

# Ecosystem

Version 0.0 beta

Generated by Doxygen 1.8.3.1

Thu Sep 24 2015 13:36:56



# Contents

|          |  |          |
|----------|--|----------|
| <b>1</b> | <b>Hierarchical Index</b>                        | <b>1</b> |
| 1.1      | Class Hierarchy . . . . .                        | 1        |
| <b>2</b> | <b>Class Index</b>                               | <b>3</b> |
| 2.1      | Class List . . . . .                             | 3        |
| <b>3</b> | <b>File Index</b>                                | <b>5</b> |
| 3.1      | File List . . . . .                              | 5        |
| <b>4</b> | <b>Class Documentation</b>                       | <b>7</b> |
| 4.1      | ARNOLDI_DATA Struct Reference . . . . .          | 7        |
| 4.1.1    | Detailed Description . . . . .                   | 8        |
| 4.1.2    | Member Data Documentation . . . . .              | 8        |
| 4.1.2.1  | k . . . . .                                      | 8        |
| 4.1.2.2  | iter . . . . .                                   | 8        |
| 4.1.2.3  | beta . . . . .                                   | 8        |
| 4.1.2.4  | hp1 . . . . .                                    | 8        |
| 4.1.2.5  | Output . . . . .                                 | 8        |
| 4.1.2.6  | Vk . . . . .                                     | 8        |
| 4.1.2.7  | Hkp1 . . . . .                                   | 8        |
| 4.1.2.8  | yk . . . . .                                     | 8        |
| 4.1.2.9  | e1 . . . . .                                     | 8        |
| 4.1.2.10 | w . . . . .                                      | 8        |
| 4.1.2.11 | v . . . . .                                      | 9        |
| 4.1.2.12 | sum . . . . .                                    | 9        |
| 4.2      | Atom Class Reference . . . . .                   | 9        |
| 4.2.1    | Detailed Description . . . . .                   | 11       |
| 4.2.2    | Constructor & Destructor Documentation . . . . . | 11       |
| 4.2.2.1  | Atom . . . . .                                   | 11       |
| 4.2.2.2  | ~Atom . . . . .                                  | 11       |
| 4.2.2.3  | Atom . . . . .                                   | 11       |
| 4.2.2.4  | Atom . . . . .                                   | 11       |

|          |                                 |    |
|----------|---------------------------------|----|
| 4.2.3    | Member Function Documentation   | 11 |
| 4.2.3.1  | Register                        | 11 |
| 4.2.3.2  | Register                        | 11 |
| 4.2.3.3  | editAtomicWeight                | 11 |
| 4.2.3.4  | editOxidationState              | 11 |
| 4.2.3.5  | editProtons                     | 11 |
| 4.2.3.6  | editNeutrons                    | 11 |
| 4.2.3.7  | editElectrons                   | 12 |
| 4.2.3.8  | editValence                     | 12 |
| 4.2.3.9  | removeProton                    | 12 |
| 4.2.3.10 | removeNeutron                   | 12 |
| 4.2.3.11 | removeElectron                  | 12 |
| 4.2.3.12 | AtomicWeight                    | 12 |
| 4.2.3.13 | OxidationState                  | 12 |
| 4.2.3.14 | Protons                         | 12 |
| 4.2.3.15 | Neutrons                        | 12 |
| 4.2.3.16 | Electrons                       | 12 |
| 4.2.3.17 | BondingElectrons                | 12 |
| 4.2.3.18 | AtomName                        | 12 |
| 4.2.3.19 | AtomSymbol                      | 13 |
| 4.2.3.20 | AtomCategory                    | 13 |
| 4.2.3.21 | AtomState                       | 13 |
| 4.2.3.22 | AtomicNumber                    | 13 |
| 4.2.3.23 | DisplayInfo                     | 13 |
| 4.2.4    | Member Data Documentation       | 13 |
| 4.2.4.1  | atomic_weight                   | 13 |
| 4.2.4.2  | oxidation_state                 | 13 |
| 4.2.4.3  | protons                         | 13 |
| 4.2.4.4  | neutrons                        | 13 |
| 4.2.4.5  | electrons                       | 13 |
| 4.2.4.6  | valence_e                       | 13 |
| 4.2.4.7  | Name                            | 13 |
| 4.2.4.8  | Symbol                          | 14 |
| 4.2.4.9  | Category                        | 14 |
| 4.2.4.10 | NaturalState                    | 14 |
| 4.2.4.11 | atomic_number                   | 14 |
| 4.3      | BACKTRACK_DATA Struct Reference | 14 |
| 4.3.1    | Detailed Description            | 14 |
| 4.3.2    | Member Data Documentation       | 15 |
| 4.3.2.1  | alpha                           | 15 |

|          |                                |    |
|----------|--------------------------------|----|
| 4.3.2.2  | rho                            | 15 |
| 4.3.2.3  | lambdaMin                      | 15 |
| 4.3.2.4  | normFkp1                       | 15 |
| 4.3.2.5  | constRho                       | 15 |
| 4.3.2.6  | Fk                             | 15 |
| 4.3.2.7  | xk                             | 15 |
| 4.4      | BiCGSTAB_DATA Struct Reference | 15 |
| 4.4.1    | Detailed Description           | 16 |
| 4.4.2    | Member Data Documentation      | 17 |
| 4.4.2.1  | maxit                          | 17 |
| 4.4.2.2  | iter                           | 17 |
| 4.4.2.3  | breakdown                      | 17 |
| 4.4.2.4  | alpha                          | 17 |
| 4.4.2.5  | beta                           | 17 |
| 4.4.2.6  | rho                            | 17 |
| 4.4.2.7  | rho_old                        | 17 |
| 4.4.2.8  | omega                          | 17 |
| 4.4.2.9  | omega_old                      | 17 |
| 4.4.2.10 | tol_rel                        | 17 |
| 4.4.2.11 | tol_abs                        | 17 |
| 4.4.2.12 | res                            | 17 |
| 4.4.2.13 | relres                         | 18 |
| 4.4.2.14 | relres_base                    | 18 |
| 4.4.2.15 | bestres                        | 18 |
| 4.4.2.16 | Output                         | 18 |
| 4.4.2.17 | x                              | 18 |
| 4.4.2.18 | bestx                          | 18 |
| 4.4.2.19 | r                              | 18 |
| 4.4.2.20 | r0                             | 18 |
| 4.4.2.21 | v                              | 18 |
| 4.4.2.22 | p                              | 18 |
| 4.4.2.23 | y                              | 18 |
| 4.4.2.24 | s                              | 18 |
| 4.4.2.25 | z                              | 19 |
| 4.4.2.26 | t                              | 19 |
| 4.5      | CGS_DATA Struct Reference      | 19 |
| 4.5.1    | Detailed Description           | 20 |
| 4.5.2    | Member Data Documentation      | 20 |
| 4.5.2.1  | maxit                          | 20 |
| 4.5.2.2  | iter                           | 20 |

|          |  |    |
|----------|--|----|
| 4.5.2.3  | breakdown                              | 20 |
| 4.5.2.4  | alpha                                  | 20 |
| 4.5.2.5  | beta                                   | 20 |
| 4.5.2.6  | rho                                    | 20 |
| 4.5.2.7  | sigma                                  | 21 |
| 4.5.2.8  | tol_rel                                | 21 |
| 4.5.2.9  | tol_abs                                | 21 |
| 4.5.2.10 | res                                    | 21 |
| 4.5.2.11 | relres                                 | 21 |
| 4.5.2.12 | relres_base                            | 21 |
| 4.5.2.13 | bestres                                | 21 |
| 4.5.2.14 | Output                                 | 21 |
| 4.5.2.15 | x                                      | 21 |
| 4.5.2.16 | bestx                                  | 21 |
| 4.5.2.17 | r                                      | 21 |
| 4.5.2.18 | r0                                     | 21 |
| 4.5.2.19 | u                                      | 22 |
| 4.5.2.20 | w                                      | 22 |
| 4.5.2.21 | v                                      | 22 |
| 4.5.2.22 | p                                      | 22 |
| 4.5.2.23 | c                                      | 22 |
| 4.5.2.24 | z                                      | 22 |
| 4.6      | Document Class Reference               | 22 |
| 4.6.1    | Constructor & Destructor Documentation | 23 |
| 4.6.1.1  | Document                               | 23 |
| 4.6.1.2  | ~Document                              | 23 |
| 4.6.1.3  | Document                               | 23 |
| 4.6.1.4  | Document                               | 23 |
| 4.6.1.5  | Document                               | 23 |
| 4.6.1.6  | Document                               | 24 |
| 4.6.1.7  | Document                               | 24 |
| 4.6.2    | Member Function Documentation          | 24 |
| 4.6.2.1  | operator=                              | 24 |
| 4.6.2.2  | operator[]                             | 24 |
| 4.6.2.3  | operator[]                             | 24 |
| 4.6.2.4  | operator()                             | 24 |
| 4.6.2.5  | operator()                             | 24 |
| 4.6.2.6  | getHeadMap                             | 24 |
| 4.6.2.7  | getDataMap                             | 24 |
| 4.6.2.8  | getHeader                              | 24 |

|          |                               |    |
|----------|-------------------------------|----|
| 4.6.2.9  | end                           | 24 |
| 4.6.2.10 | end                           | 24 |
| 4.6.2.11 | begin                         | 24 |
| 4.6.2.12 | begin                         | 24 |
| 4.6.2.13 | clear                         | 24 |
| 4.6.2.14 | resetKeys                     | 24 |
| 4.6.2.15 | changeKey                     | 24 |
| 4.6.2.16 | revalidateAllKeys             | 24 |
| 4.6.2.17 | addPair                       | 24 |
| 4.6.2.18 | addPair                       | 24 |
| 4.6.2.19 | setName                       | 24 |
| 4.6.2.20 | setAlias                      | 24 |
| 4.6.2.21 | setNameAliasPair              | 24 |
| 4.6.2.22 | setState                      | 24 |
| 4.6.2.23 | DisplayContents               | 24 |
| 4.6.2.24 | addHeadKey                    | 24 |
| 4.6.2.25 | copyAnchor2Alias              | 24 |
| 4.6.2.26 | size                          | 25 |
| 4.6.2.27 | getName                       | 25 |
| 4.6.2.28 | getAlias                      | 25 |
| 4.6.2.29 | getState                      | 25 |
| 4.6.2.30 | isAlias                       | 25 |
| 4.6.2.31 | isAnchor                      | 25 |
| 4.6.2.32 | getAnchoredHeader             | 25 |
| 4.6.2.33 | getHeadFromSubAlias           | 25 |
| 4.6.3    | Member Data Documentation     | 25 |
| 4.6.3.1  | Head_Map                      | 25 |
| 4.7      | DOGFISH_DATA Struct Reference | 25 |
| 4.7.1    | Detailed Description          | 26 |
| 4.7.2    | Member Data Documentation     | 26 |
| 4.7.2.1  | total_steps                   | 26 |
| 4.7.2.2  | time_old                      | 26 |
| 4.7.2.3  | time                          | 26 |
| 4.7.2.4  | Print2File                    | 26 |
| 4.7.2.5  | Print2Console                 | 27 |
| 4.7.2.6  | DirichletBC                   | 27 |
| 4.7.2.7  | NonLinear                     | 27 |
| 4.7.2.8  | t_counter                     | 27 |
| 4.7.2.9  | t_print                       | 27 |
| 4.7.2.10 | NumComp                       | 27 |

|          |                                |    |
|----------|--------------------------------|----|
| 4.7.2.11 | end_time                       | 27 |
| 4.7.2.12 | total_sorption_old             | 27 |
| 4.7.2.13 | total_sorption                 | 27 |
| 4.7.2.14 | fiber_length                   | 27 |
| 4.7.2.15 | fiber_diameter                 | 27 |
| 4.7.2.16 | OutputFile                     | 27 |
| 4.7.2.17 | eval_R                         | 28 |
| 4.7.2.18 | eval_DI                        | 28 |
| 4.7.2.19 | eval_kf                        | 28 |
| 4.7.2.20 | eval_qs                        | 28 |
| 4.7.2.21 | user_data                      | 28 |
| 4.7.2.22 | finch_dat                      | 28 |
| 4.7.2.23 | param_dat                      | 28 |
| 4.8      | DOGFISH_PARAM Struct Reference | 28 |
| 4.8.1    | Detailed Description           | 29 |
| 4.8.2    | Member Data Documentation      | 29 |
| 4.8.2.1  | intraparticle_diffusion        | 29 |
| 4.8.2.2  | film_transfer_coeff            | 29 |
| 4.8.2.3  | surface_concentration          | 29 |
| 4.8.2.4  | initial_sorption               | 29 |
| 4.8.2.5  | sorbed_molefraction            | 29 |
| 4.8.2.6  | species                        | 29 |
| 4.9      | FINCH_DATA Struct Reference    | 29 |
| 4.9.1    | Detailed Description           | 33 |
| 4.9.2    | Member Data Documentation      | 34 |
| 4.9.2.1  | d                              | 34 |
| 4.9.2.2  | dt                             | 34 |
| 4.9.2.3  | dt_old                         | 34 |
| 4.9.2.4  | T                              | 34 |
| 4.9.2.5  | dz                             | 34 |
| 4.9.2.6  | L                              | 34 |
| 4.9.2.7  | s                              | 34 |
| 4.9.2.8  | t                              | 34 |
| 4.9.2.9  | t_old                          | 34 |
| 4.9.2.10 | uT                             | 34 |
| 4.9.2.11 | uT_old                         | 34 |
| 4.9.2.12 | uAvg                           | 35 |
| 4.9.2.13 | uAvg_old                       | 35 |
| 4.9.2.14 | uIC                            | 35 |
| 4.9.2.15 | vIC                            | 35 |



|          |              |    |
|----------|--------------|----|
| 4.9.2.16 | DIC          | 35 |
| 4.9.2.17 | kIC          | 35 |
| 4.9.2.18 | RIC          | 35 |
| 4.9.2.19 | uo           | 35 |
| 4.9.2.20 | vo           | 35 |
| 4.9.2.21 | Do           | 35 |
| 4.9.2.22 | ko           | 35 |
| 4.9.2.23 | Ro           | 35 |
| 4.9.2.24 | kfn          | 36 |
| 4.9.2.25 | kfnp1        | 36 |
| 4.9.2.26 | lambda_I     | 36 |
| 4.9.2.27 | lambda_E     | 36 |
| 4.9.2.28 | LN           | 36 |
| 4.9.2.29 | CN           | 36 |
| 4.9.2.30 | Update       | 36 |
| 4.9.2.31 | Dirichlet    | 36 |
| 4.9.2.32 | CheckMass    | 36 |
| 4.9.2.33 | ExplicitFlux | 36 |
| 4.9.2.34 | Iterative    | 36 |
| 4.9.2.35 | SteadyState  | 36 |
| 4.9.2.36 | NormTrack    | 37 |
| 4.9.2.37 | beta         | 37 |
| 4.9.2.38 | tol_rel      | 37 |
| 4.9.2.39 | tol_abs      | 37 |
| 4.9.2.40 | max_iter     | 37 |
| 4.9.2.41 | total_iter   | 37 |
| 4.9.2.42 | nl_method    | 37 |
| 4.9.2.43 | CL_I         | 37 |
| 4.9.2.44 | CL_E         | 37 |
| 4.9.2.45 | CC_I         | 37 |
| 4.9.2.46 | CC_E         | 37 |
| 4.9.2.47 | CR_I         | 37 |
| 4.9.2.48 | CR_E         | 38 |
| 4.9.2.49 | fL_I         | 38 |
| 4.9.2.50 | fL_E         | 38 |
| 4.9.2.51 | fC_I         | 38 |
| 4.9.2.52 | fC_E         | 38 |
| 4.9.2.53 | fR_I         | 38 |
| 4.9.2.54 | fR_E         | 38 |
| 4.9.2.55 | OI           | 38 |

|          |               |    |
|----------|---------------|----|
| 4.9.2.56 | OE            | 38 |
| 4.9.2.57 | NI            | 38 |
| 4.9.2.58 | NE            | 38 |
| 4.9.2.59 | MI            | 38 |
| 4.9.2.60 | ME            | 39 |
| 4.9.2.61 | uz_I_I        | 39 |
| 4.9.2.62 | uz_lm1_I      | 39 |
| 4.9.2.63 | uz_lp1_I      | 39 |
| 4.9.2.64 | uz_I_E        | 39 |
| 4.9.2.65 | uz_lm1_E      | 39 |
| 4.9.2.66 | uz_lp1_E      | 39 |
| 4.9.2.67 | unm1          | 39 |
| 4.9.2.68 | un            | 39 |
| 4.9.2.69 | unp1          | 39 |
| 4.9.2.70 | u_star        | 39 |
| 4.9.2.71 | ubest         | 39 |
| 4.9.2.72 | vn            | 39 |
| 4.9.2.73 | vnp1          | 39 |
| 4.9.2.74 | Dn            | 39 |
| 4.9.2.75 | Dnp1          | 40 |
| 4.9.2.76 | kn            | 40 |
| 4.9.2.77 | knp1          | 40 |
| 4.9.2.78 | Sn            | 40 |
| 4.9.2.79 | Snp1          | 40 |
| 4.9.2.80 | Rn            | 40 |
| 4.9.2.81 | Rnp1          | 40 |
| 4.9.2.82 | Fn            | 40 |
| 4.9.2.83 | Fnp1          | 40 |
| 4.9.2.84 | gl            | 40 |
| 4.9.2.85 | gE            | 40 |
| 4.9.2.86 | res           | 40 |
| 4.9.2.87 | pres          | 41 |
| 4.9.2.88 | callroutine   | 41 |
| 4.9.2.89 | setic         | 41 |
| 4.9.2.90 | settime       | 41 |
| 4.9.2.91 | setpreprocess | 41 |
| 4.9.2.92 | solve         | 41 |
| 4.9.2.93 | setparams     | 41 |
| 4.9.2.94 | discretize    | 41 |
| 4.9.2.95 | setbcs        | 41 |

|                                    |    |
|------------------------------------|----|
| 4.9.2.96 evalres                   | 41 |
| 4.9.2.97 evalprecon                | 41 |
| 4.9.2.98 setpostprocess            | 41 |
| 4.9.2.99 resettime                 | 42 |
| 4.9.2.100 picard_dat               | 42 |
| 4.9.2.101 pjfnk_dat                | 42 |
| 4.9.2.102 param_data               | 42 |
| 4.10 GCR_DATA Struct Reference     | 42 |
| 4.10.1 Detailed Description        | 43 |
| 4.10.2 Member Data Documentation   | 43 |
| 4.10.2.1 restart                   | 43 |
| 4.10.2.2 maxit                     | 43 |
| 4.10.2.3 iter_outer                | 43 |
| 4.10.2.4 iter_inner                | 43 |
| 4.10.2.5 total_iter                | 44 |
| 4.10.2.6 breakdown                 | 44 |
| 4.10.2.7 alpha                     | 44 |
| 4.10.2.8 beta                      | 44 |
| 4.10.2.9 tol_rel                   | 44 |
| 4.10.2.10 tol_abs                  | 44 |
| 4.10.2.11 res                      | 44 |
| 4.10.2.12 relres                   | 44 |
| 4.10.2.13 relres_base              | 44 |
| 4.10.2.14 bestres                  | 44 |
| 4.10.2.15 Output                   | 44 |
| 4.10.2.16 x                        | 44 |
| 4.10.2.17 bestx                    | 45 |
| 4.10.2.18 r                        | 45 |
| 4.10.2.19 c_temp                   | 45 |
| 4.10.2.20 u_temp                   | 45 |
| 4.10.2.21 u                        | 45 |
| 4.10.2.22 c                        | 45 |
| 4.10.2.23 transpose_dat            | 45 |
| 4.11 GMRESLP_DATA Struct Reference | 45 |
| 4.11.1 Detailed Description        | 46 |
| 4.11.2 Member Data Documentation   | 46 |
| 4.11.2.1 restart                   | 46 |
| 4.11.2.2 maxit                     | 46 |
| 4.11.2.3 iter                      | 46 |
| 4.11.2.4 steps                     | 46 |

|           |                               |    |
|-----------|-------------------------------|----|
| 4.11.2.5  | tol_rel                       | 46 |
| 4.11.2.6  | tol_abs                       | 47 |
| 4.11.2.7  | res                           | 47 |
| 4.11.2.8  | relres                        | 47 |
| 4.11.2.9  | relres_base                   | 47 |
| 4.11.2.10 | bestres                       | 47 |
| 4.11.2.11 | Output                        | 47 |
| 4.11.2.12 | x                             | 47 |
| 4.11.2.13 | bestx                         | 47 |
| 4.11.2.14 | r                             | 47 |
| 4.11.2.15 | arnoldi_dat                   | 47 |
| 4.12      | GMRESR_DATA Struct Reference  | 47 |
| 4.12.1    | Detailed Description          | 48 |
| 4.12.2    | Member Data Documentation     | 49 |
| 4.12.2.1  | gcr_restart                   | 49 |
| 4.12.2.2  | gcr_maxit                     | 49 |
| 4.12.2.3  | gmres_restart                 | 49 |
| 4.12.2.4  | gmres_maxit                   | 49 |
| 4.12.2.5  | N                             | 49 |
| 4.12.2.6  | total_iter                    | 49 |
| 4.12.2.7  | iter_outer                    | 49 |
| 4.12.2.8  | iter_inner                    | 49 |
| 4.12.2.9  | GCR_Output                    | 49 |
| 4.12.2.10 | GMRES_Output                  | 49 |
| 4.12.2.11 | gmres_tol                     | 49 |
| 4.12.2.12 | gcr_rel_tol                   | 49 |
| 4.12.2.13 | gcr_abs_tol                   | 50 |
| 4.12.2.14 | arg                           | 50 |
| 4.12.2.15 | gcr_dat                       | 50 |
| 4.12.2.16 | gmres_dat                     | 50 |
| 4.12.2.17 | matvec                        | 50 |
| 4.12.2.18 | terminal_precon               | 50 |
| 4.12.2.19 | matvec_data                   | 50 |
| 4.12.2.20 | term_precon                   | 50 |
| 4.13      | GMRESRP_DATA Struct Reference | 50 |
| 4.13.1    | Detailed Description          | 51 |
| 4.13.2    | Member Data Documentation     | 52 |
| 4.13.2.1  | restart                       | 52 |
| 4.13.2.2  | maxit                         | 52 |
| 4.13.2.3  | iter_outer                    | 52 |

|           |                             |    |
|-----------|-----------------------------|----|
| 4.13.2.4  | iter_inner                  | 52 |
| 4.13.2.5  | iter_total                  | 52 |
| 4.13.2.6  | tol_rel                     | 52 |
| 4.13.2.7  | tol_abs                     | 52 |
| 4.13.2.8  | res                         | 52 |
| 4.13.2.9  | relres                      | 52 |
| 4.13.2.10 | relres_base                 | 52 |
| 4.13.2.11 | bestres                     | 52 |
| 4.13.2.12 | Output                      | 52 |
| 4.13.2.13 | x                           | 53 |
| 4.13.2.14 | bestx                       | 53 |
| 4.13.2.15 | r                           | 53 |
| 4.13.2.16 | Vk                          | 53 |
| 4.13.2.17 | H                           | 53 |
| 4.13.2.18 | H_bar                       | 53 |
| 4.13.2.19 | y                           | 53 |
| 4.13.2.20 | e0                          | 53 |
| 4.13.2.21 | e0_bar                      | 53 |
| 4.13.2.22 | w                           | 53 |
| 4.13.2.23 | v                           | 53 |
| 4.13.2.24 | sum                         | 53 |
| 4.14      | GPAST_DATA Struct Reference | 54 |
| 4.14.1    | Detailed Description        | 54 |
| 4.14.2    | Member Data Documentation   | 54 |
| 4.14.2.1  | x                           | 54 |
| 4.14.2.2  | y                           | 54 |
| 4.14.2.3  | He                          | 54 |
| 4.14.2.4  | q                           | 55 |
| 4.14.2.5  | gama_inf                    | 55 |
| 4.14.2.6  | qo                          | 55 |
| 4.14.2.7  | Plo                         | 55 |
| 4.14.2.8  | po                          | 55 |
| 4.14.2.9  | poi                         | 55 |
| 4.14.2.10 | present                     | 55 |
| 4.15      | GSTA_DATA Struct Reference  | 55 |
| 4.15.1    | Detailed Description        | 56 |
| 4.15.2    | Member Data Documentation   | 56 |
| 4.15.2.1  | qmax                        | 56 |
| 4.15.2.2  | m                           | 56 |
| 4.15.2.3  | dHo                         | 56 |

|   |    |
|---|----|
| 4.15.2.4 dSo . . . . .                                  | 56 |
| 4.16 GSTA_OPT_DATA Struct Reference . . . . .           | 56 |
| 4.16.1 Detailed Description . . . . .                   | 57 |
| 4.16.2 Member Data Documentation . . . . .              | 57 |
| 4.16.2.1 total_eval . . . . .                           | 57 |
| 4.16.2.2 n_par . . . . .                                | 57 |
| 4.16.2.3 qmax . . . . .                                 | 57 |
| 4.16.2.4 iso . . . . .                                  | 57 |
| 4.16.2.5 Fobj . . . . .                                 | 57 |
| 4.16.2.6 q . . . . .                                    | 57 |
| 4.16.2.7 P . . . . .                                    | 57 |
| 4.16.2.8 best_par . . . . .                             | 58 |
| 4.16.2.9 Kno . . . . .                                  | 58 |
| 4.16.2.10 all_pars . . . . .                            | 58 |
| 4.16.2.11 norms . . . . .                               | 58 |
| 4.16.2.12 opt_qmax . . . . .                            | 58 |
| 4.17 Header Class Reference . . . . .                   | 58 |
| 4.17.1 Constructor & Destructor Documentation . . . . . | 59 |
| 4.17.1.1 Header . . . . .                               | 59 |
| 4.17.1.2 ~Header . . . . .                              | 59 |
| 4.17.1.3 Header . . . . .                               | 59 |
| 4.17.1.4 Header . . . . .                               | 59 |
| 4.17.1.5 Header . . . . .                               | 59 |
| 4.17.1.6 Header . . . . .                               | 59 |
| 4.17.1.7 Header . . . . .                               | 59 |
| 4.17.2 Member Function Documentation . . . . .          | 59 |
| 4.17.2.1 operator= . . . . .                            | 60 |
| 4.17.2.2 operator[] . . . . .                           | 60 |
| 4.17.2.3 operator[] . . . . .                           | 60 |
| 4.17.2.4 operator() . . . . .                           | 60 |
| 4.17.2.5 operator() . . . . .                           | 60 |
| 4.17.2.6 getSubMap . . . . .                            | 60 |
| 4.17.2.7 getDataMap . . . . .                           | 60 |
| 4.17.2.8 getSubHeader . . . . .                         | 60 |
| 4.17.2.9 end . . . . .                                  | 60 |
| 4.17.2.10 end . . . . .                                 | 60 |
| 4.17.2.11 begin . . . . .                               | 60 |
| 4.17.2.12 begin . . . . .                               | 60 |
| 4.17.2.13 clear . . . . .                               | 60 |
| 4.17.2.14 resetKeys . . . . .                           | 60 |

|           |  |    |
|-----------|--|----|
| 4.17.2.15 | changeKey                              | 60 |
| 4.17.2.16 | addPair                                | 60 |
| 4.17.2.17 | addPair                                | 60 |
| 4.17.2.18 | setName                                | 60 |
| 4.17.2.19 | setAlias                               | 60 |
| 4.17.2.20 | setNameAliasPair                       | 60 |
| 4.17.2.21 | setState                               | 60 |
| 4.17.2.22 | DisplayContents                        | 60 |
| 4.17.2.23 | addSubKey                              | 60 |
| 4.17.2.24 | copyAnchor2Alias                       | 60 |
| 4.17.2.25 | size                                   | 60 |
| 4.17.2.26 | getName                                | 60 |
| 4.17.2.27 | getAlias                               | 60 |
| 4.17.2.28 | getState                               | 60 |
| 4.17.2.29 | isAlias                                | 61 |
| 4.17.2.30 | isAnchor                               | 61 |
| 4.17.2.31 | getAnchoredSub                         | 61 |
| 4.17.3    | Member Data Documentation              | 61 |
| 4.17.3.1  | Sub_Map                                | 61 |
| 4.18      | KeyValueMap Class Reference            | 61 |
| 4.18.1    | Constructor & Destructor Documentation | 62 |
| 4.18.1.1  | KeyValueMap                            | 62 |
| 4.18.1.2  | ~KeyValueMap                           | 62 |
| 4.18.1.3  | KeyValueMap                            | 62 |
| 4.18.1.4  | KeyValueMap                            | 62 |
| 4.18.1.5  | KeyValueMap                            | 62 |
| 4.18.2    | Member Function Documentation          | 62 |
| 4.18.2.1  | operator=                              | 62 |
| 4.18.2.2  | operator[]                             | 62 |
| 4.18.2.3  | operator[]                             | 62 |
| 4.18.2.4  | getMap                                 | 62 |
| 4.18.2.5  | end                                    | 62 |
| 4.18.2.6  | end                                    | 62 |
| 4.18.2.7  | begin                                  | 62 |
| 4.18.2.8  | begin                                  | 62 |
| 4.18.2.9  | clear                                  | 62 |
| 4.18.2.10 | addKey                                 | 62 |
| 4.18.2.11 | editValue4Key                          | 62 |
| 4.18.2.12 | editValue4Key                          | 62 |
| 4.18.2.13 | addPair                                | 62 |

|           |  |    |
|-----------|--|----|
| 4.18.2.14 | <a href="#">addPair</a>                                    | 62 |
| 4.18.2.15 | <a href="#">addPair</a>                                    | 63 |
| 4.18.2.16 | <a href="#">findType</a>                                   | 63 |
| 4.18.2.17 | <a href="#">assertType</a>                                 | 63 |
| 4.18.2.18 | <a href="#">findAllTypes</a>                               | 63 |
| 4.18.2.19 | <a href="#">DisplayMap</a>                                 | 63 |
| 4.18.2.20 | <a href="#">size</a>                                       | 63 |
| 4.18.2.21 | <a href="#">getString</a>                                  | 63 |
| 4.18.2.22 | <a href="#">getBool</a>                                    | 63 |
| 4.18.2.23 | <a href="#">getDouble</a>                                  | 63 |
| 4.18.2.24 | <a href="#">getInt</a>                                     | 63 |
| 4.18.2.25 | <a href="#">getValue</a>                                   | 63 |
| 4.18.2.26 | <a href="#">getType</a>                                    | 63 |
| 4.18.2.27 | <a href="#">getPair</a>                                    | 63 |
| 4.18.3    | <a href="#">Member Data Documentation</a>                  | 63 |
| 4.18.3.1  | <a href="#">Key_Value</a>                                  | 63 |
| 4.19      | <a href="#">MAGPIE_DATA Struct Reference</a>               | 63 |
| 4.19.1    | <a href="#">Detailed Description</a>                       | 63 |
| 4.19.2    | <a href="#">Member Data Documentation</a>                  | 64 |
| 4.19.2.1  | <a href="#">gsta_dat</a>                                   | 64 |
| 4.19.2.2  | <a href="#">mspd_dat</a>                                   | 64 |
| 4.19.2.3  | <a href="#">gpast_dat</a>                                  | 64 |
| 4.19.2.4  | <a href="#">sys_dat</a>                                    | 64 |
| 4.20      | <a href="#">MassBalance Class Reference</a>                | 64 |
| 4.20.1    | <a href="#">Constructor &amp; Destructor Documentation</a> | 64 |
| 4.20.1.1  | <a href="#">MassBalance</a>                                | 64 |
| 4.20.1.2  | <a href="#">~MassBalance</a>                               | 64 |
| 4.20.2    | <a href="#">Member Function Documentation</a>              | 64 |
| 4.20.2.1  | <a href="#">Initialize_List</a>                            | 65 |
| 4.20.2.2  | <a href="#">Display_Info</a>                               | 65 |
| 4.20.2.3  | <a href="#">Set_Delta</a>                                  | 65 |
| 4.20.2.4  | <a href="#">Set_TotalConcentration</a>                     | 65 |
| 4.20.2.5  | <a href="#">Set_Name</a>                                   | 65 |
| 4.20.2.6  | <a href="#">Get_Delta</a>                                  | 65 |
| 4.20.2.7  | <a href="#">Sum_Delta</a>                                  | 65 |
| 4.20.2.8  | <a href="#">Get_TotalConcentration</a>                     | 65 |
| 4.20.2.9  | <a href="#">Get_Name</a>                                   | 65 |
| 4.20.2.10 | <a href="#">Eval_Residual</a>                              | 65 |
| 4.20.3    | <a href="#">Member Data Documentation</a>                  | 65 |
| 4.20.3.1  | <a href="#">List</a>                                       | 65 |



|           |  |    |
|-----------|--|----|
| 4.20.3.2  | Delta                                  | 65 |
| 4.20.3.3  | TotalConcentration                     | 65 |
| 4.20.3.4  | Name                                   | 65 |
| 4.21      | MasterSpeciesList Class Reference      | 65 |
| 4.21.1    | Constructor & Destructor Documentation | 66 |
| 4.21.1.1  | MasterSpeciesList                      | 66 |
| 4.21.1.2  | ~MasterSpeciesList                     | 66 |
| 4.21.1.3  | MasterSpeciesList                      | 66 |
| 4.21.2    | Member Function Documentation          | 66 |
| 4.21.2.1  | operator=                              | 66 |
| 4.21.2.2  | set_list_size                          | 66 |
| 4.21.2.3  | set_species                            | 66 |
| 4.21.2.4  | set_species                            | 66 |
| 4.21.2.5  | DisplayInfo                            | 66 |
| 4.21.2.6  | DisplayAll                             | 66 |
| 4.21.2.7  | DisplayConcentrations                  | 66 |
| 4.21.2.8  | set_alkalinity                         | 66 |
| 4.21.2.9  | list_size                              | 66 |
| 4.21.2.10 | get_species                            | 66 |
| 4.21.2.11 | get_index                              | 66 |
| 4.21.2.12 | charge                                 | 67 |
| 4.21.2.13 | alkalinity                             | 67 |
| 4.21.2.14 | speciesName                            | 67 |
| 4.21.2.15 | Eval_ChargeResidual                    | 67 |
| 4.21.3    | Member Data Documentation              | 67 |
| 4.21.3.1  | size                                   | 67 |
| 4.21.3.2  | species                                | 67 |
| 4.21.3.3  | residual_alkalinity                    | 67 |
| 4.22      | Matrix< T > Class Template Reference   | 67 |
| 4.22.1    | Detailed Description                   | 70 |
| 4.22.2    | Constructor & Destructor Documentation | 70 |
| 4.22.2.1  | Matrix                                 | 70 |
| 4.22.2.2  | Matrix                                 | 70 |
| 4.22.2.3  | Matrix                                 | 70 |
| 4.22.2.4  | ~Matrix                                | 70 |
| 4.22.3    | Member Function Documentation          | 70 |
| 4.22.3.1  | operator()                             | 70 |
| 4.22.3.2  | operator()                             | 70 |
| 4.22.3.3  | operator=                              | 70 |
| 4.22.3.4  | set_size                               | 70 |

|                                      |    |
|--------------------------------------|----|
| 4.22.3.5 zeros                       | 70 |
| 4.22.3.6 edit                        | 71 |
| 4.22.3.7 rows                        | 71 |
| 4.22.3.8 columns                     | 71 |
| 4.22.3.9 determinate                 | 71 |
| 4.22.3.10 norm                       | 71 |
| 4.22.3.11 sum                        | 71 |
| 4.22.3.12 inner_product              | 71 |
| 4.22.3.13 cofactor                   | 71 |
| 4.22.3.14 operator+                  | 71 |
| 4.22.3.15 operator-                  | 71 |
| 4.22.3.16 operator*                  | 71 |
| 4.22.3.17 operator/                  | 71 |
| 4.22.3.18 operator*                  | 72 |
| 4.22.3.19 transpose                  | 72 |
| 4.22.3.20 transpose_multiply         | 72 |
| 4.22.3.21 adjoint                    | 72 |
| 4.22.3.22 inverse                    | 72 |
| 4.22.3.23 Display                    | 72 |
| 4.22.3.24 tridiagonalSolve           | 72 |
| 4.22.3.25 ladshawSolve               | 72 |
| 4.22.3.26 tridiagonalFill            | 72 |
| 4.22.3.27 naturalLaplacian3D         | 72 |
| 4.22.3.28 sphericalBCFill            | 73 |
| 4.22.3.29 ConstantICFill             | 73 |
| 4.22.3.30 SolnTransform              | 73 |
| 4.22.3.31 sphericalAvg               | 73 |
| 4.22.3.32 IntegralAvg                | 73 |
| 4.22.3.33 IntegralTotal              | 73 |
| 4.22.3.34 tridiagonalVectorFill      | 74 |
| 4.22.3.35 columnVectorFill           | 74 |
| 4.22.3.36 columnProjection           | 74 |
| 4.22.3.37 dirichletBCFill            | 74 |
| 4.22.3.38 diagonalSolve              | 74 |
| 4.22.3.39 upperTriangularSolve       | 74 |
| 4.22.3.40 lowerTriangularSolve       | 74 |
| 4.22.3.41 upperHessenberg2Triangular | 74 |
| 4.22.3.42 lowerHessenberg2Triangular | 74 |
| 4.22.3.43 upperHessenbergSolve       | 75 |
| 4.22.3.44 lowerHessenbergSolve       | 75 |

|   |    |
|---|----|
| 4.22.3.45 columnExtract . . . . .                       | 75 |
| 4.22.3.46 rowExtract . . . . .                          | 75 |
| 4.22.3.47 columnReplace . . . . .                       | 75 |
| 4.22.3.48 rowReplace . . . . .                          | 75 |
| 4.22.3.49 rowShrink . . . . .                           | 75 |
| 4.22.3.50 columnShrink . . . . .                        | 75 |
| 4.22.3.51 rowExtend . . . . .                           | 75 |
| 4.22.3.52 columnExtend . . . . .                        | 75 |
| 4.22.4 Member Data Documentation . . . . .              | 76 |
| 4.22.4.1 num_rows . . . . .                             | 76 |
| 4.22.4.2 num_cols . . . . .                             | 76 |
| 4.22.4.3 Data . . . . .                                 | 76 |
| 4.23 Mechanism Class Reference . . . . .                | 76 |
| 4.23.1 Member Data Documentation . . . . .              | 76 |
| 4.23.1.1 List . . . . .                                 | 76 |
| 4.23.1.2 reactions . . . . .                            | 76 |
| 4.23.1.3 weight . . . . .                               | 76 |
| 4.23.1.4 species_index . . . . .                        | 76 |
| 4.24 MIXED_GAS Struct Reference . . . . .               | 76 |
| 4.24.1 Detailed Description . . . . .                   | 77 |
| 4.24.2 Member Data Documentation . . . . .              | 77 |
| 4.24.2.1 N . . . . .                                    | 77 |
| 4.24.2.2 CheckMolefractions . . . . .                   | 77 |
| 4.24.2.3 total_pressure . . . . .                       | 78 |
| 4.24.2.4 gas_temperature . . . . .                      | 78 |
| 4.24.2.5 velocity . . . . .                             | 78 |
| 4.24.2.6 char_length . . . . .                          | 78 |
| 4.24.2.7 molefraction . . . . .                         | 78 |
| 4.24.2.8 total_density . . . . .                        | 78 |
| 4.24.2.9 total_dyn_vis . . . . .                        | 78 |
| 4.24.2.10 kinematic_viscosity . . . . .                 | 78 |
| 4.24.2.11 total_molecular_weight . . . . .              | 78 |
| 4.24.2.12 total_specific_heat . . . . .                 | 78 |
| 4.24.2.13 Reynolds . . . . .                            | 78 |
| 4.24.2.14 binary_diffusion . . . . .                    | 78 |
| 4.24.2.15 species_dat . . . . .                         | 79 |
| 4.25 Molecule Class Reference . . . . .                 | 79 |
| 4.25.1 Detailed Description . . . . .                   | 81 |
| 4.25.2 Constructor & Destructor Documentation . . . . . | 81 |
| 4.25.2.1 Molecule . . . . .                             | 81 |

|           |                               |    |
|-----------|-------------------------------|----|
| 4.25.2.2  | ~Molecule                     | 81 |
| 4.25.2.3  | Molecule                      | 81 |
| 4.25.3    | Member Function Documentation | 82 |
| 4.25.3.1  | Register                      | 82 |
| 4.25.3.2  | Register                      | 82 |
| 4.25.3.3  | setFormula                    | 82 |
| 4.25.3.4  | recalculateMolarWeight        | 82 |
| 4.25.3.5  | setMolarWeigth                | 82 |
| 4.25.3.6  | editCharge                    | 82 |
| 4.25.3.7  | editOneOxidationState         | 83 |
| 4.25.3.8  | editAllOxidationStates        | 83 |
| 4.25.3.9  | calculateAvgOxiState          | 83 |
| 4.25.3.10 | editEnthalpy                  | 83 |
| 4.25.3.11 | editEntropy                   | 83 |
| 4.25.3.12 | editHS                        | 83 |
| 4.25.3.13 | editEnergy                    | 83 |
| 4.25.3.14 | removeOneAtom                 | 83 |
| 4.25.3.15 | removeAllAtoms                | 83 |
| 4.25.3.16 | Charge                        | 84 |
| 4.25.3.17 | MolarWeight                   | 84 |
| 4.25.3.18 | HaveHS                        | 84 |
| 4.25.3.19 | HaveEnergy                    | 84 |
| 4.25.3.20 | isRegistered                  | 84 |
| 4.25.3.21 | Enthalpy                      | 84 |
| 4.25.3.22 | Entropy                       | 84 |
| 4.25.3.23 | Energy                        | 84 |
| 4.25.3.24 | MoleculeName                  | 84 |
| 4.25.3.25 | MolecularFormula              | 84 |
| 4.25.3.26 | MoleculePhase                 | 84 |
| 4.25.3.27 | DisplayInfo                   | 84 |
| 4.25.4    | Member Data Documentation     | 85 |
| 4.25.4.1  | charge                        | 85 |
| 4.25.4.2  | molar_weight                  | 85 |
| 4.25.4.3  | formation_enthalpy            | 85 |
| 4.25.4.4  | formation_entropy             | 85 |
| 4.25.4.5  | formation_energy              | 85 |
| 4.25.4.6  | Phase                         | 85 |
| 4.25.4.7  | atoms                         | 85 |
| 4.25.4.8  | Name                          | 85 |
| 4.25.4.9  | Formula                       | 85 |

|   |    |
|---|----|
| 4.25.4.10 haveG . . . . .                     | 85 |
| 4.25.4.11 haveHS . . . . .                    | 85 |
| 4.25.4.12 registered . . . . .                | 86 |
| 4.26 MONKFISH_DATA Struct Reference . . . . . | 86 |
| 4.26.1 Member Data Documentation . . . . .    | 87 |
| 4.26.1.1 total_steps . . . . .                | 87 |
| 4.26.1.2 time_old . . . . .                   | 87 |
| 4.26.1.3 time . . . . .                       | 87 |
| 4.26.1.4 Print2File . . . . .                 | 87 |
| 4.26.1.5 Print2Console . . . . .              | 87 |
| 4.26.1.6 DirichletBC . . . . .                | 87 |
| 4.26.1.7 NonLinear . . . . .                  | 87 |
| 4.26.1.8 haveMinMax . . . . .                 | 87 |
| 4.26.1.9 MultiScale . . . . .                 | 87 |
| 4.26.1.10 level . . . . .                     | 87 |
| 4.26.1.11 t_counter . . . . .                 | 87 |
| 4.26.1.12 t_print . . . . .                   | 87 |
| 4.26.1.13 NumComp . . . . .                   | 87 |
| 4.26.1.14 end_time . . . . .                  | 87 |
| 4.26.1.15 total_sorption_old . . . . .        | 87 |
| 4.26.1.16 total_sorption . . . . .            | 87 |
| 4.26.1.17 single_fiber_density . . . . .      | 87 |
| 4.26.1.18 avg_fiber_density . . . . .         | 87 |
| 4.26.1.19 max_fiber_density . . . . .         | 87 |
| 4.26.1.20 min_fiber_density . . . . .         | 87 |
| 4.26.1.21 max_porosity . . . . .              | 87 |
| 4.26.1.22 min_porosity . . . . .              | 87 |
| 4.26.1.23 domain_diameter . . . . .           | 87 |
| 4.26.1.24 Output . . . . .                    | 87 |
| 4.26.1.25 eval_eps . . . . .                  | 87 |
| 4.26.1.26 eval_rho . . . . .                  | 87 |
| 4.26.1.27 eval_Dex . . . . .                  | 87 |
| 4.26.1.28 eval_ads . . . . .                  | 88 |
| 4.26.1.29 eval_Ret . . . . .                  | 88 |
| 4.26.1.30 eval_Cex . . . . .                  | 88 |
| 4.26.1.31 eval_kf . . . . .                   | 88 |
| 4.26.1.32 user_data . . . . .                 | 88 |
| 4.26.1.33 finch_dat . . . . .                 | 88 |
| 4.26.1.34 param_dat . . . . .                 | 88 |
| 4.26.1.35 dog_dat . . . . .                   | 88 |

|           |                                 |    |
|-----------|---------------------------------|----|
| 4.27      | MONKFISH_PARAM Struct Reference | 88 |
| 4.27.1    | Member Data Documentation       | 88 |
| 4.27.1.1  | interparticle_diffusion         | 88 |
| 4.27.1.2  | exterior_concentration          | 88 |
| 4.27.1.3  | exterior_transfer_coeff         | 88 |
| 4.27.1.4  | sorbed_molefraction             | 88 |
| 4.27.1.5  | initial_sorption                | 88 |
| 4.27.1.6  | sorption_bc                     | 88 |
| 4.27.1.7  | intraparticle_diffusion         | 88 |
| 4.27.1.8  | film_transfer_coeff             | 89 |
| 4.27.1.9  | avg_sorption                    | 89 |
| 4.27.1.10 | avg_sorption_old                | 89 |
| 4.27.1.11 | species                         | 89 |
| 4.28      | mSPD_DATA Struct Reference      | 89 |
| 4.28.1    | Detailed Description            | 89 |
| 4.28.2    | Member Data Documentation       | 89 |
| 4.28.2.1  | s                               | 89 |
| 4.28.2.2  | v                               | 89 |
| 4.28.2.3  | eMax                            | 89 |
| 4.28.2.4  | eta                             | 90 |
| 4.28.2.5  | gama                            | 90 |
| 4.29      | NUM_JAC_DATA Struct Reference   | 90 |
| 4.29.1    | Detailed Description            | 90 |
| 4.29.2    | Member Data Documentation       | 90 |
| 4.29.2.1  | eps                             | 90 |
| 4.29.2.2  | Fx                              | 90 |
| 4.29.2.3  | Fxp                             | 90 |
| 4.29.2.4  | dxj                             | 91 |
| 4.30      | OPTRANS_DATA Struct Reference   | 91 |
| 4.30.1    | Detailed Description            | 91 |
| 4.30.2    | Member Data Documentation       | 91 |
| 4.30.2.1  | li                              | 91 |
| 4.30.2.2  | Ai                              | 91 |
| 4.31      | PCG_DATA Struct Reference       | 91 |
| 4.31.1    | Detailed Description            | 92 |
| 4.31.2    | Member Data Documentation       | 93 |
| 4.31.2.1  | maxit                           | 93 |
| 4.31.2.2  | iter                            | 93 |
| 4.31.2.3  | alpha                           | 93 |
| 4.31.2.4  | beta                            | 93 |

|           |  |    |
|-----------|--|----|
| 4.31.2.5  | tol_rel                                | 93 |
| 4.31.2.6  | tol_abs                                | 93 |
| 4.31.2.7  | res                                    | 93 |
| 4.31.2.8  | relres                                 | 93 |
| 4.31.2.9  | relres_base                            | 93 |
| 4.31.2.10 | bestres                                | 93 |
| 4.31.2.11 | Output                                 | 93 |
| 4.31.2.12 | x                                      | 93 |
| 4.31.2.13 | bestx                                  | 94 |
| 4.31.2.14 | r                                      | 94 |
| 4.31.2.15 | r_old                                  | 94 |
| 4.31.2.16 | z                                      | 94 |
| 4.31.2.17 | z_old                                  | 94 |
| 4.31.2.18 | p                                      | 94 |
| 4.31.2.19 | Ap                                     | 94 |
| 4.32      | PeriodicTable Class Reference          | 94 |
| 4.32.1    | Detailed Description                   | 95 |
| 4.32.2    | Constructor & Destructor Documentation | 95 |
| 4.32.2.1  | PeriodicTable                          | 95 |
| 4.32.2.2  | ~PeriodicTable                         | 95 |
| 4.32.2.3  | PeriodicTable                          | 95 |
| 4.32.2.4  | PeriodicTable                          | 95 |
| 4.32.2.5  | PeriodicTable                          | 95 |
| 4.32.3    | Member Function Documentation          | 95 |
| 4.32.3.1  | DisplayTable                           | 95 |
| 4.32.4    | Member Data Documentation              | 95 |
| 4.32.4.1  | Table                                  | 95 |
| 4.32.4.2  | number_elements                        | 96 |
| 4.33      | PICARD_DATA Struct Reference           | 96 |
| 4.33.1    | Detailed Description                   | 96 |
| 4.33.2    | Member Data Documentation              | 97 |
| 4.33.2.1  | maxit                                  | 97 |
| 4.33.2.2  | iter                                   | 97 |
| 4.33.2.3  | tol_rel                                | 97 |
| 4.33.2.4  | tol_abs                                | 97 |
| 4.33.2.5  | res                                    | 97 |
| 4.33.2.6  | relres                                 | 97 |
| 4.33.2.7  | relres_base                            | 97 |
| 4.33.2.8  | bestres                                | 97 |
| 4.33.2.9  | Output                                 | 97 |

|  |     |
|--|-----|
| 4.33.2.10 x0 . . . . .                     | 97  |
| 4.33.2.11 bestx . . . . .                  | 97  |
| 4.33.2.12 r . . . . .                      | 98  |
| 4.34 PJFNK_DATA Struct Reference . . . . . | 98  |
| 4.34.1 Detailed Description . . . . .      | 99  |
| 4.34.2 Member Data Documentation . . . . . | 99  |
| 4.34.2.1 nl_iter . . . . .                 | 99  |
| 4.34.2.2 l_iter . . . . .                  | 100 |
| 4.34.2.3 nl_maxit . . . . .                | 100 |
| 4.34.2.4 linear_solver . . . . .           | 100 |
| 4.34.2.5 nl_tol_abs . . . . .              | 100 |
| 4.34.2.6 nl_tol_rel . . . . .              | 100 |
| 4.34.2.7 lin_tol_rel . . . . .             | 100 |
| 4.34.2.8 lin_tol_abs . . . . .             | 100 |
| 4.34.2.9 nl_res . . . . .                  | 100 |
| 4.34.2.10 nl_relres . . . . .              | 100 |
| 4.34.2.11 nl_res_base . . . . .            | 100 |
| 4.34.2.12 nl_bestres . . . . .             | 100 |
| 4.34.2.13 eps . . . . .                    | 100 |
| 4.34.2.14 NL_Output . . . . .              | 101 |
| 4.34.2.15 L_Output . . . . .               | 101 |
| 4.34.2.16 LineSearch . . . . .             | 101 |
| 4.34.2.17 Bounce . . . . .                 | 101 |
| 4.34.2.18 F . . . . .                      | 101 |
| 4.34.2.19 Fv . . . . .                     | 101 |
| 4.34.2.20 v . . . . .                      | 101 |
| 4.34.2.21 x . . . . .                      | 101 |
| 4.34.2.22 bestx . . . . .                  | 101 |
| 4.34.2.23 gmreslp_dat . . . . .            | 101 |
| 4.34.2.24 pcg_dat . . . . .                | 101 |
| 4.34.2.25 bicgstab_dat . . . . .           | 101 |
| 4.34.2.26 cgs_dat . . . . .                | 102 |
| 4.34.2.27 gmresrp_dat . . . . .            | 102 |
| 4.34.2.28 gcr_dat . . . . .                | 102 |
| 4.34.2.29 gmresr_dat . . . . .             | 102 |
| 4.34.2.30 backtrack_dat . . . . .          | 102 |
| 4.34.2.31 res_data . . . . .               | 102 |
| 4.34.2.32 precon_data . . . . .            | 102 |
| 4.34.2.33 funeval . . . . .                | 102 |
| 4.34.2.34 precon . . . . .                 | 102 |



|           |  |     |
|-----------|--|-----|
| 4.35      | Precipitation Class Reference . . . . .          | 102 |
| 4.36      | PURE_GAS Struct Reference . . . . .              | 103 |
| 4.36.1    | Detailed Description . . . . .                   | 103 |
| 4.36.2    | Member Data Documentation . . . . .              | 104 |
| 4.36.2.1  | molecular_weight . . . . .                       | 104 |
| 4.36.2.2  | Sutherland_Temp . . . . .                        | 104 |
| 4.36.2.3  | Sutherland_Const . . . . .                       | 104 |
| 4.36.2.4  | Sutherland_Viscosity . . . . .                   | 104 |
| 4.36.2.5  | specific_heat . . . . .                          | 104 |
| 4.36.2.6  | molecular_diffusion . . . . .                    | 104 |
| 4.36.2.7  | dynamic_viscosity . . . . .                      | 104 |
| 4.36.2.8  | density . . . . .                                | 104 |
| 4.36.2.9  | Schmidt . . . . .                                | 104 |
| 4.37      | Reaction Class Reference . . . . .               | 104 |
| 4.37.1    | Constructor & Destructor Documentation . . . . . | 105 |
| 4.37.1.1  | Reaction . . . . .                               | 105 |
| 4.37.1.2  | ~Reaction . . . . .                              | 105 |
| 4.37.2    | Member Function Documentation . . . . .          | 105 |
| 4.37.2.1  | Initialize_List . . . . .                        | 106 |
| 4.37.2.2  | Display_Info . . . . .                           | 106 |
| 4.37.2.3  | Set_Stoichiometric . . . . .                     | 106 |
| 4.37.2.4  | Set_Equilibrium . . . . .                        | 106 |
| 4.37.2.5  | Set_Enthalpy . . . . .                           | 106 |
| 4.37.2.6  | Set_Entropy . . . . .                            | 106 |
| 4.37.2.7  | Set_EnthalpyANDEntropy . . . . .                 | 106 |
| 4.37.2.8  | Set_Energy . . . . .                             | 106 |
| 4.37.2.9  | checkSpeciesEnergies . . . . .                   | 106 |
| 4.37.2.10 | calculateEnergies . . . . .                      | 106 |
| 4.37.2.11 | calculateEquilibrium . . . . .                   | 106 |
| 4.37.2.12 | haveEquilibrium . . . . .                        | 106 |
| 4.37.2.13 | Get_Stoichiometric . . . . .                     | 106 |
| 4.37.2.14 | Get_Equilibrium . . . . .                        | 106 |
| 4.37.2.15 | Get_Enthalpy . . . . .                           | 106 |
| 4.37.2.16 | Get_Entropy . . . . .                            | 106 |
| 4.37.2.17 | Get_Energy . . . . .                             | 106 |
| 4.37.2.18 | Eval_Residual . . . . .                          | 106 |
| 4.37.3    | Member Data Documentation . . . . .              | 106 |
| 4.37.3.1  | List . . . . .                                   | 106 |
| 4.37.3.2  | Stoichiometric . . . . .                         | 106 |
| 4.37.3.3  | Equilibrium . . . . .                            | 106 |

