Guide to Using HYDRA3D: A Three-Dimensional Digital-Image-Based Cement Microstructural Model

Dale P. Bentz
Edward J. Garboczi

United States Department of Commerce
National Institute of Standards and Technology
Guide to Using HYDRA3D: A Three-Dimensional Digital-Image-Based Cement Microstructural Model

Dale P. Bentz
Edward J. Garboczi

January 1992

U.S. Department of Commerce
Barbara Hackman Franklin, Secretary-Designate
National Institute of Standards and Technology
John W. Lyons, Director
Building and Fire Research Laboratory
Gaithersburg, MD 20899
ABSTRACT

A computer program, HYDRA3D, to simulate cement microstructural development and quantify microstructural characteristics has been developed. HYDRA3D is a menu driven program available in either Fortran or C which allows a user to create a starting microstructure, execute hydration, measure phase fractions, and assess phase connectivity. This manual outlines the conceptual model on which HYDRA3D is based, describes the programs in detail, and provides examples of program usage. A system calling diagram, source code listings, guidelines for modifying the programs, and system requirements are provided in the Appendices.

Keywords: Cement, computer modelling, hydration, interfacial zone, microstructure, percolation, simulation, software.
TABLE OF CONTENTS

ABSTRACT ................................................................. iii
LIST OF FIGURES ......................................................... vi
LIST OF TABLES .......................................................... vi
1. INTRODUCTION ......................................................... 1
2. MODEL DESCRIPTION .................................................. 1
3. PROGRAM DESCRIPTION ............................................. 6
   3.1 MENU SYSTEM ..................................................... 6
   3.2 MODULE DESCRIPTIONS ......................................... 9
4. EXAMPLES ............................................................... 13
   4.1: EFFECT OF MINERAL ADMIXTURE PARTICLE SIZE ON
        INTERFACIAL ZONE MICROSTRUCTURE ......................... 13
   4.2: EFFECT OF W/C RATIO ON CONNECTIVITY OF CAPILLARY
        POROSITY ....................................................... 20
5. REFERENCES ............................................................. 25
APPENDIX A CALLING DIAGRAM FOR HYDRA3D ....................... 27
APPENDIX B FORTRAN LISTING FOR HYDRA3D ......................... 28
APPENDIX C C LISTING FOR HYDRA3D ................................ 58
APPENDIX D MODIFYING HYDRA3D ................................... 87
APPENDIX E SYSTEM REQUIREMENTS ................................. 88
LIST OF FIGURES

Figure 1. Schematic Diagram of Steps in the Hydration Model ............... 4
Figure 2. Main Menu for HYDRA3D ............................................ 6
Figure 3. Original Particle Packings for Large and Small Mineral Admixture Particles .......................................................... 17
Figure 4. Phase Fractions vs. Distance from Aggregate Surface After Hydration for Large and Small Mineral Admixture Particles ........ 18
Figure 5. Total C₃S + C-S-H + Admixture Phase Fraction vs. Distance from Aggregate Surface After Hydration ......................... 19
Figure 6. Fraction Connected Porosity vs. Degree of Hydration ............. 23
Figure 7. Fraction Connected Porosity vs. Total Porosity ..................... 24

LIST OF TABLES

Table I. Volume Occupied by Spheres of Various Radii ......................... 14
Table II. Memory Requirements for HYDRA3D on Various Systems .......... 88
Table III. Timing Benchmarks for HYDRA3D .................................... 89
1. INTRODUCTION

In the past few years, researchers at the National Institute of Standards and Technology (NIST) have developed a three-dimensional digital-image-based model for simulating the microstructural development of hydrating cement (specifically tricalcium silicate, C₃S¹) paste [1-3]. The advantages of a digital-image-based approach over a continuum approach are several. First, since each particle in the model is a collection of pixels (or voxels), real particle shapes may be used as well as conventional geometric shapes such as spheres. Second, the rules for microstructural development can be mechanism-based, as the processes of diffusion, nucleation, and surface reaction can be conveniently simulated on a lattice of pixels. Third, since the microstructure is defined on a lattice, techniques may be borrowed from statistical physics to easily assess material properties such as percolation [4], electrical conductivity [5], and ionic diffusivity [6]. Finally, by utilizing a graphics workstation, the simulated microstructural development can be visualized "in situ" to provide new insights into the hydration process and the relationships between microstructure and material properties.

Over time, interest in the model by the cement research community outside of NIST has increased to a point where other researchers have requested copies of the model computer program, HYDRA3D. It is intended that this document serve as a technical guide to using HYDRA3D as well as providing sufficient details that the interested user might modify the code to better serve their purpose. The model code is available from the Cementitious Materials Modelling Laboratory at NIST. The authors may be contacted for details concerning acquisition.

Section 2 of this manual provides an overview of the conceptual model on which HYDRA3D is based. Section 3 describes the programs (HYDRA3D.F and HYDRA3D.C as the code has been developed in both Fortran and C), detailing both the menu system by which the user controls program execution and the separate modules (subroutines) comprising the overall code. Example runs including results are provided in section 4. A calling diagram and commented source code listings are given in the Appendices along with suggested guidelines for modifying the source code and information on system requirements for executing the model.

2. MODEL DESCRIPTION

The key to the microstructural model is the representation of a unit volume as a discrete three-dimensional array of elements (pixels). Each element is uniquely identified as belonging to a single phase of the hydrating cement system and is

¹ Standard cement chemistry notation is used throughout this document. That is, C=CaO, S=SiO₂, H=H₂O, A=Al₂O₃, and F=Fe₂O₃.
typically 1 μm on a side. The hydration process is simulated by performing operations on these individual elements according to a prescribed set of rules as outlined below.

A starting cement microstructure consists of digitized spherical particles randomly dispersed in water. Thus, each C_3S particle consists of a set of pixels representing a sphere of a desired diameter (e.g. 3 to 20 μm). The spheres are placed at random (x,y,z) locations in the volume such that they do not overlap one another. Since the model is relatively small, being 100*100*100 pixels in size, periodic boundaries are utilized to eliminate artificial edge effects. Thus, if a spherical particle extends out one face of the volume, it is completed on the opposite side. The cement particles are added in order of largest to smallest to aid in achieving a random dispersion without having to relocate the smaller particles to fit the larger ones into the system.

In addition to cement particles, two other types of particles may be added to the system. First, a single flat aggregate of user-specified thickness may be placed in the middle of the 3-D system. The aggregate extends the length of the system in the y and z directions. This allows one to study the microstructural development occurring in the interfacial zone in a model concrete system [7,8]. Second, inert or pozzolanic mineral admixture particles may be added to the system. In the case of pozzolanic particles, the particles react with the CH produced during hydration to produce pozzolanic or secondary C-S-H. Pozzolanic admixture particles are assumed to be pure silica with a molar volume of 27 cm³/mole but these assumptions may be modified as outlined in Appendix D. The mineral admixture particles may be either 1 pixel (μm) in size or of some variable diameters as in the case of the C_3S particles.

The user may control the water-to-cement (w/c) ratio of a starting microstructure by varying the number and size of C_3S particles added to the system. In a system containing only cement and water, if f is the volume fraction of pixels which are C_3S, then

\[
\frac{w}{c} = \frac{1 - f}{3.2 \ast f}
\]  

(1)

where 3.2 is the specific gravity of cement. For systems containing aggregates and/or mineral admixtures, the above equation no longer holds. In such systems, to determine the water-to-solids (w/s) ratio, the general equation

\[
\frac{w}{s} = \frac{1 - f_{agg} - f_{cement} - f_{\text{min. adm.}}}{\rho_{cement} \ast f_{cement} + \rho_{\text{min. adm.}} \ast f_{\text{min. adm.}}}
\]  

(2)

where \(f_i\) and \(\rho_i\) represent the volume fraction and density of component i respectively, should be utilized.
The user of the model should keep in mind that the mineral admixture and cement are typically of different densities, so that weight and volume fractions are not identical. For example, the density of cement is about 3.2 g/cm³ while that of silica fume is typically 2.2 g/cm³. Thus, replacing 10% by weight of the cement with silica fume results in a system in which 13.9% of the total solids volume is silica fume.

Hydration of a starting microstructure is modeled as an iterative process consisting of discrete cycles as illustrated schematically for a two-dimensional system in Figure 1. Each cycle consists of three processes: dissolution, diffusion, and reaction. Basically, material dissolves off the surfaces of the original cement particles, diffuses around within the available pore space, and reacts to form hydration products. The reactions are based on the hydration of tricalcium silicate which is assumed to react with water as follows [9]:

\[ C_3S + 5.3H = C_{1.7}SH_{4.0} + 1.3CH \] (3)

On a volume basis, this reaction is equivalent to 1 unit of C₃S producing 1.7 units of C-S-H and 0.61 units of CH [9]. Thus, 1 unit of solid reactant reacts with water to produce 2.31 units of solid product. This volume expansion in terms of solids is responsible for the evolution of a cement paste from a colloidal dispersion into a rigid solid material. The similarity between this expansion factor for C₃S hydration and cement (C₃S, C₂S, C₃A, C₄AF, and gypsum) hydration has been discussed elsewhere [4,6] and is one basis for considering the model to be a true model of cement microstructure and not limited solely to pure C₃S. Indeed, favorable comparisons between model and real cement systems have been made [1,4-8].

When pozzolanic mineral admixtures are present in the system, diffusing CH species are allowed to react at pozzolanic surfaces according to:

\[ 1.7 \ CH + S + 2.3 \ H = C_{1.7}SH_{4.0} \] (4) .

Assuming specific volumes of 27 cm³/mole for S, 33.1 cm³/mole for CH, and 124 cm³/mole for C-S-H, this reaction is equivalent to one unit of mineral admixture reacting with 2.08 volume units of CH to produce 4.6 volume units of pozzolanic or secondary C-S-H. Conversely, inert mineral admixtures do not react with any of the cementitious species present in the system.

In the dissolution phase, the entire 3-D microstructure is first scanned to identify all C₃S pixels which have one or more neighbors which are water-filled porosity. Six neighbors (±1 in the x, y, and z directions) are checked to evaluate this criteria. Next, in a second pass through the microstructure, each C₃S pixel identified in the first pass attempts to take a one step random walk. If the step is into porosity, the C₃S pixel is dissolved and a diffusing C-S-H species is created at the step’s destination location. If the step is into solid material, the C₃S pixel remains as solid
C₃S for this cycle of the hydration. This algorithm produces the desired effect that sharp edges will dissolve at a faster rate than flat surfaces and smaller particles will dissolve at a faster rate than larger ones due to their larger surface to volume ratio. Finally, after all C₃S pixels have been checked for dissolution, extra CH and C-S-H diffusing species are added at random locations in the water-filled space to maintain the correct volume stoichiometry according to equation 3. For example, if in a given cycle, 100 pixels of C₃S dissolve, 70 extra pixels of diffusing C-S-H and 61 extra pixels of diffusing CH would be added to the system.

Once the correct number of diffusing species have been created, each one executes a random walk in the available porosity until reaction occurs. Thus, for each diffusion step, every diffusing species remaining in the system continues its random walk. For each step, a direction is chosen at random and the diffusing species is allowed to move one pixel in the chosen direction if that pixel is currently unoccupied (porosity). Since diffusing species may occasionally become trapped in regions where no reaction (based on the rules outlined below) may occur, after some large number of steps, all remaining diffusing species are converted into solid product.

The reaction rules differ for the C-S-H and CH diffusing species. The C-S-H species execute random walks in the pore space until they encounter (run into) a solid C₃S or solid C-S-H surface. When this event occurs, the diffusing C-S-H species is converted into solid C-S-H. These rules for C-S-H formation combine aspects of both the through solution (diffusion) and topochemical mechanisms (surface reaction) for C-S-H formation outlined in the cement literature [10].

Solid CH forms by a nucleation and growth process in the pore space. For each random step taken by a diffusing CH species, there is a probability that it will nucleate at its current location. This probability is exponentially proportional to the number of
diffusing CH species remaining in pore solution and is given by

$$P_{\text{nuc}} = P_0 \times [1 - \exp (-[\text{CH}] / [\text{CH}]_{\text{max}})]$$  (5)

where $P_{\text{nuc}}$ is the probability of nucleation, $P_0$ is the maximum probability of nucleation, $[\text{CH}]$ is the number of CH diffusing pixels remaining in solution at each step, and $[\text{CH}]_{\text{max}}$ is a scale factor [7]. By varying $P_0$ and $[\text{CH}]_{\text{max}}$, the number and size of CH crystals formed during the hydration can be controlled. In addition to nucleation, if a diffusing CH species encounters a solid CH surface during its random walk, it is converted into solid CH at its present location, resulting in growth of the CH crystal.

When pozzolanic mineral admixtures are present in the system, the diffusing CH species may react at the pozzolanic surfaces to form pozzolanic C-S-H. Since according to equation 4, this reaction is expansive in terms of solids, there is a probability that two volume elements of pozzolanic C-S-H instead of one are formed whenever this reaction occurs. This probability is 0.73 as 2.08 units of diffusing CH should produce 3.6 units of pozzolanic C-S-H, in addition to the volume element originally occupied by the pozzolanic mineral admixture. This extra pozzolanic C-S-H is added at a random neighboring location if possible, and at a random location in the pore space otherwise. When all of the pozzolanic mineral admixture has been consumed by reaction with diffusing CH, the pozzolanic reaction is discontinued.

When all diffusing species generated during a given dissolution phase have reacted, a new hydration cycle is begun with a new dissolution. The degree of hydration, $\alpha$, after $m$ cycles of hydration is given by

$$\alpha (m) = \frac{[\text{C}_3\text{S}]_0 - [\text{C}_3\text{S}]_m}{[\text{C}_3\text{S}]_0}$$  (6)

where $[\text{C}_3\text{S}]_i$ is the number of solid $\text{C}_3\text{S}$ elements remaining after $i$ cycles of hydration. Within HYDRA3D, the user may specify either the number of hydration cycles to execute or the desired degree of hydration to be achieved.

Since the model microstructure is available in a digitized format, phase fractions can be easily assessed by simply counting the number of pixels of each phase. These phase fractions may be assessed globally or as a function of distance from the aggregate surface when an aggregate is present in the system. In addition, due to the underlying 3-D lattice structure of the model, the connectivity or percolation of the individual phases or total solids may be easily determined using a simple burning algorithm [11]. This algorithm is a simple way of identifying all pixels that are part of a spanning cluster, if such a cluster exists, and works as follows. Conceptually, all the pixels belonging to the phase(s) of interest are classified to be "combustible". A "fire" is started on one side of the model's unit cell, and allowed to propagate only
along these combustible pixels. Within HYDRA3D, this burning algorithm is executed in a non-periodic manner such that the fire cannot exit one side of the unit cell and enter the other (like the diffusing species can during the hydration). If any pixels on the opposite side of the model cell are found to have been "burned", then a spanning cluster of the phase of interest must exist. The number of "burned" pixels are counted to determine the fraction of the phase of interest that is a part of the spanning cluster. This connectivity can be assessed after any number of hydration cycles.

3. PROGRAM DESCRIPTION

In developing HYDRA3D, an effort has been made to incorporate the guidelines for the development of computer-based models as outlined by Kaetzel et al [12]. Three major software engineering considerations for such code are accessibility, maintainability, and transportability. To increase accessibility, the code for the digital-image-based hydration model has been developed in both Fortran and C, enabling transfer to a wider audience of researchers. Additionally, a menu system has been incorporated into the programs to guide the user in executing the code and to provide flexibility in the analyses selected by the user. A modular approach, as shown in the calling diagram provided in Appendix A, has been utilized to enhance code maintainability. Transportability has been enhanced by including a portable random number generator [13] as one of the modules of the code. It is hoped that these steps will increase the usability of HYDRA3D by other cement researchers.

3.1 MENU SYSTEM

The program is menu driven, with the user being prompted to supply input parameters based on their selection from the main menu shown in Figure 2. Each menu choice is described in detail below.

Input User Choice:
1) Add a flat aggregate to microstructure
2) Add spherical particles (C3S or filler) to microstructure
3) Add one-pixel filler particles to microstructure
4) Hydrate microstructure
5) Measure phase fractions
6) Measure phase fractions as a function of distance from aggregate surface
7) Measure single phase connectivity
8) Measure total solids connectivity
9) Exit

Figure 2. Main Menu for HYDRA3D
1) Add a flat aggregate to microstructure

This selection is utilized to place a thin plate-like aggregate in the middle of the 3-D microstructure. The aggregate extends the length of the box (100 pixels) in the y and z directions with the user specifying the thickness in pixels in the x direction. The input thickness should be an even integer to maintain system symmetry, with equal numbers of (paste) pixels to the left and right of the centered aggregate. Under normal operating conditions, if an aggregate is desired, it should be added to the system before any cement or mineral admixture particles. If any such particles are in the system when the aggregate is placed, it will be superimposed over those particles with all pixels within the aggregate boundaries being set to the phase ID of aggregate.

2) Add spherical particles (C₃S or filler) to microstructure

Via this selection, the user may add spherical (digitized spheres) cement or mineral admixture particles to the 3-D microstructure. Once selected, the user will first be prompted to enter the number of different sizes of spheres to be placed. Next, the user will be prompted to enter the number, radius, and phase ID of the spheres for each size class. The user should begin with the largest radius size class and proceed sequentially to the smallest radius size class. The specified spheres will be placed at random locations in the 3-D microstructure using periodic boundaries as described in section 2.

When a user is adding both mineral admixture and C₃S particles of the same radius, they should place all the largest particles (C₃S and filler) first, all the 2nd largest particles next, etc. This provides the best chance for all particles to be placed. If some of the smaller particles are placed before the larger ones, there may be no remaining spaces into which the larger ones can fit. This procedure is illustrated in one of the examples given in section 4.2.

3) Add one-pixel filler particles to microstructure

This selection is used to add small (1 pixel or 1 μm) mineral admixture particles to a starting microstructure. The user must supply the number of particles to be added and whether they are inert or pozzolanic. Since these particles are typically smaller than the cement particles used in the model (just as silica fume is finer than cement), they should be added after the cement particles and larger filler particles (if any) have been placed in the system.

4) Hydrate microstructure

This selection is utilized to execute the hydration model on a 3-D microstructure. As described in section 2, a given cycle of hydration consists of dissolution, diffusion, and reaction. When this menu item is selected, the user will be
prompted to choose either to specify the number of hydration cycles to execute or the desired degree of hydration and must supply a numerical value for the item selected. Additionally, the user must specify the maximum number of diffusion steps to take in a cycle before converting all remaining diffusing species to product (see section 2), and two parameters to control CH nucleation based on equation 5. The value for the maximum number of diffusion steps to execute in a given cycle is generally on the order of 5000 to 10000. Typical values for the two parameters controlling CH nucleation would be 0.001 for the maximum probability of CH nucleation, $P_0$, and 200000. for the exponential scale factor, $[\text{CH}]_{\text{max}}$.

5) Measure phase fractions

This selection will simply return counts of the volume (pixels) occupied by each phase in the microstructure. This item is useful in verifying the original w/c ratio of the system and to determine the degree of hydration, based on the C$_3$S remaining after some number of cycles of hydration.

6) Measure phase fractions as a function of distance from the aggregate surface

When an aggregate is present in the 3-D microstructure, this selection may be utilized to measure the phase fractions present as a function of distance from the surface of the aggregate. Output consists of a table of the phase fractions of all phases present for each (pixel) distance from the aggregate. This selection allows one to study the microstructure of an interfacial zone in a cement-based composite [7,8].

