TEMPEST
version 5.0

by

Alfred Wong and Tom Pistor

Memorandum No. UCB/ERL M98/50

20 August 1998

TEMPEST
version 5.0

by

Alfred Wong and Tom Pistor

Memorandum No. UCB/ERL M98/50

20 August 1998

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

TEMPEST version 5.0

Alfred Wong and Tom Pistor

Electronics Research Laboratory

University of California, Berkeley

1.0 Introduction

1.1 Overview

The acronym TEMPEST3 stands for ``Time-domain Electromagnetic Massively Parallel Evaluation of Scattering from Topography." The computer program solves Maxwell's equations using a time-domain finite-difference algorithm, where the electric and magnetic field nodes are spatially and temporally staggered over a three-dimensional topography of interest. Version 3.0 takes advantage of the inherent parallel nature of electromagnetic wave propagation and is implemented on the computer architecture connection machine 5 (CM-5)8. Due to the limited availability of the CM-5, version 4.0 is implemented on any single-processor computer architecture such as a work station or even a personal computer. Version 5.0 is also intended for single processor architectures but offers several improved features. The simulation domain may represent periodic, symmetric or isolated topographies. The algorithm is capable of simulating problems such as scattering from asymmetrical alignment marks, transmission through phase-shifting masks, effects of line-edge profiles in metrology as well as dynamic bleaching of photoresist over arbitrary non-planar, inhomogeneous wafer topographies.

Illumination is assumed to be monochromatic, with the electric field linearly polarized in any user-specified direction. The incident angle can take on discrete values depending on the illumination wavelength and the dimension of the simulation domain. Illumination is assumed to be coherent and can consist of any intensity and phase profile such as that calculated from SPLAT9 or other computer programs. The photosensitizer concentration is calculated if photoresit is present. The matrix containing this concentration information can be used in other simulation programs such as SAMPLE4 for simulation of resist development.

TEMPEST parses topography information from an input file which can be checked for correctness. The input geometry is then simulated until the electromagnetic field reaches steady-state or, in the case of non-convergence, the simulation domain is excited for a user chosen number of wave cycles. Information on the simulation parameters is written to an output file. Topography and field data are written to files where they can be analysed.

1.2 New in Version 5.0

Version 5.0 of the program TEMPEST has several extensions from the previous version (version 4.0). First and foremost is TEMPEST's new ability to update nodes representing different materials with different updating equations. This allows more computation time and memory to be devoted to nodes which represent complicated materials (such as a dispersive metal) rather than wasted on nodes representing simple materials (such as air or glass).

This localization of the updating equations leads to a new implementation of boundary conditions. TEMPEST 5.0 treats boundary conditions merely as nodes with ``special" updating equations. This conveniently allows the implementation of the newly invented boundary condition, the PML boundary condition[references].

Other side effect improvements:

PML boundary condition now allows the simulation of fully isolated topographies.

Excitation is no longer limited to a single planar distribution of excited nodes near the top of the domain. Plane sources with x,y and z directed normals and point sources can be placed anywhere in the domain.

Convergence checking considers the convergence status of a 3D grid of points distributed evenly throughout the domain rather than a single plane near the top. This helps to avoid false convergences.

Output data is no longer written in the PLOTMTV format as in version 4.0 but rather in a binary data format which requires much less disk space and is more portable.

Philisophical Changes:

Rather than writing output data (such as the steady state fields and refractive index) in the PLOTMTV format it is written in a binary format which uses much less space and can be read by a number of other programs. One such program is the MATLAB[ref] program. The new philosophy is to use MATLAB to analyse the data since MATLAB is a well developed, and widely used program. Several MATLAB script files have been developed for this purpose.

Move the TEMPEST-SPLAT interface out of TEMPEST and into MATLAB script files.

1.3 Applications Information

TEMPEST is capable of simulating a number of optical phenomena occurring in photolithography and optical metrology applications. These include bleaching of photoresist over arbitrary topography (e.g., reflective notching), dark-field and bright-field imaging of wafer features (e.g., alignment marks and line structures), generation of image profiles through masks (e.g., reduction phase-shift mask), and calculation of the electric and magnetic fields at all points in the three-dimensional simulation domain at any instant in time. Several studies involving these topics have already been performed, and the user is encouraged to refer to these publications[2,3,6,7,11,12,13,14]. In general, TEMPEST simulations can be placed into two categories:

Lithography which involves dynamic changes during exposure, and

Scattering analysis.

These categories are distinguished primarily by the desired output. For lithography the desired output is the final PAC concentration. For scattering purposes the primary output desired are the electromagnetic fields and the diffraction harmonics.

1.4 Source Code Information

TEMPEST ver5.0 is written in the C programming language. The source code is distributed into twelve text files. Six header files are also included. The header files are:

defn.h constant definitions
global.h global variable definitions
header.h header files to be included
lib.h libraries to include
routn.h subroutines used in the program
ver.h version and release date information

The files which contain the source code are:

bulk.c interior updating equations
in.c reads information from the input file
init.c initialization of variables
iterate.c fields computation by iteration
loop.c calls different loops according to the model
main.c program control
math.c mathematical functions
misc.c miscellaneous functions
out.c prints simulation results to output files
parproc.c communications for parallel processing
pml.c prints simulation results to output files
resist.c models photoresist bleaching

Several MATLAB script files also have been written and are used to view the output data:

plotbin.m view binary data
plotbinsten.m view binary data and stencil in topography
plotam.m view field time averaged amplitude
plotamsten.m `` with topography stenciled in
plotam6.m view total field
plotam6sten.m `` with topography stenciled in

1.5 About this Manual

Section 1.0 Introduction

A general overview of TEMPEST is provided.

Section 2.0 Installation

This section discusses how the simulation program TEMPEST is compiled and installed.

