μMatIC

Micro modelling of Materials
Simulation Code

User's Manual

Version 2.0
July 2009

Table of Contents

1 Introduction

This user's manual includes a brief description of the microstructural modelling of materials (μMatIC) control volume computer program that has been developed to simulate the solidification process of alloys. The μMatIC code is a combination of a a finite difference solute diffusion solver with a volume of fluid style scalar implemented to track the solid-liquid-gas interface. The computational model domain is divided into identically sized orthogonal volumes (or cells). Conceptually, each control volume is characterized by different variables (e.g. temperature, crystallographic orientation) and states (e.g. phases: liquid, solid, gas or growing combination of the above).

The 3-dimensional μMatIC code is able to predict microstructural features of the solidification including dendritic morphology, spacing, grain structure, concentration profile, pores, casting defects, etc. The crystallographic orientation of grains is implemented into the code using a decentred square growth technique.

The code is written in the C programming language.

1.1 Licensing

The code is coyrighted. However, it is open-source and distributed under a BSD style agreement. If you use this manual or software, you are agreeing to:

The authors would also appreciate, but do not require, your citing appropriate references from the below list if you use this software. In addition, we welcome feedback and updates. If you add features or have a great applicatino, we're happy to add your papers to the list to be considered for referencing!

1.2 References for μMatIC

The code was first written in 2D and later re-written in 3D. The below papers give the equations upon which it is based, as well as some of the assumptions and limitations. If you find μMatIC useful, the authors would appreciate it if you would cite some of the below references in your publications, as appropriate. There is a more full list of the references in the section *References for $\mu M\alpha\tau IX$.*

1. Lee, P.D., and Hunt, J.D., "A Model of the Interaction of Porosity and the Developing Microstructure", Modeling of Casting, Welding, and Advanced Solidification Processes VII, Ed. M. Cross & J. Campbell, TMS, 585-592, 1995.
2. Lee, P.D. and Hunt, J.D., "Hydrogen porosity in directionally solidified Aluminium-Copper alloys: A Mathematical Model", Acta Mat. 49 (8), 1383-1398, 2001.
3. Wang, W., Lee, P.D. and McLean, M., "A Model Of Solidification Microstructures In Nickel Based Superalloys: Predicting Primary Dendrite Spacing Selection", Acta Mat. 51, 2971-87, 2003.
4. Atwood, R.C. and Lee, P.D., "Simulation of the Three Dimensional Morphology of Solidification Porosity in an Aluminium-Silicon Alloy", Acta Mat. 51 (18), 5447-5466, 2003.

1.3 Feedback

Please send any questions or comments regarding the μMatIc software or this manual to:

Peter D. Lee

Department of Materials, Imperial College London

Prince Consort Road, London SW7 2BP

Tel: 0207 594-6801; Fax: 0207 594-6758

E-mail: p.d.lee@imperial.ac.uk

2 How to install the μMatIC code

The μMatIc code can be installed on Gnu/LINUX systems, and Gnu/Cygwin/WINDOWS (through cygwin; a Unix-like environment) operating systems. It may be installed on other Unix-like environments assuming the library dependencies can be met.

The freely available libraries that are required are:

- A standard C library (libc, glibc)
- A standard C mathematics library (libm)

The freely available libraries that are optional and provide useful features are:

- The zlib compression library (libz)
  - Provides compressed checkpoint files saving approx 90% of the disk space for the output
- a PNG image library (libpng) and
- the GD front-end graphics library (using the above PNG library)
  - provides immediate, but limited, image output without requiring a postprocessor

In addition, the *Gnu Make* program and *cmake* build system are recommended for easy compilation of the source code.

The program has been compiled with *Gnu C compiler (gcc)* and *Intel C Compiler (icc)*, but has not been tested with other compilers.

2.1 Optional Prerequisites

The following packages should be installed and correctly configured in the system for the package use the optional compression and simple image generation features.

- Compression library zlib (required for PNG library)
- PNG library libpng (required for GD library)
- GD Image library libgd (easy-to-use interface for PNG library, plus other image features)

So-called *'devel'* (developer) versions of these libraries need to be installed (eg. from RedHat Package Manager *.rpm* files with the '-devel' suffix; in Cygwin *setup.exe* be sure to tag the *devel* packages for installation). These files include the system headers, which allow the code to call functions from the libraries.

On systems without package managers, these packages may be compiled and installed using their respective source files. They must be installed in the order shown above, since PNG uses compression, and GD uses PNG. Other formats for GD may be disabled during its configuration if those libraries (JPEG, TIFF, etc) are not available. They are not used by this software.

