- *SymPy* : SymPy is an important Python library for symbolic computation that provides computer algebra system (CAS) while keeping the code as simple as possible in order to be comprehensible as well as easily extensible. SymPy is written fully in Python and does not require any external libraries. SymPy contains features ranging from basic symbolic arithmetic to algebra, calculus, discrete mathematics and quantum physics and it is capable of formatting the result of the computations as LaTeX code. SymPy is free software as well as is licensed under New BSD license.

- *IPython* : IPython is a command shell for interactive computing in many programming languages, mainly developed for the Python programming language that offers enhanced introspection, rich media, more shell syntax, tab completion, as well as rich history. IPython provides an important features such as Powerful interactive shells, browser-based notebook, inline plots, Support for interactive data visualization, use of GUI toolkits, embeddable interpreters, high performance tools for parallel computing.

- *Nose* : Nose is a unit test frameworks in Python that developed by Jason Pellerin. It is fairly stable, fantastic plug-in architecture, integrates with distutils, adapted to mimic any other unit test discovery framework.

b. Other packages

- *Chaco* : Chaco (Python plotting application toolkit) facilitates writing plotting applications at all levels of complexity, from simple scripts with hard-coded data to large plotting programs with complex data interrelationships as well as a multitude of interactive tools. Chaco generates attractive static plots for publication as well as presentation, it works well for interactive data visualization as well as exploration.

- *Mayavi* : Mayavi provide interactive as well as easy visualization of 3D data that offers a rich user interface with dialogs to interact with all data as well as objects in the visualization. Mayavi basically integrates seamlessly with numpy and scipy for 3D plotting and can even be used in IPython interactively, similarly to Matplotlib. Mayavi is a reusable tool that can be combined with the Envisage application-building framework to assemble domain-specific tools.

- *Cython* : The Cython language is a superset of the **Python** language that supports calling **C functions** as well as declaring **C types** on variables and class attributes. PyPy support is work in progress as well as is considered mostly usable since Cython 0.17. All of this makes Cython the ideal language for **wrapping** external C libraries, **embedding** CPython into existing applications, as well as for **fast C modules** that speed up the execution of Python code.

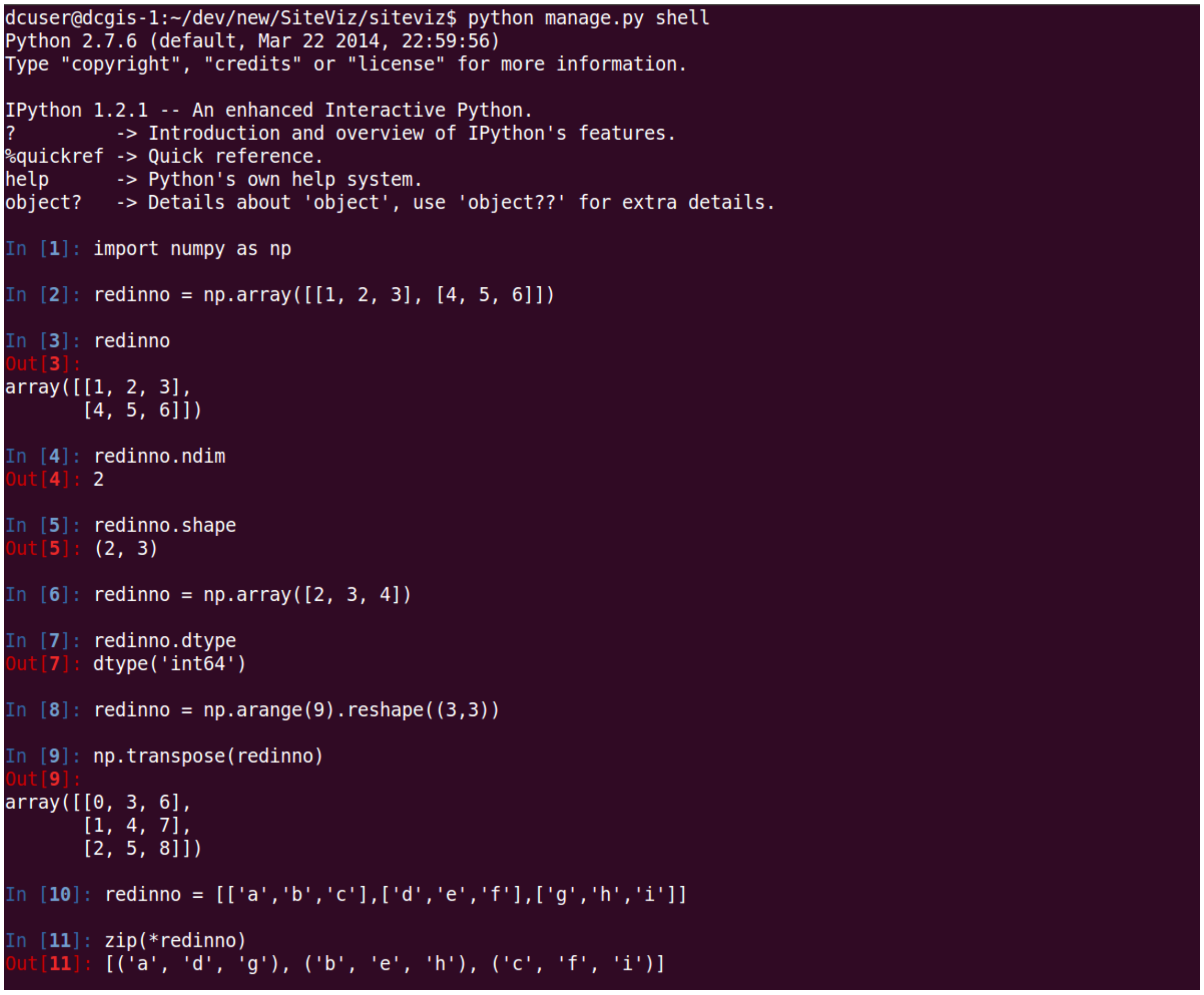
- *Scikits* : SciKits (SciPy Toolkits), are add-on packages for SciPy, hosted as well as developed separately from the main SciPy distribution. All SciKits are available under the 'scikits' namespace as well as are licensed under OSI-approved licenses.

