Ondřej's CitcomS Notes

Ondřej Šrámek

University of Colorado at Boulder

November 13, 2008

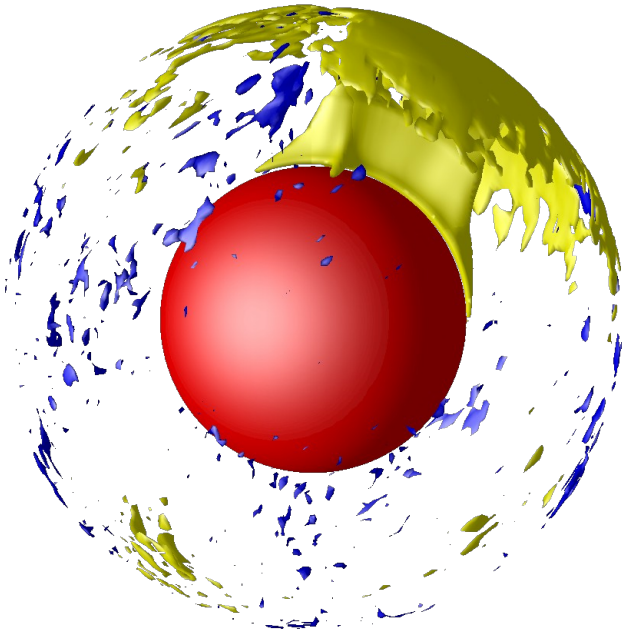


Table of Contents

Equations 4
Overview 4
  Biblio 4
  Code statistics 4
Code Structure 5
  main structure (Citcom.c) 5
  Source header files 5
    advection.h 5
    element_definitions.h 5
    sphere_communication.h 5
    viscosity_descriptions.h 5
    Convection_variables.h 6
    global_defs.h 6
    temperature_descriptions.h 6
    Zeros_of_J0_and_J1.h 6
  Source C files 6
    Advection_diffusion.c 6
    AKM_additions.c 6
    Boundary_conditions.c 6
    Construct_arrays.c 7
    Convection.c 7
    Drive_solvers.c 7
    Element_calculations.c 7
    General_matrix_functions.c 8
    Geometry_cartesian.c 8
    Global_operations.c 8
    Instructions.c 8
    Nodal_mesh.c 8
    Output.c 9
    Pan_problem_misc_functions.c 9
    Parallel_related.c 9
    Parsing.c 9
    Phase_change.c 9
    Plates_others.c 10
    Process_buoyancy.c 10
    Process_velocity.c 10
    Profiling.c 10
    Shape_functions.c 10
    Size_does_matter.c 10
    Solver_conj_grad.c 10
    Solver_multigrid.c 10
    Sphere_harmonics.c 10
    Sphere_related.c 11
    Stokes_flow_Incomp.c 11
    Topo_gravity.c 11
    Trace.c 11
    Viscosity_structures.c 12
Mesh 13
  Functions that set up the mesh located in: 14
  Mesh-related input & derived parameters 14
Viscosity 17

cluster.py script....17
Execution....17
Execution speed....17
Input file 18
Ouput files 18
in the execution directory: 18
in data directory on the main cluster node (eg nappo):....18
in data directory on the first computer node:....19
in data directory on the second computer node:....20
in data directories on odd# nodes:....20
in data directories on even# nodes:....20
in data directories on all computing nodes:....20
File output code flow....21
Data processing....22
spectrum analyses....22
files for Data Explorer....23
Data Explorer (dx)....23
Plume–cap separation....23
Temperature (and composition) movies....23

Equations

Overview

CitcomS = Citcom [*Moresi & Gurnis 1996*] +
+ spherical geometry & new grid design & parallel computing & full multigrid algorithm [Zhong & Zuber 2000]
+ thermochamical convection [McNamara & Zhong 2004]

- finite element method (*FEM*)
- primitive variable formulation (*i.e.*, velocity and pressure; see *Hughes 1987*)
- 3-D spherical shell geometry
- full multigrid (*FMG*) algorithm w/ consistent projection scheme
- tracers for monitoring composition

Method outline:

mixed formulation with primitive variables and Uzawa algorithm with two-loop iterations to solve Stokes
outer loop iteration for pressure: preconditioned conjugate gradient method
inner loop iteration for velocity: full multigrid method (*FMG*)
Gauss-Seidel smoothing for interior points
Jacobi iteration smoothing for points shared by processors
incompressibility constraint
brick elements (only approximates sphericity):
8 velocity nodes w/ trilinear interpolation
1 constant pressure node
streamline upwind Petrov-Galerkin method (*SUPG*)
predictor-corrector and 2nd order Runge-Kutta to update tracer positions
particle-ratio method to project tracer distribution to composition
Poisson equation with spectral method

Written in C, uses MPI for for inter-processor communication.

