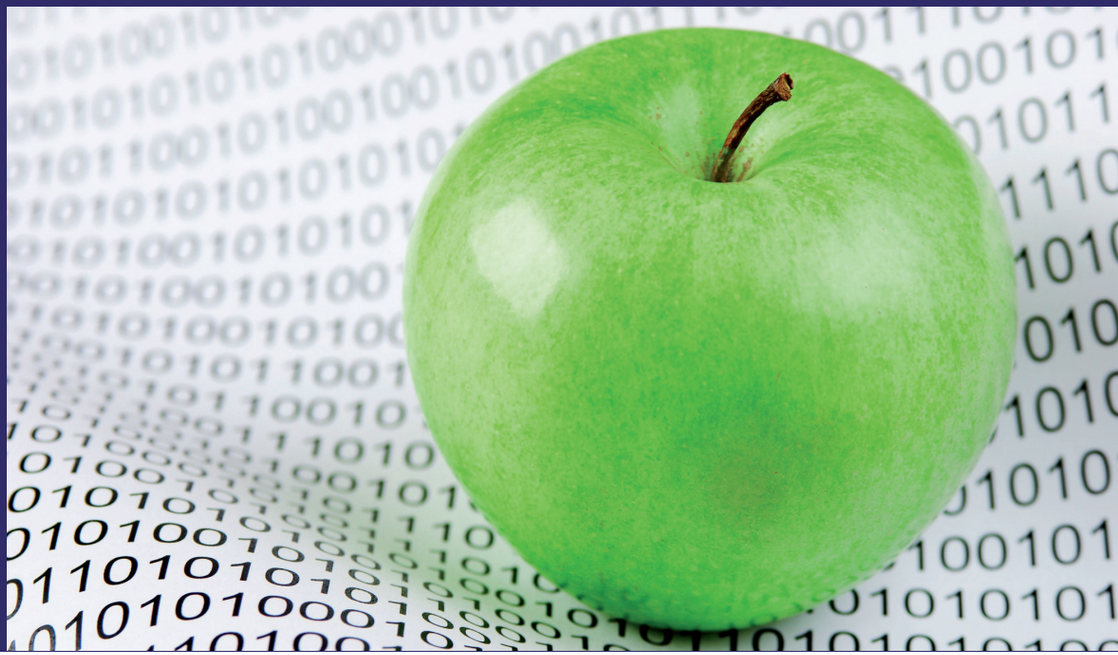
The book has been organized in two chapters: Design of Graphical User Interface (GUI) for NMAG and Design of modules for Magnetic Moment based Optimization in USPEX. Both chapters start with introduction and highlights. Then a detailed feasibility study is performed. Next section deals with design specifications, structured analysis, Data Flow Diagram, logical design, code development and at the end physical design together with different file formats. After this a special section is introduced to test the functionality of this newly designed code. Next section gives an overview of hardware and software requirements. A short section on future work is also provided after conclusion and benefits. At the end references are provided.

Ravi Agarwal, M.Tech. (Nanotechnology): Studied at Centre for Converging Technologies, University of Rajasthan, India. Newton Bhabha Fellow at Aberystwyth University.

978-3-659-96338-4



Ravi Agarwal
Narendra Kumar Agrawal

NMAG and USPEX: Add-On

Utility for Enhanced Functionality

LAP LAMBERT Academic Publishing

Ravi Agarwal
Narendra Kumar Agrawal

NMAG and USPEX: Add-On

Ravi Agarwal
Narendra Kumar Agrawal

NMAG and USPEX: Add-On

Utility for Enhanced Functionality

LAP LAMBERT Academic Publishing

Impressum / Imprint
Bibliografische Information der Deutschen Nationalbibliothek: Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.
Alle in diesem Buch genannten Marken und Produktnamen unterliegen warenzeichen-, marken- oder patentrechtlichem Schutz bzw. sind Warenzeichen oder eingetragene Warenzeichen der jeweiligen Inhaber. Die Wiedergabe von Marken, Produktnamen, Gebrauchsnamen, Handelsnamen, Warenbezeichnungen u.s.w. in diesem Werk berechtigt auch ohne besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne der Warenzeichen- und Markenschutzgesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürften.

Bibliographic information published by the Deutsche Nationalbibliothek: The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at http://dnb.d-nb.de.
Any brand names and product names mentioned in this book are subject to trademark, brand or patent protection and are trademarks or registered trademarks of their respective holders. The use of brand names, product names, common names, trade names, product descriptions etc. even without a particular marking in this work is in no way to be construed to mean that such names may be regarded as unrestricted in respect of trademark and brand protection legislation and could thus be used by anyone.

Coverbild / Cover image: www.ingimage.com

Verlag / Publisher:
LAP LAMBERT Academic Publishing
ist ein Imprint der / is a trademark of
OmniScriptum GmbH & Co. KG
Bahnhofstraße 28, 66111 Saarbrücken, Deutschland / Germany
Email: info@omniscriptum.com

Herstellung: siehe letzte Seite /
Printed at: see last page
ISBN: 978-3-659-96338-4

Zugl. / Approved by: Jaipur, University of Rajasthan, 2012

Copyright © Ravi Agarwal, Narendra Kumar Agrawal
Copyright © 2016 OmniScriptum GmbH & Co. KG
Alle Rechte vorbehalten. / All rights reserved. Saarbrücken 2016

Contents

List of Figures

List of Tables

Chapter 1: Design of Graphical User Interface for NMAG

1.1 Introduction

1.1.1 NMAG Overview

Nmag is a flexible finite element micro magnetic simulation package with user interface based on the Python programming language *[1.1]*. It has been developed at the University of Southampton with substantial contributions from Hans Fangohr, Thomas Fischbacher, Matteo Franchin, Giuliano Bordignon, Jacek Generowicz, Andreas Knittel, Michael Walter, and Maximilian Albert.

The main features of Nmag are:

- based on finite elements (suitable for non-cuboidal structures)
- problem description in Python, therefore high degree of flexibility
- inbuilt mesh post processing tools
- efficient data storage (binary compressed) and extraction into vtk files (of course the raw data can be extracted)
- arbitrary crystal anisotropy
- (pseudo) periodic boundary conditions ("macro geometry approach")
- Spin torque transfer (Zhang-Li model for uniform current density)
- Supports use of matrix compression library (HLib) for BEM

This software was developed at the University of Southampton, United Kingdom. It is released under the GNU General Public License (GNU GPL) as published by the Free Software Foundation.

1.1.1.1 NMAG Philosophy

Many specialized simulation codes used in research today consist of a highly specialized core application which initially was written to simulate the behaviour of some very specific system. Often, the core application then evolved into a more broadly applicable tool through the introduction of additional parameters. Some simulation codes reach a point where it becomes evident that they need an amount of flexibility that can only be provided by including some script programming capabilities *[1.2]*.

The approach underlying Nmag turns this very common pattern of software evolution (which we also have seen in web browsers, CAD software, word processors, etc.) on its head: rather than gradually providing more and more flexibility in an ad-hoc manner through adding configuration parameters, slowly evolving into an extensive specialized programming language, Nmag starts out as an extension to a widely used programming language (Python) from which it gains all its flexibility and evolves towards more specialized notions to conveniently define and study the properties of very specific physical systems *[1.3]*.

The main advantage of this approach is two-fold: first, we do not gradually evolve another ad-hoc (and potentially badly implemented) special purpose programming language. Second, by drawing upon the capabilities of a well-supported existing framework for flexibility, we get a lot of additional power for free: the user can employ readily available and well supported Python libraries for tasks such as data post-processing and analysis, e.g. generating images for web pages etc. In addition to this, some users may benefit from the capability to use Nmag interactively from a command prompt, which can be very helpful during the development phase of an involved simulation script *[1.4]*.

At present, Nmag is based on the Python programming language. This seems to be a somewhat reasonable choice at present, as Python is especially friendly towards casual users who do not want to be forced to first become expert programmers before they can produce any useful results. Furthermore, Python is quite widespread and widely supported these days.

1.1.1.2 Micro magnetic Modelling

Micro magnetics deals with the interactions between magnetic moments on sub-micrometre length scales. These are governed by several competing energy terms. Dipolar energy is the energy which causes magnets to align north to south pole. Exchange energy will attempt to make the magnetic moments in the immediately surrounding space lie parallel to one another (if the material is ferromagnetic) or anti parallel to one another (if anti ferromagnetic). Anisotropy energy is low when the magnetic moments are aligned along a particular crystal direction. Zeeman energy is at its lowest when magnetic moments lie parallel to an external magnetic field.

Since the most efficient magnetic alignment (also known as a configuration) is the one in which the energy is lowest, the sum of these four energy terms will attempt to become as small as possible at the expense of the others, yielding complex physical interactions. The competition of these interactions under different conditions is responsible for the overall behaviour of a magnetic system.

To carry out micro magnetic simulations, a set of partial differential equations have to be solved repeatedly. In order to be able to do this, the simulated geometry has to be spatially discretised. The two methods that are most widely spread in micro magnetic modelling are the so-called finite difference (FD) method and the finite element (FE) method. With either the FD or the FE method, we need to integrate the Landau-Lifshitz and Gilbert equation numerically over time (this is a coupled set of ordinary differential equations). All these calculations are carried out by the Micro magnetic packages and the user does not have to worry about these.

The finite difference method subdivides space into many small cuboids. Sometimes the name cell is used to describe one of these cuboids. (Warning: in finite difference simulations, the simulated geometry is typically enclosed by a (big) cuboid which is also referred to as simulation cell. Usually (!) it is clear from the context which is meant.) Typically, all simulation cells in one finite difference simulation have the same geometry. A typical size for such a cell could be a cube of dimensions 3nm by 3nm by 3nm.