7) Measure single phase connectivity

This selection allows the user to determine the percolation characteristics of any individual phase (porosity, etc.) present in the microstructure. The user will be prompted to input the ID of the phase of interest, after which the program will determine the number of pixels of this phase which are accessible from the top of the system and the number of pixels of this phase which are part of a pathway(s) through the system going from top to bottom. The burning algorithm described in the end of section 2 is utilized to determine these values.

8) Measure total solids connectivity

This selection is identical to selection #7 except that the percolation characteristics of all solids present in the system (with the exception of any aggregate) are assessed. As the hydration proceeds, the solids will eventually form a rigid connected structure which spans the 3-D microstructure. This point should correspond to the set point (time) commonly measured by cement researchers.
3.2 MODULE DESCRIPTIONS

Program Name: MAIN
Purpose: To initialize 3-D microstructure, present the user with a menu of choices, and call routines to execute selected menu options
Calls: INTRO, ADDAGG, CREATE, FILLER, HYDRATE, MEASURE, MEASAGG, CONNECT, CONSOLD
I/O: Outputs menu of selections to user
   User inputs integer seed for random number generator
   User inputs integers corresponding to menu choices

Module Name: INTRO
Purpose: To display an introduction to HYDRA3D to the system user
Called By: Main program
I/O: Introductory text is output to the standard output unit (screen)

Module Name: ADDAGG
Purpose: To place a single plate-like aggregate in the middle of the 3-D microstructure.
Called By: Main program
I/O: User inputs an even integer for aggregate thickness
Notes: Aggregate extends length of the system in the y and z directions and specified thickness in x direction.
   Aggregate is assigned a phase ID of 8 and is inert with respect to diffusing C-S-H and CH species.

Module Name: CREATE
Purpose: To obtain user input specifying spherical particles to be placed in 3-D microstructure and call routine to execute placement.
Called By: Main program
Calls: GSPHERE
I/O: User inputs parameters specifying spherical particles to be generated
   Number of classes
   Number of spheres, radius, and phase ID for each class

Module Name: GSPHERE
Purpose: To generate digitized spheres of a fixed number of classes by finding random (x,y,z) locations at which the spheres can be placed.
Called By: CREATE
Calls: CHKSPH, RAN1
Passed In: Number of classes to be generated
   Arrays of number, radius, and phase ID for each class
Notes: For each random location generated, sphere placement is first tested (no overlap with existing spheres) and then, if possible, performed.
Module Name: CHKSPH
Purpose: To check or perform placement of a digitized sphere of specified radius and center location by operating on all pixels falling within the boundaries of the sphere.
Called By: GSPHERE
Passed In: X, Y, and Z coordinates for sphere center
          Radius of sphere
          Flag indicating if sphere placement is to be checked or performed
Returns: Flag indicating if sphere placement is possible
Notes: Periodic boundaries are employed in checking and placing spheres.
       Sphere diameter = 2*sphere radius + 1 so that spheres can always be centered on a pixel.

Module Name: FILLER
Purpose: To obtain user input as to the number and type of filler particles to be generated and call routine to execute placement.
Called By: Main program
Calls: GENFILL
I/O: User inputs number and phase ID of filler particles to be generated

Module Name: GENFILL
Purpose: To place one pixel mineral admixture particles at random unoccupied (pore) locations in 3-D microstructure.
Called By: FILLER
Calls: RAN1
Passed In: Number and phase ID of 1-pixel filler particles to generate

Module Name: HYDRATE
Purpose: To obtain user input and control hydration for some number of cycles or to some degree of hydration.
Called By: Main program
Calls: DISSOLV, MVCHANT, MCSHANT
I/O: User inputs parameters to control hydration
       Number of hydration cycles or degree of hydration to execute
       Maximum number of diffusion steps per cycle
       Maximum probability for CH nucleation
       Exponential scale factor for CH nucleation
       Outputs number of diffusing species generated during each hydration cycle
Notes: For each cycle, hydration is controlled by calling a routine to perform the dissolution and then calling routines to move the diffusing C-S-H and CH species until all diffusing species have reacted.
Module Name: DISSOLV
Purpose: To perform dissolution for each hydration cycle.
Called By: HYDRATE
Calls: RAN1
Returns: Arrays holding x, y, and z locations of generated diffusing species and number of diffusing species generated.
Notes: Dissolution is performed by randomly dissolving C\textsubscript{3}S species in contact with pore space and creating the appropriate numbers of diffusing C-S-H and CH species. Extra diffusing species are added at totally random unoccupied locations in the 3-D microstructure.

Module Name: MVCHANT
Purpose: To execute a single diffusion step for a diffusing CH species.
Called By: HYDRATE
Calls: RAN1, ADDEXT
Passed In: Current location of diffusing CH species
          Current nucleation probability for CH
Returns: New location of diffusing CH species
          Flag indicating if reaction has occurred
Notes: If nucleation is probable, the diffusing CH species is converted to solid CH at its current location. Otherwise, the diffusing CH species undergoes a one-step random walk in 3-D. If it collides with solid CH, it is converted to solid CH. If it collides with pozzolanic material, the pozzolanic reaction occurs as long as sufficient pozzolanic mineral admixture remains in the system. If the step is into pore space, the location of the diffusing CH species is updated and returned to the calling routine. If any reaction occurs, a flag is set and also returned to the calling routine.

Module Name: ADDEXT
Purpose: To add extra pozzolanic C-S-H to microstructure when a diffusing CH species reacts at a pozzolanic surface.
Called By: MVCHANT
Calls: RAN1
Passed In: X, Y, and Z coordinates of reaction site
Notes: The six neighboring locations of the reaction site are sampled at random, looking for an unoccupied site to place the extra pozzolanic C-S-H. If no such site exists, the extra pozzolanic C-S-H is placed at a random unoccupied location in the 3-D microstructure.

Module Name: MCSHANT
Purpose: To execute a single diffusion step for a diffusing C-S-H species.
Called By: HYDRATE
Calls: RAN1
Passed In: Current location of diffusing C-S-H species
Returns: New location of diffusing C-S-H species
Flag indicating if reaction has occurred
Notes: The diffusing C-S-H species executes a one-step random walk in 3-D.
If it collides with solid C-S-H or C₃S, it is converted to solid C-S-H. If the step is into pore space, the location of the diffusing C-S-H species is updated and returned to the calling routine. If reaction occurs, a flag is set and also returned to the calling routine.

Module Name: RAN1
Purpose: To generate random numbers for use by other routines in the model program.
Called By: GSPHERE, GENFILL, DISSOLV, MVCHANT, ADDEXT, MCSHANT
Passed In: Seed for random number generator
Returns: A real number in the range [0.0, 1.0)

Module Name: MEASURE
Purpose: To determine phase fractions of all phases present in the 3-D microstructure.
Called By: Main program
I/O: Outputs global phase counts (in pixels) to the standard output unit (screen)

Module Name: MEASAGG
Purpose: To assess phase fractions as a function of distance from aggregate surface to enable quantitative characterization of interfacial zone microstructure.
Called By: Main program
I/O: Outputs phase fractions as a function of distance from aggregate surface in tabular format

Module Name: CONNECT
Purpose: To assess the percolation characteristics of an individual phase using a simple burning algorithm.
Called By: Main program
I/O: User inputs ID of phase of interest
Outputs number of elements of phase which are accessible from top
Outputs number of elements of phase which are part of pathways through the 3-D microstructure from top to bottom
Notes: Burning algorithm is non-periodic.

Module Name: CONSOLD
Purpose: To assess the percolation characteristics of total solids present in the 3-D microstructure using a simple burning algorithm.
Called By: Main program
I/O: Outputs number of elements of total solids which are accessible from top
Outputs number of elements of total solids which are part of pathways
through the 3-D microstructure from top to bottom
Notes: Aggregate is not included in total solids as it always forms a continuous
path from top to bottom of the 3-D microstructure.
Burning is non-periodic.

4. EXAMPLES

HYDRA3D can be executed either interactively or in a batch mode by piping the
standard input and output to datafiles. On a UNIX-based system, for example, typing
hydra3d <hydrate.inp >hydrate.out
would execute the program hydra3d, reading the menu selections and other input
from the datafile hydrate.inp and outputting all results to hydrate.out. The output file
could then be imported to a spreadsheet or graphical analysis package for further
analysis as illustrated below. All examples were executed in this batch mode using
the Fortran version of HYDRA3D.

4.1: EFFECT OF MINERAL ADMIXTURE PARTICLE SIZE ON INTERFACIAL ZONE
MICROSTRUCTURE

For this example, we will create starting systems consisting of an aggregate,
cement particles, and either large or one-pixel mineral admixture particles added at
20% replacement for cement on a weight basis. The one pixel admixture particles
would represent well dispersed silica fume while the larger admixture particles would
be indicative of an agglomerated silica fume system. The systems will be hydrated
to the same degree of hydration and the interfacial zone microstructures compared
based on the phase fractions present as a function of distance from the aggregate
surface. A w/s ratio of 0.45 will be utilized and hydration will be executed to 70%.
The cement particles (and large mineral admixture particles) will consist of equal
volume fractions of particles 21 and 11 pixels in diameter while the aggregate
thickness will be set at 2 pixels. The user of the system must decide on these system
variables before a simulation can be executed.

To know how many of each size particle are needed, the user must know how
many pixels are occupied by a single sphere of each particle radius. These values are	abulated in Table I for sphere radii ranging from 1 to 10. As noted earlier, the sphere
diameter is equal to 2* radius + 1 so that the spheres may always be centered on a
pixel. It is not recommended that particles larger than 10 pixels in radius (21 in
diameter) be utilized in simulations so that a ratio of about 5 between system size
(100³) and individual particle size may be maintained. (If memory is available, the
system size can be increased to 200³ and larger particles used.) Using Table I, if the
Table I

<table>
<thead>
<tr>
<th>Radius (pixels)</th>
<th>Diameter</th>
<th>Pixels per Sphere</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>19</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>81</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>179</td>
</tr>
<tr>
<td>4</td>
<td>9</td>
<td>389</td>
</tr>
<tr>
<td>5</td>
<td>17</td>
<td>739</td>
</tr>
<tr>
<td>6</td>
<td>13</td>
<td>1189</td>
</tr>
<tr>
<td>7</td>
<td>15</td>
<td>1791</td>
</tr>
<tr>
<td>8</td>
<td>17</td>
<td>2553</td>
</tr>
<tr>
<td>9</td>
<td>15</td>
<td>3695</td>
</tr>
<tr>
<td>10</td>
<td>21</td>
<td>4945</td>
</tr>
</tbody>
</table>

User knows the total number of pixels of a given size sphere needed, they may calculate how many spheres will be required. Since only integer numbers of spheres may be placed, some adjustment (rounding) may be required.

For this example, the aggregate occupies 20,000 pixels (2*100*100) so that 980,000 are left for the cement paste. Based on a 20% wt replacement of cement with mineral admixture, a w/s ratio of 0.45, and specific gravities of 3.2 for cement and 2.2 for the mineral admixture (silica fume), one can calculate based on equation 2 that 309,769 pixels of cement and 112,645 pixels of mineral admixture are needed. Since half of the volume of cement should be composed of each particle size (11 and 21 pixels), we should place 31 of the 21 pixel diameter particles and 212 of the 11 pixel diameter particles for the cement, for a total of 309,963 cement pixels. For the large mineral admixture particles, these numbers are 11 and 79 respectively, for a total of 112,776 mineral admixture pixels. This will result in a w/s ratio of 0.4494, very close to our desired value of 0.45. When small one-pixel mineral admixtures are to be used, we will simply add 112,776 of them to the system (to have exactly the same number of total admixture pixels as in the case of the larger particles). In order to assure that both systems hydrate to the same degree of hydration, we will specify hydration to a specified degree of hydration (70%) as opposed to hydration for some fixed number of cycles.
The data files (examples of hydrate.inp) used to execute the two simulations are as follows:

Example 1: Simulation with large mineral admixture particles

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1639</td>
<td>random number seed</td>
</tr>
<tr>
<td>1</td>
<td>menu selection to add aggregate</td>
</tr>
<tr>
<td>2</td>
<td>aggregate thickness</td>
</tr>
<tr>
<td>2</td>
<td>menu selection to add spherical particles</td>
</tr>
<tr>
<td>4</td>
<td>number of different classes of particles to add</td>
</tr>
<tr>
<td>31</td>
<td>number of particles of class #1</td>
</tr>
<tr>
<td>10</td>
<td>radius of particles of class #1</td>
</tr>
<tr>
<td>1</td>
<td>phase ID of particles of class #1 (C\textsubscript{3}\textsubscript{S})</td>
</tr>
<tr>
<td>11</td>
<td>number of particles of class #2</td>
</tr>
<tr>
<td>10</td>
<td>radius of particles of class #2</td>
</tr>
<tr>
<td>10</td>
<td>phase ID of particles of class #2 (min. admixture)</td>
</tr>
<tr>
<td>212</td>
<td>number of particles of class #3</td>
</tr>
<tr>
<td>5</td>
<td>radius of particles of class #3</td>
</tr>
<tr>
<td>1</td>
<td>phase ID of particles of class #3 (C\textsubscript{3}\textsubscript{S})</td>
</tr>
<tr>
<td>79</td>
<td>number of particles of class #4</td>
</tr>
<tr>
<td>5</td>
<td>radius of particles of class #4</td>
</tr>
<tr>
<td>10</td>
<td>phase ID of particles of class #4 (min. admixture)</td>
</tr>
<tr>
<td>5</td>
<td>menu selection to measure global phase fractions</td>
</tr>
<tr>
<td>6</td>
<td>menu selection to measure phase fractions vs. distance from aggregate surface</td>
</tr>
<tr>
<td>4</td>
<td>menu selection to execute hydration</td>
</tr>
<tr>
<td>1</td>
<td>selection to specify desired degree of hydration</td>
</tr>
<tr>
<td>0.7</td>
<td>degree of hydration desired</td>
</tr>
<tr>
<td>10000</td>
<td>max. # of diffusion steps per cycle</td>
</tr>
<tr>
<td>0.001</td>
<td>maximum probability for CH nucleation</td>
</tr>
<tr>
<td>200000.</td>
<td>exponential scale factor for CH nucleation</td>
</tr>
<tr>
<td>5</td>
<td>menu selection to measure global phase fractions</td>
</tr>
<tr>
<td>6</td>
<td>menu selection to measure phase fractions vs. distance from aggregate surface</td>
</tr>
<tr>
<td>9</td>
<td>menu selection to exit program</td>
</tr>
</tbody>
</table>

Example 2: Simulation with one pixel mineral admixture particles

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1639</td>
<td>random number seed</td>
</tr>
<tr>
<td>1</td>
<td>menu selection to add aggregate</td>
</tr>
<tr>
<td>2</td>
<td>aggregate thickness</td>
</tr>
<tr>
<td>2</td>
<td>menu selection to add spherical particles</td>
</tr>
<tr>
<td>2</td>
<td>number of different classes of particles to add</td>
</tr>
<tr>
<td>31</td>
<td>number of particles of class #1</td>
</tr>
<tr>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>10</td>
<td>radius of particles of class #1</td>
</tr>
<tr>
<td>1</td>
<td>phase ID of particles of class #1 (C₃S)</td>
</tr>
<tr>
<td>212</td>
<td>number of particles of class #2</td>
</tr>
<tr>
<td>5</td>
<td>radius of particles of class #2</td>
</tr>
<tr>
<td>1</td>
<td>phase ID of particles of class #2 (C₃S)</td>
</tr>
<tr>
<td>3</td>
<td>menu selection to add one-pixel particles</td>
</tr>
<tr>
<td>112776</td>
<td>number of one pixel particles to add</td>
</tr>
<tr>
<td>10</td>
<td>phase ID of one pixel particles</td>
</tr>
<tr>
<td>5</td>
<td>menu selection to measure global phase fractions</td>
</tr>
<tr>
<td>6</td>
<td>menu selection to measure phase fractions vs. distance from aggregate surface</td>
</tr>
<tr>
<td>4</td>
<td>menu selection to execute hydration</td>
</tr>
<tr>
<td>1</td>
<td>selection to specify desired degree of hydration</td>
</tr>
<tr>
<td>0.7</td>
<td>degree of hydration desired</td>
</tr>
<tr>
<td>10000</td>
<td>max. # of diffusion steps per cycle</td>
</tr>
<tr>
<td>0.001</td>
<td>maximum probability of CH nucleation</td>
</tr>
<tr>
<td>200000</td>
<td>exponential scale factor for CH nucleation</td>
</tr>
<tr>
<td>5</td>
<td>menu selection to measure global phase fractions</td>
</tr>
<tr>
<td>6</td>
<td>menu selection to measure phase fractions vs. distance from aggregate surface</td>
</tr>
<tr>
<td>9</td>
<td>menu selection to exit program</td>
</tr>
</tbody>
</table>

The major results of this example are the phase fractions as a function of distance from the aggregate surface for the two systems. The global phase fraction data can be used to assure that 70% hydration has been achieved. The phase fraction vs. distance data were imported into a spreadsheet program to produce graphs of the various phase fractions as a function of distance for systems containing large and small mineral admixtures. These graphs are given in Figures 3, 4, and 5.

Figure 3 shows the original particle packings. As shown previously [7], the cement particles are not able to pack efficiently near the aggregate surface so that the porosity in the interfacial zone increases relative to that in the bulk paste. This also holds true for the larger mineral admixture particles. However, the smaller (one pixel) mineral admixture particles are able to pack much more efficiently than the cement particles against the aggregate surface and, since this is the highest porosity region, there are actually more mineral admixture particles in the interfacial zone than in the bulk paste for this system.

The differences in original packings for the two systems manifest themselves in different microstructures developing as hydration occurs. For systems with no mineral admixtures, it has been shown that the interfacial zone is higher in porosity and CH and lower in C-S-H and C₃S than the bulk paste [7]. The effects of mineral admixtures on this baseline microstructure are dependent on mineral admixture particle size as shown in Figure 4. Both particle sizes decrease the CH formed due to the
Figure 3. Original Particle Packings for a) Large and b) Small Mineral Admixture Particles
Figure 4. Phase Fractions vs. Distance from Aggregate Surface After Hydration for a) Large and b) Small Mineral Admixture Particles
occurrence of the pozzolanic reaction but a greater reduction is achieved with the smaller particles as the pozzolanic surfaces are distributed more uniformly throughout the microstructure. In fact, because with the one pixel admixture particles there is actually more pozzolanic material near the aggregate surface (Figure 3), there is also more pozzolanic C-S-H formed in the interfacial zone than in the bulk. This will aid in offsetting the deficiency in primary C-S-H (that formed directly from C₃S hydration) normally found in this zone.

To quantitatively examine this effect, the total of the C₃S, C-S-H, and pozzolanic (admixture and C-S-H) phases are plotted as a function of distance from the aggregate surface in Figure 5. From the figure, it is evident that the smaller admixture particles result in a much more homogeneous distribution of these phases as the decrease in this total phase fraction present as the aggregate surface is approached is suppressed. While the effects of admixture particle size and reactivity have been investigated in more detail [8], this simple study would suggest that adequate dispersion of silica fume is needed to maximize its performance in concrete, as the microstructure of an agglomerated system has been shown to be inferior to that of a well-dispersed one.

