STRUCTURE OF GOMA

WESTON ORTIZ

DIRECTORY LAYOUT

```
|-- cmake                   # CMake modules
|-- CMakeLists.txt          # CMake configuration
|-- docs                    # docs, has some undocumented things
|-- include                 # Include files in here
|   |-- ac_conti.h
|   |-- ...
|   |-- brkfix
|   |   |-- bbb.h
|   |   |-- ...
|-- LICENSE                 # Open source license
|-- scripts                 # build scripts, some postproc utilities
|-- src                     # Source files (c/cpp/fortran files)
    |-- ac_conti.c
    |-- ...
    |-- brkfix
    |   |-- bbb.c
    |   |-- ...
```

FILE NAMING CONVENTION

- `ac_` Augmenting condition
- `bc_` Boundary condition
- `dp_` Distributed Processing
- `el_` element (num nodes, gauss points, etc...)
- `loca_` builtin loca library
- `mm_` Moving mesh (more of a catch all)
- `rd_/wr_` I/O read/write
- `rf_` Reacting flow (historical name from `rf_salsa`, which Goma was forked)
- `sl_` Interfaces to solvers like Trilinos
- `user_` user routines, less used now

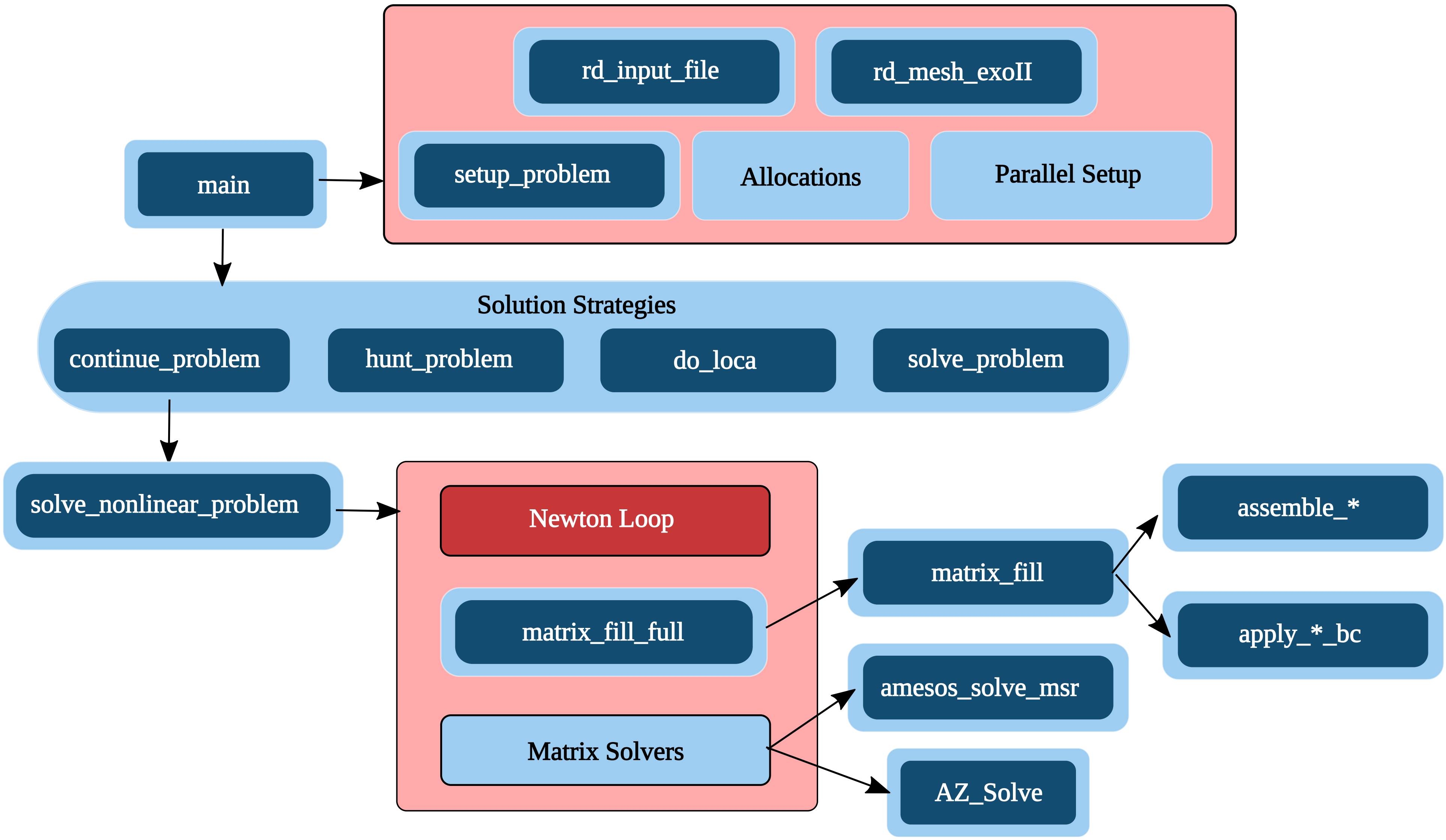
WHERE TO LOOK FOR FILES

- `mm_input*` Input for equations,BCs,materials
- `mm_fill_*` Assembly functions (Equations)
- `rf_*` Older basic finite element routines like shape functions
- `rf_solve.c` Steady and Transient solver

FILES WITH MODELS

- `mm_post_proc.c` Post Processing
- `mm_fill_terms.c` Many assembly functions
- `mm_viscosity.c` Viscosity models
- `mm_std_models.c` Many source models
- `mm_flux.c` Flux and Volume integration
- `mm_fill_species.c` Species BCs and equation
- `mm_ns_bc.c` Integrated BCs