2.1.1 Debugging

For debugging the program, the following extra libraries have proven useful:

- Electric Fence debugging library libefence (for debugging only)

http://perens.com/FreeSoftware/

- DMALLOC debugging library libdmalloc (for debugging only) This is usually available as a package in the Gnu/Linux distribution as it is currently popular.

On a Gnu/Linux system, the following debuggers have been tried:

- Gdb, this is the basic debugger , most other available 'debuggers' are actually user-friendly interfaces to this package
- Ddd 'Data Display Debugger', an X based visual interface

  This package is currenly under active maintenance so check for updated versions if there are any problems with it. One of the principal maintainers is at Oxford.

- Splint, a static analyser (like 'lint' but more so…)

2.2 Obtain and Compile the code

2.2.1 Obtaining the current version

2.2.1.1 Users

1. Obtain the archive of source code from the µMatIC web site (to be determined) or by contacting the authors at the address given at the front of this manual.

2. Unpack the source code. For example if the version is *081108* then issue the command line:

```
tar –zxvf umatic-source-081108.tgz
```

This will create a folder called 'casource' containing the source code.

3. Create a 'build' directory
4. Enter the build directory and run the *ccmake* command, providing the path to the **casource** folder you just created:

```
mkdir build
cd build
ccmake ../casource
```

This should provide a text menu with several options. The options prefixed with 'UMAT' are specific to μMatIC and adjust compile-time options which may depend upon your system.

Table 1. Options specific to μMatIC in the *cmake* menu

UMAT_32	Build a 32 bit version on 64 bit system (requres 32 bit libraries)
UMAT_BUILD_MAIN	Build the main program *umat_run*
UMAT_BUILD_READER	Build the post-processing block reader *umat_read*
UMAT_BUILD_UTIL	Build some additional utilities such as the macromodel result importing program
UMAT_COMPRESS_BLOCKS	store block data in compressed form (requires *zlib*) Set 'OFF' if the *zlib* library is not available.
UMAT_DEBUG_MANYDUMPS	Produce lots of output for debugging
UMAT_EXTERNAL_LIB	Build a library *libumat.so* that may be called by a coupled macro-model
UMAT_EXTERNAL_STUB	Use an empty subroutine required by certain macro-models
UMAT_IMAGE_OUTPUT	Produce images of selected x-y slices. This requries *libz libpng* and *libgd*, set to 'OFF' if these libraries are not available.
UMAT_LIST_ALL_NUC	Produce output of ALL solidification nuclei (This may use up your disk space, use carefully)

5. Select the options, the push 'c' to configure, read the output message, push 'e' to exit the output message, and if all is well, push 'g' to 'generate' the *makefile* build-control script.

6. Run *make* to process the script and build the program. Optionally, speed up the compilation by telling *make* to run on multiple processors. For example if you have a quadcore processor or two dual-core processors, you might ask for all processors by the **–j4** flag to *make*, or use a greater number if you have more cpu cores available:

```
make –j4
```

You now should have a working *umat_run* in your *build* directory.

2.2.1.2 Developers

Once you have obtained access to the repository, you can check out the source files to your home directory (using *SVN* archiving)

The following example shows how to create a directory named "*ca_source*" including the latest version of the code using SVN. This assumes that the location of the SVN source repository is in located at `http://xe01/svn/ca_svn.` The latest main version is known as the 'trunk' in SVN usage (i.e. it is not a 'branch') and the name of the project is 'casource'. The *svn* command has the form *svn checkout <repository>/<version>/<project> <target-location>* , so the command would be:

```
svn co http://xe01/svn/ca_svn/trunk/casource ca_source
```

where *<ca_source>* is a subdirectory that will be created by this command, in which the source files will be located. The name of the whole package is *casource* which appears as the last path element in the *svn* command. The *svn* keyword '*checkout*' may be abbreviated '*co*'.

Once the source files are checked out you should usually only need the *update* and *checkin* commands most of the time

```
svn up
svn ci
```

These may be applied to individual files; if no file is specified they apply to all files that have been changed.

To check out a branch version, replace '*trunk*' with '*branches/<branch-name>*' in the command, for example:

```
svn co http://xe01/svn/ca_svn/branches/procast_2006/casource ca_p2006
```

There are also options for checking out the version as of a certain date, and many other options. To make sure these options are usful, frequent check-ins are needed. You can only retrieve a version that was checked in. The storage method used by SVN ensures that a minimum disk space is occupied when there are only small differences between check-ins.

```
IT IS ESPECIALLY IMPORTANT TO CHECK IN A VERSION JUST BEFORE RUNS
THAT WILL BE USED IN PUBLICATIONS OR THESES.
```

That way at a later date you can recreate those runs, bugs and all, if necessary.

