

# CitcomS

narrated by Ondřej Šrámek

at

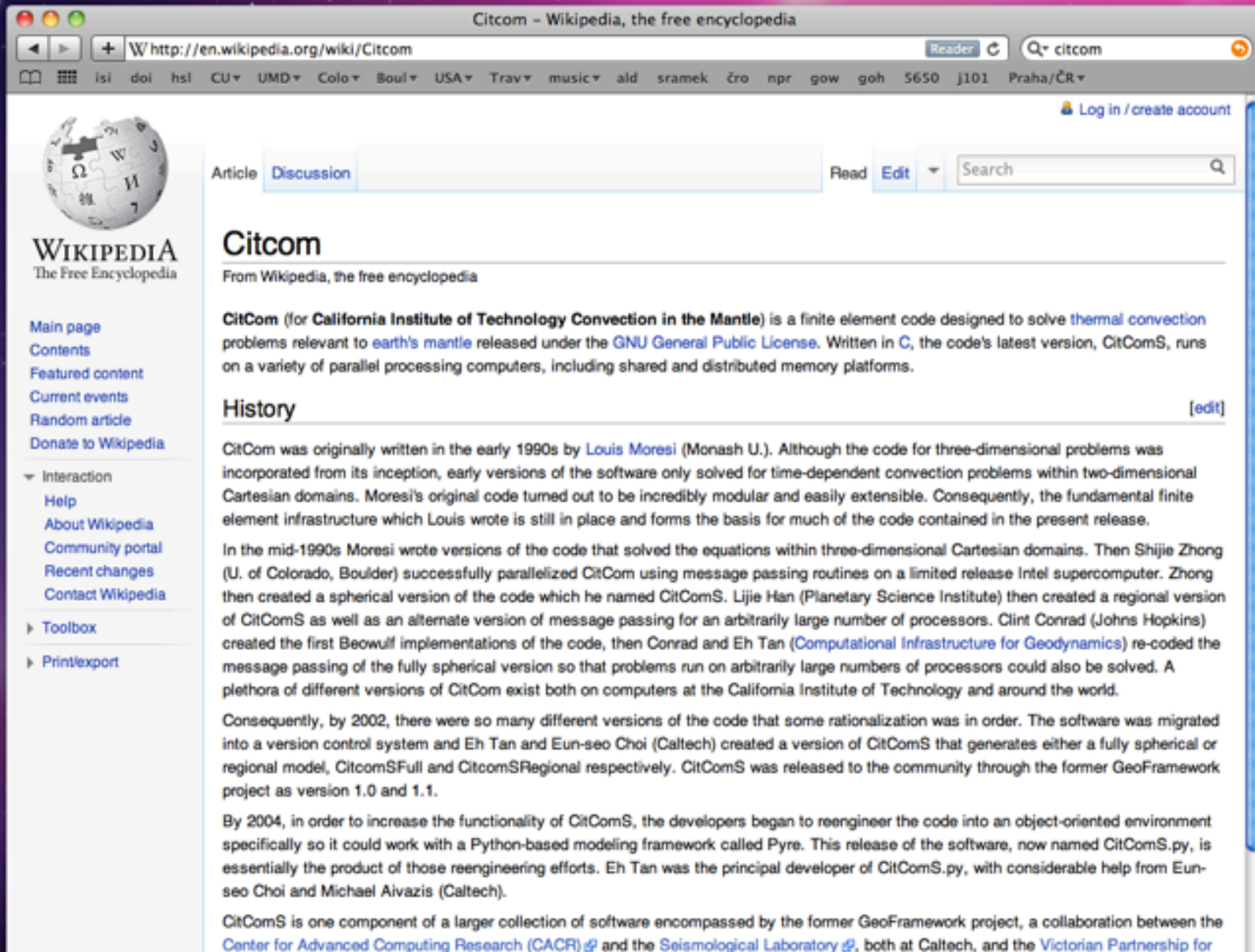
Department of Geophysics  
Charles University in Prague

March 28 & 30, 2011

# CitcomS

- 3-D spherical global finite element highly parallel convection code
- written by geophysicist(s) specifically for convection in planetary mantles
- variable viscosity, tracers & compositions, phase transitions, dynamic topography, geoid
- open source
- maintained, documented, supported (CIIG)

<http://en.wikipedia.org/wiki/Citcom>



The screenshot shows a web browser window with the title "Citcom - Wikipedia, the free encyclopedia". The address bar contains "http://en.wikipedia.org/wiki/Citcom". The browser's search bar shows "citcom". The page content includes the Wikipedia logo, a navigation menu on the left, and the main article text. The article title is "Citcom" and it is categorized as "Article" and "Discussion". The text describes CitCom as a finite element code for solving thermal convection problems, developed by Louis Moresi and others at Caltech. It mentions various versions of the code and its migration to a version control system.

Citcom - Wikipedia, the free encyclopedia

Wikipedia - The Free Encyclopedia

Main page  
Contents  
Featured content  
Current events  
Random article  
Donate to Wikipedia

Interaction  
Help  
About Wikipedia  
Community portal  
Recent changes  
Contact Wikipedia

Toolbox  
Print/export

Article Discussion Read Edit Search

## Citcom

From Wikipedia, the free encyclopedia

**CitCom** (for **California Institute of Technology Convection in the Mantle**) is a finite element code designed to solve **thermal convection** problems relevant to **earth's mantle** released under the **GNU General Public License**. Written in **C**, the code's latest version, **CitComS**, runs on a variety of parallel processing computers, including shared and distributed memory platforms.

### History

CitCom was originally written in the early 1990s by **Louis Moresi** (Monash U.). Although the code for three-dimensional problems was incorporated from its inception, early versions of the software only solved for time-dependent convection problems within two-dimensional Cartesian domains. Moresi's original code turned out to be incredibly modular and easily extensible. Consequently, the fundamental finite element infrastructure which Louis wrote is still in place and forms the basis for much of the code contained in the present release.

In the mid-1990s Moresi wrote versions of the code that solved the equations within three-dimensional Cartesian domains. Then Shijie Zhong (U. of Colorado, Boulder) successfully parallelized CitCom using message passing routines on a limited release Intel supercomputer. Zhong then created a spherical version of the code which he named CitComS. Lijie Han (Planetary Science Institute) then created a regional version of CitComS as well as an alternate version of message passing for an arbitrarily large number of processors. Clint Conrad (Johns Hopkins) created the first Beowulf implementations of the code, then Conrad and Eh Tan (**Computational Infrastructure for Geodynamics**) re-coded the message passing of the fully spherical version so that problems run on arbitrarily large numbers of processors could also be solved. A plethora of different versions of CitCom exist both on computers at the California Institute of Technology and around the world.