Biblio

- Moresi & Gurnis 1996 EPSL — Citcom
- Zhong *et al.* 2000 JGR — CitcomS
- McNamara & Zhong 2004 JGR — tracers
- Zhong *et al.* 2008 G^3 — CitcomS benchmark
- Hughes 1987 book — FEM primitive variable
- Ramage & Wathen 1994 IJNMF — Uzawa + incompressibility
- Brooks 1981 PhD Thesis — streamline upwing Petrov-Galerkin
- Bathe 1995 book — importance of brick elements for pressure determination
- Tackley & King 2003 G^3 — particle ratio method

Code statistics

- 8 header files (*.h) 3518 lines or 93K total
- 32 code source files (*.c) 29732 lines or 812K total

Code Structure

`main` *structure (`Citcom.c`)*

parallel process initilization
read_instructions — *Instructions.c*

··· (post_process if postprocessing) — *Process_velocity.c*

plate_velocity_boundary_conditions — *AKM_additions.c*
general_stokes_solver — *Drive_solvers.c*
remove_rigid_rot — *AKM_additions.c*

process_temp_field — *Process_buoyancy.c*
process_new_velocity — *Process_velocity.c*
write_bulk_data_to_files — *AKM_additions.c*
(process_temp_field, process_new_velocity, write_bulk_data_to_files: some post-processing & output)

iteration (if time stepping):

process_heating — *Advection_diffusion.c*

(E.next_buoyancy_field) — *Convection.c -> AD.c*

tracing — *prediction -* — *Trace.c*

process_temp_field — some post-processing & output — *Process_buoyancy.c*

plate_velocity_boundary_conditions — *AKM_additions.c*
general_stokes_solver — *Drive_solvers.c*
remove_rigid_rot — *AKM_additions.c*
process_new_velocity — some post-processing & output — *Process_velocity.c*

tracing — *- correction* — *Trace.c*

write_bulk_data_to_files — some post-processing & output — *AKM_additions.c*

end iteration.

parallel_process_termination

Source header files

`advection.h`

...

`element_definitions.h`

...

`sphere_communication.h`

...

`viscosity_descriptions.h`

...

Convection_variables.h

...

global_defs.h

defines structure `struct All_variables` and all embedded structures

temperature_descriptions.h

...

Zeros_of_J0_and_J1.h

...

Source C files

Advection_diffusion.c

void advection_diffusion_parameters(E)
void advection_diffusion_allocate_memory(E)
void PG_timestep(E)
void predictor(E,field,fielddot)
void corrector(E,field,fielddot,Dfielddot)
void pg_solver(E,T,Tdot,DTdot,bc,FLAGS)
void pg_shape_fn(E,el,PG,VV,rtf,diffusion,m)
void element_residual(E,el,PG,VV,field,fielddot,Eres,rtf,diff,BC,FLAGS,m)
void std_timestep(E)
void process_heating(E)

AKM_additions.c

...
void setup_material_groups(E)
...
void setup_depth_dependent_properties(E)
...
void get_net_rotation(E)
void remove_rigid_rot(E)
void remove_rigid_rot_akm(E)
...

Boundary_conditions.c

void velocity_boundary_conditions(E)
void temperature_boundary_conditions(E)
void velocity_refl_vert_bc(E)
void temperature_refl_vert_bc(E)
void horizontal_bc(E,BC,ROW,dirn,value,mask,onoff,level,m)
void velocity_apply_periodic_bcs(E)
void temperature_apply_periodic_bcs(E)
void strip_bcs_from_residual(E,Res,level)

void get_bcs_id_for_residual(E,level,m)
void temperatures_conform_bcs(E)
void velocities_conform_bcs(E,U)

Construct_arrays.c

void construct_ien(E)
void construct_id(E)
void construct_lm(E)
void construct_node_maps(E)
void construct_node_ks(E)
void rebuild_BI_on_boundary(E)
void construct_masks(E)
void construct_sub_element(E)
void construct_elt_ks(E)
void construct_elt_gs(E)
void construct_stiffness_B_matrix(E)
void construct_mat_group(E)

Convection.c

void convection_initial_temperature(E)
void read_temp_bin(E)
void read_UP_bin(E)
void read_temp(E)
void read_UP(E)
void get_slab_onto_grid (E)
void modify_slab_density(E)
void read_density_caps(E)
void get_layer_T(E,m,S,kk,k)
void read_slab_sphere_h (E)
void compute_slab(E)
void compute_geoid_slab(E)
void compute_geoid_slab_after_intp(E)
void read_slabgeoid(E)
void set_convection_defaults(E)
void read_convection_settings(E)
void convection_derived_values(E)
void convection_allocate_memory(E)
void convection_initial_fields(E)
void convection_boundary_conditions(E)

Drive_solvers.c

general_stokes_solver(E)

Element_calculations.c

void assemble_forces(E,penalty)
void get_elt_k(E,el,elt_k,lev,m,icon)
void assemble_del2_u(E,u,Au,level,strip_bcs)
void e_assemble_del2_u(E,u,Au,level,strip_bcs)
void n_assemble_del2_u(E,u,Au,level,strip_bcs)
void build_diagonal_of_K(E,el,elt_k,level,m)

void build_diagonal_of_Ahat(E)
void assemble_div_u(E,U,divU,level)
void assemble_grad_p(E,P,gradP,lev)
double assemble_dAhatp_entry(E,e,level,m)
void get_elt_g(E,el,elt_del,lev,m)
void get_elt_h(E,el,elt_h,m)
void get_elt_f(E,el,elt_f,bcs,m)
void get_aug_k(E,el,elt_k,level,m)

General_matrix_functions.c

...

Geometry_cartesian.c

void set_2dc_defaults(E)
void set_2pt5dc_defaults(E)
void set_3dc_defaults(E)
void set_3dsphere_defaults(E)

Global_operations.c

...