Notes:

The SVN Manual is available online at* http://svnbook.red-bean.com/ *Please try to find the answer to your question there first.

Windows: You can use the* cygwin *port of the command-line* svn *client, as above. There is also a native Windows client called* 'tortoise svn' *with a graphical user interface. Either of these will work find, but you cannot use both of these on the same working copy since the versions do not match.

3 How to run the code

3.1 Input files

There are three key input files necessary for any job to be prepared:

◊ *umat_ctrl.in*

- The *umat _ctrl.in* is an input file giving all the control options of the code. The names of the following files may be redefined in this file if necessary.

◊ *umat _geoplus.in*

The *umat _geoplus.in* is an input file giving all of the geometry and output format data.

◊ *umat _matprop.*in

The *umat _matprop.in* is an input file giving all of the material property data.

Additional files are required to define individual component properties of an alloy. The names of these files are defined in *ca_ctrl.in* and typically these names are used:

◊ *props_alloy0.in*

Properties for the first solute component of the alloy

◊ *props_alloy1.in*

Properties for the next solute component of the alloy

◊ *props_gas.in*

Properties for the dissolved gas component

Certain options may require supplementary input files. These files are specified in the control file if the option is selected.

A typical example of each of these input files is listed in the appendix 7.1.

In case the problem includes any solid region (e.g. mould), a geometry input file should also be created and the following line should be added to the umat_ctrl.in file:

```
InpFileName <geometry_filename.inp>
```

The geometry file can be created using the *ImageJ* program, which can downloaded from the web, or any program that can save raw binary 8 bit files. The created mould image shall be exported as raw pixel data (8bits) to a file, e.g., *mould1.inp*. Any program may be used to create this file; it need only be a file containing (width x height) bytes in which a value 0 represents liquid and 255 represents mould.

3.2 Run the code

Run the code as follows:

```
umat_run –c umat_ctrl.in
```

Due to the space limitation in the home directories, the developers recommend that administrators create temporary or scratch directories where the jobs may be run.

On busy systems a queueing system is recommended, such as PBS, Torque, or SunGridEngine.

3.3 Signalling the Job

The ca program has the facility to detect certain signals which can be sent by the operating system. The user or a script can send these signals to the running job, to affect its behaviour as follows:

Signal	Effect
KILL	Immediately halt and abandon the run (usual effect)
TERM	Immediately stop processing, and jump to the final output stage to finish the run
USR1	Finish the current time step, then jump to the output stage, save the state as a block (.blk) file, and finish the run.
USR2	Finish the current time step, then save the state as a block (.blk) file, then continue the run.
FPE	(debug only – needs a compilation flag and debugger) Execute the specified routine upon a numerical error (current version only provides a location for setting the debugger break point)

3.4 Post processing

3.4.1 Extracting from the Block file

At selected time steps, or upon receiving the signals described above, a *Block file* is produced. This file contains all the data at a specific timestep. The desired information may be extracted, and the file may also be used to continue the simulation from that timestep. If *UMAT_COMPRESS_BLOCKS* is enabled, the file has the extension **.blz,** otherwise it has the extension **.blk.**

In order to extract data, the program *umat_read* must be used. This is build from the same sources as the simulation program setting the option *UMAT_BUILD_READER* in the ccmake options screen

> **Developers: If you frequently change your source code, be aware that the files produced by one version of the simulation program will not always be readable by another version of the *ca_read* program. The simulation and reader programs must be built from identical source code for trouble free operation.**

Use the umat_read program with suitable arguments. Typeing *umat_read* at the command line with no arguments produces some useful help messages. Check your version in case the commands have been changed since the version at the time of writing this manual.

```
ca_read

*************************************************
*   ./ca_read: Read output file from the CA model
*
*   Usage: ./ca_read -i input_file
*       The following command line options are allowed:
*       -i              -> specify input file
*       -b              -> specify subblock number and write a binary
brick of bytes for that data
*       -q              -> quiet mode
*       -f              -> print compile-time options (cflags)
*       -v              -> print version information
*       -t              -> specify which data and type to write
*                          for selected -b block:
*                       f          float, fraction solid
*                       a          float, alloy component
*                       g          float, gas component
*                       u          float, calculated undercooling
*                       i          unsigned short int, grain number
*                       b          byte (unsigned char), grain number
(mod 256)
*                       s          byte (unsigned char), solid
(yes/no)
*                       p          byte (unsigned char), pores
(yes/no)
*                       F          CA_FLOAT, fraction solid
*                       A          CA_FLOAT, alloy component
*                       G          CA_FLOAT, gas component
*                       U          CA_FLOAT, calculated undercooling
*                       I          int, grain number
*                          these options may be combined into a
string, eg: -t FAb

*       -h              -> print this message
```

For example

```
ca_read –i BLOCK_myrun_t0000.blz –b 0 –t fa
```

will produce the fraction-solid and alloy solute concentration data, converted to single-precision float, from the named compressed block-data file.