Consequently, by 2002, there were so many different versions of the code that some rationalization was in order. The software was migrated into a version control system and Eh Tan and Eun-seo Choi (Caltech) created a version of CitComS that generates either a fully spherical or regional model, CitcomSFull and CitcomSRegional respectively. CitComS was released to the community through the former GeoFramework project as version 1.0 and 1.1.

By 2004, in order to increase the functionality of CitComS, the developers began to reengineer the code into an object-oriented environment specifically so it could work with a Python-based modeling framework called Pyre. This release of the software, now named CitComS.py, is essentially the product of those reengineering efforts. Eh Tan was the principal developer of CitComS.py, with considerable help from Eun-seo Choi and Michael Aivazis (Caltech).

CitComS is one component of a larger collection of software encompassed by the former GeoFramework project, a collaboration between the **Center for Advanced Computing Research (CACR)** and the **Seismological Laboratory**, both at Caltech, and the **Victorian Partnership for**

**Citcom** = **C**alifornia **I**nstitute of **T**echnology **C**onvection in the **M**antle

**Citcom** ... 2-D then 3-D cartesian serial [Moresi & Solomatov 1995, Moresi & Gurnis 1996]

+ spherical geometry

+ new grid design

+ parallel computing

+ full multigrid algorithm

→ **CitcomS** [Zhong & Zuber 2000]

+ **tracers** (chemical components) [McNamara & Zhong 2004]

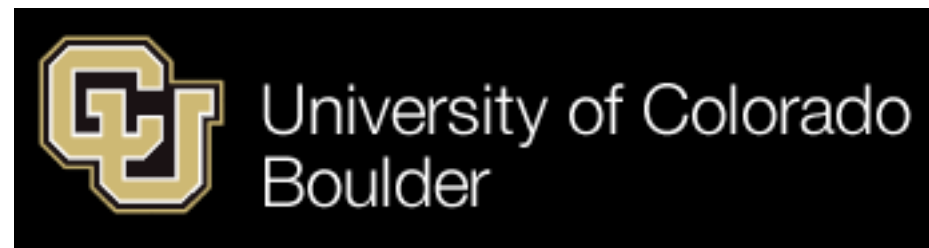
In early 2000's Shijie Zhong provided the code to "CIG" or Computational Infrastructure in Geodynamics

[www.geodynamics.org](http://www.geodynamics.org) where it has been maintained,

documented and supported, messed with (i.e., Pyre framework...)



open source version  
You will use this one



Shijie's version  
I am familiar with this one

The “guts” are the same but the “skin” is different  
(Pyre, file structure, ...)

# Can solve

- classical Boussinesq
- extended Boussinesq
- compressible, anelastic (i.e, TALA or truncated anelastic liquid approximation)
  
- global spherical shell
- regional spherical shell domain

# Governing equations – dimensional

TALA (truncated anelastic liquid approximation)

$$(\rho u_i)_{,i} = 0 \quad (1.1)$$

$$-P_{,i} + \left( \eta(u_{i,j} + u_{j,i} - \frac{2}{3}u_{k,k}\delta_{ij}) \right)_{,i} - \delta\rho g\delta_{ir} = 0 \quad (1.2)$$

$$\rho c_P (T_{,t} + u_i T_{,i}) = \rho c_P \kappa T_{,ii} + \rho \alpha g u_r T + \Phi + \rho (Q_{L,t} + u_i Q_{L,i}) + \rho H \quad (1.3)$$

$$\frac{\partial C}{\partial t} + (\mathbf{u} \cdot \nabla) C = 0$$

$$\delta\rho = -\alpha\bar{\rho}(T - \bar{T}_a) + \delta\rho_{ph}\Gamma + \delta\rho_{ch}C \quad (1.4)$$

thermal expansion, phase change, compositional buoyancy

[from CIG manual]

# Governing equations – non-dimensional

$$u_{i,i} + \frac{1}{\bar{\rho}} \frac{d\bar{\rho}}{dr} u_r = 0$$

$$-P_{,i} + \left( \eta(u_{i,j} + u_{j,i} - \frac{2}{3}u_{k,k}\delta_{ij}) \right)_{,i} + (Ra\bar{\rho}\alpha T - Ra_b\Gamma - Ra_c C)g\delta_{ir} = 0$$

$$\bar{\rho}c_P (T_{,t} + u_i T_{,i}) \left( 1 + 2\Gamma(1 - \Gamma) \frac{\gamma_{ph}^2}{d_{ph}} \frac{Ra_b}{Ra} Di (T + T_0) \right) = \bar{\rho}c_P \kappa T_{,ii}$$

$$- \bar{\rho}\alpha g u_r Di (T + T_0) \left( 1 + 2\Gamma(1 - \Gamma) \frac{\gamma_{ph}}{d_{ph}} \frac{Ra_b}{Ra} \right) + \frac{Di}{Ra} \Phi + \bar{\rho}H$$

$$Ra = \frac{\rho_0 g_0 \alpha_0 \Delta T R_0^3}{\eta_0 \kappa_0}$$

$$Ra_b = Ra \frac{\delta\rho_{ph}}{\rho_0 \alpha_0 \Delta T}$$

$$Ra_c = Ra \frac{\delta\rho_{ch}}{\rho_0 \alpha_0 \Delta T}$$

$$Ra_H = Ra_H \frac{R_0^3 - R_{CMB}^3}{3R_0^3}$$

[from CIG manual]

$$Di = \frac{\alpha_0 g_0 R_0}{c_{P0}}$$



# Viscosity

Obviously, user can define various viscosity options. I will use the following setting in 'inputTC10':

Viscosity=system	viscosity depends on system state (temperature, ...)
rheol=3 TDEPV=on	option 3 is defined in function <code>visc_from_T</code> in file 'Viscosity_Structure.c'
VISC_UPDATE=on	viscosity is updated (every other <u>timestep</u> )
<u>visc_smooth_method</u> =1	??

