Adding Diagnostics to XOOPIC

1 A top level explanation of how XOOPIC runs

The XOOPIC GUI, XGrafix, controls the simulation once it is running. Based on user input, it displays diagnostics or runs the simulation.

XGrafix causes the simulation to advance a timestep by calling the XGMainLoop() function, (in xgmain.C) which in turn calls Physics(), which executes the numerical model of the simulation, and Maintain_Diagnostics(), which extracts information from the physics model for examination.

Summarizing: The user controls XGrafix, XGrafix calls/controls XGMainLoop, which calls Physics() and Maintain_Diagnostics().

It is in Maintain_Diagnostics that one would process raw Physics() data, such as taking FFT's of electric fields on the mesh, or maintaining a history of some quantity. Maintain_Diagnostics in turn calls history(), which maintains quantities as a function of time.

2 Files to modify to add a diagnostic

These are the files which are relevant to initializing and maintaining a diagnostic:

xg/diagn.C: This file contains the function Set_Diagnostics(), which gets initializes the diagnostic arrays and sets up histories, and gets

xg/initwin.C: This file contains the calls to XGrafix2.0, which set up the graphics windows for the diagnostics.

xg/diagn.h: declarations and definitions for the diagnostics. It is also a means of giving initwin.C access to variables declared in diagn.C.

For some diagnostics, you might have to modify the files in the physics/ to have them provide additional data for your diagnostics.

3 Example of adding a simple diagnostic

Suppose a time history of the electrostatic potential at a point A is needed. This is a list of the things that would need to be done to add the diagnostic:

3.1 Step 1

in xg/diagn.C:

At the top of the file, declare a place to store the time history:

Scalar *phi_history_at_A;

3.2 Step 2

In the function Set_Diagnostics in xg/diagn.C, allocate space for this diagnostic and write zeros to the newly allocated memory.

phi_history_at_A = new Scalar[HISTMAX+1]; //allocate memory
//initialize it to zero:
memset(phi_history_at_A,0,(HISTMAX+1)*sizeof(Scalar));

It is important to initialize memory to zero: XGrafix may operate on the data before it is valid, (such as trying to display the history before the simulation has advanced a timestep), and many architectures will crash if you try to access invalid floating-point data.

3.3 Step 3

In the history() function, add the line:

phi_history_at_A[hist_hi] = phi[jm/2][km/2];

Add this line near the line:

energy_e[hist_hi] = Uetot;

in the history() function. A history of phi at the grid location (jm/2,km/2) will now be maintained in the array "phi_history_at_A". "jm" and "km" are the maximum values of j and k respectively, so this diagnostic will take the

potential history near the middle of the device.

3.4 Step 4

Also in the history function, add the line **phi_history_at_A[i] = phi_history_at_A[k];**

near the line:

energy_e[i] = energy_e[k];

Since the time history is taken through the entire simulation, but the phi_history array has only HISTMAX+1 entries, we must "comb" the time history periodically. This line ensures that this diagnostic will be properly combed.

3.5 Step 5

Add this line to diagn.h:

extern Scalar *phi_history_at_A;

3.6 Step 6

Make the call to XGrafix in initwin.C which will set up a window to display this data in.

In initwin.C, add these lines to the InitWin() function:

XGSet2D("linlin","phi", "Potential history at point A", "closed", 700,400, 1.0,1.0,True,True,0,0,0.0,0.0);
XGCurve(t_array, phi_history_at_A, &hist_hi, 1);

These function calls are from the XGrafix 2.0 library. Refer to the manual for XGrafix 2.0 for a description of XGrafix and these function calls.

3.7 Step 7

Recompile xoopic and examine the diagnostic to see if you have carried out the previous steps correctly. Adding this diagnostic took 9 lines of code.