3.4.2 TecPlot postprocessor

Since Tecplot only takes ASCII format data, the calculated binary data (*.bin) needs to be converted into ASCII format. Programs called `i_res` and `f_res` can be used to do this conversion for integer and float data, respectively. One can find the source code in the *ca_aux_progs* subdirectory of the source code directory, and make the executables (in this subdirectory) using the target *res*

```
make res
```

This will create both single and double precision versions as described below. Users may wish to modify these programs to suit their purposes, and copy the executables into their *bin* directory. If you wish to use your own version, please be sure that your version is encountered in the system path in the correct order. Use the *which* command to verify the precise executable that the system is using.

There are two forms,

- `f_res_double` for results generated by the ca executable when it is compiled with ***double*** precision.

- `f_res_single` for results generated by the ca executable when it is compiled with ***single*** precision

The syntax is:

```
f_res_single –i <filename> -d <xsize ysize>
```

or

```
f_res_single –i <filename> -g
```

to automatically read the file named *ca_geoplus.in* in the local directory to obtain the size. By default the results are saved in the input filename with the .dat extension replacing the original extension, but any output filename may be specified with the –*o* flag.

where <*filename*> is the filename and <*xsize ysize*> are the dimensions of the slice plane. You can run the post-processor inside a small script:

```
for i in F_G*.bin
do
f_res_single –i $i -g
done
```

to process all the Gas concentration results at once, if the size is 50 x 50. Then you can load a file into Tecplot. The tecplot pore files (TP*) are already readable by Tecplot.

3.4.3 AMIRA or VGSTUDIOMAX postprocessor

Other software (VGSTUDIOMAX, AMIRA) may use the binary block output by *ca_read* as a *RAW* input file if you specify the size of the file correctly. Check your *umat_geoplus.in* file to find the **NCellsPerSB** (number of cells per subblock), these are the x,y, and z extent of the data voxels in the RAW files.

3.5 Restart

The *restart* option in µMatIC can be used to change output settings (e.g. frequency to write output) or to vary some of the input parameter (e.g. growth velocity). To run the restart, a new input file is needed which contains a base filename and parameters to be changed. For example, the restart filename *umat_restart.in* should contain:

```
BlockRestartFileName BLOCK_<base filename>.blz
```

.blk if compression is not enabled

```
SlicePFreq <new_parameter>
Velocity <new_velocity>
```

…(other parameters normally found in umat_ctrl.in)

GeoFileName <name for new geoplus parameters file>

MatFileName <name for new material property parameter files>

To restart:

```
umat_run  –r <umat_restart.in>
```

Any options to be changed will be read and placed in the corresponding variables in the program. However, this does not mean that the desired effect will take place. Developers are responsible for implementing a handler for any changes that they need to use at the restart time. Currently implemented handlers include:

- Changing the output parameters (SlicePFreq, ScreenPFreq etc)
- Changing the finish time or finish f_s
- Changing the time step

Other restart changes will require additional programming before they will have the desired effect. Please discuss with the developers about the need for implementing such features and the method for implementing them, though you are of course free to try on your own!

4 Run the code on a PC

The CA-FD program can be run on a PC through cygwin, which is a Linux-like environment for Windows. It consists of two parts: A DLL (cygwin1.dll) which acts as a Linux emulation layer providing substantial Linux API functionality, and a collection of tools, which provide Linux look and feel. The cygwin DLL works with all non-beta, non "release candidate", ix86 versions of Windows since Windows 95, with the exception of Windows CE. There have also been reports of problems on Windows Server 2003.

4.1 Install cygwin

To install the cygwin net release, go to http://sources.redhat.com/cygwin/ and click on the "Install cygwin Now" This will download a GUI installer called "setup.exe" which can be run to download a cygwin installation via the internet.