Let's assume we would like to simulate a sphere. The following picture shows an approximation of the shape of the sphere by cubes. This is the finite difference approach. For clarity, rather larger cubes were chosen to resolve the sphere – in an actual simulation one would typically use a much smaller cell size in order to resolve geometry better.

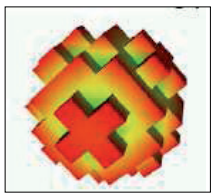


Figure 1.1 Discretised sphere using Cube as basic element

On the other hand, the finite element method (typically) subdivides space into many small tetrahedra. The tetrahedra are sometimes referred to as the (finite element) mesh elements. Typically, the geometry of these tetrahedra does vary throughout the simulated region. This allows combining the tetrahedra to approximate complicated geometries. The spherical shape is approximated better than with the finite differences.

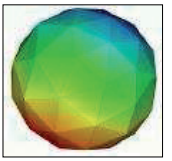


Figure 1.2 Discretised sphere using Tetrahedra as basic element

The first step in setting up a micro magnetic simulation is to describe the geometry. In the case of finite difference calculations, it will depend on the package to be used and how small simulation cells should be. In the case of finite element calculations, a finite element mesh is to be created first and then proceed for micro magnetic modelling and simulation.

1.1.2 Graphical User Interface (GUI)

In computing, a graphical user interface (GUI) is a type of user interface that allows users to interact with electronic devices with images rather than text commands. A GUI represents the information and actions available to a user through graphical icons and visual indicators such as secondary notation, as opposed to text-based interfaces, typed command labels or text navigation. The actions are usually performed through direct manipulation of the graphical elements. A GUI uses a combination of technologies and devices to provide a platform that the user can interact with, for the tasks of gathering and producing information.

1.1.3 Objective: Design of Graphical User Interface for NMAG

The first objective of work was to design a highly interactive Graphical User Interface for the scientific tool NMAG. This includes to manage the overall task of micro magnetic simulation using NMAG, starting from creating structures, generating finite element mesh, defining properties, creating simulation script (most challenging task), run the calculation and at the end to visualize and analyse the results, through the Graphical User Interface. The

aim was to enhance the functionality of the tool by making it more users friendly. The end result was to provide a tool which will help the users who have good understanding of the tool, by reducing the time to design the problem *[1.5]*, as well as to help the peoples who are new to this tool by providing them appropriate way to proceed and the result of work is able to fulfil these objectives.

1.2 The need of GUI designing for NMAG

1.2.1 Productivity

GUI saves the effort, time and resources involved in the manual operation of simulation through commands. In a GUI based system the information can be inserted, altered, updated and deleted easily *[1.6]*.

1.2.2 Accessibility

Accessibility issue include:

- To facilitate wider and deeper access to information
- To increase the retrieve ability of the resources
- To achieve a new level of simulation management.
- To improve the existing services and to introduce new services.
- To improve control over collection.
- To have an efficient control over the entire operation.
- To avoid the duplication of work.
- To facilitate sharing of the resources.

Some of the factors that forced the GUI design are:

- Information Explosion
- Time saving
- Cost effectiveness
- Data manipulation

1.2.3 Limitation of Existing System

The disadvantage of command based interface is of course that a novice user may be confronted with much more freedom than he can handle. Since the commands available in command line interfaces can be numerous, complicated operations can be completed using a short sequence of words and symbols. This allows for greater efficiency and productivity once many commands are learned, but reaching this level takes some time because the command words are not easily discoverable and not mnemonic.

Key limitations of existing system are:

- The existing system is manual.
- Wastage of paper and other resources.
- Selection procedure takes too much time.
- Huge amount of data gets collected during data entry.
- Consumes lot of time.
- Delay in results.

1.2.4 Advantage of proposed system

GUIs were introduced in reaction to the perceived steep learning curve of command-line interfaces, which require commands to be typed on the keyboard.

Key advantages of proposed system are:

- Fully GUI based system.
- Easy maintenance of large volume of data.
- No delay in simulation procedure.
- The user can take immediate action.
- Easily reports will be generated.
- Lesser manpower and manual work.

1.3 Feasibility Study

1.3.1 Concept of feasibility test

Preliminary investigation examines project feasibility, the likelihood that system will be useful. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system.

1.3.1.1 Technical Feasibility

The technical issues include:

- Does the necessary technology exist to do what is suggested?
- Do the proposed equipment have the technical capacity to hold the data required to use the new system?
- Will the proposed system provide some significant improvement in the existing system?
- Can the system be upgraded if developed?
- Are there technical guarantees of accuracy, reliability, ease of access and data security?

The current system developed is technically feasible. It is a GUI based interface for users of NMAG tool. Thus it provides an easy access to the users. The purpose of GUI is to create, establish and maintain a work flow among various entities in order to facilitate all concerned users in their various capacities or roles. Therefore, it provides the technical guarantee of accuracy, reliability and security. The software and hardware requirements for the development of this project are not many and are generally available with great ease and most of them are open source. The work for the project is done with the current equipment and existing software technology.

1.3.1.2 Operational Feasibility

Proposed projects are beneficial only if they can be turned out into information system. That will meet the users operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. The well-planned design

would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

Important issues to test the operational feasibility of a project includes:

- Is there sufficient support from the user to provide important feedback?
- Will the system be used and work properly if it is being developed and implemented?
- Will there be any resistance from the user that will undermine the possible application benefits?

1.3.1.3 Economic Feasibility

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economic feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs.

The system is economically feasible. It does not require any additional hardware or software, as the interface for this system is developed using the existing resources and technologies available.

1.4 Analysis and Design

1.4.1 Design Specifications

1.4.1.1 Design specifications from key owner of the tool

A brief plan of GUI, provided by Prof. Hans Fangohr (key owner of tool), is as follows:

Assumption:
- allow only one material
- Uniform constant magnetic external magnetic field,
On start-up:
- Ask user whether current directory should be the project directory
If not, allow to navigate to project directory, or to create a new subdirectory (if possible)
1. Create Geometry
2. Mesh geometry

3. Create simulation script
4. Run simulation
5. Analyse results
6. Visualise

1. Create Geometry:
- can offer default geometries, like
- sphere (ellipsoid), radius 50nm
- cube (thin film), l=50nm
- Cylinder, l=100nm, r=15nm
Ideally: allow user to enter origins, corners, radii, then write .geo file for this.
- should all be centred around origin
- express units in nanometre
-> .geo file
2. Mesh geometry / Create mesh:
*- list all *.geo files in directory and ask user to select one (if there are more than one)*
- provide buttons to
- assisted creation of geometry file (-> takes us to 'Create Geometry')
- visualise geometry
- generate mesh
- edit .geo file
3. Write simulation script:
- Select mesh (could propose mesh we have dealt with last as the default)
- select Material (ideally from list (initially Py, Co, Ni, Fe).
Allow user to modify these Material parameters manually.
Includes anisotropy
- select time dependent simulation
- How long, how often to save data, what data to save
- Suggestion:
- define save interval for averaged data
- define save interval for (spatially resolved) magnetisation (m)
- define save interval for all (spatially resolved) fields
Question: communicate to user that saving spatially resolved fields will save averages automatically
- Hysteresis loop:
4. Run simulation

1.4.1.2 Getting Information from the review of Existing Software Tool

I have gone through the detailed software description and working, made discussions with guide and list out the features expected by a user while working on GUI.

1.4.2 Structured Analysis

It is a set of techniques and graphical tools that allow the analyst to develop a new kind of system specifications that are easily understandable to the user. One tool for this is Data Flow Diagram (DFD).

1.4.2.1 DFD for NMAG GUI work flow

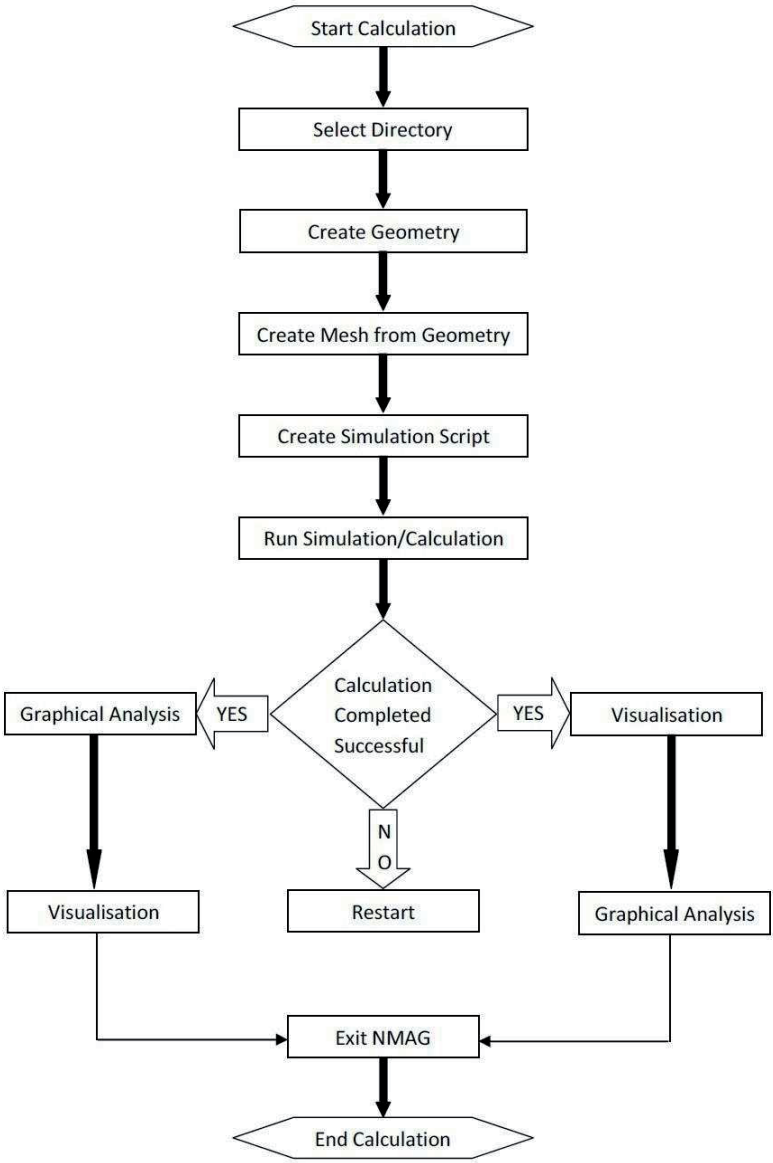


Figure 1.3 DFD for NMAG GUI work flow

1.4.3 Design Objectives

1.4.3.1 User Interface, Input and Output Design

User interface, input and output design begins the systems design phase of the SDLC. The user interface should include all the tasks, commands and communication between users and the system. In GUI environment a user can display and work with multiple windows on the single screen. Input Design Issues during input design, determine how data will be captured and entered into the system. Data capture is the identification and recording of source data. Data entry is the process of converting source data into computer readable form and entering it into or from system.