- *h5py* : The h5py package is a Pythonic interface to the HDF5 binary data format that uses straightforward NumPy as well as Python metaphors, such as dictionary and NumPy array syntax. In addition to the easy-to-use high level interface, h5py rests on an object-oriented Cython wrapping of the HDF5 C API and almost anything you can do from C in HDF5, you can do from h5py.

- *PyTables* : PyTables is a package for managing hierarchical datasets as well as designed to efficiently and easily cope with extremely large amounts of data. PyTables is built on top of the HDF5 library, that using Python language and the NumPy package. It features an object-oriented interface that, combined with C extensions for the performance-critical parts of the code, makes it a fast, so extremely easy to use tool for interactively browse, process as well as search large amounts of data. PyTables optimizes memory as well as disk resources so that data takes much less space than other solutions like relational or object oriented databases.

3. RESULTS

NumPy is an important Python extension module that provides easily efficient operation on arrays of homogeneous data. NumPy allows python to serve as a high-level language for manipulating numerical data, much such as IDL, MATLAB, or Yorick.

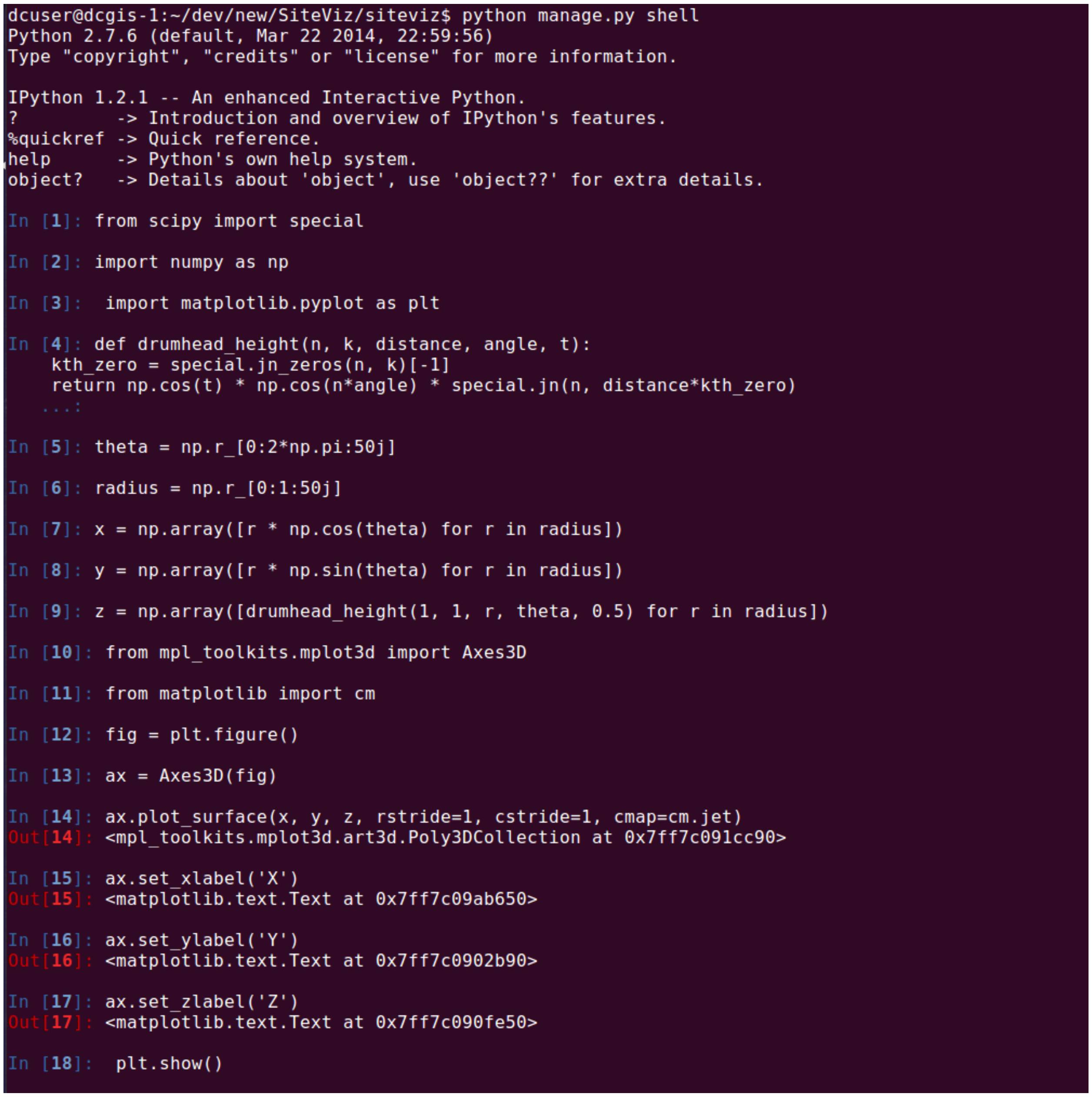


```
dcuser@dcgis-1:~/dev/new/SiteViz/siteviz$ python manage.py shell
Python 2.7.6 (default, Mar 22 2014, 22:59:56)
Type "copyright", "credits" or "license" for more information.

IPython 1.2.1 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: import numpy as np

In [2]: redinno = np.array([[1, 2, 3], [4, 5, 6]])

In [3]: redinno
Out[3]:
array([[1, 2, 3],
       [4, 5, 6]])

In [4]: redinno.ndim
Out[4]: 2

In [5]: redinno.shape
Out[5]: (2, 3)

In [6]: redinno = np.array([2, 3, 4])

In [7]: redinno.dtype
Out[7]: dtype('int64')

In [8]: redinno = np.arange(9).reshape((3,3))

In [9]: np.transpose(redinno)
Out[9]:
array([[0, 3, 6],
       [1, 4, 7],
       [2, 5, 8]])

In [10]: redinno = [['a','b','c'],['d','e','f'],['g','h','i']]

In [11]: zip(*redinno)
Out[11]: [('a', 'd', 'g'), ('b', 'e', 'h'), ('c', 'f', 'i')]
```