|           |                                |     |
|-----------|--------------------------------|-----|
| 4.37.3.4  | enthalpy                       | 106 |
| 4.37.3.5  | entropy                        | 106 |
| 4.37.3.6  | energy                         | 106 |
| 4.37.3.7  | CanCalcHS                      | 106 |
| 4.37.3.8  | CanCalcG                       | 106 |
| 4.37.3.9  | HaveHS                         | 106 |
| 4.37.3.10 | HaveG                          | 107 |
| 4.37.3.11 | HaveEquil                      | 107 |
| 4.38      | SCOPSOWL_DATA Struct Reference | 107 |
| 4.38.1    | Member Data Documentation      | 108 |
| 4.38.1.1  | total_steps                    | 108 |
| 4.38.1.2  | coord_macro                    | 108 |
| 4.38.1.3  | coord_micro                    | 108 |
| 4.38.1.4  | level                          | 108 |
| 4.38.1.5  | sim_time                       | 108 |
| 4.38.1.6  | t_old                          | 108 |
| 4.38.1.7  | t                              | 108 |
| 4.38.1.8  | t_counter                      | 108 |
| 4.38.1.9  | t_print                        | 108 |
| 4.38.1.10 | Print2File                     | 108 |
| 4.38.1.11 | Print2Console                  | 108 |
| 4.38.1.12 | SurfDiff                       | 108 |
| 4.38.1.13 | Heterogeneous                  | 108 |
| 4.38.1.14 | gas_velocity                   | 108 |
| 4.38.1.15 | total_pressure                 | 108 |
| 4.38.1.16 | gas_temperature                | 108 |
| 4.38.1.17 | pellet_radius                  | 108 |
| 4.38.1.18 | crystal_radius                 | 108 |
| 4.38.1.19 | char_macro                     | 108 |
| 4.38.1.20 | char_micro                     | 108 |
| 4.38.1.21 | binder_fraction                | 108 |
| 4.38.1.22 | binder_porosity                | 108 |
| 4.38.1.23 | binder_poresize                | 108 |
| 4.38.1.24 | pellet_density                 | 108 |
| 4.38.1.25 | DirichletBC                    | 108 |
| 4.38.1.26 | NonLinear                      | 108 |
| 4.38.1.27 | y                              | 108 |
| 4.38.1.28 | tempy                          | 109 |
| 4.38.1.29 | OutputFile                     | 109 |
| 4.38.1.30 | eval_ads                       | 109 |

|   |     |
|---|-----|
| 4.38.1.31 eval_retard . . . . .                   | 109 |
| 4.38.1.32 eval_diff . . . . .                     | 109 |
| 4.38.1.33 eval_surfDiff . . . . .                 | 109 |
| 4.38.1.34 eval_kf . . . . .                       | 109 |
| 4.38.1.35 user_data . . . . .                     | 109 |
| 4.38.1.36 gas_dat . . . . .                       | 109 |
| 4.38.1.37 magpie_dat . . . . .                    | 109 |
| 4.38.1.38 finch_dat . . . . .                     | 109 |
| 4.38.1.39 param_dat . . . . .                     | 109 |
| 4.38.1.40 skua_dat . . . . .                      | 109 |
| 4.39 SCOPSOWL_OPT_DATA Struct Reference . . . . . | 109 |
| 4.39.1 Member Data Documentation . . . . .        | 110 |
| 4.39.1.1 num_curves . . . . .                     | 110 |
| 4.39.1.2 evaluation . . . . .                     | 110 |
| 4.39.1.3 total_eval . . . . .                     | 110 |
| 4.39.1.4 current_points . . . . .                 | 110 |
| 4.39.1.5 num_params . . . . .                     | 110 |
| 4.39.1.6 diffusion_type . . . . .                 | 110 |
| 4.39.1.7 adsorb_index . . . . .                   | 110 |
| 4.39.1.8 max_guess_iter . . . . .                 | 110 |
| 4.39.1.9 Optimize . . . . .                       | 110 |
| 4.39.1.10 Rough . . . . .                         | 110 |
| 4.39.1.11 current_temp . . . . .                  | 110 |
| 4.39.1.12 current_press . . . . .                 | 110 |
| 4.39.1.13 current_equil . . . . .                 | 110 |
| 4.39.1.14 simulation_equil . . . . .              | 110 |
| 4.39.1.15 max_bias . . . . .                      | 110 |
| 4.39.1.16 min_bias . . . . .                      | 110 |
| 4.39.1.17 e_norm . . . . .                        | 110 |
| 4.39.1.18 f_bias . . . . .                        | 110 |
| 4.39.1.19 e_norm_old . . . . .                    | 110 |
| 4.39.1.20 f_bias_old . . . . .                    | 110 |
| 4.39.1.21 param_guess . . . . .                   | 111 |
| 4.39.1.22 param_guess_old . . . . .               | 111 |
| 4.39.1.23 rel_tol_norm . . . . .                  | 111 |
| 4.39.1.24 abs_tol_bias . . . . .                  | 111 |
| 4.39.1.25 y_base . . . . .                        | 111 |
| 4.39.1.26 q_data . . . . .                        | 111 |
| 4.39.1.27 q_sim . . . . .                         | 111 |
| 4.39.1.28 t . . . . .                             | 111 |

|   |     |
|---|-----|
| 4.39.1.29 ParamFile . . . . .                       | 111 |
| 4.39.1.30 CompareFile . . . . .                     | 111 |
| 4.39.1.31 owl_dat . . . . .                         | 111 |
| 4.40 SCOPSOWL_PARAM_DATA Struct Reference . . . . . | 111 |
| 4.40.1 Member Data Documentation . . . . .          | 112 |
| 4.40.1.1 qAvg . . . . .                             | 112 |
| 4.40.1.2 qAvg_old . . . . .                         | 112 |
| 4.40.1.3 Qst . . . . .                              | 112 |
| 4.40.1.4 Qst_old . . . . .                          | 112 |
| 4.40.1.5 dq_dc . . . . .                            | 112 |
| 4.40.1.6 xIC . . . . .                              | 112 |
| 4.40.1.7 qIntegralAvg . . . . .                     | 112 |
| 4.40.1.8 qIntegralAvg_old . . . . .                 | 112 |
| 4.40.1.9 QstAvg . . . . .                           | 112 |
| 4.40.1.10 QstAvg_old . . . . .                      | 112 |
| 4.40.1.11 qo . . . . .                              | 112 |
| 4.40.1.12 Qsto . . . . .                            | 112 |
| 4.40.1.13 dq_dco . . . . .                          | 112 |
| 4.40.1.14 pore_diffusion . . . . .                  | 112 |
| 4.40.1.15 film_transfer . . . . .                   | 112 |
| 4.40.1.16 activation_energy . . . . .               | 112 |
| 4.40.1.17 ref_diffusion . . . . .                   | 112 |
| 4.40.1.18 ref_temperature . . . . .                 | 112 |
| 4.40.1.19 affinity . . . . .                        | 112 |
| 4.40.1.20 ref_pressure . . . . .                    | 112 |
| 4.40.1.21 Adsorbable . . . . .                      | 112 |
| 4.40.1.22 speciesName . . . . .                     | 112 |
| 4.41 SHARK_DATA Struct Reference . . . . .          | 112 |
| 4.41.1 Member Data Documentation . . . . .          | 114 |
| 4.41.1.1 MasterList . . . . .                       | 114 |
| 4.41.1.2 ReactionList . . . . .                     | 114 |
| 4.41.1.3 MassBalanceList . . . . .                  | 114 |
| 4.41.1.4 UnsteadyList . . . . .                     | 114 |
| 4.41.1.5 OtherList . . . . .                        | 114 |
| 4.41.1.6 numvar . . . . .                           | 114 |
| 4.41.1.7 num_ssr . . . . .                          | 114 |
| 4.41.1.8 num_mbe . . . . .                          | 114 |
| 4.41.1.9 num_usr . . . . .                          | 114 |
| 4.41.1.10 num_other . . . . .                       | 114 |
| 4.41.1.11 act_fun . . . . .                         | 114 |

|                                      |     |
|--------------------------------------|-----|
| 4.41.1.12 totalsteps . . . . .       | 114 |
| 4.41.1.13 timesteps . . . . .        | 114 |
| 4.41.1.14 pH_index . . . . .         | 114 |
| 4.41.1.15 pOH_index . . . . .        | 114 |
| 4.41.1.16 simulationtime . . . . .   | 114 |
| 4.41.1.17 dt . . . . .               | 114 |
| 4.41.1.18 dt_min . . . . .           | 114 |
| 4.41.1.19 t_out . . . . .            | 114 |
| 4.41.1.20 t_count . . . . .          | 114 |
| 4.41.1.21 time . . . . .             | 114 |
| 4.41.1.22 time_old . . . . .         | 114 |
| 4.41.1.23 pH . . . . .               | 114 |
| 4.41.1.24 Norm . . . . .             | 114 |
| 4.41.1.25 dielectric_const . . . . . | 114 |
| 4.41.1.26 temperature . . . . .      | 114 |
| 4.41.1.27 steadystate . . . . .      | 114 |
| 4.41.1.28 TimeAdaptivity . . . . .   | 115 |
| 4.41.1.29 const_pH . . . . .         | 115 |
| 4.41.1.30 SpeciationCurve . . . . .  | 115 |
| 4.41.1.31 Console_Output . . . . .   | 115 |
| 4.41.1.32 File_Output . . . . .      | 115 |
| 4.41.1.33 Contains_pH . . . . .      | 115 |
| 4.41.1.34 Contains_pOH . . . . .     | 115 |
| 4.41.1.35 Converged . . . . .        | 115 |
| 4.41.1.36 X_old . . . . .            | 115 |
| 4.41.1.37 X_new . . . . .            | 115 |
| 4.41.1.38 Conc_old . . . . .         | 115 |
| 4.41.1.39 Conc_new . . . . .         | 115 |
| 4.41.1.40 activity_new . . . . .     | 115 |
| 4.41.1.41 activity_old . . . . .     | 115 |
| 4.41.1.42 EvalActivity . . . . .     | 115 |
| 4.41.1.43 Residual . . . . .         | 115 |
| 4.41.1.44 lin_precon . . . . .       | 115 |
| 4.41.1.45 Newton_data . . . . .      | 115 |
| 4.41.1.46 activity_data . . . . .    | 115 |
| 4.41.1.47 residual_data . . . . .    | 115 |
| 4.41.1.48 precon_data . . . . .      | 115 |
| 4.41.1.49 other_data . . . . .       | 115 |
| 4.41.1.50 OutputFile . . . . .       | 115 |
| 4.41.1.51 yaml_object . . . . .      | 115 |

|   |     |
|---|-----|
| 4.42 SKUA_DATA Struct Reference . . . . .     | 115 |
| 4.42.1 Member Data Documentation . . . . .    | 116 |
| 4.42.1.1 total_steps . . . . .                | 116 |
| 4.42.1.2 coord . . . . .                      | 116 |
| 4.42.1.3 sim_time . . . . .                   | 116 |
| 4.42.1.4 t_old . . . . .                      | 116 |
| 4.42.1.5 t . . . . .                          | 116 |
| 4.42.1.6 t_counter . . . . .                  | 116 |
| 4.42.1.7 t_print . . . . .                    | 116 |
| 4.42.1.8 qTn . . . . .                        | 116 |
| 4.42.1.9 qTnp1 . . . . .                      | 116 |
| 4.42.1.10 Print2File . . . . .                | 116 |
| 4.42.1.11 Print2Console . . . . .             | 116 |
| 4.42.1.12 gas_velocity . . . . .              | 116 |
| 4.42.1.13 pellet_radius . . . . .             | 117 |
| 4.42.1.14 char_measure . . . . .              | 117 |
| 4.42.1.15 DirichletBC . . . . .               | 117 |
| 4.42.1.16 NonLinear . . . . .                 | 117 |
| 4.42.1.17 y . . . . .                         | 117 |
| 4.42.1.18 OutputFile . . . . .                | 117 |
| 4.42.1.19 eval_diff . . . . .                 | 117 |
| 4.42.1.20 eval_kf . . . . .                   | 117 |
| 4.42.1.21 user_data . . . . .                 | 117 |
| 4.42.1.22 magpie_dat . . . . .                | 117 |
| 4.42.1.23 gas_dat . . . . .                   | 117 |
| 4.42.1.24 finch_dat . . . . .                 | 117 |
| 4.42.1.25 param_dat . . . . .                 | 117 |
| 4.43 SKUA_OPT_DATA Struct Reference . . . . . | 117 |
| 4.43.1 Member Data Documentation . . . . .    | 118 |
| 4.43.1.1 num_curves . . . . .                 | 118 |
| 4.43.1.2 evaluation . . . . .                 | 118 |
| 4.43.1.3 total_eval . . . . .                 | 118 |
| 4.43.1.4 current_points . . . . .             | 118 |
| 4.43.1.5 num_params . . . . .                 | 118 |
| 4.43.1.6 diffusion_type . . . . .             | 118 |
| 4.43.1.7 adsorb_index . . . . .               | 118 |
| 4.43.1.8 max_guess_iter . . . . .             | 118 |
| 4.43.1.9 Optimize . . . . .                   | 118 |
| 4.43.1.10 Rough . . . . .                     | 118 |
| 4.43.1.11 current_temp . . . . .              | 118 |

|  |     |
|--|-----|
| 4.43.1.12 current_press . . . . .                      | 118 |
| 4.43.1.13 current_equil . . . . .                      | 118 |
| 4.43.1.14 simulation_equil . . . . .                   | 118 |
| 4.43.1.15 max_bias . . . . .                           | 118 |
| 4.43.1.16 min_bias . . . . .                           | 118 |
| 4.43.1.17 e_norm . . . . .                             | 118 |
| 4.43.1.18 f_bias . . . . .                             | 118 |
| 4.43.1.19 e_norm_old . . . . .                         | 118 |
| 4.43.1.20 f_bias_old . . . . .                         | 118 |
| 4.43.1.21 param_guess . . . . .                        | 119 |
| 4.43.1.22 param_guess_old . . . . .                    | 119 |
| 4.43.1.23 rel_tol_norm . . . . .                       | 119 |
| 4.43.1.24 abs_tol_bias . . . . .                       | 119 |
| 4.43.1.25 y_base . . . . .                             | 119 |
| 4.43.1.26 q_data . . . . .                             | 119 |
| 4.43.1.27 q_sim . . . . .                              | 119 |
| 4.43.1.28 t . . . . .                                  | 119 |
| 4.43.1.29 ParamFile . . . . .                          | 119 |
| 4.43.1.30 CompareFile . . . . .                        | 119 |
| 4.43.1.31 skua_dat . . . . .                           | 119 |
| 4.44 SKUA_PARAM Struct Reference . . . . .             | 119 |
| 4.44.1 Member Data Documentation . . . . .             | 119 |
| 4.44.1.1 activation_energy . . . . .                   | 119 |
| 4.44.1.2 ref_diffusion . . . . .                       | 119 |
| 4.44.1.3 ref_temperature . . . . .                     | 120 |
| 4.44.1.4 affinity . . . . .                            | 120 |
| 4.44.1.5 ref_pressure . . . . .                        | 120 |
| 4.44.1.6 film_transfer . . . . .                       | 120 |
| 4.44.1.7 xIC . . . . .                                 | 120 |
| 4.44.1.8 y_eff . . . . .                               | 120 |
| 4.44.1.9 Qstn . . . . .                                | 120 |
| 4.44.1.10 Qstnp1 . . . . .                             | 120 |
| 4.44.1.11 xn . . . . .                                 | 120 |
| 4.44.1.12 xnp1 . . . . .                               | 120 |
| 4.44.1.13 Adsorbable . . . . .                         | 120 |
| 4.44.1.14 speciesName . . . . .                        | 120 |
| 4.45 Speciation_Test01_Data Struct Reference . . . . . | 120 |
| 4.45.1 Member Data Documentation . . . . .             | 120 |
| 4.45.1.1 N . . . . .                                   | 120 |
| 4.45.1.2 logKw . . . . .                               | 120 |

|           |  |     |
|-----------|--|-----|
| 4.45.1.3  | logKa1                                 | 120 |
| 4.45.1.4  | logKa2                                 | 121 |
| 4.45.1.5  | CT                                     | 121 |
| 4.45.1.6  | NaT                                    | 121 |
| 4.45.1.7  | x                                      | 121 |
| 4.45.1.8  | Jacobian                               | 121 |
| 4.45.1.9  | NumJac                                 | 121 |
| 4.45.1.10 | logC                                   | 121 |
| 4.45.1.11 | C                                      | 121 |
| 4.46      | SubHeader Class Reference              | 121 |
| 4.46.1    | Constructor & Destructor Documentation | 122 |
| 4.46.1.1  | SubHeader                              | 122 |
| 4.46.1.2  | ~SubHeader                             | 122 |
| 4.46.1.3  | SubHeader                              | 122 |
| 4.46.1.4  | SubHeader                              | 122 |
| 4.46.1.5  | SubHeader                              | 122 |
| 4.46.1.6  | SubHeader                              | 122 |
| 4.46.2    | Member Function Documentation          | 122 |
| 4.46.2.1  | operator=                              | 122 |
| 4.46.2.2  | operator[]                             | 122 |
| 4.46.2.3  | operator[]                             | 122 |
| 4.46.2.4  | getMap                                 | 122 |
| 4.46.2.5  | clear                                  | 122 |
| 4.46.2.6  | addPair                                | 122 |
| 4.46.2.7  | addPair                                | 122 |
| 4.46.2.8  | setName                                | 122 |
| 4.46.2.9  | setAlias                               | 122 |
| 4.46.2.10 | setAlias                               | 122 |
| 4.46.2.11 | setNameAliasPair                       | 122 |
| 4.46.2.12 | setState                               | 122 |
| 4.46.2.13 | DisplayContents                        | 122 |
| 4.46.2.14 | getName                                | 123 |
| 4.46.2.15 | getAlias                               | 123 |
| 4.46.2.16 | isAlias                                | 123 |
| 4.46.2.17 | isAnchor                               | 123 |
| 4.46.2.18 | getState                               | 123 |
| 4.46.3    | Member Data Documentation              | 123 |
| 4.46.3.1  | Data_Map                               | 123 |
| 4.46.3.2  | name                                   | 123 |
| 4.46.3.3  | alias                                  | 123 |



|           |                                  |     |
|-----------|----------------------------------|-----|
| 4.46.3.4  | state                            | 123 |
| 4.47      | SYSTEM_DATA Struct Reference     | 123 |
| 4.47.1    | Detailed Description             | 124 |
| 4.47.2    | Member Data Documentation        | 124 |
| 4.47.2.1  | T                                | 124 |
| 4.47.2.2  | PT                               | 124 |
| 4.47.2.3  | qT                               | 124 |
| 4.47.2.4  | PI                               | 124 |
| 4.47.2.5  | pi                               | 124 |
| 4.47.2.6  | As                               | 124 |
| 4.47.2.7  | N                                | 125 |
| 4.47.2.8  | I                                | 125 |
| 4.47.2.9  | J                                | 125 |
| 4.47.2.10 | K                                | 125 |
| 4.47.2.11 | total_eval                       | 125 |
| 4.47.2.12 | avg_norm                         | 125 |
| 4.47.2.13 | max_norm                         | 125 |
| 4.47.2.14 | Sys                              | 125 |
| 4.47.2.15 | Par                              | 125 |
| 4.47.2.16 | Recover                          | 125 |
| 4.47.2.17 | Carrier                          | 125 |
| 4.47.2.18 | Ideal                            | 125 |
| 4.47.2.19 | Output                           | 125 |
| 4.48      | TRAJECTORY_DATA Struct Reference | 126 |
| 4.48.1    | Member Data Documentation        | 127 |
| 4.48.1.1  | mu_0                             | 127 |
| 4.48.1.2  | rho_f                            | 127 |
| 4.48.1.3  | eta                              | 127 |
| 4.48.1.4  | Hamaker                          | 127 |
| 4.48.1.5  | Temp                             | 127 |
| 4.48.1.6  | k                                | 127 |
| 4.48.1.7  | Rs                               | 127 |
| 4.48.1.8  | L                                | 127 |
| 4.48.1.9  | porosity                         | 127 |
| 4.48.1.10 | V_separator                      | 127 |
| 4.48.1.11 | a                                | 127 |
| 4.48.1.12 | V_wire                           | 127 |
| 4.48.1.13 | L_wire                           | 127 |
| 4.48.1.14 | A_separator                      | 127 |
| 4.48.1.15 | A_wire                           | 127 |

|                                  |     |
|----------------------------------|-----|
| 4.48.1.16 B0                     | 127 |
| 4.48.1.17 H0                     | 127 |
| 4.48.1.18 Ms                     | 127 |
| 4.48.1.19 b                      | 127 |
| 4.48.1.20 chi_p                  | 127 |
| 4.48.1.21 rho_p                  | 127 |
| 4.48.1.22 Q_in                   | 127 |
| 4.48.1.23 V0                     | 127 |
| 4.48.1.24 Y_initial              | 127 |
| 4.48.1.25 dt                     | 127 |
| 4.48.1.26 M                      | 127 |
| 4.48.1.27 mp                     | 127 |
| 4.48.1.28 beta                   | 128 |
| 4.48.1.29 q_bar                  | 128 |
| 4.48.1.30 sigma_v                | 128 |
| 4.48.1.31 sigma_vz               | 128 |
| 4.48.1.32 sigma_z                | 128 |
| 4.48.1.33 sigma_n                | 128 |
| 4.48.1.34 sigma_m                | 128 |
| 4.48.1.35 n_rand                 | 128 |
| 4.48.1.36 m_rand                 | 128 |
| 4.48.1.37 s_rand                 | 128 |
| 4.48.1.38 t_rand                 | 128 |
| 4.48.1.39 POL                    | 128 |
| 4.48.1.40 H                      | 128 |
| 4.48.1.41 dX                     | 128 |
| 4.48.1.42 dY                     | 128 |
| 4.48.1.43 X                      | 128 |
| 4.48.1.44 Y                      | 128 |
| 4.48.1.45 Cap                    | 128 |
| 4.49 UI_DATA Struct Reference    | 128 |
| 4.49.1 Detailed Description      | 129 |
| 4.49.2 Member Data Documentation | 129 |
| 4.49.2.1 value_type              | 129 |
| 4.49.2.2 user_input              | 129 |
| 4.49.2.3 input_files             | 129 |
| 4.49.2.4 path                    | 129 |
| 4.49.2.5 count                   | 129 |
| 4.49.2.6 max                     | 130 |
| 4.49.2.7 option                  | 130 |

|   |     |
|---|-----|
| 4.49.2.8 Path . . . . .                                 | 130 |
| 4.49.2.9 Files . . . . .                                | 130 |
| 4.49.2.10 MissingArg . . . . .                          | 130 |
| 4.49.2.11 BasicUI . . . . .                             | 130 |
| 4.49.2.12 argc . . . . .                                | 130 |
| 4.49.2.13 argv . . . . .                                | 130 |
| 4.50 UnsteadyPrecipitation Class Reference . . . . .    | 130 |
| 4.51 UnsteadyReaction Class Reference . . . . .         | 131 |
| 4.51.1 Constructor & Destructor Documentation . . . . . | 132 |
| 4.51.1.1 UnsteadyReaction . . . . .                     | 132 |
| 4.51.1.2 ~UnsteadyReaction . . . . .                    | 132 |
| 4.51.2 Member Function Documentation . . . . .          | 132 |
| 4.51.2.1 Initialize_List . . . . .                      | 132 |
| 4.51.2.2 Display_Info . . . . .                         | 132 |
| 4.51.2.3 Set_Species_Index . . . . .                    | 132 |
| 4.51.2.4 Set_Species_Index . . . . .                    | 132 |
| 4.51.2.5 Set_Stoichiometric . . . . .                   | 132 |
| 4.51.2.6 Set_Equilibrium . . . . .                      | 132 |
| 4.51.2.7 Set_Enthalpy . . . . .                         | 132 |
| 4.51.2.8 Set_Entropy . . . . .                          | 132 |
| 4.51.2.9 Set_EnthalpyANDEntropy . . . . .               | 132 |
| 4.51.2.10 Set_Energy . . . . .                          | 133 |
| 4.51.2.11 Set_InitialValue . . . . .                    | 133 |
| 4.51.2.12 Set_MaximumValue . . . . .                    | 133 |
| 4.51.2.13 Set_Forward . . . . .                         | 133 |
| 4.51.2.14 Set_Reverse . . . . .                         | 133 |
| 4.51.2.15 Set_ForwardRef . . . . .                      | 133 |
| 4.51.2.16 Set_ReverseRef . . . . .                      | 133 |
| 4.51.2.17 Set_ActivationEnergy . . . . .                | 133 |
| 4.51.2.18 Set_Affinity . . . . .                        | 133 |
| 4.51.2.19 Set_TimeStep . . . . .                        | 133 |
| 4.51.2.20 checkSpeciesEnergies . . . . .                | 133 |
| 4.51.2.21 calculateEnergies . . . . .                   | 133 |
| 4.51.2.22 calculateEquilibrium . . . . .                | 133 |
| 4.51.2.23 calculateRate . . . . .                       | 133 |
| 4.51.2.24 haveEquilibrium . . . . .                     | 133 |
| 4.51.2.25 haveRate . . . . .                            | 133 |
| 4.51.2.26 Get_Species_Index . . . . .                   | 133 |
| 4.51.2.27 Get_Stoichiometric . . . . .                  | 133 |
| 4.51.2.28 Get_Equilibrium . . . . .                     | 133 |

|           |  |     |
|-----------|--|-----|
| 4.51.2.29 | <a href="#">Get_Enthalpy</a>                               | 133 |
| 4.51.2.30 | <a href="#">Get_Entropy</a>                                | 133 |
| 4.51.2.31 | <a href="#">Get_Energy</a>                                 | 133 |
| 4.51.2.32 | <a href="#">Get_InitialValue</a>                           | 133 |
| 4.51.2.33 | <a href="#">Get_MaximumValue</a>                           | 133 |
| 4.51.2.34 | <a href="#">Get_Forward</a>                                | 133 |
| 4.51.2.35 | <a href="#">Get_Reverse</a>                                | 133 |
| 4.51.2.36 | <a href="#">Get_ForwardRef</a>                             | 133 |
| 4.51.2.37 | <a href="#">Get_ReverseRef</a>                             | 133 |
| 4.51.2.38 | <a href="#">Get_ActivationEnergy</a>                       | 134 |
| 4.51.2.39 | <a href="#">Get_Affinity</a>                               | 134 |
| 4.51.2.40 | <a href="#">Get_TimeStep</a>                               | 134 |
| 4.51.2.41 | <a href="#">Eval_ReactionRate</a>                          | 134 |
| 4.51.2.42 | <a href="#">Eval_Residual</a>                              | 134 |
| 4.51.2.43 | <a href="#">Eval_Residual</a>                              | 134 |
| 4.51.2.44 | <a href="#">Eval_IC_Residual</a>                           | 134 |
| 4.51.2.45 | <a href="#">Explicit_Eval</a>                              | 134 |
| 4.51.3    | <a href="#">Member Data Documentation</a>                  | 134 |
| 4.51.3.1  | <a href="#">initial_value</a>                              | 134 |
| 4.51.3.2  | <a href="#">max_value</a>                                  | 134 |
| 4.51.3.3  | <a href="#">forward_rate</a>                               | 134 |
| 4.51.3.4  | <a href="#">reverse_rate</a>                               | 134 |
| 4.51.3.5  | <a href="#">forward_ref_rate</a>                           | 134 |
| 4.51.3.6  | <a href="#">reverse_ref_rate</a>                           | 134 |
| 4.51.3.7  | <a href="#">activation_energy</a>                          | 134 |
| 4.51.3.8  | <a href="#">temperature_affinity</a>                       | 134 |
| 4.51.3.9  | <a href="#">time_step</a>                                  | 134 |
| 4.51.3.10 | <a href="#">HaveForward</a>                                | 134 |
| 4.51.3.11 | <a href="#">HaveReverse</a>                                | 134 |
| 4.51.3.12 | <a href="#">HaveForRef</a>                                 | 134 |
| 4.51.3.13 | <a href="#">HaveRevRef</a>                                 | 134 |
| 4.51.3.14 | <a href="#">species_index</a>                              | 134 |
| 4.52      | <a href="#">ValueTypePair Class Reference</a>              | 134 |
| 4.52.1    | <a href="#">Constructor &amp; Destructor Documentation</a> | 135 |
| 4.52.1.1  | <a href="#">ValueTypePair</a>                              | 135 |
| 4.52.1.2  | <a href="#">~ValueTypePair</a>                             | 135 |
| 4.52.1.3  | <a href="#">ValueTypePair</a>                              | 135 |
| 4.52.1.4  | <a href="#">ValueTypePair</a>                              | 135 |
| 4.52.1.5  | <a href="#">ValueTypePair</a>                              | 135 |
| 4.52.2    | <a href="#">Member Function Documentation</a>              | 135 |

|           |  |     |
|-----------|--|-----|
| 4.52.2.1  | operator=                              | 135 |
| 4.52.2.2  | editValue                              | 135 |
| 4.52.2.3  | editPair                               | 135 |
| 4.52.2.4  | findType                               | 135 |
| 4.52.2.5  | assertType                             | 135 |
| 4.52.2.6  | DisplayPair                            | 135 |
| 4.52.2.7  | getString                              | 135 |
| 4.52.2.8  | getBool                                | 136 |
| 4.52.2.9  | getDouble                              | 136 |
| 4.52.2.10 | getInt                                 | 136 |
| 4.52.2.11 | getValue                               | 136 |
| 4.52.2.12 | getType                                | 136 |
| 4.52.2.13 | getPair                                | 136 |
| 4.52.3    | Member Data Documentation              | 136 |
| 4.52.3.1  | Value_Type                             | 136 |
| 4.52.3.2  | type                                   | 136 |
| 4.53      | yaml_cpp_class Class Reference         | 136 |
| 4.53.1    | Constructor & Destructor Documentation | 136 |
| 4.53.1.1  | yaml_cpp_class                         | 136 |
| 4.53.1.2  | ~yaml_cpp_class                        | 136 |
| 4.53.2    | Member Function Documentation          | 137 |
| 4.53.2.1  | setInputFile                           | 137 |
| 4.53.2.2  | readInputFile                          | 137 |
| 4.53.2.3  | cleanup                                | 137 |
| 4.53.2.4  | executeYamlRead                        | 137 |
| 4.53.2.5  | getYamlWrapper                         | 137 |
| 4.53.2.6  | DisplayContents                        | 137 |
| 4.53.3    | Member Data Documentation              | 137 |
| 4.53.3.1  | yaml_wrapper                           | 137 |
| 4.53.3.2  | input_file                             | 137 |
| 4.53.3.3  | file_name                              | 137 |
| 4.53.3.4  | token_parser                           | 137 |
| 4.53.3.5  | current_token                          | 137 |
| 4.53.3.6  | previous_token                         | 137 |
| 4.54      | YamlWrapper Class Reference            | 137 |
| 4.54.1    | Constructor & Destructor Documentation | 138 |
| 4.54.1.1  | YamlWrapper                            | 138 |
| 4.54.1.2  | ~YamlWrapper                           | 138 |
| 4.54.1.3  | YamlWrapper                            | 138 |
| 4.54.1.4  | YamlWrapper                            | 138 |

|           |                               |            |
|-----------|-------------------------------|------------|
| 4.54.2    | Member Function Documentation | 138        |
| 4.54.2.1  | operator=                     | 138        |
| 4.54.2.2  | operator()                    | 138        |
| 4.54.2.3  | operator()                    | 138        |
| 4.54.2.4  | getDocMap                     | 138        |
| 4.54.2.5  | getDocument                   | 138        |
| 4.54.2.6  | end                           | 138        |
| 4.54.2.7  | end                           | 138        |
| 4.54.2.8  | begin                         | 138        |
| 4.54.2.9  | begin                         | 138        |
| 4.54.2.10 | clear                         | 138        |
| 4.54.2.11 | resetKeys                     | 138        |
| 4.54.2.12 | changeKey                     | 138        |
| 4.54.2.13 | revalidateAllKeys             | 139        |
| 4.54.2.14 | DisplayContents               | 139        |
| 4.54.2.15 | addDockKey                    | 139        |
| 4.54.2.16 | copyAnchor2Alias              | 139        |
| 4.54.2.17 | size                          | 139        |
| 4.54.2.18 | getAnchoredDoc                | 139        |
| 4.54.2.19 | getDocFromHeadAlias           | 139        |
| 4.54.2.20 | getDocFromSubAlias            | 139        |
| 4.54.3    | Member Data Documentation     | 139        |
| 4.54.3.1  | Doc_Map                       | 139        |
| <b>5</b>  | <b>File Documentation</b>     | <b>141</b> |
| 5.1       | dogfish.h File Reference      | 141        |
| 5.1.1     | Detailed Description          | 142        |
| 5.1.2     | Function Documentation        | 142        |
| 5.1.2.1   | print2file_species_header     | 142        |
| 5.1.2.2   | print2file_DOGFISH_header     | 142        |
| 5.1.2.3   | print2file_DOGFISH_result_old | 143        |
| 5.1.2.4   | print2file_DOGFISH_result_new | 143        |
| 5.1.2.5   | default_Retardation           | 143        |
| 5.1.2.6   | default_IntraDiffusion        | 143        |
| 5.1.2.7   | default_FilmMTCoeff           | 143        |
| 5.1.2.8   | default_SurfaceConcentration  | 143        |
| 5.1.2.9   | setup_DOGFISH_DATA            | 143        |
| 5.1.2.10  | DOGFISH_Executioner           | 143        |
| 5.1.2.11  | set_DOGFISH_ICs               | 144        |
| 5.1.2.12  | set_DOGFISH_timestep          | 144        |

|          |                                |     |
|----------|--------------------------------|-----|
| 5.1.2.13 | DOGFISH_preprocesses           | 144 |
| 5.1.2.14 | set_DOGFISH_params             | 144 |
| 5.1.2.15 | DOGFISH_postprocesses          | 144 |
| 5.1.2.16 | DOGFISH_reset                  | 144 |
| 5.1.2.17 | DOGFISH                        | 144 |
| 5.1.2.18 | DOGFISH_TESTS                  | 145 |
| 5.2      | eel.h File Reference           | 145 |
| 5.2.1    | Detailed Description           | 145 |
| 5.2.2    | Function Documentation         | 146 |
| 5.2.2.1  | EEL_TESTS                      | 146 |
| 5.3      | egret.h File Reference         | 146 |
| 5.3.1    | Detailed Description           | 147 |
| 5.3.2    | Macro Definition Documentation | 148 |
| 5.3.2.1  | Rstd                           | 148 |
| 5.3.2.2  | RE3                            | 148 |
| 5.3.2.3  | Po                             | 148 |
| 5.3.2.4  | Cstd                           | 148 |
| 5.3.2.5  | CE3                            | 148 |
| 5.3.2.6  | Pstd                           | 148 |
| 5.3.2.7  | PE3                            | 148 |
| 5.3.2.8  | Nu                             | 148 |
| 5.3.2.9  | PSI                            | 148 |
| 5.3.2.10 | Dp_ij                          | 148 |
| 5.3.2.11 | D_ij                           | 148 |
| 5.3.2.12 | Mu                             | 148 |
| 5.3.2.13 | D_ii                           | 149 |
| 5.3.2.14 | ReNum                          | 149 |
| 5.3.2.15 | ScNum                          | 149 |
| 5.3.2.16 | FilmMTCoeff                    | 149 |
| 5.3.3    | Function Documentation         | 149 |
| 5.3.3.1  | initialize_data                | 149 |
| 5.3.3.2  | set_variables                  | 149 |
| 5.3.3.3  | calculate_properties           | 149 |
| 5.3.3.4  | EGRET_TESTS                    | 149 |
| 5.4      | error.h File Reference         | 150 |
| 5.4.1    | Detailed Description           | 150 |
| 5.4.2    | Macro Definition Documentation | 151 |
| 5.4.2.1  | mError                         | 151 |
| 5.4.3    | Enumeration Type Documentation | 151 |
| 5.4.3.1  | error_type                     | 151 |

|          |                                |     |
|----------|--------------------------------|-----|
| 5.4.4    | Function Documentation         | 152 |
| 5.4.4.1  | error                          | 152 |
| 5.5      | finch.h File Reference         | 152 |
| 5.5.1    | Detailed Description           | 154 |
| 5.5.2    | Enumeration Type Documentation | 155 |
| 5.5.2.1  | finch_solve_type               | 155 |
| 5.5.2.2  | finch_coord_type               | 155 |
| 5.5.3    | Function Documentation         | 155 |
| 5.5.3.1  | max                            | 155 |
| 5.5.3.2  | min                            | 156 |
| 5.5.3.3  | minmod                         | 156 |
| 5.5.3.4  | uTotal                         | 156 |
| 5.5.3.5  | uAverage                       | 156 |
| 5.5.3.6  | check_Mass                     | 156 |
| 5.5.3.7  | l_direct                       | 156 |
| 5.5.3.8  | lark_picard_step               | 156 |
| 5.5.3.9  | nl_picard                      | 156 |
| 5.5.3.10 | setup_FINCH_DATA               | 156 |
| 5.5.3.11 | print2file_dim_header          | 157 |
| 5.5.3.12 | print2file_time_header         | 157 |
| 5.5.3.13 | print2file_result_old          | 157 |
| 5.5.3.14 | print2file_result_new          | 157 |
| 5.5.3.15 | print2file_newline             | 157 |
| 5.5.3.16 | print2file_tab                 | 157 |
| 5.5.3.17 | default_execution              | 157 |
| 5.5.3.18 | default_ic                     | 157 |
| 5.5.3.19 | default_timestep               | 157 |
| 5.5.3.20 | default_preprocess             | 158 |
| 5.5.3.21 | default_solve                  | 158 |
| 5.5.3.22 | default_params                 | 158 |
| 5.5.3.23 | minmod_discretization          | 158 |
| 5.5.3.24 | vanAlbada_discretization       | 158 |
| 5.5.3.25 | ospre_discretization           | 158 |
| 5.5.3.26 | default_bcs                    | 158 |
| 5.5.3.27 | default_res                    | 158 |
| 5.5.3.28 | default_precon                 | 159 |
| 5.5.3.29 | default_postprocess            | 159 |
| 5.5.3.30 | default_reset                  | 159 |
| 5.5.3.31 | FINCH_TESTS                    | 159 |
| 5.6      | flock.h File Reference         | 159 |



|          |                                |     |
|----------|--------------------------------|-----|
| 5.6.1    | Detailed Description           | 159 |
| 5.7      | gsta_opt.h File Reference      | 160 |
| 5.7.1    | Detailed Description           | 161 |
| 5.7.2    | Macro Definition Documentation | 162 |
| 5.7.2.1  | Po                             | 162 |
| 5.7.2.2  | R                              | 162 |
| 5.7.2.3  | Na                             | 162 |
| 5.7.3    | Function Documentation         | 162 |
| 5.7.3.1  | roundIt                        | 162 |
| 5.7.3.2  | twoFifths                      | 162 |
| 5.7.3.3  | orderMag                       | 162 |
| 5.7.3.4  | minValue                       | 162 |
| 5.7.3.5  | minIndex                       | 162 |
| 5.7.3.6  | avgPar                         | 163 |
| 5.7.3.7  | avgValue                       | 163 |
| 5.7.3.8  | weightedAvg                    | 163 |
| 5.7.3.9  | rSq                            | 163 |
| 5.7.3.10 | isSmooth                       | 163 |
| 5.7.3.11 | orthoLinReg                    | 163 |
| 5.7.3.12 | eduGuess                       | 164 |
| 5.7.3.13 | gstaFunc                       | 164 |
| 5.7.3.14 | gstaObjFunc                    | 164 |
| 5.7.3.15 | eval_GSTA                      | 164 |
| 5.7.3.16 | gsta_optimize                  | 164 |
| 5.8      | lark.h File Reference          | 165 |
| 5.8.1    | Detailed Description           | 168 |
| 5.8.2    | Enumeration Type Documentation | 169 |
| 5.8.2.1  | krylov_method                  | 169 |
| 5.8.3    | Function Documentation         | 169 |
| 5.8.3.1  | update_arnoldi_solution        | 169 |
| 5.8.3.2  | arnoldi                        | 169 |
| 5.8.3.3  | gmresLeftPreconditioned        | 170 |
| 5.8.3.4  | fom                            | 171 |
| 5.8.3.5  | gmresRightPreconditioned       | 172 |
| 5.8.3.6  | pcg                            | 172 |
| 5.8.3.7  | bicgstab                       | 173 |
| 5.8.3.8  | cgs                            | 174 |
| 5.8.3.9  | operatorTranspose              | 174 |
| 5.8.3.10 | gcr                            | 175 |
| 5.8.3.11 | gmresPreconditioner            | 175 |

|           |                                |     |
|-----------|--------------------------------|-----|
| 5.8.3.12  | gmresr                         | 176 |
| 5.8.3.13  | picard                         | 176 |
| 5.8.3.14  | jacvec                         | 177 |
| 5.8.3.15  | backtrackLineSearch            | 177 |
| 5.8.3.16  | pjfnk                          | 178 |
| 5.8.3.17  | NumericalJacobian              | 179 |
| 5.8.3.18  | LARK_TESTS                     | 179 |
| 5.9       | macaw.h File Reference         | 179 |
| 5.9.1     | Detailed Description           | 180 |
| 5.9.2     | Macro Definition Documentation | 180 |
| 5.9.2.1   | M_PI                           | 180 |
| 5.9.3     | Function Documentation         | 180 |
| 5.9.3.1   | MACAW_TESTS                    | 180 |
| 5.10      | magpie.h File Reference        | 181 |
| 5.10.1    | Detailed Description           | 182 |
| 5.10.2    | Macro Definition Documentation | 183 |
| 5.10.2.1  | DBL_EPSILON                    | 183 |
| 5.10.2.2  | Z                              | 183 |
| 5.10.2.3  | A                              | 183 |
| 5.10.2.4  | V                              | 183 |
| 5.10.2.5  | Po                             | 183 |
| 5.10.2.6  | R                              | 183 |
| 5.10.2.7  | Na                             | 183 |
| 5.10.2.8  | kB                             | 183 |
| 5.10.2.9  | shapeFactor                    | 183 |
| 5.10.2.10 | lnKo                           | 184 |
| 5.10.2.11 | He                             | 184 |
| 5.10.3    | Function Documentation         | 184 |
| 5.10.3.1  | qo                             | 184 |
| 5.10.3.2  | dq_dp                          | 184 |
| 5.10.3.3  | q_p                            | 184 |
| 5.10.3.4  | PI                             | 184 |
| 5.10.3.5  | Qst                            | 185 |
| 5.10.3.6  | eMax                           | 185 |
| 5.10.3.7  | lnact_mSPD                     | 185 |
| 5.10.3.8  | grad_mSPD                      | 185 |
| 5.10.3.9  | qT                             | 186 |
| 5.10.3.10 | initialGuess_mSPD              | 186 |
| 5.10.3.11 | eval_po_PI                     | 186 |
| 5.10.3.12 | eval_po_qo                     | 186 |

|  |     |
|--|-----|
| 5.10.3.13 eval_po . . . . .                        | 187 |
| 5.10.3.14 eval_eta . . . . .                       | 187 |
| 5.10.3.15 eval_GPAST . . . . .                     | 187 |
| 5.10.3.16 MAGPIE . . . . .                         | 187 |
| 5.10.3.17 MAGPIE_SCENARIOS . . . . .               | 188 |
| 5.11 mola.h File Reference . . . . .               | 190 |
| 5.11.1 Detailed Description . . . . .              | 191 |
| 5.11.2 Function Documentation . . . . .            | 192 |
| 5.11.2.1 MOLA_TESTS . . . . .                      | 192 |
| 5.12 monkfish.h File Reference . . . . .           | 192 |
| 5.12.1 Function Documentation . . . . .            | 193 |
| 5.12.1.1 default_porosity . . . . .                | 193 |
| 5.12.1.2 default_density . . . . .                 | 193 |
| 5.12.1.3 default_interparticle_diffusion . . . . . | 193 |
| 5.12.1.4 default_monk_adsorption . . . . .         | 193 |
| 5.12.1.5 default_monk_equilibrium . . . . .        | 193 |
| 5.12.1.6 default_monkfish_retardation . . . . .    | 193 |
| 5.12.1.7 default_exterior_concentration . . . . .  | 193 |
| 5.12.1.8 default_film_transfer . . . . .           | 193 |
| 5.12.1.9 setup_MONKFISH_DATA . . . . .             | 193 |
| 5.12.1.10 MONKFISH_TESTS . . . . .                 | 193 |
| 5.13 sandbox.h File Reference . . . . .            | 193 |
| 5.13.1 Function Documentation . . . . .            | 194 |
| 5.13.1.1 Speciation_Test01_Function . . . . .      | 194 |
| 5.13.1.2 Speciation_Test01_Jacobian . . . . .      | 194 |
| 5.13.1.3 Speciation_Test01_Guess . . . . .         | 194 |
| 5.13.1.4 Speciation_Test01_MatVec . . . . .        | 194 |
| 5.13.1.5 RUN_SANDBOX . . . . .                     | 194 |
| 5.14 school.h File Reference . . . . .             | 194 |
| 5.15 scopsowl.h File Reference . . . . .           | 194 |
| 5.15.1 Macro Definition Documentation . . . . .    | 195 |
| 5.15.1.1 SCOPSOWL_HPP_ . . . . .                   | 195 |
| 5.15.1.2 Dp . . . . .                              | 195 |
| 5.15.1.3 Dk . . . . .                              | 195 |
| 5.15.1.4 avgDp . . . . .                           | 195 |
| 5.15.2 Function Documentation . . . . .            | 195 |
| 5.15.2.1 print2file_species_header . . . . .       | 195 |
| 5.15.2.2 print2file_SCOPSOWL_time_header . . . . . | 195 |
| 5.15.2.3 print2file_SCOPSOWL_header . . . . .      | 195 |
| 5.15.2.4 print2file_SCOPSOWL_result_old . . . . .  | 196 |

|           |                                |     |
|-----------|--------------------------------|-----|
| 5.15.2.5  | print2file_SCOPSOWL_result_new | 196 |
| 5.15.2.6  | default_adsorption             | 196 |
| 5.15.2.7  | default_retardation            | 196 |
| 5.15.2.8  | default_pore_diffusion         | 196 |
| 5.15.2.9  | default_surf_diffusion         | 196 |
| 5.15.2.10 | default_effective_diffusion    | 196 |
| 5.15.2.11 | const_pore_diffusion           | 196 |
| 5.15.2.12 | default_filmMassTransfer       | 196 |
| 5.15.2.13 | const_filmMassTransfer         | 196 |
| 5.15.2.14 | setup_SCOPSOWL_DATA            | 196 |
| 5.15.2.15 | SCOPSOWL_Executioner           | 196 |
| 5.15.2.16 | set_SCOPSOWL_ICs               | 196 |
| 5.15.2.17 | set_SCOPSOWL_timestep          | 196 |
| 5.15.2.18 | SCOPSOWL_preprocesses          | 196 |
| 5.15.2.19 | set_SCOPSOWL_params            | 196 |
| 5.15.2.20 | SCOPSOWL_postprocesses         | 196 |
| 5.15.2.21 | SCOPSOWL_reset                 | 196 |
| 5.15.2.22 | SCOPSOWL                       | 196 |
| 5.15.2.23 | LARGE_CYCLE_TEST01             | 196 |
| 5.15.2.24 | SMALL_CYCLE_TEST02             | 196 |
| 5.15.2.25 | CURVE_TEST03                   | 196 |
| 5.15.2.26 | CURVE_TEST04                   | 196 |
| 5.15.2.27 | CURVE_TEST05                   | 196 |
| 5.15.2.28 | SCOPSOWL_SCENARIOS             | 196 |
| 5.15.2.29 | SCOPSOWL_TESTS                 | 196 |
| 5.16      | scopsowl_opt.h File Reference  | 197 |
| 5.16.1    | Function Documentation         | 197 |
| 5.16.1.1  | SCOPSOWL_OPT_set_y             | 197 |
| 5.16.1.2  | initial_guess_SCOPSOWL         | 197 |
| 5.16.1.3  | eval_SCOPSOWL_Uptake           | 197 |
| 5.16.1.4  | SCOPSOWL_OPTIMIZE              | 197 |
| 5.17      | shark.h File Reference         | 197 |
| 5.17.1    | Macro Definition Documentation | 199 |
| 5.17.1.1  | Rstd                           | 199 |
| 5.17.2    | Typedef Documentation          | 199 |
| 5.17.2.1  | SHARK_DATA                     | 199 |
| 5.17.3    | Enumeration Type Documentation | 199 |
| 5.17.3.1  | valid_act                      | 199 |
| 5.17.4    | Function Documentation         | 199 |
| 5.17.4.1  | print2file_shark_info          | 199 |

|           |                                |     |
|-----------|--------------------------------|-----|
| 5.17.4.2  | print2file_shark_header        | 199 |
| 5.17.4.3  | print2file_shark_results_new   | 199 |
| 5.17.4.4  | print2file_shark_results_old   | 199 |
| 5.17.4.5  | ideal_solution                 | 199 |
| 5.17.4.6  | Davies_equation                | 199 |
| 5.17.4.7  | DebyeHuckel_equation           | 199 |
| 5.17.4.8  | DaviesLadshaw_equation         | 199 |
| 5.17.4.9  | act_choice                     | 199 |
| 5.17.4.10 | linsearch_choice               | 199 |
| 5.17.4.11 | linearsolve_choice             | 199 |
| 5.17.4.12 | Convert2LogConcentration       | 199 |
| 5.17.4.13 | Convert2Concentration          | 199 |
| 5.17.4.14 | read_scenario                  | 199 |
| 5.17.4.15 | read_options                   | 199 |
| 5.17.4.16 | read_species                   | 199 |
| 5.17.4.17 | read_massbalance               | 200 |
| 5.17.4.18 | read_equilrxn                  | 200 |
| 5.17.4.19 | read_unsteadyrxn               | 200 |
| 5.17.4.20 | setup_SHARK_DATA               | 200 |
| 5.17.4.21 | shark_add_customResidual       | 200 |
| 5.17.4.22 | shark_parameter_check          | 200 |
| 5.17.4.23 | shark_energy_calculations      | 200 |
| 5.17.4.24 | shark_temperature_calculations | 200 |
| 5.17.4.25 | shark_pH_finder                | 200 |
| 5.17.4.26 | shark_guess                    | 200 |
| 5.17.4.27 | shark_initial_conditions       | 200 |
| 5.17.4.28 | shark_executioner              | 200 |
| 5.17.4.29 | shark_timestep_const           | 200 |
| 5.17.4.30 | shark_timestep_adapt           | 200 |
| 5.17.4.31 | shark_preprocesses             | 200 |
| 5.17.4.32 | shark_solver                   | 200 |
| 5.17.4.33 | shark_postprocesses            | 200 |
| 5.17.4.34 | shark_reset                    | 200 |
| 5.17.4.35 | shark_residual                 | 200 |
| 5.17.4.36 | SHARK                          | 200 |
| 5.17.4.37 | SHARK_SCENARIO                 | 200 |
| 5.17.4.38 | SHARK_TESTS                    | 200 |
| 5.18      | skua.h File Reference          | 200 |
| 5.18.1    | Macro Definition Documentation | 201 |
| 5.18.1.1  | SKUA_HPP_                      | 201 |

|           |                             |     |
|-----------|-----------------------------|-----|
| 5.18.1.2  | D_inf                       | 201 |
| 5.18.1.3  | D_o                         | 201 |
| 5.18.1.4  | D_c                         | 201 |
| 5.18.2    | Function Documentation      | 202 |
| 5.18.2.1  | print2file_species_header   | 202 |
| 5.18.2.2  | print2file_SKUA_time_header | 202 |
| 5.18.2.3  | print2file_SKUA_header      | 202 |
| 5.18.2.4  | print2file_SKUA_results_old | 202 |
| 5.18.2.5  | print2file_SKUA_results_new | 202 |
| 5.18.2.6  | default_Dc                  | 202 |
| 5.18.2.7  | default_kf                  | 202 |
| 5.18.2.8  | const_Dc                    | 202 |
| 5.18.2.9  | simple_darken_Dc            | 202 |
| 5.18.2.10 | theoretical_darken_Dc       | 202 |
| 5.18.2.11 | empirical_kf                | 202 |
| 5.18.2.12 | const_kf                    | 202 |
| 5.18.2.13 | molefractionCheck           | 202 |
| 5.18.2.14 | setup_SKUA_DATA             | 202 |
| 5.18.2.15 | SKUA_Executioner            | 202 |
| 5.18.2.16 | set_SKUA_ICs                | 202 |
| 5.18.2.17 | set_SKUA_timestep           | 202 |
| 5.18.2.18 | SKUA_preprocesses           | 202 |
| 5.18.2.19 | set_SKUA_params             | 202 |
| 5.18.2.20 | SKUA_postprocesses          | 202 |
| 5.18.2.21 | SKUA_reset                  | 202 |
| 5.18.2.22 | SKUA                        | 202 |
| 5.18.2.23 | SKUA_CYCLE_TEST01           | 202 |
| 5.18.2.24 | SKUA_CYCLE_TEST02           | 202 |
| 5.18.2.25 | SKUA_LOW_TEST03             | 202 |
| 5.18.2.26 | SKUA_MID_TEST04             | 202 |
| 5.18.2.27 | SKUA_SCENARIOS              | 202 |
| 5.18.2.28 | SKUA_TESTS                  | 203 |
| 5.19      | skua_opt.h File Reference   | 203 |
| 5.19.1    | Function Documentation      | 203 |
| 5.19.1.1  | SKUA_OPT_set_y              | 203 |
| 5.19.1.2  | initial_guess_SKUA          | 203 |
| 5.19.1.3  | eval_SKUA_Uptake            | 203 |
| 5.19.1.4  | SKUA_OPTIMIZE               | 203 |
| 5.20      | Trajectory.h File Reference | 203 |
| 5.20.1    | Function Documentation      | 204 |

|           |                                |     |
|-----------|--------------------------------|-----|
| 5.20.1.1  | Magnetic_R                     | 204 |
| 5.20.1.2  | Magnetic_T                     | 204 |
| 5.20.1.3  | Grav_R                         | 204 |
| 5.20.1.4  | Grav_T                         | 204 |
| 5.20.1.5  | Van_R                          | 204 |
| 5.20.1.6  | V_RAD                          | 204 |
| 5.20.1.7  | V_THETA                        | 204 |
| 5.20.1.8  | Brown_RAD                      | 204 |
| 5.20.1.9  | Brown_THETA                    | 204 |
| 5.20.1.10 | POLAR                          | 204 |
| 5.20.1.11 | RADIAL_FORCE                   | 204 |
| 5.20.1.12 | TANGENTIAL_FORCE               | 204 |
| 5.20.1.13 | CARTESIAN                      | 205 |
| 5.20.1.14 | DISPLACEMENT                   | 205 |
| 5.20.1.15 | LOCATION                       | 205 |
| 5.20.1.16 | Removal_Efficiency             | 205 |
| 5.20.1.17 | Trajectory_SetupConstants      | 205 |
| 5.20.1.18 | Number_Generator               | 205 |
| 5.20.1.19 | Run_Trajectory                 | 205 |
| 5.21      | ui.h File Reference            | 205 |
| 5.21.1    | Detailed Description           | 207 |
| 5.21.2    | Macro Definition Documentation | 207 |
| 5.21.2.1  | UI_HPP_                        | 207 |
| 5.21.2.2  | ECO_VERSION                    | 207 |
| 5.21.2.3  | ECO_EXECUTABLE                 | 207 |
| 5.21.3    | Enumeration Type Documentation | 207 |
| 5.21.3.1  | valid_options                  | 207 |
| 5.21.4    | Function Documentation         | 208 |
| 5.21.4.1  | au_i_help                      | 208 |
| 5.21.4.2  | bui_help                       | 208 |
| 5.21.4.3  | allLower                       | 208 |
| 5.21.4.4  | exit                           | 208 |
| 5.21.4.5  | help                           | 209 |
| 5.21.4.6  | version                        | 209 |
| 5.21.4.7  | test                           | 209 |
| 5.21.4.8  | exec                           | 209 |
| 5.21.4.9  | path                           | 209 |
| 5.21.4.10 | input                          | 210 |
| 5.21.4.11 | valid_test_string              | 210 |
| 5.21.4.12 | valid_exec_string              | 210 |

|           |                                |     |
|-----------|--------------------------------|-----|
| 5.21.4.13 | number_files                   | 210 |
| 5.21.4.14 | valid_addon_options            | 210 |
| 5.21.4.15 | display_help                   | 211 |
| 5.21.4.16 | display_version                | 211 |
| 5.21.4.17 | invalid_input                  | 211 |
| 5.21.4.18 | valid_input_main               | 211 |
| 5.21.4.19 | valid_input_tests              | 212 |
| 5.21.4.20 | valid_input_execute            | 212 |
| 5.21.4.21 | test_loop                      | 212 |
| 5.21.4.22 | exec_loop                      | 212 |
| 5.21.4.23 | run_test                       | 212 |
| 5.21.4.24 | run_exec                       | 213 |
| 5.21.4.25 | run_executable                 | 213 |
| 5.22      | yaml_wrapper.h File Reference  | 213 |
| 5.22.1    | Typedef Documentation          | 214 |
| 5.22.1.1  | data_type                      | 214 |
| 5.22.1.2  | header_state                   | 214 |
| 5.22.2    | Enumeration Type Documentation | 214 |
| 5.22.2.1  | data_type                      | 214 |
| 5.22.2.2  | header_state                   | 214 |
| 5.22.3    | Function Documentation         | 214 |
| 5.22.3.1  | YAML_WRAPPER_TESTS             | 214 |
| 5.22.3.2  | YAML_CPP_TEST                  | 214 |