Input Design has six main objectives:

- To select a suitable input and data entry method
- To reduce the input volume
- To design attractive data entry screen
- To use validation check to reduce input errors
- To design required source documents
- To develop effective input control

Output Design issues include:

- What is the purpose of output?
- Who wants the information, why it is needed and how it will be used?
- Will the output be printed, viewed on screen or both?

Reports like any other elements of the user-computer interface should be attractive, professional, and easy to use. System analysts should realize that printed output is highly visible and manages sometimes judges project by the quality of reports they receive.

1.4.3.2 System Design

System design is highly creative process, which can be greatly facilitated by the following stages:

- Proper problem definition
- Pictorial description of the existing system
- Set of requirements of the new system
- Overall design of the whole system

During analysis the focus is on what needs to be done, independent of how it is done. During design, decisions are made about how the problem will be solved, first at a high level, then at increasingly detailed level.

System design is the first stage in which the basic approach of solving the problem is selected. During system design, the overall structure and style is decided. The system architecture is the overall organization of the system into components called sub-systems. The architecture provides the context in which more detailed decisions are made in later design stages. By making high level decisions that apply to the entire system, the system designer partitions the problem into sub-systems so that several designers working independently on different subsystems can be further work.

The system designer must make the following decisions:

- Organize the system into sub-systems.
- Identify concurrency inherent in the system.
- Strategy for data stores (data structure)
- Handling global resources.
- Basic architecture for system.

Input design: It is a process of converting user oriented inputs to a computer-based format. Input data are collected and organized into group of similar data once identified; appropriate input media are selected for processing. The goal of designing input data is to make data entry easy, logical and free from errors as possible in entering data.

Output design: Computer is the most important and direct source of information to the user. Efficient intelligible output design should improve the system's relationships with the user and help in decision making. The goal of designing output data is that it must be in user understandable form and consistent.

1.4.4 Logical design and code development

In this step of design we draw a logical model of our system of interest. On the basis of the structured analysis such as DFD, Structured English we draw a conceptual model of our system of interest. In work logical design is as follows:

Start new simulation:
if (Select Directory)
either create directory
or change directory
or exit
if (Create Geometry)
either select from default geometries
insert parameters, edit geometry, view geometry
or create custom geometry
insert parameters, edit geometry, view geometry
or exit
if (Create Mesh from geometry)
either create/edit/view geometry
or generate mesh
view mesh
or exit
if (Create Simulation Script)
insert parameters
create script
view/edit script
exit
if (Run Simulation)
start calculation
if (Restart Run)
restart calculation
if (Analyse Spatially Averaged Data)
either draw different plots using Xmgrace or gnuplot
or exit
if (Visualise)
Visualise calculation results using Mayavi
if (Exit NMAG)
exit from NMAG
: end Simulation

1.4.5 Physical Design

After deciding what has to be done actual work of designing start in which we actually design the system based on the logical design. The final physical design appears as given below *[1.7]*.

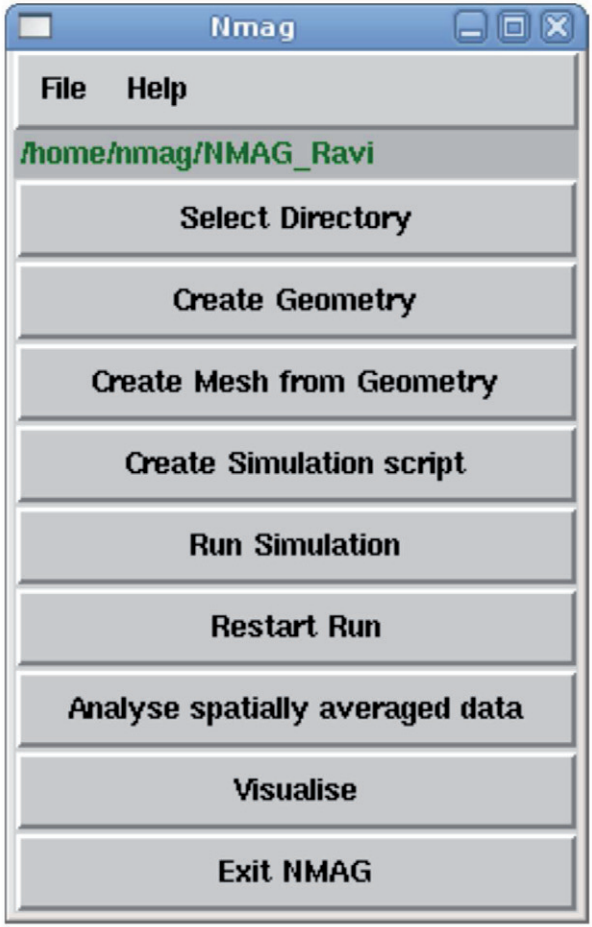


Figure 1.4 The Main Window of NMAG GUI

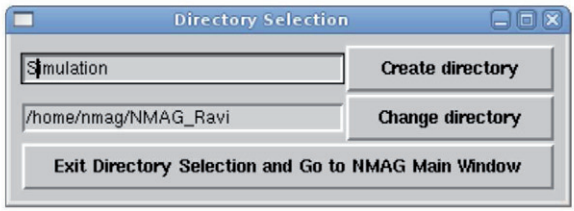


Figure 1.5 Select Directory Window

Figure 1.6 Create Geometry Window

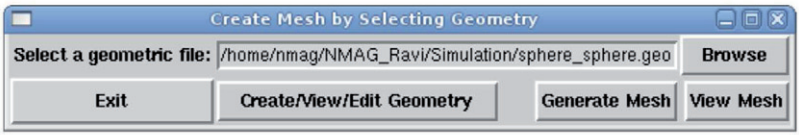


Figure 1.7 Create Mesh from Geometry Window

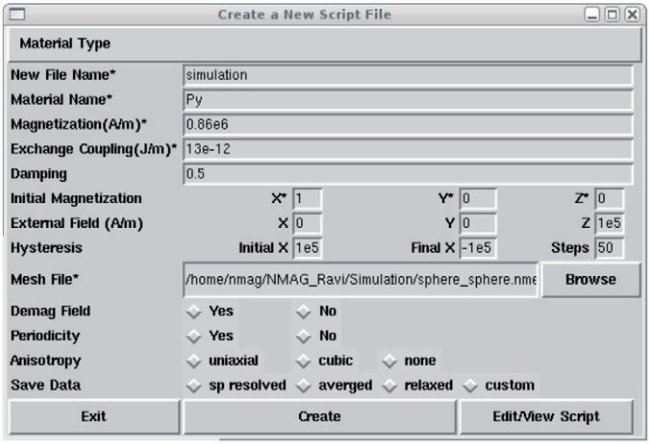


Figure 1.8 Create Simulation Script

Figure 1.9 Analyse Spatially Averaged Data Window

1.4.6 Files and file names [1.2]

1.4.6.1 geo files (.geo)

This file contains specification for geometry.

1.4.6.2 mesh files (.nmesh, .nmesh.h5)

Files that contain a finite element mesh.

1.4.6.3 Simulation scripts (.py)

Files that contain simulation program code. The ending is (by convention) .py which reflects that the programming language used is Python.

1.4.6.4 Data files (.ndt)

ndt stands for Nmag Data Table. ndt files are ascii files where each row corresponds to one-time step (or, more generally, configuration of the system). The columns contain:

_ metadata such as

– a unique identifier for every row

– the time at which the row was written

_ (spatially) averaged field data

The first two lines contain information about what data is stored in the various columns:

1. The first line provides a header

2. The second line provides the SI units

All other lines contain the actual data.

The file can be loaded into any data processing software (such as MS Excel, Origin, Matlab, Gnuplot ...). However, often it is more convenient to use the ncol tool to select the relevant columns, and only to pass the filtered data to a post-processing (e.g. plotting) program. Data is written into the ndt file whenever the save_data_ method of the simulation object is called.

1.4.6.5 Data files (.h5)

The h5 data files store spatially resolved fields. The format is a binary and compressed hdf5 format to which we have convenient access via the pytables package for Python. The user should not have to worry about reading this file directly, but use the nmagpp tool to access the data.

1.4.6.6 File names for data files

The filenames for the ndt and h5 data files are given by concatenation of the simulation name, the extension _dat. and the extension (.h5 or .ndt).