Figure 1: Example of Using NumPy Package

SciPy is a set of open source (BSD licensed) scientific as well as numerical tools for Python that supports special functions, integration, ordinary differential equation solvers, gradient optimization, parallel programming tools, an expression-to-C++ compiler for fast execution, etc.

NumPy would contain nothing but the array data type as well as the most operations like indexing, sorting, reshaping, basic element wise functions, etc. and all numerical code would reside in SciPy. One of NumPy's very important goals is compatibility, which tries to retain all features supported by either of its predecessors. NumPy contains few linear algebra functions, even though these more properly belong in SciPy. SciPy contains more fully-featured versions of the linear algebra modules, and many other numerical algorithms. If you are doing scientific computing using python, then you should probably install both NumPy as well as SciPy. Most new and important features belong in SciPy rather than NumPy.



```
dcuser@dcgis-1:~/dev/new/SiteViz/siteviz$ python manage.py shell
Python 2.7.6 (default, Mar 22 2014, 22:59:56)
Type "copyright", "credits" or "license" for more information.

IPython 1.2.1 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: from scipy import special

In [2]: import numpy as np

In [3]:  import matplotlib.pyplot as plt

In [4]: def drumhead_height(n, k, distance, angle, t):
    kth_zero = special.jn_zeros(n, k)[-1]
    return np.cos(t) * np.cos(n*angle) * special.jn(n, distance*kth_zero)
   ...:

In [5]: theta = np.r_[0:2*np.pi:50j]

In [6]: radius = np.r_[0:1:50j]

In [7]: x = np.array([r * np.cos(theta) for r in radius])

In [8]: y = np.array([r * np.sin(theta) for r in radius])

In [9]: z = np.array([drumhead_height(1, 1, r, theta, 0.5) for r in radius])

In [10]: from mpl_toolkits.mplot3d import Axes3D

In [11]: from matplotlib import cm

In [12]: fig = plt.figure()

In [13]: ax = Axes3D(fig)

In [14]: ax.plot_surface(x, y, z, rstride=1, cstride=1, cmap=cm.jet)
Out[14]: <mpl_toolkits.mplot3d.art3d.Poly3DCollection at 0x7ff7c091cc90>

In [15]: ax.set_xlabel('X')
Out[15]: <matplotlib.text.Text at 0x7ff7c09ab650>

In [16]: ax.set_ylabel('Y')
Out[16]: <matplotlib.text.Text at 0x7ff7c0902b90>

In [17]: ax.set_zlabel('Z')
Out[17]: <matplotlib.text.Text at 0x7ff7c090fe50>

In [18]:  plt.show()
```

Figure 2: Example of Using SciPy Library Package

Plotting functionality is beyond the scope of NumPy as well as SciPy that focus on numerical objects and algorithms. Many packages exist that integrate closely with NumPy to produce high quality plots, like the immensely popular Matplotlib and the extensible, modular toolkit Chaco.