Figure 5. Total C₃S + C-S-H + Admixture Phase Fraction vs. Distance from Aggregate Surface After Hydration
4.2: EFFECT OF W/C RATIO ON CONNECTIVITY OF CAPILLARY POROSITY

For this example, we will vary the w/c ratio of ordinary cement paste and monitor the percolation or connectivity of the capillary porosity as hydration occurs. That is, we will hydrate a cement microstructure for a few cycles, measure the connectivity of the porosity phase, hydrate a few more cycles, etc. The w/c ratios employed are 0.35 and 0.5. The cement particles will consists of equal numbers of four sizes of particles, with diameters of 15, 11, 7, and 3 pixels respectively. Since each set of particles (one each of each of the four sizes) occupies 2728 pixels, 173 of each will provide a w/c ratio of 0.3497 while 141 of each will yield a w/c ratio of 0.4999. Because the hydration rate is faster during the early cycles of the hydration process, the number of cycles executed to obtain each data point will be increased as the hydration proceeds.

The data files (hydrate.inp) used to execute the two simulations are as follows:

Examples 1 (2): Simulations with w/c = 0.3497 (w/c = 0.4999)

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>8291</td>
<td>random number seed</td>
</tr>
<tr>
<td>2</td>
<td>menu selection to add cement particles</td>
</tr>
<tr>
<td>4</td>
<td>number of classes of particles to add</td>
</tr>
<tr>
<td>173 (141)</td>
<td>number of particles of class #1 to add</td>
</tr>
<tr>
<td>7</td>
<td>radius of particles of class #1</td>
</tr>
<tr>
<td>1</td>
<td>phase ID of particles of class #1 (C₃S)</td>
</tr>
<tr>
<td>173 (141)</td>
<td>number of particles of class #2 to add</td>
</tr>
<tr>
<td>5</td>
<td>radius of particles of class #2</td>
</tr>
<tr>
<td>1</td>
<td>phase ID of particles of class #2</td>
</tr>
<tr>
<td>173 (141)</td>
<td>number of particles of class #3 to add</td>
</tr>
<tr>
<td>3</td>
<td>radius of particles of class #3</td>
</tr>
<tr>
<td>1</td>
<td>phase ID of particles of class #3</td>
</tr>
<tr>
<td>173 (141)</td>
<td>number of particles of class #4 to add</td>
</tr>
<tr>
<td>1</td>
<td>radius of particles of class #4</td>
</tr>
<tr>
<td>1</td>
<td>phase ID of particles of class #4</td>
</tr>
<tr>
<td>5</td>
<td>menu selection to measure global phase fractions</td>
</tr>
<tr>
<td>7</td>
<td>menu selection to measure single phase connectivity</td>
</tr>
<tr>
<td>0</td>
<td>phase ID of which to assess percolation (porosity)</td>
</tr>
<tr>
<td>4</td>
<td>menu selection to execute hydration</td>
</tr>
<tr>
<td>0</td>
<td>selection to specify number of hydration cycles</td>
</tr>
<tr>
<td>1</td>
<td>number of hydration cycles to execute</td>
</tr>
<tr>
<td>10000</td>
<td>max. # of diffusion steps per cycle</td>
</tr>
<tr>
<td>0.001</td>
<td>maximum probability for CH nucleation</td>
</tr>
<tr>
<td>200000.</td>
<td>exponential scale factor for CH nucleation</td>
</tr>
<tr>
<td>5</td>
<td>menu selection to measure global phase fractions</td>
</tr>
<tr>
<td>7</td>
<td>menu selection to measure single phase connectivity</td>
</tr>
</tbody>
</table>
phase ID of which to assess percolation (porosity)
select number of hydration cycles
number of hydration cycles to execute
max. # of diffusion steps per cycle
maximum probability for CH nucleation
exponential scale factor for CH nucleation
menu selection to measure global phase fractions
menu selection to measure single phase connectivity
phase ID of which to assess percolation (porosity)
menu selection to execute hydration
select number of hydration cycles to execute
max. # of diffusion steps per cycle
maximum probability for CH nucleation
exponential scale factor for CH nucleation
menu selection to measure global phase fractions
menu selection to measure single phase connectivity
phase ID of which to assess percolation (porosity)
menu selection to execute hydration
select number of hydration cycles to execute
max. # of diffusion steps per cycle
maximum probability for CH nucleation
exponential scale factor for CH nucleation
menu selection to measure global phase fractions
menu selection to measure single phase connectivity
phase ID of which to assess percolation (porosity)
menu selection to execute hydration
select number of hydration cycles to execute
max. # of diffusion steps per cycle
maximum probability for CH nucleation
exponential scale factor for CH nucleation
menu selection to measure global phase fractions
<table>
<thead>
<tr>
<th>Menu Selection</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Measure single phase connectivity</td>
</tr>
<tr>
<td>0</td>
<td>Phase ID of which to assess percolation (porosity)</td>
</tr>
<tr>
<td>4</td>
<td>Execute hydration</td>
</tr>
<tr>
<td>0</td>
<td>Specify number of hydration cycles</td>
</tr>
<tr>
<td>4</td>
<td>Number of hydration cycles to execute</td>
</tr>
<tr>
<td>5000</td>
<td>Max. # of diffusion steps per cycle</td>
</tr>
<tr>
<td>0.001</td>
<td>Maximum probability for CH nucleation</td>
</tr>
<tr>
<td>200000.</td>
<td>Exponential scale factor for CH nucleation</td>
</tr>
<tr>
<td>5</td>
<td>Measure global phase fractions</td>
</tr>
<tr>
<td>7</td>
<td>Measure single phase connectivity</td>
</tr>
<tr>
<td>0</td>
<td>Phase ID of which to assess percolation (porosity)</td>
</tr>
<tr>
<td>4</td>
<td>Execute hydration</td>
</tr>
<tr>
<td>0</td>
<td>Specify number of hydration cycles</td>
</tr>
<tr>
<td>10</td>
<td>Number of hydration cycles to execute</td>
</tr>
<tr>
<td>5000</td>
<td>Max. # of diffusion steps per cycle</td>
</tr>
<tr>
<td>0.001</td>
<td>Maximum probability for CH nucleation</td>
</tr>
<tr>
<td>200000.</td>
<td>Exponential scale factor for CH nucleation</td>
</tr>
<tr>
<td>5</td>
<td>Measure global phase fractions</td>
</tr>
<tr>
<td>7</td>
<td>Measure single phase connectivity</td>
</tr>
<tr>
<td>0</td>
<td>Phase ID of which to assess percolation (porosity)</td>
</tr>
<tr>
<td>4</td>
<td>Execute hydration</td>
</tr>
<tr>
<td>0</td>
<td>Specify number of hydration cycles</td>
</tr>
<tr>
<td>15</td>
<td>Number of hydration cycles to execute</td>
</tr>
<tr>
<td>5000</td>
<td>Max. # of diffusion steps per cycle</td>
</tr>
<tr>
<td>0.001</td>
<td>Maximum probability for CH nucleation</td>
</tr>
<tr>
<td>200000.</td>
<td>Exponential scale factor for CH nucleation</td>
</tr>
<tr>
<td>5</td>
<td>Measure global phase fractions</td>
</tr>
<tr>
<td>7</td>
<td>Measure single phase connectivity</td>
</tr>
<tr>
<td>0</td>
<td>Phase ID of which to assess percolation (porosity)</td>
</tr>
<tr>
<td>4</td>
<td>Execute hydration</td>
</tr>
<tr>
<td>0</td>
<td>Specify number of hydration cycles</td>
</tr>
<tr>
<td>55</td>
<td>Number of hydration cycles to execute</td>
</tr>
<tr>
<td>5000</td>
<td>Max. # of diffusion steps per cycle</td>
</tr>
<tr>
<td>0.001</td>
<td>Maximum probability for CH nucleation</td>
</tr>
<tr>
<td>200000.</td>
<td>Exponential scale factor for CH nucleation</td>
</tr>
<tr>
<td>5</td>
<td>Measure global phase fractions</td>
</tr>
<tr>
<td>7</td>
<td>Measure single phase connectivity</td>
</tr>
<tr>
<td>0</td>
<td>Phase ID of which to assess percolation (porosity)</td>
</tr>
<tr>
<td>4</td>
<td>Execute hydration</td>
</tr>
<tr>
<td>0</td>
<td>Specify number of hydration cycles</td>
</tr>
<tr>
<td>100</td>
<td>Number of hydration cycles to execute</td>
</tr>
<tr>
<td>5000</td>
<td>Max. # of diffusion steps per cycle</td>
</tr>
<tr>
<td>0.001</td>
<td>Maximum probability for CH nucleation</td>
</tr>
<tr>
<td>200000.</td>
<td>Exponential scale factor for CH nucleation</td>
</tr>
</tbody>
</table>
Based on the output from the menu selections to measure global phase fractions and to measure phase connectivity, the fraction of the total porosity which forms a connected path across the three-dimensional microstructure can be determined as a function of degree of hydration (amount of cement reacted). Figure 6 provides a plot of this connected fraction vs. degree of hydration for the two w/c ratios used in this example. The connected fraction varies from 1, as initially the capillary porosity is totally connected, to 0, as ultimately the hydration products fill in the porosity to an extent that discontinuity of this phase occurs. Discontinuity of the capillary porosity would be expected to have major effects on transport properties as the primary pathways for transport would shift from being the capillary pores (μm in size) to being the gel pores in the C-S-H (nm in size). Since the higher (0.5) w/c ratio contains more porosity initially, more hydration is required to achieve this discontinuity. In fact, for high enough w/c ratios (w/c > 0.57), even at complete hydration, the capillary porosity remains continuous across the system [4].
Since the phase fractions of all phases are known throughout the hydration, one can also plot the fraction connected porosity vs. the total capillary porosity as shown in Figure 7. Now, the data for the two w/c ratios overlap, suggesting that it is the total capillary porosity fraction which controls the connectivity of porosity, at least for the range of w/c ratios and cement particle sizes investigated in this study.

Figure 7. Fraction Connected Porosity vs. Total Porosity
5. REFERENCES


This program simulates the microstructural development of cement (specifically C3S) as it hydrates. Model is image based (3-D lattice of 100*100*100 elements) and hydration is modelled as a dissolution/diffusion/reaction cyclic process. Tools are provided to assess phase fractions and phase connectivity as the hydration proceeds. Also includes the capability to add a single inert aggregate to the system and assess phase fractions as a function of distance from aggregate surface.

PHASE ID ASSIGNMENTS

0- Porosity
1- Cement (C3S)
2- Diffusing Calcium hydroxide (CH) species
3- Diffusing Calcium silicate hydrate (CSH) species
4- Solid CSH
5- Solid CH
8- Aggregate
9- Inert filler
10- Pozzolanic filler/ pozzolanic CSH

Temporary IDs
6- Surface site eligible for dissolution
7- Burnt site in percolation routine

INTEGER CEMENT(101,101,101),CYCLENO,NTIMES,NFILL,NPZR
INTEGER SYSSIZE, AGGSIZE, NCEMENT, NCEMDIS
INTEGER USERC
INTEGER*4 ISEED
DIMENSION R(97)
COMMON /A/CEMENT, CYCLENO, NTIMES, NFILL, NPZR, ISEED, SYSSIZE,
   + AGGSIZE, NCEMENT, NCEMDIS
COMMON /FR/IX1, IX2, IX3, R

C
C 3-D microstructure is stored in the 3-D array CEMENT
C
System size set at 100
C
SYSSIZE = 100
C
Clear the microstructure to all porosity
C
DO 10 I = 1, SYSSIZE
DO 10 J = 1, SYSSIZE
DO 10 K = 1, SYSSIZE
10   CEMENT(I, J, K) = 0
C
Call Routine to display an introduction
C
CALL INTRO

C  NCEMENT is counter for cement originally present
C  NCEMDIS is counter for cement reacted so far
C  NFILL is counter of number of pozzolanic filler species
C  NPZR is counter for number of pozzolanic filler species
   which have been consumed
C
NCEMENT = 0
NCEMDIS = 0
NFILL = 0
NPZR = 0
AGGSIZE = 0
WRITE(6, *)’ENTER RANDOM NUMBER SEED (Integer)’
READ(5, *) ISEED
WRITE(6, *) ISEED

C  Present user with menu of choices and execute
C  appropriate option until user elects to stop
C
WRITE(6,*)'INPUT USER CHOICE'
WRITE(6,*) 1) ADD A FLAT INERT AGGREGATE TO MICROSTRUCTURE'
WRITE(6,*) 2) ADD SPHERICAL PARTICLES TO MICROSTRUCTURE'
WRITE(6,*) 3) ADD ONE-PIXEL FILLER PARTICLES TO MICROSTRUCTURE'
WRITE(6,*) 4) HYDRATE MICROSTRUCTURE'
WRITE(6,*) 5) MEASURE PHASE FRACTIONS'
WRITE(6,*) 6) MEASURE PHASE FRACTIONS AS A FUNCTION'
WRITE(6,*) OF DISTANCE FROM AGGREGATE SURFACE'
WRITE(6,*) 7) MEASURE SINGLE PHASE CONNECTIVITY'
WRITE(6,*) 8) MEASURE TOTAL SOLIDS CONNECTIVITY'
WRITE(6,*) 9) EXIT'
READ(5,*)USERC
WRITE(6,*)USERC
IF(USERC.EQ.1) THEN
   CALL ADDAGG
ELSE IF(USERC.EQ.2) THEN
   CALL CREATE
ELSE IF (USERC.EQ.3) THEN
   CALL FILLER
ELSE IF (USERC.EQ.4) THEN
   CALL HYDRATE
ELSE IF (USERC.EQ.5) THEN
   CALL MEASURE
ELSE IF (USERC.EQ.6) THEN
   CALL MEASAGG
ELSE IF (USERC.EQ.7) THEN
   CALL CONNECT
ELSE IF (USERC.EQ.8) THEN
   CALL CONSOLD
ELSE IF (USERC.EQ.9) THEN
   GOTO 50
ENDIF
GOTO 40
50 STOP
END
FUNCTION RAN1(IDUM)

C Portable random number generator, RAN1
C To generate uniform random deviates between 0 and 1.0
C From: Numerical Recipes in Fortran
         Press, Flannery, Teukolsky, and Vetterling
C
DIMENSION R(97)
PARAMETER (M1 = 259200,IA1 = 7141,IC1 = 54773,RM1 = 1./M1)
PARAMETER (M2 = 134456, IA2 = 8121, IC2 = 28411, RM2 = 1./M2)
PARAMETER (M3 = 243000, IA3 = 4561, IC3 = 51349)
COMMON /FR/IX1,IX2,IX3,R
DATA IFF /0/

IF(IDUM.LT.0.OR.IFF.EQ.0) THEN
  IFF = 1
  IX1 = MOD(IC1-IDUM,M1)
  IX1 = MOD(IA1*IX1 + IC1,M1)
  IX2 = MOD(IX1,M2)
  IX1 = MOD(IA1*IX1 + IC1,M1)
  IX3 = MOD(IX1,M3)
  DO 11 J = 1,97
    IX1 = MOD(IA1*IX1 + IC1,M1)
    IX2 = MOD(IA2*IX2 + IC2,M2)
    R(J) = (FLOAT(IX1) + FLOAT(IX2)*RM2)*RM1
  11  CONTINUE
  IDUM = 1
ENDIF
IX1 = MOD(IA1*IX1 + IC1,M1)
IX2 = MOD(IA2*IX2 + IC2,M2)
IX3 = MOD(IA3*IX3 + IC3,M3)
J = 1 + (97*IX3)/M3
IF(J.GT.97.0.OR.J.LT.1) THEN
  WRITE(6,*),'ERROR IN RANDOM NUMBER GENERATOR'
ENDIF
RAN1 = R(J)
R(J) = (FLOAT(IX1) + FLOAT(IX2)*RM2)*RM1
RETURN
END

SUBROUTINE INTRO

Subroutine INTRO to display an introduction to HYDRA3D
Called by: Main Program

WRITE(6,*),'Welcome to HYDRA3D, a digital-image-based cement'
WRITE(6,*),'microstructural model. This program has been '
WRITE(6,*),'developed to simulate the microstructural '
WRITE(6,*),'development of cement, specifically tricalcium'
WRITE(6,*),'silicate, C3S, as it reacts with water. In '
WRITE(6,*),'addition to C3S, the user may also add a single'
WRITE(6,*),'inert aggregate and/or inert or pozzolanic mineral'
WRITE(6,*),'admixture particles to the microstructure. In'
WRITE(6,*),'addition to hydration, the user may also assess'
WRITE(6,*)'phase fractions (either globally or as a function')
WRITE(6,*)'of distance from the aggregate surface) and '
WRITE(6,*)'the connectivity of any individual phase or of'
WRITE(6,*)'the total solids (neglecting any aggregate) present'
WRITE(6,*)'in the system.'
WRITE(6,*)'
RETURN
END
SUBROUTINE ADDAGG
INTEGER CEMENT(101,101,101),CYCLENO,NTIMES,NFILL,NPZR
INTEGER SYSSIZE,AGGSIZE,NCEMENT,NCEMDIS
INTEGER*4 ISEED
COMMON /A/CEMENT,CYCLENO,NTIMES,NFILL,NPZR,ISEED,SYSSIZE,
+ AGGSIZE,NCEMENT,NCEMDIS
INTEGER IX,IY,IZ

C Subroutine ADDAGG to add a flat aggregate of user-specified
C thickness to microstructure
C Called by: Main program
C
WRITE(6,*)'ENTER AGGREGATE THICKNESS (even integer)'
READ(5,*)AGGSIZE
WRITE(6,*)AGGSIZE

DO 45 IX = ((SYSSIZE-AGGSIZE + 2)/2),((SYSSIZE + AGGSIZE)/2)
DO 45 IY = 1,SYSSIZE
DO 45 IZ = 1,SYSSIZE
45 CEMENT(IX,IY,IZ) = 8
RETURN
END
SUBROUTINE CREATE
INTEGER CEMENT(101,101,101),CYCLENO,NTIMES,NFILL,NPZR
INTEGER SYSSIZE,AGGSIZE,NCEMENT,NCEMDIS
INTEGER*4 ISEED
COMMON /A/CEMENT,CYCLENO,NTIMES,NFILL,NPZR,ISEED,SYSSIZE,
+ AGGSIZE,NCEMENT,NCEMDIS
INTEGER NUMSIZE,SPHNUM(10),SPHRAD(10),SPHID(10)

C Subroutine CREATE to obtain user input to create a starting system
C consisting of digitized spheres of various radii and phase IDs
C Called by: Main program
C Calls: Subroutine GSPHERE
C
11 WRITE(6,*)'ENTER NUMBER OF DIFFERENT SIZE SPHERES TO USE(MAX 10)'