When a simulation object is created, for example in a file called mybar.py:

import nmag

sim = nmag.Simulation(name="bar")

then the simulation name is bar.

If no name is provided, i.e. the file mybar.py starts like this:

import nmag

sim = nmag.Simulation()

then the simulation name will be the run id. The run id is the filename of the simulation script (without the .py extension), i.e. the simulation name then will be mybar.

Let us assume for the rest of this section that the simulation name is bar. Once we use the save_data_ command, for example like this:

sim.save_data()

a ndt file will be created, with name bar_dat.ndt (= bar + _dat. + ndt).

Similarly, if we write the fields spatially resolved:

sim.save_data(fields='all')

a h5 data file with name bar_dat.h5 (= bar + _dat. + h5) will be created.

1.4.6.7 File names for log files

A log file is created that stores (most of) the messages displayed to stdout (i.e. the screen). The name of the log file starts with the name of the simulation script (without the .py extension), and ends with _log.log. For example, a simulation script with name mybar.py will have an associated log file with name mybar_log.log. Another three files will be created if the (undocumented) --dumpconf switch is provided. These are:

_ mybar_log.conf: This can be used to configure what data is logged.

_ mybar_ocaml.conf: Configuration of some variables used in the ocaml code

_ mybar_nmag.conf: Some variables used in the nmag code

1.4.6.8 .dat and .txt files

These are the files extension for files to save some specific data after extracting the main files, so as to plot by any plotting tool.

1.4.6.9 .vtk files

These files store results for visualization tool like Mayavi

1.4.6.10 .ps and .pdf files

These are the extensions to store the results in proper printable format.

1.5 Implementation

1.5.1 System testing and quality assurance

In this phase the system is tested. Normally programs are written as a series of individual modules, these subjects to separate and detailed test. The system is then tested as a whole. The separate modules are brought together and tested as a complete system. The system is tested to ensure that interfaces between modules work (integration testing), the system works on the intended platform and with the expected volume of data (volume testing) and that the system does what the user requires (acceptance/beta testing).

The developed software is tested by the user/developer to detect the possible errors. A software test program should aim at detecting maximum number of errors in the software. The process of testing involves creating the test cases and implementing those test cases on the software to detect errors. Testing validates that the actual performances of the software meets the user requirements. The Testing Strategy and plan document is a formal document that contains the test plan and the list of all types of test to be done on the software. Test Plan consists of the components to be tested, people involved in testing, test cases and the acceptance criterion for the software. The test planned for the software should be such that they detect the deviation of the software from client requirements. The test planned in the test plan should begin from smaller module and move to cover the entire system. A testing strategy describes the steps that are performed during the testing of software.

Levels of Quality Assurance:

Alpha testing: In this test software goes through a phase in which errors and failures based on simulated user requirement are verified and studied.

Beta testing: In this modified software is checked at actual user's site on a live environment.

Note: THIS SYSTEM HAS SUCCESSFULLY GONE THROUGH BOTH TESTING.

1.5.2 System Testing

1.5.2.1 Example 1: Demag field in uniformly magnetised sphere [1.2]

.geo file

```
algebraic3d
solid main = sphere (0, 0, 0; 40)-maxh=3.0 ;
tlo main;
```

Mesh:

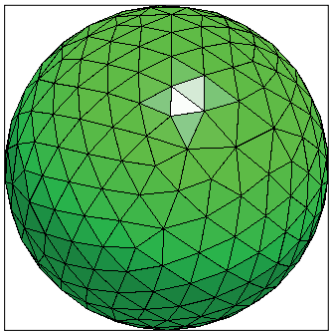


Figure 1.10 Mesh for the sphere geometry

Simulation Script:

```
import sys
sys.argv.append('--clean')
import nmag
from nmag import SI

#create simulation object
sim = nmag.Simulation()

# define magnetic material
Py = nmag.MagMaterial(name = 'Py',
                      Ms = SI(1e6, 'A/m'),
                      exchange_coupling = SI(13.0e-12, 'J/m'))

# load mesh
sim.load_mesh('sphere1.nmesh.h5',
              [('sphere', Py)],
              unit_length = SI(1e-9, 'm'))

# set initial magnetization
sim.set_m([1,0,0])

# set external field
sim.set_H_ext([0,0,0], SI('A/m'))

# Save and display data in a variety of ways
sim.save_data(fields='all') # save all fields spatially resolved
```

```
                                        # together with average data

    # sample demag field through sphere
    for i in range(-10,11):
        x = i*1e-9                  #position in meters
        H_demag = sim.probe_subfield_siv('H_demag', [x,0,0])
        print "x =", x, ": H_demag = ", H_demag
```

Results:

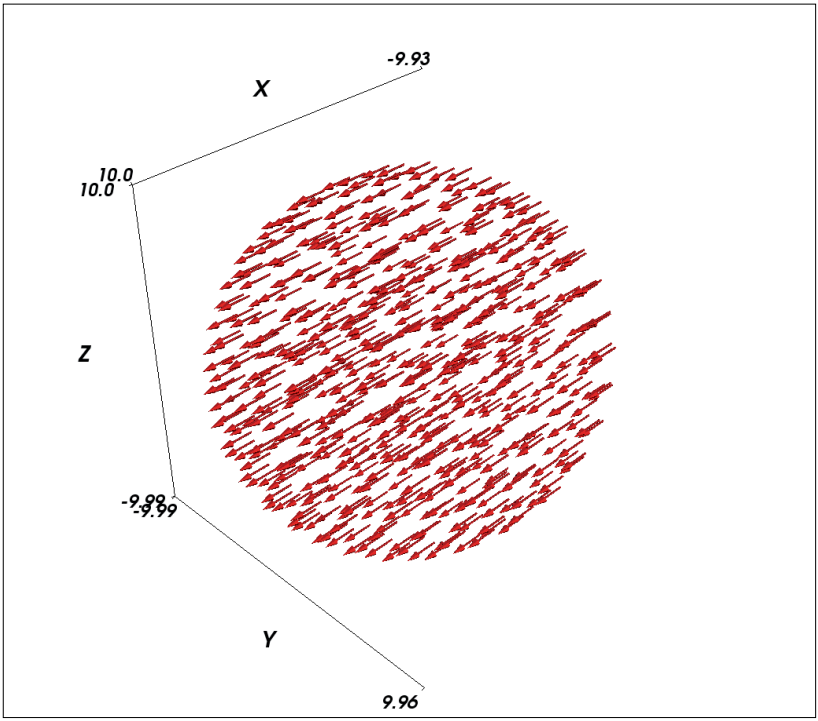


Figure 1.11 Demag field in uniformly magnetised sphere for given parameters

1.5.2.2 Example 2: Simple hysteresis loop [1.2]

.geo files:

```
algebraic3d
solid test = ellipsoid (0, 0, 0; 30.0, 0, 0; 0, 10.0, 0; 0, 0, 10.0) -maxh=3;
tlo test;
```

Mesh:

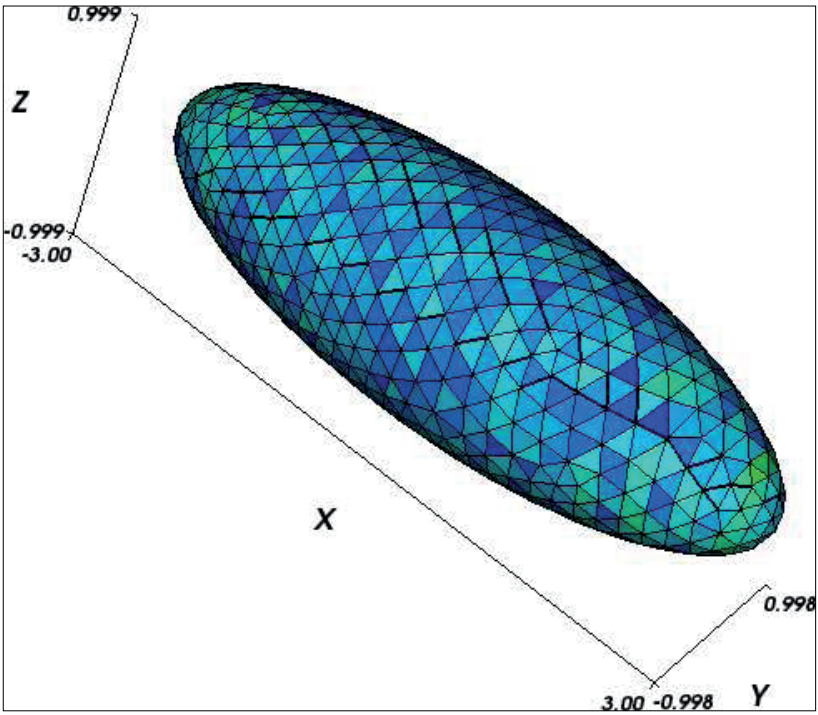


Figure 1.12 Mesh for ellipsoid

Simulation Script:

```
import nmag
from nmag import SI, at

#create simulation object
sim = nmag.Simulation()

# define magnetic material
Py = nmag.MagMaterial(name="Py",
                      Ms=SI(1e6,"A/m"),
                      exchange_coupling=SI(13.0e-12, "J/m"))

# load mesh: the mesh dimensions are scaled by 0.5 nm
```

```
sim.load_mesh("ellipsoid.nmesh.h5",
              [("ellipsoid", Py)],
              unit_length=SI(1e-9,"m"))

# set initial magnetization
sim.set_m([1.,0.,0.])

Hs = nmag.vector_set(direction=[1.,0.01,0],
                     norm_list=[ 1.00,  0.95, [], -1.00,
                           -0.95, -0.90, [],  1.00],
                     units=1e6*SI('A/m'))

# loop over the applied fields Hs
sim.hysteresis(Hs, save=[('restart','fields', at('convergence'))])
```