Section 3.0 Tutorial by Example

Two tutorial examples to let the user become familiar with the commands and procedures of running TEMPEST are given

Section 4.0 Detailed Command Descriptions

Detailed descriptions of the commands, input file keywords and MATLAB script commands are presented in this section.

Section 5.0 Useful Hints

Some useful hints for running TEMPEST are given.

Section A.0 Useful Hints
Section B.0 Useful Hints

The conventions used throughout this users' guide are as follows:

Words which are in ALL CAPITAL LETTERS represent programs or shell scripts.

Words in bold letters represent commands to be typed exactly as it appears on the users' guide.

Words in italics represent commands or arguments which can take on different values or character strings.

Words inside [square brackets] are options to commands which may be omitted.

Words in the Courier font represent characters in a file or output from programs.

The string ``ws%" represents the unix prompt on a workstation.

The string ``pc>" represents the DOS prompt on a PC.

The string ``prompt:" represents either and unis or DOS prompt.

The string ``>>" represents the MATLAB prompt.

2.0 Installation

2.1 Resource Requirements

The computing environment of the user should:

be capable of compiling and running C programs (UNIX or DOS).

have at least 20 mega bytes of disk space available to hold the data files

have MATLAB ver 4.0 or higher software. (The user could write his/her own script files for another program or even write their own analysis programs.)

2.2 Installation on a UNIX Machine

This software package consists of the main program TEMPEST as well as other supporting routines to be run on the user's workstation. All the programs except for SPLAT are written in the programming language C.

TEMPEST (Program for electromagnetic simulation.)

AMPHA1D and AMPHA2D (Programs which calculate the amplitudes and phases of the electromagnetic fields from the instantaneous field values.)

CTR2MTV and DPL2MTV (Programs which convert CONTOUR and DRAWPLOT data format to PLOTMTV data format.)

FACTOR1D and FACTOR2D (Programs which scale the axes of plot files.)

REDUCE2D (Program for reducing data file size.)

TOTAL1D and TOTAL2D (Programs which calculates the total electric or magnetic field from the values of the x, y, and z components.)

PLOTMTV (Program for displaying data files.)

SPLAT (Program written in Fortran for calculating aerial images.)

To install TEMPEST, move to a directory (say $HOME/bin) where TEMPEST should reside. Then insert the tape into the tape drive, and type the following command:

```
ws% tar xv
```

After tape reading is completed, which takes about 5 minutes, a TEMPEST directory is created in the $HOME/bin directory. The programs are organized as follows:

2.3 Compilation

2.3.1 TEMPEST

To compile TEMPEST, follow the steps:

```
ws% cd $HOME/bin/TEMPEST/code

cm% make
```

3.0 Tutorial by Example

The easiest way to learn how to use TEMPEST is by example. This chapter gives a brief overview of how to use TEMPEST and then presents a few example simulations.

3.1 Overview

The basic steps in running TEMPEST simulation are shown as follows:

The input file to TEMPEST contains five types of specifications:

simulation domain

illumination

topography

analysis (numerical parameters)

data (output specifications)

The input file to TEMPEST can be given any name and the entries in the input file can occur in random order. The input file to TEMPEST consists of a list of keywords; each keyword may be followed by numbers, words, or other keywords. A detail description of the keywords is found in Section 4.1. Comments can be included in the input file by including the string ``/*" before the comment and the string ``*/" after the comment.

To execute the program TEMPEST, the user must supply two arguments: the first being the name of the input file, and the second is the name of the output file to which information concerning the simulation is to be written.

The amount of memory needed to run the program is proportional to the number of simulation nodes in the domain, i.e., the product of the number of nodes in the x-, y-, and z-directions. As a rule of thumb, a domain with 1 million nodes (100 by 100 by 100) requires about 30 MBytes of memory (30 bytes/node). A simulation can take anywhere from a few minutes to several hours depending on the size of the problem.

3.2 Example 1 - Contact Hole

As a first example, transmission through a 1X contact hole is studied. The input file is located in the directory tempest/examples/ and is named hole.in. The dimension of the simulation domain is 1.0 mm by 1.0 mm by 0.25 mm. (The simulated volume is 0.5 mm by 0.5 mm by 0.25 mm via the use of symmetry.) The chrome mask has a glass substrate with a layer of 80 nm thick chrome. The square opening on the chrome has a width of 0.25 mm on each side. Illumination is assumed to be normally incident at 248 nm, with the electric field linearly polarized in the x-direction.

The input file hole.in is shown as follows:

```
/*
input file name: hole.in
*/
/*
The structure is symmetric with respect to both the x- and y-axes.
*/
x_symmetry
y_symmetry
/*
The simulation window is 0.5 by 0.5 by 0.25 micron with 70 simulation nodes
in the x dirction. TEMPEST automatically calculates the number of nodes in the other directions
*/
x_node 70
z_node 51
x_dim 0.5
```

```
    y_dim 0.5
    z_dim .3642857
    wavelength 0.248
    plane_source xy position 0.0 0.5 0.0 0.5 0.2286 1 0 0 1 0 uniform 0 0
    /* Block #0 is glass with index of (1.5, j0.0). */
    rectangle position 0.0 0.5 0.0 0.5 0.0 0.25 index 1.5 0.0
    /* Block #1 is air with index of (1.0, j0.0). */
    rectangle position 0.0 0.5 0.0 0.5 0.0 0.10 index 1.0 0.0
    /* Block #2 is chromium with index of (2.5, j2.0). */
    rectangle position 0.0 0.5 0.0 0.5 0.10 0.18 index 2.5 2.0
    /* Block #3 represents the contact hole opening (air) with index of 1.0. */
    rectangle position 0.375 0.5 0.375 0.5 0.10 0.18 index 1.0 0.0
    rectangle node 0 69 0 69 35 42 pml 0 0 1 2.25 1 0 0
    rectangle node 0 69 0 69 43 50 pml 0 0 -1 1 1 0 0
    /*
    rectangle node 0 69 0 69 0 0 bc -z 1 0
    rectangle node 0 69 0 69 34 34 bc +z 1.5 0
    */
    /* The instantaneous electric field in the x-direction of a plane just
    underneath the opening is requested at two instants of the wave cycle. */
    plot block node 0 69 0 69 20 20 hole.xy.20.blk
    plot block node 0 69 35 35 0 50 hole.zx.35.blk
    plot ex steady 0.00 node 0 69 0 69 15 15 hole.xy.15.e.x.i
    plot ex steady 0.25 node 0 69 0 69 15 15 hole.xy.15.e.x.q
    plot ex nonsteady 1.00 node 0 69 35 35 0 50 hole.zx.35.e.x.1.00
    plot ex nonsteady 2.00 node 0 69 35 35 0 50 hole.zx.35.e.x.2.00
```

