HEAT BUMP MODELING IN HIGH HEAT-LOAD X-RAY OPTICS*

E. Windisch IV, Wayne State, Detroit, MI, 48201, U.S.A.

Abstract

Thermal deformation in high heat load X-Ray optics limits coherence and reduces flux in the radiation that CHESS (Cornell High Energy Synchrotron Souce) scientists use for imaging. Empirical models of deformation have been created from measurements of the surfaces after exposure to the X-Ray beam, but no usable simulation software currently exists to model these deformations virtually. 'Heatbump', is a GUI-equipped MATLAB program designed to calculate three dimensional energy depostion in a solid. It uses an executable file from the existing program XOP to generate a spectrum, and a set of user input parameters to generate a text file of energy absorptions by position that can be entered directly into ANSYS for analysis.

XOP & WS.EXE

'Heatbump' uses the executable file 'ws.exe', a component of XOP 2.1, to create a spectrum of radiation that serves as the basis for all of the calculations performed. XOP 2.1 is a separate program designed by Manuel Sanchez del Rio and Roger J. Dejus that generates spectra based on synchrotron source parameters. 'Ws.exe' is the component that calculates spectra from Wigglers, which are the only source currently handled by 'heatbump'. Input parameters include: beam energy, beam current, period, number of periods, deflection parameters (K) for the x and y directions, minimum photon energy, maximum photon energy, number of energy steps, distance from source, x and y positions of the aperture, x and y slit widths and number of integrations points in the x and y directions. The spectrum that ws.exe generates is a table with two columns; the first being the photon energy and the second being power per .01 percent bandwidth of that photon energy. The table has as many rows as the number of energy steps set by the user, dividing up the energy range set by the minimum and maximum photon energies into equal steps.

HEATBUMP

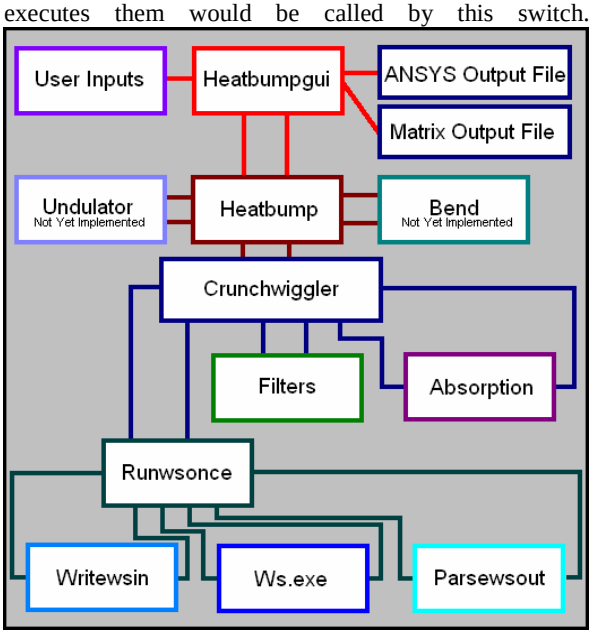
What makes 'heatbump' useful for modelling heat deformation accurately is its ability to generate power values as a function of position on a target surface. XOP runs 'ws.exe' once, generating only a single spectrum and a total power number for an aperture of a fixed size, centered at a given position relative to the center of the beam. 'Heatbump,' conversely, runs 'ws.exe' repeatedly, using small apertures and varying their positions for each iteration, until it has covered the entire surface of the target. This process generates spectra and total power figures in Watts for every location on the surface. The way 'ws.exe' was written, however, allows it to accept only positive x and y values for the position of the center of the aperture it's analyzing. For this reason, heatbump scans only a quarter of the surface (where x and y are both positive) and then mirrors the data three times to create a picture of the entire surface. This method makes the reasonable assumption that the radiation is symmetric about both the x and y axes.

PROGRAM STRUCTURE

'Heatbump' calls seven subfunctions to perform various tasks during its execution. A list with descriptions of these functions, including 'heatbump' itself, is given below.

'Heatbump'

Heatbump is the main calling function for the program and includes a 'switch' that determines the course of the program based on the source of the radiation. At this time, only wigglers are supported, but if bend magnet and undulator source capabilities were added, the code that executes them would be called by this switch.



FLOW CHART OF PROGRAM

'Crunchwiggler'

Crunchwiggler organizes and tabulates all of the data used in the program. It creates the necessary loops to call all of the computational subfunctions, including 'ws.exe', the proper number of times, sending the correct data to each subfunction and collecting and formatting the output data.