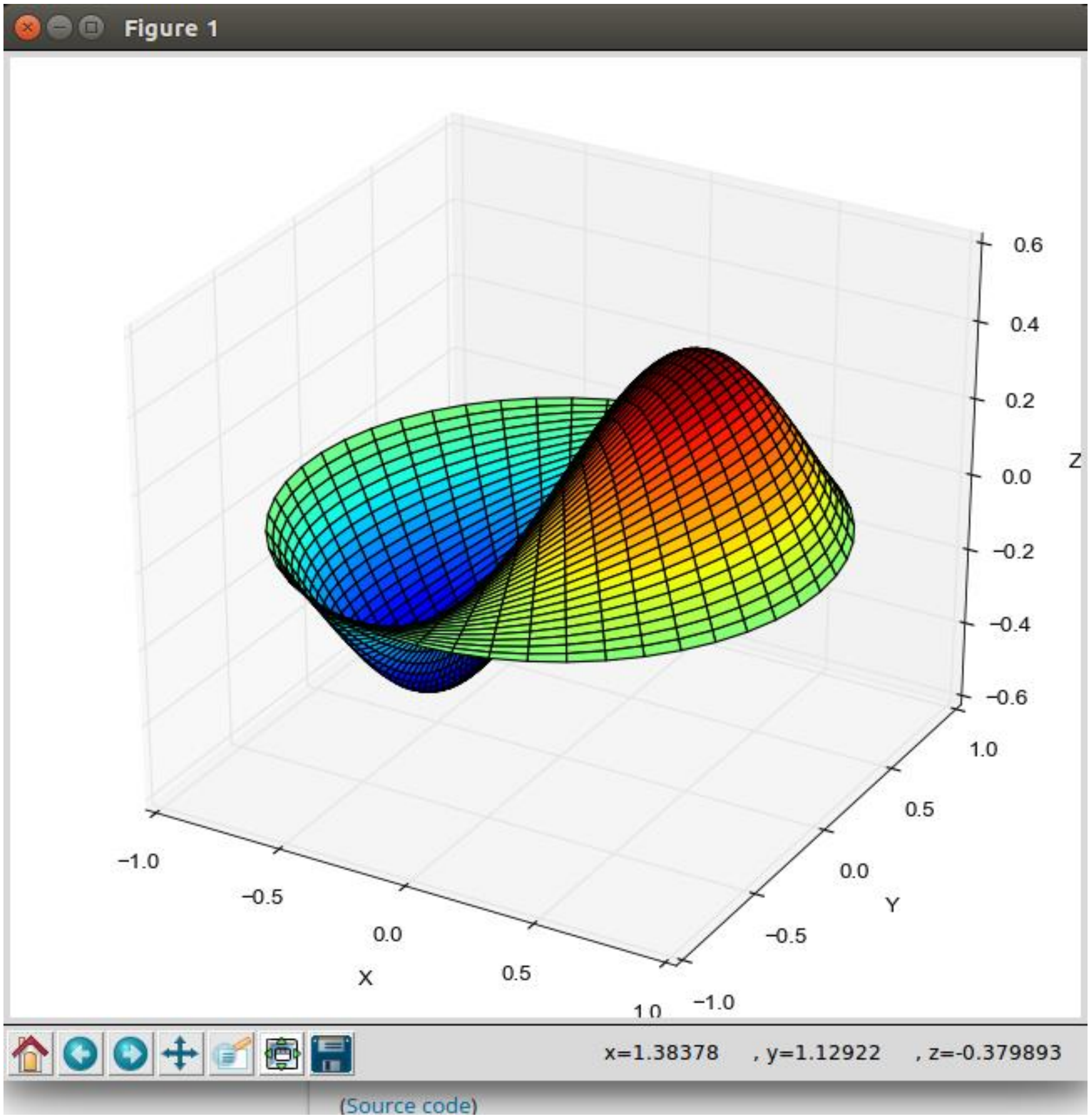
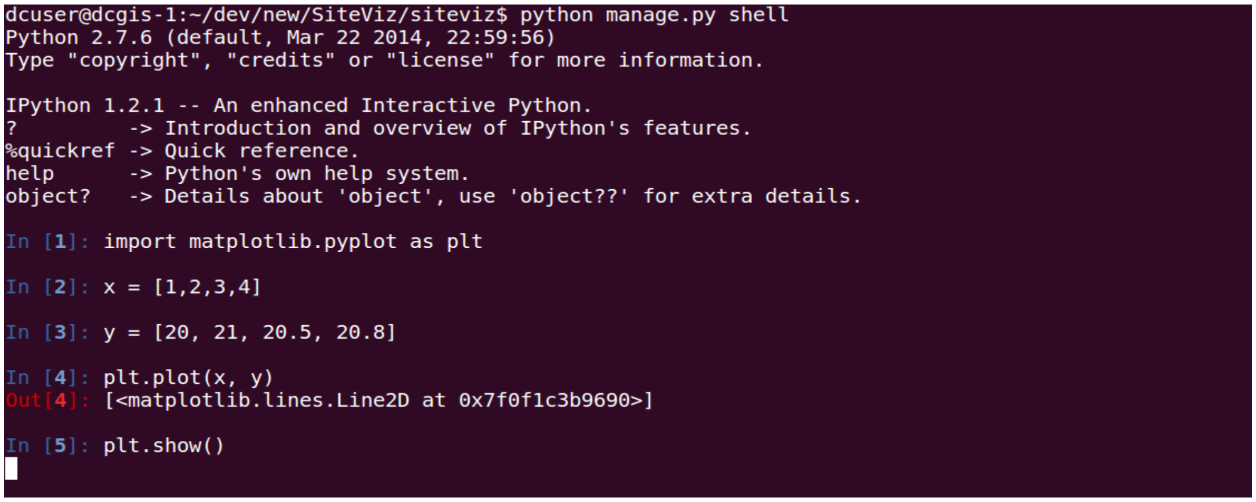