## Chapter 1

# Hierarchical Index

### 1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

|                             |     |
|-----------------------------|-----|
| ARNOLDI_DATA . . . . .      | 7   |
| Atom . . . . .              | 9   |
| BACKTRACK_DATA . . . . .    | 14  |
| BiCGSTAB_DATA . . . . .     | 15  |
| CGS_DATA . . . . .          | 19  |
| DOGFISH_DATA . . . . .      | 25  |
| DOGFISH_PARAM . . . . .     | 28  |
| FINCH_DATA . . . . .        | 29  |
| GCR_DATA . . . . .          | 42  |
| GMRESLP_DATA . . . . .      | 45  |
| GMRESR_DATA . . . . .       | 47  |
| GMRESRP_DATA . . . . .      | 50  |
| GPAST_DATA . . . . .        | 54  |
| GSTA_DATA . . . . .         | 55  |
| GSTA_OPT_DATA . . . . .     | 56  |
| KeyValueMap . . . . .       | 61  |
| MAGPIE_DATA . . . . .       | 63  |
| MassBalance . . . . .       | 64  |
| Matrix< T > . . . . .       | 67  |
| Matrix< double > . . . . .  | 67  |
| Matrix< int > . . . . .     | 67  |
| Mechanism . . . . .         | 76  |
| MIXED_GAS . . . . .         | 76  |
| Molecule . . . . .          | 79  |
| MasterSpeciesList . . . . . | 65  |
| MONKFISH_DATA . . . . .     | 86  |
| MONKFISH_PARAM . . . . .    | 88  |
| mSPD_DATA . . . . .         | 89  |
| NUM_JAC_DATA . . . . .      | 90  |
| OPTRANS_DATA . . . . .      | 91  |
| PCG_DATA . . . . .          | 91  |
| PeriodicTable . . . . .     | 94  |
| PICARD_DATA . . . . .       | 96  |
| PJFNK_DATA . . . . .        | 98  |
| PURE_GAS . . . . .          | 103 |
| Reaction . . . . .          | 104 |
| Precipitation . . . . .     | 102 |

|                                  |     |
|----------------------------------|-----|
| UnsteadyPrecipitation . . . . .  | 130 |
| UnsteadyReaction . . . . .       | 131 |
| SCOPSOWL_DATA . . . . .          | 107 |
| SCOPSOWL_OPT_DATA . . . . .      | 109 |
| SCOPSOWL_PARAM_DATA . . . . .    | 111 |
| SHARK_DATA . . . . .             | 112 |
| SKUA_DATA . . . . .              | 115 |
| SKUA_OPT_DATA . . . . .          | 117 |
| SKUA_PARAM . . . . .             | 119 |
| Speciation_Test01_Data . . . . . | 120 |
| SubHeader . . . . .              | 121 |
| Document . . . . .               | 22  |
| Header . . . . .                 | 58  |
| SYSTEM_DATA . . . . .            | 123 |
| TRAJECTORY_DATA . . . . .        | 126 |
| UI_DATA . . . . .                | 128 |
| ValueTypePair . . . . .          | 134 |
| yaml_cpp_class . . . . .         | 136 |
| YamlWrapper . . . . .            | 137 |

## Chapter 2

# Class Index

### 2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

|   |    |
|---|----|
| <a href="#">ARNOLDI_DATA</a>  |    |
| Data structure for the construction of the Krylov subspaces for a linear system . . . . .   | 7  |
| <a href="#">Atom</a>  |    |
| <a href="#">Atom</a> object to hold information about specific atoms in the periodic table (click <a href="#">Atom</a> to go to function definitions) . . . . . | 9  |
| <a href="#">BACKTRACK_DATA</a>  |    |
| Data structure for the implementation of Backtracking Linesearch . . . . .  | 14 |
| <a href="#">BiCGSTAB_DATA</a>   |    |
| Data structure for the implementation of the BiCGSTAB algorithm for non-symmetric linear systems . . . . .  | 15 |
| <a href="#">CGS_DATA</a>  |    |
| Data structure for the implementation of the CGS algorithm for non-symmetric linear systems .   | 19 |
| <a href="#">Document</a> . . . . .  | 22 |
| <a href="#">DOGFISH_DATA</a>  |    |
| Primary data structure for running the DOGFISH application . . . . .  | 25 |
| <a href="#">DOGFISH_PARAM</a>   |    |
| Data structure for species-specific parameters . . . . .  | 28 |
| <a href="#">FINCH_DATA</a>  |    |
| Data structure for the FINCH object . . . . .   | 29 |
| <a href="#">GCR_DATA</a>  |    |
| Data structure for the implementation of the GCR algorithm for non-symmetric linear systems .   | 42 |
| <a href="#">GMRESLP_DATA</a>  |    |
| Data structure for implementation of the Restarted GMRES algorithm with Left Preconditioning  | 45 |
| <a href="#">GMRESR_DATA</a>   |    |
| Data structure for the implementation of GCR with Nested GMRES preconditioning (i.e., GMRESR) . . . . .   | 47 |
| <a href="#">GMRESRP_DATA</a>  |    |
| Data structure for the Restarted GMRES algorithm with Right Preconditioning . . . . .   | 50 |
| <a href="#">GPAST_DATA</a>  |    |
| GPAST Data Structure . . . . .  | 54 |
| <a href="#">GSTA_DATA</a>   |    |
| GSTA Data Structure . . . . .   | 55 |
| <a href="#">GSTA_OPT_DATA</a>   |    |
| Data structure used in the GSTA optimization routines . . . . .   | 56 |
| <a href="#">Header</a> . . . . .  | 58 |
| <a href="#">KeyValueMap</a> . . . . .   | 61 |
| <a href="#">MAGPIE_DATA</a>   |    |
| MAGPIE Data Structure . . . . .   | 63 |

|  |     |
|--|-----|
| <a href="#">MassBalance</a>  | 64  |
| <a href="#">MasterSpeciesList</a>  | 65  |
| <a href="#">Matrix&lt; T &gt;</a>  |     |
| Templated C++ <a href="#">Matrix</a> Class Object (click <a href="#">Matrix</a> to go to function definitions)                             | 67  |
| <a href="#">Mechanism</a>  | 76  |
| <a href="#">MIXED_GAS</a>  |     |
| Data structure holding information necessary for computing mixed gas properties  | 76  |
| <a href="#">Molecule</a>   |     |
| C++ <a href="#">Molecule</a> Object built from <a href="#">Atom</a> Objects (click <a href="#">Molecule</a> to go to function definitions) | 79  |
| <a href="#">MONKFISH_DATA</a>  | 86  |
| <a href="#">MONKFISH_PARAM</a>   | 88  |
| <a href="#">mSPD_DATA</a>  |     |
| MSPD Data Structure  | 89  |
| <a href="#">NUM_JAC_DATA</a>   |     |
| Data structure to form a numerical jacobian matrix with finite differences   | 90  |
| <a href="#">OPTRANS_DATA</a>   |     |
| Data structure for implementation of linear operator transposition   | 91  |
| <a href="#">PCG_DATA</a>   |     |
| Data structure for implementation of the PCG algorithms for symmetric linear systems   | 91  |
| <a href="#">PeriodicTable</a>  |     |
| Class object that store a digital copy of all <a href="#">Atom</a> objects   | 94  |
| <a href="#">PICARD_DATA</a>  |     |
| Data structure for the implementation of a Picard or Fixed-Point iteration for non-linear systems  | 96  |
| <a href="#">PJFNK_DATA</a>   |     |
| Data structure for the implementation of the PJFNK algorithm for non-linear systems  | 98  |
| <a href="#">Precipitation</a>  | 102 |
| <a href="#">PURE_GAS</a>   |     |
| Data structure holding all the parameters for each pure gas spieces  | 103 |
| <a href="#">Reaction</a>   | 104 |
| <a href="#">SCOPSOWL_DATA</a>  | 107 |
| <a href="#">SCOPSOWL_OPT_DATA</a>  | 109 |
| <a href="#">SCOPSOWL_PARAM_DATA</a>  | 111 |
| <a href="#">SHARK_DATA</a>   | 112 |
| <a href="#">SKUA_DATA</a>  | 115 |
| <a href="#">SKUA_OPT_DATA</a>  | 117 |
| <a href="#">SKUA_PARAM</a>   | 119 |
| <a href="#">Speciation_Test01_Data</a>   | 120 |
| <a href="#">SubHeader</a>  | 121 |
| <a href="#">SYSTEM_DATA</a>  |     |
| System Data Structure  | 123 |
| <a href="#">TRAJECTORY_DATA</a>  | 126 |
| <a href="#">UI_DATA</a>  |     |
| Data structure holding the UI arguments  | 128 |
| <a href="#">UnsteadyPrecipitation</a>  | 130 |
| <a href="#">UnsteadyReaction</a>   | 131 |
| <a href="#">ValueTypePair</a>  | 134 |
| <a href="#">yaml_cpp_class</a>   | 136 |
| <a href="#">YamlWrapper</a>  | 137 |

## Chapter 3

# File Index

### 3.1 File List

Here is a list of all files with brief descriptions:

|                                |  |     |
|--------------------------------|--|-----|
| <a href="#">dogfish.h</a>      | Diffusion Object Governing Fiber Interior Sorption History . . . . .                   | 141 |
| <a href="#">eel.h</a>          | Easy-access Element Library . . . . .  | 145 |
| <a href="#">egret.h</a>        | Estimation of Gas-phase pRopErTies . . . . .   | 146 |
| <a href="#">error.h</a>        | All error types are defined here . . . . .   | 150 |
| <a href="#">finch.h</a>        | Flux-limiting Implicit Non-oscillatory Conservative High-resolution scheme . . . . .   | 152 |
| <a href="#">flock.h</a>        | FundamentaL Off-gas Collection of Kernels . . . . .                                    | 159 |
| <a href="#">gsta_opt.h</a>     | Generalized Statistical Thermodynamic Adsorption (GSTA) Optimization Routine . . . . . | 160 |
| <a href="#">lark.h</a>         | Linear Algebra Residual Kernels . . . . .  | 165 |
| <a href="#">macaw.h</a>        | MAtrix CAlculation Workspace . . . . .   | 179 |
| <a href="#">magpie.h</a>       | Multicomponent Adsorption Generalized Procedure for Isothermal Equilibria . . . . .    | 181 |
| <a href="#">mola.h</a>         | <a href="#">Molecule</a> Object Library from Atoms . . . . .                           | 190 |
| <a href="#">monkfish.h</a>     | . . . . .  | 192 |
| <a href="#">sandbox.h</a>      | . . . . .  | 193 |
| <a href="#">school.h</a>       | . . . . .  | 194 |
| <a href="#">scopsowl.h</a>     | . . . . .  | 194 |
| <a href="#">scopsowl_opt.h</a> | . . . . .  | 197 |
| <a href="#">shark.h</a>        | . . . . .  | 197 |
| <a href="#">skua.h</a>         | . . . . .  | 200 |
| <a href="#">skua_opt.h</a>     | . . . . .  | 203 |
| <a href="#">Trajectory.h</a>   | . . . . .  | 203 |
| <a href="#">ui.h</a>           | User Interface for Ecosystem . . . . .   | 205 |
| <a href="#">yaml_wrapper.h</a> | . . . . .  | 213 |



## Chapter 4

# Class Documentation

### 4.1 ARNOLDI\_DATA Struct Reference

Data structure for the construction of the Krylov subspaces for a linear system.

```
#include <lark.h>
```

#### Public Attributes

- int `k`  
*Desired size of the Krylov subspace.*
- int `iter`  
*Actual size of the Krylov subspace.*
- double `beta`  
*Normalization parameter.*
- double `hp1`  
*Additional row element of  $H$  (separate storage for holding)*
- bool `Output` = true  
*True = print messages to console.*
- `std::vector< Matrix< double > > Vk`  
 *$(N) \times (k)$  orthonormal vector basis stored as a vector of column matrices*
- `Matrix< double > Hkp1`  
 *$(k+1) \times (k)$  upper Hessenberg matrix*
- `Matrix< double > yk`  
 *$(k) \times (1)$  vector search direction*
- `Matrix< double > e1`  
 *$(k) \times (1)$  orthonormal vector with 1 in first position*
- `Matrix< double > w`  
 *$(N) \times (1)$  interim result of the matrix\_vector multiplication*
- `Matrix< double > v`  
 *$(N) \times (1)$  holding cell for the column entries of  $V_k$  and other interims*
- `Matrix< double > sum`  
 *$(N) \times (1)$  running sum of subspace vectors for use in altering  $w$*

### 4.1.1 Detailed Description

Data structure for the construction of the Krylov subspaces for a linear system.

C-style object used in conjunction with the Arnoldi algorithm to construct an orthonormal basis and upper Hessenberg representation of a given linear operator. This is used to solve a linear system both iteratively (i.e., in conjunction with GMRESLP) and directly (i.e., in conjunction with FOM). Alternatively, you can just store the factorized components for later use in another routine.

### 4.1.2 Member Data Documentation

#### 4.1.2.1 `int ARNOLDI_DATA::k`

Desired size of the Krylov subspace.

#### 4.1.2.2 `int ARNOLDI_DATA::iter`

Actual size of the Krylov subspace.

#### 4.1.2.3 `double ARNOLDI_DATA::beta`

Normalization parameter.

#### 4.1.2.4 `double ARNOLDI_DATA::hp1`

Additional row element of H (separate storage for holding)

#### 4.1.2.5 `bool ARNOLDI_DATA::Output = true`

True = print messages to console.

#### 4.1.2.6 `std::vector< Matrix<double> > ARNOLDI_DATA::Vk`

(N) x (k) orthonormal vector basis stored as a vector of column matrices

#### 4.1.2.7 `Matrix<double> ARNOLDI_DATA::Hkp1`

(k+1) x (k) upper Hessenberg matrix

#### 4.1.2.8 `Matrix<double> ARNOLDI_DATA::yk`

(k) x (1) vector search direction

#### 4.1.2.9 `Matrix<double> ARNOLDI_DATA::e1`

(k) x (1) orthonormal vector with 1 in first position

#### 4.1.2.10 `Matrix<double> ARNOLDI_DATA::w`

(N) x (1) interim result of the matrix\_vector multiplication



4.1.2.11 `Matrix<double> ARNOLDI_DATA::v`

(N) x (1) holding cell for the column entries of  $V_k$  and other interims

4.1.2.12 `Matrix<double> ARNOLDI_DATA::sum`

(N) x (1) running sum of subspace vectors for use in altering  $w$

The documentation for this struct was generated from the following file:

- [lark.h](#)

## 4.2 Atom Class Reference

`Atom` object to hold information about specific atoms in the periodic table (click [Atom](#) to go to function definitions)

```
#include <eel.h>
```

### Public Member Functions

- [Atom](#) ()  
*Default Constructor.*
- [~Atom](#) ()  
*Default Destructor.*
- [Atom](#) (std::string [Name](#))  
*Constructor by [Atom](#) Name.*
- [Atom](#) (int number)  
*Constructor by Atomic number.*
- void [Register](#) (std::string [Symbol](#))  
*Register an atom object by symbol.*
- void [Register](#) (int number)  
*Register an atom object by number.*
- void [editAtomicWeight](#) (double AW)  
*Manually changes the atomic weight.*
- void [editOxidationState](#) (int state)  
*Manually changes the oxidation state.*
- void [editProtons](#) (int proton)  
*Manually changes the number of protons.*
- void [editNeutrons](#) (int neutron)  
*Manually changes the number of neutrons.*
- void [editElectrons](#) (int electron)  
*Manually changes the number of electrons.*
- void [editValence](#) (int val)  
*Manually changes the number of valence electrons.*
- void [removeProton](#) ()  
*Manually removes 1 proton and adjusts weight.*
- void [removeNeutron](#) ()  
*Manually removes 1 neutron and adjusts weight.*
- void [removeElectron](#) ()  
*Manually removes 1 electron from valence.*
- double [AtomicWeight](#) ()

- Returns the current atomic weight (g/mol)*
- int [OxidationState](#) ()  
*Returns the current oxidation state.*
- int [Protons](#) ()  
*Returns the current number of protons.*
- int [Neutrons](#) ()  
*Returns the current number of neutrons.*
- int [Electrons](#) ()  
*Returns the current number of electrons.*
- int [BondingElectrons](#) ()  
*Returns the number of electrons available for bonding.*
- std::string [AtomName](#) ()  
*Returns the name of the atom.*
- std::string [AtomSymbol](#) ()  
*Returns the symbol of the atom.*
- std::string [AtomCategory](#) ()  
*Returns the category of the atom.*
- std::string [AtomState](#) ()  
*Returns the state of the atom.*
- int [AtomicNumber](#) ()  
*Returns the atomic number of the atom.*
- void [DisplayInfo](#) ()  
*Displays [Atom](#) information to console.*

### Protected Attributes

- double [atomic\\_weight](#)  
*Holds the atomic weight of the atom.*
- int [oxidation\\_state](#)  
*Holds the oxidation state of the atom.*
- int [protons](#)  
*Holds the number of protons in the atom.*
- int [neutrons](#)  
*Holds the number of neutrons in the atom.*
- int [electrons](#)  
*Holds the number of electrons in the atom.*
- int [valence\\_e](#)  
*Holds the number of valence electrons in the atom.*

### Private Attributes

- std::string [Name](#)  
*Holds the name of the atom.*
- std::string [Symbol](#)  
*Holds the atomic symbol for the atom.*
- std::string [Category](#)  
*Holds the category of the atom (e.g., Alkali Metal)*
- std::string [NaturalState](#)  
*Holds the natural state of the atom (e.g., Gas)*
- int [atomic\\_number](#)  
*Holds the atomic number of the atom.*

### 4.2.1 Detailed Description

[Atom](#) object to hold information about specific atoms in the periodic table (click [Atom](#) to go to function definitions)

C++ class object holding data and functions associated with atoms. Objects can be registered at the time of object construction, or after declaring an [Atom](#) object. Registration can be done via the atomic symbol or atomic number. Valid atoms go from Hydrogen (1) to Ununoctium (118).

### 4.2.2 Constructor & Destructor Documentation

#### 4.2.2.1 `Atom::Atom ( )`

Default Constructor.

#### 4.2.2.2 `Atom::~Atom ( )`

Default Destructor.

#### 4.2.2.3 `Atom::Atom ( std::string Name )`

Constructor by [Atom](#) Name.

#### 4.2.2.4 `Atom::Atom ( int number )`

Constructor by Atomic number.

### 4.2.3 Member Function Documentation

#### 4.2.3.1 `void Atom::Register ( std::string Symbol )`

Register an atom object by symbol.

#### 4.2.3.2 `void Atom::Register ( int number )`

Register an atom object by number.

#### 4.2.3.3 `void Atom::editAtomicWeight ( double AW )`

Manually changes the atomic weight.

#### 4.2.3.4 `void Atom::editOxidationState ( int state )`

Manually changes the oxidation state.

#### 4.2.3.5 `void Atom::editProtons ( int proton )`

Manually changes the number of protons.

#### 4.2.3.6 `void Atom::editNeutrons ( int neutron )`

Manually changes the number of neutrons.

**4.2.3.7 void Atom::editElectrons ( int *electron* )**

Manually changes the number of electrons.

**4.2.3.8 void Atom::editValence ( int *val* )**

Manually changes the number of valence electrons.

**4.2.3.9 void Atom::removeProton ( )**

Manually removes 1 proton and adjusts weight.

**4.2.3.10 void Atom::removeNeutron ( )**

Manually removes 1 neutron and adjusts weight.

**4.2.3.11 void Atom::removeElectron ( )**

Manually removes 1 electron from valence.

**4.2.3.12 double Atom::AtomicWeight ( )**

Returns the current atomic weight (g/mol)

**4.2.3.13 int Atom::OxidationState ( )**

Returns the current oxidation state.

**4.2.3.14 int Atom::Protons ( )**

Returns the current number of protons.

**4.2.3.15 int Atom::Neutrons ( )**

Returns the current number of neutrons.

**4.2.3.16 int Atom::Electrons ( )**

Returns the current number of electrons.

**4.2.3.17 int Atom::BondingElectrons ( )**

Returns the number of electrons available for bonding.

**4.2.3.18 std::string Atom::AtomName ( )**

Returns the name of the atom.

**4.2.3.19** `std::string Atom::AtomSymbol ( )`

Returns the symbol of the atom.

**4.2.3.20** `std::string Atom::AtomCategory ( )`

Returns the category of the atom.

**4.2.3.21** `std::string Atom::AtomState ( )`

Returns the state of the atom.

**4.2.3.22** `int Atom::AtomicNumber ( )`

Returns the atomic number of the atom.

**4.2.3.23** `void Atom::DisplayInfo ( )`

Displays [Atom](#) information to console.

**4.2.4 Member Data Documentation****4.2.4.1** `double Atom::atomic_weight` `[protected]`

Holds the atomic weight of the atom.

**4.2.4.2** `int Atom::oxidation_state` `[protected]`

Holds the oxidation state of the atom.

**4.2.4.3** `int Atom::protons` `[protected]`

Holds the number of protons in the atom.

**4.2.4.4** `int Atom::neutrons` `[protected]`

Holds the number of neutrons in the atom.

**4.2.4.5** `int Atom::electrons` `[protected]`

Holds the number of electrons in the atom.

**4.2.4.6** `int Atom::valence_e` `[protected]`

Holds the number of valence electrons in the atom.

**4.2.4.7** `std::string Atom::Name` `[private]`

Holds the name of the atom.

#### 4.2.4.8 `std::string Atom::Symbol` [private]

Holds the atomic symbol for the atom.

#### 4.2.4.9 `std::string Atom::Category` [private]

Holds the category of the atom (e.g., Alkali Metal)

#### 4.2.4.10 `std::string Atom::NaturalState` [private]

Holds the natural state of the atom (e.g., Gas)

#### 4.2.4.11 `int Atom::atomic_number` [private]

Holds the atomic number of the atom.

The documentation for this class was generated from the following file:

- [eel.h](#)

### 4.3 BACKTRACK\_DATA Struct Reference

Data structure for the implementation of Backtracking Linesearch.

```
#include <lark.h>
```

#### Public Attributes

- double `alpha` = 1e-4  
*Scaling parameter for determination of search step size.*
- double `rho` = 0.1  
*Scaling parameter for to change step size by.*
- double `lambdaMin` = DBL\_EPSILON  
*Smallest allowable step length.*
- double `normFkp1`  
*New residual norm of the Newton step.*
- bool `constRho` = false  
*True = use a constant value for rho.*
- `Matrix< double > Fk`  
*Old residual vector of the Newton step.*
- `Matrix< double > xk`  
*Old solution vector of the Newton step.*

#### 4.3.1 Detailed Description

Data structure for the implementation of Backtracking Linesearch.

C-style object used in conjunction with the Backtracking Linesearch algorithm to smooth out convergence of Newton based iterative methods for non-linear systems of equations. The actual algorithm has been separated from the interior of the Newton method so that it can be included in any future Newton based iterative methods being developed.

### 4.3.2 Member Data Documentation

4.3.2.1 `double BACKTRACK_DATA::alpha = 1e-4`

Scaling parameter for determination of search step size.

4.3.2.2 `double BACKTRACK_DATA::rho = 0.1`

Scaling parameter for to change step size by.

4.3.2.3 `double BACKTRACK_DATA::lambdaMin = DBL_EPSILON`

Smallest allowable step length.

4.3.2.4 `double BACKTRACK_DATA::normFkp1`

New residual norm of the Newton step.

4.3.2.5 `bool BACKTRACK_DATA::constRho = false`

True = use a constant value for rho.

4.3.2.6 `Matrix<double> BACKTRACK_DATA::Fk`

Old residual vector of the Newton step.

4.3.2.7 `Matrix<double> BACKTRACK_DATA::xk`

Old solution vector of the Newton step.

The documentation for this struct was generated from the following file:

- [lark.h](#)

## 4.4 BiCGSTAB\_DATA Struct Reference

Data structure for the implementation of the BiCGSTAB algorithm for non-symmetric linear systems.

```
#include <lark.h>
```

### Public Attributes

- `int maxit = 0`  
*Maximum allowable iterations - default = min(2\*vector\_size, 1000)*
- `int iter = 0`  
*Actual number of iterations.*
- `bool breakdown`  
*Boolean to determine if the method broke down.*
- `double alpha`  
*Step size parameter for next solution.*
- `double beta`

- Step size parameter for search direction.*
- double [rho](#)
- Scaling parameter for alpha and beta.*
- double [rho\\_old](#)
- Previous scaling parameter for alpha and beta.*
- double [omega](#)
- Scaling parameter and additional step length.*
- double [omega\\_old](#)
- Previous scaling parameter and step length.*
- double [tol\\_rel](#) = 1e-6
- Relative tolerance for convergence - default = 1e-6.*
- double [tol\\_abs](#) = 1e-6
- Absolute tolerance for convergence - default = 1e-6.*
- double [res](#)
- Absolute residual norm.*
- double [relres](#)
- Relative residual norm.*
- double [relres\\_base](#)
- Initial residual norm.*
- double [bestres](#)
- Best found residual norm.*
- bool [Output](#) = true
- True = print messages to console.*
- [Matrix](#)< double > [x](#)
- Current solution to the linear system.*
- [Matrix](#)< double > [bestx](#)
- Best found solution to the linear system.*
- [Matrix](#)< double > [r](#)
- Residual vector for the linear system.*
- [Matrix](#)< double > [r0](#)
- Initial residual vector.*
- [Matrix](#)< double > [v](#)
- Search direction for p.*
- [Matrix](#)< double > [p](#)
- Search direction for updating.*
- [Matrix](#)< double > [y](#)
- Preconditioned search direction.*
- [Matrix](#)< double > [s](#)
- Residual updating vector.*
- [Matrix](#)< double > [z](#)
- Preconditioned residual updating vector.*
- [Matrix](#)< double > [t](#)
- Search direction for residual updates.*

#### 4.4.1 Detailed Description

Data structure for the implementation of the BiCGSTAB algorithm for non-symmetric linear systems.

C-style object used in conjunction with the Bi-Conjugate Gradient STABalized (BiCGSTAB) algorithm to solve a linear system of equations. This algorithm is generally more efficient than any GMRES or GCR variant, but may not always reduce the residual at each step. However, if used with preconditioning, then this algorithm is very efficient, especially when used for solving grid-based linear systems.



#### 4.4.2 Member Data Documentation

##### 4.4.2.1 int BiCGSTAB\_DATA::maxit = 0

Maximum allowable iterations - default =  $\min(2 \times \text{vector\_size}, 1000)$

##### 4.4.2.2 int BiCGSTAB\_DATA::iter = 0

Actual number of iterations.

##### 4.4.2.3 bool BiCGSTAB\_DATA::breakdown

Boolean to determine if the method broke down.

##### 4.4.2.4 double BiCGSTAB\_DATA::alpha

Step size parameter for next solution.

##### 4.4.2.5 double BiCGSTAB\_DATA::beta

Step size parameter for search direction.

##### 4.4.2.6 double BiCGSTAB\_DATA::rho

Scaling parameter for alpha and beta.

##### 4.4.2.7 double BiCGSTAB\_DATA::rho\_old

Previous scaling parameter for alpha and beta.

##### 4.4.2.8 double BiCGSTAB\_DATA::omega

Scaling parameter and additional step length.

##### 4.4.2.9 double BiCGSTAB\_DATA::omega\_old

Previous scaling parameter and step length.

##### 4.4.2.10 double BiCGSTAB\_DATA::tol\_rel = 1e-6

Relative tolerance for convergence - default =  $1e-6$ .

##### 4.4.2.11 double BiCGSTAB\_DATA::tol\_abs = 1e-6

Absolution tolerance for convergence - default =  $1e-6$ .

##### 4.4.2.12 double BiCGSTAB\_DATA::res

Absolute residual norm.

**4.4.2.13 double BiCGSTAB\_DATA::relres**

Relative residual norm.

**4.4.2.14 double BiCGSTAB\_DATA::relres\_base**

Initial residual norm.

**4.4.2.15 double BiCGSTAB\_DATA::bestres**

Best found residual norm.

**4.4.2.16 bool BiCGSTAB\_DATA::Output = true**

True = print messages to console.

**4.4.2.17 Matrix<double> BiCGSTAB\_DATA::x**

Current solution to the linear system.

**4.4.2.18 Matrix<double> BiCGSTAB\_DATA::bestx**

Best found solution to the linear system.

**4.4.2.19 Matrix<double> BiCGSTAB\_DATA::r**

Residual vector for the linear system.

**4.4.2.20 Matrix<double> BiCGSTAB\_DATA::r0**

Initial residual vector.

**4.4.2.21 Matrix<double> BiCGSTAB\_DATA::v**

Search direction for p.

**4.4.2.22 Matrix<double> BiCGSTAB\_DATA::p**

Search direction for updating.

**4.4.2.23 Matrix<double> BiCGSTAB\_DATA::y**

Preconditioned search direction.

**4.4.2.24 Matrix<double> BiCGSTAB\_DATA::s**

Residual updating vector.

## 4.4.2.25 Matrix&lt;double&gt; BiCGSTAB\_DATA::z

Preconditioned residual updating vector.

## 4.4.2.26 Matrix&lt;double&gt; BiCGSTAB\_DATA::t

Search direction for residual updates.

The documentation for this struct was generated from the following file:

- [lark.h](#)

## 4.5 CGS\_DATA Struct Reference

Data structure for the implementation of the CGS algorithm for non-symmetric linear systems.

```
#include <lark.h>
```

### Public Attributes

- int [maxit](#) = 0  
*Maximum allowable iterations - default = min(2\*vector\_size, 1000)*
- int [iter](#) = 0  
*Actual number of iterations.*
- bool [breakdown](#)  
*Boolean to determine if the method broke down.*
- double [alpha](#)  
*Step size parameter for next solution.*
- double [beta](#)  
*Step size parameter for search direction.*
- double [rho](#)  
*Scaling parameter for alpha and beta.*
- double [sigma](#)  
*Scaling parameter and additional step length.*
- double [tol\\_rel](#) = 1e-6  
*Relative tolerance for convergence - default = 1e-6.*
- double [tol\\_abs](#) = 1e-6  
*Absolute tolerance for convergence - default = 1e-6.*
- double [res](#)  
*Absolute residual norm.*
- double [relres](#)  
*Relative residual norm.*
- double [relres\\_base](#)  
*Initial residual norm.*
- double [bestres](#)  
*Best found residual norm.*
- bool [Output](#) = true  
*True = print messages to console.*
- Matrix< double > [x](#)  
*Current solution to the linear system.*
- Matrix< double > [bestx](#)

- *Best found solution to the linear system.*
- `Matrix< double > r`  
*Residual vector for the linear system.*
- `Matrix< double > r0`  
*Initial residual vector.*
- `Matrix< double > u`  
*Search direction for v.*
- `Matrix< double > w`  
*Updates sigma and u.*
- `Matrix< double > v`  
*Search direction for x.*
- `Matrix< double > p`  
*Preconditioning result for w, z, and matvec for Ax.*
- `Matrix< double > c`  
*Holds the matvec result between A and p.*
- `Matrix< double > z`  
*Full search direction for x.*

#### 4.5.1 Detailed Description

Data structure for the implementation of the CGS algorithm for non-symmetric linear systems.

C-style object to be used in conjunction with the Conjugate Gradient Squared (CGS) algorithm to solve linear systems of equations. This algorithm is slightly less computational work than BiCGSTAB, but is much less stable. As a result, I do not recommend using this algorithm unless you also use some form of preconditioning.

#### 4.5.2 Member Data Documentation

##### 4.5.2.1 `int CGS_DATA::maxit = 0`

Maximum allowable iterations - default =  $\min(2 * \text{vector\_size}, 1000)$

##### 4.5.2.2 `int CGS_DATA::iter = 0`

Actual number of iterations.

##### 4.5.2.3 `bool CGS_DATA::breakdown`

Boolean to determine if the method broke down.

##### 4.5.2.4 `double CGS_DATA::alpha`

Step size parameter for next solution.

##### 4.5.2.5 `double CGS_DATA::beta`

Step size parameter for search direction.

##### 4.5.2.6 `double CGS_DATA::rho`

Scaling parameter for alpha and beta.

**4.5.2.7 double CGS\_DATA::sigma**

Scaling parameter and additional step length.

**4.5.2.8 double CGS\_DATA::tol\_rel = 1e-6**

Relative tolerance for convergence - default = 1e-6.

**4.5.2.9 double CGS\_DATA::tol\_abs = 1e-6**

Absolution tolerance for convergence - default = 1e-6.

**4.5.2.10 double CGS\_DATA::res**

Absolute residual norm.

**4.5.2.11 double CGS\_DATA::relres**

Relative residual norm.

**4.5.2.12 double CGS\_DATA::relres\_base**

Initial residual norm.

**4.5.2.13 double CGS\_DATA::bestres**

Best found residual norm.

**4.5.2.14 bool CGS\_DATA::Output = true**

True = print messages to console.

**4.5.2.15 Matrix<double> CGS\_DATA::x**

Current solution to the linear system.

**4.5.2.16 Matrix<double> CGS\_DATA::bestx**

Best found solution to the linear system.

**4.5.2.17 Matrix<double> CGS\_DATA::r**

Residual vector for the linear system.

**4.5.2.18 Matrix<double> CGS\_DATA::r0**

Initial residual vector.

**4.5.2.19 Matrix<double> CGS\_DATA::u**

Search direction for v.

**4.5.2.20 Matrix<double> CGS\_DATA::w**

Updates sigma and u.

**4.5.2.21 Matrix<double> CGS\_DATA::v**

Search direction for x.

**4.5.2.22 Matrix<double> CGS\_DATA::p**

Preconditioning result for w, z, and matvec for Ax.

**4.5.2.23 Matrix<double> CGS\_DATA::c**

Holds the matvec result between A and p.

**4.5.2.24 Matrix<double> CGS\_DATA::z**

Full search direction for x.

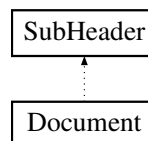
The documentation for this struct was generated from the following file:

- [lark.h](#)

**4.6 Document Class Reference**

```
#include <yaml_wrapper.h>
```

Inheritance diagram for Document:

**Public Member Functions**

- [Document](#) ()
- [~Document](#) ()
- [Document](#) (const [Document](#) &doc)
- [Document](#) (std::string name)
- [Document](#) (const [KeyValueMap](#) &map)
- [Document](#) (std::string name, const [KeyValueMap](#) &map)
- [Document](#) (std::string key, const [Header](#) &head)
- [Document](#) & [operator=](#) (const [Document](#) &doc)
- [ValueTypePair](#) & [operator\[\]](#) (const std::string key)

- [ValueTypePair operator\[\]](#) (const std::string key) const
- [Header & operator\(\)](#) (const std::string key)
- [Header operator\(\)](#) (const std::string key) const
- std::map< std::string, [Header](#) > & [getHeadMap](#) ()
- [KeyValueMap & getDataMap](#) ()
- [Header & getHeader](#) (std::string key)
- std::map< std::string, [Header](#) > ::const\_iterator [end](#) () const
- std::map< std::string, [Header](#) > ::iterator [end](#) ()
- std::map< std::string, [Header](#) > ::const\_iterator [begin](#) () const
- std::map< std::string, [Header](#) > ::iterator [begin](#) ()
- void [clear](#) ()
- void [resetKeys](#) ()
- void [changeKey](#) (std::string oldKey, std::string newKey)
- void [revalidateAllKeys](#) ()
- void [addPair](#) (std::string key, std::string val)
- void [addPair](#) (std::string key, std::string val, int t)
- void [setName](#) (std::string name)
- void [setAlias](#) (std::string alias)
- void [setNameAliasPair](#) (std::string n, std::string a, int s)
- void [setState](#) (int state)
- void [DisplayContents](#) ()
- void [addHeadKey](#) (std::string key)
- void [copyAnchor2Alias](#) (std::string alias, [Header](#) &ref)
- int [size](#) ()
- std::string [getName](#) ()
- std::string [getAlias](#) ()
- int [getState](#) ()
- bool [isAlias](#) ()
- bool [isAnchor](#) ()
- [Header & getAnchoredHeader](#) (std::string alias)
- [Header & getHeadFromSubAlias](#) (std::string alias)

### Private Attributes

- std::map< std::string, [Header](#) > [Head\\_Map](#)

### Additional Inherited Members

#### 4.6.1 Constructor & Destructor Documentation

4.6.1.1 [Document::Document \( \)](#)

4.6.1.2 [Document::~~Document \( \)](#)

4.6.1.3 [Document::Document \( const \*\*Document\*\* & doc \)](#)

4.6.1.4 [Document::Document \( std::string name \)](#)

4.6.1.5 [Document::Document \( const \*\*KeyValueMap\*\* & map \)](#)

4.6.1.6 Document::Document ( std::string *name*, const KeyValueType & *map* )

4.6.1.7 Document::Document ( std::string *key*, const Header & *head* )

## 4.6.2 Member Function Documentation

4.6.2.1 Document& Document::operator= ( const Document & *doc* )

4.6.2.2 ValuePair& Document::operator[] ( const std::string *key* )

4.6.2.3 ValuePair Document::operator[] ( const std::string *key* ) const

4.6.2.4 Header& Document::operator() ( const std::string *key* )

4.6.2.5 Header Document::operator() ( const std::string *key* ) const

4.6.2.6 std::map<std::string, Header>& Document::getHeadMap ( )

4.6.2.7 KeyValueType& Document::getDataMap ( )

4.6.2.8 Header& Document::getHeader ( std::string *key* )

4.6.2.9 std::map<std::string, Header>::const\_iterator Document::end ( ) const

4.6.2.10 std::map<std::string, Header>::iterator Document::end ( )

4.6.2.11 std::map<std::string, Header>::const\_iterator Document::begin ( ) const

4.6.2.12 std::map<std::string, Header>::iterator Document::begin ( )

4.6.2.13 void Document::clear ( )

4.6.2.14 void Document::resetKeys ( )

4.6.2.15 void Document::changeKey ( std::string *oldKey*, std::string *newKey* )

4.6.2.16 void Document::revalidateAllKeys ( )

4.6.2.17 void Document::addPair ( std::string *key*, std::string *val* )

4.6.2.18 void Document::addPair ( std::string *key*, std::string *val*, int *t* )

4.6.2.19 void Document::setName ( std::string *name* )

4.6.2.20 void Document::setAlias ( std::string *alias* )

4.6.2.21 void Document::setNameAliasPair ( std::string *n*, std::string *a*, int *s* )

4.6.2.22 void Document::setState ( int *state* )

4.6.2.23 void Document::DisplayContents ( )

4.6.2.24 void Document::addHeadKey ( std::string *key* )

4.6.2.25 void Document::copyAnchor2Alias ( std::string *alias*, Header & *ref* )



- 4.6.2.26 `int Document::size ( )`
- 4.6.2.27 `std::string Document::getName ( )`
- 4.6.2.28 `std::string Document::getAlias ( )`
- 4.6.2.29 `int Document::getState ( )`
- 4.6.2.30 `bool Document::isAlias ( )`
- 4.6.2.31 `bool Document::isAnchor ( )`
- 4.6.2.32 `Header& Document::getAnchoredHeader ( std::string alias )`
- 4.6.2.33 `Header& Document::getHeadFromSubAlias ( std::string alias )`

### 4.6.3 Member Data Documentation

- 4.6.3.1 `std::map<std::string, Header> Document::Head_Map [private]`

The documentation for this class was generated from the following file:

- [yaml\\_wrapper.h](#)

## 4.7 DOGFISH\_DATA Struct Reference

Primary data structure for running the DOGFISH application.

```
#include <dogfish.h>
```

### Public Attributes

- unsigned long int `total_steps` = 0  
*Total number of solver steps taken.*
- double `time_old` = 0.0  
*Old value of time (hrs)*
- double `time` = 0.0  
*Current value of time (hrs)*
- bool `Print2File` = true  
*True = results to .txt; False = no printing.*
- bool `Print2Console` = true  
*True = results to console; False = no printing.*
- bool `DirichletBC` = false  
*False = uses film mass transfer for BC, True = Dirichlet BC.*
- bool `NonLinear` = false  
*False = Solve directly, True = Solve iteratively.*
- double `t_counter` = 0.0  
*Counter for the time output.*
- double `t_print`  
*Print output at every t\_print time (hrs)*
- int `NumComp`  
*Number of species to track.*

- double `end_time`  
*Units: hours.*
- double `total_sorption_old`  
*Per mass or volume of single fiber.*
- double `total_sorption`  
*Per mass or volume of single fiber.*
- double `fiber_length`  
*Units: um.*
- double `fiber_diameter`  
*Units: um.*
- FILE \* `OutputFile`  
*Output file pointer to the output file for postprocesses and results.*
- double(\* `eval_R`)(int i, int l, const void \*data)  
*Function pointer to evaluate retardation coefficient.*
- double(\* `eval_DI`)(int i, int l, const void \*data)  
*Function pointer to evaluate intraparticle diffusivity.*
- double(\* `eval_kf`)(int i, const void \*data)  
*Function pointer to evaluate film mass transfer coefficient.*
- double(\* `eval_qs`)(int i, const void \*data)  
*Function pointer to evaluate fiber surface concentration.*
- const void \* `user_data`  
*Data structure for users info to calculate all parameters.*
- std::vector< `FINCH_DATA` > `finch_dat`  
*Data structure for `FINCH_DATA` objects.*
- std::vector< `DOGFISH_PARAM` > `param_dat`  
*Data structure for `DOGFISH_PARAM` objects.*

#### 4.7.1 Detailed Description

Primary data structure for running the DOGFISH application.

C-style object to hold information for the adsorption simulations. Contains function pointers and other data structures. This information is passed around to other functions used to simulate the fiber diffusion physics.

#### 4.7.2 Member Data Documentation

##### 4.7.2.1 unsigned long int DOGFISH\_DATA::total\_steps = 0

Total number of solver steps taken.

##### 4.7.2.2 double DOGFISH\_DATA::time\_old = 0.0

Old value of time (hrs)

##### 4.7.2.3 double DOGFISH\_DATA::time = 0.0

Current value of time (hrs)

##### 4.7.2.4 bool DOGFISH\_DATA::Print2File = true

True = results to .txt; False = no printing.

4.7.2.5 `bool DOGFISH_DATA::Print2Console = true`

True = results to console; False = no printing.

4.7.2.6 `bool DOGFISH_DATA::DirichletBC = false`

False = uses film mass transfer for BC, True = Dirichlet BC.

4.7.2.7 `bool DOGFISH_DATA::NonLinear = false`

False = Solve directly, True = Solve iteratively.

4.7.2.8 `double DOGFISH_DATA::t_counter = 0.0`

Counter for the time output.

4.7.2.9 `double DOGFISH_DATA::t_print`

Print output at every `t_print` time (hrs)

4.7.2.10 `int DOGFISH_DATA::NumComp`

Number of species to track.

4.7.2.11 `double DOGFISH_DATA::end_time`

Units: hours.

4.7.2.12 `double DOGFISH_DATA::total_sorption_old`

Per mass or volume of single fiber.

4.7.2.13 `double DOGFISH_DATA::total_sorption`

Per mass or volume of single fiber.

4.7.2.14 `double DOGFISH_DATA::fiber_length`

Units:  $\mu\text{m}$ .

4.7.2.15 `double DOGFISH_DATA::fiber_diameter`

Units:  $\mu\text{m}$ .

4.7.2.16 `FILE* DOGFISH_DATA::OutputFile`

Output file pointer to the output file for postprocesses and results.

4.7.2.17 `double(* DOGFISH_DATA::eval_R)(int i, int l, const void *data)`

Function pointer to evaluate retardation coefficient.

4.7.2.18 `double(* DOGFISH_DATA::eval_DI)(int i, int l, const void *data)`

Function pointer to evaluate intraparticle diffusivity.

4.7.2.19 `double(* DOGFISH_DATA::eval_kf)(int i, const void *data)`

Function pointer to evaluate film mass transfer coefficient.

4.7.2.20 `double(* DOGFISH_DATA::eval_qs)(int i, const void *data)`

Function pointer to evaluate fiber surface concentration.

4.7.2.21 `const void* DOGFISH_DATA::user_data`

Data structure for users info to calculate all parameters.

4.7.2.22 `std::vector<FINCH_DATA> DOGFISH_DATA::finch_dat`

Data structure for [FINCH\\_DATA](#) objects.

4.7.2.23 `std::vector<DOGFISH_PARAM> DOGFISH_DATA::param_dat`

Data structure for [DOGFISH\\_PARAM](#) objects.

The documentation for this struct was generated from the following file:

- [dogfish.h](#)

## 4.8 DOGFISH\_PARAM Struct Reference

Data structure for species-specific parameters.

```
#include <dogfish.h>
```

### Public Attributes

- double [intraparticle\\_diffusion](#)  
*Units:  $\mu\text{m}^2/\text{hr}$ .*
- double [film\\_transfer\\_coeff](#)  
*Units:  $\mu\text{m}/\text{hr}$ .*
- double [surface\\_concentration](#)  
*Units:  $\text{mg}/\text{g}$ .*
- double [initial\\_sorption](#)  
*Units:  $\text{mg}/\text{g}$ .*
- double [sorbed\\_molefraction](#)  
*Molefraction of sorbed species.*
- [Molecule species](#)  
*Adsorbed species [Molecule](#) Object.*

### 4.8.1 Detailed Description

Data structure for species-specific parameters.

C-style object to hold information on all adsorbing species. Parameters are given descriptive names to indicate what each is for.

### 4.8.2 Member Data Documentation

4.8.2.1 double DOGFISH\_PARAM::intraparticle\_diffusion

Units:  $\text{um}^2/\text{hr}$ .

4.8.2.2 double DOGFISH\_PARAM::film\_transfer\_coeff

Units:  $\text{um}/\text{hr}$ .

4.8.2.3 double DOGFISH\_PARAM::surface\_concentration

Units:  $\text{mg}/\text{g}$ .

4.8.2.4 double DOGFISH\_PARAM::initial\_sorption

Units:  $\text{mg}/\text{g}$ .

4.8.2.5 double DOGFISH\_PARAM::sorbed\_molefraction

Molefraction of sorbed species.

4.8.2.6 Molecule DOGFISH\_PARAM::species

Adsorbed species [Molecule](#) Object.

The documentation for this struct was generated from the following file:

- [dogfish.h](#)

## 4.9 FINCH\_DATA Struct Reference

Data structure for the FINCH object.

```
#include <finch.h>
```

### Public Attributes

- int [d](#) = 0  
*Dimension of the problem: 0 = cartesian, 1 = cylindrical, 2 = spherical.*
- double [dt](#) = 0.0125  
*Time step.*
- double [dt\\_old](#) = 0.0125  
*Previous time step.*

- double **T** = 1.0  
*Total time.*
- double **dz** = 0.1  
*Space step.*
- double **L** = 1.0  
*Total space.*
- double **s** = 1.0  
*Char quantity (spherical = 1, cylindrical = length, cartesian = area)*
- double **t** = 0.0  
*Current Time.*
- double **t\_old** = 0.0  
*Previous Time.*
- double **uT** = 0.0  
*Total amount of conserved quantity in domain.*
- double **uT\_old** = 0.0  
*Old Total amount of conserved quantity.*
- double **uAvg** = 0.0  
*Average amount of conserved quantity in domain.*
- double **uAvg\_old** = 0.0  
*Old Average amount of conserved quantity.*
- double **uIC** = 0.0  
*Initial condition of Conserved Quantity (if constant)*
- double **vIC** = 1.0  
*Initial condition of Velocity (if constant)*
- double **DIC** = 1.0  
*Initial condition of Dispersion (if constant)*
- double **kIC** = 1.0  
*Initial condition of **Reaction** (if constant)*
- double **RIC** = 1.0  
*Initial condition of the Time Coefficient (if constant)*
- double **uo** = 1.0  
*Boundary Value of Conserved Quantity.*
- double **vo** = 1.0  
*Boundary Value of Velocity.*
- double **Do** = 1.0  
*Boundary Value of Dispersion.*
- double **ko** = 1.0  
*Boundary Value of **Reaction**.*
- double **Ro** = 1.0  
*Boundary Value of Time Coefficient.*
- double **kfn** = 1.0  
*Film mass transfer coefficient Old.*
- double **kfnp1** = 1.0  
*Film mass transfer coefficient New.*
- double **lambda\_I**  
*Boundary Coefficient for Implicit Neumann (Calculated at Runtime)*
- double **lambda\_E**  
*Boundary Coefficient for Explicit Neumann (Calculated at Runtime)*
- int **LN** = 10  
*Number of nodes.*
- bool **CN** = true

*True if Crank-Nicholson, false if Implicit, never use explicit.*

- bool `Update` = false

*Flag to check if the system needs updating.*

- bool `Dirichlet` = false

*Flag to indicate use of Dirichlet or Neumann starting boundary.*

- bool `CheckMass` = false

*Flag to indicate whether or not mass is to be checked.*

- bool `ExplicitFlux` = false

*Flag to indicate whether or not to use fully explicit flux limiters.*

- bool `Iterative` = true

*Flag to indicate whether to solve directly, or iteratively.*

- bool `SteadyState` = false

*Flag to determine whether or not to solve the steady-state problem.*

- bool `NormTrack` = true

*Flag to determine whether or not to track the norms during simulation.*

- double `beta` = 0.5

*Scheme type indicator: 0.5=CN & 1.0=Implicit; all else NULL.*

- double `tol_rel` = 1e-6

*Relative Tolerance for Convergence.*

- double `tol_abs` = 1e-6

*Absolute Tolerance for Convergence.*

- int `max_iter` = 20

*Maximum number of iterations allowed.*

- int `total_iter` = 0

*Total number of iterations made.*

- int `nl_method` = `FINCH_Picard`

*Non-linear solution method - default = FINCH\_Picard.*

- std::vector< double > `CL_I`

*Left side, implicit coefficients (Calculated at Runtime)*

- std::vector< double > `CL_E`

*Left side, explicit coefficients (Calculated at Runtime)*

- std::vector< double > `CC_I`

*Centered, implicit coefficients (Calculated at Runtime)*

- std::vector< double > `CC_E`

*Centered, explicit coefficients (Calculated at Runtime)*

- std::vector< double > `CR_I`

*Right side, implicit coefficients (Calculated at Runtime)*

- std::vector< double > `CR_E`

*Right side, explicit coefficients (Calculated at Runtime)*

- std::vector< double > `fL_I`

*Left side, implicit fluxes (Calculated at Runtime)*

- std::vector< double > `fL_E`

*Left side, explicit fluxes (Calculated at Runtime)*

- std::vector< double > `fC_I`

*Centered, implicit fluxes (Calculated at Runtime)*

- std::vector< double > `fC_E`

*Centered, explicit fluxes (Calculated at Runtime)*

- std::vector< double > `fR_I`

*Right side, implicit fluxes (Calculated at Runtime)*

- std::vector< double > `fR_E`

*Right side, explicit fluxes (Calculated at Runtime)*

- `std::vector< double > OI`  
*Implicit upper diagonal matrix elements (Calculated at Runtime)*
- `std::vector< double > OE`  
*Explicit upper diagonal matrix elements (Calculated at Runtime)*
- `std::vector< double > NI`  
*Implicit diagonal matrix elements (Calculated at Runtime)*
- `std::vector< double > NE`  
*Explicit diagonal matrix elements (Calculated at Runtime)*
- `std::vector< double > MI`  
*Implicit lower diagonal matrix elements (Calculated at Runtime)*
- `std::vector< double > ME`  
*Explicit lower diagonal matrix elements (Calculated at Runtime)*
- `std::vector< double > uz_l_I`
- `std::vector< double > uz_lm1_I`
- `std::vector< double > uz_lp1_I`  
*Implicit local slopes (Calculated at Runtime)*
- `std::vector< double > uz_l_E`
- `std::vector< double > uz_lm1_E`
- `std::vector< double > uz_lp1_E`  
*Explicit local slopes (Calculated at Runtime)*
- `Matrix< double > unm1`  
*Conserved Quantity Older.*
- `Matrix< double > un`  
*Conserved Quantity Old.*
- `Matrix< double > unp1`  
*Conserved Quantity New.*
- `Matrix< double > u_star`  
*Conserved Quantity Projected New.*
- `Matrix< double > ubest`  
*Best found solution if solving iteratively.*
- `Matrix< double > vn`  
*Velocity Old.*
- `Matrix< double > vnp1`  
*Velocity New.*
- `Matrix< double > Dn`  
*Dispersion Old.*
- `Matrix< double > Dnp1`  
*Dispersion New.*
- `Matrix< double > kn`  
*Reaction Old.*
- `Matrix< double > knp1`  
*Reaction New.*
- `Matrix< double > Sn`  
*Forcing Function Old.*
- `Matrix< double > Snp1`  
*Forcing Function New.*
- `Matrix< double > Rn`  
*Time Coeff Old.*
- `Matrix< double > Rnp1`  
*Time Coeff New.*
- `Matrix< double > Fn`



- Flux Limiter Old.*
- [Matrix](#)< double > [Fnp1](#)
- Flux Limiter New.*
- [Matrix](#)< double > [gl](#)
- Implicit Side Boundary Conditions.*
- [Matrix](#)< double > [gE](#)
- Explicit Side Boundary Conditions.*
- [Matrix](#)< double > [res](#)
- Current residual.*
- [Matrix](#)< double > [pres](#)
- Current search direction.*
- [int](#)(\* [callroutine](#) )(const void \*user\_data)
- Function pointer to executioner (DEFAULT = default\_execution)*
- [int](#)(\* [setic](#) )(const void \*user\_data)
- Function pointer to initial conditions (DEFAULT = default\_ic)*
- [int](#)(\* [settime](#) )(const void \*user\_data)
- Function pointer to set time step (DEFAULT = default\_timestep)*
- [int](#)(\* [setpreprocess](#) )(const void \*user\_data)
- Function pointer to preprocesses (DEFAULT = default\_preprocess)*
- [int](#)(\* [solve](#) )(const void \*user\_data)
- Function pointer to the solver (DEFAULT = default\_solve)*
- [int](#)(\* [setparams](#) )(const void \*user\_data)
- Function pointer to set parameters (DEFAULT = default\_params)*
- [int](#)(\* [discretize](#) )(const void \*user\_data)
- Function pointer to discretization (DEFAULT = ospre\_discretization)*
- [int](#)(\* [setbcs](#) )(const void \*user\_data)
- [int](#)(\* [evalres](#) )(const [Matrix](#)< double > &x, [Matrix](#)< double > &res, const void \*user\_data)
- Function pointer to the residual function (DEFAULT = default\_res)*
- [int](#)(\* [evalprecon](#) )(const [Matrix](#)< double > &b, [Matrix](#)< double > &p, const void \*user\_data)
- Function pointer to the preconditioning function (DEFAULT = default\_precon)*
- [int](#)(\* [setpostprocess](#) )(const void \*user\_data)
- Function pointer to the postprocesses (DEFAULT = default\_postprocess)*
- [int](#)(\* [resettime](#) )(const void \*user\_data)
- Function pointer to reset time (DEFAULT = default\_reset)*
- [PICARD\\_DATA](#) [picard\\_dat](#)
- Data structure for PICARD method (no need to use this)*
- [PJFNK\\_DATA](#) [pjfnk\\_dat](#)
- Data structure for PJFNK method (more rigours method)*
- const void \* [param\\_data](#)
- User's data structure used to evaluate the parameter function (Must override if setparams is overridden)*

#### 4.9.1 Detailed Description

Data structure for the FINCH object.