FLOW OF GOMA



rd_input_file
rd_mesh_exoII
main
setup_problem
Allocations
Parallel Setup
Solution Strategies
continue_problem
hunt_problem
do_loca
solve_problem
solve_nonlinear_problem
Newton Loop
matrix_fill_full
Matrix Solvers
assemble_*
matrix_fill
apply_*_bc
amesos_solve_msr
AZ_Solve

TOOLS FOR EXPLORING GOMA

The number one tool you want to have is some kind of global search

- QtCreator / VSCode / IDE's generally will have this built-in
- Ripgrep is usually faster than grep and has nice defaults and output (https://github.com/BurntSushi/ripgrep)
- If none of those are available `git-grep` is your friend (https://git-scm.com/docs/git-grep)
- silver searcher, ack, other grep like tools

GREP EXAMPLE

```
$ grep -R "fluid_stress"
$ git grep "fluid_stress"
$ rg "fluid_stress"
```

FUNCTION REFERENCE OF GOMA

FUNCTION REFERENCE OF GOMA: I/O

- I/O and General Setup:
  - `translate_command_line()`: command line arguments like `-a`, `-i`, `-brk`
  - `pd_alloc(),...*_alloc`: global allocations....
  - `read_input_file()`: Reads the input file (problem setup, enabled equations, time stepping)
    - `rd_bc_specs()`: Read boundary conditions
    - `rd_mp_specs()`: Read material properties
    - `rd_post_process_specs()`: Read Post Processing
    - Quite a few more take a look at read_input_file

FUNCTION REFERENCE OF GOMA: PARALLEL

- `noahs_raven()` Share some initial information with other processors
- `raven_landing()` Using above allocate structures on all processors
- `noahs_ark()` Share information from Proc 0 to all other processors
- `ark_landing()` Allocate space for structures based on above
- `noahs_dove` send information from Proc 0 to all other processors

FUNCTION REFERENCE OF GOMA: SOLUTION PROCESS

- `solve_problem()` Transient and Steady state solution
- `continue_problem()` Generic Continuation
- `hunt_problem()` Hunting continuation
- `do_loca()` Solutions using built-in loca library

FUNCTION REFERENCE OF GOMA: SPARSE MATRICES

Usually setup in Solution Process functions like `solve_problem`

- `alloc_MSR_sparse_arrays()` MSR matrix
- `alloc_VBR_sparse_arrays()` VBR matrix
- `EpetraCreateRowMatrix()` Epetra CSR
- `EpetraCreateGomaProblemGraph()` Epetra problem graph

FUNCTION REFERENCE OF GOMA: SOLUTION VECTORS

Usually setup in Solution Process functions like `solve_problem`

- `x` Solution Vector
- `x_old` Previous solution
- `xdot` Time derivative
- `x_older` Previous Previous solution

FUNCTION REFERENCE OF GOMA: ADAPTIVE TIMESTEPPING

In `solve_problem()`

- `predict_solution()` Update initial guess for next time step with explicit step, setup xdot
- `time_step_control()` Restrictions on time step size based on predicted solution

FUNCTION REFERENCE OF GOMA: NONLINEAR SOLVE

`solve_nonlinear_problem` Solves the nonlinear problem using Newton's method, in here we have a newton loop and call assembly and solve functions

$$\mathbf{J}(x,\dot{x})\Delta x = -\mathbf{R}(x,\dot{x})$$
$$x = x_{old} + \Delta x$$

FUNCTION REFERENCE OF GOMA: ASSEMBLY PROCESS

In `solve_nonlinear_problem()`

- `matrix_fill_full()` Element Loop
  - `matrix_fill()` Element contributions, contains almost everything used for the finite element assembly
    - loading basis functions
    - loading field variables
    - element residuals and Jacobians
    - boundary conditions

FUNCTION REFERENCE OF GOMA: FILL DEEP DIVE

- `matrix_fill()` Element contributions
  - `load_elem_dofptr()` loads `esp` from solution vectors
    - `load_ei`
      loads `ei` which contains DOF info and element information

FUNCTION REFERENCE OF GOMA: FILL DEEP DIVE

In `matrix_fill()`

- Gauss Loop
  - `load_basis_functions()` loads basis functions ϕ_i, $\frac{d\phi_i}{d\xi_d}$
  - `beer_belly()` loads element Jacobians, J_{el}, $|J_{el}|$, B
  - `load_fv()` load field variables $u = \sum_j \phi_j u_j$
  - `load_bf_grad()` loads basis gradients $\nabla \phi_i$
  - `load_fv_grads()` load field gradients $\nabla u = \sum_j \nabla \phi_j u_j$

FUNCTION REFERENCE OF GOMA: FILL DEEP DIVE

In `matrix_fill()`

- Gauss Loop continued
  - `assemble_*()` Compute Element Residual and Jacobian for equation
  - `assemble_momentum()` NS Momentum
  - `assemble_continuity()` NS Continuity
  - `assemble_energy()` Energy equation
  - `assemble_mass_transport()` Species equations

FUNCTION REFERENCE OF GOMA: FILL DEEP DIVE

In `matrix_fill()`

- After Gauss loop (highlights)
  - `apply_embedded_bc()` LS boundary conditions
  - `apply_integrated_bc()` Strong and Weak integrated BCs
  - `apply_point_colloc_bc()` Collocated BCs (at nodes)
  - `put_dirichlet_in_matrix()`Zero rows and puts 1 on diagonal for Dirichlet

FUNCTION REFERENCE OF GOMA: FILL DEEP DIVE

In `matrix_fill()`

- After Gauss loop (highlights)
  - `load_lec()` Local element contributions into global: $J(global_i, global_j) += lec_J(i, j)$, $R(global_i) += lec_R(i)$