To run TEMPEST, type

```
prompt: tempest hole.in hole.out
```

This will produce the following screen output:

```
    T E M P E S T
    ver5.0
    May 1997
    Starting TEMPEST ...
    Running single process simulation.
    Reading input file [hole.in] ...
    Done reading input file [hole.in].
    Initializing parameters and variables ...
    Number of flux components: 470400
    Number of resist nodes: 0
    Max index: 2.5000, Min index: 1.0000
    dx: 0.0071um
    time step after adjustment: 1.292561e-17
    Warning:
    Minimum number of simulation nodes per wavelength for the topography
    is 13.89, fewer than 15. Simulation result may be inaccurate.
    Done initializing parameters and variables.
    Memory allocated: 8945792
    Writing plot file [hole.xy.20.blk] ...
    Writing plot file [hole.zx.35.blk] ...
    Convergence error after 1 cycles: 20.36% (58s)
    Writing plot file [hole.zx.35.e.x.1.00] ...
    Convergence error after 2 cycles: 37.66% (132s)
    Writing plot file [hole.zx.35.e.x.2.00] ...
    Convergence error after 3 cycles: 42.81% (75s)
    Convergence error after 4 cycles: 19.81% (44s)
    Convergence error after 5 cycles: 13.08% (56s)
    Convergence error after 6 cycles: 4.59% (64s)
    Convergence error after 7 cycles: 3.15% (64s)
    Convergence error after 8 cycles: 2.69% (64s)
    Convergence error after 9 cycles: 1.95% (63s)
    Convergence error after 10 cycles: 0.79% (64s)
    Convergence error after 11 cycles: 0.37% (63s)
    Convergence error after 12 cycles: 0.70% (64s)
    Convergence error after 13 cycles: 0.28% (65s)
    Convergence error after 14 cycles: 0.09% (63s)
    Success after 15 cycles (63s).
    Success after 16 cycles (65s).
    Convergence error after 17 cycles: 0.09% (65s)
```

```
Success after 18 cycles (62s).
Success after 19 cycles (61s).
Success after 20 cycles (62s).
Writing plot file [hole.xy.15.e.x.i] ...
Writing plot file [hole.xy.15.e.x.q] ...
Writing to output file [hole.out] ...
Done writing.
Program executed successfully.
TEMPEST run time: 0:22:14.
Have a nice day.
```

Seven files are generated by this simulation:

hole.out (output file which contains information on the simulation run)

hole.xy.20.blk

hole.zx.35.blk

hole.zx.35.e.x.1.00

hole.zx.35.e.x.2.00

hole.xy.15.e.x.i

hole.xy.15.e.x.q

3.3 Example 2: 2D PML with planar segment excitation

This example demonstrates the use of multidimensional PML, 2D simulation, planar segment excitation, isolated topography and dispersive materials.

```
/*
input file name: pml2.in
Testing PML2D
May 16/97
*/
x_node 61
y_node 2
z_node 122
x_dim .854
err_tol .2
wavelength .365
plane_source xy node 15 45 0 1 101 0 1 0 1 0 uniform 0 0
rectangle position 0 .84 0 .028 0 1.68 index 1 0
rectangle node 20 40 0 1 20 40 dispersive 1.2 3.4
rectangle node 0 9 0 1 0 121 pml -1 0 0 1 1 0 0
rectangle node 51 60 0 1 0 121 pml 1 0 0 1 1 0 0
rectangle node 0 60 0 1 0 9 pml 0 0 -1 1 1 0 0
rectangle node 0 60 0 1 112 121 pml 0 0 1 1 1 0 0
rectangle node 0 9 0 1 0 9 pml -1 0 -1 1 1 0 0
rectangle node 51 60 0 1 0 9 pml 1 0 -1 1 1 0 0
rectangle node 51 60 0 1 112 121 pml 1 0 1 1 1 0 0
rectangle node 0 9 0 1 112 121 pml -1 0 1 1 1 0 0
plot ey nonsteady 0.75 node 0 60 0 0 0 121 pml2.zx.0.e.y.0.75
plot ey nonsteady 1.00 node 0 60 0 0 0 121 pml2.zx.0.e.y.1.00
plot ey steady 0.00 node 0 60 0 0 0 121 pml2.zx.0.e.y.i
plot ey steady 0.25 node 0 60 0 0 0 121 pml2.zx.0.e.y.q
plot refractive node 0 60 0 0 0 121 pml2.zx.0.ref
plot block node 0 60 0 0 0 121 pml2.zx.0.blk
```