Figure 3: Result of Using SciPy Library Package

Matplotlib is a python plotting library for making publication quality plots using a syntax familiar to MATLAB users that uses numpy for numerics. Output formats contain PDF, Postscript, SVG, and PNG, and screen display.



```
dcuser@dcgis-1:~/dev/new/SiteViz/siteviz$ python manage.py shell
Python 2.7.6 (default, Mar 22 2014, 22:59:56)
Type "copyright", "credits" or "license" for more information.

IPython 1.2.1 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: import matplotlib.pyplot as plt

In [2]: x = [1,2,3,4]

In [3]: y = [20, 21, 20.5, 20.8]

In [4]: plt.plot(x, y)
Out[4]: [<matplotlib.lines.Line2D at 0x7f0f1c3b9690>]

In [5]: plt.show()
```

Figure 4: Example of Using Matplotlib Package

Matplotlib provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits such as wxPython, Qt, orGTK+.

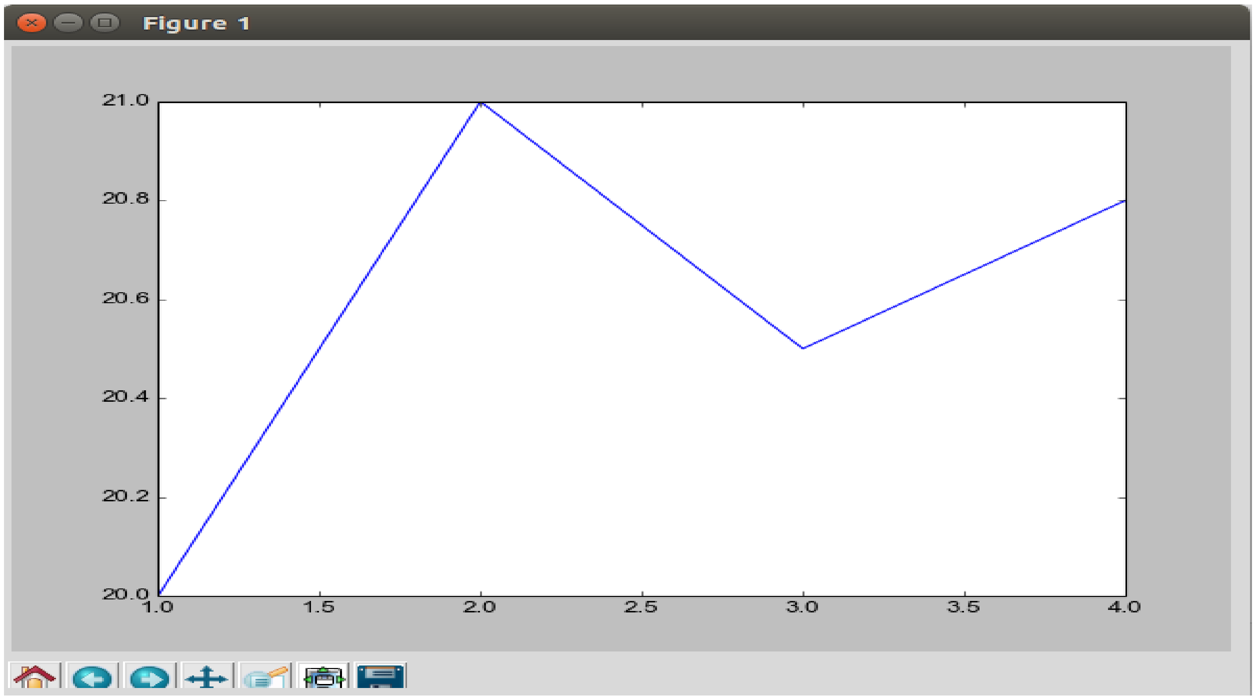
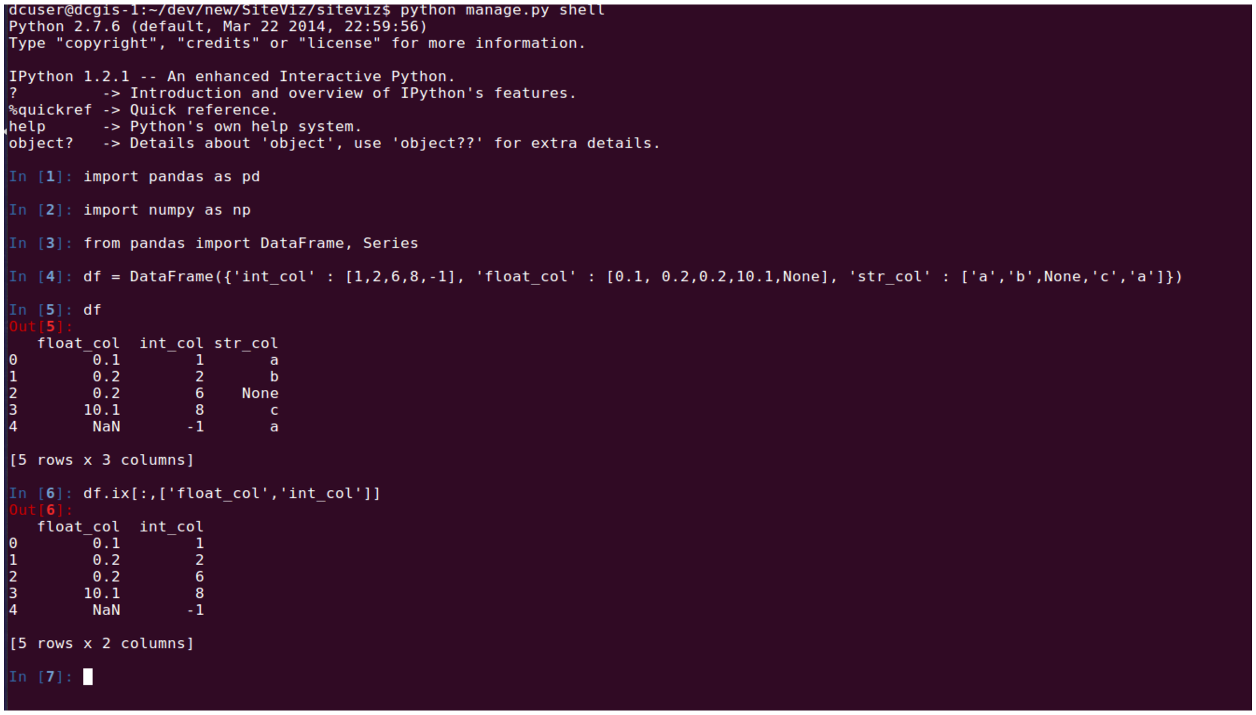


Figure 5: Result of Using Matplotlib Package