C-style object that holds data, functions, and other structures necessary to discretize and solve a FINCH problem. All of this information must be overridden or initialized prior to running a FINCH simulation. Many, many default functions are provided to make it easier to incorporate FINCH into other problems. The main function to override will be the setparams function. This will be a function that the user provides to tell the FINCH simulation how the parameters of the problem vary in time and space and whether or not they are coupled the the variable u. All functions are overridable and several can be skipped entirely, or called directly at different times in the execution of a particular routine. This make FINCH extremely flexible to the user.

**Note**

All parameters and dimensions do not carry any units with them. The user is required to keep track of all their own units in their particular problem and ensure that units will cancel and be consistent in their own physical model.

**4.9.2 Member Data Documentation****4.9.2.1 int FINCH\_DATA::d = 0**

Dimension of the problem: 0 = cartesian, 1 = cylindrical, 2 = spherical.

**4.9.2.2 double FINCH\_DATA::dt = 0.0125**

Time step.

**4.9.2.3 double FINCH\_DATA::dt\_old = 0.0125**

Previous time step.

**4.9.2.4 double FINCH\_DATA::T = 1.0**

Total time.

**4.9.2.5 double FINCH\_DATA::dz = 0.1**

Space step.

**4.9.2.6 double FINCH\_DATA::L = 1.0**

Total space.

**4.9.2.7 double FINCH\_DATA::s = 1.0**

Char quantity (spherical = 1, cylindrical = length, cartesian = area)

**4.9.2.8 double FINCH\_DATA::t = 0.0**

Current Time.

**4.9.2.9 double FINCH\_DATA::t\_old = 0.0**

Previous Time.

**4.9.2.10 double FINCH\_DATA::uT = 0.0**

Total amount of conserved quantity in domain.

**4.9.2.11 double FINCH\_DATA::uT\_old = 0.0**

Old Total amount of conserved quantity.

4.9.2.12 double FINCH\_DATA::uAvg = 0.0

Average amount of conserved quantity in domain.

4.9.2.13 double FINCH\_DATA::uAvg\_old = 0.0

Old Average amount of conserved quantity.

4.9.2.14 double FINCH\_DATA::uIC = 0.0

Initial condition of Conserved Quantity (if constant)

4.9.2.15 double FINCH\_DATA::vIC = 1.0

Initial condition of Velocity (if constant)

4.9.2.16 double FINCH\_DATA::DIC = 1.0

Initial condition of Dispersion (if constant)

4.9.2.17 double FINCH\_DATA::kIC = 1.0

Initial condition of [Reaction](#) (if constant)

4.9.2.18 double FINCH\_DATA::RIC = 1.0

Initial condition of the Time Coefficient (if constant)

4.9.2.19 double FINCH\_DATA::uo = 1.0

Boundary Value of Conserved Quantity.

4.9.2.20 double FINCH\_DATA::vo = 1.0

Boundary Value of Velocity.

4.9.2.21 double FINCH\_DATA::Do = 1.0

Boundary Value of Dispersion.

4.9.2.22 double FINCH\_DATA::ko = 1.0

Boundary Value of [Reaction](#).

4.9.2.23 double FINCH\_DATA::Ro = 1.0

Boundary Value of Time Coefficient.

4.9.2.24 `double FINCH_DATA::kfn = 1.0`

Film mass transfer coefficient Old.

4.9.2.25 `double FINCH_DATA::kfnp1 = 1.0`

Film mass transfer coefficient New.

4.9.2.26 `double FINCH_DATA::lambda_I`

Boundary Coefficient for Implicit Neumann (Calculated at Runtime)

4.9.2.27 `double FINCH_DATA::lambda_E`

Boundary Coefficient for Explicit Neumann (Calculated at Runtime)

4.9.2.28 `int FINCH_DATA::LN = 10`

Number of nodes.

4.9.2.29 `bool FINCH_DATA::CN = true`

True if Crank-Nicholson, false if Implicit, never use explicit.

4.9.2.30 `bool FINCH_DATA::Update = false`

Flag to check if the system needs updating.

4.9.2.31 `bool FINCH_DATA::Dirichlet = false`

Flag to indicate use of Dirichlet or Neumann starting boundary.

4.9.2.32 `bool FINCH_DATA::CheckMass = false`

Flag to indicate whether or not mass is to be checked.

4.9.2.33 `bool FINCH_DATA::ExplicitFlux = false`

Flag to indicate whether or not to use fully explicit flux limiters.

4.9.2.34 `bool FINCH_DATA::Iterative = true`

Flag to indicate whether to solve directly, or iteratively.

4.9.2.35 `bool FINCH_DATA::SteadyState = false`

Flag to determine whether or not to solve the steady-state problem.

4.9.2.36 `bool FINCH_DATA::NormTrack = true`

Flag to determine whether or not to track the norms during simulation.

4.9.2.37 `double FINCH_DATA::beta = 0.5`

Scheme type indicator: 0.5=CN & 1.0=Implicit; all else NULL.

4.9.2.38 `double FINCH_DATA::tol_rel = 1e-6`

Relative Tolerance for Convergence.

4.9.2.39 `double FINCH_DATA::tol_abs = 1e-6`

Absolute Tolerance for Convergence.

4.9.2.40 `int FINCH_DATA::max_iter = 20`

Maximum number of iterations allowed.

4.9.2.41 `int FINCH_DATA::total_iter = 0`

Total number of iterations made.

4.9.2.42 `int FINCH_DATA::nl_method = FINCH_Picard`

Non-linear solution method - default = FINCH\_Picard.

4.9.2.43 `std::vector<double> FINCH_DATA::CL_I`

Left side, implicit coefficients (Calculated at Runtime)

4.9.2.44 `std::vector<double> FINCH_DATA::CL_E`

Left side, explicit coefficients (Calculated at Runtime)

4.9.2.45 `std::vector<double> FINCH_DATA::CC_I`

Centered, implicit coefficients (Calculated at Runtime)

4.9.2.46 `std::vector<double> FINCH_DATA::CC_E`

Centered, explicit coefficients (Calculated at Runtime)

4.9.2.47 `std::vector<double> FINCH_DATA::CR_I`

Right side, implicit coefficients (Calculated at Runtime)

4.9.2.48 `std::vector<double> FINCH_DATA::CR_E`

Right side, explicit coefficients (Calculated at Runtime)

4.9.2.49 `std::vector<double> FINCH_DATA::fL_I`

Left side, implicit fluxes (Calculated at Runtime)

4.9.2.50 `std::vector<double> FINCH_DATA::fL_E`

Left side, explicit fluxes (Calculated at Runtime)

4.9.2.51 `std::vector<double> FINCH_DATA::fC_I`

Centered, implicit fluxes (Calculated at Runtime)

4.9.2.52 `std::vector<double> FINCH_DATA::fC_E`

Centered, explicit fluxes (Calculated at Runtime)

4.9.2.53 `std::vector<double> FINCH_DATA::fR_I`

Right side, implicit fluxes (Calculated at Runtime)

4.9.2.54 `std::vector<double> FINCH_DATA::fR_E`

Right side, explicit fluxes (Calculated at Runtime)

4.9.2.55 `std::vector<double> FINCH_DATA::OI`

Implicit upper diagonal matrix elements (Calculated at Runtime)

4.9.2.56 `std::vector<double> FINCH_DATA::OE`

Explicit upper diagonal matrix elements (Calculated at Runtime)

4.9.2.57 `std::vector<double> FINCH_DATA::NI`

Implicit diagonal matrix elements (Calculated at Runtime)

4.9.2.58 `std::vector<double> FINCH_DATA::NE`

Explicit diagonal matrix elements (Calculated at Runtime)

4.9.2.59 `std::vector<double> FINCH_DATA::MI`

Implicit lower diagonal matrix elements (Calculated at Runtime)

4.9.2.60 `std::vector<double> FINCH_DATA::ME`

Explicit lower diagonal matrix elements (Calculated at Runtime)

4.9.2.61 `std::vector<double> FINCH_DATA::uz_I_I`

4.9.2.62 `std::vector<double> FINCH_DATA::uz_lm1_I`

4.9.2.63 `std::vector<double> FINCH_DATA::uz_lp1_I`

Implicit local slopes (Calculated at Runtime)

4.9.2.64 `std::vector<double> FINCH_DATA::uz_I_E`

4.9.2.65 `std::vector<double> FINCH_DATA::uz_lm1_E`

4.9.2.66 `std::vector<double> FINCH_DATA::uz_lp1_E`

Explicit local slopes (Calculated at Runtime)

4.9.2.67 `Matrix<double> FINCH_DATA::unm1`

Conserved Quantity Older.

4.9.2.68 `Matrix<double> FINCH_DATA::un`

Conserved Quantity Old.

4.9.2.69 `Matrix<double> FINCH_DATA::unp1`

Conserved Quantity New.

4.9.2.70 `Matrix<double> FINCH_DATA::u_star`

Conserved Quantity Projected New.

4.9.2.71 `Matrix<double> FINCH_DATA::ubest`

Best found solution if solving iteratively.

4.9.2.72 `Matrix<double> FINCH_DATA::vn`

Velocity Old.

4.9.2.73 `Matrix<double> FINCH_DATA::vnp1`

Velocity New.

4.9.2.74 `Matrix<double> FINCH_DATA::Dn`

Dispersion Old.

**4.9.2.75 Matrix<double> FINCH\_DATA::Dnp1**

Dispersion New.

**4.9.2.76 Matrix<double> FINCH\_DATA::kn**

[Reaction](#) Old.

**4.9.2.77 Matrix<double> FINCH\_DATA::knp1**

[Reaction](#) New.

**4.9.2.78 Matrix<double> FINCH\_DATA::Sn**

Forcing Function Old.

**4.9.2.79 Matrix<double> FINCH\_DATA::Snp1**

Forcing Function New.

**4.9.2.80 Matrix<double> FINCH\_DATA::Rn**

Time Coeff Old.

**4.9.2.81 Matrix<double> FINCH\_DATA::Rnp1**

Time Coeff New.

**4.9.2.82 Matrix<double> FINCH\_DATA::Fn**

Flux Limiter Old.

**4.9.2.83 Matrix<double> FINCH\_DATA::Fnp1**

Flux Limiter New.

**4.9.2.84 Matrix<double> FINCH\_DATA::gl**

Implicit Side Boundary Conditions.

**4.9.2.85 Matrix<double> FINCH\_DATA::gE**

Explicit Side Boundary Conditions.

**4.9.2.86 Matrix<double> FINCH\_DATA::res**

Current residual.



**4.9.2.87** `Matrix<double> FINCH_DATA::pres`

Current search direction.

**4.9.2.88** `int(* FINCH_DATA::callroutine)(const void *user_data)`

Function pointer to executioner (DEFAULT = default\_execution)

**4.9.2.89** `int(* FINCH_DATA::setic)(const void *user_data)`

Function pointer to initial conditions (DEFAULT = default\_ic)

**4.9.2.90** `int(* FINCH_DATA::settime)(const void *user_data)`

Function pointer to set time step (DEFAULT = default\_timestep)

**4.9.2.91** `int(* FINCH_DATA::setpreprocess)(const void *user_data)`

Function pointer to preprocesses (DEFAULT = default\_preprocess)

**4.9.2.92** `int(* FINCH_DATA::solve)(const void *user_data)`

Function pointer to the solver (DEFAULT = default\_solve)

**4.9.2.93** `int(* FINCH_DATA::setparams)(const void *user_data)`

Function pointer to set parameters (DEFAULT = default\_params)

**4.9.2.94** `int(* FINCH_DATA::discretize)(const void *user_data)`

Function pointer to discretization (DEFAULT = ospre\_discretization)

**4.9.2.95** `int(* FINCH_DATA::setbcs)(const void *user_data)`

Function pointer to set boundary conditions (DEFAULT = default\_bcs)

**4.9.2.96** `int(* FINCH_DATA::evalres)(const Matrix< double > &x, Matrix< double > &res, const void *user_data)`

Function pointer to the residual function (DEFAULT = default\_res)

**4.9.2.97** `int(* FINCH_DATA::evalprecon)(const Matrix< double > &b, Matrix< double > &p, const void *user_data)`

Function pointer to the preconditioning function (DEFAULT = default\_precon)

**4.9.2.98** `int(* FINCH_DATA::setpostprocess)(const void *user_data)`

Function pointer to the postprocesses (DEFAULT = default\_postprocess)

4.9.2.99 `int(* FINCH_DATA::resetime)(const void *user_data)`

Function pointer to reset time (DEFAULT = default\_reset)

4.9.2.100 `PICARD_DATA FINCH_DATA::picard_dat`

Data structure for PICARD method (no need to use this)

4.9.2.101 `PJFNK_DATA FINCH_DATA::pjfnk_dat`

Data structure for PJFNK method (more rigours method)

4.9.2.102 `const void* FINCH_DATA::param_data`

User's data structure used to evaluate the parameter function (Must override if setparams is overridden)

The documentation for this struct was generated from the following file:

- [finch.h](#)

## 4.10 GCR\_DATA Struct Reference

Data structure for the implementation of the GCR algorithm for non-symmetric linear systems.

```
#include <lark.h>
```

### Public Attributes

- `int restart = -1`  
*Restart parameter for outer iterations - default = 50.*
- `int maxit = 0`  
*Maximum allowable outer iterations.*
- `int iter_outer = 0`  
*Number of outer iterations taken.*
- `int iter_inner = 0`  
*Number of inner iterations taken.*
- `int total_iter = 0`  
*Total number of iterations taken.*
- `bool breakdown = false`  
*Boolean to determine if a step has failed.*
- `double alpha`  
*Inner iteration step size.*
- `double beta`  
*Outer iteration step size.*
- `double tol_rel = 1e-6`  
*Relative tolerance for convergence - default = 1e-6.*
- `double tol_abs = 1e-6`  
*Absolute tolerance for convergence - default = 1e-6.*
- `double res`  
*Absolute residual norm for linear system.*
- `double relres`

- double `relres_base`  
*Relative residual norm for linear system.*
- double `bestres`  
*Initial residual norm of the linear system.*
- bool `Output` = true  
*Best found residual norm of the linear system.*
- `Matrix< double > x`  
*True = print messages to the console.*
- `Matrix< double > bestx`  
*Current solution to the linear system.*
- `Matrix< double > r`  
*Best found solution to the linear system.*
- `Matrix< double > c_temp`  
*Residual Vector.*
- `Matrix< double > u_temp`  
*Temporary c vector to be updated.*
- `std::vector< Matrix< double > > u`  
*Temporary u vector to be updated.*
- `std::vector< Matrix< double > > c`  
*Vector span for updating x.*
- `OPTRANS_DATA transpose_dat`  
*Vector span for updating r.*
- `OPTRANS_DATA transpose_dat`  
*Data structure for Operator Transposition.*

#### 4.10.1 Detailed Description

Data structure for the implementation of the GCR algorithm for non-symmetric linear systems.

C-style object used in conjunction with the Generalized Conjugate Residual (GCR) algorithm for solving a non-symmetric linear system of equations. When the linear system in question has a positive-definite-symmetric component to it, then this algorithm is equivalent to GMRESRP. However, it is generally less efficient than GMRESRP and can suffer breakdowns.

#### 4.10.2 Member Data Documentation

4.10.2.1 `int GCR_DATA::restart = -1`

Restart parameter for outer iterations - default = 50.

4.10.2.2 `int GCR_DATA::maxit = 0`

Maximum allowable outer iterations.

4.10.2.3 `int GCR_DATA::iter_outer = 0`

Number of outer iterations taken.

4.10.2.4 `int GCR_DATA::iter_inner = 0`

Number of inner iterations taken.

4.10.2.5 `int GCR_DATA::total_iter = 0`

Total number of iterations taken.

4.10.2.6 `bool GCR_DATA::breakdown = false`

Boolean to determine if a step has failed.

4.10.2.7 `double GCR_DATA::alpha`

Inner iteration step size.

4.10.2.8 `double GCR_DATA::beta`

Outer iteration step size.

4.10.2.9 `double GCR_DATA::tol_rel = 1e-6`

Relative tolerance for convergence - default = 1e-6.

4.10.2.10 `double GCR_DATA::tol_abs = 1e-6`

Absolute tolerance for convergence - default = 1e-6.

4.10.2.11 `double GCR_DATA::res`

Absolute residual norm for linear system.

4.10.2.12 `double GCR_DATA::relres`

Relative residual norm for linear system.

4.10.2.13 `double GCR_DATA::relres_base`

Initial residual norm of the linear system.

4.10.2.14 `double GCR_DATA::bestres`

Best found residual norm of the linear system.

4.10.2.15 `bool GCR_DATA::Output = true`

True = print messages to the console.

4.10.2.16 `Matrix<double> GCR_DATA::x`

Current solution to the linear system.

4.10.2.17 `Matrix<double> GCR_DATA::bestx`

Best found solution to the linear system.

4.10.2.18 `Matrix<double> GCR_DATA::r`

Residual Vector.

4.10.2.19 `Matrix<double> GCR_DATA::c_temp`

Temporary c vector to be updated.

4.10.2.20 `Matrix<double> GCR_DATA::u_temp`

Temporary u vector to be updated.

4.10.2.21 `std::vector<Matrix<double> > GCR_DATA::u`

Vector span for updating x.

4.10.2.22 `std::vector<Matrix<double> > GCR_DATA::c`

Vector span for updating r.

4.10.2.23 `OPTRANS_DATA GCR_DATA::transpose_dat`

Data structure for Operator Transposition.

The documentation for this struct was generated from the following file:

- [lark.h](#)

## 4.11 GMRESLP\_DATA Struct Reference

Data structure for implementation of the Restarted GMRES algorithm with Left Preconditioning.

```
#include <lark.h>
```

### Public Attributes

- `int restart = -1`  
*Restart parameter - default = min(vector\_size,50)*
- `int maxit = 0`  
*Maximum allowable iterations - default = min(vector\_size,1000)*
- `int iter = 0`  
*Number of iterations needed for convergence.*
- `int steps = 0`  
*Total number of gmres iterations and krylov iterations.*
- `double tol_rel = 1e-6`  
*Relative tolerance for convergence - default = 1e-6.*

- double `tol_abs` = 1e-6  
*Absolute tolerance for convergence - default = 1e-6.*
- double `res`  
*Absolute residual norm of the linear system.*
- double `relres`  
*Relative residual norm of the linear system.*
- double `relres_base`  
*Initial residual norm of the linear system.*
- double `bestres`  
*Best found residual norm of the linear system.*
- bool `Output` = true  
*True = print messages to console.*
- `Matrix`< double > `x`  
*Current solution to the linear system.*
- `Matrix`< double > `bestx`  
*Best found solution to the linear system.*
- `Matrix`< double > `r`  
*Residual vector for the linear system.*
- `ARNOLDI_DATA` `arnoldi_dat`  
*Data structure for the krylov subspace.*

#### 4.11.1 Detailed Description

Data structure for implementation of the Restarted GMRES algorithm with Left Preconditioning.

C-style object used in conjunction with Generalized Minimum RESidual Left-Preconditioned (GMRESLP) and Full Orthogonalization Method (FOM) algorithms to iteratively or directly solve a linear system of equations. When using with GMRESLP, you can only check/observe the linear residuals before a restart or after the Arnoldi space is constructed. This is because this object uses Left-side Preconditioning. A faster routine may be GMRESRP, which is able to construct residuals after each Arnoldi iteration.

#### 4.11.2 Member Data Documentation

##### 4.11.2.1 int GMRESLP\_DATA::restart = -1

Restart parameter - default = min(vector\_size,50)

##### 4.11.2.2 int GMRESLP\_DATA::maxit = 0

Maximum allowable iterations - default = min(vector\_size,1000)

##### 4.11.2.3 int GMRESLP\_DATA::iter = 0

Number of iterations needed for convergence.

##### 4.11.2.4 int GMRESLP\_DATA::steps = 0

Total number of gmres iterations and krylov iterations.

##### 4.11.2.5 double GMRESLP\_DATA::tol\_rel = 1e-6

Relative tolerance for convergence - default = 1e-6.

4.11.2.6 `double GMRESLP_DATA::tol_abs = 1e-6`

Absolution tolerance for convergence - default = 1e-6.

4.11.2.7 `double GMRESLP_DATA::res`

Absolution redisual norm of the linear system.

4.11.2.8 `double GMRESLP_DATA::relres`

Relative residual norm of the linear system.

4.11.2.9 `double GMRESLP_DATA::relres_base`

Initial residual norm of the linear system.

4.11.2.10 `double GMRESLP_DATA::bestres`

Best found residual norm of the linear system.

4.11.2.11 `bool GMRESLP_DATA::Output = true`

True = print messages to console.

4.11.2.12 `Matrix<double> GMRESLP_DATA::x`

Current solution to the linear system.

4.11.2.13 `Matrix<double> GMRESLP_DATA::bestx`

Best found solution to the linear system.

4.11.2.14 `Matrix<double> GMRESLP_DATA::r`

Residual vector for the linear system.

4.11.2.15 `ARNOLDI_DATA GMRESLP_DATA::arnoldi_dat`

Data structure for the kyrlov subspace.

The documentation for this struct was generated from the following file:

- [lark.h](#)

## 4.12 GMRESR\_DATA Struct Reference

Data structure for the implementation of GCR with Nested GMRES preconditioning (i.e., GMRESR)

```
#include <lark.h>
```

## Public Attributes

- int `gcr_restart` = -1  
*Number of GCR restarts (default = 50, max = N)*
- int `gcr_maxit` = 0  
*Number of GCR iterations.*
- int `gmres_restart` = -1  
*Number of GMRES restarts (max = 20)*
- int `gmres_maxit` = 1  
*Number of GMRES iterations (max = 5, default = 1)*
- int `N`  
*Dimension of the linear system.*
- int `total_iter`  
*Total GMRES and GCR iterations.*
- int `iter_outer`  
*Total GCR iterations.*
- int `iter_inner`  
*Total GMRES iterations.*
- bool `GCR_Output` = true  
*True = print GCR messages.*
- bool `GMRES_Output` = false  
*True = print GMRES messages.*
- double `gmres_tol` = 0.1  
*Tolerance relative to GCR iterations.*
- double `gcr_rel_tol` = 1e-6  
*Relative outer residual tolerance.*
- double `gcr_abs_tol` = 1e-6  
*Absolute outer residual tolerance.*
- `Matrix< double > arg`  
*Argument matrix passed between preconditioner and iterator.*
- `GCR_DATA gcr_dat`  
*Data structure for the outer GCR steps.*
- `GMRESRP_DATA gmres_dat`  
*Data structure for the inner GMRES steps.*
- `int(* matvec)(const Matrix< double > &x, Matrix< double > &Ax, const void *matvec_data)`  
*User supplied matrix-vector product function.*
- `int(* terminal_precon)(const Matrix< double > &r, Matrix< double > &p, const void *precon_data)`  
*Optional user supplied terminal preconditioner.*
- `const void * matvec_data`  
*Data structure for the user's matvec function.*
- `const void * term_precon`  
*Data structure for the user's terminal preconditioner.*

### 4.12.1 Detailed Description

Data structure for the implementation of GCR with Nested GMRES preconditioning (i.e., GMRESR)

C-style object to be used in conjunction with the Generalized Minimum RESidual Recurive (GMRESR) algorithm. Although the name suggests that this method used GMRES recursively, what it is actually doing is nesting GMRESRP iterations inside the GCR method to form a preconditioner for GCR. The name GMRESR came from literature (Vorst and Vuik, "GMRESR: A family of nested GMRES methods", 1991).



### 4.12.2 Member Data Documentation

4.12.2.1 `int GMRESR_DATA::gcr_restart = -1`

Number of GCR restarts (default = 50, max = N)

4.12.2.2 `int GMRESR_DATA::gcr_maxit = 0`

Number of GCR iterations.

4.12.2.3 `int GMRESR_DATA::gmres_restart = -1`

Number of GMRES restarts (max = 20)

4.12.2.4 `int GMRESR_DATA::gmres_maxit = 1`

Number of GMRES iterations (max = 5, default = 1)

4.12.2.5 `int GMRESR_DATA::N`

Dimension of the linear system.

4.12.2.6 `int GMRESR_DATA::total_iter`

Total GMRES and GCR iterations.

4.12.2.7 `int GMRESR_DATA::iter_outer`

Total GCR iterations.

4.12.2.8 `int GMRESR_DATA::iter_inner`

Total GMRES iterations.

4.12.2.9 `bool GMRESR_DATA::GCR_Output = true`

True = print GCR messages.

4.12.2.10 `bool GMRESR_DATA::GMRES_Output = false`

True = print GMRES messages.

4.12.2.11 `double GMRESR_DATA::gmres_tol = 0.1`

Tolerance relative to GCR iterations.

4.12.2.12 `double GMRESR_DATA::gcr_rel_tol = 1e-6`

Relative outer residual tolerance.

4.12.2.13 `double GMRESR_DATA::gcr_abs_tol = 1e-6`

Absolute outer residual tolerance.

4.12.2.14 `Matrix<double> GMRESR_DATA::arg`

Argument matrix passed between preconditioner and iterator.

4.12.2.15 `GCR_DATA GMRESR_DATA::gcr_dat`

Data structure for the outer GCR steps.

4.12.2.16 `GMRESRP_DATA GMRESR_DATA::gmres_dat`

Data structure for the inner GMRES steps.

4.12.2.17 `int(* GMRESR_DATA::matvec)(const Matrix< double > &x, Matrix< double > &Ax, const void *matvec_data)`

User supplied matrix-vector product function.

4.12.2.18 `int(* GMRESR_DATA::terminal_precon)(const Matrix< double > &r, Matrix< double > &p, const void *precon_data)`

Optional user supplied terminal preconditioner.

4.12.2.19 `const void* GMRESR_DATA::matvec_data`

Data structure for the user's matvec function.

4.12.2.20 `const void* GMRESR_DATA::term_precon`

Data structure for the user's terminal preconditioner.

The documentation for this struct was generated from the following file:

- [lark.h](#)

## 4.13 GMRESRP\_DATA Struct Reference

Data structure for the Restarted GMRES algorithm with Right Preconditioning.

```
#include <lark.h>
```

### Public Attributes

- `int restart = -1`  
*Restart parameter - default = min(50,vector\_size)*
- `int maxit = 0`  
*Maximum allowable outer iterations.*
- `int iter_outer = 0`

- *Total number of outer iterations.*  
int `iter_inner` = 0
- *Total number of inner iterations.*  
int `iter_total` = 0
- *Total number of overall iterations.*  
double `tol_rel` = 1e-6
- *Relative tolerance for convergence - default = 1e-6.*  
double `tol_abs` = 1e-6
- *Absolute tolerance for convergence - default = 1e-6.*  
double `res`
- *Absolute residual norm for linear system.*  
double `relres`
- *Relative residual norm for linear system.*  
double `relres_base`
- *Initial residual norm of the linear system.*  
double `bestres`
- *Best found residual norm of the linear system.*  
bool `Output` = true
- *True = print messages to console.*  
Matrix< double > `x`
- *Current solution to the linear system.*  
Matrix< double > `bestx`
- *Best found solution to the linear system.*  
Matrix< double > `r`
- *Residual vector for the linear system.*  
std::vector< Matrix< double > > `Vk`  
*(N x k) orthonormal vector basis*
- std::vector< std::vector< double > > `H`  
*(k+1 x k) upper Hessenberg storage matrix*
- std::vector< std::vector< double > > `H_bar`  
*(k+1 x k) Factorized matrix*
- std::vector< double > `y`  
*(k x 1) Vector search direction*
- std::vector< double > `e0`  
*(k+1 x 1) Normalized vector with residual info*
- std::vector< double > `e0_bar`  
*(k+1 x 1) Factorized normal vector*
- Matrix< double > `w`  
*(N) x (1) interim result of the matrix\_vector multiplication*
- Matrix< double > `v`  
*(N) x (1) holding cell for the column entries of Vk and other interims*
- Matrix< double > `sum`  
*(N) x (1) running sum of subspace vectors for use in altering w*

#### 4.13.1 Detailed Description

Data structure for the Restarted GMRES algorithm with Right Preconditioning.

C-style object used in conjunction with Generalized Minimum RESidual Right Preconditioned (GMRESRP) algorithm to iteratively solve a linear system of equations. Unlike GMRESLP, the GMRESRP method is capable of checking linear residuals at both the inner and outer steps. As a result, this algorithm may terminate earlier than GMRESLP if it has found a suitable solution during one of the inner steps.

### 4.13.2 Member Data Documentation

#### 4.13.2.1 `int GMRESRP_DATA::restart = -1`

Restart parameter - default = min(50,vector\_size)

#### 4.13.2.2 `int GMRESRP_DATA::maxit = 0`

Maximum allowable outer iterations.

#### 4.13.2.3 `int GMRESRP_DATA::iter_outer = 0`

Total number of outer iterations.

#### 4.13.2.4 `int GMRESRP_DATA::iter_inner = 0`

Total number of inner iterations.

#### 4.13.2.5 `int GMRESRP_DATA::iter_total = 0`

Total number of overall iterations.

#### 4.13.2.6 `double GMRESRP_DATA::tol_rel = 1e-6`

Relative tolerance for convergence - default = 1e-6.

#### 4.13.2.7 `double GMRESRP_DATA::tol_abs = 1e-6`

Absolute tolerance for convergence - default = 1e-6.

#### 4.13.2.8 `double GMRESRP_DATA::res`

Absolute residual norm for linear system.

#### 4.13.2.9 `double GMRESRP_DATA::relres`

Relative residual norm for linear system.

#### 4.13.2.10 `double GMRESRP_DATA::relres_base`

Initial residual norm of the linear system.

#### 4.13.2.11 `double GMRESRP_DATA::bestres`

Best found residual norm of the linear system.

#### 4.13.2.12 `bool GMRESRP_DATA::Output = true`

True = print messages to console.

**4.13.2.13** `Matrix<double> GMRESRP_DATA::x`

Current solution to the linear system.

**4.13.2.14** `Matrix<double> GMRESRP_DATA::bestx`

Best found solution to the linear system.

**4.13.2.15** `Matrix<double> GMRESRP_DATA::r`

Residual vector for the linear system.

**4.13.2.16** `std::vector< Matrix<double> > GMRESRP_DATA::Vk`

(N x k) orthonormal vector basis

**4.13.2.17** `std::vector< std::vector< double > > GMRESRP_DATA::H`

(k+1 x k) upper Hessenberg storage matrix

**4.13.2.18** `std::vector< std::vector< double > > GMRESRP_DATA::H_bar`

(k+1 x k) Factorized matrix

**4.13.2.19** `std::vector< double > GMRESRP_DATA::y`

(k x 1) Vector search direction

**4.13.2.20** `std::vector< double > GMRESRP_DATA::e0`

(k+1 x 1) Normalized vector with residual info

**4.13.2.21** `std::vector< double > GMRESRP_DATA::e0_bar`

(k+1 x 1) Factorized normal vector

**4.13.2.22** `Matrix<double> GMRESRP_DATA::w`

(N) x (1) interim result of the matrix\_vector multiplication

**4.13.2.23** `Matrix<double> GMRESRP_DATA::v`

(N) x (1) holding cell for the column entries of Vk and other interims

**4.13.2.24** `Matrix<double> GMRESRP_DATA::sum`

(N) x (1) running sum of subspace vectors for use in altering w

The documentation for this struct was generated from the following file:

- [lark.h](#)

## 4.14 GPAST\_DATA Struct Reference

GPAST Data Structure.

```
#include <magpie.h>
```

### Public Attributes

- double [x](#)  
*Adsorbed mole fraction.*
- double [y](#)  
*Gas phase mole fraction.*
- double [He](#)  
*Henry's Coefficient (mol/kg/kPa)*
- double [q](#)  
*Amount adsorbed for each component (mol/kg)*
- std::vector< double > [gama\\_inf](#)  
*Infinite dilution activities.*
- double [qo](#)  
*Pure component capacities (mol/kg)*
- double [Plo](#)  
*Pure component spreading pressures (mol/kg)*
- std::vector< double > [po](#)  
*Pure component reference state pressures (kPa)*
- double [poi](#)  
*Reference state pressures solved for using Recover eval GPAST.*
- bool [present](#)  
*If true, then the component is present; if false, then the component is not present.*

### 4.14.1 Detailed Description

GPAST Data Structure.

C-style object holding all parameter information associated with the Generalized Predictive Adsorbed Solution Theory (GPAST) system of equations. Each species in the gas phase will have one of these objects.

### 4.14.2 Member Data Documentation

#### 4.14.2.1 double GPAST\_DATA::x

Adsorbed mole fraction.

#### 4.14.2.2 double GPAST\_DATA::y

Gas phase mole fraction.

#### 4.14.2.3 double GPAST\_DATA::He

Henry's Coefficient (mol/kg/kPa)

## 4.14.2.4 double GPAST\_DATA::q

Amount adsorbed for each component (mol/kg)

## 4.14.2.5 std::vector&lt;double&gt; GPAST\_DATA::gama\_inf

Infinite dilution activities.

## 4.14.2.6 double GPAST\_DATA::qo

Pure component capacities (mol/kg)

## 4.14.2.7 double GPAST\_DATA::Plo

Pure component spreading pressures (mol/kg)

## 4.14.2.8 std::vector&lt;double&gt; GPAST\_DATA::po

Pure component reference state pressures (kPa)

## 4.14.2.9 double GPAST\_DATA::poi

Reference state pressures solved for using Recover eval GPAST.

## 4.14.2.10 bool GPAST\_DATA::present

If true, then the component is present; if false, then the component is not present.

The documentation for this struct was generated from the following file:

- [magpie.h](#)

## 4.15 GSTA\_DATA Struct Reference

GSTA Data Structure.

```
#include <magpie.h>
```

### Public Attributes

- double [qmax](#)  
*Theoretical maximum capacity of adsorbate-adsorbent pair (mol/kg)*
- int [m](#)  
*Number of parameters in the GSTA isotherm.*
- std::vector< double > [dHo](#)  
*Enthalpies for each site (J/mol)*
- std::vector< double > [dSo](#)  
*Entropies for each site (J/(K\*mol))*

### 4.15.1 Detailed Description

GSTA Data Structure.

C-style object holding all parameter information associated with the Generalized Statistical Thermodynamic Adsorption (GSTA) isotherm model. Each species in the gas phase will have one of these objects.

### 4.15.2 Member Data Documentation

#### 4.15.2.1 `double GSTA_DATA::qmax`

Theoretical maximum capacity of adsorbate-adsorbent pair (mol/kg)

#### 4.15.2.2 `int GSTA_DATA::m`

Number of parameters in the GSTA isotherm.

#### 4.15.2.3 `std::vector<double> GSTA_DATA::dHo`

Enthalpies for each site (J/mol)

#### 4.15.2.4 `std::vector<double> GSTA_DATA::dSo`

Entropies for each site (J/(K\*mol))

The documentation for this struct was generated from the following file:

- [magpie.h](#)

## 4.16 GSTA\_OPT\_DATA Struct Reference

Data structure used in the GSTA optimization routines.

```
#include <gsta_opt.h>
```

### Public Attributes

- `int total_eval`  
*Keeps track of the total number of function evaluations.*
- `int n_par`  
*Number of parameters being optimized for.*
- `double qmax`  
*Maximum theoretical adsorption capacity (M/M) (0 if unknown)*
- `int iso`  
*Keeps isotherm that is currently being optimized.*
- `std::vector< std::vector< double > > Fobj`  
*Creates a dynamic array to store all Fobj values.*
- `std::vector< std::vector< double > > q`
- `std::vector< std::vector< double > > P`



*Creates a dynamic array for q and P data pairs.*

- `std::vector< std::vector< double > >` `best_par`

*Used to store the values of the parameters of best fit.*

- `std::vector< std::vector< double > >` `Kno`

*Dimensionless parameters determined from best\_par.*

- `std::vector< std::vector< std::vector< double > > >` `all_pars`

*Used to create a ragged array of all parameters.*

- `std::vector< std::vector< double > >` `norms`

*Used to store the values of all the calculated norms.*

- `std::vector< double >` `opt_qmax`

*If qmax is unknown, this vector holds it's optimized values.*

### 4.16.1 Detailed Description

Data structure used in the GSTA optimization routines.

C-style structure that keeps track of all information during the optimization routine. All solutions and parameters to the GSTA isotherm are held in order to find the best solution with the fewest parameters.

### 4.16.2 Member Data Documentation

#### 4.16.2.1 `int GSTA_OPT_DATA::total_eval`

Keeps track of the total number of function evaluations.

#### 4.16.2.2 `int GSTA_OPT_DATA::n_par`

Number of parameters being optimized for.

#### 4.16.2.3 `double GSTA_OPT_DATA::qmax`

Maximum theoretical adsorption capacity (M/M) (0 if unknown)

#### 4.16.2.4 `int GSTA_OPT_DATA::iso`

Keeps isotherm that is currently being optimized.

#### 4.16.2.5 `std::vector<std::vector<double> > GSTA_OPT_DATA::Fobj`

Creates a dynamic array to store all Fobj values.

#### 4.16.2.6 `std::vector<std::vector<double> > GSTA_OPT_DATA::q`

#### 4.16.2.7 `std::vector<std::vector<double> > GSTA_OPT_DATA::P`

Creates a dynamic array for q and P data pairs.

#### 4.16.2.8 `std::vector<std::vector<double>> GSTA_OPT_DATA::best_par`

Used to store the values of the parameters of best fit.

#### 4.16.2.9 `std::vector<std::vector<double>> GSTA_OPT_DATA::Kno`

Dimensionless parameters determined from best\_par.

#### 4.16.2.10 `std::vector<std::vector<std::vector<double>>> GSTA_OPT_DATA::all_pars`

Used to create a ragged array of all parameters.

#### 4.16.2.11 `std::vector<std::vector<double>> GSTA_OPT_DATA::norms`

Used to store the values of all the calculated norms.

#### 4.16.2.12 `std::vector<double> GSTA_OPT_DATA::opt_qmax`

If qmax is unknown, this vector holds it's optimized values.

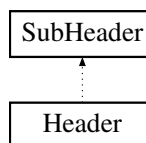
The documentation for this struct was generated from the following file:

- [gsta\\_opt.h](#)

## 4.17 Header Class Reference

```
#include <yaml_wrapper.h>
```

Inheritance diagram for Header:



### Public Member Functions

- [Header](#) ()
- [~Header](#) ()
- [Header](#) (const [Header](#) &head)
- [Header](#) (std::string [name](#))
- [Header](#) (const [KeyValueMap](#) &map)
- [Header](#) (std::string [name](#), const [KeyValueMap](#) &map)
- [Header](#) (std::string key, const [SubHeader](#) &sub)
- [Header](#) & [operator=](#) (const [Header](#) &head)
- [ValueTypePair](#) & [operator\[\]](#) (const std::string key)
- [ValueTypePair](#) [operator\[\]](#) (const std::string key) const
- [SubHeader](#) & [operator\(\)](#) (const std::string key)
- [SubHeader](#) [operator\(\)](#) (const std::string key) const
- std::map< std::string, [SubHeader](#) > & [getSubMap](#) ()

- [KeyValueMap](#) & [getDataMap](#) ()
- [SubHeader](#) & [getSubHeader](#) (std::string key)
- std::map< std::string, [SubHeader](#) >::const\_iterator [end](#) () const
- std::map< std::string, [SubHeader](#) >::iterator [end](#) ()
- std::map< std::string, [SubHeader](#) >::const\_iterator [begin](#) () const
- std::map< std::string, [SubHeader](#) >::iterator [begin](#) ()
- void [clear](#) ()
- void [resetKeys](#) ()
- void [changeKey](#) (std::string oldKey, std::string newKey)
- void [addPair](#) (std::string key, std::string val)
- void [addPair](#) (std::string key, std::string val, int t)
- void [setName](#) (std::string name)
- void [setAlias](#) (std::string alias)
- void [setNameAliasPair](#) (std::string n, std::string a, int s)
- void [setState](#) (int state)
- void [DisplayContents](#) ()
- void [addSubKey](#) (std::string key)
- void [copyAnchor2Alias](#) (std::string alias, [SubHeader](#) &ref)
- int [size](#) ()
- std::string [getName](#) ()
- std::string [getAlias](#) ()
- int [getState](#) ()
- bool [isAlias](#) ()
- bool [isAnchor](#) ()
- [SubHeader](#) & [getAnchoredSub](#) (std::string alias)

### Private Attributes

- std::map< std::string, [SubHeader](#) > [Sub\\_Map](#)

### Additional Inherited Members

#### 4.17.1 Constructor & Destructor Documentation

4.17.1.1 [Header::Header](#) ( )

4.17.1.2 [Header::~Header](#) ( )

4.17.1.3 [Header::Header](#) ( const [Header](#) & *head* )

4.17.1.4 [Header::Header](#) ( std::string *name* )

4.17.1.5 [Header::Header](#) ( const [KeyValueMap](#) & *map* )

4.17.1.6 [Header::Header](#) ( std::string *name*, const [KeyValueMap](#) & *map* )

4.17.1.7 [Header::Header](#) ( std::string *key*, const [SubHeader](#) & *sub* )

#### 4.17.2 Member Function Documentation

- 4.17.2.1 `Header& Header::operator= ( const Header & head )`
- 4.17.2.2 `ValueTypePair& Header::operator[] ( const std::string key )`
- 4.17.2.3 `ValueTypePair Header::operator[] ( const std::string key ) const`
- 4.17.2.4 `SubHeader& Header::operator() ( const std::string key )`
- 4.17.2.5 `SubHeader Header::operator() ( const std::string key ) const`
- 4.17.2.6 `std::map<std::string, SubHeader>& Header::getSubMap ( )`
- 4.17.2.7 `KeyValueMap& Header::getDataMap ( )`
- 4.17.2.8 `SubHeader& Header::getSubHeader ( std::string key )`
- 4.17.2.9 `std::map<std::string, SubHeader>::const_iterator Header::end ( ) const`
- 4.17.2.10 `std::map<std::string, SubHeader>::iterator Header::end ( )`
- 4.17.2.11 `std::map<std::string, SubHeader>::const_iterator Header::begin ( ) const`
- 4.17.2.12 `std::map<std::string, SubHeader>::iterator Header::begin ( )`
- 4.17.2.13 `void Header::clear ( )`
- 4.17.2.14 `void Header::resetKeys ( )`
- 4.17.2.15 `void Header::changeKey ( std::string oldKey, std::string newKey )`
- 4.17.2.16 `void Header::addPair ( std::string key, std::string val )`
- 4.17.2.17 `void Header::addPair ( std::string key, std::string val, int t )`
- 4.17.2.18 `void Header::setName ( std::string name )`
- 4.17.2.19 `void Header::setAlias ( std::string alias )`
- 4.17.2.20 `void Header::setNameAliasPair ( std::string n, std::string a, int s )`
- 4.17.2.21 `void Header::setState ( int state )`
- 4.17.2.22 `void Header::DisplayContents ( )`
- 4.17.2.23 `void Header::addSubKey ( std::string key )`
- 4.17.2.24 `void Header::copyAnchor2Alias ( std::string alias, SubHeader & ref )`
- 4.17.2.25 `int Header::size ( )`
- 4.17.2.26 `std::string Header::getName ( )`
- 4.17.2.27 `std::string Header::getAlias ( )`
- 4.17.2.28 `int Header::getState ( )`

4.17.2.29 `bool Header::isAlias ( )`

4.17.2.30 `bool Header::isAnchor ( )`

4.17.2.31 `SubHeader& Header::getAnchoredSub ( std::string alias )`

### 4.17.3 Member Data Documentation

4.17.3.1 `std::map<std::string, SubHeader> Header::Sub_Map [private]`

The documentation for this class was generated from the following file:

- [yaml\\_wrapper.h](#)

## 4.18 KeyValueType Class Reference

```
#include <yaml_wrapper.h>
```

### Public Member Functions

- [KeyValueType](#) ( )
- [~KeyValueType](#) ( )
- [KeyValueType](#) (const std::map< std::string, std::string > &map)
- [KeyValueType](#) (std::string key, std::string value)
- [KeyValueType](#) (const [KeyValueType](#) &map)
- [KeyValueType](#) & [operator=](#) (const [KeyValueType](#) &map)
- [ValueTypePair](#) & [operator\[\]](#) (const std::string key)
- [ValueTypePair](#) [operator\[\]](#) (const std::string key) const
- std::map< std::string, [ValueTypePair](#) > & [getMap](#) ( )
- std::map< std::string, [ValueTypePair](#) > ::const\_iterator [end](#) ( ) const
- std::map< std::string, [ValueTypePair](#) > ::iterator [end](#) ( )
- std::map< std::string, [ValueTypePair](#) > ::const\_iterator [begin](#) ( ) const
- std::map< std::string, [ValueTypePair](#) > ::iterator [begin](#) ( )
- void [clear](#) ( )
- void [addKey](#) (std::string key)
- void [editValue4Key](#) (std::string val, std::string key)
- void [editValue4Key](#) (std::string val, int type, std::string key)
- void [addPair](#) (std::string key, [ValueTypePair](#) val)
- void [addPair](#) (std::string key, std::string val)
- void [addPair](#) (std::string key, std::string val, int type)
- void [findType](#) (std::string key)
- void [assertType](#) (std::string key, int type)
- void [findAllTypes](#) ( )
- void [DisplayMap](#) ( )
- int [size](#) ( )
- std::string [getString](#) (std::string key)

- bool [getBool](#) (std::string key)
- double [getDouble](#) (std::string key)
- int [getInt](#) (std::string key)
- std::string [getValue](#) (std::string key)
- int [getType](#) (std::string key)
- [ValueTypePair](#) & [getPair](#) (std::string key)

### Private Attributes

- std::map< std::string,  
[ValueTypePair](#) > [Key\\_Value](#)

### 4.18.1 Constructor & Destructor Documentation

4.18.1.1 [KeyValueMap::KeyValueMap](#) ( )

4.18.1.2 [KeyValueMap::~~KeyValueMap](#) ( )

4.18.1.3 [KeyValueMap::KeyValueMap](#) ( const std::map< std::string, std::string > & *map* )

4.18.1.4 [KeyValueMap::KeyValueMap](#) ( std::string *key*, std::string *value* )

4.18.1.5 [KeyValueMap::KeyValueMap](#) ( const [KeyValueMap](#) & *map* )

### 4.18.2 Member Function Documentation

4.18.2.1 [KeyValueMap& KeyValueMap::operator=](#) ( const [KeyValueMap](#) & *map* )

4.18.2.2 [ValueTypePair& KeyValueMap::operator\[\]](#) ( const std::string *key* )

4.18.2.3 [ValueTypePair](#) [KeyValueMap::operator\[\]](#) ( const std::string *key* ) const

4.18.2.4 std::map<std::string, [ValueTypePair](#) >& [KeyValueMap::getMap](#) ( )

4.18.2.5 std::map<std::string, [ValueTypePair](#)>::const\_iterator [KeyValueMap::end](#) ( ) const

4.18.2.6 std::map<std::string, [ValueTypePair](#)>::iterator [KeyValueMap::end](#) ( )

4.18.2.7 std::map<std::string, [ValueTypePair](#)>::const\_iterator [KeyValueMap::begin](#) ( ) const

4.18.2.8 std::map<std::string, [ValueTypePair](#)>::iterator [KeyValueMap::begin](#) ( )

4.18.2.9 void [KeyValueMap::clear](#) ( )

4.18.2.10 void [KeyValueMap::addKey](#) ( std::string *key* )

4.18.2.11 void [KeyValueMap::editValue4Key](#) ( std::string *val*, std::string *key* )

4.18.2.12 void [KeyValueMap::editValue4Key](#) ( std::string *val*, int *type*, std::string *key* )

4.18.2.13 void [KeyValueMap::addPair](#) ( std::string *key*, [ValueTypePair](#) *val* )

4.18.2.14 void [KeyValueMap::addPair](#) ( std::string *key*, std::string *val* )

4.18.2.15 void KeyValueTypeMap::addPair ( std::string key, std::string val, int type )

4.18.2.16 void KeyValueTypeMap::findType ( std::string key )

4.18.2.17 void KeyValueTypeMap::assertType ( std::string key, int type )

4.18.2.18 void KeyValueTypeMap::findAllTypes ( )

4.18.2.19 void KeyValueTypeMap::DisplayMap ( )

4.18.2.20 int KeyValueTypeMap::size ( )

4.18.2.21 std::string KeyValueTypeMap::getString ( std::string key )

4.18.2.22 bool KeyValueTypeMap::getBool ( std::string key )

4.18.2.23 double KeyValueTypeMap::getDouble ( std::string key )

4.18.2.24 int KeyValueTypeMap::getInt ( std::string key )

4.18.2.25 std::string KeyValueTypeMap::getValue ( std::string key )

4.18.2.26 int KeyValueTypeMap::getType ( std::string key )

4.18.2.27 ValueTypePair& KeyValueTypeMap::getPair ( std::string key )

### 4.18.3 Member Data Documentation

4.18.3.1 std::map<std::string, ValueTypePair> KeyValueTypeMap::Key\_Value [private]

The documentation for this class was generated from the following file:

- [yaml\\_wrapper.h](#)

## 4.19 MAGPIE\_DATA Struct Reference

MAGPIE Data Structure.

```
#include <magpie.h>
```

### Public Attributes

- std::vector< [GSTA\\_DATA](#) > gsta\_dat
- std::vector< [mSPD\\_DATA](#) > mspd\_dat
- std::vector< [GPAST\\_DATA](#) > gpast\_dat
- [SYSTEM\\_DATA](#) sys\_dat

### 4.19.1 Detailed Description

MAGPIE Data Structure.

C-style object holding all information necessary to run a MAGPIE simulation. This is the data structure that will be used in other sub-routines when a mixed gas adsorption simulation needs to be run.