During setup, follow the instructions on each screen to install. The selections below are recommended:

Choose a download source---- install from internet (you may have problems if you save it on your local disk and install later) download site: ftp://ftp.mirror.ac.uk or ftp://ftp.skynet.be

Select package: (you can always add your other selections after installation)

- ◊ *admin: cron, cygrunsrv*
- ◊ *base: ash, bash,cygwin.grep,gzip,readline,sed,sh-utils, tar,which*
- ◊ *database:/ perhaps not*
- ◊ *devel: binutils, cmake, ctag, cvs, svn (subversion), gcc, gcc-mingw,gdb,make, mingw-runtime, crs.*
- ◊ *doc: cygwin-doc*
- ◊ *editor: ed, emacs, vim*
- ◊ *games/*
- ◊ *graphics: jpeg, libpng, libpng10, libpng10-devel, libpng12,libpng12-devel, libpng2, tiff,* ***gd, libgd2, libgd-devel***
- ◊ *interpreters:/*
- ◊ *libs: default (libraries for your selections will be added here automatically)*
- ◊ *mails/*

- ◊ *math/*
- ◊ *net/*
- ◊ *publishing/*
- ◊ *shells: bash, tcsh*
- ◊ *system: setup*
- ◊ *text: less, more*
- ◊ *utils: default*
- ◊ *web/*
- ◊ *X11/ (optional for graphic interface tools) base, fonts, Ddd (debugger) Xv, an image viewer, is available from other sources. (lassauge.free.fr/cygwin)*

The 2-d images that will be created are normal png files and may be viewed with any image viewer, so the X based image utilities are not really necessary.

4.2 Environment variables

4.2.1 Change memory size

By default no cygwin program can allocate more than 384mb of memory (program + data). You need to change the maximum memory to run "ca" by using the "regtool" program in the cygwin package. An example is given to set memory limit to 1024mb.

```
regtool –i set /HKLM/Software/Cygnus\ Solutions/Cygwin/heap_chunk_in_mb 1024
```

To check:

```
regtool –v list /HKLM/software/Cygnus\ Solutions/Cygwin
```

Exit all running cygwin process and restart again. Memory can be allocated up to the size of the system swap space minus the size of any running processes. The system swap should be at least as large as the physically installed RAM.

4.2.2 Install image library

Cygwin package

- This is now available as a package along with cygwin install, select *gd*, *libgd2* and *libgd-devel*

Manual installation

To run µMatIC with the graphic output (png files), in addition to the png and zlib compression libraries, the 'gd image library' is also needed. It is available from Thomas Boutell (boutell.com) as open-source free software. After downloading, it needs to be compiled on the cygwin platform using the provided configure script. Since freetype and jpeg option of the gd is not available without another download and installation, and is not used, the library shall be made without freetype as follows:

```
./configure --without–freetype –without-jpeg
make install
```

4.2.3 Obtain and compile µMatIc

You need to have a copy of the source code in your local disk, and compile it before running. Cygwin includes a cmake package that provides the ccmake utility, so the same instructions provided above may be followed once this is installed.

If you compile any program such "hello.c", you might find that you can't run it:

```
bash$ gcc -o hello hello.c
bash$ hello
bash: hello: command not found
```

Unlike windows, bash does not look for programs in the current directory by default. You can add "./" to your PATH. But this is not recommended by cygwin for security reasons. Just tell bash where to find it, when you type it on the command line:

```
bash$ ./hello
```

FAQ: http://cygwin.com/faq/

5 Specific use of the code

5.1 Tutorial Packages

5.1.1 Packages available

7. Bicrystal tutorial
   Basic operation, fixed or stochastic nucleation, solute properties, temperature gradient
8. Turbine blade tutorial
   Creating and using a shaped boundary condition and temperature distribution
9. Porosity tutorial
   Gas properties, pore nucleation, temperature gradient or constant temperature
10. Investment casting tutorial
    Macromodel coupling, solute source boundary condition

5.1.2 *Obtaining the packages within Imperial College Local Network*

Tutorial packages:

Please see the section "Source Code Repository" of the documents on our local web server, there's a brief explanation of SVN.

http://xe01/documentation/SourceCodeRepository.pdf
NOTE: this will be migrated to the website: www.imperial.ac.uk/AdvancedAlloys

Then obtain Tortoise SVN (http://tortoisesvn.net), download the latest Windows installer and install it.

Check out the package using Tortoise SVN. The commands are added to the Microsoft Explorer context menu, use "SVN Checkout...". This package is the the whole set including μMatIC manual and all the tutorials. Ensure that "Head Revision" is selcted in the Tortoise SVN dialog box (this should be the default)

Use command-line svn for getting the files on a unix server, in this case you may instead specify just the relevant tutorial file folder, eg.

```
svn co http://xe01/svn/ca_docs/tutorials/BicrystalTutorialFiles my_bicrystal
```

Where you may substitute any valid directory name for 'my_bicrystal'

The latest version of each file is also visible through an ordinary web browser (within the local network)

5.2 VAR simulation

5.2.1 Create temperature data files

Run the macrocode (e.g. SMPC code) to create the temperature data files in text format (e.g. csv). Each data file should have a header whose two last rows contain the variable names and their units. The header can be either space or comma separated.