32
READ(5,*)NUMSIZE
IF(NUMSIZE.LT.1.OR.NUMSIZE.GT.10) THEN
   GOTO 11
ENDIF
WRITE(6,*)NUMSIZE
WRITE(6,*)'ENTER NUMBER, SIZE, AND PHASE FOR EACH CLASS'
WRITE(6,*)'(LARGEST RADIUS 1ST)'
DO 21 I = 1,NUMSIZE
   WRITE(6,*)'ENTER NUMBER OF SPHERES OF CLASS',I
   READ(5,*)SPHNUM(I)
   WRITE(6,*)SPHNUM(I)
   WRITE(6,*)'ENTER RADIUS OF SPHERES OF CLASS',I
   WRITE(6,*)'(Integer <= 10 PLEASE)'
   READ(5,*)SPHRAD(I)
   WRITE(6,*)SPHRAD(I)
   WRITE(6,*)'ENTER PHASE ID TO BE ASSIGNED TO SPHERES OF CLASS',I
   WRITE(6,*)'(1- C3S, 9- Inert filler 10- Pozzolanic filler'
   READ(5,*)SPHID(I)
21 WRITE(6,*)SPHID(I)
C
CALL GSPHERE(NUMSIZE,SPHNUM,SPHRAD,SPHID)
RETURN
END
SUBROUTINE GSPHERE(NUMGEN,NUMEACH,SIZEEACH,PHEACH)
INTEGER CEMENT(101,101,101),CYCLENO,NTIMES,NFILL,NPZR
INTEGER SYSSIZE,AGGSIZE,NCEMENT,NCEMDIS
INTEGER*4 ISEED
COMMON /A/CEMENT,CYCLENO,NTIMES,NFILL,NPZR,ISEED,SYSSIZE,
   + AGGSIZE,NCEMENT,NCEMDIS
INTEGER NUMGEN,NUMEACH(10),SIZEEACH(10),PHEACH(10),RADIUS
INTEGER PHSPH,COUNT,X,Y,Z
REAL RX,RY,RZ
C
Subroutine GSPHERE to generate spheres of a fixed number of size
classes
Input: Number of size classes, number, size, and ID of spheres
   in each size class
Called by: Subroutine CREATE
Calls: Subroutine CHKSPH
C
Generate the requested number of each size of sphere
C
DO 131 I = 1,NUMGEN
   RADIUS = SIZEEACH(I)
C
PHSPH = PHEACH(I)
DO 131 J = 1, NUMEACH(I)
20  COUNT = 0

C
C  Generate a random center location for this sphere
C
RX = RAN1(ISEED)
RY = RAN1(ISEED)
RZ = RAN1(ISEED)
X = SYSSIZE*RX + 1
Y = SYSSIZE*RY + 1
Z = SYSSIZE*RZ + 1

C
C  Call routine to check is sphere can be placed at chosen location
C
CALL CHKSPH(COUNT,X,Y,Z,RADIUS,1,PHSPH)
IF(COUNT.NE.0) THEN
GOTO 20
ENDIF

C
C  Place the sphere at the selected location
C
131  CALL CHKSPH(COUNT,X,Y,Z,RADIUS,2,PHSPH)
RETURN
END

SUBROUTINE CHKSPH(SFLG,XIN,YIN,ZIN,RADD,WFLG,PHID)
INTEGER CEMENT(101,101,101), CYCLENO, NTIMES, NFILL, NPZR
INTEGER SYSSIZE, AGGSIZE, NCEMENT, NCEMDIS
INTEGER*4 ISEED
COMMON /A/ CEMENT, CYCLENO, NTIMES, NFILL, NPZR, ISEED, SYSSIZE,
+ AGGSIZE, NCEMENT, NCEMDIS
INTEGER SFLG, XIN, YIN, ZIN, RADD, WFLG, PHID, NOFITS
INTEGER XP, YP, ZP
REAL DIST

C
C  Subroutine CHKSPH to either check or perform sphere location
C  Inputs: Flag used to send back indication if sphere fits,
C          x, y, and z coordinates of sphere center, radius of
C          sphere, and flag indicating if sphere is to be
C          checked (1) or placed (2)
C  Called by: Subroutine GSPHERE
C
C
NOFITS = 0
Check all pixels within the sphere

DO 221 I = XIN-RADD, XIN + RADD
DO 221 J = YIN-RADD, YIN + RADD
DO 221 K = ZIN-RADD, ZIN + RADD
XP = I
YP = J
ZP = K

Use periodic boundaries to wrap a sphere from one side of the 3-D system to the other

IF(XP.LT.1) THEN
  XP = SYSSIZE + XP
ENDIF
IF(XP.GT.SYSSIZE) THEN
  XP = XP - SYSSIZE
ENDIF
IF(YP.LT.1) THEN
  YP = SYSSIZE + YP
ENDIF
IF(YP.GT.SYSSIZE) THEN
  YP = YP - SYSSIZE
ENDIF
IF(ZP.LT.1) THEN
  ZP = SYSSIZE + ZP
ENDIF
IF(ZP.GT.SYSSIZE) THEN
  ZP = ZP - SYSSIZE
ENDIF

Compute the distance from the center of the sphere to this point

DIST = SQRT(FLOAT((I-XIN) * (I-XIN) + (J-YIN) * (J-YIN) +
& (K-ZIN) * (K-ZIN)))
IF((DIST-0.5).LE.RADD) THEN
  IF(WFLG.EQ.2) THEN
    CEMENT(XP, YP, ZP) = PHID
ENDIF

Update counter of C3S species if necessary

IF(PHID.EQ.1) THEN
  NCEMENT = NCEMENT + 1
ENDIF
Update counter of pozzolanic filler species if necessary

IF(PHID.EQ.10) THEN
  NFILL = NFILL + 1
ENDIF

IF(WFLG.EQ.1.AND.CEMENT(XP,YP,ZP).NE.0) THEN
  Sphere can not be fit at present location without overlapping
  some other sphere so notify calling routine of lack of fit

  NOFITS = 1
  GOTO 231
ENDIF
ENDIF
CONTINUE

231  SFLG = NOFITS
RETURN
END

SUBROUTINE FILLER

INTEGER CEMENT(101,101,101),CYCLENO,NTIMES,NFILL,NPZR
INTEGER SYSSIZE,AGGSIZE,NCEMENT,NCEMDIS
INTEGER*4 ISEED
COMMON /A/CEMENT,CYCLENO,NTIMES,NFILL,NPZR,ISEED,SYSSIZE,
  + AGGSIZE,NCEMENT,NCEMDIS
INTEGER FILLID,NADD

Subroutine FILLER to obtain user input as to number and type of
filler particles
Called by: Main program
Calls: Subroutine GENFILL

WRITE(6,*)'ENTER NUMBER OF FILLER PARTICLES TO ADD'
READ(5,*)NADD
WRITE(6,*)NADD
WRITE(6,*)'ENTER FILLER ID TO USE (10-POZZOLANIC, 9-INERT)'
READ(5,*)FILLID
WRITE(6,*)FILLID
IF(FILLID.NE.9.AND.FILLID.NE.10) THEN
  STOP
ENDIF
CALL GENFILL(NADD,FILLID)
IF(FILLID.EQ.10) THEN
NFILL = NFILL + NADD
ENDIF
RETURN
END
SUBROUTINE GENFILL(NTOPL,FILVAL)
INTEGER CEMENT(101,101,101),CYCLENO,NTIMES,NFILL,NPZR
INTEGER SYSSIZE,AGGSIZE,NCEMENT,NCEMDis
INTEGER*4 ISEED
COMMON /A/CEMENT,CYCLENO,NTIMES,NFILL,NPZR,ISEED,SYSSIZE,
+ AGGSIZE,NCEMENT,NCEMDis
INTEGER NTOPL,FILVAL,PFILL,XFILL,YFILL,ZFILL
REAL RXF,RYF,RZF
C
C Subroutine GENFILL to locate one pixel filler particles at random
C unoccupied locations in the 3-D microstructure
C Input: Number and ID of filler particles to generate
C Called by: Subroutine FILLER
C
DO 683 I = 1,NTOPL
PFILL = 0
C
PFILL indicates successful placement of this filler particle
C
Generate a random location and attempt to place filler particle
there
C
684  RXF = RAN1(ISEED)
    RYF = RAN1(ISEED)
    RZF = RAN1(ISEED)
    XFILL = SYSSIZE*RXF + 1
    YFILL = SYSSIZE*RYF + 1
    ZFILL = SYSSIZE*RZF + 1
C
    IF(CEMENT(XFILL,YFILL,ZFILL).EQ.0) THEN
      PFILL = 1
      CEMENT(XFILL,YFILL,ZFILL) = FILVAL
    ENDIF
    IF(PFILL.EQ.0) THEN
      GOTO 684
    ENDIF
683  CONTINUE
RETURN
END
SUBROUTINE HYDRATE
INTEGER CEMENT(101,101,101),CYCLENO,NTIMES,NFILL,NPZR
INTEGER SYSSIZE,AGGSIZE,CEMENT,NCEMDIS
INTEGER*4 ISEED
COMMON /A/CEMENT,CYCLENO,NTIMES,NFILL,NPZR,ISEED,SYSSIZE,
+ AGGSIZE,CEMENT,NCEMDIS
INTEGER NITER,NUMANTS,ANTSX(500000),ANTSY(500000)
INTEGER ANTSZ(500000),XANT,YANT,ZANT,NUMLEFT,NBLEFT,ALIVE
INTEGER NBLUE,ANTTYPE,STOPCH
REAL NUCPROB,NUCSCALE,BETERM,BBIAS,ALPHA,ALMAX

C Subroutine HYDRATE to obtain user input and control execution
C of a number of hydration cycles
C Called by: MAIN program
C Calls: Subroutines DISSOLV, MVCHANT, MCSHANT
C
C NITER is number of hydration cycles to perform
C NTIMES is number of random steps diffused per cycle
C before all diffusing species convert to solid
C at current location
C NUCPROB and NUCSCALE are parameters to control the number
C and size of nucleating CH crystals by controlling
C the nucleation probability
C
WRITE(6,*)’DO YOU WISH TO SPECIFY 0) MAX. # OF CYCLES OR’
WRITE(6,*)’1) MAXIMUM DEGREE OF HYDRATION’
READ(5,*)STOPCH
IF(STOPCH.EQ.0) THEN
   WRITE(6,*)’ENTER NO. OF CYCLES (ITERATIONS) TO PERFORM’
   READ(5,*)NITER
   WRITE(6,*)NITER
   ALMAX = 1.0
ENDIF
IF(STOPCH.EQ.1) THEN
   WRITE(6,*)’ENTER DESIRED DEGREE OF HYDRATION’
   READ(5,*)ALMAX
   WRITE(6,*)ALMAX
   NITER = 5000
ENDIF
WRITE(6,*)’ENTER MAX. NUMBER OF DIFFUSION STEPS PER CYCLE’
READ(5,*)NTIMES
WRITE(6,*)NTIMES
WRITE(6,*)’ENTER MAXIMUM PROBABILITY FOR CH NUCLEATION (0.0-1.0)’
READ(5,*)NUCPROB
WRITE(6,*)NUCPROB

38
WRITE(6, *)'ENTER EXPONENTIAL SCALE FACTOR FOR CH NUCLEATION'
READ(5, *)NUCSCALE
WRITE(6, *)NUCSCALE

Loop for each hydration cycle

ALPHA = FLOAT(NCEMDIS)/FLOAT(NCEMENT)
DO 22 I = 1, NITER
  IF(ALPHA.GE.ALMAX) THEN
    GOTO 28
  ENDIF

NUMANTS is counter for number of diffusing species generated

    NUMANTS = 0
    CALL DISSOLV(ANTSX, ANTSY, ANTSZ, NUMANTS)
    NUMLEFT = NUMANTS
    ALPHA = FLOAT(NCEMDIS)/FLOAT(NCEMENT)
    WRITE(6, *)'NUMBER DISSOLVED = ', NUMANTS

NBLUE is counter for number of diffusing CH species

    NBLUE = 61 * NUMANTS/231

Loop for each diffusion step

    DO 32 J = 1, NTIMES
        CYCLENO = J
        IF(NUMLEFT.EQ.0) THEN
            GOTO 22
       ENDIF
        NUMLEFT = 0
        NLEFT = 0

Compute probability of CH nucleation based on current system

    BETERM = EXP(-FLOAT(NBLUE)/NUCSCALE)
    BBIAS = NUCPROB*(1.-BETERM)

Loop for each diffusing species remaining
DO 52 K = 1, NUMANTS

C Obtain the location and type of this diffusing species

C XANT = ANTSX(K)
YANT = ANTSY(K)
ZANT = ANTSZ(K)
ANTTYPE = CEMENT(XANT, YANT, ZANT)

C ALIVE indicates if species is still diffusing or has become solid

C ALIVE = 1
IF((ANTTYPE.NE.2).AND.(ANTTYPE.NE.3)) THEN
  WRITE(6,*)'Error- Cement pixel is ',ANTTYPE
ENDIF
IF(ANTTYPE.EQ.2) THEN
  Call routine to move a CH diffusing species
  CALL MVCHANT(XANT, YANT, ZANT, ALIVE, BBIAS)
ENDIF
IF(ANTTYPE.EQ.3) THEN
  Call routine to move a CSH diffusing species
  CALL MCSHANT(XANT, YANT, ZANT, ALIVE)
ENDIF
IF(ALIVE.EQ.1) THEN
  NUMLEFT = NUMLEFT + 1
ENDIF
C Store new location of diffusing species if still alive
C
C ANTSX(NUMLEFT) = XANT
ANTSX(NUMLEFT) = YANT
ANTSZ(NUMLEFT) = ZANT
IF(ANTTYPE.EQ.2) THEN
  NBLEFT = NBLEFT + 1
ENDIF
ENDIF
52 CONTINUE
NUMANTS = NUMLEFT
NBLUE = NBLEFT
32 CONTINUE
22 CONTINUE
28 RETURN
END
SUBROUTINE DISSOLV(DISX, DISY, DISZ, NUMDIS)
INTEGER CEMENT(101,101,101),CYCLEN0,NTIMES,NFILL,NPZR
INTEGER SYSSIZE,AGGSIZE,NCEMENT,NCEMDIS
INTEGER*4 ISEED
COMMON /A/CEMENT,CYCLENO,NTIMES,NFILL,NPZR,ISEED,SYSSIZE,
+ AGGSIZE,NCEMENT,NCEMDIS
INTEGER DISX(500000),DISY(500000),DISZ(500000),NUMDIS
INTEGER XC,YC,ZC,NEDGE,NCHADD,NCSHADD,NUMORG
REAL RDIS,RDX,RDY,RDZ
INTEGER IRND

C Subroutine DISSOLV to perform dissolution at beginning of hydration cycle
C Returns arrays of x,y, and z coordinates of generated diffusing species and number of species generated
C Called by: Subroutine HYDRATE
C
NUMDIS=0

C HIGHLIGHT ALL C3S EDGE POINTS

DO 15 I=1,SYSSIZE
DO 15 J=1,SYSSIZE
DO 15 K=1,SYSSIZE
IF(CEMENT(I,J,K).EQ.1) THEN
  NEDGE=1

C Check all six neighbors in 3-D to see if pixel is on edge
C Note that periodic boundaries are employed throughout

XC=I-1
IF(XC.LT.1) XC=SYSSIZE
NEDGE=NEDGE*CEMENT(XC,J,K)
XC=I+1
IF(XC.GT.SYSSIZE) XC=1
NEDGE=NEDGE*CEMENT(XC,J,K)
YC=J-1
IF(YC.LT.1) YC=SYSSIZE
NEDGE=NEDGE*CEMENT(I,YC,K)
YC=J+1
IF(YC.GT.SYSSIZE) YC=1
NEDGE=NEDGE*CEMENT(I,YC,K)
ZC=K-1
IF(ZC.LT.1) ZC=SYSSIZE
NEDGE=NEDGE*CEMENT(I,J,ZC)

41
\[ ZC = K + 1 \]
\[ \text{IF}(ZC \gt SYSSIZE) \ ZC = 1 \]
\[ \text{NEDGE} = \text{NEDGE} \times \text{CEMENT}(I, J, ZC) \]

\[ \text{C} \]
\[ \text{C} \]
\[ \text{C} \]
\[ \text{C} \]

If on edge, assign temporary ID for second stage

\[ \text{IF}(\text{NEDGE} \text{EQ} 0) \ \text{CEMENT}(I, J, K) = 6 \]
\[ \text{ENDIF} \]
\[ 15 \ \text{CONTINUE} \]

\[ \text{C} \]
\[ \text{C} \]
\[ \text{C} \]

Randomly dissolve all edge points

\[ \text{DO 25} \ I = 1, \text{SYSSIZE} \]
\[ \text{DO 25} \ J = 1, \text{SYSSIZE} \]
\[ \text{DO 25} \ K = 1, \text{SYSSIZE} \]
\[ \text{IF}(\text{CEMENT}(I, J, K) \text{EQ} 6) \ \text{THEN} \]

\[ \text{C} \]
\[ \text{C} \]
\[ \text{C} \]

Dissolution is performed by having pixel of interest execute a one-step random walk and seeing if it steps into pore space

\[ \text{RDIS} = \text{RAN1}(\text{ISEED}) \]
\[ \text{IRND} = 6 \times \text{RDIS} + 1 \]
\[ \text{XC} = I \]
\[ \text{YC} = J \]
\[ \text{ZC} = K \]
\[ \text{IF}(\text{IRND} \text{EQ} 1) \ \text{THEN} \]
\[ \quad \text{XC} = \text{XC} - 1 \]
\[ \quad \text{IF}(\text{XC} \text{LT} 1) \ \text{XC} = \text{SYSSIZE} \]
\[ \text{ENDIF} \]
\[ \text{IF}(\text{IRND} \text{EQ} 2) \ \text{THEN} \]
\[ \quad \text{XC} = \text{XC} + 1 \]
\[ \quad \text{IF}(\text{XC} \text{GT} \text{SYSSIZE}) \ \text{XC} = 1 \]
\[ \text{ENDIF} \]
\[ \text{IF}(\text{IRND} \text{EQ} 3) \ \text{THEN} \]
\[ \quad \text{YC} = \text{YC} - 1 \]
\[ \quad \text{IF}(\text{YC} \text{LT} 1) \ \text{YC} = \text{SYSSIZE} \]
\[ \text{ENDIF} \]
\[ \text{IF}(\text{IRND} \text{EQ} 4) \ \text{THEN} \]
\[ \quad \text{YC} = \text{YC} + 1 \]
\[ \quad \text{IF}(\text{YC} \text{GT} \text{SYSSIZE}) \ \text{YC} = 1 \]
\[ \text{ENDIF} \]
\[ \text{IF}(\text{IRND} \text{EQ} 5) \ \text{THEN} \]
\[ \quad \text{ZC} = \text{ZC} - 1 \]
IF(ZC.LT.1) ZC = SYSSIZE
ENDIF
IF(IRND.EQ.6) THEN
   ZC = ZC + 1
   IF(ZC.GT.SYSSIZE) ZC = 1
ENDIF
IF(CEMENT(XC,YC,ZC).EQ.0) THEN
   C
   Generate a diffusing C-S-H species at the new location
   C
   CEMENT(XC,YC,ZC) = 3
   CEMENT(I,J,K) = 0
   NUMDIS = NUMDIS + 1
   DISX(NUMDIS) = XC
   DISY(NUMDIS) = YC
   DISZ(NUMDIS) = ZC
   ELSE
   CEMENT(I,J,K) = 1
   ENDIF
   ENDIF
25 CONTINUE
C
ADD IN EXTRA DIFFUSING SPECIES
C
WRITE(6,'*')'ORIGINAL DISSOLVED = ',NUMDIS
NUMORG = NUMDIS
NCEMDIS = NCEMDIS + NUMDIS
C
Expansion factors 0.7 for C-S-H and 0.61 for CH are
taken from data of Young & Hansen
C
NCSHADD = 0.7*FLOAT(NUMDIS)
NCHADD = 0.61*FLOAT(NUMDIS)
NUMDIS = NUMDIS + NCSHADD + NCHADD
IF(NUMDIS.GT.500000) THEN
   WRITE(6,'*')'Too many dissolved species created'
   WRITE(6,'*')'Abortting execution'
   STOP
ENDIF
DO 35 I = 1,NCHADD + NCSHADD
C
Locate extra diffusing species at random unoccupied
C sites in the pore space