In option 3, the viscosity is calculated based on dimensional equation

$$\eta = \eta_{ref}(r) \exp\left[\frac{E' + PV'}{RT}\right] = \eta_{ref}(r) \exp\left[\frac{E' + V' \rho_0 g (1-r)}{RT}\right],$$

that is temperature- and pressure/depth-dependent viscosity (through activation energy  $E'$  and activation volume  $V'$ ) superimposed on a prescribed radial profile  $\eta_{ref}(r)$  (viscosity layering). Taking the CMB value as the reference viscosity and performing non-dimensionalization, one gets

$$\eta = \eta_r \exp\left[\frac{E + V(1-r)}{T_s + T} - \frac{E + V(1-r_c)}{T_s + 1}\right],$$

where

$$E = E'/(R \Delta T), \quad V = \rho_0 g R_0 V'/(R \Delta T), \quad \eta_r(r) = \eta_{ref}(r)/\eta_{ref}(r_{CMB}), \quad T_s = T_{surf}/\Delta T$$

(see Roberts & Zhong 2006 JGR; note that different formulations were used in other CitcomS papers, e.g. Zhong et al. 2000 JGR, Zhong et al. 2008 G3)

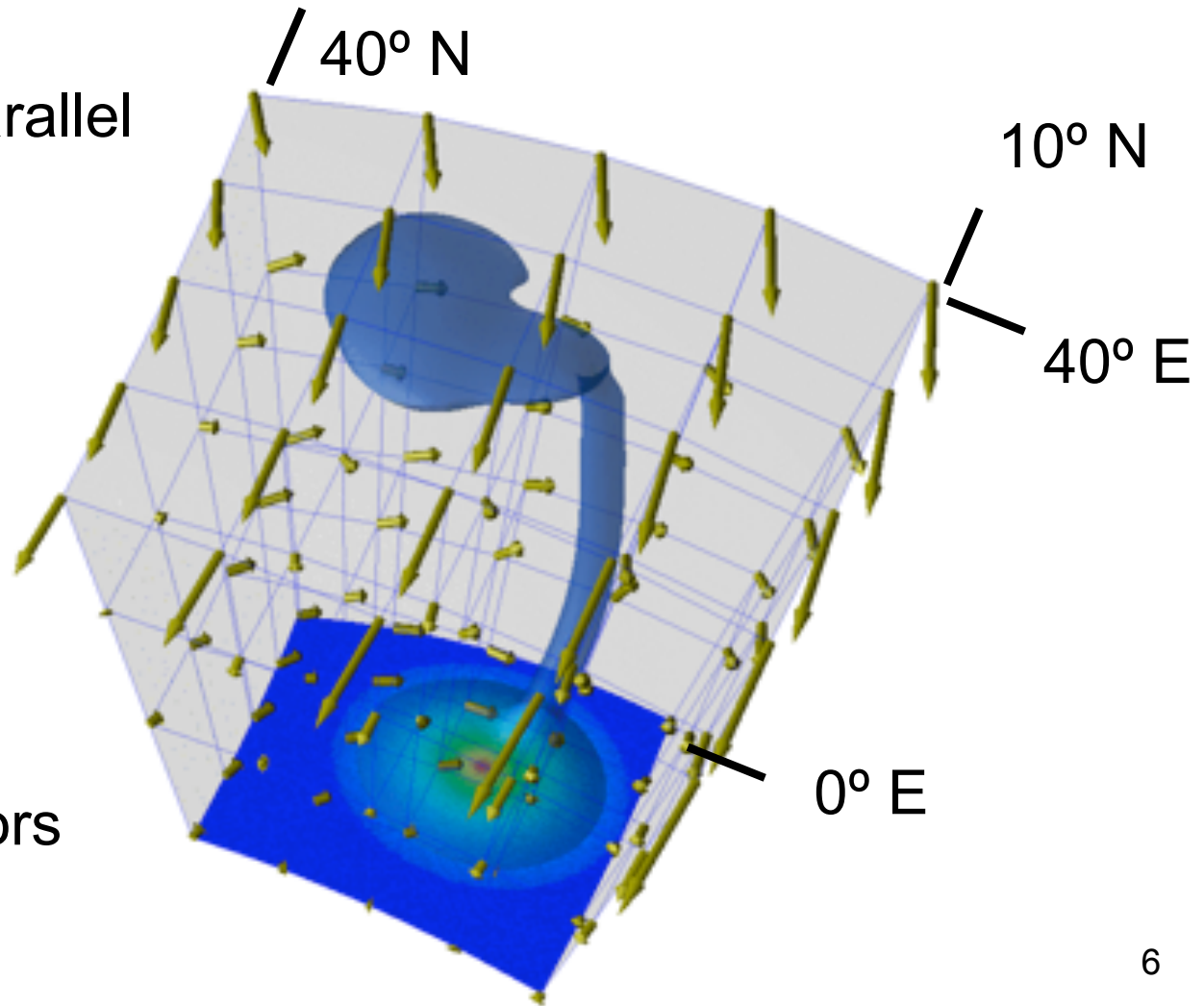
[This relates to CU Boulder version]

Many different options – see CIG Manual, Sec.A.I.I I  
(or you can code your own...)

# Regional Mesh

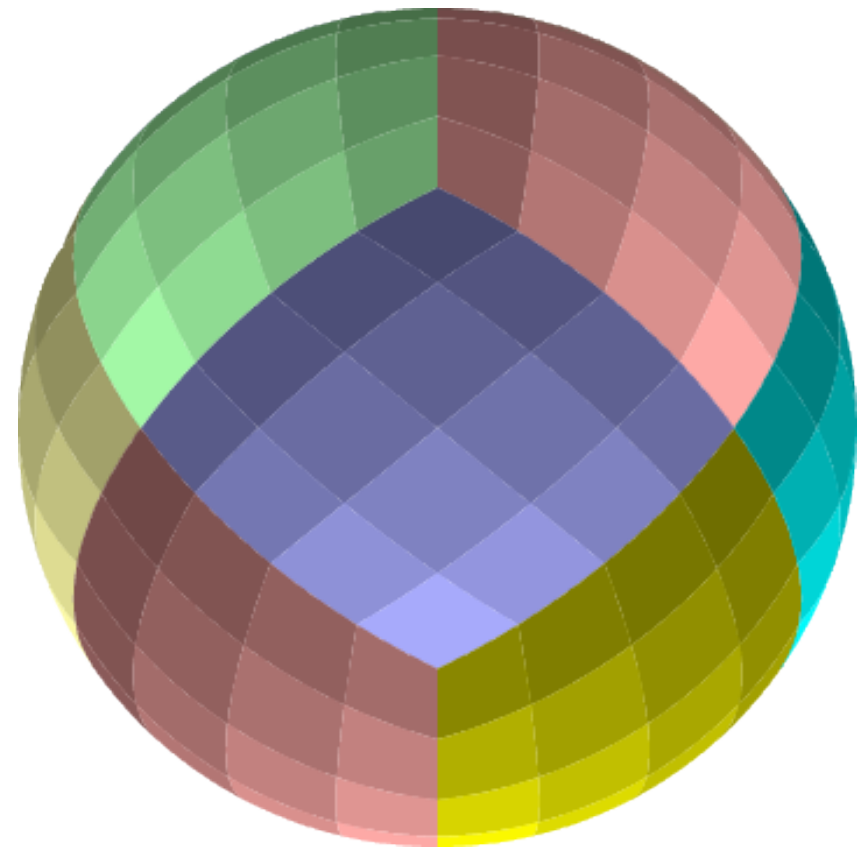
Grid lines are parallel to longitude and latitude