It is recommended to choose the process time in seconds at which the data has been saved for the name of the data files (e.g. 35000.csv for the time slice of 35000 s).

There must not be any space within each field, while the data at each line can be either space or comma separated.

5.2.2 Make a list file containing list of the data files

Prepare an Excel spreadsheet with three columns containing the filename, time and offset of each data file. The list file might have a header line including “filename, time(s), offset(m)”. The offset should be set as follows:

- Once the macro grid is growing (extending), the offset is the ingot height at the time corresponding to the first temperature file in the list. This value should be considered for all the other files afterwards until the macro grid is fixed. It is assumed that the casting velocity at this period remains constant.
- Once the macro grid starts moving over the CA grid, the offset is calculated via: $Offset_{t-\Delta t} - V_t * \Delta t$. The velocity at each time, V_t, can be time-dependent (e.g. process perturbation). If this is the case, the appropriate velocity can be calculated from the melt rate schedule.

Save the workbook as ‘Save As Text (MS-DOS)’. The default name for such list file is “*out_list.csv*”.

5.2.3 Create the Finite Grid binary files

Run the *read_list* code to convert the text *.csv* data files to *.fgb* binary files. The following command line options are allowed:

◊	*-d*	*specify the solution mode: 0 for steady and 1 for transient*
◊	*-e*	*specify input file extension (default .csv)*
◊	*-i*	*specify input file*
◊	*-o*	*specify output file*
◊	*-z*	*specify column for z data*
◊	*-Z*	*specify initial z offset*
◊	*-r*	*specify column for r data*
◊	*-c*	*specify column for data*
◊	*-t*	*write a Tecplot file of original data*
◊	*-f*	*flip the z values in the fg structure*
◊	*-V*	*Verbose mode (lots of messages and output files)*
◊	*-u*	*change unit system (1:cm→m; 2:cm→m & °F→°C; 3:in→m & °F→°C)*
◊	*-n*	*specify number of header lines*
◊	*-h*	*print this message*

```
readlist_lin –d 1 –i <list filename> -e <.dat> -z <z_column> –r
<r_column> -c <T_column> -n <headernumber> -u <1/2/3> -f -o
<outfilename> -t -V
```

Example: `readlist_lin –d` 1 `–i` *out_list.csv* `-z` 0 `-r` 1 `-c` 2 `-u` 1 `-n` 4 `-f -t`

It is most convenient to use shell scripts to carry out this process thus avoiding retyping the flags thus risking bugs introduced by typographic errors.

5.2.4 Edit the input files

In the *ca_ctrl.in* file, the following two lines should be added:

```
FgridFileName list filename (e.g. out_list.csv)
FgridTransient solution mode
```

The solution mode is 0 for the steady state condition and 1 for the transient solution. However there is no steady-state method implemented; this needs someone to write the routine. In the meantime, "transient" will be selected, ignoring this flag for now.

5.2.5 Run the µMatIC code

In order to run the µMatIC code, one needs the following input files:

- ◊ *ca_ctrl.in*
- ◊ *ca_geoplus.in*
- ◊ *ca_matprop.in*
- ◊ *out_list.csv*
- ◊ **.fgb* */* the binary data files created by the* `readlist.c` *program */*

Run the code as follows:

```
ca_linux –c ca_ctrl.in
```

5.3 Moving Windows (decentered octahedron ONLY)

A moving frame of reference technique can applied in the simulations of directional solidification. This technique allows the dendrites to grow to a sufficient length to ensure a stable state is reached, without increasing the domain size and hence reducing the computational resources required.

The following changes are needed in input files:

In *ca_ctrl.in:*

```
Window_moving    1       /* 0=F, 1=T, moving window technique */
```

In *ca_geoplus*.in*:*

```
Window_Velo <velocity>
```

The <*velocity*> is the window moving velocity in m/s which is the same as pulling velocity for directional solidification.

This option cannot be combined with macrocode input.

Combining the moving window and directional options you must be aware that both motions (gradient velocity and window velocity) will be superimposed. You may wish to choose a zero gradient velocity if you are trying to simulate a steady-state situation.

5.4 Decentred Octahedron (Square)

A modified, decentred-octahedron growth technique is implemented in the μMatIC program to account for the effect of crystallographic anisotropy. It can be switched on by choosing options in *ca_ctrl.in:*

```
Decentred_octahedron  1  /* 0 =false, 1=true */
```

5.5 Porosity

Porosity may be simulated by activating the *pore* and *gas* diffusion options in ca_ctrl.in:

```
Pore 1 /*Porosity*/
Diffuse 1 /*GAS diffusion*/
```

Suitable parameters then need to be chosen for the initial gas concentration and the porosity nucleation behaviour in the *ca_matprop.in* file or other specified material properties file.