43
C 45    RDX = RAN1(ISEED)
      RDY = RAN1(ISEED)
      RDZ = RAN1(ISEED)
      XC = SYSSIZE*RDY + 1
      YC = SYSSIZE*RDY + 1
      ZC = SYSSIZE*RDZ + 1
      IF(CEMENT(XC,YC,ZC).EQ.0) THEN
        DISX(NUMORG + I) = XC
        DISY(NUMORG + I) = YC
        DISZ(NUMORG + I) = ZC
        IF(I.LE.NCHADD) THEN
          CEMENT(XC,YC,ZC) = 2
        ELSE
          CEMENT(XC,YC,ZC) = 3
        ENDIF
      ELSE
        GOTO 45
      ENDF
    ENDF

35    CONTINUE
RETURN
END

SUBROUTINE MVCHANT(XMCH,YMCH,ZMCH,CHYET,PNUC)
INTEGER CEMENT(101,101,101),CYCLENO,NTIMES,NFILL,NPZR
INTEGER SYSSIZE,AGGSIZE,NCEMENT,NCEMDIS
INTEGER*4 ISEED
COMMON /A/CEMENT,CYCLENO,NTIMES,NFILL,NPZR,ISEED,SYSSIZE,
   + AGGSIZE,NCEMENT,NCEMDIS
INTEGER XMCH,YMCH,ZMCH,CHYET,XM,YM,ZM,MGEN
REAL PNUC,PGEN
REAL RM

C Subroutine MVCHANT to move a diffusing CH species and check
C for nucleation or surface reaction
C Inputs: Current location of diffusing species and
         current probability for nucleation (PNUC)
C Returns: Flag indicating if reaction has occurred (CHYET)
C Called by: Subroutine HYDRATE
C Calls: Subroutine ADDEXT

C FIRST CHECK FOR NUCLEATION
C If last diffusion step in this cycle, convert to solid CH
PGEN = RAN1(ISEED)
IF((PNUC.GT.PGEN).OR.(CYCLENO.EQ.NTIMES)) THEN
    CEMENT(XMCH,YMCH,ZMCH) = 5
    CHYET = 0
ELSE
    C GENERATE A RANDOM MOVE
    RM = RAN1(ISEED)
    MGEN = 6*RM + 1
    XM = XMCH
    YM = YMCH
    ZM = ZMCH
    IF(MGEN.EQ.1) THEN
        XM = XM-1
        IF(XM.LT.1) XM = SYSSIZE
    ENDIF
    IF(MGEN.EQ.2) THEN
        XM = XM + 1
        IF(XM.GT.SYSSIZE) XM = 1
    ENDIF
    IF(MGEN.EQ.3) THEN
        YM = YM-1
        IF(YM.LT.1) YM = SYSSIZE
    ENDIF
    IF(MGEN.EQ.4) THEN
        YM = YM + 1
        IF(YM.GT.SYSSIZE) YM = 1
    ENDIF
    IF(MGEN.EQ.5) THEN
        ZM = ZM-1
        IF(ZM.LT.1) ZM = SYSSIZE
    ENDIF
    IF(MGEN.EQ.6) THEN
        ZM = ZM + 1
        IF(ZM.GT.SYSSIZE) ZM = 1
    ENDIF
ENDIF

C Check for surface reaction
C
C IF(CEMENT(XM,YM,ZM).EQ.5) THEN
    CEMENT(XMCH,YMCH,ZMCH) = 5
    CHYET = 0
ENDIF

C Check for pozzolanic reaction of diffusing CH
Each pozzolanic filler particle may react with up to 2.08 CH diffusing species. Assumes pozzolanic filler is pure silica with a molar volume of 27 cm$^3$/mole.

\[
\text{XNFILL} = \text{NFILL} \\
\text{IF}((\text{CEMENT}(X, Y, Z).EQ.10).AND.(\text{NPZR}.LE.\text{INT}(2.08\times\text{XNFILL}))) \\
& \text{THEN} \\
\text{CEMENT}(X_{\text{CH}}, Y_{\text{CH}}, Z_{\text{CH}}) = 10 \\
\text{NPZR} = \text{NPZR} + 1 \\
\text{CHYET} = 0 \\
\text{RM} = \text{RAN1}(\text{ISEED}) \\
\text{ENDIF} \\
\text{ENDIF}
\]

In pozzolanic reaction there is a probability of 0.73 that two pixels of product should be produced instead of one.

\[
\text{IF}(\text{RM}.\text{LE.}(0.73)) \text{ THEN} \\
\text{CALL ADDEXT}(X_{\text{CH}}, Y_{\text{CH}}, Z_{\text{CH}}) \\
\text{ENDIF} \\
\text{ENDIF}
\]

If new location is pore space, perform the move.

\[
\text{IF}((\text{CEMENT}(X, Y, Z).EQ.0) \text{ THEN} \\
\text{CEMENT}(X, Y, Z) = 2 \\
\text{CEMENT}(X_{\text{CH}}, Y_{\text{CH}}, Z_{\text{CH}}) = 0 \\
X_{\text{CH}} = X \\
Y_{\text{CH}} = Y \\
Z_{\text{CH}} = Z \\
\text{ENDIF} \\
\text{ENDIF} \\
\text{RETURN} \\
\text{END}
\]

SUBROUTINE ADDEXT(XINN, YINN, ZINN)
INTEGER CEMENT(101, 101, 101), CYCLENO, NTIMES, NFILL, NPZR
INTEGER SYSSIZE, AGGSIZE, NCEMENT, NCEMDIS
INTEGER*4 ISEED
COMMON /A/CEMENT, CYCLENO, NTIMES, NFILL, NPZR, ISEED, SYSSIZE, + AGGSIZE, NCEMENT, NCEMDIS
INTEGER XINN, YINN, ZINN, XCHR, YCHR, ZCHR, FCHR, MGEN1
INTEGER TFACT, TRIED(6)
REAL RMADD

Subroutine ADDEXT to add in extra pozzolanic CSH when CH...
reacts with silica fume (pozzolanic filler)
Inputs: x, y, and z coordinates of new pozz. CSH
Called by: Subroutine MVCHANT
ADD IN EXTRA POZZOLANIC CSH TO ACCOUNT FOR VOLUME BALANCE
RANDOM TRIES AT NEIGHBORING LOCATIONS (6) UNTIL SUCCESSFUL
OR ALL 6 HAVE BEEN TRIED

DO 93 I1 = 1, 6
  93 TRIED(I1) = 0
  FCHR = 0
  TFACT = 0
  RMADD = RAN1(ISEED)
  MGEN1 = 6 * RMADD + 1
  XCHR = XINN
  YCHR = YINN
  ZCHR = ZINN
  IF ((MGEN1 .EQ. 1) .AND. (TRIED(1) .EQ. 0)) THEN
    XCHR = XCHR - 1
    TRIED(1) = 1
    TFACT = TFACT + 1
    IF (XCHR .LT. 1) XCHR = SYSSIZE
  ENDIF
  IF ((MGEN1 .EQ. 2) .AND. (TRIED(2) .EQ. 0)) THEN
    XCHR = XCHR + 1
    TRIED(2) = 1
    TFACT = TFACT + 1
    IF (XCHR .GT. SYSSIZE) XCHR = 1
  ENDIF
  IF ((MGEN1 .EQ. 3) .AND. (TRIED(3) .EQ. 0)) THEN
    YCHR = YCHR - 1
    TRIED(3) = 1
    TFACT = TFACT + 1
    IF (YCHR .LT. 1) YCHR = SYSSIZE
  ENDIF
  IF ((MGEN1 .EQ. 4) .AND. (TRIED(4) .EQ. 0)) THEN
    YCHR = YCHR + 1
    TRIED(4) = 1
    TFACT = TFACT + 1
    IF (YCHR .GT. SYSSIZE) YCHR = 1
  ENDIF
  IF ((MGEN1 .EQ. 5) .AND. (TRIED(5) .EQ. 0)) THEN
    ZCHR = ZCHR - 1
    TRIED(5) = 1
  ENDIF

47
TRACT = TFACT + 1
IF(ZCHR.LT.1) ZCHR = SYSSIZE
ENDIF
IF((MGEN1.EQ.6).AND.(TRIED(6).EQ.0)) THEN
  ZCHR = ZCHR + 1
  TRIED(6) = 1
  TFACT = TFACT + 1
ENDIF
IF(ZCHR.GT.SYSSIZE) ZCHR = 1
ENDIF
IF(CEMENT(XCHR,YCHR,ZCHR).EQ.0) THEN
  CEMENT(XCHR,YCHR,ZCHR) = 10
  FCHR = 1
  GOTO 89
ENDIF
IF(TFACT.EQ.6) GOTO 89
GOTO 88
89 IF(FCHR.EQ.0) THEN
  TRY EXTRA CSH AT RANDOM LOCATIONS IN PORE SPACE
  UNTIL IT FINDS AN EMPTY SITE
  RMADD = RAN1(ISEED)
  XCHR = SYSSIZE*RMADD + 1
  RMADD = RAN1(ISEED)
  YCHR = SYSSIZE*RMADD + 1
  RMADD = RAN1(ISEED)
  ZCHR = SYSSIZE*RMADD + 1
  IF(CEMENT(XCHR,YCHR,ZCHR).EQ.0) THEN
    CEMENT(XCHR,YCHR,ZCHR) = 10
    FCHR = 1
  ENDIF
  GOTO 89
ENDIF
RETURN
END
SUBROUTINE MCSHANT(XMCSH,YMCSH,ZMCSH,CSHYET)
INTEGER CEMENT(101,101,101),CYCLENO,NTIMES,NFILL,NPZR
INTEGER SYSSIZE,AGGSIZE,NCEMENT,NCEMDIS
INTEGER*4 ISEED
COMMON /A/CEMENT,CYCLENO,NTIMES,NFILL,NPZR,ISEED,SYSSIZE,
  + AGGSIZE,NCEMENT,NCEMDIS
INTEGER XMCSH,YMCSH,ZMCSH,XS,YS,ZS,MSGEN,CSHYET
REAL RMCSH
C
Subroutine MCSHANT to move a diffusing CSH species and check for reaction

Inputs: x,y, and z coordinates of current location of species

Returns: Flag (CSHYET) indicating if reaction has occurred

GENERATE MOVE

RMCSH = RAN1(ISED)
MSGEN = 6*RMCSH + 1
XS = XMCSH
YS = YMCSH
ZS = ZMCSH
IF(MSGEN.EQ.1) THEN
  XS = XS - 1
  IF(XS.LT.1) XS = SYSSIZE
ENDIF
IF(MSGEN.EQ.2) THEN
  XS = XS + 1
  IF(XS.GT.SYSSIZE) XS = 1
ENDIF
IF(MSGEN.EQ.3) THEN
  YS = YS - 1
  IF(YS.LT.1) YS = SYSSIZE
ENDIF
IF(MSGEN.EQ.4) THEN
  YS = YS + 1
  IF(YS.GT.SYSSIZE) YS = 1
ENDIF
IF(MSGEN.EQ.5) THEN
  ZS = ZS - 1
  IF(ZS.LT.1) ZS = SYSSIZE
ENDIF
IF(MSGEN.EQ.6) THEN
  ZS = ZS + 1
  IF(ZS.GT.SYSSIZE) ZS = 1
ENDIF

If diffusing CSH encounters C3S or solid CSH, it is converted to solid CSH

IF((CEMENT(XS,YS,ZS).EQ.1).OR.(CEMENT(XS,YS,ZS).EQ.4).AND.(NTIMES.EQ.CYCLENO)) THEN
  CEMENT(XMCSH,YMCSH,ZMCSH) = 4
  CSHYET = 0
ENDIF
IF((CEMENT(XS,YS,ZS).EQ.0).AND.(NTIMES.NE.CYCLENO)) THEN
   CEMENT(XS,YS,ZS) = 3
   CEMENT(XMCSH,YMCSH,ZMCSH) = 0
   XMCSH = XS
   YMCSH = YS
   ZMCSH = ZS
ENDIF
RETURN
END

SUBROUTINE MEASURE
INTEGER CEMENT(101,101,101),CYCLENO,NTIMES,NFILL,NPZR
INTEGER SYSSIZE,AGGSIZE,NCEMENT,NCEMDIS
INTEGER*4 ISEE
COMMON /A/CEMENT,CYCLENO,NTIMES,NFILL,NPZR,ISEED,SYSSIZE,
+AGGSIZE,NCEMENT,NCEMDIS
INTEGER NPOR,NCSH,NC3S,NCH,NINERT,NPOZZ,NAGG

C
C Subroutine MEASURE to assess phase fractions in 3-D system
C Called by: MAIN program
C
C Initialize counters for various phases
C
NPOR = 0
NCSH = 0
NCH = 0
NC3S = 0
NINERT = 0
NPOZZ = 0
NAGG = 0
C
C Update counters for all locations in 3-D system
C
DO 14 I = 1,SYSSIZE
DO 14 J = 1,SYSSIZE
DO 14 K = 1,SYSSIZE
   IF(CEMENT(I,J,K).EQ.0) NPOR = NPOR + 1
   IF(CEMENT(I,J,K).EQ.4) NCSH = NCSH + 1
   IF(CEMENT(I,J,K).EQ.1) NC3S = NC3S + 1
   IF(CEMENT(I,J,K).EQ.9) NINERT = NINERT + 1
   IF(CEMENT(I,J,K).EQ.10) NPOZZ = NPOZZ + 1
   IF(CEMENT(I,J,K).EQ.5) NCH = NCH + 1
   IF(CEMENT(I,J,K).EQ.8) NAGG = NAGG + 1
14  CONTINUE
Output Results

WRITE(6,*)'Porosity = ',NPOR
WRITE(6,*)'C3S = ',NC3S
WRITE(6,*)'C-S-H = ',NCSH
WRITE(6,*)'CH = ',NCH
WRITE(6,*)'Inert = ',NINERT
WRITE(6,*)'Pozzolanic = ',NPOZZ
WRITE(6,*)'Aggregate = ',NAGG
RETURN
END

SUBROUTINE MEASAGG
INTEGER CEMENT(101,101,101),CYCLENO,NTIMES,NFILL,NPZR
INTEGER SYSSIZE,AGGSIZE,NCEMENT,NCEMDIS
INTEGER*4 ISEED
COMMON /A/CEMENT,CYCLENO,NTIMES,NFILL,NPZR,ISEED,SYSSIZE,+
AGGSIZE,NCEMENT,NCEMDIS
INTEGER PHASE(11),PTOT,PHREAD,IDIST,ICNT,IYL,IXR,IY,IZ

Subroutine MEASAGG to measure phase fractions as a function of
distance from aggregate surface
Called by: MAIN program

WRITE(6,*)'Distance Porosity C3S C-S-H CH Inert Pozz'

Measure phase fractions as distance from aggregate increases

DO 61 IDIST = 1,((SYSSIZE-AGGSIZE)/2)

Initialize phase counts for this distance

PTOT = 0
DO 62 ICNT = 1,11
62 PHASE(ICNT) = 0

Check all pixels which are this distance from aggregate

IXL = ((SYSSIZE-AGGSIZE + 2)/2)-IDIST
IXR = ((SYSSIZE + AGGSIZE)/2) + IDIST
DO 63 IY = 1,SYSSIZE
DO 63 IZ = 1,SYSSIZE

Check pixel left of aggregate
C

PHID = 1 + CEMENT(IY,IY,IY)
PHASE(PHID) = PHASE(PHID) + 1
PTOT = PTOT + 1

C
Check pixel right of aggregate
C

PHID = 1 + CEMENT(IY,IY,IY)
PHASE(PHID) = PHASE(PHID) + 1
PTOT = PTOT + 1
WRITE(6,*)IDIST,PHASE(1),PHASE(2),PHASE(5),PHASE(6),
+ PHASE(10),PHASE(11)

CONTINUE
RETURN
END

SUBROUTINE CONNECT
INTEGER CEMENT(101,101,101),CYCLENO,NTIMES,NFILL,NPZR
INTEGER SYSSIZE,AGGSIZE,NCEMENT,NCEMDIS
INTEGER*4 ISEED
COMMON /A/CEMENT,CYCLENO,NTIMES,NFILL,NPZR,ISEED,SYSSIZE,
+ AGGSIZE,NCEMENT,NCEMDIS
INTEGER NTOP,NTHROUGH,NCUR,NMATX(8000),NMATY(8000)
INTEGER NMATZ(8000),XCN,YCN,ZCN,X1,Y1,Z1,IGOOD,NNEW
INTEGER NTOT,NNEWX(8000),NNEWY(8000),NNEWZ(8000)
INTEGER NPIX

Subroutine CONNECT to assess connectivity (percolation) of
a single phase

WRITE(6,*)'ENTER PHASE TO ANALYZE 0) PORES 1) C3S'
WRITE(6,*)' 4) CSH 5) CH 9) Inert 10) Pozzolanic'
READ(5,*)NPIX
WRITE(6,*)NPIX

Counters for number of pixels of phase accessible from
top and number which are part of a percolation path

NTOP = 0
NTHROUGH = 0

Percolation is assessed from top (K = 1) to bottom (K = SYSSIZE)

K = 1
Start a burn pattern from all pixels on the top surface

DO 17 I=1,SYSSIZE
DO 17 J=1,SYSSIZE
NCUR=0
NTOT=0
IGOOD=0
IF(CEMENT(I,J,K).EQ.NPIX) THEN

Use a temporary ID of 7 to identify burnt pixels

CEMENT(I,J,K) = 7
NTOT = NTOT + 1
NCUR = NCUR + 1

Store the burn front coordinates in the matrices NMAT* and NNEW*

NMATX(NCUR) = I
NMATY(NCUR) = J
NMATZ(NCUR) = 1
NNEW = 0

Propagate fire from all pixels in the current burn front

DO 27 INEW = 1,NCUR
XCN = NMATX(INEW)
YCN = NMATY(INEW)
ZCN = NMATZ(INEW)

Check for propagation in all six directions

DO 37 JNEW = 1,6
X1 = XCN
Y1 = YCN
Z1 = ZCN
IF(JNEW.EQ.1) X1 = XCN-1
IF(JNEW.EQ.2) X1 = XCN + 1
IF(JNEW.EQ.3) Y1 = YCN-1
IF(JNEW.EQ.4) Y1 = YCN + 1
IF(JNEW.EQ.5) Z1 = ZCN-1
IF(JNEW.EQ.6) Z1 = ZCN + 1