### 4.19.2 Member Data Documentation

4.19.2.1 `std::vector<GSTA_DATA> MAGPIE_DATA::gsta_dat`

4.19.2.2 `std::vector<mSPD_DATA> MAGPIE_DATA::mspd_dat`

4.19.2.3 `std::vector<GPAST_DATA> MAGPIE_DATA::gpast_dat`

4.19.2.4 `SYSTEM_DATA MAGPIE_DATA::sys_dat`

The documentation for this struct was generated from the following file:

- [magpie.h](#)

## 4.20 MassBalance Class Reference

```
#include <shark.h>
```

### Public Member Functions

- [MassBalance](#) ()
- [~MassBalance](#) ()
- void [Initialize\\_List](#) ([MasterSpeciesList](#) &[List](#))
- void [Display\\_Info](#) ()
- void [Set\\_Delta](#) (int i, double v)
- void [Set\\_TotalConcentration](#) (double v)
- void [Set\\_Name](#) (std::string name)
- double [Get\\_Delta](#) (int i)
- double [Sum\\_Delta](#) ()
- double [Get\\_TotalConcentration](#) ()
- std::string [Get\\_Name](#) ()
- double [Eval\\_Residual](#) (const [Matrix](#)< double > &x)

### Protected Attributes

- [MasterSpeciesList](#) \* [List](#)
- std::vector< double > [Delta](#)
- double [TotalConcentration](#)

### Private Attributes

- std::string [Name](#)

### 4.20.1 Constructor & Destructor Documentation

4.20.1.1 `MassBalance::MassBalance ( )`

4.20.1.2 `MassBalance::~~MassBalance ( )`

### 4.20.2 Member Function Documentation



- 4.20.2.1 void MassBalance::Initialize\_List ( MasterSpeciesList & List )
- 4.20.2.2 void MassBalance::Display\_Info ( )
- 4.20.2.3 void MassBalance::Set\_Delta ( int *i*, double *v* )
- 4.20.2.4 void MassBalance::Set\_TotalConcentration ( double *v* )
- 4.20.2.5 void MassBalance::Set\_Name ( std::string *name* )
- 4.20.2.6 double MassBalance::Get\_Delta ( int *i* )
- 4.20.2.7 double MassBalance::Sum\_Delta ( )
- 4.20.2.8 double MassBalance::Get\_TotalConcentration ( )
- 4.20.2.9 std::string MassBalance::Get\_Name ( )
- 4.20.2.10 double MassBalance::Eval\_Residual ( const Matrix< double > & *x* )

### 4.20.3 Member Data Documentation

- 4.20.3.1 MasterSpeciesList\* MassBalance::List [protected]
- 4.20.3.2 std::vector<double> MassBalance::Delta [protected]
- 4.20.3.3 double MassBalance::TotalConcentration [protected]
- 4.20.3.4 std::string MassBalance::Name [private]

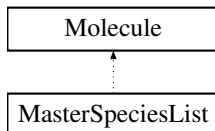
The documentation for this class was generated from the following file:

- [shark.h](#)

## 4.21 MasterSpeciesList Class Reference

```
#include <shark.h>
```

Inheritance diagram for MasterSpeciesList:



### Public Member Functions

- [MasterSpeciesList](#) ()
- [~MasterSpeciesList](#) ()
- [MasterSpeciesList](#) (const [MasterSpeciesList](#) &msl)
- [MasterSpeciesList](#) & [operator=](#) (const [MasterSpeciesList](#) &msl)
- void [set\\_list\\_size](#) (int *i*)
- void [set\\_species](#) (int *i*, std::string *formula*)

- void `set_species` (int i, int `charge`, double `enthalpy`, double `entropy`, double `energy`, bool `HS`, bool `G`, std::string `Phase`, std::string `Name`, std::string `Formula`, std::string `lin_formula`)
- void `DisplayInfo` (int i)
- void `DisplayAll` ()
- void `DisplayConcentrations` (Matrix< double > &C)
- void `set_alkalinity` (double `alk`)
- int `list_size` ()
- Molecule & `get_species` (int i)
- int `get_index` (std::string `name`)
- double `charge` (int i)
- double `alkalinity` ()
- std::string `speciesName` (int i)
- double `Eval_ChargeResidual` (const Matrix< double > &x)

### Protected Attributes

- int `size`
- std::vector< Molecule > `species`
- double `residual_alkalinity`

### Additional Inherited Members

#### 4.21.1 Constructor & Destructor Documentation

- 4.21.1.1 MasterSpeciesList::MasterSpeciesList ( )
- 4.21.1.2 MasterSpeciesList::~~MasterSpeciesList ( )
- 4.21.1.3 MasterSpeciesList::MasterSpeciesList ( const MasterSpeciesList & *msl* )

#### 4.21.2 Member Function Documentation

- 4.21.2.1 MasterSpeciesList& MasterSpeciesList::operator= ( const MasterSpeciesList & *msl* )
- 4.21.2.2 void MasterSpeciesList::set\_list\_size ( int *i* )
- 4.21.2.3 void MasterSpeciesList::set\_species ( int *i*, std::string *formula* )
- 4.21.2.4 void MasterSpeciesList::set\_species ( int *i*, int *charge*, double *enthalpy*, double *entropy*, double *energy*, bool *HS*, bool *G*, std::string *Phase*, std::string *Name*, std::string *Formula*, std::string *lin\_formula* )
- 4.21.2.5 void MasterSpeciesList::DisplayInfo ( int *i* )
- 4.21.2.6 void MasterSpeciesList::DisplayAll ( )
- 4.21.2.7 void MasterSpeciesList::DisplayConcentrations ( Matrix< double > & *C* )
- 4.21.2.8 void MasterSpeciesList::set\_alkalinity ( double *alk* )
- 4.21.2.9 int MasterSpeciesList::list\_size ( )
- 4.21.2.10 Molecule& MasterSpeciesList::get\_species ( int *i* )
- 4.21.2.11 int MasterSpeciesList::get\_index ( std::string *name* )

4.21.2.12 double MasterSpeciesList::charge ( int *i* )

4.21.2.13 double MasterSpeciesList::alkalinity ( )

4.21.2.14 std::string MasterSpeciesList::speciesName ( int *i* )

4.21.2.15 double MasterSpeciesList::Eval\_ChargeResidual ( const Matrix< double > & *x* )

### 4.21.3 Member Data Documentation

4.21.3.1 int MasterSpeciesList::size [protected]

4.21.3.2 std::vector<Molecule> MasterSpeciesList::species [protected]

4.21.3.3 double MasterSpeciesList::residual\_alkalinity [protected]

The documentation for this class was generated from the following file:

- [shark.h](#)

## 4.22 Matrix< T > Class Template Reference

Templated C++ [Matrix](#) Class Object (click [Matrix](#) to go to function definitions)

```
#include <macaw.h>
```

### Public Member Functions

- [Matrix](#) (int [rows](#), int [columns](#))  
*Constructor for matrix with given number of rows and columns.*
- T & [operator](#)() (int *i*, int *j*)  
*Access operator for the matrix element at row *i* and column *j* (e.g.,  $a_{ij} = A(i,j)$ )*
- T [operator](#)() (int *i*, int *j*) const  
*Constant access operator for the the matrix element at row *i* and column *j*.*
- [Matrix](#) (const [Matrix](#) &M)  
*Copy constructor for constructing a matrix as a copy of another matrix.*
- [Matrix](#) & [operator=](#) (const [Matrix](#) &M)  
*Equals operator for setting one matrix equal to another matrix.*
- [Matrix](#) ()  
*Default constructor for creating an empty matrix.*
- [~Matrix](#) ()  
*Default destructor for clearing out memory.*
- void [set\\_size](#) (int *i*, int *j*)  
*Function to set/change the size of a matrix to *i* rows and *j* columns.*
- void [zeros](#) ()  
*Function to set/change all values in a matrix to zeros.*
- void [edit](#) (int *i*, int *j*, T value)  
*Function to set/change the element of a matrix at row *i* and column *j* to given value.*
- int [rows](#) ()  
*Function to return the number of rows in a given matrix.*
- int [columns](#) ()  
*Function to return the number of columns in a matrix.*

- T [determinate](#) ()  
*Function to compute the determinate of a matrix and return that value.*
- T [norm](#) ()  
*Function to compute the L2-norm of a matrix and return that value.*
- T [sum](#) ()  
*Function to compute the sum of all elements in a matrix and return that value.*
- T [inner\\_product](#) (const [Matrix](#) &x)  
*Function to compute the inner product between this matrix and matrix x.*
- [Matrix](#) & [cofactor](#) (const [Matrix](#) &M)  
*Function to convert this matrix to a cofactor matrix of the given matrix M.*
- [Matrix](#) operator+ (const [Matrix](#) &M)  
*Operator to add this matrix and matrix M and return the new matrix result.*
- [Matrix](#) operator- (const [Matrix](#) &M)  
*Operator to subtract this matrix and matrix M and return the new matrix result.*
- [Matrix](#) operator\* (const T)  
*Operator to multiply this matrix by a scalar T return the new matrix result.*
- [Matrix](#) operator/ (const T)  
*Operator to divide this matrix by a scalar T and return the new matrix result.*
- [Matrix](#) operator\* (const [Matrix](#) &M)  
*Operator to multiply this matrix and matrix M and return the new matrix result.*
- [Matrix](#) & [transpose](#) (const [Matrix](#) &M)  
*Function to convert this matrix to the transpose of the given matrix M.*
- [Matrix](#) & [transpose\\_multiply](#) (const [Matrix](#) &MT, const [Matrix](#) &v)  
*Function to convert this matrix into the result of the given matrix M transposed and multiplied by the other given matrix v.*
- [Matrix](#) & [adjoint](#) (const [Matrix](#) &M)  
*Function to convert this matrix to the adjoint of the given matrix.*
- [Matrix](#) & [inverse](#) (const [Matrix](#) &M)  
*Function to convert this matrix to the inverse of the given matrix.*
- void [Display](#) (const std::string Name)  
*Function to display the contents of this matrix given a Name for the matrix.*
- [Matrix](#) & [tridiagonalSolve](#) (const [Matrix](#) &A, const [Matrix](#) &b)  
*Function to solve  $Ax=b$  for x if A is symmetric, tridiagonal (this->x)*
- [Matrix](#) & [ladshawSolve](#) (const [Matrix](#) &A, const [Matrix](#) &d)  
*Function to solve  $Ax=d$  for x if A is non-symmetric, tridiagonal (this->x)*
- [Matrix](#) & [tridiagonalFill](#) (const T A, const T B, const T C, bool [Spherical](#))  
*Function to fill in this matrix with coefficients A, B, and C to form a tridiagonal matrix.*
- [Matrix](#) & [naturalLaplacian3D](#) (int m)  
*Function to fill out this matrix with coefficients from a 3D Laplacian function.*
- [Matrix](#) & [sphericalBCFill](#) (int node, const T coeff, T variable)  
*Function to fill out a column matrix with spherical specific boundary conditions.*
- [Matrix](#) & [ConstantICFill](#) (const T IC)  
*Function to set all values in a column matrix to a given constant.*
- [Matrix](#) & [SolnTransform](#) (const [Matrix](#) &A, bool Forward)  
*Function to transform the values in a column matrix from cartesian to spherical coordinates.*
- T [sphericalAvg](#) (double radius, double dr, double bound, bool Dirichlet)  
*Function to compute a spatial average of this column matrix in spherical coordinates.*
- T [IntegralAvg](#) (double radius, double dr, double bound, bool Dirichlet)  
*Function to compute a spatial average of this column matrix in spherical coordinates.*
- T [IntegralTotal](#) (double dr, double bound, bool Dirichlet)  
*Function to compute a spatial total of this column matrix in spherical coordinates.*

- [Matrix](#) & [tridiagonalVectorFill](#) (const std::vector< T > &A, const std::vector< T > &B, const std::vector< T > &C)  
*Function to fill in this matrix, in tridiagonal fashion, using the vectors of coefficients.*
- [Matrix](#) & [columnVectorFill](#) (const std::vector< T > &A)  
*Function to fill in a column matrix with the values of the given vector object.*
- [Matrix](#) & [columnProjection](#) (const [Matrix](#) &b, const [Matrix](#) &b\_old, const double dt, const double dt\_old)  
*Function to project a column matrix solution in time based on older state vectors.*
- [Matrix](#) & [dirichletBCFill](#) (int node, const T coeff, T variable)  
*Function to fill in a column matrix with all zeros except at the given node.*
- [Matrix](#) & [diagonalSolve](#) (const [Matrix](#) &D, const [Matrix](#) &v)  
*Function to solve the system  $Dx=v$  for  $x$  given that  $D$  is diagonal (this->x)*
- [Matrix](#) & [upperTriangularSolve](#) (const [Matrix](#) &U, const [Matrix](#) &v)  
*Function to solve the system  $Ux=v$  for  $x$  given that  $U$  is upper Triangular (this->x)*
- [Matrix](#) & [lowerTriangularSolve](#) (const [Matrix](#) &L, const [Matrix](#) &v)  
*Function to solve the system  $Lx=v$  for  $x$  given that  $L$  is lower Triangular (this->x)*
- [Matrix](#) & [upperHessenberg2Triangular](#) ([Matrix](#) &b)  
*Function to convert this square matrix to upper Triangular (assuming this is upper Hessenberg)*
- [Matrix](#) & [lowerHessenberg2Triangular](#) ([Matrix](#) &b)  
*Function to convert this square matrix to lower Triangular (assuming this is lower Hessenberg)*
- [Matrix](#) & [upperHessenbergSolve](#) (const [Matrix](#) &H, const [Matrix](#) &v)  
*Function to solve the system  $Hx=v$  for  $x$  given that  $H$  is upper Hessenberg (this->x)*
- [Matrix](#) & [lowerHessenbergSolve](#) (const [Matrix](#) &H, const [Matrix](#) &v)  
*Function to solve the system  $Hx=v$  for  $x$  given that  $H$  is lower Hessenberg (this->x)*
- [Matrix](#) & [columnExtract](#) (int j, const [Matrix](#) &M)  
*Function to set this column matrix to the jth column of the given matrix M.*
- [Matrix](#) & [rowExtract](#) (int i, const [Matrix](#) &M)  
*Function to set this row matrix to the ith row of the given matrix M.*
- [Matrix](#) & [columnReplace](#) (int j, const [Matrix](#) &v)  
*Function to this matrices' jth column with the given column matrix v.*
- [Matrix](#) & [rowReplace](#) (int i, const [Matrix](#) &v)  
*Function to this matrices' ith row with the given row matrix v.*
- void [rowShrink](#) ()  
*Function to delete the last row of this matrix.*
- void [columnShrink](#) ()  
*Function to delete the last column of this matrix.*
- void [rowExtend](#) (const [Matrix](#) &v)  
*Function to add the row matrix v to the end of this matrix.*
- void [columnExtend](#) (const [Matrix](#) &v)  
*Function to add the column matrix v to the end of this matrix.*

### Protected Attributes

- int [num\\_rows](#)  
*Number of rows of the matrix.*
- int [num\\_cols](#)  
*Number of columns of the matrix.*
- std::vector< T > [Data](#)  
*Storage vector for the elements of the matrix.*

### 4.22.1 Detailed Description

```
template<class T>class Matrix< T >
```

Templated C++ [Matrix](#) Class Object (click [Matrix](#) to go to function definitions)

C++ templated class object containing many different functions, actions, and solver routines associated with Dense Matrices. Operator overloads are also provided to give the user a more natural way of operating matrices on other matrices or scalars. These operator overloads are especially useful for reducing the amount of code needed to be written when working with matrix-based problems.

### 4.22.2 Constructor & Destructor Documentation

4.22.2.1 `template<class T > Matrix< T >::Matrix ( int rows, int columns )`

Constructor for matrix with given number of rows and columns.

4.22.2.2 `template<class T > Matrix< T >::Matrix ( const Matrix< T > & M )`

Copy constructor for constructing a matrix as a copy of another matrix.

4.22.2.3 `template<class T > Matrix< T >::Matrix ( )`

Default constructor for creating an empty matrix.

4.22.2.4 `template<class T > Matrix< T >::~~Matrix ( )`

Default destructor for clearing out memory.

### 4.22.3 Member Function Documentation

4.22.3.1 `template<class T > T & Matrix< T >::operator() ( int i, int j )`

Access operator for the matrix element at row i and column j (e.g.,  $a_{ij} = A(i,j)$ )

4.22.3.2 `template<class T > T Matrix< T >::operator() ( int i, int j ) const`

Constant access operator for the the matrix element at row i and column j.

4.22.3.3 `template<class T > Matrix< T > & Matrix< T >::operator= ( const Matrix< T > & M )`

Equals operator for setting one matrix equal to another matrix.

4.22.3.4 `template<class T > void Matrix< T >::set_size ( int i, int j )`

Function to set/change the size of a matrix to i rows and j columns.

4.22.3.5 `template<class T > void Matrix< T >::zeros ( )`

Function to set/change all values in a matrix to zeros.

4.22.3.6 `template<class T> void Matrix< T >::edit ( int i, int j, T value )`

Function to set/change the element of a matrix at row *i* and column *j* to given value.

4.22.3.7 `template<class T> int Matrix< T >::rows ( )`

Function to return the number of rows in a given matrix.

4.22.3.8 `template<class T> int Matrix< T >::columns ( )`

Function to return the number of columns in a matrix.

4.22.3.9 `template<class T> T Matrix< T >::determinate ( )`

Function to compute the determinate of a matrix and return that value.

4.22.3.10 `template<class T> T Matrix< T >::norm ( )`

Function to compute the L2-norm of a matrix and return that value.

4.22.3.11 `template<class T> T Matrix< T >::sum ( )`

Function to compute the sum of all elements in a matrix and return that value.

4.22.3.12 `template<class T> T Matrix< T >::inner_product ( const Matrix< T > & x )`

Function to compute the inner product between this matrix and matrix *x*.

4.22.3.13 `template<class T> Matrix< T > & Matrix< T >::cofactor ( const Matrix< T > & M )`

Function to convert this matrix to a cofactor matrix of the given matrix *M*.

4.22.3.14 `template<class T> Matrix< T > Matrix< T >::operator+ ( const Matrix< T > & M )`

Operator to add this matrix and matrix *M* and return the new matrix result.

4.22.3.15 `template<class T> Matrix< T > Matrix< T >::operator- ( const Matrix< T > & M )`

Operator to subtract this matrix and matrix *M* and return the new matrix result.

4.22.3.16 `template<class T> Matrix< T > Matrix< T >::operator* ( const T a )`

Operator to multiply this matrix by a scalar *T* return the new matrix result.

4.22.3.17 `template<class T> Matrix< T > Matrix< T >::operator/ ( const T a )`

Operator to divide this matrix by a scalar *T* and return the new matrix result.

**4.22.3.18** `template<class T> Matrix< T > Matrix< T >::operator* ( const Matrix< T > & M )`

Operator to multiply this matrix and matrix M and return the new matrix result.

**4.22.3.19** `template<class T> Matrix< T > & Matrix< T >::transpose ( const Matrix< T > & M )`

Function to convert this matrix to the transpose of the given matrix M.

**4.22.3.20** `template<class T> Matrix< T > & Matrix< T >::transpose_multiply ( const Matrix< T > & MT, const Matrix< T > & v )`

Function to convert this matrix into the result of the given matrix M transposed and multiplied by the other given matrix v.

**4.22.3.21** `template<class T> Matrix< T > & Matrix< T >::adjoint ( const Matrix< T > & M )`

Function to convert this matrix to the adjoint of the given matrix.

**4.22.3.22** `template<class T> Matrix< T > & Matrix< T >::inverse ( const Matrix< T > & M )`

Function to convert this matrix to the inverse of the given matrix.

**4.22.3.23** `template<class T> void Matrix< T >::Display ( const std::string Name )`

Function to display the contents of this matrix given a Name for the matrix.

**4.22.3.24** `template<class T> Matrix< T > & Matrix< T >::tridiagonalSolve ( const Matrix< T > & A, const Matrix< T > & b )`

Function to solve  $Ax=b$  for x if A is symmetric, tridiagonal (this->x)

**4.22.3.25** `template<class T> Matrix< T > & Matrix< T >::ladshawSolve ( const Matrix< T > & A, const Matrix< T > & d )`

Function to solve  $Ax=d$  for x if A is non-symmetric, tridiagonal (this->x)

**4.22.3.26** `template<class T> Matrix< T > & Matrix< T >::tridiagonalFill ( const T A, const T B, const T C, bool Spherical )`

Function to fill in this matrix with coefficients A, B, and C to form a tridiagonal matrix.

This function fills in the diagonal elements of a square matrix with coefficient B, upper diagonal with C, and lower diagonal with A. The boolean will apply a transformation to those coefficients, if the problem happens to stem from 1-D diffusion in spherical coordinates.

**4.22.3.27** `template<class T> Matrix< T > & Matrix< T >::naturalLaplacian3D ( int m )`

Function to fill out this matrix with coefficients from a 3D Laplacian function.

This function will fill out the coefficients of the matrix with the coefficients that stem from discretizing a 3D Laplacian on a natural grid with 2nd order finite differences.



4.22.3.28 `template<class T> Matrix< T > & Matrix< T >::sphericalBCFill ( int node, const T coeff, T variable )`

Function to fill out a column matrix with spherical specific boundary conditions.

This function will fill out a column matrix with zeros at all nodes except for the node indicated. That node's value will be the product of the node id with the coeff and variable values given.

4.22.3.29 `template<class T> Matrix< T > & Matrix< T >::ConstantICFill ( const T IC )`

Function to set all values in a column matrix to a given constant.

4.22.3.30 `template<class T> Matrix< T > & Matrix< T >::SolnTransform ( const Matrix< T > & A, bool Forward )`

Function to transform the values in a column matrix from cartesian to spherical coordinates.

4.22.3.31 `template<class T> T Matrix< T >::sphericalAvg ( double radius, double dr, double bound, bool Dirichlet )`

Function to compute a spatial average of this column matrix in spherical coordinates.

This function is used to compute an average value of a variable, represented in this column matrix, by integrating over the domain of the sphere. (Assumes you have variable value at center node)

#### Parameters

|                  |  |
|------------------|--|
| <i>radius</i>    | radius of the sphere                                 |
| <i>dr</i>        | space between each node                              |
| <i>bound</i>     | value of the variable at the boundary                |
| <i>Dirichlet</i> | True if problem has a Dirichlet BC, False if Neumann |

4.22.3.32 `template<class T> T Matrix< T >::IntegralAvg ( double radius, double dr, double bound, bool Dirichlet )`

Function to compute a spatial average of this column matrix in spherical coordinates.

This function is used to compute an average value of a variable, represented in this column matrix, by integrating over the domain of the sphere. (Assumes you DO NOT have variable value at center node)

#### Parameters

|                  |  |
|------------------|--|
| <i>radius</i>    | radius of the sphere                                 |
| <i>dr</i>        | space between each node                              |
| <i>bound</i>     | value of the variable at the boundary                |
| <i>Dirichlet</i> | True if problem has a Dirichlet BC, False if Neumann |

4.22.3.33 `template<class T> T Matrix< T >::IntegralTotal ( double dr, double bound, bool Dirichlet )`

Function to compute a spatial total of this column matrix in spherical coordinates.

This function is used to compute an average value of a variable, represented in this column matrix, by integrating over the domain of the sphere. (Assumes you DO NOT have variable value at center node)

#### Parameters

|                  |  |
|------------------|--|
| <i>dr</i>        | space between each node                              |
| <i>bound</i>     | value of the variable at the boundary                |
| <i>Dirichlet</i> | True if problem has a Dirichlet BC, False if Neumann |

4.22.3.34 `template<class T> Matrix< T > & Matrix< T >::tridiagonalVectorFill ( const std::vector< T > & A, const std::vector< T > & B, const std::vector< T > & C )`

Function to fill in this matrix, in tridiagonal fashion, using the vectors of coefficients.

4.22.3.35 `template<class T> Matrix< T > & Matrix< T >::columnVectorFill ( const std::vector< T > & A )`

Function to fill in a column matrix with the values of the given vector object.

4.22.3.36 `template<class T> Matrix< T > & Matrix< T >::columnProjection ( const Matrix< T > & b, const Matrix< T > & b_old, const double dt, const double dt_old )`

Function to project a column matrix solution in time based on older state vectors.

This function is used in [finch.h](#) to form `Matrix u_star`. It uses the size of the current step and old step, `dt` and `dt_old` respectively, to form an approximation for the next state. The current state and older state of the variables are passed as `b` and `b_old` respectively.

4.22.3.37 `template<class T> Matrix< T > & Matrix< T >::dirichletBCFill ( int node, const T coeff, T variable )`

Function to fill in a column matrix with all zeros except at the given node.

Similar to `sphericalBCFill`, this function will set the values of all elements in the column matrix to zero except at the given node, where the value is set to the product of `coeff` and `variable`. This is often used to set BCs in [finch.h](#) or other related files/simulations.

4.22.3.38 `template<class T> Matrix< T > & Matrix< T >::diagonalSolve ( const Matrix< T > & D, const Matrix< T > & v )`

Function to solve the system  $Dx=v$  for  $x$  given that  $D$  is diagonal (this-> $x$ )

4.22.3.39 `template<class T> Matrix< T > & Matrix< T >::upperTriangularSolve ( const Matrix< T > & U, const Matrix< T > & v )`

Function to solve the system  $Ux=v$  for  $x$  given that  $U$  is upper Triangular (this-> $x$ )

4.22.3.40 `template<class T> Matrix< T > & Matrix< T >::lowerTriangularSolve ( const Matrix< T > & L, const Matrix< T > & v )`

Function to solve the system  $Lx=v$  for  $x$  given that  $L$  is lower Triangular (this-> $x$ )

4.22.3.41 `template<class T> Matrix< T > & Matrix< T >::upperHessenberg2Triangular ( Matrix< T > & b )`

Function to convert this square matrix to upper Triangular (assuming this is upper Hessenberg)

During this transformation, a column vector (`b`) is also being transformed to represent the BCs in a linear system. This algorithm uses Givens Rotations to efficiently convert the upper Hessenberg matrix to an upper triangular matrix.

4.22.3.42 `template<class T> Matrix< T > & Matrix< T >::lowerHessenberg2Triangular ( Matrix< T > & b )`

Function to convert this square matrix to lower Triangular (assuming this is lower Hessenberg)

During this transformation, a column vector (b) is also being transformed to represent the BCs in a linear system. This algorithm uses Givens Rotations to efficiently convert the lower Hessenberg matrix to an lower triangular matrix.

**4.22.3.43** `template<class T> Matrix< T> & Matrix< T>::upperHessenbergSolve ( const Matrix< T> & H, const Matrix< T> & v )`

Function to solve the system  $Hx=v$  for  $x$  given that  $H$  is upper Hessenberg (this->x)

**4.22.3.44** `template<class T> Matrix< T> & Matrix< T>::lowerHessenbergSolve ( const Matrix< T> & H, const Matrix< T> & v )`

Function to solve the system  $Hx=v$  for  $x$  given that  $H$  is lower Hessenberg (this->x)

**4.22.3.45** `template<class T> Matrix< T> & Matrix< T>::columnExtract ( int j, const Matrix< T> & M )`

Function to set this column matrix to the jth column of the given matrix M.

**4.22.3.46** `template<class T> Matrix< T> & Matrix< T>::rowExtract ( int i, const Matrix< T> & M )`

Function to set this row matrix to the ith row of the given matrix M.

**4.22.3.47** `template<class T> Matrix< T> & Matrix< T>::columnReplace ( int j, const Matrix< T> & v )`

Function to this matrices' jth column with the given column matrix v.

**4.22.3.48** `template<class T> Matrix< T> & Matrix< T>::rowReplace ( int i, const Matrix< T> & v )`

Function to this matrices' ith row with the given row matrix v.

**4.22.3.49** `template<class T> void Matrix< T>::rowShrink ( )`

Function to delete the last row of this matrix.

**4.22.3.50** `template<class T> void Matrix< T>::columnShrink ( )`

Function to delete the last column of this matrix.

**4.22.3.51** `template<class T> void Matrix< T>::rowExtend ( const Matrix< T> & v )`

Function to add the row matrix v to the end of this matrix.

**4.22.3.52** `template<class T> void Matrix< T>::columnExtend ( const Matrix< T> & v )`

Function to add the column matrix v to the end of this matrix.

#### 4.22.4 Member Data Documentation

4.22.4.1 `template<class T> int Matrix< T >::num_rows` [protected]

Number of rows of the matrix.

4.22.4.2 `template<class T> int Matrix< T >::num_cols` [protected]

Number of columns of the matrix.

4.22.4.3 `template<class T> std::vector<T> Matrix< T >::Data` [protected]

Storage vector for the elements of the matrix.

The documentation for this class was generated from the following file:

- [macaw.h](#)

### 4.23 Mechanism Class Reference

```
#include <shark.h>
```

#### Protected Attributes

- [MasterSpeciesList](#) \* [List](#)
- `std::vector< UnsteadyReaction >` [reactions](#)
- `std::vector< double >` [weight](#)
- `int` [species\\_index](#)

#### 4.23.1 Member Data Documentation

4.23.1.1 `MasterSpeciesList* Mechanism::List` [protected]

4.23.1.2 `std::vector<UnsteadyReaction> Mechanism::reactions` [protected]

4.23.1.3 `std::vector<double> Mechanism::weight` [protected]

4.23.1.4 `int Mechanism::species_index` [protected]

The documentation for this class was generated from the following file:

- [shark.h](#)

### 4.24 MIXED\_GAS Struct Reference

Data structure holding information necessary for computing mixed gas properties.

```
#include <egret.h>
```

## Public Attributes

- int [N](#)  
*Given: Total number of gas species.*
- bool [CheckMolefractions](#) = true  
*Given: True = Check Molefractions for errors.*
- double [total\\_pressure](#)  
*Given: Total gas pressure (kPa)*
- double [gas\\_temperature](#)  
*Given: Gas temperature (K)*
- double [velocity](#)  
*Given: Gas phase velocity (cm/s)*
- double [char\\_length](#)  
*Given: Characteristic Length (cm)*
- std::vector< double > [molefraction](#)  
*Given: Gas molefractions of each species (-)*
- double [total\\_density](#)  
*Calculated: Total gas density (g/cm<sup>3</sup>) {use RE3}.*
- double [total\\_dyn\\_vis](#)  
*Calculated: Total dynamic viscosity (g/cm/s)*
- double [kinematic\\_viscosity](#)  
*Calculated: Kinematic viscosity (cm<sup>2</sup>/s)*
- double [total\\_molecular\\_weight](#)  
*Calculated: Total molecular weight (g/mol)*
- double [total\\_specific\\_heat](#)  
*Calculated: Total specific heat (J/g/K)*
- double [Reynolds](#)  
*Calculated: Value of the Reynold's number (-)*
- [Matrix](#)< double > [binary\\_diffusion](#)  
*Calculated: Tensor matrix of binary gas diffusivities (cm<sup>2</sup>/s)*
- std::vector< [PURE\\_GAS](#) > [species\\_dat](#)  
*Vector of the pure gas info of all species.*

## 4.24.1 Detailed Description

Data structure holding information necessary for computing mixed gas properties.

C-style object holding the mixed gas information necessary for performing gas dynamic simulations. This object works in conjunction with the `calculate_variables` function and uses the kinetic theory of gases to estimate mixed gas properties.

## 4.24.2 Member Data Documentation

## 4.24.2.1 int MIXED\_GAS::N

Given: Total number of gas species.

## 4.24.2.2 bool MIXED\_GAS::CheckMolefractions = true

Given: True = Check Molefractions for errors.

**4.24.2.3 double MIXED\_GAS::total\_pressure**

Given: Total gas pressure (kPa)

**4.24.2.4 double MIXED\_GAS::gas\_temperature**

Given: Gas temperature (K)

**4.24.2.5 double MIXED\_GAS::velocity**

Given: Gas phase velocity (cm/s)

**4.24.2.6 double MIXED\_GAS::char\_length**

Given: Characteristic Length (cm)

**4.24.2.7 std::vector<double> MIXED\_GAS::molefraction**

Given: Gas molefractions of each species (-)

**4.24.2.8 double MIXED\_GAS::total\_density**

Calculated: Total gas density (g/cm<sup>3</sup>) {use RE3}.

**4.24.2.9 double MIXED\_GAS::total\_dyn\_vis**

Calculated: Total dynamic viscosity (g/cm/s)

**4.24.2.10 double MIXED\_GAS::kinematic\_viscosity**

Calculated: Kinematic viscosity (cm<sup>2</sup>/s)

**4.24.2.11 double MIXED\_GAS::total\_molecular\_weight**

Calculated: Total molecular weight (g/mol)

**4.24.2.12 double MIXED\_GAS::total\_specific\_heat**

Calculated: Total specific heat (J/g/K)

**4.24.2.13 double MIXED\_GAS::Reynolds**

Calculated: Value of the Reynold's number (-)

**4.24.2.14 Matrix<double> MIXED\_GAS::binary\_diffusion**

Calculated: Tensor matrix of binary gas diffusivities (cm<sup>2</sup>/s)

4.24.2.15 `std::vector<PURE_GAS> MIXED_GAS::species_dat`

Vector of the pure gas info of all species.

The documentation for this struct was generated from the following file:

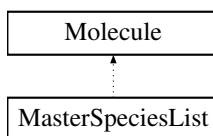
- [egret.h](#)

## 4.25 Molecule Class Reference

C++ [Molecule](#) Object built from [Atom](#) Objects (click [Molecule](#) to go to function definitions)

```
#include <mola.h>
```

Inheritance diagram for Molecule:



### Public Member Functions

- [Molecule](#) ()  
*Default Constructor (builds an empty molecule object)*
- [~Molecule](#) ()  
*Default Destructor (clears out memory)*
- [Molecule](#) (int [charge](#), double enthalpy, double entropy, double energy, bool HS, bool G, std::string [Phase](#), std::string [Name](#), std::string [Formula](#), std::string lin\_formula)  
*Construct any molecule from the available information.*
- void [Register](#) (int [charge](#), double enthalpy, double entropy, double energy, bool HS, bool G, std::string [Phase](#), std::string [Name](#), std::string [Formula](#), std::string lin\_formula)  
*Function to register this molecule from the available information.*
- void [Register](#) (std::string formula)  
*Function to register this molecule based on the given formula (if formula is in library)*
- void [setFormula](#) (std::string form)  
*Sets the formula for a molecule.*
- void [recalculateMolarWeight](#) ()  
*Forces molecule to recalculate its molar weight.*
- void [setMolarWeigth](#) (double mw)  
*Set the molar weight of species to a constant.*
- void [editCharge](#) (int c)  
*Change the ionic charge of a molecule.*
- void [editOneOxidationState](#) (int state, std::string Symbol)  
*Change oxidation state of one of the given atoms (always first match found)*
- void [editAllOxidationStates](#) (int state, std::string Symbol)  
*Change oxidation state of all of the given atoms.*
- void [calculateAvgOxiState](#) (std::string Symbol)  
*Function to calculate the average oxidation state of the atoms.*
- void [editEnthalpy](#) (double enthalpy)  
*Edit the molecules formation enthalpy (J/mol)*

- void [editEntropy](#) (double entropy)  
*Edit the molecules formation entropy (J/K/mol)*
- void [editHS](#) (double H, double S)  
*Edit both formation enthalpy and entropy.*
- void [editEnergy](#) (double energy)  
*Edit Gibb's formation energy.*
- void [removeOneAtom](#) (std::string Symbol)  
*Removes one atom of the symbol given (always the first atom found)*
- void [removeAllAtoms](#) (std::string Symbol)  
*Removes all atoms of the symbol given.*
- int [Charge](#) ()  
*Return the charge of the molecule.*
- double [MolarWeight](#) ()  
*Return the molar weight of the molecule.*
- bool [HaveHS](#) ()  
*Returns true if enthalpy and entropy are known.*
- bool [HaveEnergy](#) ()  
*Returns true if the Gibb's energy is known.*
- bool [isRegistered](#) ()  
*Returns true if the molecule has been registered.*
- double [Enthalpy](#) ()  
*Return the formation enthalpy of the molecule.*
- double [Entropy](#) ()  
*Return the formation entropy of the molecule.*
- double [Energy](#) ()  
*Return the Gibb's formation energy of the molecule.*
- std::string [MoleculeName](#) ()  
*Return the common name of the molecule.*
- std::string [MolecularFormula](#) ()  
*Return the molecular formula of the molecule.*
- std::string [MoleculePhase](#) ()  
*Return the phase of the molecule.*
- void [DisplayInfo](#) ()  
*Function to display molecule information.*

## Protected Attributes

- int [charge](#)  
*Ionic charge of the molecule - specified.*
- double [molar\\_weight](#)  
*Molar weight of the molecule (g/mol) - determined from atoms or specified.*
- double [formation\\_enthalpy](#)  
*Enthalpy of formation of the molecule (J/mol) - constant.*
- double [formation\\_entropy](#)  
*Entropy of formation of the molecule (J/K/mol) - constant.*
- double [formation\\_energy](#)  
*Gibb's energy of formation (J/mol) - given.*
- std::string [Phase](#)  
*Phase of the molecule (i.e. Solid, Liquid, Aqueous, Gas...)*
- std::vector< [Atom](#) > [atoms](#)  
*Atoms which make up the molecule - based on Formula.*



### Private Attributes

- std::string [Name](#)  
*Name of the [Molecule](#) - Common Name (i.e. H2O = Water)*
- std::string [Formula](#)  
*Formula for the molecule - specified (i.e. H2O)*
- bool [haveG](#)  
*True = given Gibb's energy of formation.*
- bool [haveHS](#)  
*True = give enthalpy and entropy of formation.*
- bool [registered](#)  
*True = the object was registered.*

### 4.25.1 Detailed Description

C++ [Molecule](#) Object built from [Atom](#) Objects (click [Molecule](#) to go to function definitions)

C++ Class Object that stores information and certain operations associated with molecules. Registered molecules are built up from their respective atoms so that the molecule can keep track of information such as molecular weight and oxidation states. Primarily, this object is used in conjunction with [shark.h](#) to formulate the system of equations necessary for solving speciation type problems in aqueous systems. However, this object is generalized enough to be of use in RedOx calculations, reaction formulation, and molecular transformations.

All information for a molecule should be initialized prior to performing operations with or on the object. There are several molecules already defined for construction by the formulas listed at the top of this section.

### 4.25.2 Constructor & Destructor Documentation

#### 4.25.2.1 `Molecule::Molecule ( )`

Default Constructor (builds an empty molecule object)

#### 4.25.2.2 `Molecule::~~Molecule ( )`

Default Destructor (clears out memory)

#### 4.25.2.3 `Molecule::Molecule ( int charge, double enthalpy, double entropy, double energy, bool HS, bool G, std::string Phase, std::string Name, std::string Formula, std::string lin_formula )`

Construct any molecule from the available information.

This constructor will build a user defined custom molecule.

#### Parameters

|                    |   |
|--------------------|---|
| <i>charge</i>      | the ionic charge of the molecule  |
| <i>enthalpy</i>    | the standard formation enthalpy of the molecule (J/mol)                           |
| <i>entropy</i>     | the standard formation entropy of the molecule (J/K/mol)                          |
| <i>energy</i>      | the standard Gibb's Free Energy of formation of the molecule (J/mol)              |
| <i>HS</i>          | boolean to be set to true if enthalpy and entropy were given                      |
| <i>G</i>           | boolean to be set to true if the energy was given                                 |
| <i>Phase</i>       | string denoting molecule's phase (i.e., Liquid, Aqueous, Gas, Solid)              |
| <i>Name</i>        | string denoting the common name of the molecule (i.e., H2O -> Water)              |
| <i>Formula</i>     | string denoting the formula by which the molecule is referenced (i.e., Cl - (aq)) |
| <i>lin_formula</i> | string denoting all the atoms in the molecule (i.e., UO2(OH)2 -> UO4H2)           |

### 4.25.3 Member Function Documentation

**4.25.3.1** void Molecule::Register ( int *charge*, double *enthalpy*, double *entropy*, double *energy*, bool *HS*, bool *G*, std::string *Phase*, std::string *Name*, std::string *Formula*, std::string *lin\_formula* )

Function to register this molecule from the available information.

This function will build a user defined custom molecule.

#### Parameters

|                    |  |
|--------------------|--|
| <i>charge</i>      | the ionic charge of the molecule   |
| <i>enthalpy</i>    | the standard formation enthalpy of the molecule (J/mol)                          |
| <i>entropy</i>     | the standard formation entropy of the molecule (J/K/mol)                         |
| <i>energy</i>      | the standard Gibb's Free Energy of formation of the molecule (J/mol)             |
| <i>HS</i>          | boolean to be set to true if enthalpy and entropy were given                     |
| <i>G</i>           | boolean to be set to true if the energy was given                                |
| <i>Phase</i>       | string denoting molecule's phase (i.e., Liquid, Aqueous, Gas, Solid)             |
| <i>Name</i>        | string denoting the common name of the molecule (i.e., H2O -> Water)             |
| <i>Formula</i>     | string denoting the formula by which the molecule is referened (i.e., Cl - (aq)) |
| <i>lin_formula</i> | string denoting all the atoms in the molecule (i.e., UO2(OH)2 -> UO4H2)          |

**4.25.3.2** void Molecule::Register ( std::string *formula* )

Function to register this molecule based on the given formula (if formula is in library)

This function will create this molecule object from the given formula, but only if that formula is already registered in the library. See the top of this class section for a list of all currently registered formulas.

#### Note

The formula is checked against a known set of molecules inside of the registration function. If the formula is unknown, an error will print to the screen. Unknown molecules should be registered using the full registration function from above. The library can only be added to by a going in and editing the source code of the mola.cpp file. However, this is a relatively simple task.

**4.25.3.3** void Molecule::setFormula ( std::string *form* )

Sets the formula for a molecule.

**4.25.3.4** void Molecule::recalculateMolarWeight ( )

Forces molecule to recalculate its molar weight.

**4.25.3.5** void Molecule::setMolarWeigth ( double *mw* )

Set the molar weight of species to a constant.

**4.25.3.6** void Molecule::editCharge ( int *c* )

Change the ionic charge of a molecule.

**4.25.3.7 void Molecule::editOneOxidationState ( int *state*, std::string *Symbol* )**

Change oxidation state of one of the given atoms (always first match found)

This function will search the list of Atoms that make up the [Molecule](#) for the given atomic Symbol. It will change the oxidation state of the first found matching atom with the given state.

**4.25.3.8 void Molecule::editAllOxidationStates ( int *state*, std::string *Symbol* )**

Change oxidation state of all of the given atoms.

This function will search the list of Atoms that make up the [Molecule](#) for the given atomic Symbol. It will change the oxidation state of all found matching atoms with the given state.

**4.25.3.9 void Molecule::calculateAvgOxiState ( std::string *Symbol* )**

Function to calculate the average oxidation state of the atoms.

This function search the atoms in the molecule for the matching atomic Symbol. It then looks at all oxidation states of that atom in the molecule and then sets all the oxidation states of that atom to the average value calculated.

**4.25.3.10 void Molecule::editEnthalpy ( double *enthalpy* )**

Edit the molecules formation enthalpy (J/mol)

**4.25.3.11 void Molecule::editEntropy ( double *entropy* )**

Edit the molecules formation entropy (J/K/mol)

**4.25.3.12 void Molecule::editHS ( double *H*, double *S* )**

Edit both formation enthalpy and entropy.

This function will change or set the values for formation enthalpy (J/mol) and formation entropy (J/K/mol) based on the given values.

**Parameters**

|          |                             |
|----------|-----------------------------|
| <i>H</i> | formation enthalpy (J/mol)  |
| <i>S</i> | formation entropy (J/K/mol) |

**4.25.3.13 void Molecule::editEnergy ( double *energy* )**

Edit Gibb's formation energy.

**4.25.3.14 void Molecule::removeOneAtom ( std::string *Symbol* )**

Removes one atom of the symbol given (always the first atom found)

**4.25.3.15 void Molecule::removeAllAtoms ( std::string *Symbol* )**

Removes all atoms of the symbol given.

4.25.3.16 `int Molecule::Charge ( )`

Return the charge of the molecule.

4.25.3.17 `double Molecule::MolarWeight ( )`

Return the molar weight of the molecule.

4.25.3.18 `bool Molecule::HaveHS ( )`

Returns true if enthalpy and entropy are known.

4.25.3.19 `bool Molecule::HaveEnergy ( )`

Returns true if the Gibb's energy is known.

4.25.3.20 `bool Molecule::isRegistered ( )`

Returns true if the molecule has been registered.

4.25.3.21 `double Molecule::Enthalpy ( )`

Return the formation enthalpy of the molecule.

4.25.3.22 `double Molecule::Entropy ( )`

Return the formation entropy of the molecule.

4.25.3.23 `double Molecule::Energy ( )`

Return the Gibb's formation energy of the molecule.

4.25.3.24 `std::string Molecule::MoleculeName ( )`

Return the common name of the molecule.

4.25.3.25 `std::string Molecule::MolecularFormula ( )`

Return the molecular formula of the molecule.

4.25.3.26 `std::string Molecule::MoleculePhase ( )`

Return the phase of the molecule.

4.25.3.27 `void Molecule::DisplayInfo ( )`

Function to display molecule information.

#### 4.25.4 Member Data Documentation

##### 4.25.4.1 `int Molecule::charge` `[protected]`

Ionic charge of the molecule - specified.

##### 4.25.4.2 `double Molecule::molar_weight` `[protected]`

Molar weight of the molecule (g/mol) - determined from atoms or specified.

##### 4.25.4.3 `double Molecule::formation_enthalpy` `[protected]`

Enthalpy of formation of the molecule (J/mol) - constant.

##### 4.25.4.4 `double Molecule::formation_entropy` `[protected]`

Entropy of formation of the molecule (J/K/mol) - constant.

##### 4.25.4.5 `double Molecule::formation_energy` `[protected]`

Gibb's energy of formation (J/mol) - given.

##### 4.25.4.6 `std::string Molecule::Phase` `[protected]`

Phase of the molecule (i.e. Solid, Liquid, Aqueous, Gas...)

##### 4.25.4.7 `std::vector<Atom> Molecule::atoms` `[protected]`

Atoms which make up the molecule - based on Formula.

##### 4.25.4.8 `std::string Molecule::Name` `[private]`

Name of the [Molecule](#) - Common Name (i.e. H2O = Water)

##### 4.25.4.9 `std::string Molecule::Formula` `[private]`

Formula for the molecule - specified (i.e. H2O)

##### 4.25.4.10 `bool Molecule::haveG` `[private]`

True = given Gibb's energy of formation.

##### 4.25.4.11 `bool Molecule::haveHS` `[private]`

True = give enthalpy and entropy of formation.

#### 4.25.4.12 bool Molecule::registered [private]

True = the object was registered.

The documentation for this class was generated from the following file:

- [mola.h](#)

## 4.26 MONKFISH\_DATA Struct Reference

```
#include <monkfish.h>
```

### Public Attributes

- unsigned long int [total\\_steps](#) = 0
- double [time\\_old](#) = 0.0
- double [time](#) = 0.0
- bool [Print2File](#) = true
- bool [Print2Console](#) = true
- bool [DirichletBC](#) = true
- bool [NonLinear](#) = false
- bool [haveMinMax](#) = false
- bool [MultiScale](#) = true
- int [level](#) = 2
- double [t\\_counter](#) = 0.0
- double [t\\_print](#)
- int [NumComp](#)
- double [end\\_time](#)
- double [total\\_sorption\\_old](#)
- double [total\\_sorption](#)
- double [single\\_fiber\\_density](#)
- double [avg\\_fiber\\_density](#)
- double [max\\_fiber\\_density](#)
- double [min\\_fiber\\_density](#)
- double [max\\_porosity](#)
- double [min\\_porosity](#)
- double [domain\\_diameter](#)
- FILE \* [Output](#)
- double(\* [eval\\_eps](#) )(int i, int l, const void \*[user\\_data](#))
- double(\* [eval\\_rho](#) )(int i, int l, const void \*[user\\_data](#))
- double(\* [eval\\_Dex](#) )(int i, int l, const void \*[user\\_data](#))
- double(\* [eval\\_ads](#) )(int i, int l, const void \*[user\\_data](#))
- double(\* [eval\\_Ret](#) )(int i, int l, const void \*[user\\_data](#))
- double(\* [eval\\_Cex](#) )(int i, const void \*[user\\_data](#))
- double(\* [eval\\_kf](#) )(int i, const void \*[user\\_data](#))
- const void \* [user\\_data](#)
- std::vector< [FINCH\\_DATA](#) > [finch\\_dat](#)
- std::vector< [MONKFISH\\_PARAM](#) > [param\\_dat](#)
- std::vector< [DOGFISH\\_DATA](#) > [dog\\_dat](#)

### 4.26.1 Member Data Documentation

- 4.26.1.1 unsigned long int MONKFISH\_DATA::total\_steps = 0
- 4.26.1.2 double MONKFISH\_DATA::time\_old = 0.0
- 4.26.1.3 double MONKFISH\_DATA::time = 0.0
- 4.26.1.4 bool MONKFISH\_DATA::Print2File = true
- 4.26.1.5 bool MONKFISH\_DATA::Print2Console = true
- 4.26.1.6 bool MONKFISH\_DATA::DirichletBC = true
- 4.26.1.7 bool MONKFISH\_DATA::NonLinear = false
- 4.26.1.8 bool MONKFISH\_DATA::haveMinMax = false
- 4.26.1.9 bool MONKFISH\_DATA::MultiScale = true
- 4.26.1.10 int MONKFISH\_DATA::level = 2
- 4.26.1.11 double MONKFISH\_DATA::t\_counter = 0.0
- 4.26.1.12 double MONKFISH\_DATA::t\_print
- 4.26.1.13 int MONKFISH\_DATA::NumComp
- 4.26.1.14 double MONKFISH\_DATA::end\_time
- 4.26.1.15 double MONKFISH\_DATA::total\_sorption\_old
- 4.26.1.16 double MONKFISH\_DATA::total\_sorption
- 4.26.1.17 double MONKFISH\_DATA::single\_fiber\_density
- 4.26.1.18 double MONKFISH\_DATA::avg\_fiber\_density
- 4.26.1.19 double MONKFISH\_DATA::max\_fiber\_density
- 4.26.1.20 double MONKFISH\_DATA::min\_fiber\_density
- 4.26.1.21 double MONKFISH\_DATA::max\_porosity
- 4.26.1.22 double MONKFISH\_DATA::min\_porosity
- 4.26.1.23 double MONKFISH\_DATA::domain\_diameter
- 4.26.1.24 FILE\* MONKFISH\_DATA::Output
- 4.26.1.25 double(\* MONKFISH\_DATA::eval\_eps)(int i, int l, const void \*user\_data)
- 4.26.1.26 double(\* MONKFISH\_DATA::eval\_rho)(int i, int l, const void \*user\_data)
- 4.26.1.27 double(\* MONKFISH\_DATA::eval\_Dex)(int i, int l, const void \*user\_data)

- 4.26.1.28 double(\* MONKFISH\_DATA::eval\_ads)(int i, int l, const void \*user\_data)
- 4.26.1.29 double(\* MONKFISH\_DATA::eval\_Ret)(int i, int l, const void \*user\_data)
- 4.26.1.30 double(\* MONKFISH\_DATA::eval\_Cex)(int i, const void \*user\_data)
- 4.26.1.31 double(\* MONKFISH\_DATA::eval\_kf)(int i, const void \*user\_data)
- 4.26.1.32 const void\* MONKFISH\_DATA::user\_data
- 4.26.1.33 std::vector<FINCH\_DATA> MONKFISH\_DATA::finch\_dat
- 4.26.1.34 std::vector<MONKFISH\_PARAM> MONKFISH\_DATA::param\_dat
- 4.26.1.35 std::vector<DOGFISH\_DATA> MONKFISH\_DATA::dog\_dat

The documentation for this struct was generated from the following file:

- [monkfish.h](#)

## 4.27 MONKFISH\_PARAM Struct Reference

```
#include <monkfish.h>
```

### Public Attributes

- double [interparticle\\_diffusion](#)
- double [exterior\\_concentration](#)
- double [exterior\\_transfer\\_coeff](#)
- double [sorbed\\_molefraction](#)
- double [initial\\_sorption](#)
- double [sorption\\_bc](#)
- double [intraparticle\\_diffusion](#)
- double [film\\_transfer\\_coeff](#)
- [Matrix](#)< double > [avg\\_sorption](#)
- [Matrix](#)< double > [avg\\_sorption\\_old](#)
- [Molecule species](#)

### 4.27.1 Member Data Documentation

- 4.27.1.1 double MONKFISH\_PARAM::interparticle\_diffusion
- 4.27.1.2 double MONKFISH\_PARAM::exterior\_concentration
- 4.27.1.3 double MONKFISH\_PARAM::exterior\_transfer\_coeff
- 4.27.1.4 double MONKFISH\_PARAM::sorbed\_molefraction
- 4.27.1.5 double MONKFISH\_PARAM::initial\_sorption
- 4.27.1.6 double MONKFISH\_PARAM::sorption\_bc
- 4.27.1.7 double MONKFISH\_PARAM::intraparticle\_diffusion



4.27.1.8 double MONKFISH\_PARAM::film\_transfer\_coeff

4.27.1.9 Matrix<double> MONKFISH\_PARAM::avg\_sorption

4.27.1.10 Matrix<double> MONKFISH\_PARAM::avg\_sorption\_old

4.27.1.11 Molecule MONKFISH\_PARAM::species

The documentation for this struct was generated from the following file:

- [monkfish.h](#)

## 4.28 mSPD\_DATA Struct Reference

MSPD Data Structure.

```
#include <magpie.h>
```

### Public Attributes

- double [s](#)  
*Area shape factor.*
- double [v](#)  
*van der Waals Volume (cm<sup>3</sup>/mol)*
- double [eMax](#)  
*Maximum lateral interaction energy (J/mol)*
- std::vector< double > [eta](#)  
*Binary interaction parameter matrix (i,j)*
- double [gama](#)  
*Activity coefficient calculated from mSPD.*

### 4.28.1 Detailed Description

MSPD Data Structure.

C-Style object holding all parameter information associated with the Modified Spreading Pressure Dependent (SPD) activity model. Each species in the gas phase will have one of these objects.

### 4.28.2 Member Data Documentation

4.28.2.1 double mSPD\_DATA::s

Area shape factor.

4.28.2.2 double mSPD\_DATA::v

van der Waals Volume (cm<sup>3</sup>/mol)

4.28.2.3 double mSPD\_DATA::eMax

Maximum lateral interaction energy (J/mol)

#### 4.28.2.4 `std::vector<double> mSPD_DATA::eta`

Binary interaction parameter matrix (i,j)

#### 4.28.2.5 `double mSPD_DATA::gama`

Activity coefficient calculated from mSPD.

The documentation for this struct was generated from the following file:

- [magpie.h](#)

## 4.29 NUM\_JAC\_DATA Struct Reference

Data structure to form a numerical jacobian matrix with finite differences.

```
#include <lark.h>
```

### Public Attributes

- `double eps = sqrt(DBL_EPSILON)`  
*Perturbation value.*
- `Matrix< double > Fx`  
*Vector of function evaluations at x.*
- `Matrix< double > Fxp`  
*Vector of function evaluations at x+eps.*
- `Matrix< double > dxj`  
*Vector of perturbed x values.*

### 4.29.1 Detailed Description

Data structure to form a numerical jacobian matrix with finite differences.

C-style object to be used in conjunction with the Numerical Jacobian algorithm. This algorithm will used double-precision finite-differences to formulate an approximate Jacobian matrix at the given variable state for the given residual/non-linear function.

### 4.29.2 Member Data Documentation

#### 4.29.2.1 `double NUM_JAC_DATA::eps = sqrt(DBL_EPSILON)`

Perturbation value.

#### 4.29.2.2 `Matrix<double> NUM_JAC_DATA::Fx`

Vector of function evaluations at x.

#### 4.29.2.3 `Matrix<double> NUM_JAC_DATA::Fxp`

Vector of function evaluations at x+eps.

## 4.29.2.4 Matrix&lt;double&gt; NUM\_JAC\_DATA::dxj

Vector of perturbed x values.

The documentation for this struct was generated from the following file:

- [lark.h](#)

## 4.30 OPTRANS\_DATA Struct Reference

Data structure for implementation of linear operator transposition.

```
#include <lark.h>
```

### Public Attributes

- Matrix< double > **li**  
The ith column vector of the identity operator.
- Matrix< double > **Ai**  
The ith column vector of the user's linear operator.

### 4.30.1 Detailed Description

Data structure for implementation of linear operator transposition.

C-style object used in conjunction with the Operator Transpose algorithm to form an action of  $A^T \cdot r$  when A is only available as a linear operator and not a matrix. This is a sub-routine required by GCR and GMRESR to stabilize the outer iterations.

### 4.30.2 Member Data Documentation

## 4.30.2.1 Matrix&lt;double&gt; OPTRANS\_DATA::li

The ith column vector of the identity operator.

## 4.30.2.2 Matrix&lt;double&gt; OPTRANS\_DATA::Ai

The ith column vector of the user's linear operator.

The documentation for this struct was generated from the following file:

- [lark.h](#)

## 4.31 PCG\_DATA Struct Reference

Data structure for implementation of the PCG algorithms for symmetric linear systems.

```
#include <lark.h>
```

## Public Attributes

- int `maxit` = 0  
*Maximum allowable iterations - default = min(vector\_size,1000)*
- int `iter` = 0  
*Actual number of iterations taken.*
- double `alpha`  
*Step size for new solution.*
- double `beta`  
*Step size for new search direction.*
- double `tol_rel` = 1e-6  
*Relative tolerance for convergence - default = 1e-6.*
- double `tol_abs` = 1e-6  
*Absolution tolerance for convergence - default = 1e-6.*
- double `res`  
*Absolute residual norm.*
- double `relres`  
*Relative residual norm.*
- double `relres_base`  
*Initial residual norm.*
- double `bestres`  
*Best found residual norm.*
- bool `Output` = true  
*True = print messages to console.*
- `Matrix< double > x`  
*Current solution to the linear system.*
- `Matrix< double > bestx`  
*Best found solution to the linear system.*
- `Matrix< double > r`  
*Residual vector for the linear system.*
- `Matrix< double > r_old`  
*Previous residual vector.*
- `Matrix< double > z`  
*Preconditioned residual vector (result of precon function)*
- `Matrix< double > z_old`  
*Previous preconditioned residual vector.*
- `Matrix< double > p`  
*Search direction.*
- `Matrix< double > Ap`  
*Result of matrix-vector multiplication.*

### 4.31.1 Detailed Description

Data structure for implementation of the PCG algorithms for symmetric linear systems.

C-style object used in conjunction with the Preconditioned Conjugate Gradient (PCG) algorithm to iteratively solve a symmetric linear system of equations. This algorithm is optimal if your linear system is symmetric, but will not work at all if your system is asymmetric. For asymmetric systems, use one of the other linear methods.

### 4.31.2 Member Data Documentation

#### 4.31.2.1 int PCG\_DATA::maxit = 0

Maximum allowable iterations - default = min(vector\_size,1000)

#### 4.31.2.2 int PCG\_DATA::iter = 0

Actual number of iterations taken.

#### 4.31.2.3 double PCG\_DATA::alpha

Step size for new solution.

#### 4.31.2.4 double PCG\_DATA::beta

Step size for new search direction.

#### 4.31.2.5 double PCG\_DATA::tol\_rel = 1e-6

Relative tolerance for convergence - default = 1e-6.

#### 4.31.2.6 double PCG\_DATA::tol\_abs = 1e-6

Absolution tolerance for convergence - default = 1e-6.

#### 4.31.2.7 double PCG\_DATA::res

Absolute residual norm.

#### 4.31.2.8 double PCG\_DATA::relres

Relative residual norm.

#### 4.31.2.9 double PCG\_DATA::relres\_base

Initial residual norm.

#### 4.31.2.10 double PCG\_DATA::bestres

Best found residual norm.

#### 4.31.2.11 bool PCG\_DATA::Output = true

True = print messages to console.

#### 4.31.2.12 Matrix<double> PCG\_DATA::x

Current solution to the linear system.

**4.31.2.13** `Matrix<double> PCG_DATA::bestx`

Best found solution to the linear system.

**4.31.2.14** `Matrix<double> PCG_DATA::r`

Residual vector for the linear system.

**4.31.2.15** `Matrix<double> PCG_DATA::r_old`

Previous residual vector.

**4.31.2.16** `Matrix<double> PCG_DATA::z`

Preconditioned residual vector (result of precon function)

**4.31.2.17** `Matrix<double> PCG_DATA::z_old`

Previous preconditioned residual vector.

**4.31.2.18** `Matrix<double> PCG_DATA::p`

Search direction.

**4.31.2.19** `Matrix<double> PCG_DATA::Ap`

Result of matrix-vector multiplication.