The whole domain can be partitioned into NxMxL processors

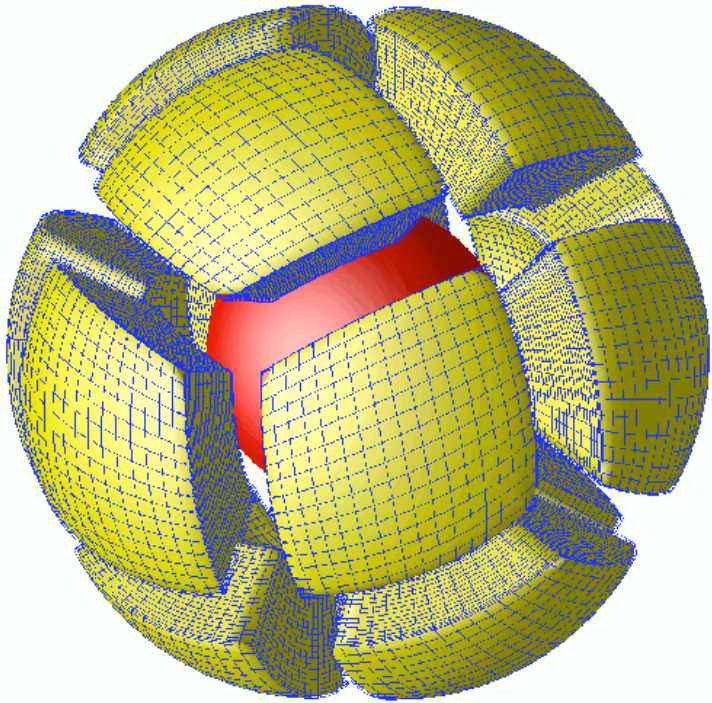


# Global Mesh

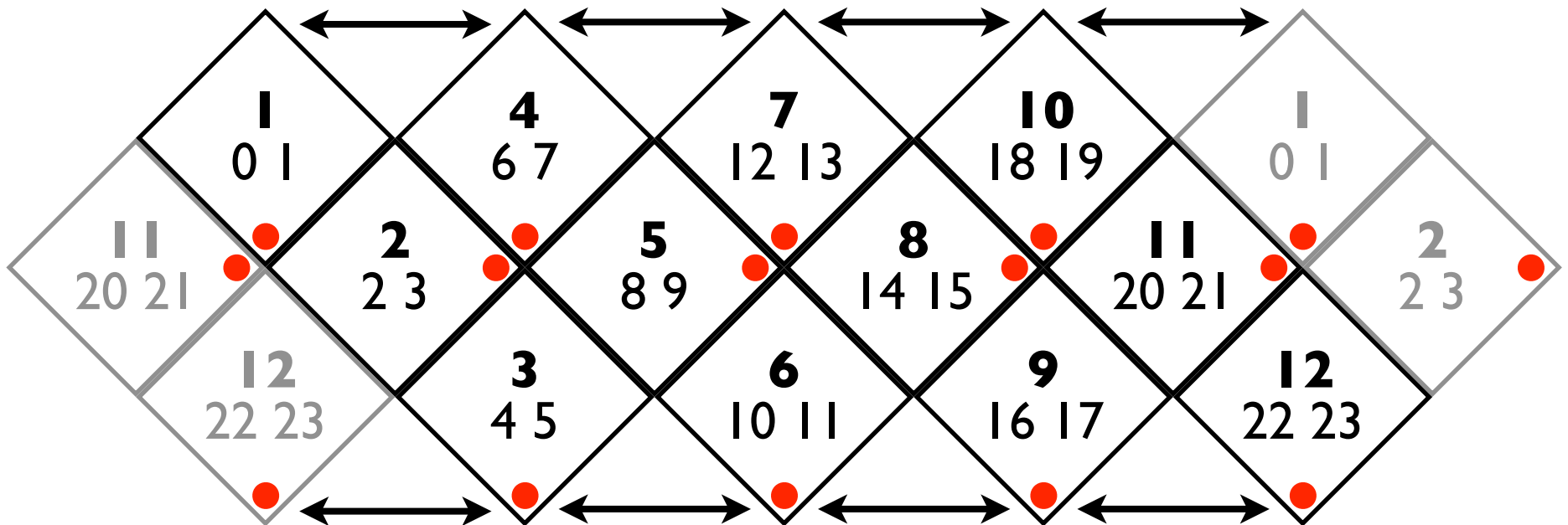
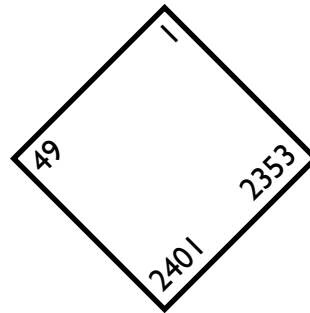
- 12 caps
- Each cap extends from the surface to the CMB
- Each cap can be partitioned into  $N \times N \times M$  processors
- $12 \times N \times N \times M$  processors in total (N=4 in this figure)

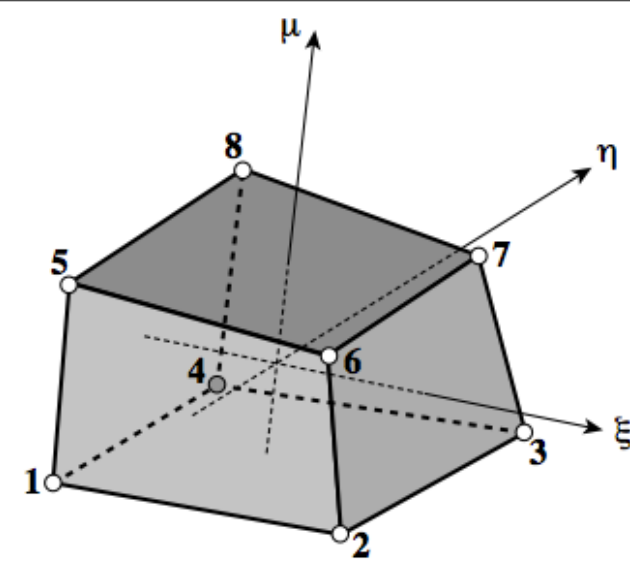
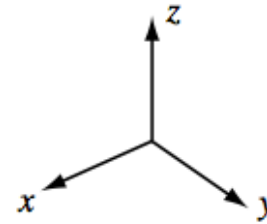