Note that burning is non-periodic
IF((X1.GE.1).AND.(X1.LE.SYSSIZE).AND.(Y1.GE.1).
&AND.(Y1.LE.SYSSIZE).AND.(Z1.GE.1).AND.(Z1.LE.SYSSIZE)) THEN
  IF(CEMENT(X1,Y1,Z1).EQ.NPIX) THEN
    NTOT = NTOT + 1
    CEMENT(X1,Y1,Z1) = 7
    NNEW = NNEW + 1
    NNEWX(NNEW) = X1
    NNEWY(NNEW) = Y1
    NNEWZ(NNEW) = Z1
  ENDIF
ENDIF
ENDIF
CONTINUE
CONTINUE
IF(NNEW.GT.0) THEN
  NCUR = NNEW
ENDIF
CONTINUE
CONTINUE
IF(Z1.EQ.SYSSIZE) THEN
  IGOOD = 1
ENDIF
ENDIF
CONTINUE
CONTINUE
IF(NNEW.GT.0) THEN
  NCUR = NNEW
ENDIF
CONTINUE
DO 47 ICUR = 1,NCUR
  NMATX(ICUR) = NNEWX(ICUR)
  NMATY(ICUR) = NNEWY(ICUR)
  NMATZ(ICUR) = NNEWZ(ICUR)
47 CONTINUE
GOTO 57
ENDIF
CONTINUE
NTOP = NTOP + NTOT
IF(IGOOD.EQ.1) THEN
  NTHROUGH = NTHROUGH + NTOT
ENDIF
CONTINUE
CONTINUE
WRITE(6,*)'Phase ID = ',NPIX
WRITE(6,*)'Number accessible from top = ',NTOP
WRITE(6,*)'Number contained in through pathways = ',NTHROUGH
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINU
Subroutine CONSOLD to assess connectivity (percolation) of total solid phases with exception of aggregate if one is present

Counters for number of pixels of solid accessible from top and number which are part of a percolation path

NTOP = 0
NTHROUGH = 0

Percolation is assessed from top (K = 1) to bottom (K = SYSSIZE)

K = 1

Start a burn pattern from all pixels on the top surface

DO 17 I = 1, SYSSIZE
DO 17 J = 1, SYSSIZE
NCUR = 0
NTOT = 0
IGOOD = 0

Use a temporary ID of 12 + present ID to identify burnt pixels
CEMENT(I,J,K) = CEMENT(I,J,K) + 12
NTOT = NTOT + 1
NCUR = NCUR + 1

Store the burn front coordinates in the matrices NMAT* and NNEW*

NMATX(NCUR) = I
NMATY(NCUR) = J
NMATZ(NCUR) = 1
NNEW = 0

Propagate fire from all pixels in the current burn front

DO 27 INEW = 1, NCUR
   XCN = NMATX(INEW)
   YCN = NMATY(INEW)
   ZCN = NMATZ(INEW)

Check for propagation in all six directions

DO 37 JNEW = 1, 6
   X1 = XCN
   Y1 = YCN
   Z1 = ZCN
   IF(JNEW.EQ.1) X1 = XCN - 1
   IF(JNEW.EQ.2) X1 = XCN + 1
   IF(JNEW.EQ.3) Y1 = YCN - 1
   IF(JNEW.EQ.4) Y1 = YCN + 1
   IF(JNEW.EQ.5) Z1 = ZCN - 1
   IF(JNEW.EQ.6) Z1 = ZCN + 1

Note that burning is non-periodic

   IF((CEMENT(X1,Y1,Z1).LT.12).AND.(CEMENT(X1,Y1,Z1).NE.0) + .AND.(CEMENT(X1,Y1,Z1).NE.8)) THEN
      NTOT = NTOT + 1
      CEMENT(X1,Y1,Z1) = CEMENT(X1,Y1,Z1) + 12
      NNEW = NNEW + 1
      NNEWX(NNEW) = X1
      NNEWY(NNEW) = Y1
      NNEWZ(NNEW) = Z1

56
Check if new burnt pixel is on the bottom face of the system

IF(Z1.EQ.SYSSIZE) THEN
  IGOOD = 1
ENDIF
ENDIF
ENDIF

CONTINUE

IF(NNEW.GT.0) THEN
  NCUR = NNEW
ENDIF

Copy the new burn front locations to matrices NMAT*

DO 47 ICUR = 1,NCUR
   NMATX(ICUR) = NNEWX(ICUR)
   NMATY(ICUR) = NNEWY(ICUR)
   NMATZ(ICUR) = NNEWZ(ICUR)
47 CONTINUE

GOTO 57
ENDIF

NTOP = NTOP + NTOT
IF(IGOOD.EQ.1) THEN
   NTHROUGH = NTHROUGH + NTOT
ENDIF

CONTINUE
WRITE(6,*)'FOR TOTAL SOLIDS '
WRITE(6,*)'Number accessible from top = ',NTOP
WRITE(6,*)'Number contained in through pathways = ',NTHROUGH

Restore all burnt pixels to original phase IDs

DO 67 I = 1,SYSSIZE
  DO 67 J = 1,SYSSIZE
    DO 67 K = 1,SYSSIZE
      IF(CEMENT(I,J,K).GT.12) CEMENT(I,J,K) = CEMENT(I,J,K)-12
 67 CONTINUE
RETURN
END
APPENDIX C
C Listing for HYDRA3D

#include <stdio.h>
#include <math.h>

#define SYSSIZE 100

/* phase identifiers */
define POROSITY 0
define C3S 1
define DIFFCH 2
define DIFFCSH 3
define CSH 4
define CH 5
define SURFID 6
define BURNT 7
define AGG 8
define INERT 9
define POZZ 10

/* number of different classes of spheres allowed */
define NUMSIZES 10

/* definitions for portable random number generator */
define M1 259200
define IA1 7141
define IC1 54773
define RM1 (1.0/M1)
define M2 134456
define IA2 8121
define IC2 28411
define RM2 (1.0/M2)
define M3 243000
define IA3 4561
define IC3 51349

/* 3-D microstructure is stored in 3-D array cement */
static unsigned short int cement [101] [101] [101];
long int nfill, npr, ncement, ncemdis;
int cycleno, ntimes, aggszie, *seed;
/* Program: HYDRA3D.C */
/* Programmer: Dale P. Bentz */
/* Building and Fire Research Laboratory */
/* National Institute of Standards and Technology */
/* Building 226 Room B-348 */
/* Gaithersburg, MD 20899-0001 */
/* (301) 975-5865 FAX (301) 975-4032 */
/* Date: 8/91 */
/* */
/* Purpose: To execute a three-dimensional cement hydration model */
/* */
/* */
/* Portable random number generator routine, ran1, from */
/* Numerical Recipes in C */
/* Press, Flannery, Teukolsky, and Vetterling */
/* Returns floating point random numbers between 0 and 1 */
float ran1(int idum)
{
    int *idum;
    
    static long ix1, ix2, ix3;
    static float r[98];
    float temp;
    static int iff = 0;
    int j;

    if (*idum < 0 || iff == 0) {
        iff = 1;
        ix1 = (IC1-(*idum)) % M1;
        ix1 = (IA1*ix1 + IC1) % M1;
        ix2 = ix1 % M2;
        ix1 = (IA1*ix1 + IC1) % M1;
        ix3 = ix1 % M3;
        for (j = 1; j <= 97; j++) {
            ix1 = (IA1*ix1 + IC1) % M1;
            ix2 = (IA2*ix2 + IC2) % M2;
            r[j] = (ix1 + ix2*RM2)*RM1;
        }
        *idum = 1;
    }
    
    ix1 = (IA1*ix1 + IC1) % M1;
\[ \text{i}x2 = \text{(IA2*ix2 + IC2)} \mod \text{M2}; \]
\[ \text{i}x3 = \text{(IA3*ix3 + IC3)} \mod \text{M3}; \]
\[ j = 1 + (97*\text{ix3})/\text{M3}; \]
if \((j > 97 \; || \; j < 1)\) printf("RAN1: This cannot happen.");
\[ \text{temp} = r[j]; \]
\[ r[j] = (\text{ix1} + \text{ix2*RM2}) * \text{RM1}; \]
return temp;

/* undefine variables in case needed later */
#undef M1
#undef IA1
#undef IC1
#undef RM1
#undef M2
#undef IA2
#undef IC2
#undef RM2
#undef M3
#undef IA3
#undef IC3

/* routine to display an introduction to the user */
void intro()
{
    printf("Welcome to HYDRA3D, a digital-image-based cement \n");
    printf("microstructural model. This program has been \n");
    printf("developed to simulate the microstructural \n");
    printf("development of cement, specifically tricalcium \n");
    printf("silicate, C3S, as it reacts with water. In \n");
    printf("addition to C3S, the user may also add a single \n");
    printf("inert aggregate and/or inert or pozzolanic mineral \n");
    printf("admixture particles to the microstructure. In \n");
    printf("addition to hydration, the user may also assess \n");
    printf("phase fractions (either globally or as a function \n");
    printf("of distance from the aggregate surface) and \n");
    printf("the connectivity of any individual phase or of \n");
    printf("the total solids (neglecting any aggregate) present \n");
    printf("in the system. \n\n");
}

/* Routine to add a flat plate aggregate through the 3-D microstructure */
void addagg();
\{  
int ix, iy, iz;

printf("Enter aggregate thickness (even number of pixels) \n");
scanf("%d", &aggsize);
printf("%d \n", aggsize);

/* Aggregate extends through entire system in y and z directions */
for(ix = ((SYSSIZE - aggsize + 2)/2); ix <= ((SYSSIZE + aggsize)/2); ix++){
  for(iy = 1; iy <= SYSSIZE; iy++){
    for(iz = 1; iz <= SYSSIZE; iz++){
      cement[ix][iy][iz] = AGG;
    }
  }
}

/* routine to check or perform placement of sphere centered */
/* at location xin, yin, zin of radius radd */
/* wflg = 1 check for fit of sphere */
/* wflg = 2 place the sphere */
int chksph(xin, yin, zin, radd, wflg, phasein)
int xin, yin, zin, radd, wflg, phasein;
{
  int sflg, nofits, xp, yp, zp, i, j, k;
  float dist;
  
nofits = 0;  /* Flag indicating if placement is possible */

/* Check all pixels within the digitized sphere volume */
for(i = xin-radd; (i <= xin + radd) && (nofits == 0)); i++){
  for(j = yin-radd; (j <= yin + radd) && (nofits == 0)); j++){
    for(k = zin-radd; (k <= zin + radd) && (nofits == 0)); k++){
      xp = i;
      yp = j;
      zp = k;
      /* use periodic boundary conditions for sphere placement */
      if(xp < 1) {xp = SYSSIZE;}
      if(yp < 1) {yp = SYSSIZE;}
      if(zp < 1) {zp = SYSSIZE;}
      if(xp > SYSSIZE) {xp = SYSSIZE;}
      if(yp > SYSSIZE) {yp = SYSSIZE;}
    }
  }
}
\}
if(zp > SYSSIZE) {zp = SYSSIZE;}
/* Compute distance from center of sphere to this pixel */
dist = sqrt((float)((i-xin)*(i-xin) + (j-yin)*(j-yin) + (k-zin)*(k-zin)));
if((dist-0.5) < (float)radd) {
    if(wflg = 2) {
        cement [xp] [yp] [zp] = phasein;
    }
    /* Update counter of C3S species if necessary */
    if(phasein = C3S) {
        ncement += 1;
    }
    /* Update counter of pozzolanic filler species if necessary */
    if(phasein = POZZ) {
        nfill += 1;
    }
}
    if((wflg = 1)&&(cement [xp] [yp] [zp] != POROSITY)) {
        nofits = 1;
    }
} /* return flag indicating if sphere will fit */
return(nofits);
}
/* routine to place spheres of various sizes and phases at random */
/* locations in 3-D microstructure */
void gsphere(numgen,numeach,sizeeach,pheach) {
    int numgen;
    long int numeach[NUMSIZES];
    int sizeeach[NUMSIZES], pheach[NUMSIZES];
{
    int count, x, y, z, radius, ig, kg, phsph;
    long int jg;
    float rx, ry, rz;

    /* Generate spheres of each size class in turn (largest first) */
    for(ig = 0; ig < numgen; ig++) {
        radius = sizeeach[ig]; /* radius for this class */
phsph = pheach[ig];  /* phase for this class */
/* loop for each sphere in this size class */
for(jg = 1; jg <= numeach[ig]; jg++) {
    do{
        /* generate a random center location for the sphere */
        rx = ran1(seed);
        ry = ran1(seed);
        rz = ran1(seed);
        x = (int)((float)SYSSIZE*rx) + 1;
        y = (int)((float)SYSSIZE*ry) + 1;
        z = (int)((float)SYSSIZE*rz) + 1;
        /* see if the sphere will fit at x, y, z */
        count = chksph(x, y, z, radius, 1, phsph);
    } while(count != 0);
    /* place the sphere at x, y, z */
    count = chksph(x, y, z, radius, 2, phsph);
}
}

/* routine to obtain user input and create a starting microstructure */
void create() {
    int numsize, sphrad[NUMSIZE], sphid[NUMSIZE];
    long int sphnum[NUMSIZE], inval1;
    int isph, inval;

    printf("Enter number of different size spheres to use (maximum is 10) \n");
    scanf("%d", &numsize);
    printf("%d \n", numsize);

    if((numsize > 0) && (numsize < (NUMSIZE + 1))) {
        printf("Enter number, size, and phase ID for each class (largest radius 1st) \n");

        /* Obtain input for each size class of spheres */
        for(isph = 0; isph < numsize; isph++) {
            printf("Enter number of spheres of class %d \n", isph + 1);
            scanf("%ld", &inval1);
            printf("%ld \n", inval1);
            sphnum[isph] = inval1;
        }
    }

    for(jg = 1; jg <= numeach[ig]; jg++) {
        /* generate a random center location for the sphere */
        rx = ran1(seed);
        ry = ran1(seed);
        rz = ran1(seed);
        x = (int)((float)SYSSIZE*rx) + 1;
        y = (int)((float)SYSSIZE*ry) + 1;
        z = (int)((float)SYSSIZE*rz) + 1;
        /* see if the sphere will fit at x, y, z */
        count = chksph(x, y, z, radius, 1, phsph);
    } while(count != 0);
    /* place the sphere at x, y, z */
    count = chksph(x, y, z, radius, 2, phsph);
}

/* routine to obtain user input and create a starting microstructure */
void create() {
    int numsize, sphrad[NUMSIZE], sphid[NUMSIZE];
    long int sphnum[NUMSIZE], inval1;
    int isph, inval;

    printf("Enter number of different size spheres to use (maximum is 10) \n");
    scanf("%d", &numsize);
    printf("%d \n", numsize);

    if((numsize > 0) && (numsize < (NUMSIZE + 1))) {
        printf("Enter number, size, and phase ID for each class (largest radius 1st) \n");

        /* Obtain input for each size class of spheres */
        for(isph = 0; isph < numsize; isph++) {
            printf("Enter number of spheres of class %d \n", isph + 1);
            scanf("%ld", &inval1);
            printf("%ld \n", inval1);
            sphnum[isph] = inval1;
        }
    }

    for(jg = 1; jg <= numeach[ig]; jg++) {
        /* generate a random center location for the sphere */
        rx = ran1(seed);
        ry = ran1(seed);
        rz = ran1(seed);
        x = (int)((float)SYSSIZE*rx) + 1;
        y = (int)((float)SYSSIZE*ry) + 1;
        z = (int)((float)SYSSIZE*rz) + 1;
        /* see if the sphere will fit at x, y, z */
        count = chksph(x, y, z, radius, 1, phsph);
    } while(count != 0);
    /* place the sphere at x, y, z */
    count = chksph(x, y, z, radius, 2, phsph);
}
printf("Enter radius of spheres of class \%d \n",isph + 1);
printf("(Integer \leq 10 please) \n");
scanf("%d",&inval);
printf("%d \n",inval);
sphrad[isph] = inval;

printf("Enter phase to create for spheres of class \%d \n",isph + 1);
printf("(1-C3S, 9- Inert filler, 10- Pozzolanic filler \n");
scanf("%d",&inval);
printf("%d \n",inval);
sphid[isph] = inval;
}
gsphere(numsize,sphnum,sphrad,sphid);
}

/* routine to place one pixel filler particles at random unoccupied locations in 3-D system */
void genfill(ntopl,idtouse)
    long int ntopl; /* Number of filler pixels to place */
    int idtouse;    /* Phase to be assigned to filler */
{
    int pfill,xfill,yfill,zfill;
    long int ifill;
    float rxf,ryf,rzf;

    /* Place each filler particle in turn */
    for(ifill = 1;ifill < ntopl;ifill + + ){
        pfill = 0;
        /* place this filler pixel at a random unoccupied location */
        while (pfill == 0){
            rxf = ran1(seed);
            ryf = ran1(seed);
            rzf = ran1(seed);

            xfill = (int)((float)SYSSIZE*rxf) + 1;
            yfill = (int)((float)SYSSIZE*ryf) + 1;
            zfill = (int)((float)SYSSIZE*rzf) + 1;

            if(cement [xfill] [yfill] [zfill] = = POROSITY){
                pfill = 1;
                cement [xfill] [yfill] [zfill] = idtouse;
            }
        }
    }
}
subroutine to obtain user input as to filler to be added */
void filler()
{
    long int nadd;
    int fillid;

    printf("Enter number of filler particles to add \n");
    scanf("%ld",&nadd);
    printf("%ld \n",nadd);
    printf("Enter filled id to use (9- inert, 10- pozollanic) \n");
    scanf("%d",&fillid);
    printf("%d \n",fillid);

    /* If phase ID is valid, call routine to place the filler */
    if((fillid = = INERT)| (fillid = = POZZ)){
        genfill(nadd,fillid);
        if(fillid = = POZZ){
            nfill += nadd;
        }
    }
}

/* routine to perform the dissolution process for a hydration cycle */
/* Locations of dissolved species are returned in arrays disx,disy, and */
/* disz */
long int dissolv(disx,disy,disz)
unsigned short int disx [200000], disy [200000], disz [200000];
{
    long int numdis,nchadd,ncshadd,numorg,i,j,k;
    int xc,yc,zc,irnd,nedge;
    float rdis,rdx,rdy,rdz;

    /* counter for number of species dissolved */
    numdis = 0;

    /* Pass 1: Highlight all edge points */
    /* (C3S with at least one neighbor porosity) */
    for(i = 1;i < = SYSSIZE;i + +){
        for(j = 1;j < = SYSSIZE;j + +){

65
for(k = 1; k <= SYSSIZE; k++)
        /* if pixel is cement, see if it is on an edge */
        if(cement[i][j][k] == C3S)
            nedge = 1;
        xc = i - 1;
        /* periodic boundaries once more */
        if(xc < 1) {xc = SYSSIZE;}
        nedge* = cement[xc][j][k];
        xc = i + 1;
        if(xc > SYSSIZE) {xc = 1;}
        nedge* = cement[xc][j][k];
        yc = j - 1;
        if(yc < 1) {yc = SYSSIZE;}
        nedge* = cement[i][yc][k];
        yc = j + 1;
        if(yc > SYSSIZE) {yc = 1;}
        nedge* = cement[i][yc][k];
        zc = k - 1;
        if(zc < 1) {zc = SYSSIZE;}
        nedge* = cement[i][j][zc];
        zc = k + 1;
        if(zc > SYSSIZE) {zc = 1;}
        nedge* = cement[i][j][zc];
        /* if nedge is zero, at least one neighbor was porosity */
        if(nedge == 0) {cement[i][j][k] = SURFID;}
    }