Results:

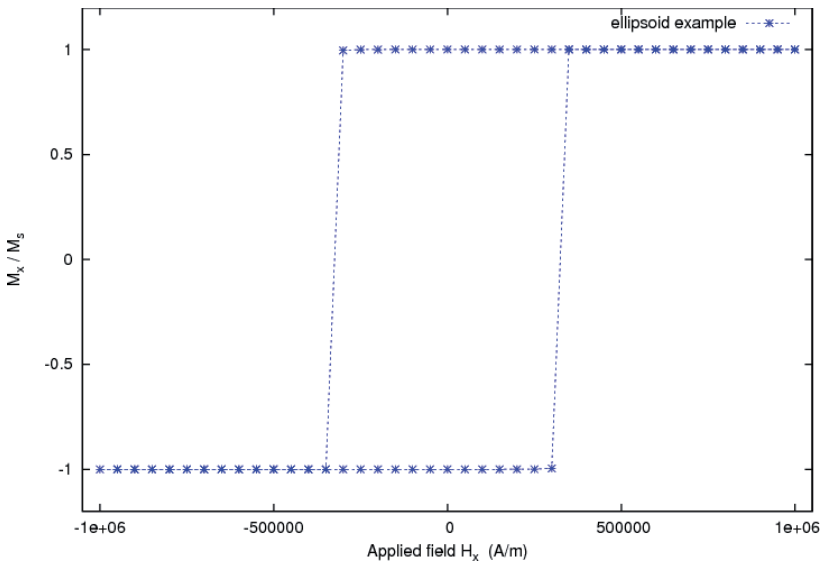


Figure 1.13 Hysteresis loop for the ellipsoid for given parameters

1.6 Hardware and software selection

A Linux based system having good hardware configuration and minimum of following software will be required to work the system in its designated way:

- Python for scripting
- Text editor preferably gedit
- Netgen for geometry visualization and mesh generation
- Proper set up having basic NMAG packages for necessary calculations
- Xmgrace or GNU plot for graphical result display
- Mayavi for pictorial results visualisation

1.7 Conclusion

It has been a great pleasure for me to work on this exciting and challenging project. This project proved good for me as it provided practical knowledge of programming in Python. It also provides knowledge about the latest technology used in developing various user interfaces that will be great demand in future. This will provide better opportunities and guidance in future for developing projects.

1.8 Benefits

The merits of this project are as follows:

- This project offers user to enter the data through simple and interactive forms. This is very helpful for the client to enter the desired information in a simplistic manner.
- Sometimes the user finds in the later stages of using project that he needs to update some of the information that he entered earlier. There are options for him by which he can update the records.
- User is provided the option of monitoring the records he entered earlier. He can see the desired records with the variety of options provided by him.
- From every part of the project the user is provided with the links through framing so that he can go from one option of the project to other as per the requirement. This is bound to be simple and very friendly as per the user is concerned.
- Data storage and retrieval will become faster and easier to maintain because data is

stored in a systematic manner and in a single data directory.

- ❖ Decision making process would be greatly enhanced because of faster processing of information since data collection from information available on computer takes much less time than manual system.
- ❖ Allocating of sample results becomes much faster.
- ❖ Easier and faster data transfer through latest technology associated with the computer and communication.
- ❖ Through these features it will increase the efficiency, accuracy and transparency.

1.9 Future Work

Developments are continuously going on in NMAG project. There is an active research is going on the NMAG to enhance its functionality and usability, due to which continuously new features are added in the tool. The future work concerned with GUI design of NMAG is to embed these new features in the current GUI, so that these new features can also be used in a simpler manner.

1.10 References

[1.1] NMAG Website: http://nmag.soton.ac.uk/nmag/index.html.

[1.2] NMAG User Manual: http://nmag.soton.ac.uk/nmag/current/manual/singlehtml/manual.html.

[1.3] Thomas Fischbacher, Matteo Franchin, Giuliano Bordignon,, Hans Fangohr, A Systematic Approach to Multiphysics Extensions of Finite-Element-Based Micro magnetic Simulations: Nmag, IEEE Transactions on Magnetics 43, 6, 2896-2898 (2007), online at http://eprints.soton.ac.uk/46725/.

[1.4] Thomas Fischbacher, Matteo Franchin, Giuliano Bordignon, Andreas Knittel, Hans Fangohr, Parallel execution and scriptability in micro magnetic simulations, Journal of Applied Physics 105, 07D527 (2009), online at http://link.aip.org/link/?JAPIAU/105/07D527/1.

Chapter 2: Design of modules for Magnetic Moment based Optimization in USPEX

2.1 Introduction

2.1.1 USPEX Overview

USPEX (Universal Structure Prediction: Evolutionary Xtallography) is a method developed jointly by Artem R. Oganov and Colin W. Glass, Andriy O. Lyakhov and Qiang Zhu, and implemented in the same-name code written by Andriy Lyakhov and Colin W. Glass and Qiang Zhu mainly at Stony Brook University, New York *[2.1]*. USPEX can be used to predict stable crystal structures at given P-T conditions, knowing only the chemical composition (or to predict both the stable compositions and structures, given the element types). Many previous attempts to solve crystal structure problem were plagued by low success rate and extreme computational costs that prevented full ab initio studies. USPEX avoids both of these problems. In fact, "uspekh" means "success" in Russian - which highlights a nearly 100% success rate for this method.

USPEX can also be used for finding low-energy metastable phases, as well as stable structures of nanoparticles, surface reconstructions, molecular packings in organic crystals, and for searching for materials with desired physical (mechanical, electronic) properties. The USPEX code is based on an efficient evolutionary algorithm developed by A.R. Oganov's group, but also has options for using alternative methods (random sampling, metadynamics, corrected PSO algorithms). USPEX is interfaced with many DFT or classical codes, such as VASP, SIESTA, GULP, Quantum Espresso, CP2K, and so on.

USPEX is characterized not only by high efficiency and reliability (witnessed by the recent blind test of inorganic crystal structure prediction, where it has scored the highest success) but also by state-of-the art analysis and visualization tools developed by Mario Valle specifically for the USPEX project. These tools enable rapid visual analysis of large datasets, with interactive screens and possibilities not only to visualize crystal structures and analyse them (in terms of fingerprint functions, space groups, bond lengths and angles, order parameters, etc.), but to explore correlations between structures and their properties, energies, and simulation parameters. USPEX is based on a carefully tuned structure prediction-specific evolutionary algorithm. USPEX searches for the structure corresponding to the global minimum of the ab initio free energy. The quality of trial structures is judged by the ab initio free energy calculated by an external ab initio code (currently for this,

USPEX code can use VASP and SIESTA, but developers may include other codes as well). The use of simpler methods, e.g. based on inter atomic potentials, is also possible - in this case the structure prediction is extremely fast.

Features of the code:

- Structure prediction with just the chemical composition.
- Incorporation of partial structural information is possible
  - Constraining search to fixed experimental cell parameters, or fixed cell shape, or fixed cell volume,
  - Starting structure search from known or hypothetical structures.
- Efficient constraint techniques, which eliminate unphysical and redundant regions of the search space.
- Handling of molecules (rather than atoms), fully or partly rigid is possible.
- Restart facilities, enabling calculations to be continued from any point along the evolutionary trajectory (if needed, with changed parameters).
- Powerful visualisation and analysis techniques implemented in the STM4 code (by M.Valle), fully interfaced with USPEX.
- USPEX is interfaced with VASP, SIESTA and GULP codes. There are development interfaces with ABINIT, LAMMPS, and MD++.
- Job submission from local workstation to remote supercomputers is possible.
- Job submission via grid is possible (grid part by S. Tikhonov and S. Sobolev).
- Many new features are now in progress.

This software was developed mainly at the Stony Brook University, New York. It is free for academic use.

2.1.1.1 USPEX Philosophy

The USPEX (Universal Structure Predictor: Evolutionary Xtallography) code possesses rather unique capabilities: it allows one to predict the stable structure of a given compound at given conditions (pressure, temperature) just from the knowledge of the chemical composition and using no experimental information *[2.2]*. From the beginning, this non-empirical crystal structure prediction was the main aim of the USPEX project. This is achieved by merging a specially developed evolutionary algorithm featuring local

optimisation and real-space representation with *ab initio* simulations. In addition to this fully non-empirical search, USPEX allows one to predict also a large set of robust metastable structures and perform several types of simulations using various degrees of prior knowledge. The problem of crystal structure prediction is very old and does, in fact, constitute the central problem of theoretical crystal chemistry. In 1988 John Maddox *[2.3]* wrote that:

"One of the continuing scandals in the physical sciences is that it remains in general impossible to predict the structure of even the simplest crystalline solids from a knowledge of their chemical composition… Solids such as crystalline water (ice) are still thought to lie beyond mortals' ken".

Following is the working of USPEX:

1. The first generation is produced by a random-number generator (only those structures which satisfy the hard constraints are allowed). Non-random start from some good structures provided by user is also possible.

2. Among the locally optimised structures, a certain number of the worst ones are rejected, and the remaining structures participate in creating the next generation through heredity, permutation and mutation. Selection probabilities for variation operators are derived from the rank of their fitness (i.e. their calculated free energies).

During heredity, new structures are produced by matching slices (chosen in random directions and with random positions) of the parent structures. A certain fraction of structures is produced by randomly shifting these slices in their matching plane, and this random operation can be viewed as "coordinate mutation". Heredity for the lattice vectors matrix elements (for this matrix we use the upper-triangular form in order to avoid unphysical whole-cell rotations) is done by taking a weighted average, using random weights. A certain fraction of the new generation is created by permutation (i.e. switching identities of two or more atoms in a structure) and mutation (random change of the cell vectors and/or atomic positions). Lattice mutation essentially incorporates into this method the ideas of metadynamics, where new structures are found by building up cell distortions of some known structure. Unlike in metadynamics, in this method the distortions are not accumulated, so to obtain new structures the strain components should be large.