Instructions.c

void read_instructions(E,argc,argv)
void allocate_common_vars(E)
void allocate_velocity_vars(E)
void interuption()
void global_default_values(E)
void global_derived_values(E)
void read_initial_settings(E)
void check_bc_consistency(E)
void set_up_nonmg_aliases(E,j)
void common_initial_fields(E)
void initial_pressure(E)
void initial_velocity(E)

Nodal_mesh.c

void node_locations(E)
void flogical_mesh_to_real(E,data,level)
void p_to_nodes(E,P,PN,lev)
void p_to_centres(E,PN,P,lev)
void v_to_intpts(E,VN,VE,lev)
void visc_to_intpts(E,VN,VE,lev)
void visc_from_gint_to_nodes(E,VE,VN,lev)
void visc_from_nodes_to_gint(E,VN,VE,lev)
void visc_from_gint_to_ele(E,VE,VN,lev)
void visc_from_ele_to_gint(E,VN,VE,lev)

Output.c

void output_velo_related(E,file_number)
void output_velo_related_bin(E,file_number)
void output_temp(E,file_number)
void output_stress(E,file_number,SXX,SYY,SZZ,SXY,SXZ,SZY)
void print_field_spectral_regular(E,TG,sphc,sphs,proc_loc,filen)

Pan_problem_misc_functions.c

int get_process_identifier()
void unique_copy_file(E,name,comment)
void thermal_buoyancy(E,buoy)
double SIN_D(x)
double COT_D(x)
void * Malloc1(bytes,file,line)
int read_previous_field(E,field,name,abbr)
float cross2d(x11,x12,x21,x22,D)
double myatan(y,x)

Parallel_related.c

void parallel_process_initilization(E,argc,argv)
void parallel_process_termination()
void parallel_process_sync()
double CPU_time0()
void parallel_processor_setup(E)
void parallel_domain_decomp0(E)
void parallel_domain_decomp2(E,GX)
void parallel_domain_boundary_nodes(E)
void parallel_communication_routs_v(E)
void parallel_communication_routs_s(E)
void exchange_id_d(E, U, lev)
void exchange_node_d(E, U, lev)
void exchange_node_f(E, U, lev)
void exchange_snode_f(E, U1, U2, lev)
void scatter_to_nlayer_id (E,AUi,AUo,lev)
void gather_to_1layer_id (E,AUi,AUo,lev)
void gather_to_1layer_node (E,AUi,AUo,lev)
void gather_to_1layer_ele (E,AUi,AUo,lev)
void gather_TG_to_me0(E,TG)
void sum_across_depth_sph(E,sphc,sphs,dest_proc)
void set_communication_sphereh(E)

Parsing.c

...

Phase_change.c

...

Plates_others.c

void read_plate_margins(E)
void elements_for_plate_margins(E)
void search_other_caps(E,t,f,ip,ii)
int search_surrounding_ele(E,m,els,t,f,ii,ip)
float min_dist(E,tt,ff,ip)
float dist(tt,ff,t1,f1)
void read_super_cont_from_file(E)
double find_cont_thickness_above(E,theta,phi)

Process_buoyancy.c

void process_temp_field(E,ii)
void heat_flux(E)

Process_velocity.c

void process_new_velocity(E,ii)
void post_process(E)
void read_field_for_post_p(E)
void scale_for_post_p(E)
void process_output_field(E,ii)
void get_surface_velo(E, SV,m)
void get_ele_visc(E, EV,m)
void get_surf_stress(E,SXX,SYY,SZZ,SXY,SXZ,SZY)
void interp_stress(E,file_number,file_name,stride) // osr
void interp_surf_velocity(E)

Profiling.c

...

Shape_functions.c

...

Size_does_matter.c

...

Solver_conj_grad.c

void set_cg_defaults(E)
void cg_allocate_vars(E)
void assemble_forces_iterative(E)

Solver_multigrid.c

...

Sphere_harmonics.c

...

Sphere_related.c

void coord_of_cap(E,m,icap)
void even_divide_arc12(elx,x1,y1,z1,x2,y2,z2,theta,fi)
void rotate_mesh(E,m,icap)

Stokes_flow_Incomp.c

void solve_constrained_flow_iterative(E)
float solve_Ahat_p_fhat(E,V,P,F,impt,steps_max)
void v_from_vector(E)
void velo_from_element(E,VV,m,el,sphere_key)

Topo_gravity.c

...

Trace.c

void tracing(E,itimes_here_this_step)
void tracer_post_processing(E)
void write_radial_horizontal_averages(E)
void write_tracers_bin(E,iflag)
void write_tracers(E,iflag)
void write_compositional_field(E)
void write_compositional_field_bin(E)
void write_compositional_field2_bin(E)
void get_bulk_composition(E)
void predict_tracers(E)
void correct_tracers(E)
void get_velocity(E,j,nelem,theta,phi,rad,velocity_vector)
void gnomonic_interpolation(E,j,nelem,theta,phi,rad,velocity_vector)
void get_2dshape(E,j,nelem,u,v,iwedge,shape2d)
void get_radial_shape(E,j,nelem,rad,shaperad)
void spherical_to_uv(E,j,theta,phi,u,v)
void trace_instructions(E)
void initialize_trace(E)
void initialize_tracer_arrays(E,j,number_of_tracers)
void find_tracers(E)
void lost_souls(E)
void reduce_tracer_arrays(E)
void put_away_later(E,j,it)
void expand_later_array(E,j)
void eject_tracer(E,j,it)
void make_regular_grid(E)
int icheck_column_neighbors(E,j,nel,x,y,z,rad)
int icheck_all_columns(E,j,x,y,z,rad)
void initialize_old_composition(E)
void initialize_old_composition2(E)
void fill_composition(E)
void compute_elemental_composition_ratio_method(E)
void map_composition_to_nodes(E)
void map_composition_to_nodes2(E)
void write_trace_instructions(E)
void viscosity_checks(E)