The TEMPEST screen output:

```
T E M P E S T
ver5.0
May 1997
Starting TEMPEST ...
Running single process simulation.
Reading input file [pml2.in] ...
Done reading input file [pml2.in].
Initializing parameters and variables ...
```

```
Number of flux components: 46566
Number of resist nodes: 0
Max index: 1.2000, Min index: 1.0000
dx: 0.0140um
time step after adjustment: 2.536477e-17
Done initializing parameters and variables.
Memory allocated: 689012
Writing plot file [pml2.zx.0.ref] ...
Writing plot file [pml2.zx.0.blk] ...
Writing plot file [pml2.zx.0.e.y.0.75] ...
Convergence error after 1 cycles: 18.15% (6s)
Writing plot file [pml2.zx.0.e.y.1.00] ...
Convergence error after 2 cycles: 32.62% (6s)
Convergence error after 3 cycles: 30.46% (4s)
Convergence error after 4 cycles: 24.31% (4s)
Convergence error after 5 cycles: 24.00% (3s)
Convergence error after 6 cycles: 7.08% (4s)
Convergence error after 7 cycles: 1.85% (4s)
Success after 8 cycles (4s).
Success after 9 cycles (4s).
Success after 10 cycles (3s).
Writing plot file [pml2.zx.0.e.y.i] ...
Writing plot file [pml2.zx.0.e.y.q] ...
Writing to output file [pml2.out] ...
Done writing.
Program executed successfully.
TEMPEST run time: 0: 0:46.
Have a nice day.
```

Now here's a MATLAB session to view the output files:

```
prompt: matlab
< M A T L A B (R) >
(c) Copyright 1984-94 The MathWorks, Inc.
All Rights Reserved
Version 4.2c
Dec 31 1994
Commands to get started: intro, demo, help help
Commands for more information: help, whatsnew, info, subscribe
>> subplot(2,2,1);
>> plotbin('pml2.zx.0.blk'); title('Block #');
size1 =
61
1
122
>> subplot(2,2,2);
>> plotbin('pml2.zx.0.ref'); title('Ref. Index');
size1 =
61
1
122
>> subplot(2,2,3);
>> plotbin('pml2.zx.0.e.y.1.00');
size1 =
61
1
122
>> subplot(2,2,4);
>> plotam('pml2.zx.0.e.y.i','pml2.zx.0.e.y.q');
>>
```

The MATLAB session produces the following plot:

4.0 Detailed Command Descriptions

4.1 Input File Keywords

The input file to TEMPEST contains five types of specifications:

simulation domain

illumination

topography

analysis (numerical parameters)

data (output specifications)

The input file to TEMPEST can be given any name (see Appendix A for suggested file naming conventions) and the entries in the input file can occur in random order. Comments can be included in the input file by including the string ``/*" before the comment and the string ``*/" after the comment. Further, this version (as does the previous version) of TEMPEST performs error checking of the input file and will be able to detect most input errors. The input file to TEMPEST consists of a list of keywords. Each keyword may be followed by numbers, words, or other keywords. The following is a description of the keywords:

4.1.1 Domain

x_node *x_simulation_nodes*

x_simulation_nodes is a number specifying the number of simulation nodes in the x-dimension of the simulation domain. x_simulation_nodes must be a positive number greater than or equal to 2. See below for restrictions.

y_node *y_simulation_nodes*

y_simulation_nodes is a number specifying the number of simulation nodes in the y-direction of the simulation domain. y_simulation_node must be a positive number greater than or equal to 1. See below for restrictions.

z_node *z_simulation_nodes*

z_simulation_nodes is a number specifying the number of simulation nodes in the z-direction of the simulation domain. z_simulation_nodes must be a
positive number greater than or equal to 2. See below for restrictions.

x_dim *x_length*

x_length is the x-dimension of the simulation domain in micrometers. See below for restrictions.

y_dim *y_length*

y_length is the y-dimension of the simulation domain in micrometers. See below for restrictions.

z_dim *z_length*

z_length is the z-dimension of the simulation domain in micrometers. See below for restrictions.
Restrictions exist on the values *x_length* , *x_simulation_nodes* , *y_length, y_simulation_nodes* , *z_length* and *z_simulation_nodes* : TEMPEST 5.0 uses a cubic discrization grid placing the restriction that Dx=Dy=Dz which means that *x_length* /*x_simulation_nodes* = *y_length* /*y_simulation_nodes* =*z_length*/*z_simulation_nodes* . Consequently it is not necessary to specify all six parameters but rather only enough to completely define all three dimensions of the domain and the discretization Dx. Any unspecified parameters will be deduced by TEMPEST. If more than the minimum number of parameters are specified then they must satisfy the above restriction (to within numerical error) otherwise TEMPEST will give an error. Also, for 2D simulations the dimension in which there exists only one node must be chosen as the y-dimension.

x_symmetry

keyword **x_symmetry** specifies that the simulated topography has mirror symmetry in the x direction. Nodes with coordinates (0,y,z) will use nodes with coordinates (1,y,z) as their west neighbors and nodes with coordinates (*x_simulation_nodes-1* ,y,z) will use nodes with corrdinates (*x_simulation_nodes* -2,y,z) as their east neighbors. If **x_symmetry** is not specified then periodic boundary conditions apply in the x-dimension where nodes with coordinates (0,y,z) use nodes with with coordinates (*x_simulation_nodes* -1,y,z) as their west neighbors and nodes with coordinates (*x_simulation_nodes* -1,y,z) use nodes with coordinates (0,y,z) as thier east neighbors.

y_symmetry

keyword **y_symmetry** specifies that the simulated topography has mirror symmetry in the y direction. Nodes with coordinates (x,0,z) will use nodes with coordinates (x,1,z) as their south neighbors and nodes with coordinates (x,*y_simulation_nodes-1* ,z) will use nodes with corrdinates (x,*y_simulation_nodes* -2,z) as their north neighbors. If **y_symmetry** is not specified then periodic boundary conditions apply in the y-dimension where nodes with coordinates (x,0,z) use nodes with with coordinates (x,*y_simulation_nodes* -1,z) as their south neighbors and nodes with coordinates (x,*y_simulation_nodes* -1,z) use nodes with coordinates (x,0,z) as thier north neighbors. If *y_simulation_nodes* is 1, for example in a 2D simulation, y_symmetry will have no effect and all nodes will use themselves as their both their north and south neighbors.

z_symmetry

keyword **z_symmetry** specifies that the simulated topography has mirror symmetry in the z direction. Nodes with coordinates (x,y,0) will use nodes with coordinates (x,y,1) as their down neighbors and nodes with coordinates (x,y,*z_simulation_nodes* -1) will use nodes with corrdinates (x,y,*z_simulation_nodes* -2) as their up neighbors. If **z_symmetry** is not specified then periodic boundary conditions apply in the z-dimension where nodes with coordinates (x,y,0) use nodes with with coordinates (x,y,*z_simulation_nodes* -1) as their down neighbors and nodes with coordinates (x,y,*z_simulation_nodes* -1) use nodes with coordinates (x,y,0) as thier up neighbors.