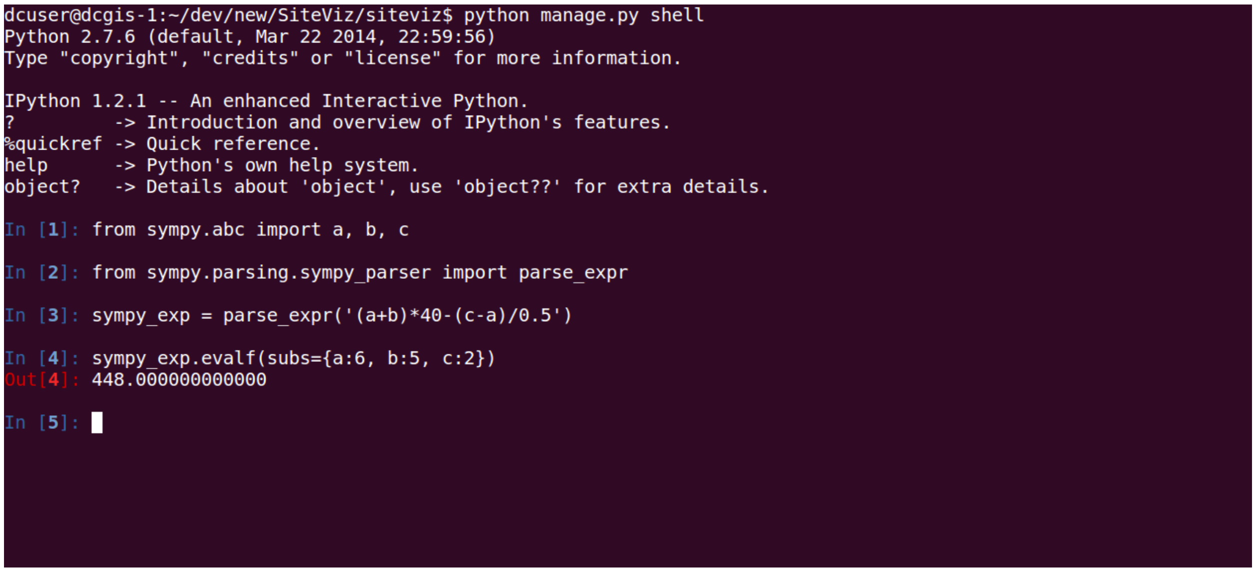
Pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.



```
dcuser@dcgis-1:~/dev/new/SiteViz/siteviz$ python manage.py shell
Python 2.7.6 (default, Mar 22 2014, 22:59:56)
Type "copyright", "credits" or "license" for more information.

IPython 1.2.1 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: import pandas as pd

In [2]: import numpy as np

In [3]: from pandas import DataFrame, Series

In [4]: df = DataFrame({'int_col' : [1,2,6,8,-1], 'float_col' : [0.1, 0.2,0.2,10.1,None], 'str_col' : ['a','b',None,'c','a']})

In [5]: df
Out[5]:
   float_col  int_col str_col
0        0.1        1       a
1        0.2        2       b
2        0.2        6    None
3       10.1        8       c
4        NaN       -1       a

[5 rows x 3 columns]

In [6]: df.ix[:,['float_col','int_col']]
Out[6]:
   float_col  int_col
0        0.1        1
1        0.2        2
2        0.2        6
3       10.1        8
4        NaN       -1

[5 rows x 2 columns]

In [7]: 
```