void setup_shared_cap_information(E)
void restart_tracers_bin(E)
void restart_tracers(E)
void make_tracer_array(E)
void make_tracer_array_super_cont(E)
void read_tracer_file(E)
void cart_to_sphere(E,x,y,z,theta,phi,rad)
void sphere_to_cart(E,theta,phi,rad,x,y,z)
int icheck_that_processor_shell(E,j,nprocessor,rad)
int icheck_processor_shell(E,j,rad)
int icheck_element(E,j,nel,x,y,z,rad)
int icheck_shell(E,nel,rad)
int icheck_element_column(E,j,nel,x,y,z,rad)
int icheck_cap(E,icap,x,y,z,rad)
int icheck_bounds(E,test_point,rnode1,rnode2,rnode3,rnode4)
double findradial(E,vec,cost,sint,cosf,sinf)
void makevector(vec,xf,yf,zf,x0,y0,z0)
void crossit(cross,A,B)
void setup_dsincos(E)
void fix_radius(E,radius,theta,phi,x,y,z)
void fix_phi(phi)
void fix_theta(theta)
int iget_element(E,j,iprevious_element,x,y,z,theta,phi,rad)
int iget_radial_element(E,j,iel,rad)
int icheck_regular_neighbors(E,j,ntheta,nphi,x,y,z,theta,phi,rad)
int iquick_element_column_search(E,j,iregel,ntheta,nphi,x,y,z,theta,phi,rad,imap,ich)
int iget_regel(E,j,theta,phi,ntheta,nphi)
void debug(E,i)
void pdebug(E,i)
void cdebug(E,i)
void define_uv_space(E)
void determine_shape_coefficients(E)
void get_cartesian_velocity_field(E)
void keep_in_sphere(E,x,y,z,theta,phi,rad)
void check_sum(E)
int isum_tracers(E)
void analytical_test(E)
void analytical_runge_kutte(E,nsteps,dt,x0_s,x0_c,xf_s,xf_c,vec)
void analytical_test_function(E,theta,phi,rad,vel_s,vel_c)
void trace_filter(E)
void read_comp(E)
void read_comp_bin(E)
void read_comp2_bin(E)
void predict_tracers_4rk(E)
void correct_tracers_4rk(E)

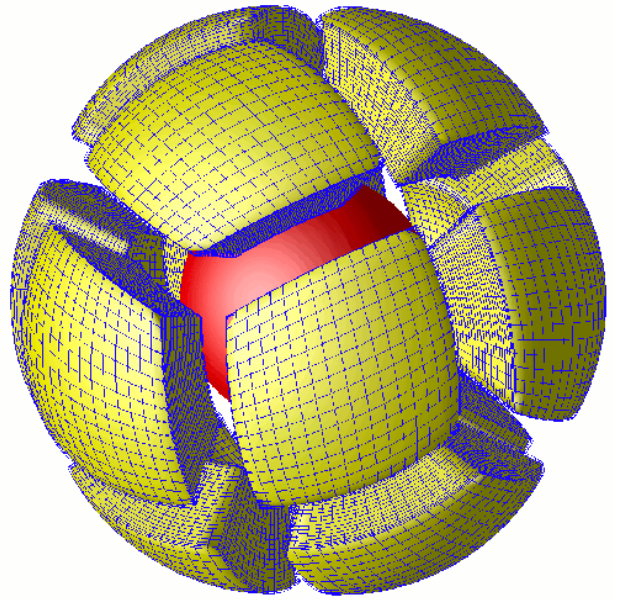
Viscosity_structures.c

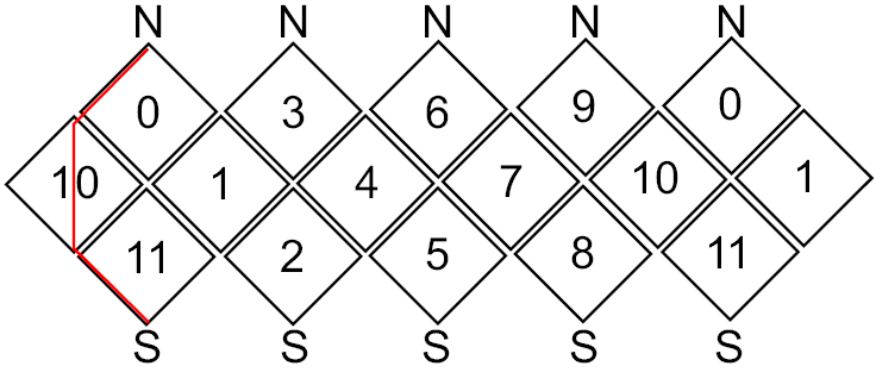
void get_viscosity_option(E)
void viscosity_for_system(E)
void get_system_viscosity(E,propogate,evisc,visc)
void visc_from_mat(E,EEta)
void visc_from_T(E,EEta,propogate)
void visc_from_S(E,EEta,propogate)

void get_compositional_viscosity_prefactor(E)
void visc_to_node_interpolate(E,evisc,visc)
void strain_rate_2_inv(E,m,EEDOT,SQRT)

Mesh

- 12 cells of spherical surface, 2 layers in radial direction, i.e. 24 domains (we use 24 processors)
  (that is Shijie's group setup; CIG permits further horizontal and radial division)
- usually each of 12 caps divided in 48x48x48 elements, that is 12x48x48x48=1327104 or **~1.3M elements total** (the **computing domain** has 48x48x24=55295 or **~55k elements**)
- non-orthogonal mesh: approx rectangular & uniform-size cells
- radial grid spacing can be variable (*e.g.*, finer in boundary layers)
- parallel computing natural *(24 processors in shijie's group version; arbitrary in CIG)*





Common points coordinates:

	theta	phi
0-3-6-9	0.47	0
0-1-3	55	89.67
3-4-6	54.53	180
6-7-9	55	270.33
0-9-10	55.47	0
0-1-10-11	90.33	45
1-2-3-4	89.67	135
4-5-6-7	89.67	225