4.1.2 Illumination

wavelength *lambda*

lambda is the free space wavelength of the incident illumination in micrometers. TEMPEST assumes monochromatic illumination.

exposure_time *exposure_time*

sets the total amount of exposure time (used when dynamic resist material is present). If not specified, the default exposure time is 30 seconds.

exposure_steps *exposure_steps*

set the number of exposure steps (used when dynamic resist material is present). The time step will be *exposure_time/exposure_steps* . When not specified, the default number of exposure steps is 5.

point_source *unit x_pos y_pos z_pos xcomp ycomp zcomp mag pha*

unit can be either **position** or **node**. If **position** is specified then *x_pos* , *y_pos* and *z_pos* are the x, y and z coordinates in micrometers of a point source excitation. If **node** is specified then *x_pos* , *y_pos* and *z_pos* are the x, y and z nodal coordinates of the point source. The amplitude, phase and polarization of the point source are determined by the remaining parameters with the following formula:

plane_source *orientation pos_unit min1 max1 min2 max2 pos3 xcomp ycomp zcomp intensity pha type arg1 arg2*

This keyword specifies the existence of a planar segment excitation. *orientation* can be one of {**xy**, **yz** or **zx**} corresponding to xy, yz or zx planar segments of excited nodes. *pos_unit* can be either **position** or **node** corresponding to the units (micrometers or node number) of the next five entries which determine the size and position of the planar segment source. *min1* and *max1* specify the extent of the source in the 1st direction and *min2* and *max2* specify the extent of the source in the 2nd direction while *pos3* specifies the position of the source in the 3rd direction. See table below for an explicit description. *intensity* is the intensity in of the source if it were a normally incident plane wave in free space. The electric field is calculated from the following formulas:

where and are magnitude and phase functions determined by the remaining arguments. *type* can be **uniform** or **file** meaning that the distribution of intensity accross the plane is uniform or nonuniform and read in from a file on disk.
If **uniform** is selected then the distribution is uniform and represents a plane wave travelling in the direction specified by *arg1* and *arg2* . *arg1* is the measured from the positive normal to the plane. *arg2* is the azimuthal angle measured from the axis in the counterclockwise direction. See diagram below:
If **file** is selected then the distribution is loaded in from disk. *arg1* and *arg2* are the filenames of the magnitude and phase files respectively.

4.1.3 Topography

The topography of the simulation domain (including the pml, first order and perfectly conduction boundary conditions) is specified with a series of block definitions. Each block definition has the following form:

primitive unit [pos1 pos2 pos3 . . .] material [arg1 arg2 arg3 arg4 . . .]

The number of positioning arguments (*pos1, pos2* etc.) depends on the primitive chosen and the number of material arguments (*arg1, arg2* etc) depends on the material chosen. In all cases *unit* can be either **position** or **node** specifing whether the positioning arguments are in micrometers or nodes (nodes are always numbered starting from zero) and *primitive* can be selected from any of the following primitives:

rectangle *unit xl xh yl yh zl zh material arg1 arg2 arg3 arg4 ...*

specifies that the existence of rectangular box bounded in the x-direction by [xl, xh], in the y-direction by [yl, yh], and in the z-direction by [zl, zh].

sphere *unit radius xcenter ycenter zcenter material arg1 arg2 arg3 arg4 ...*

specifies the existence of a sphere with radius radius centered at (xcenter, ycenter, zcenter).

xcylinder *unit radius xl xh ycenter zcenter material arg1 arg2 arg3 arg4 ...*

specifies the existence of a circular cylinder oriented along the x-direction from xl to xh with radius radius. The circular cross-section on the yz-plane is centered at (ycenter, zcenter).

ycylinder *unit radius yl yh zcenter xcenter material arg1 arg2 arg3 arg4 ...*

specifies the existence a circular cylinder oriented along the y-direction from yl to yh with radius radius. The circular cross-section on the zx-plane is centered at (zcenter, xcenter).

zcylinder *unit radius zl zh xcenter ycenter material arg1 arg2 arg3 arg4 ...*

zcylinder specifies the existence of a circular cylinder oriented along the z-direction from zl to zh with radius radius. The circular cross-section on the xy-plane is centered at (xcenter, ycenter).

xwedge *unit xl xh y1 z1 y2 z2 y3 z3 material arg1 arg2 arg3 arg4 ...*

xwedge specifies the existence of a wedge with a triangular interface with the yz-plane defined by the coordinates (y1, z1), (y2, z2), and (y3, z3). The wedge is oriented along the x-direction from xl to xh.

ywedge *unit yl yh z1 x1 z2 x2 z3 x3 material arg1 arg2 arg3 arg4 .. .*

ywedge specifies the existence of a wedge with a triangular interface with the zx-plane defined by the coordinates (z1, x1), (z2, x2), and (z3, x3). The wedge is oriented along the y-direction from yl to yh.

zwedge *unit zl zh x1 y1 x2 y2 x3 y3 material arg1 arg2 arg3 arg4 ...*

zwedge specifies the existence of a wedge with a triangular interface with the xy-plane defined by the coordinates (x1, y1), (x2, y2), and (x3, y3). The wedge is oriented along the y-direction from zl to zh.