    /* Pass 2: Dissolve all highlighted edge points at random */
    /* by allowing each to attempt a one step random walk */

    for(i = 1; i <= SYSSIZE; i++)
        for(j = 1; j <= SYSSIZE; j++)
            for(k = 1; k <= SYSSIZE; k++)
                if(cement[i][j][k] == SURFID)
                    /* Choose a random direction for the step */
                    rdis = ran1(seed);
                    irnd = (int)(6.*rdis) + 1;
                    xc = i;
                    yc = j;
                    zc = k;
                    /* dissolution is attempted by performing */
/* a one-step random walk */
switch (irnd) {
    case 1:
        xc = 1;
        if(xc < 1) {xc = SYSSIZE;}
        break;
    case 2:
        xc += 1;
        if(xc > SYSSIZE) {xc = 1;}
        break;
    case 3:
        yc = 1;
        if(yc < 1) {yc = SYSSIZE;}
        break;
    case 4:
        yc += 1;
        if(yc > SYSSIZE) {yc = 1;}
        break;
    case 5:
        zc = 1;
        if(zc < 1) {zc = SYSSIZE;}
        break;
    case 6:
        zc += 1;
        if(zc > SYSSIZE) {zc = 1;}
        break;
    default:
        break;
}

/* if step is not into porosity, remain as solid C3S */
if(cement [xc] [yc] [zc] != POROSITY){
    cement [i] [j] [k] = C3S;
}

/* if step is into porosity, perform the dissolution */
/* and store location of dissolved species */
if(cement [xc] [yc] [zc] == POROSITY){
    cement [xc] [yc] [zc] = DIFFCSH;
    cement [i] [j] [k] = POROSITY;
    numdis += 1;
    disx[numdis] = xc;
    disy[numdis] = yc;
    disz[numdis] = zc;
}
printf("Original number species dissolved = \%ld \n", numdis);
numorg = numdis;
cemdis += numdis;

/* Add in extra diffusing species */
/* One dissolved unit of C3S should produce 1.7 units of C-S-H */
/* and 0.61 units of CH */
/* Expansion factors of 0.7 and 0.61 are taken from work of */
ncshadd = (int)((float)numdis*0.7);
ncchadd = (int)((float)numdis*0.61);
umdis += (nchadd + ncshadd);
if(numdis > 200000){
    printf("Too many dissolved species generated \n");
    printf("Aborting run \n");
    exit(1);
}

for(i = 1; i <= (nchadd + ncshadd); i += 1){
    nedge = 0; /* flag indicating successful placement */
    do{
    /* extra diffusing species are added at totally */
    /* random unoccupied locations in system */
rndx = ran1(seed);
rndy = ran1(seed);
rndz = ran1(seed);
xc = (int)((float)SYSSIZE*rndx) + 1;
yc = (int)((float)SYSSIZE*rndy) + 1;
zc = (int)((float)SYSSIZE*rndz) + 1;
    if(cement [xc] [yc] [zc] = = POROSITY){
        nedge = 1;
        disx[numorg + i] = xc;
        disy[numorg + i] = yc;
        disz[numorg + i] = zc;
        if(i <= nchadd){
            cement [xc] [yc] [zc] = DIFFCH;
        }
        if(i > nchadd){
    }
cement [xc] [yc] [zc] = DIFFCSH;
}
}
} while (nedge = = 0);

/* return number of generated diffusing species */
return(numdis);
}

/* routine to add in extra pozzolanic CSH when diffusing CH */
/* species reacts with pozzolanic filler */
void addext(xinn,yinn,zinn)
    int xinn,yinn,zinn; /* location of reaction site */
{
    int xchr,ychr,zchr,fchr,mgen1;
    long int tfact; /* indicates when all 6 neighbors have been tested */
    float rmadd;

    /* Add in extra pozzolanic CSH to account for volume balance */
    /* Random tries at neighboring locations (6) until all tried */
    fchr = 0; /* Flag indicating successful placement */
    tfact = 1;
    while((tfact! = 30030) &&(fchr = = 0)){
    /* Generate a random direction to test */
    rmadd = ran1(seed);
    mgen1 = (int)(6.*rmadd) + 1;
    xchr = xinn;
    ychr = yinn;
    zchr = zinn;

    switch(mgen1){
        case 1:
            xchr - = 1;
            /* periodic boundaries once again */
            if(xchr < 1) {xchr = SYSSIZE;}
            if((tfact%2)! = 0){tfact* = 2;}
            break;
        case 2:
            xchr + = 1;
            if(xchr > SYSSIZE) {xchr = 1;}
            if((tfact%3)! = 0){tfact* = 3;}
            break;
        case 3:
ychr- = 1;
if(ychr < 1) {ychr = SYSSIZE;}
if((tfact%5)! = 0){tfact* = 5;}
break;
case 4:
ychr + = 1;
if(ychr > SYSSIZE) {ychr = 1;}
if((tfact%7)! = 0){tfact* = 7;}
break;
case 5:
zchr- = 1;
if(zchr < 1) {zchr = SYSSIZE;}
if((tfact%11)! = 0){tfact* = 11;}
break;
case 6:
zchr + = 1;
if(zchr > SYSSIZE) {zchr = 1;}
if((tfact%13)! = 0){tfact* = 13;}
break;
default:
break;
}
if(cement [xchr] [ychr] [zchr] = = POROSITY){
cement [xchr] [ychr] [zchr] = POZZ;
fchr = 1;
}

/* If initial efforts unsuccessful, then */
/* Add extra CSH at random location in pore space */
while(fchr = = 0){
    rmadd = ran1(seed);
    xchr = (int)((float)SYSSIZE*rmadd) + 1;
    rmadd = ran1(seed);
    ychr = (int)((float)SYSSIZE*rmadd) + 1;
    rmadd = ran1(seed);
    zchr = (int)((float)SYSSIZE*rmadd) + 1;
    if(cement [xchr] [ychr] [zchr] = = POROSITY){
        cement [xchr] [ychr] [zchr] = POZZ;
        fchr = 1;
    }
}
/* routine to move/react a diffusing CH species */
int mvchant(xmch1,ymch1,zmch1,pnuc)
    int *xmch1,*ymch1,*zmch1; /* location of CH species to move */
    float pnuc;
{
    int xm,ym,zm,mgen,chyet,xmch,ymch,zmch;
    float pgen,rm;

    zmch = (*zmch1);
    xmch = (*xmch1);
    ymch = (*ymch1);
    chyet = 1;
    /* first allow for random nucleation at current location */
    /* if last step in this cycle, convert to solid CH */
    pgen = ran1(seed);
    if((pnuc > pgen) || (cycleno == ntimes)){
        cement [xmch] [ymch] [zmch] = CH;
        chyet = 0;
    }
    /* If no nucleation, allow for diffusion */
    if(chyet == 1){

    /* Generate a move */
    rm = ran1(seed);
    mgen = (int)(6.*rm) + 1;
    xm = xmch;
    ym = ymch;
    zm = zmch;
    switch (mgen){
        case 1:
            xm- = 1;
            if(xm<1) {xm = SYSSIZE;}
            break;
        case 2:
            xm+ = 1;
            if(xm>SYSSIZE) {xm = 1;}
            break;
        case 3:
            ym- = 1;
            if(ym<1) {ym = SYSSIZE;}
            break;
        case 4:

    }*/
ym += 1;
if(ym > SYSSIZE) {ym = 1;}
break;

case 5:
    zm -= 1;
    if(zm < 1) {zm = SYSSIZE;}
    break;

case 6:
    zm += 1;
    if(zm > SYSSIZE) {zm = 1;}
    break;

default:
    break;

} /* check for growth of a solid CH crystal */
if(cement [xm] [ym] [zm] == CH){
    cement [xmch] [ymch] [zmch] = CH;
    chyet = 0;
}

/* check for pozzolanic reaction of CH */ /* Each volume unit of pozzolanic filler can */ /* react with 2.08 volume units of CH to produce */ /* 4.6 volume units of pozzolanic C-S-H */ /* This assumes the pozzolanic filler is pure silica */ /* with a molar volume of 27 cm^3/mole */
if((cement [xm] [ym] [zm] == POZZ) &&
   (npr <= (int)((float)nfill*2.08))){
    cement [xmch] [ymch] [zmch] = POZZ;
    chyet = 0;
    npr += 1;
    rm = ran1(seed);
    /* Expansion probability = (4.6-1)/2.08-1 = 0.73 */
    /* where the 1 taken away from the 4.6 represents */
    /* the original silica fume particle pixel */
    if(rm <= 0.73){
      addext(xmch,ymch,zmch);
    }
}

if(cement [xm] [ym] [zm] == POROSITY){
/* Diffusion by moving species to new location */
    cement [xm] [ym] [zm] = DIFFCH;
    cement [xmch] [ymch] [zmch] = POROSITY;
    xmc = xm;
    ymch = ym;
zmch = zm;
}
*zmch1 = zmch;
*ymch1 = ymch;
*xmch1 = xmch;

/* return flag indicating if diffusing CH species reacted */
return(chyet);
}

/* routine to move/react a diffusing CSH species */
int mcshant(xmcsh1,ymcsh1, zmcsh1)
    int *xmcsh1,*ymcsh1,*zmcsh1;
{
    int xmcsh, ymcsh, zmcsh, xs, ys, zs, msgen, cshyet;
    float rmcsh;
    cshyet = 1;
    /* choose a random direction for the move */
    rmcsh = ran1(seed);
    msgen = (int)(6. * rmcsh) + 1;
    xmcsh = (*xmcsh1);
    ymcsh = (*ymcsh1);
    zmcsh = (*zmcsh1);
    xs = xmcsh;
    ys = ymcsh;
    zs = zmcsh;
    switch(msgen) {
        case 1:
            xs = xs - 1;
            if(xs < 1) {xs = SYSSIZE;}
            break;
        case 2:
            xs += 1;
            if(xs > SYSSIZE) {xs = 1;}
            break;
        case 3:
            ys -= 1;
            if(ys < 1) {ys = SYSSIZE;}
            break;
        case 4:  

ys += 1;
if(ys > SYSSIZE) {ys = 1;}
break;
case 5:
    zs = 1;
    if(zs < 1) {zs = SYSSIZE;}
    break;
case 6:
    zs += 1;
    if(zs > SYSSIZE) {zs = 1;}
    break;
default:
    break;
}

/* check for reaction at solid C3S or C-S-H surface */
/* If last diffusion step this cycle, convert to solid C-S-H */
if((cement [xs] [ys] [zs] = = C3S) | (cement [xs] [ys] [zs] = = CSH)
   | (ntimes = = cycleno))
    cement [xmcsh] [ymcsh] [zmcsh] = CSH;
cshyet = 0;
}
if((cshyet! = 0)&&(cement [xs] [ys] [zs] = = POROSITY))
/* Diffusion by moving species to new location */
cement [xs] [ys] [zs] = DIFFCSH;
cement [xmcsh] [ymcsh] [zmcsh] = POROSITY;
xmcsh = xs;
ymcsh = ys;
zmcsh = zs;

*xmcsh1 = xmcsh;
*ymcsh1 = ymcsh;
*zmcsh1 = zmcsh;

/* return flag indicating if diffusing CSH species has reacted */
return(cshyet);
}

/* routine to control the hydration process */
void hydrate()
{
    int niter,stopch;
    static unsigned short int antsz[200000],antsx[200000],antsy[200000];
int xant, yant, zant, alive;
int anttype, i, j, k;
long int numants, numleft, nbleft, nblue;
float nucscale, nucprob;
float beterm, bbias, bnew, almax, alpha;

do{
    printf("Do you wish to specify 0) max. # of cycles or 1) max.
            degree of hydration \n");
    scanf("%d", &stopch);
} while ((stopch == 1) && (stopch == 0));
printf("%d \n", stopch);

if(stopch == 0){
    printf("Enter number of hydration cycles (iterations) to perform \n");
    scanf("%d", &niter);
    printf("%d \n", niter);
    almax = 1.0;
}
if(stopch == 1){
    printf("Enter maximum degree of hydration \n");
    scanf("%f", &almax);
    printf("%f \n", almax);
    niter = 5000;
}
printf("Enter maximum number of diffusion steps per cycle \n");
scanf("%d", &ntimes);
printf("%d \n", ntimes);

printf("Enter maximum probability for CH nucleation \n");
scanf("%f", &nucprob);
printf("%f \n", nucprob);
printf("Enter exponential scale factor for CH nucleation \n");
scanf("%f", &nucscale);
printf("%f \n", nucscale);

alpha = (float)ncemdis/(float)ncement;
for(i = 1; (i <= niter) && (alpha < almax); i++){
    numants = 0;
    /* perform the dissolution step */
    numants = dissolv(antsx, antsy, antsz);
    numleft = numants; /* Number of diffusing species remaining */
    alpha = (float)ncemdis/(float)ncement;
printf("Number dissolved = %ld \n",numants);
nblue = (int)(61.*(float)numants)/231.0;

/* diffuse until all species have reacted or maximum number of */
/* steps is exceeded */
for(j = 1; (j <= ntimes) && (numleft != 0); j + + ){

cycleno = j;
numleft = 0;
nleft = 0;
bnew = (float)nblue;
bnew/ = nuscale;
beterm = exp(-bnew);
/* determine new probability of CH nucleation */
bbias = nucprob*(1.-beterm);

/* Move each remaining diffusing species in turn */
for(k = 1; k <= numants; k + + ){

/* get the current location and type */
/* of this diffusing species */
xant = antsx[k];
yant = antsy[k];
zant = antsz[k];
anttype = cement [xant] [yant] [zant];
alive = 1; /* Flag indicating reaction */

/* Note that addresses are passed to routines mvchant and */
/* mcshant so that new locations of diffusing species can */
/* be updated (returned) */
if(anttype == DIFFCH){
    alive = mvchant(&xant,&yant,&zant,bbias);
}
if(anttype == DIFFCSH){
    alive = mcshant(&xant,&yant,&zant);
}
if(alive == 1){
    numleft += 1;
    /* store the new location of the */
    /* diffusing species */
    /* Note that we use only one array here, because */
    /* the number of diffusing species remaining after */
    /* a diffusion step is always less than the number */
    /* present at the start of the step */

76
antsx[numleft] = xant;
antsy[numleft] = yant;
antsz[numleft] = zant;

if(anttype == DIFFCH){
    nbleft += 1;
}

/* Update number of diffusing species still in system */
numants = numleft;
nblue = nbleft;
}

} /* routine to assess global phase fractions present in 3-D system */
void measure()
{
    long int npor,ncsh,nc3s,nch,ninert,npozz,nagg;
    int i,j,k;

    /* counters for the various phase fractions */
    npor = 0;
    ncsh = 0;
    nch = 0;
    nc3s = 0;
    npozz = 0;
    ninert = 0;
    nagg = 0;

    /* Check all pixels in 3-D microstructure */
    for(i = 1; i <= SYSSIZE; i++){
        for(j = 1; j <= SYSSIZE; j++){
            for(k = 1; k <= SYSSIZE; k++){

                if(cement [i][j][k] == POROSITY) {npor += 1;}
                if(cement [i][j][k] == CSH) {ncsh += 1;}
                if(cement [i][j][k] == C3S) {nc3s += 1;}
                if(cement [i][j][k] == INERT) {ninert += 1;}
                if(cement [i][j][k] == POZZ) {npozz += 1;}
                if(cement [i][j][k] == CH) {nch += 1;}
                if(cement [i][j][k] == AGG) {nagg += 1;}

            }
        }
    }

}
/* Output results */
    printf("Porosity = %ld \n", npor);
    printf("C3S = %ld \n", nc3s);
    printf("C-S-H = %ld \n", ncsh);
    printf("CH = %ld \n", nch);
    printf("Inert = %ld \n", ninert);
    printf("Pozzolanic = %ld \n", npozz);
    printf("Aggregate = %ld \n", nagg);
}

/* Routine to measure phase fractions as a function of distance from */
/* aggregate surface */
void measagg()
{
    int phase[11], ptot;
    int icnt, ix, iy, iz, phid, idist;

    printf("Distance Porosity C3S C-S-H CH Inert Pozzolanic \n");

    /* Increase distance from aggregate in increments of one */
    for(idist = 1; idist <= (SYSSIZE-aggsize)/2; idist + +){

        /* Initialize phase counts for this distance */
        for(icnt = 0; icnt < 11; icnt + +){
            phase[icnt] = 0;
        }
        ptot = 0;

        /* Check all pixels which are this distance from aggregate surface */
        for(iy = 1; iy <= SYSSIZE; iy + +){
            for(iz = 1; iz <= SYSSIZE; iz + +){
                /* Pixel left of aggregate surface */
                ix = ((SYSSIZE-aggsize + 2)/2)-idist;
                phid = cement [ix] [iy] [iz];
                ptot + = 1;
                phase[phid] + = 1;

                /* Pixel right of aggregate surface */
                ix = ((SYSSIZE + aggsize)/2) + idist;

                /* Check all pixels which are this distance from aggregate surface */
                for(iy = 1; iy <= SYSSIZE; iy + +){
                    for(iz = 1; iz <= SYSSIZE; iz + +){
                        /* Pixel left of aggregate surface */
                        ix = ((SYSSIZE-aggsize + 2)/2)-idist;
                        phid = cement [ix] [iy] [iz];
                        ptot + = 1;
                        phase[phid] + = 1;

                        /* Pixel right of aggregate surface */
                        ix = ((SYSSIZE + aggsize)/2) + idist;

                        /* Check all pixels which are this distance from aggregate surface */
                        for(iy = 1; iy <= SYSSIZE; iy + +){
                            for(iz = 1; iz <= SYSSIZE; iz + +){
                                /* Pixel left of aggregate surface */
                                ix = ((SYSSIZE-aggsize + 2)/2)-idist;
                                phid = cement [ix] [iy] [iz];
                                ptot + = 1;
                                phase[phid] + = 1;

                                /* Pixel right of aggregate surface */
                                ix = ((SYSSIZE + aggsize)/2) + idist;

                                /* Check all pixels which are this distance from aggregate surface */
                                for(iy = 1; iy <= SYSSIZE; iy + +){
                                    for(iz = 1; iz <= SYSSIZE; iz + +){
                                        /* Pixel left of aggregate surface */
                                        ix = ((SYSSIZE-aggsize + 2)/2)-idist;
                                        phid = cement [ix] [iy] [iz];
                                        ptot + = 1;
                                        phase[phid] + = 1;

                                        /* Pixel right of aggregate surface */
                                        ix = ((SYSSIZE + aggsize)/2) + idist;

                                        /* Check all pixels which are this distance from aggregate surface */
                                        for(iy = 1; iy <= SYSSIZE; iy + +){
                                            for(iz = 1; iz <= SYSSIZE; iz + +){
                                                /* Pixel left of aggregate surface */
                                                ix = ((SYSSIZE-aggsize + 2)/2)-idist;
                                                phid = cement [ix] [iy] [iz];
                                                ptot + = 1;
                                                phase[phid] + = 1;

                                                /* Pixel right of aggregate surface */
                                                ix = ((SYSSIZE + aggsize)/2) + idist;

                                                /* Check all pixels which are this distance from aggregate surface */
                                                for(iy = 1; iy <= SYSSIZE; iy + +){
                                                    for(iz = 1; iz <= SYSSIZE; iz + +){
                                                        /* Pixel left of aggregate surface */
                                                        ix = ((SYSSIZE-aggsize + 2)/2)-idist;
                                                        phid = cement [ix] [iy] [iz];
                                                        ptot + = 1;
                                                        phase[phid] + = 1;

                                                        /* Pixel right of aggregate surface */
                                                        ix = ((SYSSIZE + aggsize)/2) + idist;

                                                        /* Check all pixels which are this distance from aggregate surface */
                                                        for(iy = 1; iy <= SYSSIZE; iy + +){
                                                            for(iz = 1; iz <= SYSSIZE; iz + +){
                                                                /* Pixel left of aggregate surface */
                                                                ix = ((SYSSIZE-aggsize + 2)/2)-idist;
                                                                phid = cement [ix] [iy] [iz];
                                                                ptot + = 1;
                                                                phase[phid] + = 1;