To avoid pathological lattices all newly obtained structures are rescaled to have a certain volume, which is then relaxed by local optimisation. The value of the rescaling volume is not very important and can be easily estimated either from the equation of state of some known structure or by optimising a random structure; this value is used only for the first generation and for subsequent generations is adapted to the volumes of several best found structures. A specified number of the best structures (usually, one) of the current generation always survive, mate and compete in the next generation.

3. The simulation is terminated after some halting criterion is met. In our experience, for systems with up to ~ 10 atoms in the cell the global minimum is often found within the first few generations, for systems with ~20 atoms in the cell this usually takes ~10-20 generations. Among the important results of the simulation are the stable crystal structure and a set of robust metastable structures at given pressure-temperature conditions.

USPEX allows one to find the stable crystal structure of a given compound at given external conditions (pressure, temperature, etc.). Moreover, it also produces a set of robust metastable structures. Unlike traditional simulation methods that only sample a small part of the free energy landscape close to some minimum, our method explores the entire free energy surface on which it locates the most promising areas. This allows one to see which aspects of structures (molecular vs. coordination or metallic vs. insulating structures, atomic coordination numbers, bond lengths and angles) are required for stability and therefore provides an interesting way of probing structural chemistry of matter at different conditions. Our present implementation in the USPEX code is very efficient for systems with up to ~100 atoms in the unit cell. With additional developments we expect the method to be efficient also for larger systems. Since no symmetry constraints are imposed during simulations, symmetry is one of the results of our algorithm. This ensures that the resulting structures are mechanically stable and do not contain any unstable Γ-phonons.

This approach enables crystal structure prediction without any experimental input. Essentially, the only input is the chemical formula (then, it typically performs simulations for different numbers of formula units in the cell). However, in some cases, we would like to be able to predict also possible stoichiometry. One of the first steps in this direction was taken in *[2.4]*, who have applied an *ab initio* evolutionary algorithm to find stable alloys. However, in their work the structure was fixed (only fcc- and bcc- structures were explored). It would be interesting to incorporate variable composition in our algorithm in order to find

stable structures *and* likely stoichiometry in a given system. The present limitation of the method to ordered periodic structures can be overcome once it becomes possible to calculate free energies of disordered and aperiodic structures. Clearly, the quality of the global minimum found by USPEX depends on the quality of the *ab initio* description of the system. Present-day DFT simulations (e.g., within the GGA) are adequate for most situations, but it is known that these simulations do not fully describe van der Waals bonding and the electronic structure of Mott insulators. In both areas there are significant achievements, which can be used for calculating *ab initio* free energies and evaluation of structures in evolutionary simulations.

2.1.1.2 Optimization

In physics, mathematics, computational science, or management science, optimization refers to the selection of a best element from some set of available alternatives.

In the simplest case, an optimization problem consists of maximizing or minimizing a real function by systematically choosing input values from within an allowed set and computing the value of the function. The generalization of optimization theory and techniques to other formulations comprises a large area of applied physics. More generally, optimization includes finding "best available" values of some objective function given a defined domain, including a variety of different types of objective functions and different types of domains.

2.1.2 Magnetic Moment

The magnetic moment of a magnet is a quantity that determines the force that the magnet can exert on electric currents and the torque that a magnetic field will exert on it. A loop of electric current, a bar magnet, an electron, a molecule, and a planet all have magnetic moments.

Both the magnetic moment and magnetic field may be considered to be vectors having a magnitude and direction. The direction of the magnetic moment points from the south to north pole of a magnet. The magnetic field produced by a magnet is proportional to its magnetic moment as well. More precisely, the term magnetic moment normally refers to a system's magnetic dipole moment, which produces the first term in the multipole expansion of a general magnetic field. The dipole component of an object's magnetic field is symmetric

about the direction of its magnetic dipole moment, and decreases as the inverse cube of the distance from the object.

2.1.3 Objective: Design of modules for Magnetic Moment based Optimization in USPEX

The second objective of the work was to design a code for USPEX which is able to do the optimization based on magnetic moment values. This includes to manage the overall task of magnetic moment based optimization, starting from to create user choice for magnetic moment based optimization, set it as fitness parameter, initialization of magnetic moment values, design of a function which can read the values of magnetic moment from result of ab initio calculation (most challenging task), write the values in output files and finally optimize by maximizing the value of magnetic moment. The aim was to enhance the functionality of the tool. The end result was to provide a tool which will attract the peoples all around the world who are working in the field of magnetism, to use this tool *[2.5]*, as well as allow the people who are already working on this tool to give some novel findings based on this enhanced functionality and the result of work is able to fulfil these objectives.

2.2 The need of Magnetic Moment based Optimization

All around the world many peoples are working for finding the materials which have higher magnetic moment. The interest is to find the materials having significant value of magnetic materials to find some novel applications. As the value of magnetic moment increases for the materials it becomes much more useful for some applications. It is very difficult to do all these things experimentally as the procedure is very tedious and costly. For simulation purpose there is no tool available which can give complete information together with stable crystal structure. As USPEX tool is currently providing stable crystal structure based on no. of parameters, a need was felt to add additional functionality in the USPEX so that a tool for the users can be made available which can give stable structure based on optimized value of magnetic moment.

2.3 Feasibility Study

2.3.1 Concept of feasibility test

Preliminary investigation examines project feasibility, the likelihood that system will be useful *[2.6]*. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system.

2.3.1.1 Technical Feasibility

The technical issues include:

- Does the necessary technology exist to do what is suggested?
- Do the proposed equipment have the technical capacity to hold the data required to use the new system?
- Will the proposed system provide some significant improvement in the existing system?
- Can the system be upgraded if developed?
- Are there technical guarantees of accuracy, reliability, ease of access and data security?

The current system developed is technically feasible. It provides the technical guarantee of accuracy, reliability and security. The software and hardware requirements for the development of this project are not many and are generally available with great ease. The work for the project is done with the current equipment and existing software technology.

2.3.1.2 Operational Feasibility

Proposed projects are beneficial only if they can be turned out into information system. That will meet the users operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

Important issues to test the operational feasibility of a project include:

- Is there sufficient support from the user to provide important feedback?
- Will the system be used and work properly if it is being developed and implemented?
- Will there be any resistance from the user that will undermine the possible application benefits?

2.3.1.3 Economic Feasibility

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economic feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs.

The system is economically feasible. It does not require any additional hardware or software, as the interface for this system is developed using the existing resources and technologies available.

2.4 Analysis and Design

2.4.1 Design Specifications

First of all, I have gone through the code, understand it's working. I found the basic algorithm and its operation. Based on the above analysis I have designed a brief outline of the design to be done in order to achieve the goal of magnetic moment based optimization. Then I discussed this with guide, one of the authors of the code and finalized the proper design scheme.

2.4.2 Structured Analysis

It is a set of techniques and graphical tools that allow the analyst to develop a new kind of system specifications that are easily understandable to the user. One tool for this is Data Flow Diagram (DFD).

2.4.2.1 DFD of work flow for Magnetic Moment based Optimisation:

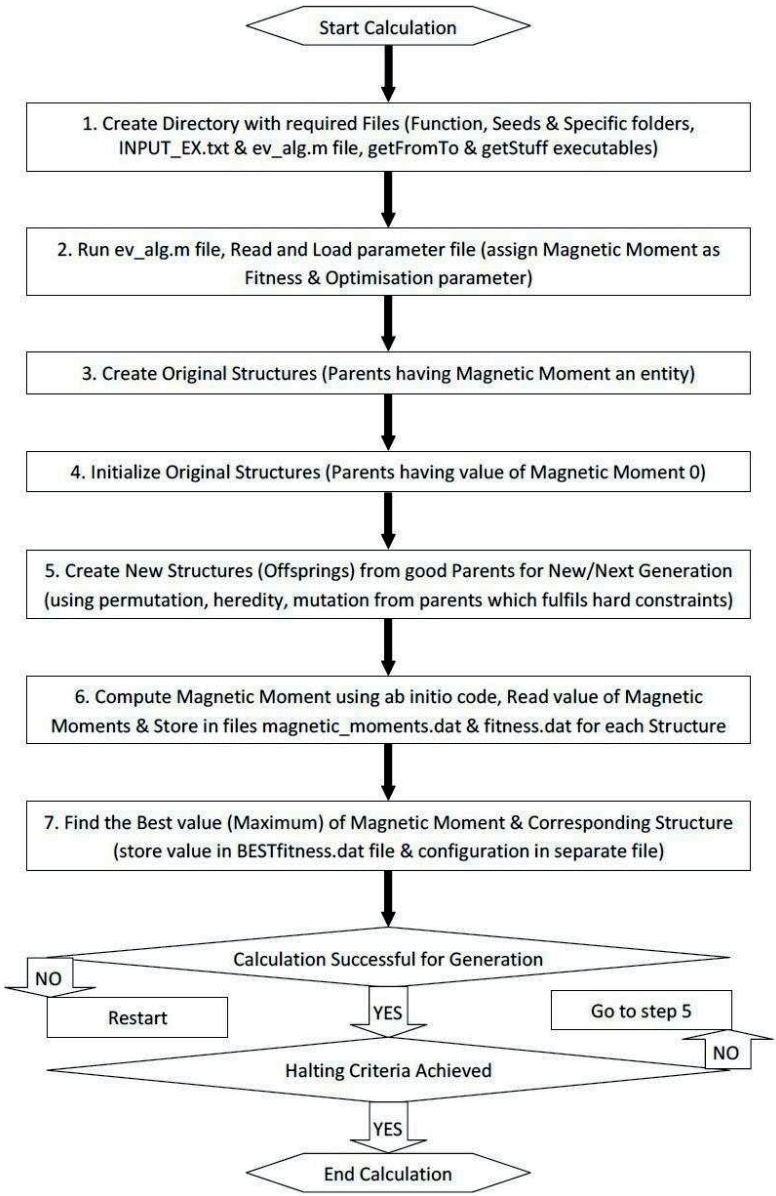


Figure 2.1 DFD of work flow for Magnetic Moment based Optimisation in USPEX

2.4.3 Design Objectives

System design is highly creative process, which can be greatly facilitated by the following stages:

- Proper problem definition
- Pictorial description of the existing system
- Set of requirements of the new system
- Overall design of the whole system

During analysis the focus is on what needs to be done, independent of how it is done. During design, decisions are made about how the problem will be solved, first at a high level, then at increasingly detailed level.