material can be selected from the following material types:

index *n k*

The block represents an isotropic, linear, non-dispersive, non-magnetic material (such air, glass, resist, silicon are at optical wavelengths). n is the refractive index of the material and k represents the lossiness of the material (k=0 implies lossless material). give formula involving n-jk)^2=epsilon complex = . For this material, mu = m0=1.whatever

dispersive *n k*

same as index except used when k>n. yadda yadaa yada

cond *conductivity*

perfect conductor approximation ....

epsmu *eps_real eps_imag mu_real mu_imag*

gindex exx exy exz eyx eyy eyz ezx ezy ezz

sexx sexy sexz seyx seyy seyz sezx sezy sezz

mxx mxy mxz myx myy myz mzx mzy mzz

smxx smxy smxz smyx smyy smyz smzx smzy smzz

-not working yet

bc *orientation n theta*

black_matter

The fields inside a block with this material specification are zero. This can be used as a perfect electric/magnetic conductor boundary condition. The can also be used in regions of the domain which in which the fields are know to be zero (such as deep inside a metal object) to save computation time.

pml xabs yabs zabs eps mu sigmae sigmam

resist *init_conc a b c*

resist specifies that the current block has dynamic change with respect to exposure. init_conc is the initial PAC value of the photoresist block. a is Dill's bleachable absorption coefficient of the photoresist in per micro-meter. b is Dill's unbleachable absorption coefficient of the photoresist in per micro-meter. c is the bleach rate of the photoresist model proposed by Dill in centi-meter squared per milli-Joule.

4.1.4 Analysis

min_cycle *min_wave_cycles*

This sets the minimum number of cycles TEMPEST will run to min_wave_cycles regardless of the convergence status. If unspecified, the default value is zero.

max_cycle *max_wave_cycles*

Simulation continues until steady-state is reached or until the domain has been excited with max_wave_cycles wave cycles, whichever comes first. If unspecified, the program will run until convergence occurs.

dt *value*

specify the time step to be used in seconds. If unspecified, TEMPEST automatically calculates the time step to be used based on .....

err_tol *fraction*

Steady-state is determined by comparing the field values at evenly distributed set of test points at the same instant in successive wave cycles. If the relative variation of the electric field intensity at each node in the test grid is less than fraction for three consecutive wave cycles, steady-state is reached. A typical value for fraction is 0.1 and the relative variation is defined:

unless the denominator is deemed extremeley small in which case the point is not included in the comparison.

4.1.5 Data

plot *plot_var [arg1 arg2 . . .] coord_unit xmin xmax ymin ymax zmin zmax file*

plot_var must be one of the keywords: **refractive, block, ex, ey, ez, hx, hy, hz** or **pac**.

refractive requests a plot of the refractive index of the materials in the domain. A value of zero is reported for PML or 1st order boundary condtion materials. For anisotropic materials, the maximum refractive index is reported.

block requests a plot of the block number. Blocks are numbered starting from zero in the order of appearance in the input file.

ex, ey, ez, hx, hy, or **hz** request a plot of the corresponding electromagnetic field component. Two additional parameters, *arg1* and *arg2* must be specified. *arg1* must be one of **steady, nonsteady** or **transient**.

steady requests a field plot at the end of the simulation when either the fields have reached steady state or the maximum number of iterations (set by the **max_cycle** command) has been reached. The second argument, *arg2* , must be a number between 0 and 1 which specifies at what fraction of the wave cycle (the phase) the instantaneous field should be reported.

nonsteady requests a field plot at the instant of time specified by *arg2* (in #'s of wave cycles since the start of the simulation).

transient - not done yet.

pac - not done yet.

coord_unit must be either **position** or **node** indicating whether the following six coordinates are specified in micrometers or node numbers.
xmin, xmax, ymin, ymax, zmin and *zmax* specify what part of the domain will be plotted.
file is the filename on disk to write the data. If the file exists, it will be overwritten. See Appendix A for suggested file naming conventions.

4.2 TEMPEST command

The program TEMPEST takes two arguments. The first of which is the name of the input file which contains the topography information. The second argument is the name of the output file to which information on the simulation run is written.

prompt: tempest input_file output_file

See Appendix A for suggested file naming convention.

4.3 Post-TEMPEST Commands

Version 5.0 no longer supports the PLOTMTV format. All output data (with the exception of the standard output file automatically generated) is written in a binary data format which uses much less space and is easily ported. Several MATLAB scripts have been developed to view and/or modify the output data. The script files are given in Appendix B. Detailed command descriptions (from the MATLAB prompt >>) follow:

>>plotbin('*filename* ');

This command calls the plotbin.m script file. It is used to plot refractive index plots, block number plots and field plots at a particular instant of time.

>>plotbinsten('*filename* ','*top_filename* ');

This command calls the plotbinsten.m script file. It is similar to the **plotbin** command above but it stencils in the topography borders. *filename* is a field file (such as the x component of the electric field at steady state) and *top_filename* is a plot file which either contains the refractive index or block number of region corresponding that of *filename* .

>>plotam('*filename.i* ','*filename.q* ');

This command calls the plotam.m script file. It is used to view the time averaged (as opposed to instantaneous) field amplitude constructed from two plot files differing by one-quarter cycle (the in-phase and quadrature field plots). filename.i and filename.q are instantaneous field plots taken at two different times one-quarter cycle apart.

>>plotamsten('*filename.i* ','*filename.q* ','*top_filename* ');

This command calls the plotamsten.m script file. It is similar to the **plotam** command above except that the topography information taken from *top_filename* is stenciled in.

>>plotam6('*filename* ');

This command calls the plotam6.m script file. It is used to assemble all three components of a field at two different times (in-phase and quadrature) (six files in total) to yield the time averaged total field intensity. It is very similar to **plotam** but includes three field components (the x,y and z) instead of just one. *filename* should be the name of the files without the .x.i, .x.q etc. extensions. The script file automatically adds them. See examples in Chapter 3 for a better understanding of how this command works.

