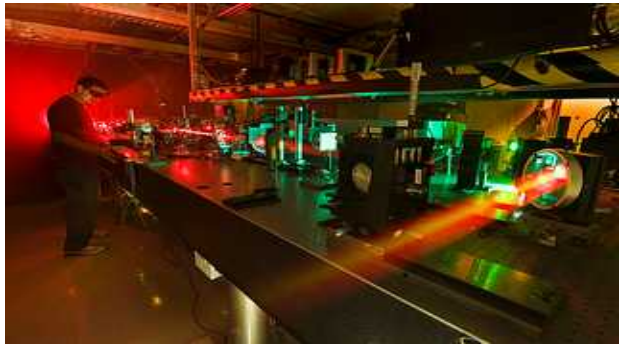
SAND2013-9760P

Exceptional service in the national interest

Sandia National Laboratories





Ion Trap Modeling And Simulation

14 November 2013

U.S. DEPARTMENT OF ENERGY | NNSA

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. SAND NO. 2011-XXXXP

Welcome and Introduction

Sandia National Laboratories

- Topics:
  - Visual Studio 2012 Quick Introduction
  - Geometry
  - Parametric Layout
  - Meshing
  - Boundary Element Solution
  - Control Solution
  - Flight Simulation
  - Optimization

Visual Studio Solutions

Sandia National Laboratories

- Work is grouped by solutions
- Solutions consist of one or more projects
- Each solution may contain at most one executable project

Visual Studio Settings

Sandia National Laboratories

- To set search path for header files:
  - Select project in solution explorer
  - Project->properties or right-click->properties
  - In left-hand pane, under Configuration Properties, select C/C++ then general
  - Add directory to Additional Include Directories
- To set search path for library (.lib) files:
  - From property dialog, select Linker then general
  - Add directory to Additional Library Directories
- To add a library as a dependency
  - From property dialog, select Linker then input
  - Type name of library in Additional Dependencies

Visual Studio Compile and Debug

Sandia National Laboratories

- To build the current solution, press F7 or select BUILD->Build Solution from the menu bar
- To start fresh, BUILD->Clean Solution then build the solution again
- To start debugging a project, press F5 or click the play symbol on the toolbar
- Set breakpoints by clicking to the left of the code in the gray area
- F10 single steps execution treating functions as atomic, F11 is the same but enters functions as they are called

Visual Studio Directories

Sandia National Laboratories

- Solutions are grouped into directories
- Each solution directory contains a Debug folder and a folder for each project
- The debug folder contains all projects' output files
- The project folders contain all source and header files

Visual Studio Hello World

Sandia National Laboratories

- To make a “Hello, World” project in visual studio:
  - FILE->New Project…
  - Select Win32 Console Application
  - Type “HelloWorld” in the Name entry field at the bottom
  - Click Finish
  - Click Finish on the next dialog
  - Write code to print “Hello, World” in _tmain function
  - Build the solution
  - Run either in debugger or bring up the DOS prompt
- Exercise: Write a Hello, World program
  - Advanced: Make a solution with 2 projects: an executable and a static library. Create format_hello in the static library and call from the .exe
  - Advanced: Create a test project that tests format_hello in the solution

BEDraw

Sandia National Laboratories

- Displays various file formats in 3D:
  - DoublePolygon files
  - GDSII files
  - Boundary Element Solution files
    - Includes colored differentiation of charge densities
- Basic Movement
  - Select polygons with shift+left click
  - Unselect by “selecting” any blank region
  - Zoom in on region with shift+drag
  - Zoom in and out with mouse wheel

BEDraw Cont.

- Basic Commands
  - Return to initial view with ‘a’
  - Show vertices with ‘v’
  - Show polygon information with ‘i’
  - Rotate with ‘o’ and arrow keys
  - Full listing under view->info
- Filtering
  - Found in view->index filter…
  - Display only data[0]==1 with “-1,1”
  - Reset with “-2”
  - Useful in GDSII files to view only certain layers
- Exercise

Geometry Library - DoublePolygon

Sandia National Laboratories

- Found in geometry.h

```
typedef struct DoublePolygon {
        struct DoublePolygon *prev;
        struct DoublePolygon *next;
        int n;                          // number of vertices
        int data[4];                    // implementer-defined metadata
        Point3D origin;                 // origin if local coordinates in global coordinates
        Point3D x;                      // local x-axis unit vector in global coordinates
        Point3D y;                      // local y-axis unit vector in global coordinates
        Point3D z;                      // local z-axis unit vector in global coordinates
        Point3DPtr vertex;              // vertices in local coordinates
} DoublePolygon, *DoublePolygonPtr;
```

Geometry Library

Sandia National Laboratories

- Manipulating DoublePolygon lists
  - AllocateDoublePolygon
  - FreeDoublePolygonList
  - DoublePolygonLinkLists
  - DoublePolygonUnlink
  - SaveDoublePolygonList
  - ReadDoublePolygonList

Geometry Library

Sandia National Laboratories

- Transforming and Creating Polygons
  - AllocateDoublePolygon
  - DoublePolygonRectangle
  - DoublePolygonRotate
  - DoublePoygonListReflectAboutXAxis
- Exercise