Figure 6: Example of Using Pandas Package

SymPy is an important Python library for symbolic mathematics that aims to become a full-featured computer algebra system (CAS) while keeping the code as simple as possible in order to be comprehensible as well as extensible. SymPy is written in Python that does not require any external libraries, except optionally for plotting support.



```
dcuser@dcgis-1:~/dev/new/SiteViz/siteviz$ python manage.py shell
Python 2.7.6 (default, Mar 22 2014, 22:59:56)
Type "copyright", "credits" or "license" for more information.

IPython 1.2.1 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: from sympy.abc import a, b, c

In [2]: from sympy.parsing.sympy_parser import parse_expr

In [3]: sympy_exp = parse_expr('(a+b)*40-(c-a)/0.5')

In [4]: sympy_exp.evalf(subs={a:6, b:5, c:2})
Out[4]: 448.000000000000

In [5]: 
```

Figure 7: Example of Using SymPy Package

IPython is a command shell for interactive computing in multiple programming languages, mainly developed for the Python programming language which offers enhanced introspection, rich media, additional shell syntax, tab completion, as well as rich history.

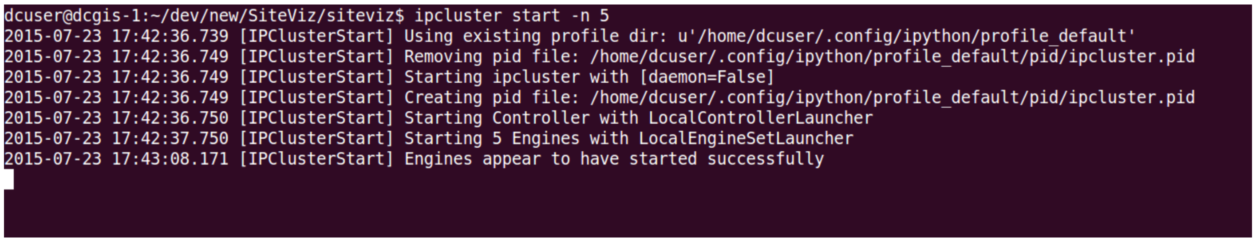

```
dcuser@dcgis-1:~/dev/new/SiteViz/siteviz$ ipcluster start -n 5
2015-07-23 17:42:36.739 [IPClusterStart] Using existing profile dir: u'/home/dcuser/.config/ipython/profile_default'
2015-07-23 17:42:36.749 [IPClusterStart] Removing pid file: /home/dcuser/.config/ipython/profile_default/pid/ipcluster.pid
2015-07-23 17:42:36.749 [IPClusterStart] Starting ipcluster with [daemon=False]
2015-07-23 17:42:36.749 [IPClusterStart] Creating pid file: /home/dcuser/.config/ipython/profile_default/pid/ipcluster.pid
2015-07-23 17:42:36.750 [IPClusterStart] Starting Controller with LocalControllerLauncher
2015-07-23 17:42:37.750 [IPClusterStart] Starting 5 Engines with LocalEngineSetLauncher
2015-07-23 17:43:08.171 [IPClusterStart] Engines appear to have started successfully
```

Figure 8: Example of Using IPython Package

IPython has a sophisticated architecture for parallel as well as distributed computing that enables parallel applications to be developed, executed, debugged and monitored interactively.