# 12 caps



need at least 12 cores for full spherical shell



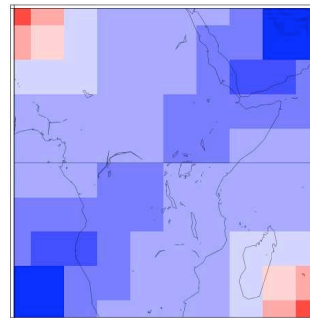
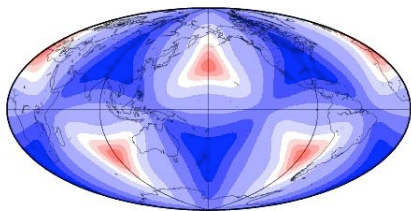


- trilinear hexahedral elements
  - brick elements – only approximate sphericity
  - 8 velocity nodes, trilinear approximation
  - 1 pressure node, constant

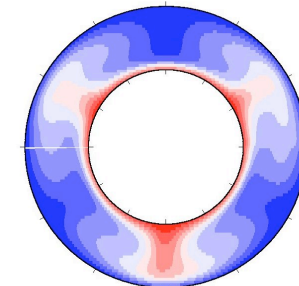
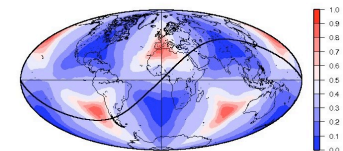
# Visualization in GMT

- `plot_layer.py`
  - plot horizontal cross section, for both regional and global versions
- `plot_annulus.py`
  - plot radial cross section, for global version only
- sample data files in *visual/samples/*