The documentation for this struct was generated from the following file:

- [lark.h](#)

## 4.32 PeriodicTable Class Reference

Class object that store a digital copy of all [Atom](#) objects.

```
#include <eel.h>
```

### Public Member Functions

- [PeriodicTable](#) ()  
*Default Constructor - Build Perodic Table.*
- [~PeriodicTable](#) ()  
*Default Destructor - Destroy the table.*
- [PeriodicTable](#) (int \*n, int N)  
*Construct a partial table from a list of atomic numbers.*
- [PeriodicTable](#) (std::vector< std::string > &Symbol)  
*Construct a partial table from a vector of atom symbols.*
- [PeriodicTable](#) (std::vector< int > &n)  
*Construct a partial table from a vector of atomic numbers.*
- void [DisplayTable](#) ()  
*Displays the periodic table via symbols.*

## Protected Attributes

- `std::vector< Atom > Table`  
*Storage vector for all atoms in the table.*

## Private Attributes

- `int number_elements`  
*Number of atom objects being stored.*

### 4.32.1 Detailed Description

Class object that store a digital copy of all [Atom](#) objects.

C++ class object to hold digitally registered [Atom](#) objects. All registered atoms (Hydrogen to Ununoctium) are stored as in a vector. Currently, this object is unused, but could be modified to be explorable and used as a constant referece for all atoms in the table.

### 4.32.2 Constructor & Destructor Documentation

#### 4.32.2.1 PeriodicTable::PeriodicTable ( )

Default Constructor - Build Periodic Table.

#### 4.32.2.2 PeriodicTable::~~PeriodicTable ( )

Default Destructor - Destroy the table.

#### 4.32.2.3 PeriodicTable::PeriodicTable ( int \* *n*, int *N* )

Construct a partial table from a list of atomic numbers.

#### 4.32.2.4 PeriodicTable::PeriodicTable ( std::vector< std::string > & *Symbol* )

Construct a partial table from a vector of atom symbols.

#### 4.32.2.5 PeriodicTable::PeriodicTable ( std::vector< int > & *n* )

Construct a partial table from a vector of atomic numbers.

### 4.32.3 Member Function Documentation

#### 4.32.3.1 void PeriodicTable::DisplayTable ( )

Displays the periodic table via symbols.

### 4.32.4 Member Data Documentation

#### 4.32.4.1 std::vector<Atom> PeriodicTable::Table [protected]

Storage vector for all atoms in the table.

#### 4.32.4.2 int PeriodicTable::number\_elements [private]

Number of atom objects being stored.

The documentation for this class was generated from the following file:

- [eel.h](#)

### 4.33 PICARD\_DATA Struct Reference

Data structure for the implementation of a Picard or Fixed-Point iteration for non-linear systems.

```
#include <lark.h>
```

#### Public Attributes

- int [maxit](#) = 0  
*Maximum allowable iterations - default = min(3\*vec\_size,1000)*
- int [iter](#) = 0  
*Actual number of iterations.*
- double [tol\\_rel](#) = 1e-6  
*Relative tolerance for convergence - default = 1e-6.*
- double [tol\\_abs](#) = 1e-6  
*Absolution tolerance for convergence - default = 1e-6.*
- double [res](#)  
*Residual norm of the iterate.*
- double [relres](#)  
*Relative residual norm of the iterate.*
- double [relres\\_base](#)  
*Initial residual norm.*
- double [bestres](#)  
*Best found residual norm.*
- bool [Output](#) = true  
*True = print messages to console.*
- [Matrix](#)< double > [x0](#)  
*Previous iterate solution vector.*
- [Matrix](#)< double > [bestx](#)  
*Best found solution vector.*
- [Matrix](#)< double > [r](#)  
*Residual of the non-linear system.*

#### 4.33.1 Detailed Description

Data structure for the implementation of a Picard or Fixed-Point iteration for non-linear systems.

C-style object used in conjunction with the Picard algorithm for solving a non-linear system of equations. This is an extraordinarily simple iterative method by which a weak or loose form of the non-linear system is solved based on an initial guess. User must supplied a residual function for the non-linear system and a function representing the weak solution. Generally, this method is less efficient than Newton methods, but is significantly cheaper.



### 4.33.2 Member Data Documentation

#### 4.33.2.1 int PICARD\_DATA::maxit = 0

Maximum allowable iterations - default =  $\min(3 \times \text{vec\_size}, 1000)$

#### 4.33.2.2 int PICARD\_DATA::iter = 0

Actual number of iterations.

#### 4.33.2.3 double PICARD\_DATA::tol\_rel = 1e-6

Relative tolerance for convergence - default = 1e-6.

#### 4.33.2.4 double PICARD\_DATA::tol\_abs = 1e-6

Absolution tolerance for convergence - default = 1e-6.

#### 4.33.2.5 double PICARD\_DATA::res

Residual norm of the iterate.

#### 4.33.2.6 double PICARD\_DATA::relres

Relative residual norm of the iterate.

#### 4.33.2.7 double PICARD\_DATA::relres\_base

Initial residual norm.

#### 4.33.2.8 double PICARD\_DATA::bestres

Best found residual norm.

#### 4.33.2.9 bool PICARD\_DATA::Output = true

True = print messages to console.

#### 4.33.2.10 Matrix<double> PICARD\_DATA::x0

Previous iterate solution vector.

#### 4.33.2.11 Matrix<double> PICARD\_DATA::bestx

Best found solution vector.

#### 4.33.2.12 Matrix<double> PICARD\_DATA::r

Residual of the non-linear system.

The documentation for this struct was generated from the following file:

- [lark.h](#)

### 4.34 PJFNK\_DATA Struct Reference

Data structure for the implementation of the PJFNK algorithm for non-linear systems.

```
#include <lark.h>
```

#### Public Attributes

- int [nl\\_iter](#) = 0  
*Number of non-linear iterations.*
- int [l\\_iter](#) = 0  
*Number of linear iterations.*
- int [nl\\_maxit](#) = 0  
*Maximum allowable non-linear steps.*
- int [linear\\_solver](#) = -1  
*Flag to denote which linear solver to use - default = PJFNK Chooses.*
- double [nl\\_tol\\_abs](#) = 1e-6  
*Absolute Convergence tolerance for non-linear system - default = 1e-6.*
- double [nl\\_tol\\_rel](#) = 1e-6  
*Relative Convergence tol for the non-linear system - default = 1e-6.*
- double [lin\\_tol\\_rel](#) = 1e-6  
*Relative tolerance of the linear solver - default = 1e-6.*
- double [lin\\_tol\\_abs](#) = 1e-6  
*Absolute tolerance of the linear solver - default = 1e-6.*
- double [nl\\_res](#)  
*Absolute redidual norm for the non-linear system.*
- double [nl\\_relres](#)  
*Relative residual for the non-linear system.*
- double [nl\\_res\\_base](#)  
*Initial residual norm for the non-linear system.*
- double [nl\\_bestres](#)  
*Best found residual norm.*
- double [eps](#) = sqrt([DBL\\_EPSILON](#))  
*Value of epsilon used jacvec - default = sqrt(DBL\_EPSILON)*
- bool [NL\\_Output](#) = true  
*True = print PJFNK messages to console.*
- bool [L\\_Output](#) = false  
*True = print Linear messages to console.*
- bool [LineSearch](#) = false  
*True = use Backtracking Linesearch for global convergence.*
- bool [Bounce](#) = false  
*True = allow Linesearch to go outside local well, False = Strict local convergence.*
- [Matrix](#)< double > [F](#)

- Stored fuction evaluation at x (also the residual)*

  - [Matrix< double > Fv](#)
- Stored function evaluation at  $x+eps*v$ .*

  - [Matrix< double > v](#)
- Stored vector of  $x+eps*v$ .*

  - [Matrix< double > x](#)
- Current solution vector for the non-linear system.*

  - [Matrix< double > bestx](#)
- Best found solution vector to the non-linear system.*

  - [GMRESLP\\_DATA gmreslp\\_dat](#)
- Data structure for the GMRESLP method.*

  - [PCG\\_DATA pcg\\_dat](#)
- Data structure for the PCG method.*

  - [BiCGSTAB\\_DATA bicgstab\\_dat](#)
- Data structure for the BiCGSTAB method.*

  - [CGS\\_DATA cgs\\_dat](#)
- Data structure for the CGS method.*

  - [GMRESRP\\_DATA gmresrp\\_dat](#)
- Data structure for the GMRESRP method.*

  - [GCR\\_DATA gcr\\_dat](#)
- Data structure for the GCR method.*

  - [GMRESR\\_DATA gmresr\\_dat](#)
- Data structure for the GMRESR method.*

  - [BACKTRACK\\_DATA backtrack\\_dat](#)
- Data structure for the Backtracking Linesearch algorithm.*

  - `const void * res\_data`
- Data structure pointer for user's residual data.*

  - `const void * precon\_data`
- Data structure pointer for user's preconditioning data.*

  - `int(* funeval )(const Matrix< double > &x, Matrix< double > &F, const void *res\_data)`
- Function pointer for the user's function  $F(x)$  using there data.*

  - `int(* precon )(const Matrix< double > &r, Matrix< double > &p, const void *precon\_data)`
- Function pointer for the user's preconditioning function for the linear system.*

#### 4.34.1 Detailed Description

Data structure for the implementation of the PJFNK algorithm for non-linear systems.

C-style object to be used in conjunction with the Preconditioned Jacobian-Free Newton-Krylov (PJFNK) method for solving a non-linear system of equations. You can use any of the Krylov methods listed in the `krylov_method` enum to solve the linear sub-problem. When FOM is specified as the Krylov method, this algorithm becomes equivalent to an exact Newton method. If no Krylov method is specified, then the algorithm will try to pick a method based on the problem size and availability of preconditioning.

#### 4.34.2 Member Data Documentation

##### 4.34.2.1 `int PJFNK_DATA::nl_iter = 0`

Number of non-linear iterations.

4.34.2.2 `int PJFNK_DATA::l_iter = 0`

Number of linear iterations.

4.34.2.3 `int PJFNK_DATA::nl_maxit = 0`

Maximum allowable non-linear steps.

4.34.2.4 `int PJFNK_DATA::linear_solver = -1`

Flag to denote which linear solver to use - default = PJFNK Chooses.

4.34.2.5 `double PJFNK_DATA::nl_tol_abs = 1e-6`

Absolute Convergence tolerance for non-linear system - default = 1e-6.

4.34.2.6 `double PJFNK_DATA::nl_tol_rel = 1e-6`

Relative Convergence tol for the non-linear system - default = 1e-6.

4.34.2.7 `double PJFNK_DATA::lin_tol_rel = 1e-6`

Relative tolerance of the linear solver - default = 1e-6.

4.34.2.8 `double PJFNK_DATA::lin_tol_abs = 1e-6`

Absolute tolerance of the linear solver - default = 1e-6.

4.34.2.9 `double PJFNK_DATA::nl_res`

Absolute residual norm for the non-linear system.

4.34.2.10 `double PJFNK_DATA::nl_relres`

Relative residual for the non-linear system.

4.34.2.11 `double PJFNK_DATA::nl_res_base`

Initial residual norm for the non-linear system.

4.34.2.12 `double PJFNK_DATA::nl_bestres`

Best found residual norm.

4.34.2.13 `double PJFNK_DATA::eps = sqrt(DBL_EPSILON)`

Value of epsilon used jacvec - default = sqrt(DBL\_EPSILON)

4.34.2.14 `bool PJFNK_DATA::NL_Output = true`

True = print PJFNK messages to console.

4.34.2.15 `bool PJFNK_DATA::L_Output = false`

True = print Linear messages to console.

4.34.2.16 `bool PJFNK_DATA::LineSearch = false`

True = use Backtracking Linesearch for global convergence.

4.34.2.17 `bool PJFNK_DATA::Bounce = false`

True = allow Linesearch to go outside local well, False = Strict local convergence.

4.34.2.18 `Matrix<double> PJFNK_DATA::F`

Stored fuction evaluation at x (also the residual)

4.34.2.19 `Matrix<double> PJFNK_DATA::Fv`

Stored function evaluation at  $x + \text{eps} * v$ .

4.34.2.20 `Matrix<double> PJFNK_DATA::v`

Stored vector of  $x + \text{eps} * v$ .

4.34.2.21 `Matrix<double> PJFNK_DATA::x`

Current solution vector for the non-linear system.

4.34.2.22 `Matrix<double> PJFNK_DATA::bestx`

Best found solution vector to the non-linear system.

4.34.2.23 `GMRESLP_DATA PJFNK_DATA::gmreslp_dat`

Data structure for the GMRESLP method.

4.34.2.24 `PCG_DATA PJFNK_DATA::pcg_dat`

Data structure for the PCG method.

4.34.2.25 `BiCGSTAB_DATA PJFNK_DATA::bicgstab_dat`

Data structure for the BiCGSTAB method.

**4.34.2.26 CGS\_DATA PJFNK\_DATA::cgs\_dat**

Data structure for the CGS method.

**4.34.2.27 GMRESRP\_DATA PJFNK\_DATA::gmresrp\_dat**

Data structure for the GMRESRP method.

**4.34.2.28 GCR\_DATA PJFNK\_DATA::gcr\_dat**

Data structure for the GCR method.

**4.34.2.29 GMRESR\_DATA PJFNK\_DATA::gmresr\_dat**

Data structure for the GMRESR method.

**4.34.2.30 BACKTRACK\_DATA PJFNK\_DATA::backtrack\_dat**

Data structure for the Backtracking Linesearch algorithm.

**4.34.2.31 const void\* PJFNK\_DATA::res\_data**

Data structure pointer for user's residual data.

**4.34.2.32 const void\* PJFNK\_DATA::precon\_data**

Data structure pointer for user's preconditioning data.

**4.34.2.33 int(\* PJFNK\_DATA::funeval)(const Matrix< double > &x, Matrix< double > &F, const void \*res\_data)**

Function pointer for the user's function  $F(x)$  using there data.

**4.34.2.34 int(\* PJFNK\_DATA::precon)(const Matrix< double > &r, Matrix< double > &p, const void \*precon\_data)**

Function pointer for the user's preconditioning function for the linear system.

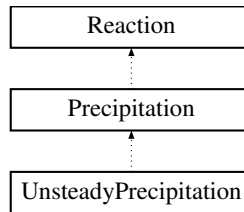
The documentation for this struct was generated from the following file:

- [lark.h](#)

## 4.35 Precipitation Class Reference

```
#include <shark.h>
```

Inheritance diagram for Precipitation:



### Additional Inherited Members

The documentation for this class was generated from the following file:

- [shark.h](#)

## 4.36 PURE\_GAS Struct Reference

Data structure holding all the parameters for each pure gas species.

```
#include <egret.h>
```

### Public Attributes

- double [molecular\\_weight](#)  
*Given: molecular weights (g/mol)*
- double [Sutherland\\_Temp](#)  
*Given: Sutherland's Reference Temperature (K)*
- double [Sutherland\\_Const](#)  
*Given: Sutherland's Constant (K)*
- double [Sutherland\\_Viscosity](#)  
*Given: Sutherland's Reference Viscosity (g/cm/s)*
- double [specific\\_heat](#)  
*Given: Specific heat of the gas (J/g/K)*
- double [molecular\\_diffusion](#)  
*Calculated: molecular diffusivities (cm<sup>2</sup>/s)*
- double [dynamic\\_viscosity](#)  
*Calculated: dynamic viscosities (g/cm/s)*
- double [density](#)  
*Calculated: gas densities (g/cm<sup>3</sup>) {use RE3}.*
- double [Schmidt](#)  
*Calculated: Value of the Schmidt number (-)*

#### 4.36.1 Detailed Description

Data structure holding all the parameters for each pure gas species.

C-style object that holds the constants and parameters associated with each pure gas species in the overall mixture. This information is used in conjunction with the kinetic theory of gases to produce approximations to many different gas properties needed in simulating gas dynamics, mobility of a gas through porous media, as well as some kinetic adsorption parameters such as diffusivities.

### 4.36.2 Member Data Documentation

#### 4.36.2.1 double PURE\_GAS::molecular\_weight

Given: molecular weights (g/mol)

#### 4.36.2.2 double PURE\_GAS::Sutherland\_Temp

Given: Sutherland's Reference Temperature (K)

#### 4.36.2.3 double PURE\_GAS::Sutherland\_Const

Given: Sutherland's Constant (K)

#### 4.36.2.4 double PURE\_GAS::Sutherland\_Viscosity

Given: Sutherland's Reference Viscosity (g/cm/s)

#### 4.36.2.5 double PURE\_GAS::specific\_heat

Given: Specific heat of the gas (J/g/K)

#### 4.36.2.6 double PURE\_GAS::molecular\_diffusion

Calculated: molecular diffusivities (cm<sup>2</sup>/s)

#### 4.36.2.7 double PURE\_GAS::dynamic\_viscosity

Calculated: dynamic viscosities (g/cm/s)

#### 4.36.2.8 double PURE\_GAS::density

Calculated: gas densities (g/cm<sup>3</sup>) {use RE3}.

#### 4.36.2.9 double PURE\_GAS::Schmidt

Calculated: Value of the Schmidt number (-)

The documentation for this struct was generated from the following file:

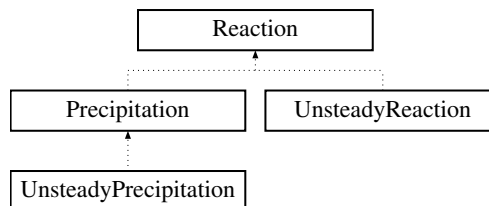
- [egret.h](#)

## 4.37 Reaction Class Reference

```
#include <shark.h>
```

Inheritance diagram for Reaction:





### Public Member Functions

- [Reaction](#) ()
- [~Reaction](#) ()
- void [Initialize\\_List](#) ([MasterSpeciesList](#) &[List](#))
- void [Display\\_Info](#) ()
- void [Set\\_Stoichiometric](#) (int i, double v)
- void [Set\\_Equilibrium](#) (double v)
- void [Set\\_Enthalpy](#) (double H)
- void [Set\\_Entropy](#) (double S)
- void [Set\\_EnthalpyANDEntropy](#) (double H, double S)
- void [Set\\_Energy](#) (double G)
- void [checkSpeciesEnergies](#) ()
- void [calculateEnergies](#) ()
- void [calculateEquilibrium](#) (double T)
- bool [haveEquilibrium](#) ()
- double [Get\\_Stoichiometric](#) (int i)
- double [Get\\_Equilibrium](#) ()
- double [Get\\_Enthalpy](#) ()
- double [Get\\_Entropy](#) ()
- double [Get\\_Energy](#) ()
- double [Eval\\_Residual](#) (const [Matrix](#)< double > &x, const [Matrix](#)< double > &gama)

### Protected Attributes

- [MasterSpeciesList](#) \* [List](#)
- std::vector< double > [Stoichiometric](#)
- double [Equilibrium](#)
- double [enthalpy](#)
- double [entropy](#)
- double [energy](#)
- bool [CanCalcHS](#)
- bool [CanCalcG](#)
- bool [HaveHS](#)
- bool [HaveG](#)
- bool [HaveEquil](#)

## 4.37.1 Constructor & Destructor Documentation

4.37.1.1 [Reaction::Reaction](#) ( )

4.37.1.2 [Reaction::~~Reaction](#) ( )

## 4.37.2 Member Function Documentation

- 4.37.2.1 void Reaction::Initialize\_List ( MasterSpeciesList & List )
- 4.37.2.2 void Reaction::Display\_Info ( )
- 4.37.2.3 void Reaction::Set\_Stoichiometric ( int *i*, double *v* )
- 4.37.2.4 void Reaction::Set\_Equilibrium ( double *v* )
- 4.37.2.5 void Reaction::Set\_Enthalpy ( double *H* )
- 4.37.2.6 void Reaction::Set\_Entropy ( double *S* )
- 4.37.2.7 void Reaction::Set\_EnthalpyANDEntropy ( double *H*, double *S* )
- 4.37.2.8 void Reaction::Set\_Energy ( double *G* )
- 4.37.2.9 void Reaction::checkSpeciesEnergies ( )
- 4.37.2.10 void Reaction::calculateEnergies ( )
- 4.37.2.11 void Reaction::calculateEquilibrium ( double *T* )
- 4.37.2.12 bool Reaction::haveEquilibrium ( )
- 4.37.2.13 double Reaction::Get\_Stoichiometric ( int *i* )
- 4.37.2.14 double Reaction::Get\_Equilibrium ( )
- 4.37.2.15 double Reaction::Get\_Enthalpy ( )
- 4.37.2.16 double Reaction::Get\_Entropy ( )
- 4.37.2.17 double Reaction::Get\_Energy ( )
- 4.37.2.18 double Reaction::Eval\_Residual ( const Matrix< double > & *x*, const Matrix< double > & *gama* )

### 4.37.3 Member Data Documentation

- 4.37.3.1 MasterSpeciesList\* Reaction::List [protected]
- 4.37.3.2 std::vector<double> Reaction::Stoichiometric [protected]
- 4.37.3.3 double Reaction::Equilibrium [protected]
- 4.37.3.4 double Reaction::enthalpy [protected]
- 4.37.3.5 double Reaction::entropy [protected]
- 4.37.3.6 double Reaction::energy [protected]
- 4.37.3.7 bool Reaction::CanCalcHS [protected]
- 4.37.3.8 bool Reaction::CanCalcG [protected]
- 4.37.3.9 bool Reaction::HaveHS [protected]

4.37.3.10 `bool Reaction::HaveG` [protected]

4.37.3.11 `bool Reaction::HaveEquil` [protected]

The documentation for this class was generated from the following file:

- [shark.h](#)

## 4.38 SCOPSOWL\_DATA Struct Reference

```
#include <scopsowl.h>
```

### Public Attributes

- unsigned long int [total\\_steps](#)
- int [coord\\_macro](#)
- int [coord\\_micro](#)
- int [level](#) = 2
- double [sim\\_time](#)
- double [t\\_old](#)
- double [t](#)
- double [t\\_counter](#) = 0.0
- double [t\\_print](#)
- bool [Print2File](#) = true
- bool [Print2Console](#) = true
- bool [SurfDiff](#) = true
- bool [Heterogeneous](#) = true
- double [gas\\_velocity](#)
- double [total\\_pressure](#)
- double [gas\\_temperature](#)
- double [pellet\\_radius](#)
- double [crystal\\_radius](#)
- double [char\\_macro](#)
- double [char\\_micro](#)
- double [binder\\_fraction](#)
- double [binder\\_porosity](#)
- double [binder\\_poresize](#)
- double [pellet\\_density](#)
- bool [DirichletBC](#) = false
- bool [NonLinear](#) = true
- std::vector< double > [y](#)
- std::vector< double > [tempy](#)
- FILE \* [OutputFile](#)
- double(\* [eval\\_ads](#) )(int i, int l, const void \*[user\\_data](#))
- double(\* [eval\\_retard](#) )(int i, int l, const void \*[user\\_data](#))
- double(\* [eval\\_diff](#) )(int i, int l, const void \*[user\\_data](#))
- double(\* [eval\\_surfDiff](#) )(int i, int l, const void \*[user\\_data](#))
- double(\* [eval\\_kf](#) )(int i, const void \*[user\\_data](#))
- const void \* [user\\_data](#)
- MIXED\_GAS \* [gas\\_dat](#)
- MAGPIE\_DATA [magpie\\_dat](#)
- std::vector< FINCH\_DATA > [finch\\_dat](#)
- std::vector< SCOPSOWL\_PARAM\_DATA > [param\\_dat](#)
- std::vector< SKUA\_DATA > [skua\\_dat](#)

#### 4.38.1 Member Data Documentation

- 4.38.1.1 unsigned long int SCOPSOWL\_DATA::total\_steps
- 4.38.1.2 int SCOPSOWL\_DATA::coord\_macro
- 4.38.1.3 int SCOPSOWL\_DATA::coord\_micro
- 4.38.1.4 int SCOPSOWL\_DATA::level = 2
- 4.38.1.5 double SCOPSOWL\_DATA::sim\_time
- 4.38.1.6 double SCOPSOWL\_DATA::t\_old
- 4.38.1.7 double SCOPSOWL\_DATA::t
- 4.38.1.8 double SCOPSOWL\_DATA::t\_counter = 0.0
- 4.38.1.9 double SCOPSOWL\_DATA::t\_print
- 4.38.1.10 bool SCOPSOWL\_DATA::Print2File = true
- 4.38.1.11 bool SCOPSOWL\_DATA::Print2Console = true
- 4.38.1.12 bool SCOPSOWL\_DATA::SurfDiff = true
- 4.38.1.13 bool SCOPSOWL\_DATA::Heterogeneous = true
- 4.38.1.14 double SCOPSOWL\_DATA::gas\_velocity
- 4.38.1.15 double SCOPSOWL\_DATA::total\_pressure
- 4.38.1.16 double SCOPSOWL\_DATA::gas\_temperature
- 4.38.1.17 double SCOPSOWL\_DATA::pellet\_radius
- 4.38.1.18 double SCOPSOWL\_DATA::crystal\_radius
- 4.38.1.19 double SCOPSOWL\_DATA::char\_macro
- 4.38.1.20 double SCOPSOWL\_DATA::char\_micro
- 4.38.1.21 double SCOPSOWL\_DATA::binder\_fraction
- 4.38.1.22 double SCOPSOWL\_DATA::binder\_porosity
- 4.38.1.23 double SCOPSOWL\_DATA::binder\_poresize
- 4.38.1.24 double SCOPSOWL\_DATA::pellet\_density
- 4.38.1.25 bool SCOPSOWL\_DATA::DirichletBC = false
- 4.38.1.26 bool SCOPSOWL\_DATA::NonLinear = true
- 4.38.1.27 std::vector<double> SCOPSOWL\_DATA::y

- 4.38.1.28 `std::vector<double> SCOPSOWL_DATA::tempy`
- 4.38.1.29 `FILE* SCOPSOWL_DATA::OutputFile`
- 4.38.1.30 `double(* SCOPSOWL_DATA::eval_ads)(int i, int l, const void *user_data)`
- 4.38.1.31 `double(* SCOPSOWL_DATA::eval_retard)(int i, int l, const void *user_data)`
- 4.38.1.32 `double(* SCOPSOWL_DATA::eval_diff)(int i, int l, const void *user_data)`
- 4.38.1.33 `double(* SCOPSOWL_DATA::eval_surfDiff)(int i, int l, const void *user_data)`
- 4.38.1.34 `double(* SCOPSOWL_DATA::eval_kf)(int i, const void *user_data)`
- 4.38.1.35 `const void* SCOPSOWL_DATA::user_data`
- 4.38.1.36 `MIXED_GAS* SCOPSOWL_DATA::gas_dat`
- 4.38.1.37 `MAGPIE_DATA SCOPSOWL_DATA::magpie_dat`
- 4.38.1.38 `std::vector<FINCH_DATA> SCOPSOWL_DATA::finch_dat`
- 4.38.1.39 `std::vector<SCOPSOWL_PARAM_DATA> SCOPSOWL_DATA::param_dat`
- 4.38.1.40 `std::vector<SKUA_DATA> SCOPSOWL_DATA::skua_dat`

The documentation for this struct was generated from the following file:

- [scopsowl.h](#)

## 4.39 SCOPSOWL\_OPT\_DATA Struct Reference

```
#include <scopsowl_opt.h>
```

### Public Attributes

- int [num\\_curves](#)
- int [evaluation](#)
- unsigned long int [total\\_eval](#)
- int [current\\_points](#)
- int [num\\_params](#) = 1
- int [diffusion\\_type](#)
- int [adsorb\\_index](#)
- int [max\\_guess\\_iter](#) = 20
- bool [Optimize](#)
- bool [Rough](#)
- double [current\\_temp](#)
- double [current\\_press](#)
- double [current\\_equil](#)
- double [simulation\\_equil](#)
- double [max\\_bias](#)
- double [min\\_bias](#)
- double [e\\_norm](#)
- double [f\\_bias](#)

- double [e\\_norm\\_old](#)
- double [f\\_bias\\_old](#)
- double [param\\_guess](#)
- double [param\\_guess\\_old](#)
- double [rel\\_tol\\_norm](#) = 0.01
- double [abs\\_tol\\_bias](#) = 1.0
- std::vector< double > [y\\_base](#)
- std::vector< double > [q\\_data](#)
- std::vector< double > [q\\_sim](#)
- std::vector< double > [t](#)
- FILE \* [ParamFile](#)
- FILE \* [CompareFile](#)
- [SCOPSOWL\\_DATA](#) owl\_dat

#### 4.39.1 Member Data Documentation

- 4.39.1.1 int SCOPSOWL\_OPT\_DATA::num\_curves
- 4.39.1.2 int SCOPSOWL\_OPT\_DATA::evaluation
- 4.39.1.3 unsigned long int SCOPSOWL\_OPT\_DATA::total\_eval
- 4.39.1.4 int SCOPSOWL\_OPT\_DATA::current\_points
- 4.39.1.5 int SCOPSOWL\_OPT\_DATA::num\_params = 1
- 4.39.1.6 int SCOPSOWL\_OPT\_DATA::diffusion\_type
- 4.39.1.7 int SCOPSOWL\_OPT\_DATA::adsorb\_index
- 4.39.1.8 int SCOPSOWL\_OPT\_DATA::max\_guess\_iter = 20
- 4.39.1.9 bool SCOPSOWL\_OPT\_DATA::Optimize
- 4.39.1.10 bool SCOPSOWL\_OPT\_DATA::Rough
- 4.39.1.11 double SCOPSOWL\_OPT\_DATA::current\_temp
- 4.39.1.12 double SCOPSOWL\_OPT\_DATA::current\_press
- 4.39.1.13 double SCOPSOWL\_OPT\_DATA::current\_equil
- 4.39.1.14 double SCOPSOWL\_OPT\_DATA::simulation\_equil
- 4.39.1.15 double SCOPSOWL\_OPT\_DATA::max\_bias
- 4.39.1.16 double SCOPSOWL\_OPT\_DATA::min\_bias
- 4.39.1.17 double SCOPSOWL\_OPT\_DATA::e\_norm
- 4.39.1.18 double SCOPSOWL\_OPT\_DATA::f\_bias
- 4.39.1.19 double SCOPSOWL\_OPT\_DATA::e\_norm\_old
- 4.39.1.20 double SCOPSOWL\_OPT\_DATA::f\_bias\_old

- 4.39.1.21 double SCOPSOWL\_OPT\_DATA::param\_guess
- 4.39.1.22 double SCOPSOWL\_OPT\_DATA::param\_guess\_old
- 4.39.1.23 double SCOPSOWL\_OPT\_DATA::rel\_tol\_norm = 0.01
- 4.39.1.24 double SCOPSOWL\_OPT\_DATA::abs\_tol\_bias = 1.0
- 4.39.1.25 std::vector<double> SCOPSOWL\_OPT\_DATA::y\_base
- 4.39.1.26 std::vector<double> SCOPSOWL\_OPT\_DATA::q\_data
- 4.39.1.27 std::vector<double> SCOPSOWL\_OPT\_DATA::q\_sim
- 4.39.1.28 std::vector<double> SCOPSOWL\_OPT\_DATA::t
- 4.39.1.29 FILE\* SCOPSOWL\_OPT\_DATA::ParamFile
- 4.39.1.30 FILE\* SCOPSOWL\_OPT\_DATA::CompareFile
- 4.39.1.31 SCOPSOWL\_DATA SCOPSOWL\_OPT\_DATA::owl\_dat

The documentation for this struct was generated from the following file:

- [scopsowl\\_opt.h](#)

## 4.40 SCOPSOWL\_PARAM\_DATA Struct Reference

```
#include <scopsowl.h>
```

### Public Attributes

- [Matrix](#)< double > [qAvg](#)
- [Matrix](#)< double > [qAvg\\_old](#)
- [Matrix](#)< double > [Qst](#)
- [Matrix](#)< double > [Qst\\_old](#)
- [Matrix](#)< double > [dq\\_dc](#)
- double [xIC](#)
- double [qIntegralAvg](#)
- double [qIntegralAvg\\_old](#)
- double [QstAvg](#)
- double [QstAvg\\_old](#)
- double [qo](#)
- double [Qsto](#)
- double [dq\\_dco](#)
- double [pore\\_diffusion](#)
- double [film\\_transfer](#)
- double [activation\\_energy](#)
- double [ref\\_diffusion](#)
- double [ref\\_temperature](#)
- double [affinity](#)
- double [ref\\_pressure](#)
- bool [Adsorbable](#)
- std::string [speciesName](#)

#### 4.40.1 Member Data Documentation

- 4.40.1.1 `Matrix<double> SCOPSOWL_PARAM_DATA::qAvg`
- 4.40.1.2 `Matrix<double> SCOPSOWL_PARAM_DATA::qAvg_old`
- 4.40.1.3 `Matrix<double> SCOPSOWL_PARAM_DATA::Qst`
- 4.40.1.4 `Matrix<double> SCOPSOWL_PARAM_DATA::Qst_old`
- 4.40.1.5 `Matrix<double> SCOPSOWL_PARAM_DATA::dq_dc`
- 4.40.1.6 `double SCOPSOWL_PARAM_DATA::xIC`
- 4.40.1.7 `double SCOPSOWL_PARAM_DATA::qIntegralAvg`
- 4.40.1.8 `double SCOPSOWL_PARAM_DATA::qIntegralAvg_old`
- 4.40.1.9 `double SCOPSOWL_PARAM_DATA::QstAvg`
- 4.40.1.10 `double SCOPSOWL_PARAM_DATA::QstAvg_old`
- 4.40.1.11 `double SCOPSOWL_PARAM_DATA::qo`
- 4.40.1.12 `double SCOPSOWL_PARAM_DATA::Qsto`
- 4.40.1.13 `double SCOPSOWL_PARAM_DATA::dq_dco`
- 4.40.1.14 `double SCOPSOWL_PARAM_DATA::pore_diffusion`
- 4.40.1.15 `double SCOPSOWL_PARAM_DATA::film_transfer`
- 4.40.1.16 `double SCOPSOWL_PARAM_DATA::activation_energy`
- 4.40.1.17 `double SCOPSOWL_PARAM_DATA::ref_diffusion`
- 4.40.1.18 `double SCOPSOWL_PARAM_DATA::ref_temperature`
- 4.40.1.19 `double SCOPSOWL_PARAM_DATA::affinity`
- 4.40.1.20 `double SCOPSOWL_PARAM_DATA::ref_pressure`
- 4.40.1.21 `bool SCOPSOWL_PARAM_DATA::Adsorbable`
- 4.40.1.22 `std::string SCOPSOWL_PARAM_DATA::speciesName`

The documentation for this struct was generated from the following file:

- [scopsowl.h](#)

#### 4.41 SHARK\_DATA Struct Reference

```
#include <shark.h>
```



## Public Attributes

- [MasterSpeciesList](#) MasterList
- `std::vector< Reaction > ReactionList`
- `std::vector< MassBalance > MassBalanceList`
- `std::vector< UnsteadyReaction > UnsteadyList`
- `std::vector< double(*) (const Matrix< double > &x, SHARK\_DATA *shark_dat, const void *data) > OtherList`
- `int numvar`
- `int num_ssr`
- `int num_mbe`
- `int num_usr`
- `int num_other = 0`
- `int act_fun = IDEAL`
- `int totalsteps = 0`
- `int timesteps = 0`
- `int pH_index = -1`
- `int pOH_index = -1`
- `double simulationtime = 0.0`
- `double dt = 0.1`
- `double dt_min = sqrt(DBL_EPSILON)`
- `double t_out = 0.0`
- `double t_count = 0.0`
- `double time = 0.0`
- `double time_old = 0.0`
- `double pH = 7.0`
- `double Norm = 0.0`
- `double dielectric_const = 78.325`
- `double temperature = 298.15`
- `bool steadystate = true`
- `bool TimeAdaptivity = false`
- `bool const_pH = false`
- `bool SpeciationCurve = false`
- `bool Console_Output = true`
- `bool File_Output = false`
- `bool Contains_pH = false`
- `bool Contains_pOH = false`
- `bool Converged = false`
- `Matrix< double > X_old`
- `Matrix< double > X_new`
- `Matrix< double > Conc_old`
- `Matrix< double > Conc_new`
- `Matrix< double > activity_new`
- `Matrix< double > activity_old`
- `int(* EvalActivity)(const Matrix< double > &x, Matrix< double > &F, const void *data)`
- `int(* Residual)(const Matrix< double > &x, Matrix< double > &F, const void *data)`
- `int(* lin_precon)(const Matrix< double > &r, Matrix< double > &p, const void *data)`
- [PJFNK\\_DATA](#) Newton\_data
- `const void * activity_data`
- `const void * residual_data`
- `const void * precon_data`
- `const void * other_data`
- `FILE * OutputFile`
- `yaml_cpp_class yaml_object`

#### 4.41.1 Member Data Documentation

4.41.1.1 MasterSpeciesList SHARK\_DATA::MasterList

4.41.1.2 std::vector<Reaction> SHARK\_DATA::ReactionList

4.41.1.3 std::vector<MassBalance> SHARK\_DATA::MassBalanceList

4.41.1.4 std::vector<UnsteadyReaction> SHARK\_DATA::UnsteadyList

4.41.1.5 std::vector< double (\*) (const Matrix<double> &x, SHARK\_DATA \*shark\_dat, const void \*data) >  
SHARK\_DATA::OtherList

4.41.1.6 int SHARK\_DATA::numvar

4.41.1.7 int SHARK\_DATA::num\_ssr

4.41.1.8 int SHARK\_DATA::num\_mbe

4.41.1.9 int SHARK\_DATA::num\_usr

4.41.1.10 int SHARK\_DATA::num\_other = 0

4.41.1.11 int SHARK\_DATA::act\_fun = IDEAL

4.41.1.12 int SHARK\_DATA::totalsteps = 0

4.41.1.13 int SHARK\_DATA::timesteps = 0

4.41.1.14 int SHARK\_DATA::pH\_index = -1

4.41.1.15 int SHARK\_DATA::pOH\_index = -1

4.41.1.16 double SHARK\_DATA::simulationtime = 0.0

4.41.1.17 double SHARK\_DATA::dt = 0.1

4.41.1.18 double SHARK\_DATA::dt\_min = sqrt(DBL\_EPSILON)

4.41.1.19 double SHARK\_DATA::t\_out = 0.0

4.41.1.20 double SHARK\_DATA::t\_count = 0.0

4.41.1.21 double SHARK\_DATA::time = 0.0

4.41.1.22 double SHARK\_DATA::time\_old = 0.0

4.41.1.23 double SHARK\_DATA::pH = 7.0

4.41.1.24 double SHARK\_DATA::Norm = 0.0

4.41.1.25 double SHARK\_DATA::dielectric\_const = 78.325

4.41.1.26 double SHARK\_DATA::temperature = 298.15

4.41.1.27 bool SHARK\_DATA::steadystate = true

- 4.41.1.28 bool SHARK\_DATA::TimeAdaptivity = false
- 4.41.1.29 bool SHARK\_DATA::const\_pH = false
- 4.41.1.30 bool SHARK\_DATA::SpeciationCurve = false
- 4.41.1.31 bool SHARK\_DATA::Console\_Output = true
- 4.41.1.32 bool SHARK\_DATA::File\_Output = false
- 4.41.1.33 bool SHARK\_DATA::Contains\_pH = false
- 4.41.1.34 bool SHARK\_DATA::Contains\_pOH = false
- 4.41.1.35 bool SHARK\_DATA::Converged = false
- 4.41.1.36 Matrix<double> SHARK\_DATA::X\_old
- 4.41.1.37 Matrix<double> SHARK\_DATA::X\_new
- 4.41.1.38 Matrix<double> SHARK\_DATA::Conc\_old
- 4.41.1.39 Matrix<double> SHARK\_DATA::Conc\_new
- 4.41.1.40 Matrix<double> SHARK\_DATA::activity\_new
- 4.41.1.41 Matrix<double> SHARK\_DATA::activity\_old
- 4.41.1.42 int(\* SHARK\_DATA::EvalActivity)(const Matrix< double > &x, Matrix< double > &F, const void \*data)
- 4.41.1.43 int(\* SHARK\_DATA::Residual)(const Matrix< double > &x, Matrix< double > &F, const void \*data)
- 4.41.1.44 int(\* SHARK\_DATA::lin\_precon)(const Matrix< double > &r, Matrix< double > &p, const void \*data)
- 4.41.1.45 PJFNK\_DATA SHARK\_DATA::Newton\_data
- 4.41.1.46 const void\* SHARK\_DATA::activity\_data
- 4.41.1.47 const void\* SHARK\_DATA::residual\_data
- 4.41.1.48 const void\* SHARK\_DATA::precon\_data
- 4.41.1.49 const void\* SHARK\_DATA::other\_data
- 4.41.1.50 FILE\* SHARK\_DATA::OutputFile
- 4.41.1.51 yaml\_cpp\_class SHARK\_DATA::yaml\_object

The documentation for this struct was generated from the following file:

- [shark.h](#)

## 4.42 SKUA\_DATA Struct Reference

```
#include <skua.h>
```

## Public Attributes

- unsigned long int [total\\_steps](#)
- int [coord](#)
- double [sim\\_time](#)
- double [t\\_old](#)
- double [t](#)
- double [t\\_counter](#) = 0.0
- double [t\\_print](#)
- double [qTn](#)
- double [qTnp1](#)
- bool [Print2File](#) = true
- bool [Print2Console](#) = true
- double [gas\\_velocity](#)
- double [pellet\\_radius](#)
- double [char\\_measure](#)
- bool [DirichletBC](#) = true
- bool [NonLinear](#) = true
- std::vector< double > [y](#)
- FILE \* [OutputFile](#)
- double(\* [eval\\_diff](#) )(int i, int l, const void \*[user\\_data](#))
- double(\* [eval\\_kf](#) )(int i, const void \*[user\\_data](#))
- const void \* [user\\_data](#)
- [MAGPIE\\_DATA](#) [magpie\\_dat](#)
- [MIXED\\_GAS](#) \* [gas\\_dat](#)
- std::vector< [FINCH\\_DATA](#) > [finch\\_dat](#)
- std::vector< [SKUA\\_PARAM](#) > [param\\_dat](#)

### 4.42.1 Member Data Documentation

4.42.1.1 unsigned long int [SKUA\\_DATA::total\\_steps](#)

4.42.1.2 int [SKUA\\_DATA::coord](#)

4.42.1.3 double [SKUA\\_DATA::sim\\_time](#)

4.42.1.4 double [SKUA\\_DATA::t\\_old](#)

4.42.1.5 double [SKUA\\_DATA::t](#)

4.42.1.6 double [SKUA\\_DATA::t\\_counter](#) = 0.0

4.42.1.7 double [SKUA\\_DATA::t\\_print](#)

4.42.1.8 double [SKUA\\_DATA::qTn](#)

4.42.1.9 double [SKUA\\_DATA::qTnp1](#)

4.42.1.10 bool [SKUA\\_DATA::Print2File](#) = true

4.42.1.11 bool [SKUA\\_DATA::Print2Console](#) = true

4.42.1.12 double [SKUA\\_DATA::gas\\_velocity](#)

- 4.42.1.13 double SKUA\_DATA::pellet\_radius
- 4.42.1.14 double SKUA\_DATA::char\_measure
- 4.42.1.15 bool SKUA\_DATA::DirichletBC = true
- 4.42.1.16 bool SKUA\_DATA::NonLinear = true
- 4.42.1.17 std::vector<double> SKUA\_DATA::y
- 4.42.1.18 FILE\* SKUA\_DATA::OutputFile
- 4.42.1.19 double(\* SKUA\_DATA::eval\_diff)(int i, int l, const void \*user\_data)
- 4.42.1.20 double(\* SKUA\_DATA::eval\_kf)(int i, const void \*user\_data)
- 4.42.1.21 const void\* SKUA\_DATA::user\_data
- 4.42.1.22 MAGPIE\_DATA SKUA\_DATA::magpie\_dat
- 4.42.1.23 MIXED\_GAS\* SKUA\_DATA::gas\_dat
- 4.42.1.24 std::vector<FINCH\_DATA> SKUA\_DATA::finch\_dat
- 4.42.1.25 std::vector<SKUA\_PARAM> SKUA\_DATA::param\_dat

The documentation for this struct was generated from the following file:

- [skua.h](#)

## 4.43 SKUA\_OPT\_DATA Struct Reference

```
#include <skua_opt.h>
```

### Public Attributes

- int [num\\_curves](#)
- int [evaluation](#)
- unsigned long int [total\\_eval](#)
- int [current\\_points](#)
- int [num\\_params](#) = 1
- int [diffusion\\_type](#)
- int [adsorb\\_index](#)
- int [max\\_guess\\_iter](#) = 20
- bool [Optimize](#)
- bool [Rough](#)
- double [current\\_temp](#)
- double [current\\_press](#)
- double [current\\_equil](#)
- double [simulation\\_equil](#)
- double [max\\_bias](#)
- double [min\\_bias](#)
- double [e\\_norm](#)
- double [f\\_bias](#)

- double [e\\_norm\\_old](#)
- double [f\\_bias\\_old](#)
- double [param\\_guess](#)
- double [param\\_guess\\_old](#)
- double [rel\\_tol\\_norm](#) = 0.1
- double [abs\\_tol\\_bias](#) = 0.1
- std::vector< double > [y\\_base](#)
- std::vector< double > [q\\_data](#)
- std::vector< double > [q\\_sim](#)
- std::vector< double > [t](#)
- FILE \* [ParamFile](#)
- FILE \* [CompareFile](#)
- [SKUA\\_DATA](#) [skua\\_dat](#)

#### 4.43.1 Member Data Documentation

- 4.43.1.1 int [SKUA\\_OPT\\_DATA::num\\_curves](#)
- 4.43.1.2 int [SKUA\\_OPT\\_DATA::evaluation](#)
- 4.43.1.3 unsigned long int [SKUA\\_OPT\\_DATA::total\\_eval](#)
- 4.43.1.4 int [SKUA\\_OPT\\_DATA::current\\_points](#)
- 4.43.1.5 int [SKUA\\_OPT\\_DATA::num\\_params](#) = 1
- 4.43.1.6 int [SKUA\\_OPT\\_DATA::diffusion\\_type](#)
- 4.43.1.7 int [SKUA\\_OPT\\_DATA::adsorb\\_index](#)
- 4.43.1.8 int [SKUA\\_OPT\\_DATA::max\\_guess\\_iter](#) = 20
- 4.43.1.9 bool [SKUA\\_OPT\\_DATA::Optimize](#)
- 4.43.1.10 bool [SKUA\\_OPT\\_DATA::Rough](#)
- 4.43.1.11 double [SKUA\\_OPT\\_DATA::current\\_temp](#)
- 4.43.1.12 double [SKUA\\_OPT\\_DATA::current\\_press](#)
- 4.43.1.13 double [SKUA\\_OPT\\_DATA::current\\_equil](#)
- 4.43.1.14 double [SKUA\\_OPT\\_DATA::simulation\\_equil](#)
- 4.43.1.15 double [SKUA\\_OPT\\_DATA::max\\_bias](#)
- 4.43.1.16 double [SKUA\\_OPT\\_DATA::min\\_bias](#)
- 4.43.1.17 double [SKUA\\_OPT\\_DATA::e\\_norm](#)
- 4.43.1.18 double [SKUA\\_OPT\\_DATA::f\\_bias](#)
- 4.43.1.19 double [SKUA\\_OPT\\_DATA::e\\_norm\\_old](#)
- 4.43.1.20 double [SKUA\\_OPT\\_DATA::f\\_bias\\_old](#)

- 4.43.1.21 double SKUA\_OPT\_DATA::param\_guess
- 4.43.1.22 double SKUA\_OPT\_DATA::param\_guess\_old
- 4.43.1.23 double SKUA\_OPT\_DATA::rel\_tol\_norm = 0.1
- 4.43.1.24 double SKUA\_OPT\_DATA::abs\_tol\_bias = 0.1
- 4.43.1.25 std::vector<double> SKUA\_OPT\_DATA::y\_base
- 4.43.1.26 std::vector<double> SKUA\_OPT\_DATA::q\_data
- 4.43.1.27 std::vector<double> SKUA\_OPT\_DATA::q\_sim
- 4.43.1.28 std::vector<double> SKUA\_OPT\_DATA::t
- 4.43.1.29 FILE\* SKUA\_OPT\_DATA::ParamFile
- 4.43.1.30 FILE\* SKUA\_OPT\_DATA::CompareFile
- 4.43.1.31 SKUA\_DATA SKUA\_OPT\_DATA::skua\_dat

The documentation for this struct was generated from the following file:

- [skua\\_opt.h](#)

## 4.44 SKUA\_PARAM Struct Reference

```
#include <skua.h>
```

### Public Attributes

- double [activation\\_energy](#)
- double [ref\\_diffusion](#)
- double [ref\\_temperature](#)
- double [affinity](#)
- double [ref\\_pressure](#)
- double [film\\_transfer](#)
- double [xIC](#)
- double [y\\_eff](#)
- double [Qstn](#)
- double [Qstnp1](#)
- double [xn](#)
- double [xnp1](#)
- bool [Adsorbable](#)
- std::string [speciesName](#)

### 4.44.1 Member Data Documentation

- 4.44.1.1 double SKUA\_PARAM::activation\_energy
- 4.44.1.2 double SKUA\_PARAM::ref\_diffusion

4.44.1.3 double SKUA\_PARAM::ref\_temperature

4.44.1.4 double SKUA\_PARAM::affinity

4.44.1.5 double SKUA\_PARAM::ref\_pressure

4.44.1.6 double SKUA\_PARAM::film\_transfer

4.44.1.7 double SKUA\_PARAM::xIC

4.44.1.8 double SKUA\_PARAM::y\_eff

4.44.1.9 double SKUA\_PARAM::Qstn

4.44.1.10 double SKUA\_PARAM::Qstnp1

4.44.1.11 double SKUA\_PARAM::xn

4.44.1.12 double SKUA\_PARAM::xnp1

4.44.1.13 bool SKUA\_PARAM::Adsorbable

4.44.1.14 std::string SKUA\_PARAM::speciesName

The documentation for this struct was generated from the following file:

- [skua.h](#)

## 4.45 Speciation\_Test01\_Data Struct Reference

```
#include <sandbox.h>
```

### Public Attributes

- int [N](#) = 4
- const double [logKw](#) = -14.0
- const double [logKa1](#) = -6.35
- const double [logKa2](#) = -10.33
- double [CT](#) = 0.1786
- double [NaT](#) = 0.1786
- std::vector< [Molecule](#) > [x](#)
- [Matrix](#)< double > [Jacobian](#)
- [Matrix](#)< double > [NumJac](#)
- [Matrix](#)< double > [logC](#)
- [Matrix](#)< double > [C](#)

### 4.45.1 Member Data Documentation

4.45.1.1 int Speciation\_Test01\_Data::N = 4

4.45.1.2 const double Speciation\_Test01\_Data::logKw = -14.0

4.45.1.3 const double Speciation\_Test01\_Data::logKa1 = -6.35



4.45.1.4 `const double Speciation_Test01_Data::logKa2 = -10.33`

4.45.1.5 `double Speciation_Test01_Data::CT = 0.1786`

4.45.1.6 `double Speciation_Test01_Data::NaT = 0.1786`

4.45.1.7 `std::vector<Molecule> Speciation_Test01_Data::x`

4.45.1.8 `Matrix<double> Speciation_Test01_Data::Jacobian`

4.45.1.9 `Matrix<double> Speciation_Test01_Data::NumJac`

4.45.1.10 `Matrix<double> Speciation_Test01_Data::logC`

4.45.1.11 `Matrix<double> Speciation_Test01_Data::C`

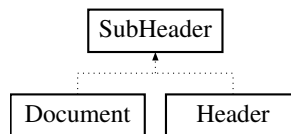
The documentation for this struct was generated from the following file:

- [sandbox.h](#)

## 4.46 SubHeader Class Reference

```
#include <yaml_wrapper.h>
```

Inheritance diagram for SubHeader:



### Public Member Functions

- [SubHeader](#) ()
- [~SubHeader](#) ()
- [SubHeader](#) (const [SubHeader](#) &subheader)
- [SubHeader](#) (const [KeyValueMap](#) &map)
- [SubHeader](#) (std::string [name](#))
- [SubHeader](#) (std::string [name](#), const [KeyValueMap](#) &map)
- [SubHeader](#) & [operator=](#) (const [SubHeader](#) &sub)
- [ValueTypePair](#) & [operator\[\]](#) (const std::string key)
- [ValueTypePair](#) [operator\[\]](#) (const std::string key) const
- [KeyValueMap](#) & [getMap](#) ()
- void [clear](#) ()
- void [addPair](#) (std::string key, std::string val)
- void [addPair](#) (std::string key, std::string val, int type)
- void [setName](#) (std::string [name](#))
- void [setAlias](#) (std::string [alias](#))
- void [setAlias](#) (std::string [alias](#), int [state](#))
- void [setNameAliasPair](#) (std::string [name](#), std::string [alias](#), int [state](#))
- void [setState](#) (int [state](#))
- void [DisplayContents](#) ()
- std::string [getName](#) ()

- std::string [getAlias](#) ()
- bool [isAlias](#) ()
- bool [isAnchor](#) ()
- int [getState](#) ()

### Protected Attributes

- [KeyValueMap Data\\_Map](#)
- std::string [name](#)
- std::string [alias](#)
- int [state](#)

## 4.46.1 Constructor & Destructor Documentation

4.46.1.1 SubHeader::SubHeader ( )

4.46.1.2 SubHeader::~~SubHeader ( )

4.46.1.3 SubHeader::SubHeader ( const SubHeader & *subheader* )

4.46.1.4 SubHeader::SubHeader ( const KeyValueMap & *map* )

4.46.1.5 SubHeader::SubHeader ( std::string *name* )

4.46.1.6 SubHeader::SubHeader ( std::string *name*, const KeyValueMap & *map* )

## 4.46.2 Member Function Documentation

4.46.2.1 SubHeader& SubHeader::operator= ( const SubHeader & *sub* )

4.46.2.2 ValueTypePair& SubHeader::operator[] ( const std::string *key* )

4.46.2.3 ValueTypePair SubHeader::operator[] ( const std::string *key* ) const

4.46.2.4 KeyValueMap& SubHeader::getMap ( )

4.46.2.5 void SubHeader::clear ( )

4.46.2.6 void SubHeader::addPair ( std::string *key*, std::string *val* )

4.46.2.7 void SubHeader::addPair ( std::string *key*, std::string *val*, int *type* )

4.46.2.8 void SubHeader::setName ( std::string *name* )

4.46.2.9 void SubHeader::setAlias ( std::string *alias* )

4.46.2.10 void SubHeader::setAlias ( std::string *alias*, int *state* )

4.46.2.11 void SubHeader::setNameAliasPair ( std::string *name*, std::string *alias*, int *state* )

4.46.2.12 void SubHeader::setState ( int *state* )

4.46.2.13 void SubHeader::DisplayContents ( )

4.46.2.14 `std::string SubHeader::getName ( )`

4.46.2.15 `std::string SubHeader::getAlias ( )`

4.46.2.16 `bool SubHeader::isAlias ( )`

4.46.2.17 `bool SubHeader::isAnchor ( )`

4.46.2.18 `int SubHeader::getState ( )`

### 4.46.3 Member Data Documentation

4.46.3.1 `KeyValueMap SubHeader::Data_Map` [protected]

4.46.3.2 `std::string SubHeader::name` [protected]

4.46.3.3 `std::string SubHeader::alias` [protected]

4.46.3.4 `int SubHeader::state` [protected]

The documentation for this class was generated from the following file:

- [yaml\\_wrapper.h](#)

## 4.47 SYSTEM\_DATA Struct Reference

System Data Structure.

```
#include <magpie.h>
```

### Public Attributes

- double [T](#)  
*System Temperature (K)*
- double [PT](#)  
*Total Pressure (kPa)*
- double [qT](#)  
*Total Amount adsorbed (mol/kg)*
- double [PI](#)  
*Total Lumped Spreading Pressure (mol/kg)*
- double [pi](#)  
*Actual Spreading pressure (J/m<sup>2</sup>)*
- double [As](#)  
*Specific surface area of adsorbent (m<sup>2</sup>/kg)*
- int [N](#)  
*Total Number of Components.*
- int [I](#)
- int [J](#)
- int [K](#)  
*Special indices used to keep track of sub-systems.*
- unsigned long int [total\\_eval](#)  
*Counter to keep track of total number of non-linear steps.*

- double [avg\\_norm](#)  
*Used to store all norms from evaluations then average at end of run.*
- double [max\\_norm](#)  
*Used to store the maximum e.norm calculated from non-linear iterations.*
- int [Sys](#)  
*Number of sub-systems to solve.*
- int [Par](#)  
*Number of binary parameters to solve for.*
- bool [Recover](#)  
*If Recover == false, standard GPAST using y's as knowns.*
- bool [Carrier](#)  
*If there is an inert carrier gas, Carrier == true.*
- bool [Ideal](#)  
*If the behavior of the system is determined to be ideal, then Ideal == true.*
- bool [Output](#)  
*Boolean to suppress output if desired (true = display, false = no display).*

#### 4.47.1 Detailed Description

System Data Structure.