>>plotam6sten('*filename* ','*top_filename* ');

This command calls the plotam6sten.m script file. It is used similar to the **plotam6** command above except that the topography information taken from *top_filename* is stenciled in.

5.0 Useful Hints

5.1 Material Interface at the Boundaries

If a material interface exists very close to an absorbing boundary condition the simulation may be inaccurate. There may be evanescent fields near these materials and it is not know how the boundary conditions behave in the presence of evanescent waves. To ensure accurate results leave one wavelength between any material interfaces and any absorbing boundary conditions.

5.2 Verification of Loaded Topography

If it is suspected that the loaded topography does not coincide with which the user has in mind, the user can verify the loaded topography by plotting the real part of the refractive index or the block number using the keyword plot as described in Section 4.

5.3 Real and Imaginary Parts of the Refractive Index

Numerical instability may occur when the imaginary part of the refractive index of a particular material block is greater than the real part. Such a situation may arise if a metal block is present. In such instances, the material type keyword ``dispersive" should be used instead of ``**index**".

5.4 Number of Simulation Nodes

The number of simulation nodes needed in order to give accurate simulation results can be determined by the following relation:

x_simulation_nodes > (15*largest_index*x_length)/wavelength

and the number of simulation nodes in the y- and z-directions can be determined from x_simulation_nodes by the ratios of x_length, y_length and z_length.

5.5 Periodic Structures

5.6 Isolated Structures

5.7 PML Thickness

The thicker the PML (in nodes) the better its performance is. Anywhere from 4 to 16 nodes of PML is suggested depending on memory restrictions. Remember, PML nodes are just like other material nodes and they take up space in the simulation domain.

5.8 2D simulations

2D simulation can be run by putting only 1 node in the y direction.

5.9 Symmetry - using it properly

Illumination will also take on the symmetry of the domain - ie. off-normal plane waves cannot exist.

5.10 Proximity to Sources and Boundary Conditions

5.11 Convergence Problems

A.0 Appendix A: Suggested File Naming Convention

The input file should have a **.in** extenstion. (ex. hole.in)

All files should have a prefex the same as the input file prefix.

The output file should have a **.out** extention. (ex. hole.out)

All plot output files should indicate which region of the domain is plotted and what data is being plotted. If only a plane of nodes is output then the following convention should be applied:

prefix .{**xy** or **yz** or **zx**}.*position* .{**e** or **h** or **ref** or **pac** or **blk**}[.{**x** or **y** or **z**}.{**i** or **q** or *time* }]

example: hole.yz.0.e.x.i means the in-phase component of the x component of the electric field at steady state for an yz plane of the domain located at x = 0.

example: hole.zx.3.blk means the block number for a zx plane of the domain located at y = 3.

example: hole.yz.35.e.x.2.25 means the x component of the electric field after 2.25 cycles of simulation (non-steady-state plot).

It is important to follow the naming conventions for some of the MATLAB scripts to work correctly.