System design is the first stage in which the basic approach of solving the problem is selected. During system design, the overall structure and style is decided. The system architecture is the overall organization of the system into components called sub-systems. The architecture provides the context in which more detailed decisions are made in later design stages. By making high level decisions that apply to the entire system, the system designer partitions the problem into sub-systems so that several designers working independently on different subsystems can be further work.

The system designer must make the following decisions:

- Organize the system into sub-systems.
- Identify concurrency inherent in the system.
- Strategy for data stores (data structure)
- Handling global resources.
- Basic architecture for system.

2.4.4 Logical design and code development

In this step of design we draw a logical model of our system of interest. On the basis of the structured analysis such as DFD, Structured English we draw a conceptual model of our system of interest. In project logical design is as follows:

Start new simulation:
if (INPUT_EA.txt file)
add magnetic moment as optimization parameter
if (createORGstruc.m file)
set magnetic moment as optimization parameter
if (initialize_POP_STRUC.m file)
add magnetic moment in initial population structure
and set magnetic moment value to zero for current generation
if (ev_alg.m file)
create magneticMoments.dat file
perform magnetic moment based optimization for current generation
and set magnetic moment value to zero for offsprings
if (readCalcFilesVASP.m file)
add a function call to read magnetic moments from ab initio code output
if (writeOutput.m)
write magnetic moment values in magneticMoments.dat file
if (writeGenerationOutput.m)
create BESTmagneticMoments.dat file
and write BEST magnetic moment value in BESTmagneticMoments.dat file
design a function to read magnetic moment values
: end Simulation

2.4.5 Physical Design

After deciding what has to be done actual work of designing start in which we actually design the system based on the logical design. The final physical design appears as given below *[2.7]*

Table 2.1 Modifications and addition of modules in USPEX for Magnetic Moment based Optimization

Code Segment to be Modified	Code Segment after Modification
File Name: ***INPUT_EA.txt***	
% optimisation criteria *enthalpy : optType (optimise by: enthalpy, volume, hardness, struc_order, aver_dist)*	*% optimisation criteria* *mag_moment : optType (optimise by: enthalpy, volume, hardness, struc_order, aver_dist, mag_moment)*
File Name: ***createORGstruc.m***	
elseif strcmpi(optType, 'aver_dist') *ORG_STRUC.optType = 5;* *End*	*elseif strcmpi(optType, 'aver_dist')* *ORG_STRUC.optType = 5;* *elseif strcmpi(optType, 'mag_moment')* *ORG_STRUC.optType = 6;* *End*
File Name: ***initialize_POP_STRUC.m***	

,'Cluster_Volume',{})	*,'Cluster_Volume',{}, 'mag_moment',{})*
POP_STRUC.POPULATION(i).survivingSince = 0;	*POP_STRUC.POPULATION(i).survivingSince = 0;* *POP_STRUC.POPULATION(i).mag_moment = 0;*
File Name: ***ev_alg.m***	
unix(['echo "------- generation' num2str(POP_STRUC.generation) ' ------- " >>' POP_STRUC.resFolder '/enthalpies']);	*unix(['echo "------- generation' num2str(POP_STRUC.generation) ' ------- " >>' POP_STRUC.resFolder '/enthalpies']);* *if ORG_STRUC.optType == 6* *unix(['echo "------- generation' num2str(POP_STRUC.generation) ' ------- " >>' POP_STRUC.resFolder '/magneticMoments.dat']);* *end*
fitness = -sqrt(fitness); *end* *[nothing, ranking] = sort(fitness);*	*fitness = -sqrt(fitness);* *elseif ORG_STRUC.optType == 6* *for fit_loop = 1:length(POP_STRUC.POPULATION)* *fitness(fit_loop) = POP_STRUC.POPULATION(fit_loop).mag_moment;* *end* *end* *for i = 1 : length(fitness)* *try* *if POP_STRUC.POPULATION(i).FITNESSES(end) > 999999 % structure with errors* *fitness(i) = 1000000;* *end* *catch* *end* *end* *[nothing, ranking] = sort(fitness);*
,'Cluster_Volume',{})	*,'Cluster_Volume',{}, 'mag_moment',{})*
OFF_STRUC.POPULATION(ind).survivingSince = 0;	*OFF_STRUC.POPULATION(ind).survivingSince = 0;* *OFF_STRUC.POPULATION(ind).mag_moment = 0;*
File Name: ***readCalcFilesVASP.m***	
[POP_STRUC.POPULATION(Ind_No).COORDINATES,POP_STRUC.POPULATION(Ind_No).LATTICE] = read_VASP ('CONTCAR', NCoords);	*[POP_STRUC.POPULATION(Ind_No).COORDINATES,POP_STRUC.POPULATION(Ind_No).LATTICE] = read_VASP ('CONTCAR', NCoords);* *if ORG_STRUC.optType == 6*

	POP_STRUC.POPULATION(Ind_No).mag_moment = readMagneticMoment(); *End*
File Name: ***writeOutput.m***	
unix(['echo ' num2str(POP_STRUC.POPULATION(Ind_No).FITNESSES) ' >> ' ORG_STRUC.homePath '/' POP_STRUC.resFolder '/enthalpies_complete.dat']);	*unix(['echo ' num2str(POP_STRUC.POPULATION(Ind_No).FITNESSES) ' >> ' ORG_STRUC.homePath '/' POP_STRUC.resFolder '/enthalpies_complete.dat']);* *if ORG_STRUC.optType == 6* *unix(['echo ' num2str(POP_STRUC.bodyCount) ' ' num2str(POP_STRUC.POPULATION(Ind_No).mag_moment) ' >>' ORG_STRUC.homePath '/' POP_STRUC.resFolder '/magneticMoments.dat']);* *end*
File Name: ***writeGenerationOutput.m***	
unix(['echo "------- generation' num2str(POP_STRUC.generation) ' -------" >>' ORG_STRUC.homePath '/' POP_STRUC.resFolder '/BESTkpoints.dat']);	*unix(['echo "------- generation' num2str(POP_STRUC.generation) ' -------" >>' ORG_STRUC.homePath '/' POP_STRUC.resFolder '/BESTkpoints.dat']);* *if ORG_STRUC.optType == 6* *unix(['echo "------- generation' num2str(POP_STRUC.generation) ' -------" >>' ORG_STRUC.homePath '/' POP_STRUC.resFolder '/BESTmagneticMoments.dat']);* *end*
unix(['echo ' num2str(POP_STRUC.PSO(POP_STRUC.bestPSOstruc).enthalpy) ' >>' ORG_STRUC.homePath '/' POP_STRUC.resFolder '/BESTenthalpies.dat']);	*unix(['echo ' num2str(POP_STRUC.PSO(POP_STRUC.bestPSOstruc).enthalpy) ' >>' ORG_STRUC.homePath '/' POP_STRUC.resFolder '/BESTenthalpies.dat']);* *if ORG_STRUC.optType == 6* *unix(['echo ' num2str(-1*POP_STRUC.PSO(POP_STRUC.bestPSOstruc).fitness) ' >>' ORG_STRUC.homePath '/' POP_STRUC.resFolder '/BESTmagneticMoments.dat']);* *end*

unix(['echo ' num2str(POP_STRUC.POPULATION(POP_STRUC.ranking(1)).FITNESSES(end)) ' >>' ORG_STRUC.homePath '/' POP_STRUC.resFolder '/BESTenthalpies.dat']);	*unix(['echo ' num2str(POP_STRUC.POPULATION(POP_STRUC.ranking(1)).FITNESSES(end)) ' >>' ORG_STRUC.homePath '/' POP_STRUC.resFolder '/BESTenthalpies.dat']);* *if ORG_STRUC.optType == 6* *unix(['echo ' num2str(-1*fitness(POP_STRUC.ranking(1))) ' >>' ORG_STRUC.homePath '/' POP_STRUC.resFolder '/BESTmagneticMoments.dat']);* *end*

Function Designed to read values of magnetic moment from ab initio code output: readMagneticMoment.m

```
function mag_moment = readMagneticMoment()
global POP_STRUC
global ORG_STRUC

[nothing, magmomStr] = unix('./getStuff OSZICAR mag 11')
magmom = str2num(magmomStr);

if isempty(magmom)
if isempty (POP_STRUC.POPULATION(Ind_No).Error)
POP_STRUC.POPULATION(Ind_No).Error = 1;
else
POP_STRUC.POPULATION(Ind_No).Error =
POP_STRUC.POPULATION(Ind_No).Error + 1;
end
mag_moment = 0;
unix(['echo PROBLEM_read_VASP_OSZICAR ' num2str(Ind_No)])
unix('pwd');
else
mag_moment = magmom(end);
end
```

2.4.6 Files and file names

2.4.6.1 *MATLAB scripts (.m)*

These are the main MATLAB script and function files, in which USPEX code is written.