## `plot_layer.py`

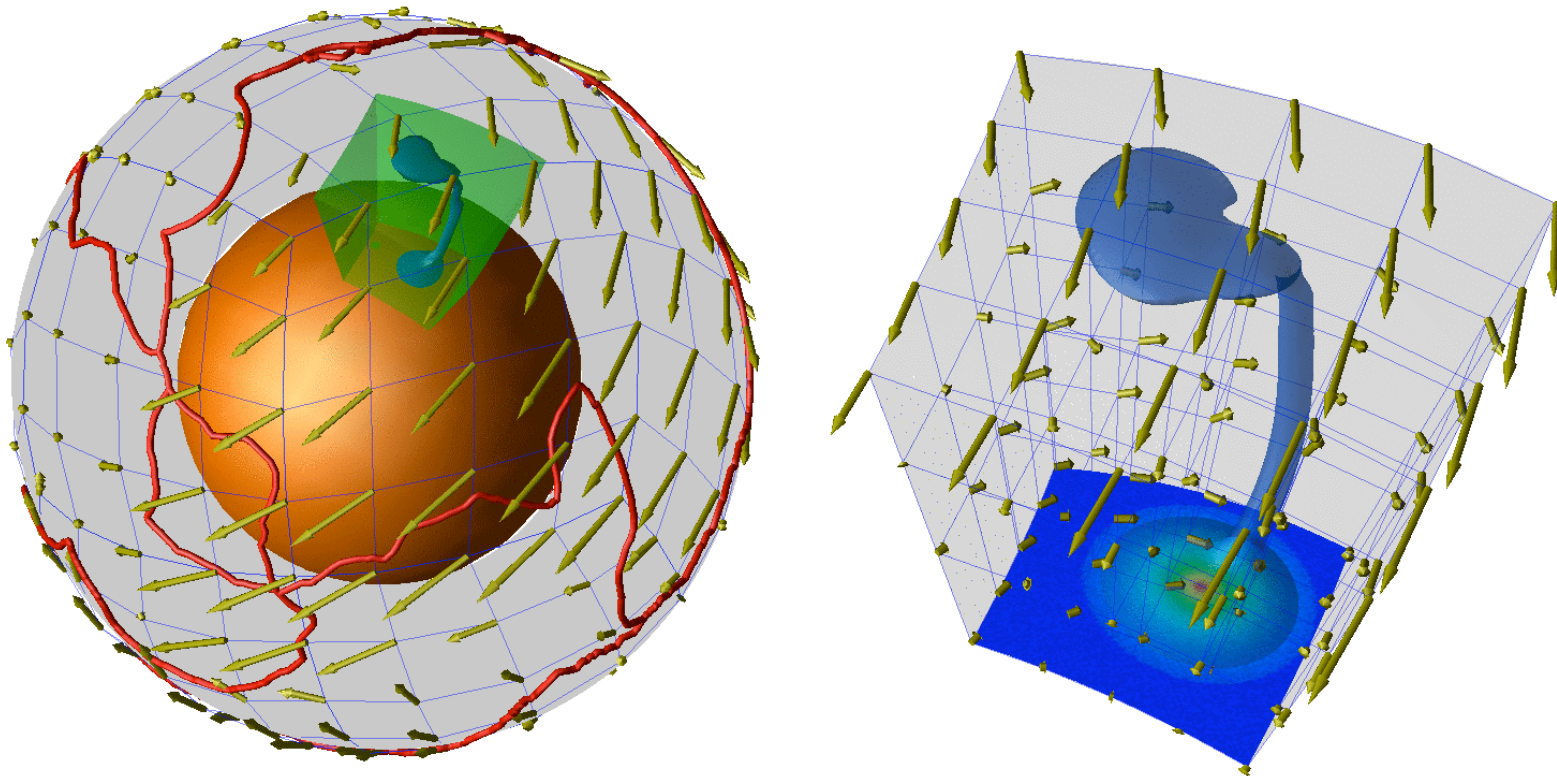


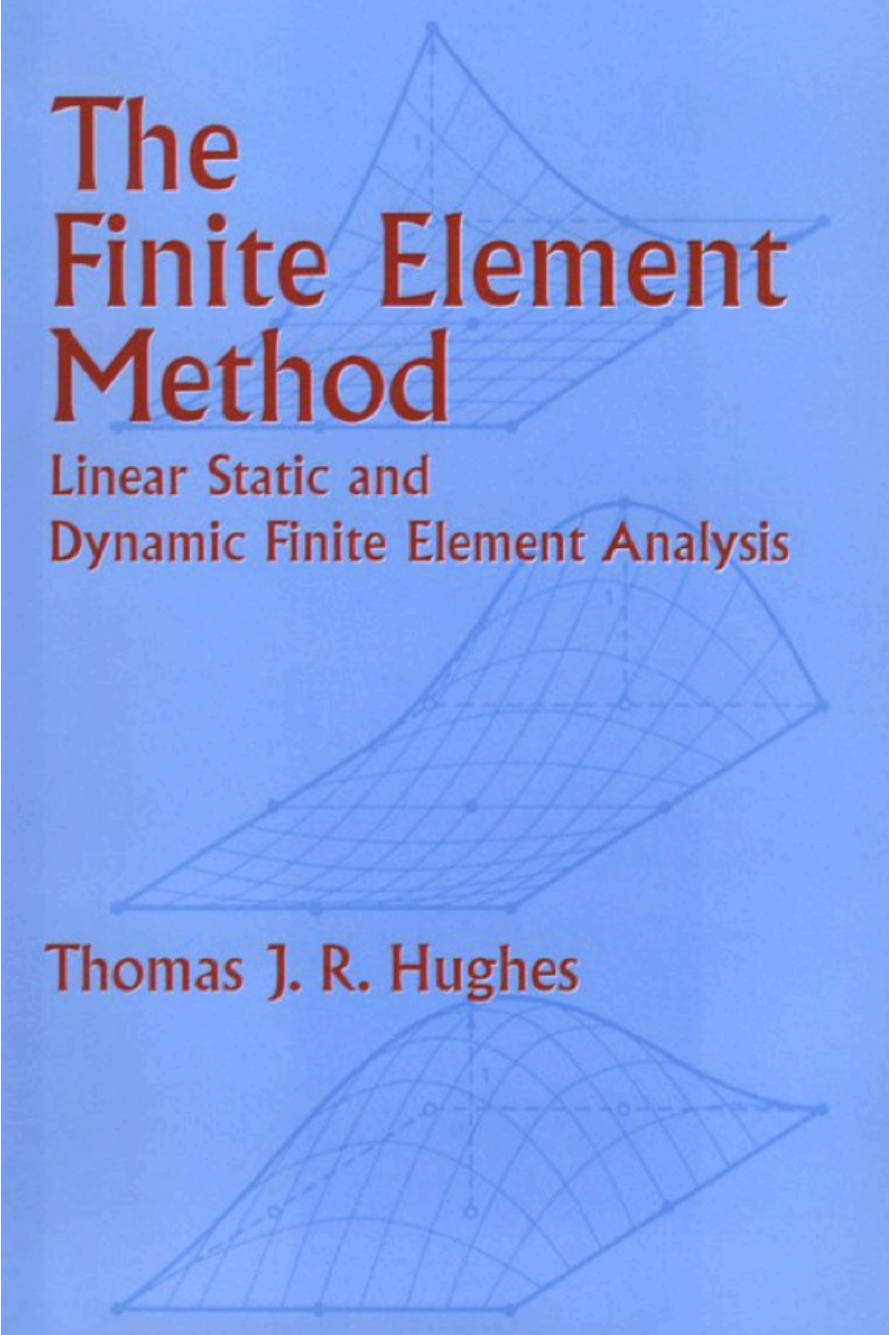
## `plot_annulus.py`





# Visualization in OpenDX





# The Finite Element Method

Linear Static and  
Dynamic Finite Element Analysis

Thomas J. R. Hughes



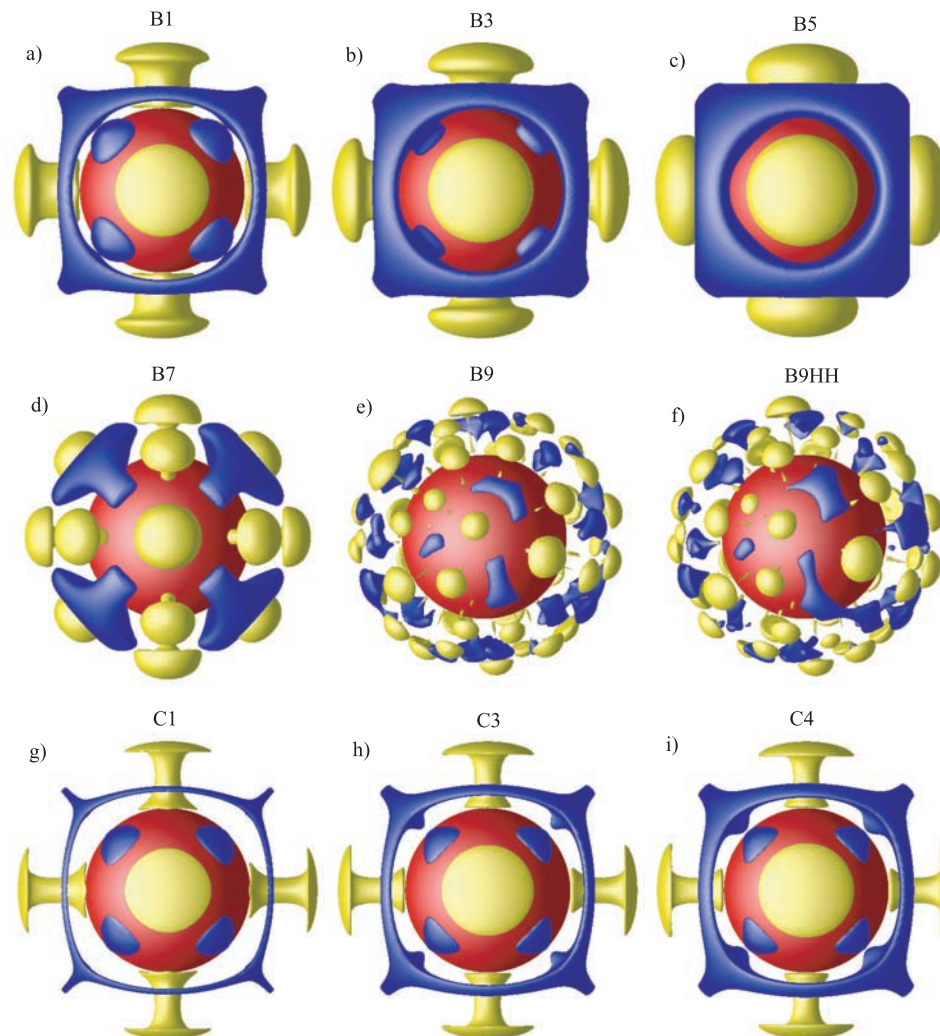
# Major ingredients of the mathematical & numerical model

- streamline upwind Petrov-Galerkin approx (SUPG) for the energy equation
- mixed formulation in primitive variables (P,v) and Uzawa algorithm with two-loop iterations for Stokes problem
- Uzawa: outer loop (P) – preconditioned conjugate gradient method; inner loop (v) – full multigrid methods
- Gauss-Seidel iteration for inner nodes, Jacobi iteration for shared nodes
- predictor–corrector and 2nd order Runge-Kutta to advect tracers, ratio method used to map tracers to composition
- Poisson equation for gravitational potential solved with a spectral method