6 Advanced use of the code

6.1 Batch running

A shell script *makeruns.sh* has been developed to assist in batch running of ca models. To set up an *experiment* consisting of several *runs* the procedure is as follows:

6.1.1 Create a table of the parameters.

- The upper left entry is the name of the experiment
- The leftmost column is the name for each run
- Each row represents a run
- The top row contains the exact keywords from the control files for the parameters to be altered
- The table entries contain the values for the parameters for that run
- Save the table as a tab-delimited text file

6.1.2 Create a set of control files

- The following control files should be created: *<ca_ctrl.in>, <ca_geoplus.in> and <ca_matprop>.*
- The script only works with these file names. If you want to use other file names (as allowed by the ca executable) you will have to write your own script.
- These files should contain all the common parameters needed, also including 'dummy' values for those parameters to be substituted.

6.1.3 Make a copy of the ca executable in the local directory

- Rename the executable to *ca_run.*
- The script only works with these file names. If you want to use other file names you will have to write your own script.

6.1.4 Run the script

```
makeruns.sh parameters.txt
```

- This will create subdirectories named after the *run* names from the parameter file
- Verify that the control files are correct

6.1.5 Submit the runs to the queue

- A simple script from the command line usually suffices:

```
for i in <name list>
do
(cd $i; ca_q_time $i ca_run 12:00:00)
done
```

- Note the subshell delimeters ( ); these are used to prevent the script from travelling to parent directories, for example, if one member of *name-list* exists but is not a directory, a common occurrence if wild-card matching is used.
- If your parameters are such that different runs will take very different lengths of time to run, you will have to group them accordingly and adjust the time (or the queue) for submission accordingly. Use a

spreadsheet in conjunction with your parameter table to calculated the number of cells*timesteps to reach the finish time and allow approx 0.5-2 microseconds per cell per timestep (intel-32 Pentium 4, 2.8 Ghz) depending upon the options selected and the machine used.

- To calibrate your machine, inspect the output file for a typical run. The total time will be printed out. The code attempts to calculate the time/cell/step but this is dependant upon the *clock* macros and functions documented in the **gcc** and **glibc** manuals. These vary depending upon the compiler and system library used, so the value may not be sensible when non-gcc compilers or non-glibc libraries are used.

6.1.6 Wait for your jobs to finish

- On hive (a beowulf cluster) the input data is transferred to a local disk on the executing node when the job is submitted in the queue. You will not see any output files in the submitting directory until the job is finished. *The location of the file is /data/jobs/<user_pbs_job_id> on the slave node.*

7 References for μMatIC

If you find μMatIC useful, the authors would appreciate it if you would include some of the below references in your publications as appropriate. Please cite the key papers and any of the appropriate technique / application specific papers.

7.1 General use and key papers the code is described in

5. Lee, P.D., and Hunt, J.D., "A Model of the Interaction of Porosity and the Developing Microstructure", Modeling of Casting, Welding, and Advanced Solidification Processes VII, Ed. M. Cross & J. Campbell, TMS, 585-592, 1995.
6. Lee, P.D. and Hunt, J.D., "Hydrogen porosity in directionally solidified Aluminium-Copper alloys: A Mathematical Model", Acta Mat. 49 (8), 1383-1398, 2001.
7. Wang, W., Lee, P.D. and McLean, M., "A Model Of Solidification Microstructures In Nickel Based Superalloys: Predicting Primary Dendrite Spacing Selection", Acta Mat. 51, 2971-87, 2003.
8. Atwood, R.C. and Lee, P.D., "Simulation of the Three Dimensional Morphology of Solidification Porosity in an Aluminium-Silicon Alloy", Acta Mat. 51 (18), 5447-5466, 2003.

7.2 Development of the intermetallic growth module:

7.3 Development of the multicomponent, multi-phase module:

9. Wang, JS, Lee, PD, Hamilton, RW, Li, M, Allison, J, "The Kinetics of Fe-Rich Intermetallic Formation in Aluminium Alloys: In-Situ Observation", Scripta Mat., 60 (7), (2009) 516-9.