7-8-9-10	90.33	315
1-2-11	125	90.33
2-4-5	124.53	180
5-7-8	125	269.67
8-10-11	125.47	0
2-5-8-11	179.53	180

Functions that set up the mesh located in:

Nodal_mesh.c	`void node_locations(E)`	main mesh function
Geometry_cartesian.c	`void set_3dsphere_defaults(E)`	4 corners of each cap
Sphere_related.c	`void coord_of_cap(E,m,icap)`	local xyz cartesian for cap

`E->SX[lev][m][1][node] = SX[0][nodes];`

lev...MG level; m...#caps/processor; 1=υ, 2=φ, 3=r; node#w/in cap

AKM_additions.c `void setup_radial_mesh(E,rr)`

grid_top_boundary

grid_node_boundary

Mesh-related input & derived parameters

from running the inital functions of Citcom:

```
E->parallel.nproc                                   24
E->parallel.nprocx                                  1
E->parallel.nprocy                                  1
E->parallel.nprocz                                  2
E->parallel.nprocxy                                 12
E->parallel.nproczy                                 0
E->parallel.nprocxz                                 0

E->sphere.caps                                      12
E->sphere.nox                                       181
E->sphere.noy                                       361
E->sphere.elx = sphere.nox – 1                      180
E->sphere.ely = sphere.noy – 1                      360
E->sphere.snel = sphere.ely* sphere.elx             64800
E->sphere.nsf = sphere.noy * sphere.nox             65341

E->mesh.nsd                                         3
E->mesh.dof                                         3
E->mesh.mgunitx                                     6
E->mesh.mgunity                                     6
E->mesh.mgunitz                                     6
E->mesh.levels                                      4
E->mesh.levmin                                      0
E->mesh.levmax = mesh.levels-1                      3
E->mesh.gridmin                                     0
E->mesh.gridmax = mesh.levmax                       3
E->mesh.nox = mesh.mgunitx * 2**(mesh.levmax) + 1   49
E->mesh.noy = mesh.mgunity * 2**(mesh.levmax) + 1   49
E->mesh.noz = mesh.mgunitz * 2**(mesh.levmax) + 1   49
E->mesh.nnx[1] = mesh.nox                           49
E->mesh.nnx[2] = mesh.noy                           49
E->mesh.nnx[3] = mesh.noz                           49
E->mesh.elx = mesh.nox-1                            48
```

```
E->mesh.ely = mesh.noy-1                                                48
E->mesh.elz = mesh.noz-1                                                48
E->mesh.nno = sphere.caps * mesh.nnx[1] * mesh.nnx[2] * mesh.nnx[3]     1354752
E->mesh.nel = sphere.caps * mesh.elx * mesh.elz * mesh.ely              1327104
E->mesh.nnov = mesh.nno                                                 1411788
E->mesh.neq = mesh.nnov * mesh.nsd                                      4064256
E->mesh.npno = mesh.nel                                                 1327104
E->mesh.nsf = mesh.nox * mesh.noy                                       2401
E->mesh.nxs                                                             1
E->mesh.nys                                                             1
E->mesh.nzs                                                             1
from mesh.levmin to mesh.levmax
      E->mesh.ELX[i]                                6,12,24,48
      E->mesh.ELY[i]                                6,12,24,48
      E->mesh.ELZ[i]                                6,12,24,48
      E->mesh.NNO[i]                                4116,26364,187500,1411788
      E->mesh.NEL[i]                                2592 ,20736,165888 ,1327104
      E->mesh.NPNO[i]                               2592 ,20736,165888 ,1327104
      E->mesh.NOX[i]                                7,13,25,49
      E->mesh.NOZ[i]                                7,13,25,49
      E->mesh.NOY[i]                                7,13,25,49
      E->mesh.NNOV[i]                               4116,26364,187500,1411788
      E->mesh.NEQ[i]                                12348 ,79092 ,562500 ,4235364

E->lmesh.elx = mesh.elx / parallel.nprocx                               48
E->lmesh.ely = mesh.ely / parallel.nprocy                               48
E->lmesh.elz = mesh.elz / parallel.nprocz                               24
E->lmesh.nox = lmesh.elx + 1                                            49
E->lmesh.noy = lmesh.ely + 1                                            49
E->lmesh.noz = lmesh.elz + 1                                            25
E->lmesh.exs = parallel.me_loc[1] * lmesh.elx                           0
E->lmesh.eys = parallel.me_loc[2] * lmesh.ely                           0
E->lmesh.ezs = parallel.me_loc[3] * lmesh.elz                           0
E->lmesh.nxs = parallel.me_loc[1] * lmesh.elx + 1                       1
E->lmesh.nys = parallel.me_loc[2] * lmesh.ely + 1                       1
E->lmesh.nzs = parallel.me_loc[3] * lmesh.elz + 1                       1
E->lmesh.nno = lmesh.noz*E->lmesh.nox * lmesh.noy                       60025
E->lmesh.nnov = lmesh.nno                                               60025
E->lmesh.nel = lmesh.ely*E->lmesh.elx * lmesh.elz                       55296
E->lmesh.npno = lmesh.nel                                               55296
E->lmesh.nsf = lmesh.nno /  lmesh.noz                                   2401
E->lmesh.snel = lmesh.elx * lmesh.ely                                   2304
at each multigrid level (i goes from mesh.levmin to mesh.levmax):
      E->lmesh.ELX[i]
      E->lmesh.ELY[i]
      E->lmesh.ELZ[i]
      E->lmesh.NOZ[i]
      E->lmesh.NOY[i]
      E->lmesh.NOX[i]
      E->lmesh.NNO[i]
      E->lmesh.NNOV[i]
      E->lmesh.SNEL[i]
      E->lmesh.NEL[i]
      E->lmesh.NPNO[i]
      E->lmesh.NEQ[i]
      E->lmesh.EXS[i]
      E->lmesh.EYS[i]
      E->lmesh.EZS[i]
      E->lmesh.NXS[i]
      E->lmesh.NYS[i]
      E->lmesh.NZS[i]
```