2.4.6.2 MATLAB variables (.mat)

Used to store population structure in USPEX.

2.4.6.3 .dat and .txt files

These are the files extension for files to save data.

2.5 Implementation

2.5.1 System testing and quality assurance

In this phase the system is tested. Normally programs are written as a series of individual modules, these subjects to separate and detailed test. The system is then tested as a whole. The separate modules are brought together and tested as a complete system. The system is tested to ensure that interfaces between modules work (integration testing), the system works on the intended platform and with the expected volume of data (volume testing) and that the system does what the user requires (acceptance/beta testing).

The developed software is tested by the user/developer to detect the possible errors. A software test program should aim at detecting maximum number of errors in the software. The process of testing involves creating the test cases and implementing those test cases on the software to detect errors. Testing validates that the actual performances of the software meets the user requirements. The Testing Strategy and plan document is a formal document that contains the test plan and the list of all types of test to be done on the software. Test Plan consists of the components to be tested, people involved in testing, test cases and the acceptance criterion for the software. The test planned for the software should be such that they detect the deviation of the software from client requirements. The test planned in the test plan should begin from smaller module and move to cover the entire system. A testing strategy describes the steps that are performed during the testing of software.

Levels of Quality Assurance:

Alpha testing: In this test software goes through a phase in which errors and failures based on simulated user requirement are verified and studied.

Beta testing: In this modified software is checked at actual user's site on a live environment.

Note: THIS SYSTEM HAS SUCCESSFULLY GONE THROUGH BOTH TESTING.

2.5.2 System Testing

2.5.2.1 Example 1: Magnetic Moment Based optimization for FePt system at a pressure of 100 Kilo Bar

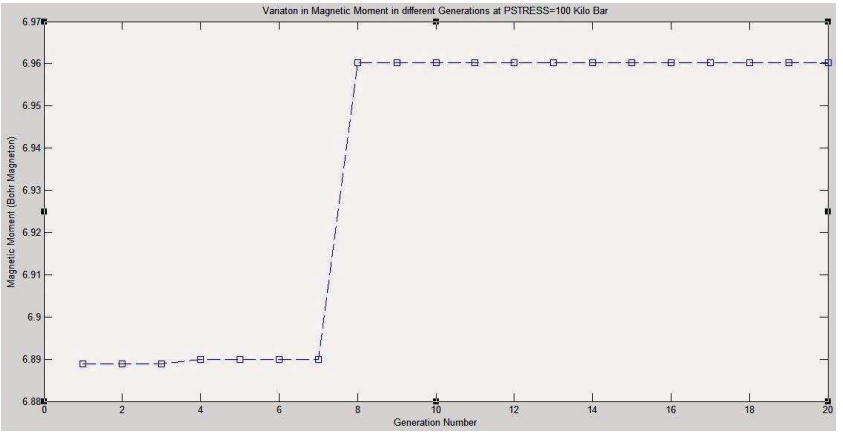


Figure 2.2 Variation in Magnetic Moment with Generation

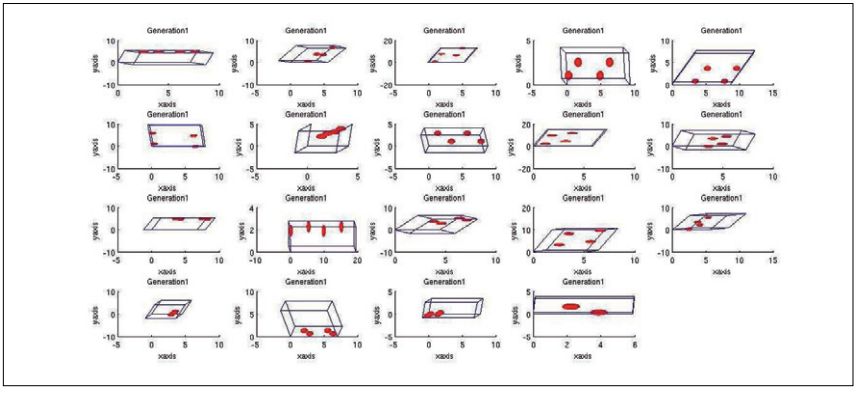


Figure 2.3 Structures in Generation 1st of calculation

2.5.2.2 Example 2: Magnetic Moment Based optimization for FePt system at a pressure of 500 Kilo Bar

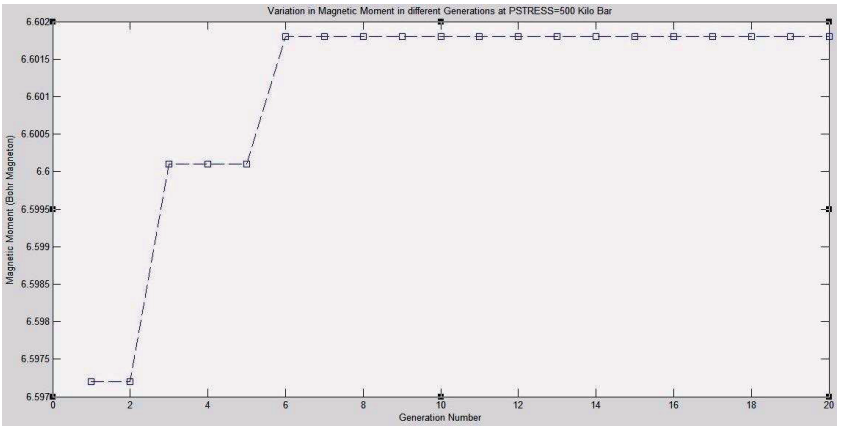


Figure 2.4 Variation in Magnetic Moment with Generation

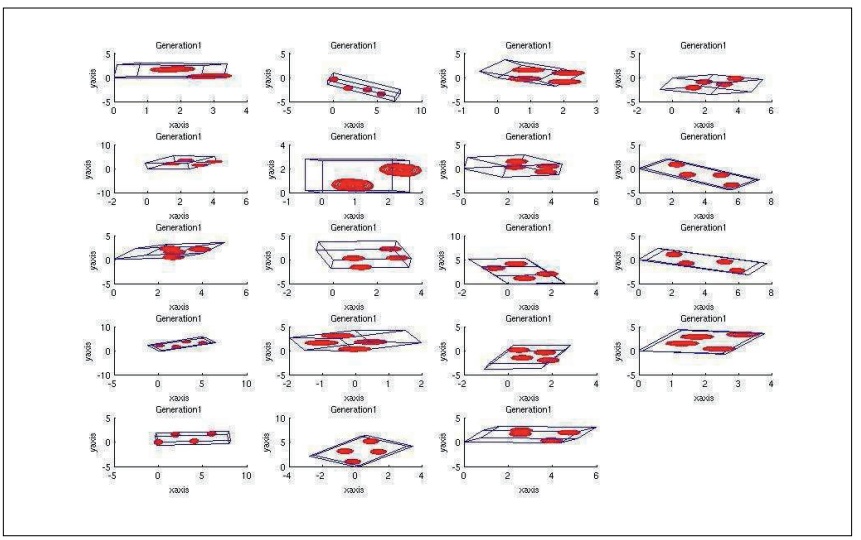


Figure 2.5 Structures in Generation 1st of calculation

2.6 Hardware and software selection

For design purpose no specific hardware requirement is needed, a computer with normal hardware configuration having Linux based operating system with Matlab and at least one of the software codes for ab initio calculation is sufficient. For the purpose of application, hardware configuration should be good to get results in reasonable time with accuracy.

2.7 Conclusion

It has been a great pleasure for me to work on this exciting and challenging project. This project proved good for me as it provided practical knowledge of programming in MATLAB. It also provided knowledge of the VASP (an ab initio calculation method) which is very useful. This will provide better opportunities in future for new projects.

2.8 Benefits

The project is identified by the merits of the system offered to the user. The merits of this project are that it will open a new way of research for the scientist. It will give a novel direction to scientist involved in magnetic studies, to find stable structure while maximizing the magnetic moment. This will have great applications.

2.9 Future Work

At present no. of ab initio codes are present. The magnetic moment based optimization is currently done for VASP code. The future work is to design the code for other ab initio code so that peoples using the code other than VASP can also be benefited from this new feature of magnetic moment based optimization.

2.10 References

[2.1] USPEX Website: http://han.ess.sunysb.edu/~USPEX/.

[2.2] USPEX Manual: http://han.ess.sunysb.edu/~USPEX/download.html.

[2.3] Maddox J. (1988), Crystals from first principles, Nature 335, 201.

[2.4] Jóhannesson G.H., Bligaard T., Ruban A.V., Skriver H.L., Jacobsen K.W., and Nørskov J.K. (2002). Combined Electronic Structure and Evolutionary Search Approach to Materials Design, Phys. Rev. Lett, 88, art, 255506.

Buy your books fast and straightforward online - at one of the world's fastest growing online book stores! Environmentally sound due to Print-on-Demand technologies.

Buy your books online at

www.get-morebooks.com

Kaufen Sie Ihre Bücher schnell und unkompliziert online – auf einer der am schnellsten wachsenden Buchhandelsplattformen weltweit!
Dank Print-On-Demand umwelt- und ressourcenschonend produziert.

Bücher schneller online kaufen

www.morebooks.de

OmniScriptum Marketing DEU GmbH
Bahnhofstr. 28
D - 66111 Saarbrücken
Telefax: +49 681 93 81 567-9

info@omniscriptum.com
www.omniscriptum.com

OMNIScriptum