# Benchmarks



ZHONG ET AL.: BENCHMARKS OF 3-D SPHERICAL CONVECTION MODELS 10.1029/2008GC002048



Zhong et al. 2008 G<sup>3</sup>

# Parallel efficiency



**Table 1.** CPU Time With Different Number of Cores

$N_c^a$	Total Time (s)	Time for Zeroth Step (s)	Iterations <sup>b</sup>	Time Per v Iteration (s)	Efficiency (%)
12(1 × 1 × 1)	69.8	9.3	112(118)	0.59	100
24(1 × 1 × 2)	64.1	9.8	95(103)	0.62	95
48(2 × 2 × 1)	53.7	9.7	73(78)	0.69	86
96(2 × 2 × 2)	53.9	8.8	74(79)	0.68	87
192(4 × 4 × 1)	47.2	11.3	55(63)	0.75	79
384(4 × 4 × 2)	46.8	8.2	55(58)	0.81	73
768(4 × 4 × 4)	52.4	10.0	58(61)	0.86	69
1536(8 × 8 × 2)	58.2	16.0	60(70)	0.83	71
3072(8 × 8 × 8)	59.1	17.5	54(57)	1.04	57

<sup>a</sup> $N_c$  stands for the number of core. The numbers in the parentheses represent the domain decomposition in each of 12 spherical caps for CitcomS. For example, 2 × 2 × 2 indicates that each cap is further divided into two in each of the three directions with the last number for the radial direction.

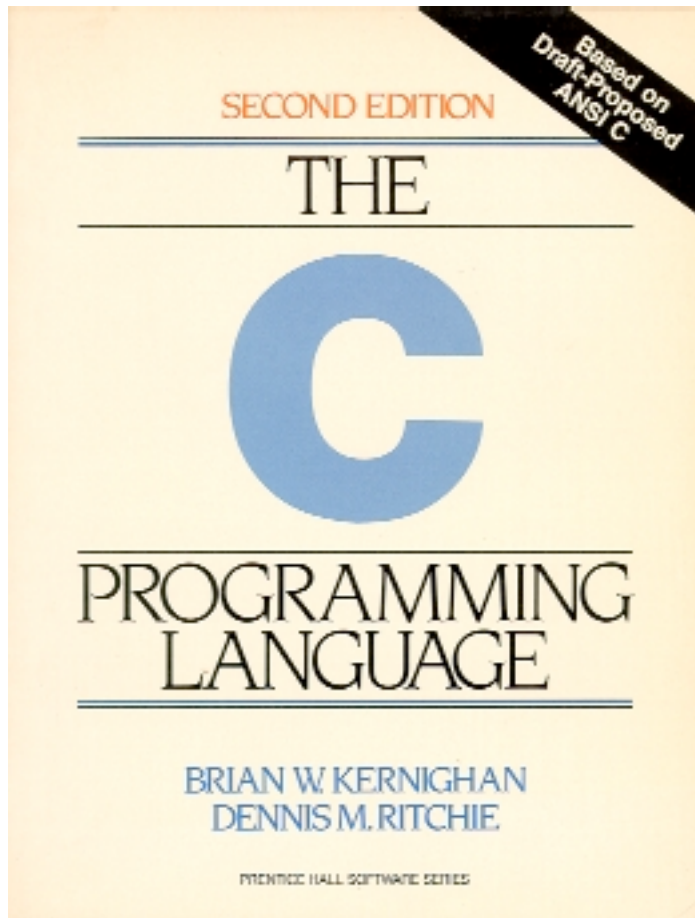
<sup>b</sup>The numbers in and out of the parentheses represent the numbers of inner loop velocity iteration and of outer loop pressure iteration, respectively.

# C

## The C Book – Table of Contents

This is the online version of *The C Book*, second edition by Mike Banahan, Declan Brady and Mark Doran, originally published by Addison Wesley in 1991. This version is made [freely available](http://publications.gbdirect.co.uk/c_book/).

[http://publications.gbdirect.co.uk/c\\_book/](http://publications.gbdirect.co.uk/c_book/)