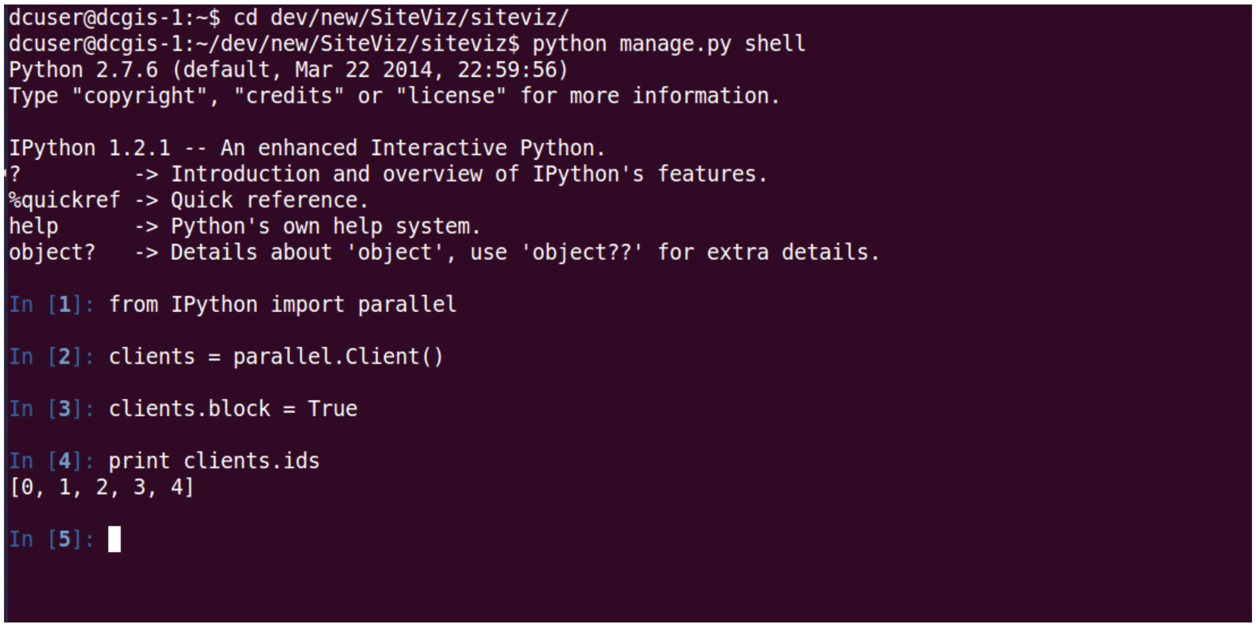

```
dcuser@dcgis-1:~$ cd dev/new/SiteViz/siteviz/
dcuser@dcgis-1:~/dev/new/SiteViz/siteviz$ python manage.py shell
Python 2.7.6 (default, Mar 22 2014, 22:59:56)
Type "copyright", "credits" or "license" for more information.

IPython 1.2.1 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: from IPython import parallel

In [2]: clients = parallel.Client()

In [3]: clients.block = True

In [4]: print clients.ids
[0, 1, 2, 3, 4]

In [5]:
```

Figure 9: Result of Using IPython Package

4. CONCLUSION

Now, a downside of using Python compared to Matlab is that is just more complicated to install. For example, on Windows, a Matlab user can just install the software, click on the icon, as well as a full graphical IDE opens. But in Python, there are few integrated scientific distributions with graphical IDEs but they are less convenient than Matlab. Using IPython involves either a command-line interface or a web interface. Installing external libraries can be a pain (it almost always involves the command-line interface that terrifies many Windows users). I think that's the main and important problem that prevents Python from widespread adoption in the Matlab community. Yet, I am pretty sure that Python will eventually become the platform of reference for scientific computing.

5. ACKNOWLEDGEMENTS

I wish to thank to Dr. Prof. K. K. Agrawal (Professor of JECRC, Jaipur), Ms. Neha Gupta (Assistant Professor of JECRC, Jaipur), Mr. Sunil Kumar Jangir (Assistant Professor of JECRC, Jaipur), Ms. Shilpi Charu (Senior Lecturer of JECRC, Jaipur), and all Faculty Member of JECRC Foundation, Jaipur for their time to time suggestions, guidelines, relevant data and technical support.

6. REFERENCES

[1] Python Scientific Lecture Notes: https://scipy-lectures.github.io/
[2] Scientific computing tools for python: http://www.scipy.org/
[3] NumPy: http://www.numpy.org/
[4] SciPy library: http://www.scipy.org/scipylib/
[5] Matplotlib: http://matplotlib.org/
[6] Pandas: http://pandas.pydata.org/
[7] SymPy: http://www.sympy.org/en/
[8] IPython: http://ipython.org/
[9] Nose: https://nose.readthedocs.org/en/latest/
[10] Chaco: http://code.enthought.com/projects/chaco/
[11] Mayavi: http://code.enthought.com/projects/mayavi/
[12] Cython: http://cython.org/
[13] Scikits: http://scikits.appspot.com/scikits
[14] h5py: http://www.h5py.org/
[15] PyTables: http://www.pytables.org/