                                                                /* Pixel right of aggregate surface */
                                                                ix = ((SYSSIZE + aggsize)/2) + idist;

                                                                /* Check all pixels which are this distance from aggregate surface */
                                                                for(iy = 1; iy <= SYSSIZE; iy + +){
                                                                    for(iz = 1; iz <= SYSSIZE; iz + +){
                                                                        /* Pixel left of aggregate surface */
                                                                        ix = ((SYSSIZE-aggsize + 2)/2)-idist;
                                                                        phid = cement [ix] [iy] [iz];
                                                                        ptot + = 1;
                                                                        phase[phid] + = 1;

                                                                        /* Pixel right of aggregate surface */
                                                                        ix = ((SYSSIZE + aggsize)/2) + idist;

                                                                        /* Check all pixels which are this distance from aggregate surface */
                                                                        for(iy = 1; iy <= SYSSIZE; iy + +){
                                                                            for(iz = 1; iz <= SYSSIZE; iz + +){
                                                                                /* Pixel left of aggregate surface */
                                                                                ix = ((SYSSIZE-aggsize + 2)/2)-idist;
                                                                                phid = cement [ix] [iy] [iz];
                                                                                ptot + = 1;
                                                                                phase[phid] + = 1;

                                                                                /* Pixel right of aggregate surface */
                                                                                ix = ((SYSSIZE + aggsize)/2) + idist;

                                                                                /* Check all pixels which are this distance from aggregate surface */
                                                                                for(iy = 1; iy <= SYSSIZE; iy + +){
                                                                                    for(iz = 1; iz <= SYSSIZE; iz + +){
                                                                                        /* Pixel left of aggregate surface */
                                                                                        ix = ((SYSSIZE-aggsize + 2)/2)-idist;
                                                                                        phid = cement [ix] [iy] [iz];
                                                                                        ptot + = 1;
                                                                                        phase[phid] + = 1;

                                                                                        /* Pixel right of aggregate surface */
                                                                                        ix = ((SYSSIZE + aggsize)/2) + idist;

                                                                                        /* Check all pixels which are this distance from aggregate surface */
                                                                                        for(iy = 1; iy <= SYSSIZE; iy + +){
                                                                                            for(iz = 1; iz <= SYSSIZE; iz + +){
                                                                                                /* Pixel left of aggregate surface */
                                                                                                ix = ((SYSSIZE-aggsize + 2)/2)-idist;
                                                                                                phid = cement [ix] [iy] [iz];
                                                                                                ptot + = 1;
                                                                                                phase[phid] + = 1;

                                                                                                /* Pixel right of aggregate surface */
                                                                                                ix = ((SYSSIZE + aggsize)/2) + idist;

                                                                                                /* Check all pixels which are this distance from aggregate surface */
                                                                                                for(iy = 1; iy <= SYSSIZE; iy + +){
                                                                                                    for(iz = 1; iz <= SYSSIZE; iz + +){
                                                                                                        /* Pixel left of aggregate surface */
                                                                                                        ix = ((SYSSIZE-aggsize + 2)/2)-idist;
                                                                                                        phid = cement [ix] [iy] [iz];
                                                                                                        ptot + = 1;
                                                                                                        phase[phid] + = 1;

                                                                                                        /* Pixel right of aggregate surface */
                                                                                                        ix = ((SYSSIZE + aggsize)/2) + idist;

                                                                                                        /* Check all pixels which are this distance from aggregate surface */
                                                                                                        for(iy = 1; iy <= SYSSIZE; iy + +){
                                                                                                            for(iz = 1; iz <= SYSSIZE; iz + +){
                                                                                                                /* Pixel left of aggregate surface */
                                                                                                                ix = ((SYSSIZE-aggsize + 2)/2)-idist;
                                                                                                                phid = cement [ix] [iy] [iz];
                                                                                                                ptot + = 1;
                                                                                                                phase[phid] + = 1;

                                                                                                                /* Pixel right of aggregate surface */
                                                                                                                ix = ((SYSSIZE + aggsize)/2) + idist;

                                                } /* end for iy */
                                            } /* end for iz */
                        } /* end for icnt */
                    } /* end for iy */
                } /* end for iz */
            } /* end for icnt */
        } /* end for iy */
    } /* end for idist */
} /* end void measagg() */
phid = cement [ix] [iy] [iz];
ptot += 1;
phase[phid] += 1;

/* Output results for this distance from surface */
    printf("%d %d %d %d %d %d %d \n", idist,phase[0],phase[1],
    phase[4],phase[5],phase[9],phase[10]);
}

/* routine to assess the connectivity (percolation) of a single phase */
/* Two matrices are used here: one to store the recently burnt locations */
    /* the other to store the newly found burnt locations */
void connect()
{
    long int ntop,nthrough,ncur,nnew,ntot;
    int i,inew,j,k,nmatx[9000],nmaty[9000],nmatz[9000];
    int xcn,ycn,zcn,npix,x1,y1,z1,igood,nnewx[9000],nnewy[9000],nnewz[9000];
    int jnew,icur;

    printf("Enter phase to analyze 0) pores 1) C3S 4) CSH 5) CH \n");
    printf(" 9) Inert 10) Pozzolanic \n");
    scanf("%d",&npix);
    printf("%d \n",npix);

    /* counters for number of pixels of phase accessible from top surface */
    /* and number which are part of a percolated pathway */
    ntop = 0;
    nthrough = 0;

    /* percolation is assessed from top to bottom only */
    /* and burning algorithm is nonperiodic in x and y directions */
    k = 1;

    for(i = 1;i <= SYSSIZE;i + +){
    for(j = 1;j <= SYSSIZE;j + +){

        ncur = 0;
        ntot = 0;
        igood = 0;     /* Indicates if bottom has been reached */
        if(cement [i] [j] [k] == npix){
/ * Start a burn front */
cement [i] [j] [k] = BURNT;
ntot + = 1;
ncur + = 1;
/* burn front is stored in matrices nmat* */
/* and nnew* */
ntot + = 1;
ncur + = 1;
/* Burn as long as new (fuel) pixels are found */
do{
    nnew = 0;
    for(inew = 1; inew < = ncur; inew + + ){
        xcn = nmatx[inew];
ycn = nmaty[inew];
zcn = nmatz[inew];

    /* Check all six neighbors */
    for(jnew = 1; jnew < = 6; jnew + + ){
        x1 = xcn;
y1 = ycn;
z1 = zcn;
        if(jnew = = 1){x1 - = 1;}
        if(jnew = = 2){x1 + = 1;}
        if(jnew = = 3){y1 - = 1;}
        if(jnew = = 4){y1 + = 1;}
        if(jnew = = 5){z1 - = 1;}
        if(jnew = = 6){z1 + = 1;}

    /* Nonperiodic so be sure to remain in the 3-D box */
    if((x1 > = 1) && (x1 < = SYSSIZE) && (y1 > = 1) && (y1 < = SYSSIZE) && (z1 > = 1) && (z1 < = SYSSIZE)) {
        if(cement [x1] [y1] [z1] = = npix) {
            ntot + = 1;
cement [x1] [y1] [z1] = BURNT;
nnew + = 1;
            if(nnew > = 9000) {
                printf("error in size of nnew \n");
            }
nnewx[nnew] = x1;
nnewy[nnew] = y1;
nnewz[nnew] = z1;
/* See if bottom of system has been reached */
*/
if(z1 == SYSSIZE){igood = 1;}
}
}
}
if(nnew > 0){
    ncur = nnew;
    /* update the burn front matrices */
    for(icur = 1; icur <= ncur; icur++){
        nmatx[icur] = nnewx[icur];
        nmaty[icur] = nnewy[icur];
        nmatz[icur] = nnewz[icur];
    }
}
} while (nnew > 0);

ntop += ntot;
if(igood == 1){
    nthrough += ntot;
}
}
}

printf("Phase ID = %d \n", npix);
printf("Number accessible from top = %ld \n", ntop);
printf("Number contained in through pathways = %ld \n", nthrough);

/* return the burnt sites to their original phase values */
for(i = 1; i <= SYSSIZE; i++){
    for(j = 1; j <= SYSSIZE; j++){
        for(k = 1; k <= SYSSIZE; k++){
            if(cement[i][j][k] == BURNT){
                cement[i][j][k] = npix;
            }
        }
    }
}

/* routine to assess the connectivity (percolation) of total solids */
/* excluding aggregate when one is present in the system */

81
void consold()
{
    long int ntop,nthrough,ncur,nnew,ntot;
    int i,inew,j,k,nmatx[9000],nmaty[9000],nmatz[9000];
    int xcn,ycn,zcn,x1,y1,z1,igood,nnewx[9000],nnewy[9000],nnewz[9000];
    int jnew,icur;

    /* counters for number of pixels of solids accessible from top surface */
    /* and number which are part of a percolated pathway */
    ntop = 0;
    nthrough = 0;

    /* percolation is assessed from top to bottom only */
    /* and burning algorithm is nonperiodic in x and y directions */
    k = 1;

    for(i = 1;i <= SYSSIZE;i ++){
        for(j = 1;j <= SYSSIZE;j ++){
            ncur = 0;
            ntot = 0;
            igood = 0;  /* Indicates if bottom has been reached */
            if((cement[i][j][k]! = POROSITY) && (cement[i][j][k]<12) && (cement[i][j][k]!=AGG)){
                /* burnt site has value 12 greater than original phase ID */
                cement[i][j][k] += 12;
                ntot += 1;
                ncur += 1;
                /* burn front is stored in matrices nmat* */
                /* and nnew* */
                nmatx[ncur] = i;
                nmaty[ncur] = j;
                nmatz[ncur] = 1;

                /* Burn as long as new (fuel) pixels are found */
                do{
                    nnew = 0;
                    for(inew = 1;inew <= ncur;inew ++){
                        xcn = nmatx[inew];
                        ycn = nmaty[inew];
                        zcn = nmatz[inew];

                        /* Check all six neighbors */
                    }
                    jnew = 1;
                    jnew += 6;
                    jnew ++;
                }
            }else{
                /* No burnt site found */
            }
        }
    }
}
\[ x1 = xcn; \]
\[ y1 = ycn; \]
\[ z1 = zcn; \]
\[ \text{if}\{jnew = 1\}\{x1 - = 1;\} \]
\[ \text{if}\{jnew = 2\}\{x1 + = 1;\} \]
\[ \text{if}\{jnew = 3\}\{y1 - = 1;\} \]
\[ \text{if}\{jnew = 4\}\{y1 + = 1;\} \]
\[ \text{if}\{jnew = 5\}\{z1 - = 1;\} \]
\[ \text{if}\{jnew = 6\}\{z1 + = 1;\} \]

/* Nonperiodic so be sure to remain in original 3-D box */

\[ \text{if}\{(x1 > = 1)\&\&(x1 <= SYSSIZE)\&\&(y1 > = 1)\&\&(y1 <= SYSSIZE)\&\&(z1 > = 1)\&\&(z1 <= SYSSIZE)\}\{ \]
\[ \text{if}\{\text{(cement [x1] [y1] [z1]!=POROSITY)\&\&(cement [x1] [y1] [z1]<12) \}
\[ \&\&(cement [x1] [y1] [z1]!=AGG)\}\{ \]
\[ \text{n}\text{tot} + = 1; \]
\[ \text{cement [x1] [y1] [z1]} + = 12; \]
\[ \text{n}\text{new} + = 1; \]
\[ \text{if}\{\text{n}\text{new} > = 9000\}\{ \]
\[ \text{printf}("\text{error in size of n}\text{new }\n"); \]
\[ \} \]
\[ \text{n}\text{new}\text{x}[\text{n}\text{new}] = x1; \]
\[ \text{n}\text{new}\text{y}[\text{n}\text{new}] = y1; \]
\[ \text{n}\text{new}\text{z}[\text{n}\text{new}] = z1; \]

/* See if bottom has been reached */
\[ \text{if}(z1 = \text{SYSSIZE})\{\text{igood} = 1;\} \]
\[ \}
\[ \}
\]
\[ \}
\]
\[ \}
\]
\[ \}
\]
\[ \} \]
\]while (nnew>0);

\[ \text{n}\text{top} + = \text{n}\text{tot}; \]
\[ \text{if}\{\text{igood} = 1\}\{ \]
\[ \text{n}\text{through} + = \text{n}\text{tot}; \]
printf("For total solids \n");
printf("Number accessible from top = %ld \n", ntop);
printf("Number contained in through pathways = %ld \n", nthrough);

/* return the burnt sites to their original phase values */
for (i = 1; i <= SYSSIZE; i++) {
    for (j = 1; j <= SYSSIZE; j++) {
        for (k = 1; k <= SYSSIZE; k++) {

            if (cement[i][j][k] > 12) {
                cement[i][j][k] = 12;
            }
        }
    }
}

main() {
    int userc;  /* User choice from menu */
    int nseed, ig, jg, kg;

    /* Display an introduction to the user */
    intro();

    printf("Enter random number seed value \n");
    scanf("%d", &nseed);
    printf("%d \n", nseed);
    seed = (&nseed);

    /* Initialize counters and system parameters */
    nfill = 0;
    ncement = 0;
    ncemdis = 0;
    npr = 0;
    aggsise = 0;

    /* clear the 3-D system to all porosity to start */
    for (ig = 1; ig <= SYSSIZE; ig++) {
for(jg = 1; jg <= SYSSIZE; jg + +) {
    for(kg = 1; kg <= SYSSIZE; kg + +) {
        cement [ig] [jg] [kg] = POROSITY;
    }
}

/* present menu and execute user choice */
do{
    printf(" \n Input User Choice \n\n\n1) Add a flat inert aggregate to microstructure \n\n2) Add spherical particles (C3S or filler) to microstructure \n\n3) Add one-pixel filler particles to microstructure \n\n4) Hydrate microstructure \n\n5) Measure phase fractions \n\n6) Measure phase fractions as a function \n\n7) Measure single phase connectivity \n\n8) Measure total solids connectivity \n\n9) Exit \n\n");
    scanf("%d", &userc);
    printf("%d \n", userc);
    switch (userc) {
    case 1:
        addagg();
        break;
    case 2:
        create();
        break;
    case 3:
        filler();
        break;
    case 4:
        hydrate();
        break;
    case 5:
        measure();
        break;
    case 6:
        measagg();
        break;
    }
case 7:
    connect();
    break;

case 8:
    console();
    break;

default:
    break;

} while (userc < 9);
APPENDIX D
Modifying HYDRA3D

It is recognized that other cement researchers may want to use HYDRA3D as a starting point for developing their own simulations of material microstructure and properties. This appendix will attempt to provide a few guidelines for this process.

1) Maintain an original version of the code.

The user should not modify the original version of HYDRA3D, but rather should copy the source file to a new file and modify the new file to suit their needs. In this manner, the original code is always available as a baseline.

2) Limit the scope of modifications.

Since HYDRA3D has been developed in a modular fashion, the user should be able to make changes without changing every module in the program. For example, to change the shape of particles (from spheres to ellipses for instance), the user should only need to change the routines CREATE, GSPHERE, and CHKSPH to create the appropriately shaped particles. Since all routines operate at the pixel level, the hydration and analysis routines are independent of initial particle shape. To change the characteristics of the mineral admixture particles, the user would modify the routines MVCHANT and possibly ADDEXT to reflect the new reaction and volume stoichiometry of the pozzolanic reaction.

New analysis routines can simply be added as new modules, with a new menu selection added to allow the user to select the new feature. The three-dimensional microstructural representation is available globally so that any new module can easily access individual pixels of the current microstructure.

3) Make changes in an incremental fashion

If a user intends to make several modifications to the code, they should be made sequentially as opposed to concurrently. Although interactions between modifications are always a concern, sequential modification should limit the time needed to develop and debug the new (changed) program.

4) Understand the existing code before modifying it

Above all, the user should be sure that they understand the workings of the existing code before attempting any modifications!
APPENDIX E
System Requirements

The system configuration needed to successfully execute the HYDRA3D program will depend largely on the computing environment. Since the program is designed to implement a rather large scale simulation (one million pixel elements), the system requirements are not trivial. Sample system configurations at NIST on which HYDRA3D has been successfully implemented are outlined in Table II. While the source code file size remains constant from system to system, the executable code file size and memory needed for execution are both functions of the system being considered, as the default implementation of integer and real variable sizes will vary from computer to computer. At a minimum, it appears that 4 Mb of system memory would be required to execute HYDRA3D without extensive paging. Disk space requirements are generally minimal.

<table>
<thead>
<tr>
<th>Computer System</th>
<th>Language</th>
<th>Source Code File Size (kilobytes)</th>
<th>Executable File Size (kilobytes)</th>
<th>Memory for Execution (Megabytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONVEX C120</td>
<td>Fortran</td>
<td>38</td>
<td>264</td>
<td>11</td>
</tr>
<tr>
<td>CONVEX C120</td>
<td>C</td>
<td>30</td>
<td>74</td>
<td>3.5</td>
</tr>
<tr>
<td>SUN 3/160 ffpa accel.</td>
<td>C</td>
<td>30</td>
<td>41</td>
<td>3.3</td>
</tr>
<tr>
<td>CRAY Y-MP</td>
<td>Fortran</td>
<td>38</td>
<td>833</td>
<td>21.5</td>
</tr>
<tr>
<td>CRAY Y-MP</td>
<td>C</td>
<td>30</td>
<td>13228</td>
<td>14.2</td>
</tr>
</tbody>
</table>

The time required to execute HYDRA3D will naturally depend on the "problem" being simulated. However, to provide some idea of time requirements, Table III lists the execution times for the Fortran and C versions of HYDRA3D on a variety of computers. All times are those necessary to execute example 2 (w/c = 0.5) from Section 4.2 of this report. The times may seem quite large, but it should be kept in mind that HYDRA3D simulates the complex process of cement hydration for a relatively large system, not a trivial task.
Table III
Timing Benchmarks for HYDRA3D
(Example 2 from Section 4.2)

<table>
<thead>
<tr>
<th>Computer System</th>
<th>Programming Language</th>
<th>Execution Time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUN 3/160 (ffpa)</td>
<td>C</td>
<td>35932</td>
</tr>
<tr>
<td>CONVEX C120</td>
<td>C</td>
<td>9159</td>
</tr>
<tr>
<td>CONVEX C120</td>
<td>Fortran</td>
<td>9009</td>
</tr>
<tr>
<td>CRAY Y-MP</td>
<td>C</td>
<td>2197</td>
</tr>
<tr>
<td>CRAY Y-MP</td>
<td>Fortran</td>
<td>1123</td>
</tr>
</tbody>
</table>
**Guide to Using HYDRA3D: A Three-Dimensional Digital-Image-Based Cement Microstructural Model**

**Dale P. Bentz and Edward J. Garboczi**

A computer program, HYDRA3D, to simulate cement microstructural development and quantify microstructural characteristics has been developed. HYDRA3D is a menu drive program available in either Fortran or C which allows a user to create a starting microstructure, execute hydration, measure phase fractions, and assess phase connectivity. This manual outlines the conceptual model on which HYDRA3D is based, describes the programs in detail, and provides examples of program usage. A system calling diagram, source code listings, guidelines for modifying the programs, and system requirements are provided in the Appendices.

**Cement; computer modelling; hydration; interfacial zone; microstructure; percolation; simulation; software**

**ELECTRONIC FORM**