C-style object holding all the data associated with the overall system to be modeled.

#### 4.47.2 Member Data Documentation

##### 4.47.2.1 double SYSTEM\_DATA::T

System Temperature (K)

##### 4.47.2.2 double SYSTEM\_DATA::PT

Total Pressure (kPa)

##### 4.47.2.3 double SYSTEM\_DATA::qT

Total Amount adsorbed (mol/kg)

##### 4.47.2.4 double SYSTEM\_DATA::PI

Total Lumped Spreading Pressure (mol/kg)

##### 4.47.2.5 double SYSTEM\_DATA::pi

Actual Spreading pressure ( $\text{J/m}^2$ )

##### 4.47.2.6 double SYSTEM\_DATA::As

Specific surface area of adsorbent ( $\text{m}^2/\text{kg}$ )

**4.47.2.7 int SYSTEM\_DATA::N**

Total Number of Components.

**4.47.2.8 int SYSTEM\_DATA::I****4.47.2.9 int SYSTEM\_DATA::J****4.47.2.10 int SYSTEM\_DATA::K**

Special indices used to keep track of sub-systems.

**4.47.2.11 unsigned long int SYSTEM\_DATA::total\_eval**

Counter to keep track of total number of non-linear steps.

**4.47.2.12 double SYSTEM\_DATA::avg\_norm**

Used to store all norms from evaluations then average at end of run.

**4.47.2.13 double SYSTEM\_DATA::max\_norm**

Used to store the maximum e.norm calculated from non-linear iterations.

**4.47.2.14 int SYSTEM\_DATA::Sys**

Number of sub-systems to solve.

**4.47.2.15 int SYSTEM\_DATA::Par**

Number of binary parameters to solve for.

**4.47.2.16 bool SYSTEM\_DATA::Recover**

If Recover == false, standard GPAST using y's as knowns.

**4.47.2.17 bool SYSTEM\_DATA::Carrier**

If there is an inert carrier gas, Carrier == true.

**4.47.2.18 bool SYSTEM\_DATA::Ideal**

If the behavior of the system is determined to be ideal, then Ideal == true.

**4.47.2.19 bool SYSTEM\_DATA::Output**

Boolean to suppress output if desired (true = display, false = no display).

The documentation for this struct was generated from the following file:

- [magpie.h](#)

## 4.48 TRAJECTORY\_DATA Struct Reference

```
#include <Trajectory.h>
```

### Public Attributes

- double [mu\\_0](#) = 12.57e-7
- double [rho\\_f](#) = 1000.0
- double [eta](#) = 0.001
- double [Hamaker](#) = 1.3e-21
- double [Temp](#) = 298
- double [k](#) = 1.38e-23
- double [Rs](#) = 0.0026925
- double [L](#) = 0.0611
- double [porosity](#) = 0.8979
- double [V\\_separator](#)
- double [a](#) = 33.0e-6
- double [V\\_wire](#)
- double [L\\_wire](#)
- double [A\\_separator](#)
- double [A\\_wire](#)
- double [B0](#) = 1.0
- double [H0](#)
- double [Ms](#) = 0.6
- double [b](#) = 0.25e-6
- double [chi\\_p](#) = 3.87e-6
- double [rho\\_p](#) = 8700.0
- double [Q\\_in](#)
- double [V0](#)
- double [Y\\_initial](#) = 20.0
- double [dt](#)
- double [M](#)
- double [mp](#)
- double [beta](#)
- double [q\\_bar](#)
- double [sigma\\_v](#)
- double [sigma\\_vz](#)
- double [sigma\\_z](#)
- double [sigma\\_n](#)
- double [sigma\\_m](#)
- double [n\\_rand](#)
- double [m\\_rand](#)
- double [s\\_rand](#)
- double [t\\_rand](#)
- [Matrix](#)< double > [POL](#)
- [Matrix](#)< double > [H](#)
- [Matrix](#)< double > [dX](#)
- [Matrix](#)< double > [dY](#)
- [Matrix](#)< double > [X](#)
- [Matrix](#)< double > [Y](#)
- [Matrix](#)< int > [Cap](#)

#### 4.48.1 Member Data Documentation

- 4.48.1.1 double TRAJECTORY\_DATA::mu\_0 = 12.57e-7
- 4.48.1.2 double TRAJECTORY\_DATA::rho\_f = 1000.0
- 4.48.1.3 double TRAJECTORY\_DATA::eta = 0.001
- 4.48.1.4 double TRAJECTORY\_DATA::Hamaker = 1.3e-21
- 4.48.1.5 double TRAJECTORY\_DATA::Temp = 298
- 4.48.1.6 double TRAJECTORY\_DATA::k = 1.38e-23
- 4.48.1.7 double TRAJECTORY\_DATA::Rs = 0.0026925
- 4.48.1.8 double TRAJECTORY\_DATA::L = 0.0611
- 4.48.1.9 double TRAJECTORY\_DATA::porosity = 0.8979
- 4.48.1.10 double TRAJECTORY\_DATA::V\_separator
- 4.48.1.11 double TRAJECTORY\_DATA::a = 33.0e-6
- 4.48.1.12 double TRAJECTORY\_DATA::V\_wire
- 4.48.1.13 double TRAJECTORY\_DATA::L\_wire
- 4.48.1.14 double TRAJECTORY\_DATA::A\_separator
- 4.48.1.15 double TRAJECTORY\_DATA::A\_wire
- 4.48.1.16 double TRAJECTORY\_DATA::B0 = 1.0
- 4.48.1.17 double TRAJECTORY\_DATA::H0
- 4.48.1.18 double TRAJECTORY\_DATA::Ms = 0.6
- 4.48.1.19 double TRAJECTORY\_DATA::b = 0.25e-6
- 4.48.1.20 double TRAJECTORY\_DATA::chi\_p = 3.87e-6
- 4.48.1.21 double TRAJECTORY\_DATA::rho\_p = 8700.0
- 4.48.1.22 double TRAJECTORY\_DATA::Q\_in
- 4.48.1.23 double TRAJECTORY\_DATA::V0
- 4.48.1.24 double TRAJECTORY\_DATA::Y\_initial = 20.0
- 4.48.1.25 double TRAJECTORY\_DATA::dt
- 4.48.1.26 double TRAJECTORY\_DATA::M
- 4.48.1.27 double TRAJECTORY\_DATA::mp

- 4.48.1.28 double TRAJECTORY\_DATA::beta
- 4.48.1.29 double TRAJECTORY\_DATA::q\_bar
- 4.48.1.30 double TRAJECTORY\_DATA::sigma\_v
- 4.48.1.31 double TRAJECTORY\_DATA::sigma\_vz
- 4.48.1.32 double TRAJECTORY\_DATA::sigma\_z
- 4.48.1.33 double TRAJECTORY\_DATA::sigma\_n
- 4.48.1.34 double TRAJECTORY\_DATA::sigma\_m
- 4.48.1.35 double TRAJECTORY\_DATA::n\_rand
- 4.48.1.36 double TRAJECTORY\_DATA::m\_rand
- 4.48.1.37 double TRAJECTORY\_DATA::s\_rand
- 4.48.1.38 double TRAJECTORY\_DATA::t\_rand
- 4.48.1.39 Matrix<double> TRAJECTORY\_DATA::POL
- 4.48.1.40 Matrix<double> TRAJECTORY\_DATA::H
- 4.48.1.41 Matrix<double> TRAJECTORY\_DATA::dX
- 4.48.1.42 Matrix<double> TRAJECTORY\_DATA::dY
- 4.48.1.43 Matrix<double> TRAJECTORY\_DATA::X
- 4.48.1.44 Matrix<double> TRAJECTORY\_DATA::Y
- 4.48.1.45 Matrix<int> TRAJECTORY\_DATA::Cap

The documentation for this struct was generated from the following file:

- [Trajectory.h](#)

## 4.49 UI\_DATA Struct Reference

Data structure holding the UI arguments.

```
#include <ui.h>
```

### Public Attributes

- [ValueTypePair](#) [value\\_type](#)  
*Data pair for input, tells what the input is and it's type.*
- std::vector< std::string > [user\\_input](#)  
*What is read in from the console at any point.*
- std::vector< std::string > [input\\_files](#)  
*A vector of input file names and directories given by user.*



- `std::string path`  
*Path to where input files are located.*
- `int count = 0`  
*Number of times a questing has been asked.*
- `int max = 3`  
*Maximum allowable recursions of a question.*
- `int option`  
*Current option choosen by the user.*
- `bool Path = false`  
*True if user gives path as an option.*
- `bool Files = false`  
*True if user gives input files as an option.*
- `bool MissingArg = true`  
*True if an input argument is missing; False if everything is ok.*
- `bool BasicUI = true`  
*True if using Basic UI; False if using Advanced UI.*
- `int argc`  
*Number of console arguments given on input.*
- `const char * argv []`  
*Actual console arguments given at execution.*

#### 4.49.1 Detailed Description

Data structure holding the UI arguments.

C-Style object for interfacing with users request upon execution of the program. User input is stored in objects below and a series of booleans is used to determine how and what to execute.

#### 4.49.2 Member Data Documentation

##### 4.49.2.1 `ValueTypePair UI_DATA::value_type`

Data pair for input, tells what the input is and it's type.

##### 4.49.2.2 `std::vector<std::string> UI_DATA::user_input`

What is read in from the console at any point.

##### 4.49.2.3 `std::vector<std::string> UI_DATA::input_files`

A vector of input file names and directories given by user.

##### 4.49.2.4 `std::string UI_DATA::path`

Path to where input files are located.

##### 4.49.2.5 `int UI_DATA::count = 0`

Number of times a questing has been asked.

#### 4.49.2.6 `int UI_DATA::max = 3`

Maximum allowable recursions of a question.

#### 4.49.2.7 `int UI_DATA::option`

Current option choosen by the user.

#### 4.49.2.8 `bool UI_DATA::Path = false`

True if user gives path as an option.

#### 4.49.2.9 `bool UI_DATA::Files = false`

True if user gives input files as an option.

#### 4.49.2.10 `bool UI_DATA::MissingArg = true`

True if an input argument is missing; False if everything is ok.

#### 4.49.2.11 `bool UI_DATA::BasicUI = true`

True if using Basic UI; False if using Advanced UI.

#### 4.49.2.12 `int UI_DATA::argc`

Number of console arguments given on input.

#### 4.49.2.13 `const char* UI_DATA::argv[]`

Actual console arguments given at execution.

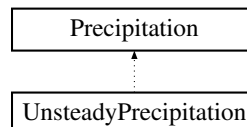
The documentation for this struct was generated from the following file:

- [ui.h](#)

## 4.50 UnsteadyPrecipitation Class Reference

```
#include <shark.h>
```

Inheritance diagram for UnsteadyPrecipitation:



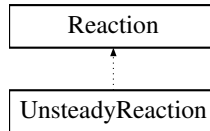
The documentation for this class was generated from the following file:

- [shark.h](#)

## 4.51 UnsteadyReaction Class Reference

```
#include <shark.h>
```

Inheritance diagram for UnsteadyReaction:



### Public Member Functions

- [UnsteadyReaction](#) ()
- [~UnsteadyReaction](#) ()
- void [Initialize\\_List](#) ([MasterSpeciesList](#) &[List](#))
- void [Display\\_Info](#) ()
- void [Set\\_Species\\_Index](#) (int i)
- void [Set\\_Species\\_Index](#) (std::string formula)
- void [Set\\_Stoichiometric](#) (int i, double v)
- void [Set\\_Equilibrium](#) (double v)
- void [Set\\_Enthalpy](#) (double H)
- void [Set\\_Entropy](#) (double S)
- void [Set\\_EnthalpyANDEntropy](#) (double H, double S)
- void [Set\\_Energy](#) (double G)
- void [Set\\_InitialValue](#) (double ic)
- void [Set\\_MaximumValue](#) (double max)
- void [Set\\_Forward](#) (double forward)
- void [Set\\_Reverse](#) (double reverse)
- void [Set\\_ForwardRef](#) (double Fref)
- void [Set\\_ReverseRef](#) (double Rref)
- void [Set\\_ActivationEnergy](#) (double E)
- void [Set\\_Affinity](#) (double b)
- void [Set\\_TimeStep](#) (double dt)
- void [checkSpeciesEnergies](#) ()
- void [calculateEnergies](#) ()
- void [calculateEquilibrium](#) (double T)
- void [calculateRate](#) (double T)
- bool [haveEquilibrium](#) ()
- bool [haveRate](#) ()
- int [Get\\_Species\\_Index](#) ()
- double [Get\\_Stoichiometric](#) (int i)
- double [Get\\_Equilibrium](#) ()
- double [Get\\_Enthalpy](#) ()
- double [Get\\_Entropy](#) ()
- double [Get\\_Energy](#) ()
- double [Get\\_InitialValue](#) ()
- double [Get\\_MaximumValue](#) ()
- double [Get\\_Forward](#) ()
- double [Get\\_Reverse](#) ()
- double [Get\\_ForwardRef](#) ()
- double [Get\\_ReverseRef](#) ()
- double [Get\\_ActivationEnergy](#) ()

- double [Get\\_Affinity](#) ()
- double [Get\\_TimeStep](#) ()
- double [Eval\\_ReactionRate](#) (const [Matrix](#)< double > &x, const [Matrix](#)< double > &gama)
- double [Eval\\_Residual](#) (const [Matrix](#)< double > &x\_new, const [Matrix](#)< double > &x\_old, const [Matrix](#)< double > &gama\_new, const [Matrix](#)< double > &gama\_old)
- double [Eval\\_Residual](#) (const [Matrix](#)< double > &x, const [Matrix](#)< double > &gama)
- double [Eval\\_IC\\_Residual](#) (const [Matrix](#)< double > &x)
- double [Explicit\\_Eval](#) (const [Matrix](#)< double > &x, const [Matrix](#)< double > &gama)

### Protected Attributes

- double [initial\\_value](#)
- double [max\\_value](#)
- double [forward\\_rate](#)
- double [reverse\\_rate](#)
- double [forward\\_ref\\_rate](#)
- double [reverse\\_ref\\_rate](#)
- double [activation\\_energy](#)
- double [temperature\\_affinity](#)
- double [time\\_step](#)
- bool [HaveForward](#)
- bool [HaveReverse](#)
- bool [HaveForRef](#)
- bool [HaveRevRef](#)
- int [species\\_index](#)

### Additional Inherited Members

#### 4.51.1 Constructor & Destructor Documentation

4.51.1.1 [UnsteadyReaction::UnsteadyReaction](#) ( )

4.51.1.2 [UnsteadyReaction::~~UnsteadyReaction](#) ( )

#### 4.51.2 Member Function Documentation

4.51.2.1 [void UnsteadyReaction::Initialize\\_List](#) ( [MasterSpeciesList](#) & *List* )

4.51.2.2 [void UnsteadyReaction::Display\\_Info](#) ( )

4.51.2.3 [void UnsteadyReaction::Set\\_Species\\_Index](#) ( int *i* )

4.51.2.4 [void UnsteadyReaction::Set\\_Species\\_Index](#) ( std::string *formula* )

4.51.2.5 [void UnsteadyReaction::Set\\_Stoichiometric](#) ( int *i*, double *v* )

4.51.2.6 [void UnsteadyReaction::Set\\_Equilibrium](#) ( double *v* )

4.51.2.7 [void UnsteadyReaction::Set\\_Enthalpy](#) ( double *H* )

4.51.2.8 [void UnsteadyReaction::Set\\_Entropy](#) ( double *S* )

4.51.2.9 [void UnsteadyReaction::Set\\_EnthalpyANDEntropy](#) ( double *H*, double *S* )

- 4.51.2.10 void UnsteadyReaction::Set\_Energy ( double *G* )
- 4.51.2.11 void UnsteadyReaction::Set\_InitialValue ( double *ic* )
- 4.51.2.12 void UnsteadyReaction::Set\_MaximumValue ( double *max* )
- 4.51.2.13 void UnsteadyReaction::Set\_Forward ( double *forward* )
- 4.51.2.14 void UnsteadyReaction::Set\_Reverse ( double *reverse* )
- 4.51.2.15 void UnsteadyReaction::Set\_ForwardRef ( double *Fref* )
- 4.51.2.16 void UnsteadyReaction::Set\_ReverseRef ( double *Rref* )
- 4.51.2.17 void UnsteadyReaction::Set\_ActivationEnergy ( double *E* )
- 4.51.2.18 void UnsteadyReaction::Set\_Affinity ( double *b* )
- 4.51.2.19 void UnsteadyReaction::Set\_TimeStep ( double *dt* )
- 4.51.2.20 void UnsteadyReaction::checkSpeciesEnergies ( )
- 4.51.2.21 void UnsteadyReaction::calculateEnergies ( )
- 4.51.2.22 void UnsteadyReaction::calculateEquilibrium ( double *T* )
- 4.51.2.23 void UnsteadyReaction::calculateRate ( double *T* )
- 4.51.2.24 bool UnsteadyReaction::haveEquilibrium ( )
- 4.51.2.25 bool UnsteadyReaction::haveRate ( )
- 4.51.2.26 int UnsteadyReaction::Get\_Species\_Index ( )
- 4.51.2.27 double UnsteadyReaction::Get\_Stoichiometric ( int *i* )
- 4.51.2.28 double UnsteadyReaction::Get\_Equilibrium ( )
- 4.51.2.29 double UnsteadyReaction::Get\_Enthalpy ( )
- 4.51.2.30 double UnsteadyReaction::Get\_Entropy ( )
- 4.51.2.31 double UnsteadyReaction::Get\_Energy ( )
- 4.51.2.32 double UnsteadyReaction::Get\_InitialValue ( )
- 4.51.2.33 double UnsteadyReaction::Get\_MaximumValue ( )
- 4.51.2.34 double UnsteadyReaction::Get\_Forward ( )
- 4.51.2.35 double UnsteadyReaction::Get\_Reverse ( )
- 4.51.2.36 double UnsteadyReaction::Get\_ForwardRef ( )
- 4.51.2.37 double UnsteadyReaction::Get\_ReverseRef ( )

- 4.51.2.38 `double UnsteadyReaction::Get_ActivationEnergy ( )`
- 4.51.2.39 `double UnsteadyReaction::Get_Affinity ( )`
- 4.51.2.40 `double UnsteadyReaction::Get_TimeStep ( )`
- 4.51.2.41 `double UnsteadyReaction::Eval_ReactionRate ( const Matrix< double > & x, const Matrix< double > & gama )`
- 4.51.2.42 `double UnsteadyReaction::Eval_Residual ( const Matrix< double > & x_new, const Matrix< double > & x_old, const Matrix< double > & gama_new, const Matrix< double > & gama_old )`
- 4.51.2.43 `double UnsteadyReaction::Eval_Residual ( const Matrix< double > & x, const Matrix< double > & gama )`
- 4.51.2.44 `double UnsteadyReaction::Eval_IC_Residual ( const Matrix< double > & x )`
- 4.51.2.45 `double UnsteadyReaction::Explicit_Eval ( const Matrix< double > & x, const Matrix< double > & gama )`

### 4.51.3 Member Data Documentation

- 4.51.3.1 `double UnsteadyReaction::initial_value` [protected]
- 4.51.3.2 `double UnsteadyReaction::max_value` [protected]
- 4.51.3.3 `double UnsteadyReaction::forward_rate` [protected]
- 4.51.3.4 `double UnsteadyReaction::reverse_rate` [protected]
- 4.51.3.5 `double UnsteadyReaction::forward_ref_rate` [protected]
- 4.51.3.6 `double UnsteadyReaction::reverse_ref_rate` [protected]
- 4.51.3.7 `double UnsteadyReaction::activation_energy` [protected]
- 4.51.3.8 `double UnsteadyReaction::temperature_affinity` [protected]
- 4.51.3.9 `double UnsteadyReaction::time_step` [protected]
- 4.51.3.10 `bool UnsteadyReaction::HaveForward` [protected]
- 4.51.3.11 `bool UnsteadyReaction::HaveReverse` [protected]
- 4.51.3.12 `bool UnsteadyReaction::HaveForRef` [protected]
- 4.51.3.13 `bool UnsteadyReaction::HaveRevRef` [protected]
- 4.51.3.14 `int UnsteadyReaction::species_index` [protected]

The documentation for this class was generated from the following file:

- [shark.h](#)

## 4.52 ValueTypePair Class Reference

```
#include <yaml_wrapper.h>
```

### Public Member Functions

- [ValueTypePair](#) ()
- [~ValueTypePair](#) ()
- [ValueTypePair](#) (const std::pair< std::string, int > &vt)
- [ValueTypePair](#) (std::string value, int [type](#))
- [ValueTypePair](#) (const [ValueTypePair](#) &vt)
- [ValueTypePair](#) & [operator=](#) (const [ValueTypePair](#) &vt)
- void [editValue](#) (std::string value)
- void [editPair](#) (std::string value, int [type](#))
- void [findType](#) ()
- void [assertType](#) (int [type](#))
- void [DisplayPair](#) ()
- std::string [getString](#) ()
- bool [getBool](#) ()
- double [getDouble](#) ()
- int [getInt](#) ()
- std::string [getValue](#) ()
- int [getType](#) ()
- std::pair< std::string, int > & [getPair](#) ()

### Private Attributes

- std::pair< std::string, int > [Value\\_Type](#)
- int [type](#)

## 4.52.1 Constructor & Destructor Documentation

4.52.1.1 [ValueTypePair::ValueTypePair](#) ( )

4.52.1.2 [ValueTypePair::~~ValueTypePair](#) ( )

4.52.1.3 [ValueTypePair::ValueTypePair](#) ( const std::pair< std::string, int > & vt )

4.52.1.4 [ValueTypePair::ValueTypePair](#) ( std::string value, int type )

4.52.1.5 [ValueTypePair::ValueTypePair](#) ( const [ValueTypePair](#) & vt )

## 4.52.2 Member Function Documentation

4.52.2.1 [ValueTypePair& ValueTypePair::operator=](#) ( const [ValueTypePair](#) & vt )

4.52.2.2 void [ValueTypePair::editValue](#) ( std::string value )

4.52.2.3 void [ValueTypePair::editPair](#) ( std::string value, int type )

4.52.2.4 void [ValueTypePair::findType](#) ( )

4.52.2.5 void [ValueTypePair::assertType](#) ( int type )

4.52.2.6 void [ValueTypePair::DisplayPair](#) ( )

4.52.2.7 std::string [ValueTypePair::getString](#) ( )

4.52.2.8 `bool ValueTypePair::getBool ( )`

4.52.2.9 `double ValueTypePair::getDouble ( )`

4.52.2.10 `int ValueTypePair::getInt ( )`

4.52.2.11 `std::string ValueTypePair::getValue ( )`

4.52.2.12 `int ValueTypePair::getType ( )`

4.52.2.13 `std::pair<std::string,int>& ValueTypePair::getPair ( )`

### 4.52.3 Member Data Documentation

4.52.3.1 `std::pair<std::string,int> ValueTypePair::Value_Type` [private]

4.52.3.2 `int ValueTypePair::type` [private]

The documentation for this class was generated from the following file:

- [yaml\\_wrapper.h](#)

## 4.53 yaml\_cpp\_class Class Reference

```
#include <yaml_wrapper.h>
```

### Public Member Functions

- [yaml\\_cpp\\_class](#) ()
- [~yaml\\_cpp\\_class](#) ()
- `int setInputFile (const char *file)`
- `int readInputFile ()`
- `int cleanup ()`
- `int executeYamlRead (const char *file)`
- [YamlWrapper](#) & [getYamlWrapper](#) ()
- `void DisplayContents ()`

### Private Attributes

- [YamlWrapper](#) [yaml\\_wrapper](#)
- `FILE * input_file`
- `const char * file_name`
- `yaml_parser_t token_parser`
- `yaml_token_t current_token`
- `yaml_token_t previous_token`

### 4.53.1 Constructor & Destructor Documentation

4.53.1.1 `yaml_cpp_class::yaml_cpp_class ( )`

4.53.1.2 `yaml_cpp_class::~~yaml_cpp_class ( )`



### 4.53.2 Member Function Documentation

- 4.53.2.1 `int yaml_cpp_class::setInputFile ( const char * file )`
- 4.53.2.2 `int yaml_cpp_class::readInputFile ( )`
- 4.53.2.3 `int yaml_cpp_class::cleanup ( )`
- 4.53.2.4 `int yaml_cpp_class::executeYamlRead ( const char * file )`
- 4.53.2.5 `YamlWrapper& yaml_cpp_class::getYamlWrapper ( )`
- 4.53.2.6 `void yaml_cpp_class::DisplayContents ( )`

### 4.53.3 Member Data Documentation

- 4.53.3.1 `YamlWrapper yaml_cpp_class::yaml_wrapper` [private]
- 4.53.3.2 `FILE* yaml_cpp_class::input_file` [private]
- 4.53.3.3 `const char* yaml_cpp_class::file_name` [private]
- 4.53.3.4 `yaml_parser_t yaml_cpp_class::token_parser` [private]
- 4.53.3.5 `yaml_token_t yaml_cpp_class::current_token` [private]
- 4.53.3.6 `yaml_token_t yaml_cpp_class::previous_token` [private]

The documentation for this class was generated from the following file:

- [yaml\\_wrapper.h](#)

## 4.54 YamlWrapper Class Reference

```
#include <yaml_wrapper.h>
```

### Public Member Functions

- [YamlWrapper](#) ()
- [~YamlWrapper](#) ()
- [YamlWrapper](#) (const [YamlWrapper](#) &yaml)
- [YamlWrapper](#) (std::string key, const [Document](#) &doc)
- [YamlWrapper](#) & [operator=](#) (const [YamlWrapper](#) &yaml)
- [Document](#) & [operator\(\)](#) (const std::string key)
- [Document](#) [operator\(\)](#) (const std::string key) const
- std::map< std::string, [Document](#) > & [getDocMap](#) ()
- [Document](#) & [getDocument](#) (std::string key)
- std::map< std::string, [Document](#) >::const\_iterator [end](#) () const
- std::map< std::string, [Document](#) >::iterator [end](#) ()
- std::map< std::string, [Document](#) >::const\_iterator [begin](#) () const

- `std::map< std::string, Document >::iterator begin ()`
- `void clear ()`
- `void resetKeys ()`
- `void changeKey (std::string oldKey, std::string newKey)`
- `void revalidateAllKeys ()`
- `void DisplayContents ()`
- `void addDocKey (std::string key)`
- `void copyAnchor2Alias (std::string alias, Document &ref)`
- `int size ()`
- `Document & getAnchoredDoc (std::string alias)`
- `Document & getDocFromHeadAlias (std::string alias)`
- `Document & getDocFromSubAlias (std::string alias)`

### Private Attributes

- `std::map< std::string, Document > Doc_Map`

## 4.54.1 Constructor & Destructor Documentation

4.54.1.1 `YamlWrapper::YamlWrapper ( )`

4.54.1.2 `YamlWrapper::~YamlWrapper ( )`

4.54.1.3 `YamlWrapper::YamlWrapper ( const YamlWrapper & yaml )`

4.54.1.4 `YamlWrapper::YamlWrapper ( std::string key, const Document & doc )`

## 4.54.2 Member Function Documentation

4.54.2.1 `YamlWrapper& YamlWrapper::operator= ( const YamlWrapper & yaml )`

4.54.2.2 `Document& YamlWrapper::operator() ( const std::string key )`

4.54.2.3 `Document YamlWrapper::operator() ( const std::string key ) const`

4.54.2.4 `std::map<std::string, Document>& YamlWrapper::getDocMap ( )`

4.54.2.5 `Document& YamlWrapper::getDocument ( std::string key )`

4.54.2.6 `std::map<std::string, Document>::const_iterator YamlWrapper::end ( ) const`

4.54.2.7 `std::map<std::string, Document>::iterator YamlWrapper::end ( )`

4.54.2.8 `std::map<std::string, Document>::const_iterator YamlWrapper::begin ( ) const`

4.54.2.9 `std::map<std::string, Document>::iterator YamlWrapper::begin ( )`

4.54.2.10 `void YamlWrapper::clear ( )`

4.54.2.11 `void YamlWrapper::resetKeys ( )`

4.54.2.12 `void YamlWrapper::changeKey ( std::string oldKey, std::string newKey )`

- 4.54.2.13 void YamlWrapper::revalidateAllKeys ( )
- 4.54.2.14 void YamlWrapper::DisplayContents ( )
- 4.54.2.15 void YamlWrapper::addDocKey ( std::string *key* )
- 4.54.2.16 void YamlWrapper::copyAnchor2Alias ( std::string *alias*, Document & *ref* )
- 4.54.2.17 int YamlWrapper::size ( )
- 4.54.2.18 Document& YamlWrapper::getAnchoredDoc ( std::string *alias* )
- 4.54.2.19 Document& YamlWrapper::getDocFromHeadAlias ( std::string *alias* )
- 4.54.2.20 Document& YamlWrapper::getDocFromSubAlias ( std::string *alias* )

#### 4.54.3 Member Data Documentation

- 4.54.3.1 std::map<std::string, Document> YamlWrapper::Doc\_Map [private]

The documentation for this class was generated from the following file:

- [yaml\\_wrapper.h](#)



## Chapter 5

# File Documentation

### 5.1 dogfish.h File Reference

Diffusion Object Governing Fiber Interior Sorption History.

```
#include "finch.h"  
#include "mola.h"
```

#### Classes

- struct [DOGFISH\\_PARAM](#)  
*Data structure for species-specific parameters.*
- struct [DOGFISH\\_DATA](#)  
*Primary data structure for running the DOGFISH application.*

#### Functions

- void [print2file\\_species\\_header](#) (FILE \*Output, [DOGFISH\\_DATA](#) \*dog\_dat, int i)  
*Function to print a species based header for the output file.*
- void [print2file\\_DOGFISH\\_header](#) ([DOGFISH\\_DATA](#) \*dog\_dat)  
*Function to print a time and space header for the output file.*
- void [print2file\\_DOGFISH\\_result\\_old](#) ([DOGFISH\\_DATA](#) \*dog\_dat)  
*Function to print out the old time results for the output file.*
- void [print2file\\_DOGFISH\\_result\\_new](#) ([DOGFISH\\_DATA](#) \*dog\_dat)  
*Function to print out the new time results for the output file.*
- double [default\\_Retardation](#) (int i, int l, const void \*data)  
*Default function for the retardation coefficient.*
- double [default\\_IntraDiffusion](#) (int i, int l, const void \*data)  
*Default function for the intraparticle diffusion coefficient.*
- double [default\\_FilmMTCoeff](#) (int i, const void \*data)  
*Default function for the film mass transfer coefficient.*
- double [default\\_SurfaceConcentration](#) (int i, const void \*data)  
*Default function for the fiber surface concentration.*
- int [setup\\_DOGFISH\\_DATA](#) (FILE \*file, double(\*eval\_R)(int i, int l, const void \*user\_data), double(\*eval\_DI)(int i, int l, const void \*user\_data), double(\*eval\_kf)(int i, const void \*user\_data), double(\*eval\_qs)(int i, const void \*user\_data), const void \*user\_data, [DOGFISH\\_DATA](#) \*dog\_dat)  
*Function will set up the memory and pointers for use in the DOGFISH simulations.*

- int [DOGFISH\\_Executioner](#) (DOGFISH\_DATA \*dog\_dat)  
*Function to serially call all other functions need to solve the system at one time step.*
- int [set\\_DOGFISH\\_ICs](#) (DOGFISH\_DATA \*dog\_dat)  
*Function called to evaluate the initial conditions for the time dependent problem.*
- int [set\\_DOGFISH\\_timestep](#) (DOGFISH\_DATA \*dog\_dat)  
*Function sets the time step size for the next step forward in the simulation.*
- int [DOGFISH\\_preprocesses](#) (DOGFISH\_DATA \*dog\_dat)  
*Function to perform preprocess actions to be used before calling any solver.*
- int [set\\_DOGFISH\\_params](#) (const void \*user\_data)  
*Function to calculate the values of all parameters for all species at all nodes.*
- int [DOGFISH\\_postprocesses](#) (DOGFISH\_DATA \*dog\_dat)  
*Function to perform post-solve actions such as printing out results.*
- int [DOGFISH\\_reset](#) (DOGFISH\_DATA \*dog\_dat)  
*Function to reset the matrices and vectors and prepare for next time step.*
- int [DOGFISH](#) (DOGFISH\_DATA \*dog\_dat)  
*Function performs all necessary steps to step the diffusion simulation through time.*
- int [DOGFISH\\_TESTS](#) ()  
*Running DOGFISH tests.*

### 5.1.1 Detailed Description

Diffusion Object Governing Fiber Interior Sorption History. dogfish.cpp

This set of objects and functions is used to numerically solve linear or non-linear diffusion physics of aqueous ions into cylindrical adsorbent fibers. Boundary conditions for this problem could be a film mass transfer, reaction, or dirichlet condition depending on the type of problem being solve.

#### Warning

Functions and methods in this file are still under construction.

#### Author

Austin Ladshaw

#### Date

04/09/2015

#### Copyright

This software was designed and built at the Georgia Institute of Technology by Austin Ladshaw for PhD research in the area of adsorption and surface science. Copyright (c) 2015, all rights reserved.

### 5.1.2 Function Documentation

5.1.2.1 void [print2file\\_species\\_header](#) ( FILE \* *Output*, DOGFISH\_DATA \* *dog\_dat*, int *i* )

Function to print a species based header for the output file.

5.1.2.2 void [print2file\\_DOGFISH\\_header](#) ( DOGFISH\_DATA \* *dog\_dat* )

Function to print a time and space header for the output file.

5.1.2.3 void print2file\_DOGFISH\_result\_old ( DOGFISH\_DATA \* *dog\_dat* )

Function to print out the old time results for the output file.

5.1.2.4 void print2file\_DOGFISH\_result\_new ( DOGFISH\_DATA \* *dog\_dat* )

Function to print out the new time results for the output file.

5.1.2.5 double default\_Retardation ( int *i*, int *l*, const void \* *data* )

Default function for the retardation coefficient.

The default retardation coefficient for this problem is 1.0 for all time and space. Therefore, this function will only ever return a 1.

5.1.2.6 double default\_IntraDiffusion ( int *i*, int *l*, const void \* *data* )

Default function for the intraparticle diffusion coefficient.

The default intraparticle diffusivity is to assume that each species *i* has a constant diffusivity. Therefore, this function returns the value of the parameter `intraparticle_diffusion` from the `DOGFISH_PARAM` structure for each adsorbing species *i*. Each species may have a different diffusivity.

5.1.2.7 double default\_FilmMTCoeff ( int *i*, const void \* *data* )

Default function for the film mass transfer coefficient.

The default film mass transfer coefficient will be to assume that this value is a constant for each species *i*. Therefore, this function returns the parameter value of `film_transfer_coeff` from the `DOGFISH_PARAM` structure for each adsorbing species *i*.

5.1.2.8 double default\_SurfaceConcentration ( int *i*, const void \* *data* )

Default function for the fiber surface concentration.

The default fiber surface concentration will be to assume that this value is a constant for each species *i*. Therefore, this function returns the parameter value of `surface_concentration` from the `DOGFISH_PARAM` structure for each adsorbing species *i*.

5.1.2.9 int setup\_DOGFISH\_DATA ( FILE \* *file*, double (\*)(int *i*, int *l*, const void \* *user\_data*) *eval\_R*, double (\*)(int *i*, int *l*, const void \* *user\_data*) *eval\_DI*, double (\*)(int *i*, const void \* *user\_data*) *eval\_kf*, double (\*)(int *i*, const void \* *user\_data*) *eval\_qs*, const void \* *user\_data*, DOGFISH\_DATA \* *dog\_dat* )

Function will set up the memory and pointers for use in the DOGFISH simulations.

The pointers to the output file, parameter functions, and data structures are passed into this function to setup the problem in memory. This function must always be called prior to calling any other DOGFISH routine and after the `DOGFISH_DATA` structure has been initialized.

5.1.2.10 int DOGFISH\_Executioner ( DOGFISH\_DATA \* *dog\_dat* )

Function to serially call all other functions need to solve the system at one time step.

This function will call the `DOGFISH_preprocesses` function, followed by the FINCH solver functions for each species *i*, then call the `DOGFISH_postprocesses` function. After completion, this would have solved the diffusion physics for a single time step.

#### 5.1.2.11 int set\_DOGFISH\_ICs ( DOGFISH\_DATA \* *dog\_dat* )

Function called to evaluate the initial conditions for the time dependent problem.

This function will use information in [DOGFISH\\_DATA](#) to setup the initial conditions, initial parameter values, and initial sorption averages for each species. This function always assumes a constant initial condition for the sorption of each species.

#### 5.1.2.12 int set\_DOGFISH\_timestep ( DOGFISH\_DATA \* *dog\_dat* )

Function sets the time step size for the next step forward in the simulation.

This function will set the next time step size based on the spatial discretization of the fiber. Maximum time step size is locked at 0.5 hours.

#### 5.1.2.13 int DOGFISH\_preprocesses ( DOGFISH\_DATA \* *dog\_dat* )

Function to perform preprocess actions to be used before calling any solver.

This function will call all of the parameter functions in order to establish boundary condition parameter values prior to calling the FINCH solvers.

#### 5.1.2.14 int set\_DOGFISH\_params ( const void \* *user\_data* )

Function to calculate the values of all parameters for all species at all nodes.

This function is passed to the [FINCH\\_DATA](#) data structure and set as the setparams function pointer. FINCH calls this function during it's solver routine to setup the non-linear form of the problem and solve the non-linear system.

##### Parameters

|                  |   |
|------------------|---|
| <i>user_data</i> | this is actually the <a href="#">DOGFISH_DATA</a> structure, but is passed anonymously to FINCH |
|------------------|---|

#### 5.1.2.15 int DOGFISH\_postprocesses ( DOGFISH\_DATA \* *dog\_dat* )

Function to perform post-solve actions such as printing out results.

This function increments the total\_steps counter in [DOGFISH\\_DATA](#) to keep a running total of all solver steps taken. Additionally, it prints out the results of the current time simulation to the output file.

#### 5.1.2.16 int DOGFISH\_reset ( DOGFISH\_DATA \* *dog\_dat* )

Function to reset the matrices and vectors and prepare for next time step.

This function will reset the matrix and vector information of [DOGFISH\\_DATA](#) and [FINCH\\_DATA](#) to prepare for the next simulation step in time.

#### 5.1.2.17 int DOGFISH ( DOGFISH\_DATA \* *dog\_dat* )

Function performs all necessary steps to step the diffusion simulation through time.

This function calls the initial conditions, set time step, executioner, and reset functions to step the simulation through time. It will only exit when the simulation time is reached or if an error occurs.



### 5.1.2.18 int DOGFISH.TESTS ( )

Running DOGFISH tests.

This function is called from the UI to run a test simulation of DOGFISH. Output is stored in a DOGFISH\_TestOutput.txt file in a sub-directory "output" from the directory in which the executable was called.

## 5.2 eel.h File Reference

Easy-access Element Library.

```
#include <stdio.h>
#include <math.h>
#include <iostream>
#include <fstream>
#include <stdlib.h>
#include <vector>
#include <time.h>
#include <float.h>
#include <string>
#include "error.h"
```

### Classes

- class [Atom](#)  
*Atom object to hold information about specific atoms in the periodic table (click [Atom](#) to go to function definitions)*
- class [PeriodicTable](#)  
*Class object that store a digital copy of all [Atom](#) objects.*

### Functions

- int [EEL\\_TESTS](#) ()  
*Test function to exercise the class objects and check for errors.*

### 5.2.1 Detailed Description

Easy-access Element Library. eel.cpp

This file contains two C++ objects: (i) [Atom](#) and (ii) [PeriodicTable](#).

The Atom class defines all relevant information necessary for dealing with actual atoms. However, this is not necessarily all the information that one may need for any simulation dealing with atoms. Instead, it is really just a place holder used to construct Molecules and hold oxidation state and molecular/atomic weight information.

The PeriodicTable class creates a digital version of a complete periodic table. Further development of this object can make it possible to query this structure for a particular atom upon user request.

### Warning

The [Atom](#) class is mostly complete, but the [PeriodicTable](#) object is just a place holder.

**Author**

Austin Ladshaw

**Date**

02/23/2015

**Copyright**

This software was designed and built at the Georgia Institute of Technology by Austin Ladshaw for PhD research in the area of adsorption and surface science. Copyright (c) 2015, all rights reserved.

**5.2.2 Function Documentation****5.2.2.1 int EEL\_TESTS ( )**

Test function to exercise the class objects and check for errors.

**5.3 egret.h File Reference**

Estimation of Gas-phase pRopErTies.

```
#include "macaw.h"
```

**Classes**

- struct [PURE\\_GAS](#)  
*Data structure holding all the parameters for each pure gas species.*
- struct [MIXED\\_GAS](#)  
*Data structure holding information necessary for computing mixed gas properties.*

**Macros**

- #define [Rstd](#) 8.3144621  
*Gas Constant in J/K/mol (or) L\*kPa/K/mol (Standard Units)*
- #define [RE3](#) 8.3144621E+3  
*Gas Constant in cm<sup>3</sup>\*kPa/K/mol (Convenient for density calculations)*
- #define [Po](#) 100.0  
*Standard state pressure (kPa)*
- #define [Cstd](#)(p, T) ((p)/(Rstd\*T))  
*Calculation of concentration/density from partial pressure (Cstd = mol/L)*
- #define [CE3](#)(p, T) ((p)/(RE3\*T))  
*Calculation of concentration/density from partial pressure (CE3 = mol/cm<sup>3</sup>)*
- #define [Pstd](#)(c, T) ((c)\*Rstd\*T)  
*Calculation of partial pressure from concentration/density (c = mol/L)*
- #define [PE3](#)(c, T) ((c)\*RE3\*T)  
*Calculation of partial pressure from concentration/density (c = mol/cm<sup>3</sup>)*
- #define [Nu](#)(mu, rho) ((mu)/(rho))  
*Calculation of kinematic viscosity from dynamic viscosity and density (cm<sup>2</sup>/s)*
- #define [PSI](#)(T) (0.873143 + (0.000072375\*T))

- Calculation of temperature correction factor for dynamic viscosity.*

  - #define `Dp_ij`(Dij, PT) ((PT\*Dij)/Po)
- Calculation of the corrected binary diffusivity (cm<sup>2</sup>/s)*

  - #define `D_ij`(MWi, MWj, rhoi, rhoj, mui, muj) ( (4.0 / sqrt(2.0)) \* pow(((1/MWi)+(1/MWj)),0.5) ) / pow( (pow((pow((rhoi/(1.385\*mui)),2.0)/MWi),0.25)+ pow((pow((rhoj/(1.385\*muj)),2.0)/MWj),0.25)),2.0 )
- Calculation of binary diffusion based on MW, density, and viscosity info (cm<sup>2</sup>/s)*

  - #define `Mu`(muo, To, C, T) (muo \* ((To + C)/(T + C)) \* pow((T/To),1.5) )
- Calculation of single species viscosity from Sutherland's Equ. (g/cm/s)*

  - #define `D_ii`(rhoi, mui) (1.385\*mui/rhoi)
- Calculation of self-diffusivity (cm<sup>2</sup>/s)*

  - #define `ReNum`(u, L, nu) (u\*L/nu)
- Calculation of the Reynold's Number (-)*

  - #define `ScNum`(nu, D) (nu/D)
- Calculation of the Schmidt Number (-)*

  - #define `FilmMTCoeff`(D, L, Re, Sc) ((D/L)\*(2.0 + (1.1\*pow(Re,0.6)\*pow(Sc,0.3))))
- Calculation of film mass transfer coefficient (cm/s)*

## Functions

- int `initialize_data` (int N, `MIXED_GAS` \*gas\_dat)

*Function to initialize the `MIXED_GAS` structure based on number of gas species.*
- int `set_variables` (double PT, double T, double us, double L, std::vector< double > &y, `MIXED_GAS` \*gas\_dat)

*Function to set the values of the parameters in the gas phase.*
- int `calculate_properties` (`MIXED_GAS` \*gas\_dat)

*Function to calculate the gas properties based on information in `MIXED_GAS`.*
- int `EGRET_TESTS` ()

*Function runs a series of tests for the EGRET file.*

### 5.3.1 Detailed Description

Estimation of Gas-phase pRopErTies. egret.cpp

This file is responsible for estimating various temperature, pressure, and concentration dependent parameters to be used in other models for gas phase adsorption, mass transfer, and or mass transport. The goal of this file is to eliminate redundancies in code such that the higher level programs operate more efficiently and cleanly. Calculations made here are based on kinetic theory of gases, ideal gas law, and some empirical models that were developed to account for changes in density and viscosity with changes in temperature between standard temperatures and up to 1000 K.

#### Author

Austin Ladshaw

#### Date

01/29/2015

#### Copyright

This software was designed and built at the Georgia Institute of Technology by Austin Ladshaw for PhD research in the area of adsorption and surface science. Copyright (c) 2015, all rights reserved.

### 5.3.2 Macro Definition Documentation

#### 5.3.2.1 #define Rstd 8.3144621

Gas Constant in J/K/mol (or) L\*kPa/K/mol (Standard Units)

#### 5.3.2.2 #define RE3 8.3144621E+3

Gas Constant in cm<sup>3</sup>\*kPa/K/mol (Convenient for density calculations)

#### 5.3.2.3 #define Po 100.0

Standard state pressure (kPa)

#### 5.3.2.4 #define Cstd( p, T ) ((p)/(Rstd\*T))

Calculation of concentration/density from partial pressure (Cstd = mol/L)

#### 5.3.2.5 #define CE3( p, T ) ((p)/(RE3\*T))

Calculation of concentration/density from partial pressure (CE3 = mol/cm<sup>3</sup>)

#### 5.3.2.6 #define Pstd( c, T ) ((c)\*Rstd\*T)

Calculation of partial pressure from concentration/density (c = mol/L)

#### 5.3.2.7 #define PE3( c, T ) ((c)\*RE3\*T)

Calculation of partial pressure from concentration/density (c = mol/cm<sup>3</sup>)

#### 5.3.2.8 #define Nu( mu, rho ) ((mu)/(rho))

Calculation of kinematic viscosity from dynamic viscosity and density (cm<sup>2</sup>/s)

#### 5.3.2.9 #define PSI( T ) (0.873143 + (0.000072375\*T))

Calculation of temperature correction factor for dynamic viscosity.

#### 5.3.2.10 #define Dp\_ij( Dij, PT ) ((PT\*Dij)/Po)

Calculation of the corrected binary diffusivity (cm<sup>2</sup>/s)

#### 5.3.2.11 #define D\_ij( MWi, MWj, rhoi, rhoj, muj, muj ) ( (4.0 / sqrt(2.0)) \* pow(((1/MWi)+(1/MWj)),0.5) ) / pow( (pow((pow((rhoi/(1.385\*muj)),2.0)/MWi),0.25)+ pow((pow((rhoj/(1.385\*muj)),2.0)/MWj),0.25)),2.0 )

Calculation of binary diffusion based on MW, density, and viscosity info (cm<sup>2</sup>/s)

#### 5.3.2.12 #define Mu( muo, To, C, T ) (muo \* ((To + C)/(T + C)) \* pow((T/To),1.5) )

Calculation of single species viscosity from Sutherland's Equ. (g/cm/s)

5.3.2.13 `#define D_ii( rhoi, mu ) (1.385*mu/rhoi)`

Calculation of self-diffusivity ( $\text{cm}^2/\text{s}$ )

5.3.2.14 `#define ReNum( u, L, nu ) (u*L/nu)`

Calculation of the Reynold's Number (-)

5.3.2.15 `#define ScNum( nu, D ) (nu/D)`

Calculation of the Schmidt Number (-)

5.3.2.16 `#define FilmMTCoeff( D, L, Re, Sc ) ((D/L)*(2.0 + (1.1*pow(Re,0.6)*pow(Sc,0.3))))`

Calculation of film mass transfer coefficient ( $\text{cm/s}$ )

### 5.3.3 Function Documentation

5.3.3.1 `int initialize_data ( int N, MIXED_GAS * gas_dat )`

Function to initialize the `MIXED_GAS` structure based on number of gas species.

This function will initialize the sizes of all vector objects in the `MIXED_GAS` structure based on the number of gas species indicated by N.

5.3.3.2 `int set_variables ( double PT, double T, double us, double L, std::vector< double > & y, MIXED_GAS * gas_dat )`

Function to set the values of the parameters in the gas phase.

The gas phase properties are a function of total pressure, gas temperature, gas velocity, characteristic length, and the mole fractions of each species in the gas phase. Prior to calculating the gas phase properties, these parameters must be set and updated as they change.

#### Parameters

|                |   |
|----------------|---|
| <i>PT</i>      | total gas pressure in kPa   |
| <i>T</i>       | gas temperature in K  |
| <i>us</i>      | gas velocity in cm/s  |
| <i>L</i>       | characteristic length in cm (this depends on the particular system) |
| <i>y</i>       | vector of gas mole fractions of each species in the mixture         |
| <i>gas_dat</i> | pointer to the <code>MIXED_GAS</code> data structure                |

5.3.3.3 `int calculate_properties ( MIXED_GAS * gas_dat )`

Function to calculate the gas properties based on information in `MIXED_GAS`.

This function uses the kinetic theory of gases, combined with other semi-empirical models, to predict and approximate several properties of the mixed gas phase that might be necessary when running any gas dynamical simulation. This includes mass and energy transfer equations, as well as adsorption kinetics in porous adsorbents.

5.3.3.4 `int EGRET_TESTS ( )`

Function runs a series of tests for the EGRET file.

The test looks at a standard air with 5 primary species of interest and calculates the gas properties from 273 K to 373 K. This function can be called from the UI.

## 5.4 error.h File Reference

All error types are defined here.

```
#include <iostream>
```

### Macros

- `#define mError(i)`

### Enumerations

- `enum error_type {`  
generic\_error, file\_dne, indexing\_error, magpie\_reverse\_error,  
simulation\_fail, invalid\_components, invalid\_boolean, invalid\_molefraction,  
invalid\_gas\_sum, invalid\_solid\_sum, scenario\_fail, out\_of\_bounds,  
non\_square\_matrix, dim\_mis\_match, empty\_matrix, opt\_no\_support,  
invalid\_fraction, ortho\_check\_fail, unstable\_matrix, no\_diffusion,  
negative\_mass, negative\_time, matvec\_mis\_match, arg\_matrix\_same,  
singular\_matrix, matrix\_too\_small, invalid\_size, nullptr\_func,  
invalid\_norm, vector\_out\_of\_bounds, zero\_vector, tensor\_out\_of\_bounds,  
non\_real\_edge, nullptr\_error, invalid\_atom, invalid\_proton,  
invalid\_neutron, invalid\_electron, invalid\_valence, string\_parse\_error,  
unregistered\_name, rxn\_rate\_error, invalid\_species, duplicate\_variable,  
missing\_information, invalid\_type, key\_not\_found, anchor\_alias\_dne,  
initial\_error, not\_a\_token, read\_error, invalid\_console\_input }  
`}`

*List of names for error type.*

### Functions

- `void error (int flag)`

*Error function customizes output message based on flag.*

#### 5.4.1 Detailed Description

All error types are defined here. error.cpp

This file defines all the different errors that may occur in any simulation in any file. Those errors are recognized by an enum with is then passed through to the error.cpp file that customizes the error message to the console. A macro will also print out the file name and line number where the error occurred.

#### Author

Austin Ladshaw

#### Date

04/28/2014

#### Copyright

This software was designed and built at the Georgia Institute of Technology by Austin Ladshaw for PhD research in the area of adsorption and surface science. Copyright (c) 2015, all rights reserved.

### 5.4.2 Macro Definition Documentation

#### 5.4.2.1 #define mError( i )

##### Value:

```
{error(i);  
std::cout << "Source: " << __FILE__ << "\nLine: " << __LINE__ << std::endl;}
```

### 5.4.3 Enumeration Type Documentation

#### 5.4.3.1 enum error\_type

List of names for error type.

##### Enumerator

- generic\_error*
- file\_dne*
- indexing\_error*
- magpie\_reverse\_error*
- simulation\_fail*
- invalid\_components*
- invalid\_boolean*
- invalid\_molefraction*
- invalid\_gas\_sum*
- invalid\_solid\_sum*
- scenario\_fail*
- out\_of\_bounds*
- non\_square\_matrix*
- dim\_mis\_match*
- empty\_matrix*
- opt\_no\_support*
- invalid\_fraction*
- ortho\_check\_fail*
- unstable\_matrix*
- no\_diffusion*
- negative\_mass*
- negative\_time*
- matvec\_mis\_match*
- arg\_matrix\_same*
- singular\_matrix*
- matrix\_too\_small*
- invalid\_size*

*nullptr\_func*  
*invalid\_norm*  
*vector\_out\_of\_bounds*  
*zero\_vector*  
*tensor\_out\_of\_bounds*  
*non\_real\_edge*  
*nullptr\_error*  
*invalid\_atom*  
*invalid\_proton*  
*invalid\_neutron*  
*invalid\_electron*  
*invalid\_valence*  
*string\_parse\_error*  
*unregistered\_name*  
*rxn\_rate\_error*  
*invalid\_species*  
*duplicate\_variable*  
*missing\_information*  
*invalid\_type*  
*key\_not\_found*  
*anchor\_alias\_dne*  
*initial\_error*  
*not\_a\_token*  
*read\_error*  
*invalid\_console\_input*

#### 5.4.4 Function Documentation

##### 5.4.4.1 void error ( int flag )

Error function customizes output message based on flag.

This error function is reference in the error.cpp file, but is not called by any other file. Instead, all other files call the [mError\(i\)](#) macro that expands into this error function call plus prints out the file name and line number where the error occurred.

## 5.5 finch.h File Reference

Flux-limiting Implicit Non-oscillatory Conservative High-resolution scheme.

```
#include "macaw.h"
#include "lark.h"
```

### Classes

- struct [FINCH\\_DATA](#)  
Data structure for the FINCH object.