B.0 Appendix B: MATLAB scripts

plotbin.m

```
function a=plotbin(file1)

% comment

fp1=fopen(file1);

size1=fread(fp1,3,'long')

if size1(1)==1

nx=size1(2);

ny=size1(3);

xlab='y axis (nodes)';

ylab='z axis (nodes)';

end

if size1(2)==1
```

```
nx=size1(1);
ny=size1(3);
xlab='x axis (nodes)';
ylab='z axis (nodes)';
end
if size1(3)==1
nx=size1(1);
ny=size1(2);
xlab='x axis (nodes)';
ylab='y axis (nodes)';
end
a=fread(fp1,[nx ny],'float');
pcolor(a');
axis('equal');
colormap('hot');
shading('flat');
colorbar;
xlabel(xlab);
ylabel(ylab);
title('Electric Field');
fclose(fp1);
plotbinsten.m
function a=plotbinsten(file1,file2)
% comment
fp1=fopen(file1);
fp2=fopen(file2);
size1=fread(fp1,3,'long')
size2=fread(fp2,3,'long')
if size1(1)==1
nx=size1(2);
ny=size1(3);
xlab='y axis (nodes)';
ylab='z axis (nodes)';
end
if size1(2)==1
nx=size1(1);
```

```
ny=size1(3);
xlab='x axis (nodes)';
ylab='z axis (nodes)';
end
if size1(3)==1
nx=size1(1);
ny=size1(2);
xlab='x axis (nodes)';
ylab='y axis (nodes)';
end
a=fread(fp1,[nx ny],'float');
dsten=fread(fp2,[nx ny],'float');
d1sten=dsten;
d1sten(2:nx+1,2:ny+1)=dsten;
dsten=abs(sign(dsten-d1sten(1:nx,1:ny)));
for x = 1:nx,
for y = 1:ny,
dsten(x,y)=dsten(x,y)*(rem(x+y,2)-.5);
end
end
pcolor((a+2*max(max(a))*dsten)');
axis('equal');
colormap('hot');
shading('flat');
colorbar;
xlabel(xlab);
ylabel(ylab);
title('Electric Field');
fclose(fp1);
fclose(fp2);
plotam.m
function [a,b,c] = plotam(file1,file2)
% comment
fp1=fopen(file1);
fp2=fopen(file2);
size1=fread(fp1,3,'long');
```

```
size2=fread(fp2,3,'long');
if size1(1)==1
nx=size1(2);
ny=size1(3);
xlab='y axis (nodes)';
ylab='z axis (nodes)';
end
if size1(2)==1
nx=size1(1);
ny=size1(3);
xlab='x axis (nodes)';
ylab='z axis (nodes)';
end
if size1(3)==1
nx=size1(1);
ny=size1(2);
xlab='x axis (nodes)';
ylab='y axis (nodes)';
end
a=fread(fp1,[nx ny],'float');
b=fread(fp2,[nx ny],'float');
c=a.*a+b.*b;
pcolor(c');
shading('flat');
caxis([0 2*average(c)]);
axis('equal');
colormap('hot');
colorbar;
xlabel(xlab);
ylabel(ylab);
title('<|E|^2>');
fclose(fp1);
fclose(fp2);
plotamsten.m
function [a,b,c] = plotamsten(file1,file2,file3)
% comment
```

```
fp1=fopen(file1);
fp2=fopen(file2);
fp3=fopen(file3);
size1=fread(fp1,3,'long');
size2=fread(fp2,3,'long');
size3=fread(fp3,3,'long');
if size1(1)==1
nx=size1(2);
ny=size1(3);
xlab='y axis (nodes)';
ylab='z axis (nodes)';
end
if size1(2)==1
nx=size1(1);
ny=size1(3);
xlab='x axis (nodes)';
ylab='z axis (nodes)';
end
if size1(3)==1
nx=size1(1);
ny=size1(2);
xlab='x axis (nodes)';
ylab='y axis (nodes)';
end
a=fread(fp1,[nx ny],'float');
b=fread(fp2,[nx ny],'float');
d=fread(fp3,[nx ny],'float');
c=a.*a+b.*b;
d1=d;
d1(2:nx+1,2:ny+1)=d;
d=abs(sign(d-d1(1:nx,1:ny)));
for x = 1:nx,
for y = 1:ny,
d(x,y)=d(x,y)*(rem(x+y,2)-.5);
end
end
```

```
pcolor((c+2*max(max(c))*d)');
shading('flat');
caxis([0 2*average(c)]);
c=c+2*max(max(c))*d;
axis('equal');
colormap('hot');
colorbar;
xlabel(xlab);
ylabel(ylab);
title('<|E|^2>');
fclose(fp1);
fclose(fp2);
fclose(fp3);
plotam6.m
function g = plotam6(file);
% comment
fp1=fopen([file '.x.i']);
fp2=fopen([file '.x.q']);
fp3=fopen([file '.y.i']);
fp4=fopen([file '.y.q']);
fp5=fopen([file '.z.i']);
fp6=fopen([file '.z.q']);
size1=fread(fp1,3,'long')
size2=fread(fp2,3,'long')
size3=fread(fp3,3,'long')
size4=fread(fp4,3,'long')
size5=fread(fp5,3,'long')
size6=fread(fp6,3,'long')
if size1(1)==1
nx=size1(2);
ny=size1(3);
xlab='y axis (nodes)';
ylab='z axis (nodes)';
end
if size1(2)==1
nx=size1(1);
```

```
ny=size1(3);
xlab='x axis (nodes)';
ylab='z axis (nodes)';
end
if size1(3)==1
nx=size1(1);
ny=size1(2);
xlab='x axis (nodes)';
ylab='y axis (nodes)';
end
a=fread(fp1,[nx ny],'float');
b=fread(fp2,[nx ny],'float');
c=fread(fp3,[nx ny],'float');
d=fread(fp4,[nx ny],'float');
e=fread(fp5,[nx ny],'float');
f=fread(fp6,[nx ny],'float');
g=a.*a+b.*b+c.*c+d.*d+e.*e+f.*f;
g=0.5*g;
pcolor(g');
shading('flat');
caxis([0 3000]);
%caxis([0 2*average(g)]);
axis('equal');
colormap('hot');
colorbar;
xlabel(xlab);
ylabel(ylab);
title('<|E|^2>');
fclose(fp1);
fclose(fp2);
fclose(fp3);
fclose(fp4);
fclose(fp5);
fclose(fp6);
plotam6sten.m
function g = plotam6sten(file,stenfile);
```

```
% comment
fp1=fopen([file '.x.i']);
fp2=fopen([file '.x.q']);
fp3=fopen([file '.y.i']);
fp4=fopen([file '.y.q']);
fp5=fopen([file '.z.i']);
fp6=fopen([file '.z.q']);
fp7=fopen([stenfile]);
size1=fread(fp1,3,'long')
size2=fread(fp2,3,'long')
size3=fread(fp3,3,'long')
size4=fread(fp4,3,'long')
size5=fread(fp5,3,'long')
size6=fread(fp6,3,'long')
size7=fread(fp7,3,'long')
if size1(1)==1
nx=size1(2);
ny=size1(3);
xlab='y axis (nodes)';
ylab='z axis (nodes)';
end
if size1(2)==1
nx=size1(1);
ny=size1(3);
xlab='x axis (nodes)';
ylab='z axis (nodes)';
end
if size1(3)==1
nx=size1(1);
ny=size1(2);
xlab='x axis (nodes)';
ylab='y axis (nodes)';
end
a=fread(fp1,[nx ny],'float');
b=fread(fp2,[nx ny],'float');
c=fread(fp3,[nx ny],'float');
```

```
d=fread(fp4,[nx ny],'float');
e=fread(fp5,[nx ny],'float');
f=fread(fp6,[nx ny],'float');
dsten=fread(fp7,[nx ny],'float');
g=a.*a+b.*b+c.*c+d.*d+e.*e+f.*f;
g=0.5*g;
d1sten=dsten;
d1sten(2:nx+1,2:ny+1)=dsten;
dsten=abs(sign(dsten-d1sten(1:nx,1:ny)));
for x = 1:nx,
for y = 1:ny,
dsten(x,y)=dsten(x,y)*(rem(x+y,2)-.5);
end
end
pcolor((g+2*max(max(g))*dsten)');
shading('flat');
caxis([0 3000]);
%caxis([0 2*average(g)]);
g=g+2*max(max(g))*dsten;
axis('equal');
colormap('hot');
colorbar;
xlabel(xlab);
ylabel(ylab);
title('<|E|^2>');
fclose(fp1);
fclose(fp2);
fclose(fp3);
fclose(fp4);
fclose(fp5);
fclose(fp6);
fclose(fp7);
```