*Work supported by NSF

'Runwsonce'

Runwsonce is a short simple script whose function is to coordinate all of the tasks required to run 'ws.exe' once. It first calls Writewsin, then it sends a direct command to the operating system to run 'ws.exe' using the MATLAB ! operator (bang) and finishes by calling parsewsout to retrieve the data from 'ws.out'. 'Ws.out' is the output file created by ws.exe every time it runs.

'Writewsin'

In order to run, 'ws.exe' needs an input file containing all of its input parameters in a particular format. This file is usually created by the GUI of XOP, but because 'heatbump' uses 'ws.exe' without XOP, heatbump itself must generate an input file for ws.exe to read when it starts. Writewsin creates a text file with extension .inp that contains all of the information ws.exe needs to run, in a format that ws.exe can read. Writewsin uses vales taken from the GUI (where they were set by the user) and from Crunchwiggler to generate this file.

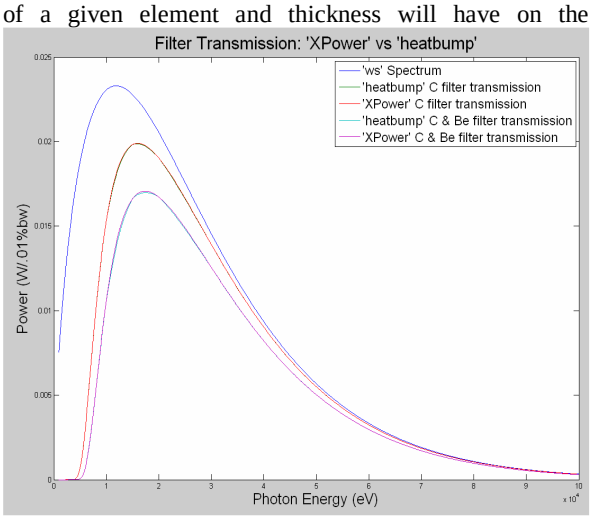
'Parsewsout'

When ws.exe runs it generates a text file called 'ws.out' that contains all of the data it generates during its execution. Included in this data is a simulated spectrum of radiation emitted by a wiggler. Parsewsout extracts this information, which is formatted into two columns, and returns it to runwsonce, which in turn returns it to curnchwiggler.

'Filters'

To create as accurate a model as possible, spectral attenuation filters have been incorporated into the code. Their thicknesses are GUI input parameters and can be set to simulate an experimental setup.

'Filters' is designed to account for the effect that filters of a given element and thickness will have on the wiggler's output spectrum. 'Filters' calculates attenuation for each energy level of photon in the spectrum and returns the attenuated spectrum to 'crunchwiggler'.



FILTER TRANSMISSION: 'HEATBUMP VS. XPOWER

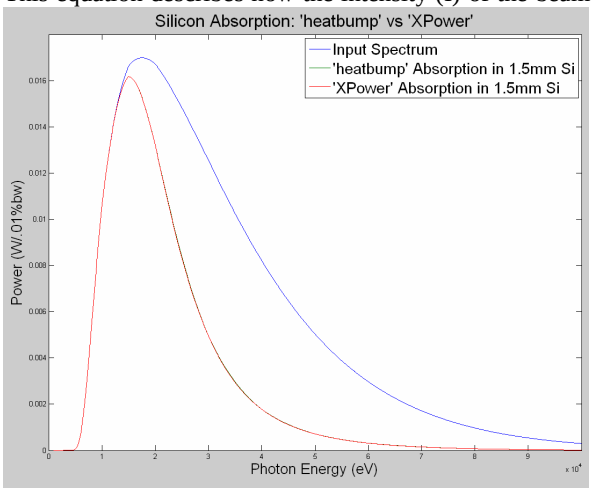
Currently 'filters' supports filters of either Carbon or Beryllium. These two elements are built in to the GUI as they are the standard filters in the CHESS West A2 hutch setup. Filters of elements other than Carbon and Beryllium can be incorporated into the program easily by making a few additions to the code in the 'filters' subscript. 'Heatbump' already passes all the required information to 'filters', making the code easily scaleable.

'Absorption'

Once 'heatbump' has calculated a filtered power for each location on the surface, 'absorption' propagates that power through the volume of material below. It performs these calculations in much the same way as 'filters' performs its calculations, but instead of returning the amount of radiation that passes through the material, it returns the amount of the radiation absorbed in Watts. 'Absorption' performs its calculations using the following relation:

$$I = I_o e^{-\alpha x}$$

This equation describes how the intensity (I) of the beam



decreases in the medium as a function of depth (x). Alpha (α) is the mass attenuation coefficient for the absorbing element. The mass attenuation coefficient is a measure of how "absorbent" the material is for a given energy level of photon. 'Abosrption' interpolates a value for alpha from the data on the NIST website (http://physics.nist.gov/PhysRefData/XrayMassCoef/tab3.html) for each photon energy in the range for its calculations.