Fixing Geometry Problems

Sandia National Laboratories

- Imperfections can cause issues during meshing
- Common types of problems
  - Overlapping polygons
  - Overlapping edges
  - Nearly identical vertices
- Exercise

Parametric Layout - Parameter

- Found in BoundaryElement.h
- Read with ReadParametersFromTextFile

```
typedef struct Parameter {
        char name[128];         // unique identifier for parameter
        int fixed;              // fixed = not used in optimization
        double value;           // current value
} Parameter, *ParameterPtr;
```

Parametric Layout

Sandia National Laboratories

- Parametric Layout is used to programmatically generate a layout that can be varied by input parameters.
- Useful for optimization and for performing studies without redoing the layout
- Standard method is to write a function that creates a DoublePolygon list from parameters.
- Exercise

Meshing

Sandia National Laboratories

- Start with cleaned up geometry – “Pre-model”
- Automated triangularization
  - DoublePolygonListFractureLongestEdgeFracture
  - DoublePolygonListShortestDivide
- Meshing Steps
  1. Define region of interest by proximity to line segments
  2. Determine target triangle size
  3. Decide on rate of increase in triangle size with distance to line segments
- Exercise

Boundary Element Solution

Sandia National Laboratories

- Create matrix with BEMatrix
- Invert matrix using GaussJordan
- The results may be cached using WriteBEArraySolutionVer1
- Time consuming for large matrices
- Exercise

Charge Vectors

Sandia National Laboratories

- The inverted matrix is not used directly
- For each electrode to which a voltage is applied, a basis vector is created
- Potentials and Electric Fields may be computed as linear combinations of basis vectors.
- Charge vectors are saved in same file as matrix, although matrix is no longer necessary and may be omitted to save space
- Code to create charge vectors not yet in library
- Exercise

Voltage Array

Sandia National Laboratories

- Computing electric potential is slow
- A good approximation can be efficiently produced by sampling in a region and then interpolating
- These samples are stored in a voltage array file as a four-dimensional array over the electrodes and all three spatial dimensions
- ComputeVArrayMultithread works either from saved files or data arrays
- Exercise

Control Solution

- The control solution is a set of voltages applied to electrodes over time used to move an ion from one point to another
- Function written in C, code is called from Python using ctypes
- Programmer required to give a time interval between DAC steps and a position at each step
- Requires a text file mapping electrode indices in data[0] to names in the output file
- Stores output in results folder. Folder must exist as a subdirectory of the current directory
- Exercise

Flight Simulation

Sandia National Laboratories

- Flight Simulation uses entirely different techniques from the control solution generation
- Good sanity check
- Shows exactly what the ion is doing under the given conditions
- Output can be visualized using Plot
- Damping may be applied to simulate cooling
- Input is a text file with many parameters
  - Default values and explanations are provided in the exercise
- Exercise

Optimization Overview

- Optimization requires a parametric layout
- Finds the best parameters satisfying some property
- Property is expressed as a cost function, a weighted sum of various properties such as trap depth or trap frequency
- The literature contains many algorithms, we will use a simple one called “slew”

Cost Function

- Cost function is used to evaluate the utility of a trap layout
- General form: $\sum w_i V_i$ where V are values derived from the layout and w is a weight appropriated to each value
- Specific values must be computed by the user
- The function BEElectricField returns the electric field at a point for a given basis vector and can be used to derive other properties of the trap
- Exercise

Optimization – Slew pseudo code

Sandia National Laboratories

let $P = \{p_0, p_1, \ldots, p_{n-1}\}$ be the ordered list of parameters

let $P: p_i \leftarrow q$ denote $\{p_0, \ldots, p_{i-1}, q, p_{i+1}, \ldots, p_{n-1}\}$

for $i \in [0 \,..\, n-1]$

let C $= \{c_j : c_j = cost(P: p_i \leftarrow q_j)\}$

$q_0 \leftarrow p_i$

$q_1 \leftarrow p_i - s_i$

$q_2 \leftarrow p_i + s_i$

$b \leftarrow \frac{c_0 + c_2}{2}, d \leftarrow \left|\frac{c_2 - 2c_1 + c_0}{2}\right|$

$q_3 \leftarrow p_i + \frac{s_i \cdot b}{2 \cdot d}$

$P = P: p_i \leftarrow q_{argmax(C)}$

Optimization – Slew

- Note that the pseudo code is partially declarative; that is we define C before all its elements have values
- The algorithm iterates through all parameters, tries values in a certain neighborhood, and computes a fourth value as a guess of where the optimum may be given the current slope
- Exercise

Optimization – Putting it Together

Sandia National Laboratories

- Optimization Steps:
  - Optimization algorithm requests cost of a particular set of parameters
  - Cost function creates a trap layout from those parameters
  - Trap layout is meshed and run through the boundary element solver
  - The boundary element solution is used to compute properties of the trap layout
  - The cost function returns a weighted sum of those factors
  - The optimizer requests a cost with a different set of parameters based on the cost it received and the particulars of the algorithm
- Final Exercise