7.4 Development of the Fluid Flow module:

10. Yuan, L., Lee, P.D., Djambazov, G., Pericleous, K. "Multiscale Modeling Of The Vacuum Arc Remelting Process For The Prediction On Microstructure Formation", J. Mod. Phy. B, Vol. 23, Nos. 6 & 7 (20 March 2009)
11. Yuan, L., Lee, P.D., Djambazov, G., Pericleous, K. "Numerical simulation of the effect of fluid flow on solute distribution and dendritic morphology", Int. J. Cast Metal. Res., Vol. 22(1-4): 204-207, 2009. DOI 10.1179/136404609X368136

7.5 Papers illustrating applications of the code

7.5.1 Porosity

12. Lee, P.D., Atwood, R.C., Dashwood, R.J. and Nagaumi, H., "Modeling of Porosity Formation in Direct Chill Cast Aluminum-Magnesium Alloys", Mat. Sci. and Eng. A, 328 (1-2), 213-222, 2002.

13. Atwood, R.C. and Lee, P.D. "A Three Phase Model of Hydrogen Pore Formation During the Equiaxed Dendritic Solidification of Aluminum-Silicon Alloys", Met. Trans. B, 33 (2), 209-221, 2002.

7.5.2 Columnar to Equiaxed Transition (CET)

14. Dong, H.B. and Lee, P.D., "Simulation of the Columnar-to-Equiaxed Transition in Directionally Solidified Al-Cu Alloys", Acta Mat. 53 (3), 659-668, 2005.

7.5.3 Superalloy blades

15. Yang, X.L., Dong, H.B., Wang, W. and Lee, P.D., "Microscale Simulation of Stray Grain Formation in Investment Cast Turbine Blades", Mat. Sci. and Eng. A, 386 (1-2), 129-139. 2004.

16. Yang, X.L., Ness, D., Lee, P.D., D'Souza, N., "Simulation of Stray Grain Formation during Single Crystal Seed Melt-back and Initial Withdrawal in the Ni-Base Superalloy, CMSX4", MSE-A, 571-577, 2005.

7.5.4 Vacuum arc remelting

17. Xu, X., Zhang, W. and Lee, P.D. "Tree-Ring Formation during Vacuum Arc Remelting of INCONEL 718: Part II. Mathematical Modeling", Met. Trans. A, 33A (6), 1805-1815, 2002.

7.5.5 Titanium Dental Materials

18. Atwood, R.C., Lee, P.D., and Curtis, R.V., "Modeling the surface layer of dental titanium investment castings", Dental Materials 21 (2), 178-186, 2005.

19. Atwood, R.C., Lee, P.D., Curtis, R.V., L. Di Silvio, "Multiscale Modeling of Titanium Investment Cast Dental Prostheses", Mat. Sci. Eng. C, 21 (2), 178-186, 2005.

20. Atwood, R.C., Lee, P.D., Curtis, R.V., Maijer, D.M., "Modeling an investment cast titanium crown", Dental Materials, 23 (1), 60-70, 2007.

7.6 Linking of the μMatIC into Through Process Modelling

21. Lee, P.D., Chirazi, A., Atwood, R.C. and Wang, W., "Multiscale modelling of solidification microstructures, including microsegregation and microporosity, in an Al-Si-Cu alloy", Mat. Sci. Eng. A, 365 (1-2), 57-65, 2004.

22. Tin, S. Lee, P.D., Kermanpur, A., Rist, M., and McLean, M., "Integrated Modeling for the Manufacture of Ni-Based Superalloy Discs from Solidification to Final Heat Treatment", Met. Trans. A., 36 (9), 2493-2504, 2005.

23. Maijer, D.M., Gao, Y., Lee, P.D., Lindley, T.C. and Fukui, T., "A Through Process Model of an A356 Brake Caliper for Fatigue Life Prediction", Met. Trans. A, 35 (10), 3275-3288, 2004.

7.7 Experimental papers upon which the nucleation models are based:

24. Lee, P.D. and Hunt, J.D., "Hydrogen Porosity in Directionally Solidified Aluminium-Copper Alloys: In-Situ Observation", Acta Mat. 45 (10), 4155-4169, 1997. *

25. Lee, P.D. and Hunt, J.D., "Measuring the Nucleation of Hydrogen Porosity During the Solidification of Aluminium-Copper Alloys", Scripta Mat. 36 (4), 399-404, 1997.

26. Atwood, R.C., Sridhar, S. and Lee, P.D., "Rate Equations For Nucleation Of Hydrogen Gas Pores During Solidification of Aluminium-7%-Silicon Alloy", Scripta Mat. 41 (12), 1255-1259, 1999.

7.8 Review of Porosity Modelling Paper

27. Lee, P.D., See, D. and Chirazi, A., "Modeling Microporosity in Aluminum-Silicon Alloys: a Review", J. Light Metals 1 (1), 15-30, 2000.