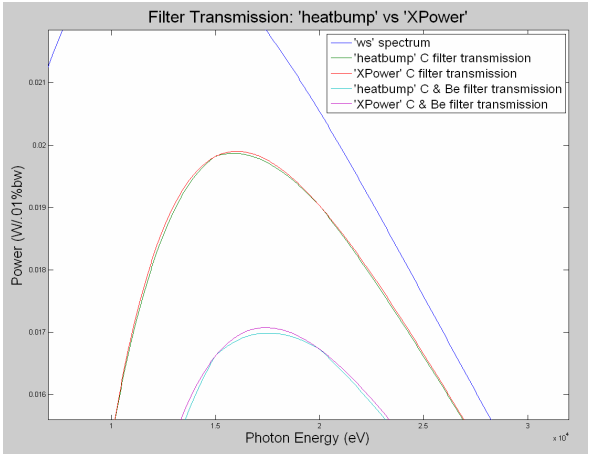
'Absorption' can calculate the amount of energy absorbed in a "slice" of the material whose thickness is defined by depths x1 and x2 using one of the following relations:

$$I = I_o e^{-\alpha x1} - I_o e^{-\alpha x2} \quad or \quad I = I_o(e^{-\alpha x1} - e^{-\alpha x2})$$

'Absorption' repeats this process of calculating energy absorbed "per slice" of material until it has reached the depth set by the user, generating a one dimensional plot of energy absorbed as a function of depth in the material. The number of divisions and the total depth to be probed are both input parameters of the program. These, in conjunction with two scaling parameters that are also user inputs, determine the thicknesses of the "slices". The first scaling parameter, called "Z Mesh Scale" can be set to "lin" to make all of the depth subdivisions the same height, or "log" to make them progressively taller as depth increases. The second scaling parameter, called "Spacing Ratio", sets the ratio of the height of the last brick to the height of the first. In the "lin" setting, the mesh heights are just the total depth to be probed divided by the number of subdivisions and the Spacing Ratio is not used. When the "log" setting is used, the Spacing Ratio value is used to calculate the factor between adjacent mesh thicknesses, resulting in very thin meshes near the surface and thicker meshes near the total depth set by the user. The heights are scaled this way so that the user can have high resolution near the surface, where most of the energy is absorbed, without having a computationally prohibitive number of divisions, while still probing deep enough to make a comprehensive model.

CHECKS AND INTERPOLATION

'Heatbump' currently uses interpolated values for mass attenuation coefficients in three places during its execution. The first two instances occur in 'filters' for calculating transmitted energy. The last is in 'absorption', during the calculation of absorbed energy in the target. In each case, the interpolated values are generated by the 'interp1' MATLAB function. The data tables that NIST provides are scaled logarithmically and so 'interp1' was given the log of each data point and the output was exponentiated to generate the most accurate values possible.

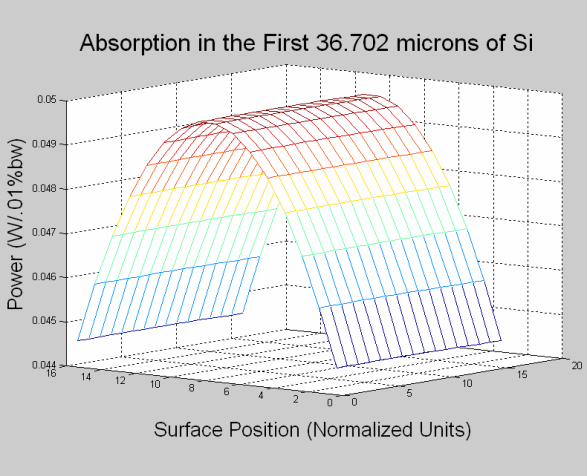


CLOSEUP OF FILTER TRANSMISSION: 'HEATBUMP VS. XPOWER'

To insure that the scripts were functioning correctly, their outputs were checked against another component of XOP: 'XPower'. 'XPower' calculates transmission and absorption of power through a solid material. The plots on this page and the previous page are the results of comparing the output spectra of 'filters' and 'absorption' to the spectra produced by 'XPower', given the same parameters. The agreement is excellent, with a maximum discrepancy of less than 1%.