Array ranks & sizes:

```
lev         mesh.levmin ... mesh.levmax
m           1 ... sphere.caps_per_proc
d           1 ... mesh.nsd (=3)
node        1 ... lmesh.nnov = lmesh.nno
nodep       1 ... lmesh.npno = lmesh.nel
snode       1 ... lmesh.nsf
nodez       1 ... lmesh.noz
scomp       1 ... 14 * lmesh.nsf
glev        mesh.gridmin ... mesh.gridmax
NODE        1 ... lmesh.NNO[glev]
```

in allocate_common_vars

```
E->P[m][nodep]                double      ... pressure
E->T[m][node]                 double      ... temperature
E->NP[m][node]                float
E->edot[m][node]              float
E->Fas670[m][node]            float
E->Fas670_b[m][snode]         float
E->stress[m][scomp]           float
E->slice.tpg[m][snode]        float
E->slice.tpgb[m][snode]       float
E->slice.divg[m][snode]       float
E->slice.vort[m][snode]       float
E->slice.shflux[m][snode]     float
E->slice.bhflux[m][snode]     float
E->Have.T[nodez]              float
E->Have.Vi[nodez]             float
E->Have.V[1:7][nodez]         float
E->X[lev][m][d][node]         double      ... global xyz-coordinates
      Note:       E->x[m][d][node] = E->X[E->mesh.levmax][m][d][node];
E->SX[lev][m][d][node]        double      ... global rυφ-coordinates
      Note:       E->sx[m][d][node] = E->SX[E->mesh.levmax][m][d][node];
E->VI[glev][m][NODE[glev]]    float
```

+ **all those finite elemental arrays...**

```
E->ECO[glev][m][node].centre[k]    double      ...element center rυφ-coordinates
E->ECO[glev][m][node].area         double      ...element volume
E->ECO[glev][m][node].size[k]      double      ...element sizes (3 directions)
E->eco[m] = E->ECO[maxlevel][m]    eco = ECO at finest grid level
```

in allocate_velocity_vars

```
E->temp[m][d][node]           double
E->temp1[m][d][node]          double
E->F[m][d][node]              double
E->U[m][d][node]              double
E->heating_adi[m][nodep]      float
E->heating_visc[m][nodep]     float
E->sphere.cap[m].V[d][node]   float       ... global rυφ-velocity
E->sphere.cap[m].VB[d][node]  float
```

in allocate_velocity_vars

```
E->x[m][i] = E->X[E->mesh.levmax][m][i]
```

tracing

```
E->trace.comp_node[m][node]   double
E->trace.comp_el[m][nodep]    double
E->trace.Have_C[nodez]        double
```

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 `visc_from_T` in file '*Viscosity_Structure.c*'
`VISC_UPDATE=on`	viscosity is updated (every other timestep)
`visc_smooth_method=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 profle $\eta_{ref}(r)$ (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*)

`cluster.py` script

cluster.py -h	displays help
cluster.py -n MarsTC10 -m 25-36	creates directory ~/MarsTC10 on computer nodes 25-36...

Execution

```
mpirun [mpirun_options...] <progname> [options...]
```

mpirun options:

`-machinefile <machine-file name>`	list of machines to run on in file <machine-file name>
`-np <np>`	number of processors to run on
`-nolocal`	don't run on the local machine
`-t`	testing - do not run, just print what would be executed
`-v`	verbose - throw in some comments

for example:

```
mpirun -np 24 -nolocal -machinefile mc1 ../TCsphere.mpi inputTC10 <junk &
```

`mc1`	machines file
`Tcsphere.mpi`	executable
`inputTC10`	input file

Execution speed

machine	*case*	*problem*	*grid*	*time / 10k*	*steps / day*
nappo		thermal	48x48x48		7.3 k
cappo	TC10	thermal	48x48x48	13.6 h	17.6 k

cappo	TC10A	thermochem	48x48x48	31.8 h	7.5 k
cappo	ROL01	thermal	48x48x64	15.8 h	15.2 k
cappo	ROL01A	thermal	48x48x64	62.4 h	3.8 k

Input file

input file (*e.g.*, `inputTC10`)

```
use_scratch="sramek"
datafile="MarsTC10/a"
maxstep=10000                 # max velocity iterations
storage_spacing=200           # write data
restart=0                     # 0 or framenumber
background_profile=1          #(0)conductive, (1) from file
background_profile_file="initial_profiles/TC20.profile"
...
radius_inner=0.485294
radius_outer=1.0
rayleigh=1.25e8               # Rayleigh number
Q0=74e-9                      # Dimensionless internal heating rate
adi_heating=1
visc_heating=1
...
Viscosity=system
rheol=3 TDEPV=on              # 1: N=N0*exp(-CT) , 2: N=N0*exp((E+Pz)/(T-T0))
VISC_UPDATE=on
visc_smooth_method=1
...
```

initial profiles (eg, TC20.profile)

radius temp Uave Uave visc

Ouput files

Notation/assumptions:

'a'	chosen file prefix
'nn'	processor number
'ff'	frame number
'MarsTC10'	data

if tracing composition (color coded)

in the execution directory:

`coord2.0 .. coord2.23`

written in *Nodal_mesh.c*, `void node_locations(E)`, line 198

υ φ r node coordinates ascii, 49x49x25=60025 entries

`E->sx[1][1][node]  E->sx[1][2][node]  E->sx[1][3][node]`

`coord_bin.0 .. coord_bin.23`

written in *Output.c*, `void output_velo_related_bin(E,file_number)`, line 214

all υ all φ all r node coordinates binary

`&(E->lmesh.nno)  E->sx[j][1][.]  E->sx[j][3][.]  E->sx[j][3][.]`

in data directory on the main cluster node (eg nappo):

`a.log` written in several places

opened in *Instructions.c*, `void global_derived_values(E)`, line 636
Stokes solver info (+ first couple lines viscosity info)

`a.info0 .. a.info23` written in *Instructions.c*, opened line 850
info on radial discretization, conductivity, expansivity,
initial T profiles; bottom/surface heat flow evolution in 0&1

in data directory on the first computer node:

Caution: time series files are rewritten upon restart !

`a.average_temperature.data`
`a.vrms.data`
`a.log_averaged_viscosity.data` ~~...disfunctional~~ corrected 19 Nov 2008
`a.cpu_time.data`
written in *AKM_additions.c*, `void write_bulk_data_to_files(E)`, line17, uses type `FILE`
called from *Citcom.c*, lines 100 & 155
time ... written every step
`E->monitor.elapsed_time average_temperature`
`E->monitor.elapsed_time average_vrms`
`E->monitor.elapsed_time log_averaged_viscosity`
`E->monitor.elapsed_time time4`

`a.botq.data`
`a.surfq.data`
written in *AKM_additions.c*, `void write_heat_flux(E,my_surfq,my_botq)`, l.717, uses type `FILE`
called from *Process_buoyancy.c*, line 196
time ..heatflux written every 20th step
`E->monitor.elapsed_time botq`
`E->monitor.elapsed_time surfq`

`a.surf_vrms.data`
`a.surf_vtheta.data`
`a.surf_vphi.data`
written in *AKM_additions.c*, `void write_surface_vrms(E,Have_V,Have_Vtheta,Have_Vphi)`, line 772
called from *Topo_gravity.c*, line 125
time surfvel... written every frame save
`E->monitor.elapsed_time surf_vrms`
`E->monitor.elapsed_time surf_vtheta`
`E->monitor.elapsed_time surf_vphi`

`a.tpgb_intp`
phi theta topo 182x360=65341 entries
`a.tpgb_sharm`
lmaxx=... lminx=... for tpgb header line 1
ll mm cos sin header line 2
ll mm cfcos cfsin (llmax+1)*(llmax+2)/2 entries (496 for llmax=30)
written in *Output.c*, `void print_field_spectral_regular(E,TG,sphc,sphs,proc_loc,filen)`, line 371
called from *Sphere_harmonics.c*, line 137
which is itself called from *Process_velocity.c*, lines 52 & 97

`a.ave_r.0.ff`
written in *Output.c*, `void output_velo_related(E,file_number)`, line 175 (or binary line 321)
r T V_r (?) V_{horiz} η horizontal averages, 1 entry per layer
`E->sx[1][3][j]` `E->Have.T[j]` `E->Have.V[1][j]` `E->Have.V[2][j]` `E->Have.Vi[j]`

`a.power_r.0.ff`
written in *Process_velocity.c*, `void process_output_field(E,ii)`, line 262
3 columns: ll r ?? (llmax+1)*(nlayer+1) entries (each degree in each layer)

```
ll  E->sphere.R[lev][i]  power[ll][i]
```

a.rotation.0.ff

written in *AKM_additions.c*, `void get_net_rotation(E)`, line 2641

r ?? ?? ?? nlayer entries

```
E->eco[1][i].centre[3]  exyz1[i]  exyz2[i]  exyz3[i]
```

a.composition.data

a.error_fraction.data

a.ave_c.0.ff

a.ave_tracers.0.ff

in data directory on the second computer node:

a.tpgt_intp — see a.tpgb_intp above

a.tpgt_sharm — see a.tpgb_sharm above

a.ave_r.1.ff — see a.ave_r.0.ff above

a.power_r.1.ff — see a.power_r.0.ff above

a.rotation.1.ff — see a.rotation.0.ff above

a.surfv_intp.ff

written in *Process_velocity.c*, `void interp_surf_velocity(E)`, line 421

also in *AKM_additions.c*, `void runtime_gmt_output(E)`, line 2257

φ υ ?? ?? 37x73=2701 entries (5x5 degree mesh)

```
f  t  TG[1][node]  -1.0*TG[0][node]
```

a.ave_c.1.ff

a.ave_tracers.1.ff

in data directories on odd# nodes:

a.botm.nn.ff

written in *Output.c*, `void output_velo_related(E,file_number)`, line 164 (or binary line 307)

1 2401 header line

.. .. 49x49=2401 entries

```
E->slice.tpgb[j][i]  E->slice.bhflux[j][i]  E->sphere.cap[j].V[1][s]  E->sphere.cap[j].V[2][s]
```

in data directories on even# nodes:

a.surf.nn.ff

written in *Output.c*, `void output_velo_related(E,file_number)`, line 150 (or binary line 288)

1 2401 header line

.. .. 49x49=2401 entries

```
E->slice.tpg[j][i]  E->slice.shflux[j][i]  E->sphere.cap[j].V[1][s]  E->sphere.cap[j].V[2][s]
       E->slice.divg[j][i]  E->slice.vort[j][i]$
```

a.stress_bin.nn.ff

in data directories on all computing nodes:

2 different nn's on each computer node

a.velo.nn.ff

written in *Output.c*, `void output_velo_related(E,file_number)`, line 95 (or binary line 257)

100 60025 4.59617e-05 header line 1
1 60025 header line 1
Temp 49x49x(nlayers+1)=60025 entries

`a.heating.nn`

written in *Advection_diffusion.c*, `void process_heating(E)`, line 759
QQ 1.25e+08 0.211367 52.9461 0.111111 header line
index Qadi Qvisc 48x48*24=55296 entries
`e E->heating_adi[m][e] E->heating_visc[m][e]`

`a.UP.nn`

written in *Output.c*, `void output_velo_related(E,file_number)`, line 136 (or binary line 271)
file# meshsize time header line 1
1 60025 header line 2
Vth Vph Vr 49x49x25=60025 entries
Press 48x48*24=55296 entries
`E->U[j][E->id[j][i].doff[1]] E->U[j][E->id[j][i].doff[2]] E->U[j][E->id[j][i].doff[3]] E->P[j][i]`

`a.visc.nn.0`

written in *Output.c*, `void output_velo_related(E,file_number)`, line 60 (or binary line 236)
file# meshsize time header line 1
1 60025 header line 2
Visc 49x49x25=60025 entries
`E->VI[E->mesh.levmax][j][i]`

`a.comp_bin.nn.ff`

`a.OLDCOMP.nn.ff` saved every *tracersave*

`a.OLDCOMP_bin.nn` saved every 10**framesave*

`a.tracers.nn.ff`

`a.tracer_log.nn`

File output code flow

main {
...
read_instructions → initialize_trace → tracer_post_processing
...
process_new_velocity → output_velo_related_bin
write_bulk_data_to_files
Iteration
process_heating
E.monitor.solution_cycles++
E.next_buoyancy_field
tracing(1) → tracer_post_processing
...
process_new_velocity → output_velo_related_bin
tracing(2) → tracer_post_processing
write_bulk_data_to_files
...
End iteration
...
}

read_instructions → initialize_trace OR tracing → tracer_post_processing

a.ave_tracers.nn.ff → write_radial_horizontal_averages
a.ave_c.nn.ff → "
a.comp_bin.nn.ff → write_compositional_field_bin
a.OLDCOMP_bin.nn → "
a.OLDCOMP_bin.nn.ff → "
a.tracers.nn → write_tracers_bin
a.tracers.nn.ff → "
a.error_fraction.data
a.composition.data
a.tracer_log (from many functions in Trace.c)

process_new_velocity → output_velo_related_bin

a.power_r.nn.ff → process_output_field
a.rotation.nn.ff → get_net_rotation
a.surf_vrms.data → get_STD_topo1 → write_surface_vrms
a.surf_vtheta.data → " "
a.surf_vphi.data → " "
a.tpgb_intp → sphere_harmonics_layer
a.tpgt_intp → "
a.visc.nn.ff → output_velo_related_bin
a.velo_bin.nn.ff → "
a.UP_bin.nn → "
a.stress_bin.nn.ff → "
a.surf_bin.nn.ff → "
a.ave_r.nn → "

write_bulk_data_to_files

a.average_temperature.data
a.vrms.data
a.log_averaged_viscosity.data
a.cpu_time.data

process_heating

a.heating.nn

next_buoyancy_field =PG_timestep) → CBF_heat_flux → write_heat_flux

a.botq.data
a.surfq.data

Data processing

spectrum analyses

post_p1.x runfile_deg

'post_p1.c'

'runfile_deg':

MarsTC10/a	... location of output files on main cpu + filename tag
MarsTC10/a	... location of data files on nodes + filename tag
25	... computer node containing power data (1st or 2nd ndoe)
49 49 25	... evaluation points in each of bottom 12 blocks
12 2	... CitcomS data in 12 horiz blocks, 2 layers
200 10000 30000 5	... framestep firstframe lastframe radiallevel

0

files for Data Explorer

directory DATA

```
Images.c
Others.c
make images.x
runfile_PL_TC
images.x runfile_PL_TC
```

input file '`runfile_PL_TC`':

`/home/sramek/Mars_CitcomS_TC/DATA/coord_bin`	location of coordinate files
`sramek/MarsTC10/b`	location of data files on nodes + filename tag
`/home/sramek/Mars_CitcomS_TC/DATA/f1`	composition files to generate
`/home/sramek/Mars_CitcomS_TC/DATA/t1`	temperature files to generate
`49 49 49`	evaluation points in each of 12 blocks (?)
`12 2`	CitcomS data in 12 horiz blocks, 2 layers
`1 0 1 1000 0`	get DeltaT, get comp, binary?, frame, skipn(?)
`25`	full ordered list of computer nodes
`26`	-"-
`27`	-"-
`..`	...

Data Explorer (dx)

dx -> Run Visual Programs -> isosurf_new_PL.net

```
bottom_surf
bottom_surf.earth
bottom_surf.general
isosurf_new_PL.net
isosurf_new_PL.cfg
super_cont_config10A.dat
temp.macro.net
```

Plume–cap separation

Run e.g.

`$ plume_location.x runfile_plume_location1`	creates e.g. f01A_plume28
`$ plume_location.x runfile_keel_location1`	creates e.g. f01A_slab53
`$ combine_plume_slab_center.x f01A_plume28 f01A_slab53`	calculates angular separation

Files from Shijie. I do not have the source code for `plume_location`, only the executable.

Temperature (and composition) movies

Using the following files from Nan:

`make_movie_GMT`	shell script
`extractlayer.c`	prepares data for image of a layer
`update_runfile.x`	prepares runfile for dx data preparation