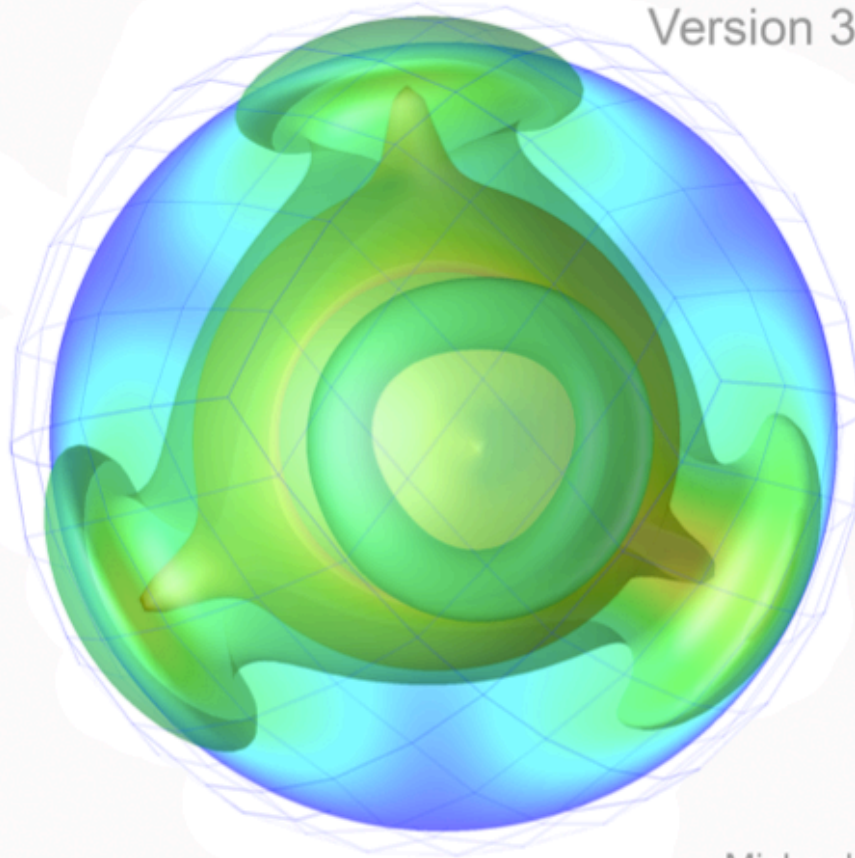


## C Reference Card (ANSI)

<b>C Reference Card (ANSI)</b>	<b>Constants</b>	<b>Flow of Control</b>
<b>Program Structure/Functions</b> type <i>func</i> (type) {...} function declarations type name external variable declarations main() { declarations local variable declarations statements } type <i>func</i> (arg1, ... ) { declarations statements return value; } /* #f comments main with args exit(arg) terminate execution	long (suffix) L or l float (suffix) F or f exponential form e octal (prefix zero) 0 hexadecimal (prefix zero-x) 0x or 0X character constant (char, octal, hex) 'a', '\ooo', '\xhh' newline, cr, tab, backspace '\n', '\r', '\t', '\b' special characters '\\', '\', '\'', '\", '\n', '\t', '\f', '\r', '\a', '\e' string constant (ends with '\0') "abc...de"	statement terminator ; block delimiters { } exit from switch, while, do, for, next iteration of while, do, for, go to label return value from function return expr <b>Flow Constructions</b> if statement if (expr) statement else if (expr) statement else statement while statement while (expr) statement for statement for (expr1; expr2; expr3) statement do statement do statement while (expr); switch statement switch (expr) { case const1: statement1; break; case const2: statement2; break; default: statement; }
<b>C Preprocessor</b> #include <filename> include library file #include "filename" include user file #define name text replacement macro Example: #define max(a,b) ((a)>(b) ? (a) : (b)) #undef name # quoted string in replace concatenate args and reorgan conditional execution #if, #else, #elif, #endif #ifdef, #ifndef, #defined, #undef line continuation char \	<b>Pointers, Arrays &amp; Structures</b> declare pointer to type type *name declare function returning pointer to type type *f() declare pointer to function returning type type (type)() generic pointer type void * null pointer NULL pointer *pointer address of object name &name array name[dim] multi-dim array name[dim1][dim2]... <b>Structures</b> struct tag { declarations }; create structure struct tag name member of structure from template name.member member of pointed to structure pointer -> member Example: (p) x and p->x are the same single value, multiple type structure union bit field with b bits member : b <b>Operators (grouped by precedence)</b> structure member operator name.member structure pointer pointer->member increment, decrement ++, -- plus, minus, logical not, bitwise not *, -, !, ~ indirection via pointer, address of object *pointer, &name cast expression to type (type) expr size of an object sizeof multiply, divide, modulus (remainder) *, /, % add, subtract +, - left, right shift, left open <<, >> comparisons >, >=, <, <=, ==, != bitwise and & bitwise exclusive or ^ bitwise or (incl)   logical and && logical or    conditional expression expr1 ? expr2 : expr3 assignment operators =, *=, /=, %= expression evaluation separation , Unary operators, conditional expressions and assignment operators group right to left, all others group left to right.	<b>ANSI Standard Libraries</b> <assert.h> <ctype.h> <errno.h> <float.h> <limits.h> <locale.h> <math.h> <setjmp.h> <signal.h> <stdarg.h> <stddef.h> <stdio.h> <stdlib.h> <string.h> <strings.h> <time.h> <b>Character Class Tests &lt;ctype.h&gt;</b> alphanumeric? isalnum(c) alpha? isalpha(c) alphanumeric? isalnum(c) control character? iscntrl(c) lower case letter? islower(c) printing character (incl space)? isprint(c) printing char convert space, letter, digit? ispspace(c) space, formatted, newline, cr, tab, vtab? isspace(c) upper case letter? isupper(c) hexadecimal digit? isxdigit(c) convert to lower case? tolower(c) convert to upper case? toupper(c)
<b>Data Types/Declarations</b> character (1 byte) char integer int float (single precision) float float (double precision) double short (16 bit integer) short long (32 bit integer) long signed positive and negative only positive pointer to int, float, ... *int, *float, ... enumeration constant enum constant (unchanging) value const declare external variable extern register variable register local to source file static no value void structure struct create name by data type typedef type name size of an object (type is size_t) sizeof object size of a data type (type is size_t) sizeof (type name)	<b>String Operations &lt;string.h&gt;</b> length of s strlen(s) copy ct to s strcpy(s,ct) up to a char strcat(s,ct) compare ca to ct strcmp(ca,ct) up to a char strstr(ct,ca) only first a char strchr(ct,ca) pointer to first c in ca pointer to last c in ca copy a chars from ct to s strcpy(s,ct) copy a chars from ct to s strncpy(s,ct,n) compare a chars of ca with ct strncmp(ca,ct,n) pointer to first c in first a chars of ca put c into first n chars of ca memset(s,c,n)	
<b>Initialization</b> initialize variable type name [=value]; initialize array type name [n][value]; initialize char string char name [n]*string		

# CitcomS

User Manual  
Version 3.1.1.1



Eh Tan  
Michael Gurnis  
Luis Armendariz  
Leif Strand  
Susan Kientz

[www.geodynamics.org](http://www.geodynamics.org)



# Bibliography

- Brooks, A. N., *A Petrov-Galerkin finite element formulation for convection dominated flows*, Ph.D. thesis, California Institute of Technology, Pasadena, California, 1981.
- Hughes, T. J. R., *The Finite Element Method: linear static and dynamic finite element analysis*, Dover Publications, Mineola, New York, 2000.
- McNamara, A. K., and S. Zhong, Thermochemical structures within a spherical mantle: Superplumes or piles?, *J. Geophys. Res.*, **109**, eid:B07402, doi:[10.1029/2003JB002847](https://doi.org/10.1029/2003JB002847), 2004.
- Moresi, L., and M. Gurnis, Constraints on the lateral strength of slabs from three-dimensional dynamic flow models, *Earth Planet. Sci. Lett.*, **138**(1-4), 15–28, doi:[10.1016/0012-821X\(95\)00221-W](https://doi.org/10.1016/0012-821X(95)00221-W), 1996.
- Moresi, L.-N., and V. S. Solomatov, Numerical investigation of 2D convection with extremely large viscosity variations, *Phys. Fluids*, **7**(9), 2154–2162, doi:[10.1063/1.868465](https://doi.org/10.1063/1.868465), 1995.
- Zhong, S., M. T. Zuber, L. Moresi, and M. Gurnis, Role of temperature-dependent viscosity and surface plates in spherical shell models of mantle convection, *J. Geophys. Res.*, **105**(B5), 11,063–11,082, doi:[10.1029/2000JB900003](https://doi.org/10.1029/2000JB900003), 2000.
- Zhong, S., A. McNamara, E. Tan, L. Moresi, and M. Gurnis, A benchmark study on mantle convection in a 3-D spherical shell using CitcomS, *Geochem. Geophys. Geosyst.*, **9**, eid:Q10017, doi:[10.1029/2008GC002048](https://doi.org/10.1029/2008GC002048), 2008.
- Tan, E., M. Gurnis, L. Armendariz, L. Strand, and S. Kientz, CitcomS User Manual version 3.1.1.1, Computational Infrastructure for Geodynamics, [www.geodynamics.org](http://www.geodynamics.org), 2010.