ENERGY DEPOSITION

Once 'absorption' has plotted energy absorption as a function of depth, 'crunchwiggler' is able to assemble the data into a three dimensional representation of absorbed energy as a function of position on the surface and depth. For each "slice" of material, 'crunchwiggler' produces a matrix of absorption data, the dimensions of which are designed to represent the x and y positions on the sample. 'Crunchwiggler' has graphing capabilities for these matrices which are commented out of the code but can be reincorporated easily.



SAMPLE OF HEATBUMP'S GRAPHING OUTPUT

OUTPUT

In the course of running, 'heatbump' produces two output text (.txt) files. The first contains all of the output data, in the form of energy absorbed per unit time and volume (W/mm^3), in a specially formatted column vector. This vector is designed to be pasted into an ANSYS command line for running simulations. The order of the values in the column vector is very important. When running a simulation in ANSYS, the sample material is divided into a given number of small regions that are numbered consecutively by ANSYS. The vector of absorption values generated by 'heatbump' needs to correspond to this numbering scheme so that the absorption data maps correctly into the simulation. Each line of the output vector is also formatted like an ANSYS command. A sample line of data looks like this:

BFE,1,HGEN,,0.01182614413,,,,

The acronyms and extra commas are for setting ANSYS parameters. The number 1 is the number of the

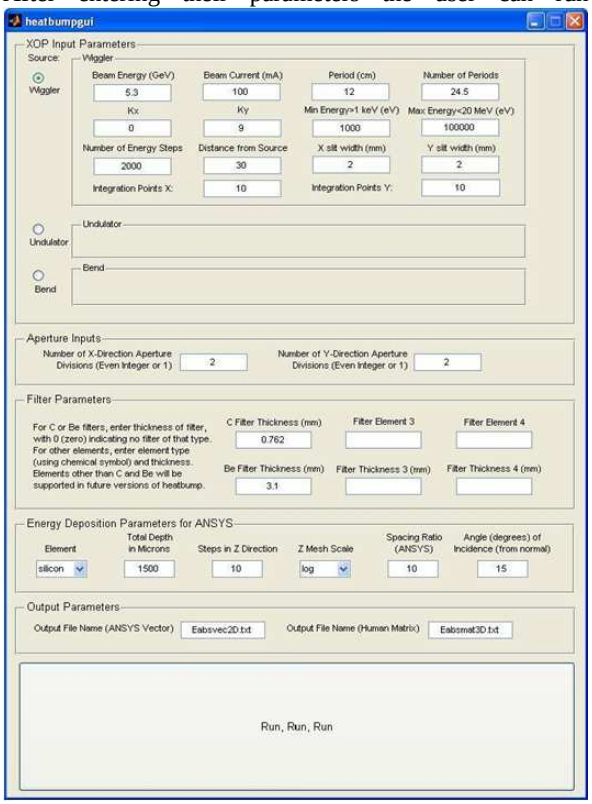
"block" that this data corresponds to and the last number is the wattage.

USING THE PROGRAM

Calling 'heatbumpgui.m' at the MATLAB command prompt launches the GUI. From that point, the user need only input the desired parameters and click 'run'. The GUI calls 'heatbump' and runs the entire program. Help documentation for each subfunction can be accessed by typing 'doc *subfunction name*'.

GUI

In order to make it as easy to use as possible, 'Heatbump' has a simple graphical user interface (GUI) generated by the script 'heatbumpgui.m'. The parameters that have limits on their values are noted in the GUI itself to avoid errors and nonsensical results. After entering their parameters the user can run



THE 'HEATBUMP' GUI

'heatbump' with a mouse click. The GUI figure is designed to close upon completion of 'heatbump'. Running the program without entering any text sends the default parameters (which are displayed in the text boxes) to the program.

RESULTS

At the time of writing, 'heatbump' data has been used to create an ANSYS model of heat deformation in the target. This model has not yet been checked against the existing experimental models of deformation in similar conditions. The initial indications are that the program's data, and the heat deformation that results from it, are in fair agreement with the observed phenomena. Hopefully 'heatbump' will become a useful tool for CHESS scientists.

ACKNOWLEDGEMENTS

This project was made possible by the direction and support of Ivan Bazarov and Peter Revesz of Cornell University. I personally wish to thank them both for their guidance during the course of the project and the opportunity to work with them this summer. I also would like to thank Jim Savino for his help with ANSYS, the project would have had no practical application without him. Thanks also to Rich Galik and Claude Pruneau for their work running the program and for all the help we received from them this summer. Finally, tremendous thanks are due to the National Science Foundation for funding this research under grant number PHY-0353994.

REFERENCES

[1] NIST Interpolation data taken from: http://physics.nist.gov/PhysRefData/XrayMassCoef/tab3. html

[2] MATLAB Help File