## Enumerations

- enum `finch_solve_type` { `FINCH_Picard`, `LARK_Picard`, `LARK_PJFNK` }  
*List of enum options to define the solver type in FINCH.*
- enum `finch_coord_type` { `Cartesian`, `Cylindrical`, `Spherical` }  
*List of enum options to define the coordinate system in FINCH.*

## Functions

- double `max` (std::vector< double > &values)  
*Function returns the maximum in a list of values.*
- double `min` (std::vector< double > &values)  
*Function returns the minimum in a list of values.*
- double `minmod` (std::vector< double > &values)  
*Function returns the result of the minmod function acting on a list of values.*
- int `uTotal` (FINCH\_DATA \*dat)  
*Function integrates the conserved quantity to return it's total in the domain.*
- int `uAverage` (FINCH\_DATA \*dat)  
*Function integrates the conserved quantity to return it's average in the domain.*
- int `check_Mass` (FINCH\_DATA \*dat)  
*Function checks the unp1 vector for negative values and will adjust if needed.*
- int `I_direct` (FINCH\_DATA \*dat)  
*Function solves the discretized FINCH problem directly by assuming it is linear.*
- int `lark_picard_step` (const Matrix< double > &x, Matrix< double > &G, const void \*data)  
*Function to perform the necessary LARK Picard iterative method (not typically used)*
- int `nl_picard` (FINCH\_DATA \*dat)  
*Function to solve the discretized FINCH problem iteratively by assuming it is non-linear.*
- int `setup_FINCH_DATA` (int(\*user\_callroutine)(const void \*user\_data), int(\*user\_setic)(const void \*user\_data), int(\*user\_timestep)(const void \*user\_data), int(\*user\_preprocess)(const void \*user\_data), int(\*user\_solve)(const void \*user\_data), int(\*user\_setparams)(const void \*user\_data), int(\*user\_discretize)(const void \*user\_data), int(\*user\_bcs)(const void \*user\_data), int(\*user\_res)(const Matrix< double > &x, Matrix< double > &res, const void \*user\_data), int(\*user\_precon)(const Matrix< double > &b, Matrix< double > &p, const void \*user\_data), int(\*user\_postprocess)(const void \*user\_data), int(\*user\_reset)(const void \*user\_data), FINCH\_DATA \*dat, const void \*param\_data)  
*Function to setup memory and set user defined functions into the FINCH object.*
- void `print2file_dim_header` (FILE \*Output, FINCH\_DATA \*dat)  
*Function will print out a dimension header for FINCH output.*
- void `print2file_time_header` (FILE \*Output, FINCH\_DATA \*dat)  
*Function will print out a time header for FINCH output.*
- void `print2file_result_old` (FILE \*Output, FINCH\_DATA \*dat)  
*Function will print out the old results to the variable u.*
- void `print2file_result_new` (FILE \*Output, FINCH\_DATA \*dat)  
*Function will print out the new results to the variable u.*
- void `print2file_newline` (FILE \*Output, FINCH\_DATA \*dat)  
*Function will force print out a blank line.*
- void `print2file_tab` (FILE \*Output, FINCH\_DATA \*dat)  
*Function will force print out a tab.*
- int `default_execution` (const void \*user\_data)  
*Default executioner function for FINCH.*
- int `default_ic` (const void \*user\_data)  
*Default initial conditions function for FINCH.*

- int [default\\_timestep](#) (const void \*user\_data)  
*Default time step function for FINCH.*
- int [default\\_preprocess](#) (const void \*user\_data)  
*Default preprocesses function for FINCH.*
- int [default\\_solve](#) (const void \*user\_data)  
*Default solve function for FINCH.*
- int [default\\_params](#) (const void \*user\_data)  
*Default params function for FINCH.*
- int [minmod\\_discretization](#) (const void \*user\_data)  
*Minmod Discretization function for FINCH.*
- int [vanAlbada\\_discretization](#) (const void \*user\_data)  
*Van Albada Discretization function for FINCH.*
- int [ospre\\_discretization](#) (const void \*user\_data)  
*Ospre Discretization function for FINCH.*
- int [default\\_bcs](#) (const void \*user\_data)  
*Default boundary conditions function for FINCH.*
- int [default\\_res](#) (const [Matrix](#)< double > &x, [Matrix](#)< double > &res, const void \*user\_data)  
*Default residual function for FINCH.*
- int [default\\_precon](#) (const [Matrix](#)< double > &b, [Matrix](#)< double > &p, const void \*user\_data)  
*Default preconditioning function for FINCH.*
- int [default\\_postprocess](#) (const void \*user\_data)
- int [default\\_reset](#) (const void \*user\_data)  
*Default reset function for FINCH.*
- int [FINCH\\_TESTS](#) ()  
*Function runs a particular FINCH test.*

### 5.5.1 Detailed Description

Flux-limiting Implicit Non-oscillatory Conservative High-resolution scheme. finch.cpp

This is a conservative finite differences scheme based on the Kurganov and Tadmor (2000) MUSCL scheme for non-linear conservation laws. It can solve 1-D conservation law problems in three different coordinate systems: (i) Cartesian - axial, (ii) Cylindrical - radial, and (iii) Spherical - radial. It is the backbone algorithm behind all 1-D PDE problems in the ecosystem software.

The form of the general conservation law problem that FINCH solves is...

$$z^d \frac{d}{dt} R \frac{du}{dz} = \frac{d}{dz} (z^d \frac{d}{dz} D \frac{du}{dz}) - \frac{d}{dz} (z^d \frac{d}{dz} v * u) - z^d \frac{d}{dz} k * u + z^d \frac{d}{dz} S$$

where R, D, v, k, and S are the parameters of the problem and d, z, and u are the coordinates, spatial dimension, and conserved quantities, respectively. The parameter R is a retardation coefficient, D is a diffusion coefficient, v is a velocity, k is a reaction coefficient, and S is a forcing function or source/sink term.

FINCH supports the use of both Dirichlet and Neuman boundary conditions as the input/inlet condition and uses the No Flux (or Natural) boundary condition for the output/outlet of the domain. For radial problems, the outlet is always taken to be the center of the cylindrical or spherical particle. This enforces the symmetry of the problem. For axial problems, the outlet is determined by the sign of the velocity term and is therefore chosen by the routine based on the actual flow direction in the domain.

Parameters of the problem can be coupled to the variable u and also be functions of space and time. The coupling of the parameters with the variable forces the problem to become non-linear, which requires iteration to solve. The default iterative method is a built-in Picard's method. This method is equivalent to an inexact Newton method, because we use the Linear Solve of this system as a weak approximation to the non-linear solve. Generally, this method is sufficient and is the most efficient. However, if a problem is particularly difficult to solve, then we can call some of the non-linear solvers developed in LARK. If PJFNK is used, then the Linear Solve for the FINCH problem is used as the Preconditioner for the Linear Solve in PJFNK.

This algorithm comes packaged with three different slope limiter functions to stabilize the velocity term for highly advectively dominate problems. The available slope limiters are: (i) minmod, (ii) van Albada, and (iii) ospre. By default, the FINCH setup function will set the slope limiter to ospre, because this method provides a reasonable compromise between accuracy and efficiency.

#### Slope Limiter Stats:

minmod -> Highest Accuracy, Lowest Efficiency

van Albada -> Lowest Accuracy, Highest Efficiency

ospre -> Average Accuracy, Average Efficiency

#### Author

Austin Ladshaw

#### Date

01/29/2015

#### Copyright

This software was designed and built at the Georgia Institute of Technology by Austin Ladshaw for PhD research in the area of adsorption and surface science. Copyright (c) 2015, all rights reserved.

### 5.5.2 Enumeration Type Documentation

#### 5.5.2.1 enum finch\_solve\_type

List of enum options to define the solver type in FINCH.

##### Enumerator

***FINCH\_Picard***

***LARK\_Picard***

***LARK\_PJFNK***

#### 5.5.2.2 enum finch\_coord\_type

List of enum options to define the coordinate system in FINCH.

##### Enumerator

***Cartesian***

***Cylindrical***

***Spherical***

### 5.5.3 Function Documentation

#### 5.5.3.1 double max ( std::vector< double > & values )

Function returns the maximum in a list of values.

### 5.5.3.2 double min ( std::vector< double > & values )

Function returns the minimum in a list of values.

### 5.5.3.3 double minmod ( std::vector< double > & values )

Function returns the result of the minmod function acting on a list of values.

### 5.5.3.4 int uTotal ( FINCH\_DATA \* dat )

Function integrates the conserved quantity to return it's total in the domain.

### 5.5.3.5 int uAverage ( FINCH\_DATA \* dat )

Function integrates the conserved quantity to return it's average in the domain.

### 5.5.3.6 int check\_Mass ( FINCH\_DATA \* dat )

Function checks the unp1 vector for negative values and will adjust if needed.

This function can be turned off or on in the [FINCH\\_DATA](#) structure. Typically, you will want to leave this on so that the routine does not return negative values for u. However, if you want to get negative values of u, then turn this option off.

### 5.5.3.7 int l\_direct ( FINCH\_DATA \* dat )

Function solves the discretized FINCH problem directly by assuming it is linear.

### 5.5.3.8 int lark\_picard\_step ( const Matrix< double > & x, Matrix< double > & G, const void \* data )

Function to perform the necessary LARK Picard iterative method (not typically used)

### 5.5.3.9 int nl\_picard ( FINCH\_DATA \* dat )

Function to solve the discretized FINCH problem iteratively by assuming it is non-linear.

#### Note

If the problem is actually linear, then this will solve it in one iteration. So it may be best to always assume the problem is non-linear.

### 5.5.3.10 int setup\_FINCH\_DATA ( int (\*)(const void \*user\_data) user\_callroutine, int (\*)(const void \*user\_data) user\_setic, int (\*)(const void \*user\_data) user\_timestep, int (\*)(const void \*user\_data) user\_preprocess, int (\*)(const void \*user\_data) user\_solve, int (\*)(const void \*user\_data) user\_setparams, int (\*)(const void \*user\_data) user\_discretize, int (\*)(const void \*user\_data) user\_bcs, int (\*)(const Matrix< double > &x, Matrix< double > &res, const void \*user\_data) user\_res, int (\*)(const Matrix< double > &b, Matrix< double > &p, const void \*user\_data) user\_precon, int (\*)(const void \*user\_data) user\_postprocess, int (\*)(const void \*user\_data) user\_reset, FINCH\_DATA \* dat, const void \* param\_data )

Function to setup memory and set user defined functions into the FINCH object.

This function MUST be called prior to running any FINCH based simulation. However, you are only every required to provide this function with the [FINCH\\_DATA](#) pointer. It is recommended, however, that you do provide the `user_setparams` and `param_data` pointers, as these will likely vary significantly from problem to problem.

After the problem is setup in memory, you do not technically have to have FINCH call all of it's own functions. You can write your own executioner, initial conditions, and other functions and decided how and when everything is called. Then just call the `solve` function in [FINCH\\_DATA](#) when you want to use the FINCH solver. This is how FINCH is used in SKUA, SCOPSOWL, DOGFISH, and MONKFISH.

**5.5.3.11** `void print2file_dim_header ( FILE * Output, FINCH_DATA * dat )`

Function will print out a dimension header for FINCH output.

**5.5.3.12** `void print2file_time_header ( FILE * Output, FINCH_DATA * dat )`

Function will print out a time header for FINCH output.

**5.5.3.13** `void print2file_result_old ( FILE * Output, FINCH_DATA * dat )`

Function will print out the old results to the variable `u`.

**5.5.3.14** `void print2file_result_new ( FILE * Output, FINCH_DATA * dat )`

Function will print out the new results to the variable `u`.

**5.5.3.15** `void print2file_newline ( FILE * Output, FINCH_DATA * dat )`

Function will force print out a blank line.

**5.5.3.16** `void print2file_tab ( FILE * Output, FINCH_DATA * dat )`

Function will force print out a tab.

**5.5.3.17** `int default_execution ( const void * user_data )`

Default executioner function for FINCH.

The default executioner function for FINCH assumes the `user_data` parameter is the [FINCH\\_DATA](#) structure and calls the `preprocesses`, `solve`, `postprocesses`, `checkMass`, `uTotal`, and `uAverage` functions in that order.

**5.5.3.18** `int default_ic ( const void * user_data )`

Default initial conditions function for FINCH.

The default initial condition function for FINCH assumes the `user_data` parameter is the [FINCH\\_DATA](#) structure and sets the initial values of all system parameters according to the given constants in that structure.

**5.5.3.19** `int default_timestep ( const void * user_data )`

Default time step function for FINCH.

The default time step function for FINCH assumes the `user_data` parameter is the [FINCH\\_DATA](#) structure and sets the time step to 1/2 the mesh size or bases the time step off of the CFL condition if the problem is not being solved iteratively and involves an advective portion.

### 5.5.3.20 `int default_preprocess ( const void * user_data )`

Default preprocesses function for FINCH.

The default preprocesses function for FINCH assumes the `user_data` parameter is the [FINCH\\_DATA](#) structure and does nothing.

### 5.5.3.21 `int default_solve ( const void * user_data )`

Default solve function for FINCH.

The default solve function for FINCH assumes the `user_data` parameter is the [FINCH\\_DATA](#) structure and calls the corresponding solution method depending on the users conditions.

### 5.5.3.22 `int default_params ( const void * user_data )`

Default params function for FINCH.

The default params function for FINCH assumes the `user_data` parameter is the [FINCH\\_DATA](#) structure and sets the values of all parameters at all nodes equal to the values of those parameters at the boundaries.

### 5.5.3.23 `int minmod_discretization ( const void * user_data )`

Minmod Discretization function for FINCH.

The minmod discretization function for FINCH assumes the `user_data` parameter is the [FINCH\\_DATA](#) structure and discretizes the time and space portion of the problem with 2nd order finite differences and uses the minmod slope limiter function to stabilize the advective physics.

### 5.5.3.24 `int vanAlbada_discretization ( const void * user_data )`

Van Albada Discretization function for FINCH.

The van Albada discretization function for FINCH assumes the `user_data` parameter is the [FINCH\\_DATA](#) structure and discretizes the time and space portion of the problem with 2nd order finite differences and uses the van Albada slope limiter function to stabilize the advective physics.

### 5.5.3.25 `int ospre_discretization ( const void * user_data )`

Ospre Discretization function for FINCH.

The ospre discretization function for FINCH assumes the `user_data` parameter is the [FINCH\\_DATA](#) structure and discretizes the time and space portion of the problem with 2nd order finite differences and uses the ospre slope limiter function to stabilize the advective physics. This is the default discretization function.

### 5.5.3.26 `int default_bcs ( const void * user_data )`

Default boundary conditions function for FINCH.

The default boundary conditions function for FINCH assumes the `user_data` parameter is the [FINCH\\_DATA](#) structure and sets the boundary conditions according to the type of problem requested. The input BCs will always be either Neumann or Dirichlet and the output BC will always be a zero flux Neumann BC.

### 5.5.3.27 `int default_res ( const Matrix< double > & x, Matrix< double > & res, const void * user_data )`

Default residual function for FINCH.

The default residual function for FINCH assumes the `user_data` parameter is the [FINCH\\_DATA](#) structure and calls the `setparams` function (passing the `param_data` structure), the discretization function, and the set BCs functions, in that order. It then forms the implicit and explicit side residuals that go into the iterative solver.

**5.5.3.28** `int default_precon ( const Matrix< double > & b, Matrix< double > & p, const void * user_data )`

Default preconditioning function for FINCH.

The default preconditioning function for FINCH assumes the `user_data` parameter is the [FINCH\\_DATA](#) structure and performs a tridiagonal linear solve using a Modified Thomas Algorithm. This preconditioner will solve the linear problem exactly if there is no advective portion of the physics. Additionally, this preconditioner is also used as the basis for forming the default FINCH non-linear iterations and is sufficient for solving most problems.

**5.5.3.29** `int default_postprocess ( const void * user_data )`

The default postprocesses function for FINCH assumes the `user_data` parameter is the [FINCH\\_DATA](#) structure and does nothing.

**5.5.3.30** `int default_reset ( const void * user_data )`

Default reset function for FINCH.

The default reset function for FINCH assumes the `user_data` parameter is the [FINCH\\_DATA](#) structure and sets all old state parameters and variables to the new state.

**5.5.3.31** `int FINCH_TESTS ( )`

Function runs a particular FINCH test.

The `FINCH_TESTS` function is used to exercise and test out the FINCH algorithms for correctness, efficiency, and accuracy. This test should never report a failure.

## 5.6 flock.h File Reference

Fundamental Off-gas Collection of Kernels.

```
#include "macaw.h"
#include "egret.h"
#include "finch.h"
#include "lark.h"
#include "skua.h"
#include "scopsowl.h"
#include "gsta_opt.h"
#include "magpie.h"
#include "skua_opt.h"
#include "scopsowl_opt.h"
#include "yaml_wrapper.h"
```

### 5.6.1 Detailed Description

Fundamental Off-gas Collection of Kernels. This is just a `.h` file that holds all the includes necessary to develop and run simulations for adsorption and/or mass/energy transfer problems for gaseous systems. Include this file into any other project or source code that needs the methods below.

### Files Included in FLOCK

[macaw.h](#) [egret.h](#) [finch.h](#) [lark.h](#) [skua.h](#) [scopsowl.h](#) [gsta\\_opt.h](#) [magpie.h](#) [skua\\_opt.h](#) [scopsowl\\_opt.h](#) [yaml\\_wrapper.h](#)

### Author

Austin Ladshaw

### Date

04/28/2014

### Copyright

This software was designed and built at the Georgia Institute of Technology by Austin Ladshaw for PhD research in the area of adsorption and surface science. Copyright (c) 2015, all rights reserved.

## 5.7 gsta\_opt.h File Reference

Generalized Statistical Thermodynamic Adsorption (GSTA) Optimization Routine.

```
#include "lmcurve.h"
#include <stdio.h>
#include <math.h>
#include <iostream>
#include <fstream>
#include <stdlib.h>
#include <vector>
#include <time.h>
#include <float.h>
#include <string>
#include "error.h"
```

### Classes

- struct [GSTA\\_OPT\\_DATA](#)  
*Data structure used in the GSTA optimization routines.*

### Macros

- #define [Po](#) 100.0  
*Standard State Pressure - Units: kPa.*
- #define [R](#) 8.3144621  
*Gas Constant - Units: J/(K\*mol) = kB \* Na.*
- #define [Na](#) 6.0221413E+23  
*Avagadro's Number - Units: molecules/mol.*

### Functions

- int [roundIt](#) (double d)  
*Function rounds a double to an integer.*
- int [twoFifths](#) (int m)



- Function returns the rounded two-fifths result of int m.*
- int `orderMag` (double x)
- Function returns the order of magnitude for the parameter x.*
- int `minValue` (std::vector< int > &array)
- Function returns the minimum integer in an array of integers.*
- int `minIndex` (std::vector< double > &array)
- Function returns the index of the minimum integer in an array of integers.*
- int `avgPar` (std::vector< int > &array)
- Function returns the average integer value in an array of integers.*
- double `avgValue` (std::vector< double > &array)
- Function returns an average in an array of doubles.*
- double `weightedAvg` (double \*enorm, double \*x, int n)
- Function returns a weighted average in an array.*
- double `rSq` (double \*x, double \*y, double slope, double vint, int m\_dat)
- Function calculates the Coefficient of Determination (R Squared) for the temperature regression.*
- bool `isSmooth` (double \*par, void \*data)
- Function looks at the list of parameters to check if they are smoothly changing.*
- void `orthoLinReg` (double \*x, double \*y, double \*par, int m\_dat, int n\_par)
- Function performs an Orthogonal Linear Regression on a set of data.*
- void `eduGuess` (double \*P, double \*q, double \*par, int k, int m\_dat, void \*data)
- Function will formed an educated guess for the next set of parameters in the GSTA analysis.*
- double `gstaFunc` (double p, const double \*K, double qmax, int n\_par)
- Function evaluates the result of the GSTA isotherm model.*
- double `gstaObjFunc` (double \*t, double \*y, double \*par, int m\_dat, void \*data)
- Function to evaluate the GSTA objective function value.*
- void `eval_GSTA` (const double \*par, int m\_dat, const void \*data, double \*fvec, int \*info)
- Function to evaluate the GSTA model and feed into the Imfit routine.*
- int `gsta_optimize` (const char \*fileName)
- Function to perform the GSTA optimization routine.*

### 5.7.1 Detailed Description

Generalized Statistical Thermodynamic Adsorption (GSTA) Optimization Routine. gsta\_opt.cpp

Optimization routine developed for the GSTA isotherm and data analysis. This algorithm was the primary subject of a publication made in Fluid Phase Equilibria. Please refer to the below paper for technical information about the algorithms.

Reference: Ladshaw, Yiaccoumi, Tsouris, and DePaoli, Fluid Phase Equilibria, 388, 169-181, 2015.

The GSTA model was first introduced by Llano-Restrepo and Mosquera (2009). Please refer to the below reference for theoretical information about the model.

Reference: Llano-Restrepo and Mosquera, Fluid Phase Equilibria, 283, 73-88, 2009.

#### Author

Austin Ladshaw

#### Date

12/17/2013

#### Copyright

This software was designed and built at the Georgia Institute of Technology by Austin Ladshaw for PhD research in the area of adsorption and surface science. Copyright (c) 2015, all rights reserved.

## 5.7.2 Macro Definition Documentation

### 5.7.2.1 `#define Po 100.0`

Standard State Pressure - Units: kPa.

### 5.7.2.2 `#define R 8.3144621`

Gas Constant - Units: J/(K\*mol) = kB \* Na.

### 5.7.2.3 `#define Na 6.0221413E+23`

Avagadro's Number - Units: molecules/mol.

## 5.7.3 Function Documentation

### 5.7.3.1 `int roundIt ( double d )`

Function rounds a double to an integer.

This function returns a rounded value of d. Rounding up for any decimal larger than 0.5 and down for all else.

### 5.7.3.2 `int twoFifths ( int m )`

Function returns the rounded two-fifths result of int m.

This function is used to determine what the maximum number of parameters should be based on the number of data points m. It is designed to prevent the algorithms from "over fitting" the data.

### 5.7.3.3 `int orderMag ( double x )`

Function returns the order of magnitude for the parameter x.

This function is used to help create initial guesses for the new GSTA parameters that are being optimized for. In order to make sure that those parameters are considered relevant in the optimization routine, we need to make the initial guesses to be around the same order of magnitude of the other GSTA parameters.

### 5.7.3.4 `int minValue ( std::vector< int > & array )`

Function returns the minimum integer in an array of integers.

This function is used to determine the minimum number of GSTA parameters that were required to adequately fit the isotherm data.

### 5.7.3.5 `int minIndex ( std::vector< double > & array )`

Function returns the index of the minimum integer in an array of integers.

This function identifies the index of the minimum number of parameters needed for the GSTA model to fit the data. This index is common for all vectors in the [GSTA\\_OPT\\_DATA](#) structure and is used to identify the most suitable solution.

### 5.7.3.6 int avgPar ( std::vector< int > & array )

Function returns the average integer value in an array of integers.

This function is used to identify the average number of parameters that all the data fitting needed for each GSTA analysis.

### 5.7.3.7 double avgValue ( std::vector< double > & array )

Function returns an average in an array of doubles.

### 5.7.3.8 double weightedAvg ( double \* enorm, double \* x, int n )

Function returns a weighted average in an array.

This averaging scheme is used to approximate the qmax parameter for the GSTA isotherm model, if that value is unknown. The weighting is based on the euclidean norms of all the fits of the data. Smaller norms are more heavily weighted since they represent a better fit of the data. Once averaging is complete and we have an estimate for qmax, the entire algorithm is re-run holding that qmax constant.

#### Parameters

|              |   |
|--------------|---|
| <i>enorm</i> | array of euclidean norms from the fitting of the data |
| <i>x</i>     | array of optimum qmax values to be averaged           |
| <i>n</i>     | the number of enorm and x values in the array         |

### 5.7.3.9 double rSq ( double \* x, double \* y, double slope, double vint, int m\_dat )

Function calculates the Coefficient of Determination (R Squared) for the temperature regression.

This function is used to determine the "fitness" of the linear regression performed on the temperature independent parameters of the GSTA isotherm. A good linear regression should return a value between 1.0 and 0.9.

#### Parameters

|              |   |
|--------------|---|
| <i>x</i>     | observations in the x-axis                          |
| <i>y</i>     | observations in the y-axis                          |
| <i>slope</i> | slope of the linear regression                      |
| <i>vint</i>  | intercept of the linear regression                  |
| <i>m_dat</i> | number of data points used in the linear regression |

### 5.7.3.10 bool isSmooth ( double \* par, void \* data )

Function looks at the list of parameters to check if they are smoothly changing.

This function takes the parameter array par and [GSTA\\_OPT\\_DATA](#) structure and checks to see if those parameters are changing smoothly. If they are erratic or non-smooth, then it could be an indication of "over fitting" of the data.

### 5.7.3.11 void orthoLinReg ( double \* x, double \* y, double \* par, int m\_dat, int n\_par )

Function performs an Orthogonal Linear Regression on a set of data.

This function takes an array of x and y observations and performs an orthogonal linear regression on that information to find optimum parameters for slope and intercept.

## Parameters

|              |  |
|--------------|--|
| <i>x</i>     | array of x-axis observations   |
| <i>y</i>     | array of y-axis observations   |
| <i>par</i>   | array of parameter results after regression  |
| <i>m_dat</i> | number of data points or observations  |
| <i>n_par</i> | number of parameters to seek (if <i>n_par</i> != 1 or 2, then <i>par</i> [0] = intercept and <i>par</i> [1] = slope) |

5.7.3.12 void eduGuess ( double \* *P*, double \* *q*, double \* *par*, int *k*, int *m\_dat*, void \* *data* )

Function will formed an educated guess for the next set of parameters in the GSTA analysis.

This function takes partial pressure and adsorption observations, *P* and *q*, and tries to give a decent initial guess to what the GSTA parameters, *par*, will be for the next iteration.

## Parameters

|              |   |
|--------------|---|
| <i>P</i>     | partial pressure observations in the data (kPa)             |
| <i>q</i>     | adsorption observations in the data (any units)             |
| <i>par</i>   | parameter array for the GSTA isotherm                       |
| <i>k</i>     | index of the current number of parameters being considered  |
| <i>m_dat</i> | number of pressure-adsorption observations in the isotherm  |
| <i>data</i>  | pointer to the <a href="#">GSTA_OPT_DATA</a> data structure |

5.7.3.13 double gstaFunc ( double *p*, const double \* *K*, double *qmax*, int *n\_par* )

Function evaluates the result of the GSTA isotherm model.

This function will evaluate the GSTA model and return the adsorbed amount given the current partial pressure *p* and the equilibrium parameters *K*.

## Parameters

|              |  |
|--------------|--|
| <i>p</i>     | current partial pressure (kPa)                       |
| <i>K</i>     | array of equilibrium parameters ( $1/\text{kPa}^n$ ) |
| <i>qmax</i>  | the theoretical maximum capacity for the isotherm    |
| <i>n_par</i> | the number of equilibrium parameters                 |

5.7.3.14 double gstaObjFunc ( double \* *t*, double \* *y*, double \* *par*, int *m\_dat*, void \* *data* )

Function to evaluate the GSTA objective function value.

The objective function seeks to penalize the relative fitness of the model based on the number of parameters it took to minimize the euclidean norms. By penalizing the fitness of the model in this fashion, we can find the best solution to the system that required the least number of equilibrium parameters.

5.7.3.15 void eval\_GSTA ( const double \* *par*, int *m\_dat*, const void \* *data*, double \* *fvec*, int \* *info* )

Function to evaluate the GSTA model and feed into the Imfit routine.

This function will formulate the residuals that go into the Levenberg-Marquardt's Algorithm for non-linear least squares regression. The form of this function is specific to how we interface with the Imfit routines.

5.7.3.16 int gsta\_optimize ( const char \* *fileName* )

Function to perform the GSTA optimization routine.

This function is callable from the UI and is used to find the optimum parameters of the GSTA isotherm model given a particular set of isotherm data for single-component adsorption equilibria.

**Parameters**

|                 |   |
|-----------------|---|
| <i>fileName</i> | name of the input file that holds the isotherm data |
|-----------------|---|

**Note**

The input file for the GSTA optimization routine is a text file holding the necessary information and data needed to run the routine. That input file has a very specific format that is detailed below.

**Number of Isotherm Curves**

Theoretical Maximum Adsorption Capacity (if unknown, provide 0)

Temperature of the ith Isotherm (K)

Number of Data points for the ith Isotherm

Partial Pressure (kPa) (tab) Corresponding Adsorbed Amount (any units)

(2nd Line down is repeated for all isotherms you are optimizing on...)

**Example:**

```
2
21.0
298.15
4
0.000165483 2.77
0.000306379 2.75
0.00044922 5.00
0.000939259 10.40
313.15
4
0.000589636 2.75
0.001063584 3.70
0.001351836 4.2
0.001543464 4.6
```

The above example would be for 2 sets of isotherms at 298.15 and 313.15 K, respectively. Maximum adsorption capacity is given as 21 (which in this has units of wt%). Each isotherm has 4 data points, which are given in a list as p (kPa) and q (wt%) pairs. Units of adsorption don't matter as long as they are consistent. If you give maximum capacity in mol/kg, then the q's in the lists must also be in mol/kg.

## 5.8 lark.h File Reference

Linear Algebra Residual Kernels.

```
#include "macaw.h"
#include <float.h>
```

## Classes

- struct [ARNOLDI\\_DATA](#)  
*Data structure for the construction of the Krylov subspaces for a linear system.*
- struct [GMRESLP\\_DATA](#)  
*Data structure for implementation of the Restarted GMRES algorithm with Left Preconditioning.*
- struct [GMRESRP\\_DATA](#)  
*Data structure for the Restarted GMRES algorithm with Right Preconditioning.*
- struct [PCG\\_DATA](#)  
*Data structure for implementation of the PCG algorithms for symmetric linear systems.*
- struct [BiCGSTAB\\_DATA](#)  
*Data structure for the implementation of the BiCGSTAB algorithm for non-symmetric linear systems.*
- struct [CGS\\_DATA](#)  
*Data structure for the implementation of the CGS algorithm for non-symmetric linear systems.*
- struct [OPTRANS\\_DATA](#)  
*Data structure for implementation of linear operator transposition.*
- struct [GCR\\_DATA](#)  
*Data structure for the implementation of the GCR algorithm for non-symmetric linear systems.*
- struct [GMRESR\\_DATA](#)  
*Data structure for the implementation of GCR with Nested GMRES preconditioning (i.e., GMRESR)*
- struct [PICARD\\_DATA](#)  
*Data structure for the implementation of a Picard or Fixed-Point iteration for non-linear systems.*
- struct [BACKTRACK\\_DATA](#)  
*Data structure for the implementation of Backtracking Linesearch.*
- struct [PJFNK\\_DATA](#)  
*Data structure for the implementation of the PJFNK algorithm for non-linear systems.*
- struct [NUM\\_JAC\\_DATA](#)  
*Data structure to form a numerical jacobian matrix with finite differences.*

## Enumerations

- enum [krylov\\_method](#) {  
[GMRESLP](#), [PCG](#), [BiCGSTAB](#), [CGS](#),  
[FOM](#), [GMRESRP](#), [GCR](#), [GMRESR](#) }  
*Enum of definitions for linear solver types in PJFNK.*

## Functions

- int [update\\_arnoldi\\_solution](#) ([Matrix](#)< double > &x, [Matrix](#)< double > &x0, [ARNOLDI\\_DATA](#) \*arnoldi\_dat)  
*Function to update the linear vector x based on the Arnoldi Krylov subspace.*
- int [arnoldi](#) (int(\*matvec)(const [Matrix](#)< double > &v, [Matrix](#)< double > &w, const void \*data), int(\*precon)(const [Matrix](#)< double > &b, [Matrix](#)< double > &p, const void \*data), [Matrix](#)< double > &r0, [ARNOLDI\\_DATA](#) \*arnoldi\_dat, const void \*matvec\_data, const void \*precon\_data)  
*Function to factor a linear operator into an orthonormal basis and upper Hessenberg matrix.*
- int [gmresLeftPreconditioned](#) (int(\*matvec)(const [Matrix](#)< double > &v, [Matrix](#)< double > &w, const void \*data), int(\*precon)(const [Matrix](#)< double > &b, [Matrix](#)< double > &p, const void \*data), [Matrix](#)< double > &b, [GMRESLP\\_DATA](#) \*gmreslp\_dat, const void \*matvec\_data, const void \*precon\_data)  
*Function to iteratively solve a non-symmetric, indefinite linear system with GMRESLP.*
- int [fom](#) (int(\*matvec)(const [Matrix](#)< double > &v, [Matrix](#)< double > &w, const void \*data), int(\*precon)(const [Matrix](#)< double > &b, [Matrix](#)< double > &p, const void \*data), [Matrix](#)< double > &b, [GMRESLP\\_DATA](#) \*gmreslp\_dat, const void \*matvec\_data, const void \*precon\_data)

*Function to directly solve a non-symmetric, indefinite linear system with FOM.*

- int [gmresRightPreconditioned](#) (int(\*matvec)(const [Matrix](#)< double > &v, [Matrix](#)< double > &w, const void \*data), int(\*precon)(const [Matrix](#)< double > &b, [Matrix](#)< double > &p, const void \*data), [Matrix](#)< double > &b, [GMRESRP\\_DATA](#) \*gmresrp\_dat, const void \*matvec\_data, const void \*precon\_data)

*Function to iteratively solve a non-symmetric, indefinite linear system with GMRESRP.*

- int [pcg](#) (int(\*matvec)(const [Matrix](#)< double > &p, [Matrix](#)< double > &Ap, const void \*data), int(\*precon)(const [Matrix](#)< double > &r, [Matrix](#)< double > &z, const void \*data), [Matrix](#)< double > &b, [PCG\\_DATA](#) \*pcg\_dat, const void \*matvec\_data, const void \*precon\_data)

*Function to iteratively solve a symmetric, definite linear system with PCG.*

- int [bicgstab](#) (int(\*matvec)(const [Matrix](#)< double > &p, [Matrix](#)< double > &Ap, const void \*data), int(\*precon)(const [Matrix](#)< double > &r, [Matrix](#)< double > &z, const void \*data), [Matrix](#)< double > &b, [BICGSTAB\\_DATA](#) \*bicgstab\_dat, const void \*matvec\_data, const void \*precon\_data)

*Function to iteratively solve a non-symmetric, definite linear system with BiCGSTAB.*

- int [cgs](#) (int(\*matvec)(const [Matrix](#)< double > &p, [Matrix](#)< double > &Ap, const void \*data), int(\*precon)(const [Matrix](#)< double > &r, [Matrix](#)< double > &z, const void \*data), [Matrix](#)< double > &b, [CGS\\_DATA](#) \*cgs\_dat, const void \*matvec\_data, const void \*precon\_data)

*Function to iteratively solve a non-symmetric, definite linear system with CGS.*

- int [operatorTranspose](#) (int(\*matvec)(const [Matrix](#)< double > &v, [Matrix](#)< double > &Av, const void \*data), [Matrix](#)< double > &r, [Matrix](#)< double > &u, [OPTRANS\\_DATA](#) \*transpose\_dat, const void \*matvec\_data)

*Function that is used to perform transposition of a linear operator and results in a new vector  $A^T r = u$ .*

- int [gcr](#) (int(\*matvec)(const [Matrix](#)< double > &x, [Matrix](#)< double > &Ax, const void \*data), int(\*precon)(const [Matrix](#)< double > &r, [Matrix](#)< double > &Mr, const void \*data), [Matrix](#)< double > &b, [GCR\\_DATA](#) \*gcr\_dat, const void \*matvec\_data, const void \*precon\_data)

*Function to iteratively solve a non-symmetric, definite linear system with GCR.*

- int [gmresPreconditioner](#) (const [Matrix](#)< double > &r, [Matrix](#)< double > &Mr, const void \*data)

*Function used in conjunction with GMRESR to apply GMRESRP iterations as a preconditioner.*

- int [gmresr](#) (int(\*matvec)(const [Matrix](#)< double > &x, [Matrix](#)< double > &Ax, const void \*data), int(\*terminal\_precon)(const [Matrix](#)< double > &r, [Matrix](#)< double > &Mr, const void \*data), [Matrix](#)< double > &b, [GMRESR\\_DATA](#) \*gmresr\_dat, const void \*matvec\_data, const void \*term\_precon\_data)

*Function to iteratively solve a non-symmetric, indefinite linear system with GMRESR.*

- int [picard](#) (int(\*res)(const [Matrix](#)< double > &x, [Matrix](#)< double > &r, const void \*data), int(\*evalx)(const [Matrix](#)< double > &x0, [Matrix](#)< double > &x, const void \*data), [Matrix](#)< double > &x, [PICARD\\_DATA](#) \*picard\_dat, const void \*res\_data, const void \*evalx\_data)

*Function to iteratively solve a non-linear system using the Picard or Fixed-Point method.*

- int [jacvec](#) (const [Matrix](#)< double > &v, [Matrix](#)< double > &Jv, const void \*data)

*Function to form a linear operator of a Jacobian matrix used along with the PJFNK method.*

- int [backtrackLineSearch](#) (int(\*feval)(const [Matrix](#)< double > &x, [Matrix](#)< double > &F, const void \*data), [Matrix](#)< double > &Fkp1, [Matrix](#)< double > &xkp1, [Matrix](#)< double > &pk, double normFk, [BACKTRACK\\_DATA](#) \*backtrack\_dat, const void \*feval\_data)

*Function to perform a Backtracking Line Search operation to smooth out convergence of PJFNK.*

- int [pjfnk](#) (int(\*res)(const [Matrix](#)< double > &x, [Matrix](#)< double > &F, const void \*data), int(\*precon)(const [Matrix](#)< double > &r, [Matrix](#)< double > &p, const void \*data), [Matrix](#)< double > &x, [PJFNK\\_DATA](#) \*pjfnk\_dat, const void \*res\_data, const void \*precon\_data)

*Function to perform the PJFNK algorithm to solve a non-linear system of equations.*

- int [NumericalJacobian](#) (int(\*Func)(const [Matrix](#)< double > &x, [Matrix](#)< double > &F, const void \*user\_data), const [Matrix](#)< double > &x, [Matrix](#)< double > &J, int Nx, int Nf, [NUM\\_JAC\\_DATA](#) \*jac\_dat, const void \*user\_data)

*Function to form a full numerical Jacobian matrix from a given non-linear function.*

- int [LARK\\_TESTS](#) ()

*Function that runs a variety of tests on all the functions in LARK.*

### 5.8.1 Detailed Description

Linear Algebra Residual Kernels. lark.cpp

The functions contained within are designed to solve generic linear and non-linear square systems of equations given a function argument and data from the user. Optionally, the user can also provide a function to return a preconditioning result that will be applied to the system.

Having the user define how the preconditioning is carried out provides two major advantages: (1) we do not need to store and large, sparse preconditioning matrices and instead only store the preconditioned vector result and (2) this allows the user to use any kind of preconditioner they see fit for their problem.

The Arnoldi function is typically not called by the user, but can be if desired. It accepts the function arguments and a residual vector to form an orthonormal basis of the Krylov subspace using the Modified Gram-Schmidt process (aka Arnoldi Iteration). This function is called by GMRES to iteratively solve a linear system of equations. Note that you can use this function to directly solve the linear system as long as that system is not too large. Construction of the basis is expensive, which is why this is used as a sub-function of an iterative method.

The Restarted GMRES function will accept function arguments for a linear system and attempt to solve said system iteratively by constructing an orthonormal basis from the Krylov function. Note that this GMRES function does support restarting and will use restarting by default if the linear system is too large.

Also included is a GMRES algorithm without restarting. This will directly solve the linear system within residual tolerance using a Full Orthogonal basis set of that system. It is equivalent to calling the Krylov method with the  $k$  parameter equal to  $N$  (i.e. the number of equations). This method is nick-named the Full Orthogonalization Method (FOM), although the true FOM algorithm in literature is slightly different.

The PJFNK function will accept function arguments for a square, non-linear system of equations and attempt to solve it iteratively using both the GMRES and Krylov functions with Newton's method to convert the non-linear system into a linear system.

Also built here is a PCG implementation for solving symmetric linear systems. Can also be called by PJFNK if we know that the linear system (i.e. the Jacobian) is symmetric. This algorithm is significantly more efficient than GMRES, but is only valid if the system of equations is symmetric.

Other linear solvers implemented in this work are the BiCGSTAB and CGS algorithms for non-symmetric, positive definite matrices. These algorithms are significantly more computationally efficient than GMRES or FOM. However, they can both break down if the linear system is poorly conditioned. In general, you only want to use these methods if you have preconditioning available and your linear system is very, very large. Otherwise, you will be better suited to using GMRES or FOM.

There is also an implementation of the Generalized Conjugate Residual (GCR) method with and without restarting. This is a GMRES-like method that should give the exact solution within  $N$  iterations, where  $N$  is the original size of the matrix. Built on top of the GCR method is a GMRESR (or GMRES Recursive) algorithm that uses GCR as the base method and performs GMRESRP iterations as a preconditioner at each iteration of GCR. This is the only linear solver that has built-in preconditioning. As a result, it may be slower than other algorithms for simple problems, but generally will have much better convergence behavior and will almost always give better residual reduction, even for hard to solve problems.

NOTE: There are three GMRES implementations: (i) gmresLP, (ii) fom, and (iii) gmresRP. GMRESLP is a restarted GMRES implementation that is left preconditioned and only checks the residual on the outer loops. This may be less efficient than GMRESRP, which can check both outer and inner loop residuals. However, GMRESRP has to use right preconditioning, which also slightly changes the convergence behavior of the linear system. GMRES with left preconditioning and without restarting will just build the full subspace by default, thus solving the system exactly, but may require too much memory. You can do a GMRESRP unrestarted by specifying that the restart parameter be equal to the size of the problem.

Basic Implementation Details:

Linear Solvers -> Solve  $Ax=b$  for  $x$

Non-Linear Solvers -> Solve  $F(x)=0$  for  $x$

All implementations require system size to be 2 or greater



**Author**

Austin Ladshaw

**Date**

10/14/2014

**Copyright**

This software was designed and built at the Georgia Institute of Technology by Austin Ladshaw for PhD research in the area of adsorption and surface science. Copyright (c) 2015, all rights reserved.

**5.8.2 Enumeration Type Documentation****5.8.2.1 enum krylov\_method**

Enum of definitions for linear solver types in PJFNK.

Enum delineates the available Krylov Subspace methods that can be used to solve the linear sub-problem at each non-linear iteration in a Newton method.

**Enumerator*****GMRESLP******PCG******BiCGSTAB******CGS******FOM******GMRESRP******GCR******GMRESR*****5.8.3 Function Documentation****5.8.3.1 int update\_arnoldi\_solution ( Matrix< double > & x, Matrix< double > & x0, ARNOLDI\_DATA \* arnoldi\_dat )**

Function to update the linear vector x based on the Arnoldi Krylov subspace.

This function will update a solution vector x based on the previous solution x0 given the orthonormal basis and upper Hessenberg matrix formed in the Arnoldi algorithm. Updating is automatically called by the GMRESLP function. It is expected that the Arnoldi algorithm has already been called prior to calling this function.

**Parameters**

|                    |   |
|--------------------|---|
| <i>x</i>           | matrix that will hold the new updated solution to the linear system |
| <i>x0</i>          | matrix that holds the previous solution to the linear system        |
| <i>arnoldi_dat</i> | pointer to the <a href="#">ARNOLDI_DATA</a> data structure          |

**5.8.3.2 int arnoldi ( int (\*)(const Matrix< double > &v, Matrix< double > &w, const void \*data) matvec, int (\*)(const Matrix< double > &b, Matrix< double > &p, const void \*data) precon, Matrix< double > & r0, ARNOLDI\_DATA \* arnoldi\_dat, const void \* matvec\_data, const void \* precon\_data )**

Function to factor a linear operator into an orthonormal basis and upper Hessenberg matrix.

This function performs the Arnoldi algorithm to factor a linear operator into an orthonormal basis and upper Hessenberg matrix. Each orthonormal vector is formed using a Modified Gram-Schmidt procedure. When used in conjunction with GMRESLP, user may supply a preconditioning operator to improve convergence of the linear system. However, this function can be used by itself to factor the user's linear operator.

#### Parameters

|                    |  |
|--------------------|--|
| <i>matvec</i>      | user supplied linear operator given as an int function                                 |
| <i>precon</i>      | user supplied preconditioning operator given as an int function                        |
| <i>r0</i>          | user supplied vector to serve as the first basis vector in the orthonormal basis       |
| <i>arnoldi_dat</i> | pointer to the <a href="#">ARNOLDI_DATA</a> data structure                             |
| <i>matvec_data</i> | user supplied void pointer to a data structure needed for the linear operator          |
| <i>precon_data</i> | user supplied void pointer to a data structure needed for the preconditioning operator |

#### Note

int (\*matvec) (const [Matrix<double>](#) &v, [Matrix<double>](#) &Av, const void \*data)

-----  
 This is a user supplied function for a linear operator. User's function must return an int of 0 upon success and anything else denotes a failure. The function accepts a matrix v that will act on the linear operator a modified the matrix entries of Av to form the result of a matrix-vector product. Void pointer data is used to pass any user data structure that the function may need in order to perform the linear operation.  
 -----

int (\*precon) (const [Matrix<double>](#) &b, [Matrix<double>](#) &Mb, const void \*data)

-----  
 This is a user supplied function for a preconditioning operator. It has the same form as the above linear operator function and should have all the same properties. The only difference is that this function must form an approximate matrix inversion on the original linear operator and modify the entries of Mb to represent the result of that approximate matrix inversion. The matrix b is given as the vector that this operator is acting on and the void pointer data is for any user data structure that the operator may need.  
 -----

5.8.3.3 int gmresLeftPreconditioned ( int(\*) (const [Matrix< double >](#) &v, [Matrix< double >](#) &w, const void \*data) *matvec*,  
 int(\*) (const [Matrix< double >](#) &b, [Matrix< double >](#) &p, const void \*data) *precon*, [Matrix< double >](#) &b,  
[GMRESLP\\_DATA](#) \* *gmreslp\_dat*, const void \* *matvec\_data*, const void \* *precon\_data* )

Function to iteratively solve a non-symmetric, indefinite linear system with GMRESLP.

This function iteratively solves a non-symmetric, indefinite linear system using the Generalized Minimum RESidual method with Left Preconditioning (GMRESLP). It calls the Arnoldi algorithm to factor a linear operator into an orthonormal basis and upper Hessenberg matrix, then uses that factorization to form an approximation to the linear system. Because this algorithm uses left-side preconditioning, it can only check the linear residuals at the outer iterations.

#### Parameters

|                    |  |
|--------------------|--|
| <i>matvec</i>      | user supplied linear operator given as an int function                                 |
| <i>precon</i>      | user supplied preconditioning operator given as an int function                        |
| <i>b</i>           | matrix of boundary conditions in the linear system Ax=b                                |
| <i>gmreslp_dat</i> | pointer to the <a href="#">GMRESLP_DATA</a> data structure                             |
| <i>matvec_data</i> | user supplied void pointer to a data structure needed for the linear operator          |
| <i>precon_data</i> | user supplied void pointer to a data structure needed for the preconditioning operator |

## Note

```
int (*matvec) (const Matrix<double> & v, Matrix<double> &Av, const void *data)
```

-----

This is a user supplied function for a linear operator. User's function must return an int of 0 upon success and anything else denotes a failure. The function accepts a matrix v that will act on the linear operator a modified the matrix entries of Av to form the result of a matrix-vector product. Void pointer data is used to pass any user data structure that the function may need in order to perform the linear operation.

```
int (*precon) (const Matrix<double> & b, Matrix<double> &Mb, const void *data)
```

-----

This is a user supplied function for a preconditioning operator. It has the same form as the above linear operator function and should have all the same properties. The only difference is that this function must form an approximate matrix inversion on the original linear operator and modify the entries of Mb to represent the result of that approximate matrix inversion. The matrix b is given as the vector that this operator is acting on and the void pointer data is for any user data structure that the operator may need.

```
5.8.3.4 int fom ( int(*) (const Matrix< double > &v, Matrix< double > &w, const void *data) matvec, int(*) (const Matrix< double > &b, Matrix< double > &p, const void *data) precon, Matrix< double > & b, GMRESLP\_DATA * gmreslp_dat, const void * matvec_data, const void * precon_data )
```

Function to directly solve a non-symmetric, indefinite linear system with FOM.

This function directly solves a non-symmetric, indefinite linear system using the Full Orthogonalization Method (FOM). This algorithm is exactly equivalent to GMRESLP without restarting. Therefore, it uses the [GMRESLP\\_DATA](#) structure and calls the GMRESLP algorithm without using restarts. As a result, it never checks linear residuals. However, this should give the exact solution upon completion, assuming the linear operator is not singular.

## Parameters

|                    |  |
|--------------------|--|
| <i>matvec</i>      | user supplied linear operator given as an int function                                 |
| <i>precon</i>      | user supplied preconditioning operator given as an int function                        |
| <i>b</i>           | matrix of boundary conditions in the linear system $Ax=b$                              |
| <i>gmreslp_dat</i> | pointer to the <a href="#">GMRESLP_DATA</a> data structure                             |
| <i>matvec_data</i> | user supplied void pointer to a data structure needed for the linear operator          |
| <i>precon_data</i> | user supplied void pointer to a data structure needed for the preconditioning operator |

## Note

```
int (*matvec) (const Matrix<double> & v, Matrix<double> &Av, const void *data)
```

-----

This is a user supplied function for a linear operator. User's function must return an int of 0 upon success and anything else denotes a failure. The function accepts a matrix v that will act on the linear operator a modified the matrix entries of Av to form the result of a matrix-vector product. Void pointer data is used to pass any user data structure that the function may need in order to perform the linear operation.

```
int (*precon) (const Matrix<double> & b, Matrix<double> &Mb, const void *data)
```

-----

This is a user supplied function for a preconditioning operator. It has the same form as the above linear operator function and should have all the same properties. The only difference is that this function must form an approximate matrix inversion on the original linear operator and modify the entries of Mb to represent the result of that approximate matrix inversion. The matrix b is given as the vector that this operator is acting on and the void pointer data is for any user data structure that the operator may need.

**5.8.3.5** `int gmresRightPreconditioned ( int(*) (const Matrix< double > &v, Matrix< double > &w, const void *data) matvec, int(*) (const Matrix< double > &b, Matrix< double > &p, const void *data) precon, Matrix< double > &b, GMRESRP_DATA * gmresrp_dat, const void * matvec_data, const void * precon_data )`

Function to iteratively solve a non-symmetric, indefinite linear system with GMRESRP.

This function iteratively solves a non-symmetric, indefinite linear system using the Generalized Minimum RESidual method with Right Preconditioning (GMRESRP). Because this algorithm uses right preconditioning, it is able to check the linear residuals at both the outer and inner iterations. This may be much for efficient compared to GMRESLP. In order to check inner residuals, this algorithm has to perform it's own internal Modified Gram-Schmidt procedure and will not call the Arnoldi algorithm.

#### Parameters

|                    |  |
|--------------------|--|
| <i>matvec</i>      | user supplied linear operator given as an int function                                 |
| <i>precon</i>      | user supplied preconditioning operator given as an int function                        |
| <i>b</i>           | matrix of boundary conditions in the linear system $Ax=b$                              |
| <i>gmresrp_dat</i> | pointer to the <a href="#">GMRESRP_DATA</a> data structure                             |
| <i>matvec_data</i> | user supplied void pointer to a data structure needed for the linear operator          |
| <i>precon_data</i> | user supplied void pointer to a data structure needed for the preconditioning operator |

#### Note

`int (*matvec) (const Matrix<double> &v, Matrix<double> &Av, const void *data)`

-----  
This is a user supplied function for a linear operator. User's function must return an int of 0 upon success and anything else denotes a failure. The function accepts a matrix v that will act on the linear operator a modified the matrix entries of Av to form the result of a matrix-vector product. Void pointer data is used to pass any user data structure that the function may need in order to perform the linear operation.  
-----

`int (*precon) (const Matrix<double> &b, Matrix<double> &Mb, const void *data)`

-----  
This is a user supplied function for a preconditioning operator. It has the same form as the above linear operator function and should have all the same properties. The only difference is that this function must form an approximate matrix inversion on the original linear operator and modify the entries of Mb to represent the result of that approximate matrix inversion. The matrix b is given as the vector that this operator is acting on and the void pointer data is for any user data structure that the operator may need.  
-----

**5.8.3.6** `int pcg ( int(*) (const Matrix< double > &p, Matrix< double > &Ap, const void *data) matvec, int(*) (const Matrix< double > &r, Matrix< double > &z, const void *data) precon, Matrix< double > &b, PCG_DATA * pcg_dat, const void * matvec_data, const void * precon_data )`

Function to iteratively solve a symmetric, definite linear system with PCG.

This function iteratively solves a symmetric, definite linear system using the Preconditioned Conjugate Gradient (PCG) method. The PCG algorithm is optimal in terms of efficiency and residual reduction, but only if the linear system is symmetric. PCG will fail if the linear operator is non-symmetric!

#### Parameters

|                    |  |
|--------------------|--|
| <i>matvec</i>      | user supplied linear operator given as an int function                                 |
| <i>precon</i>      | user supplied preconditioning operator given as an int function                        |
| <i>b</i>           | matrix of boundary conditions in the linear system $Ax=b$                              |
| <i>pcg_dat</i>     | pointer to the <a href="#">PCG_DATA</a> data structure                                 |
| <i>matvec_data</i> | user supplied void pointer to a data structure needed for the linear operator          |
| <i>precon_data</i> | user supplied void pointer to a data structure needed for the preconditioning operator |

## Note

---

```
int (*matvec) (const Matrix<double> & v, Matrix<double> &Av, const void *data)
```

---

This is a user supplied function for a linear operator. User's function must return an int of 0 upon success and anything else denotes a failure. The function accepts a matrix v that will act on the linear operator a modified the matrix entries of Av to form the result of a matrix-vector product. Void pointer data is used to pass any user data structure that the function may need in order to perform the linear operation.

---

```
int (*precon) (const Matrix<double> & b, Matrix<double> &Mb, const void *data)
```

---

This is a user supplied function for a preconditioning operator. It has the same form as the above linear operator function and should have all the same properties. The only difference is that this function must form an approximate matrix inversion on the original linear operator and modify the entries of Mb to represent the result of that approximate matrix inversion. The matrix b is given as the vector that this operator is acting on and the void pointer data is for any user data structure that the operator may need.

---

```
5.8.3.7 int bicgstab ( int(*) (const Matrix< double > &p, Matrix< double > &Ap, const void *data) matvec,
int(*) (const Matrix< double > &r, Matrix< double > &z, const void *data) precon, Matrix< double > & b,
BiCGSTAB_DATA * bicg_dat, const void * matvec_data, const void * precon_data )
```

Function to iteratively solve a non-symmetric, definite linear system with BiCGSTAB.

This function iteratively solves a non-symmetric, definite linear system using the Bi-Conjugate Gradient STABILized (BiCGSTAB) method. This is a highly efficient algorithm for solving non-symmetric problems, but will occasionally breakdown and fail. Most common failures are caused by poor preconditioning. Works very well for grid-based linear systems.

## Parameters

|                    |  |
|--------------------|--|
| <i>matvec</i>      | user supplied linear operator given as an int function                                 |
| <i>precon</i>      | user supplied preconditioning operator given as an int function                        |
| <i>b</i>           | matrix of boundary conditions in the linear system $Ax=b$                              |
| <i>bicg_dat</i>    | pointer to the <a href="#">BiCGSTAB_DATA</a> data structure                            |
| <i>matvec_data</i> | user supplied void pointer to a data structure needed for the linear operator          |
| <i>precon_data</i> | user supplied void pointer to a data structure needed for the preconditioning operator |

## Note

---

```
int (*matvec) (const Matrix<double> & v, Matrix<double> &Av, const void *data)
```

---

This is a user supplied function for a linear operator. User's function must return an int of 0 upon success and anything else denotes a failure. The function accepts a matrix v that will act on the linear operator a modified the matrix entries of Av to form the result of a matrix-vector product. Void pointer data is used to pass any user data structure that the function may need in order to perform the linear operation.

---

```
int (*precon) (const Matrix<double> & b, Matrix<double> &Mb, const void *data)
```

---

This is a user supplied function for a preconditioning operator. It has the same form as the above linear operator function and should have all the same properties. The only difference is that this function must form an approximate matrix inversion on the original linear operator and modify the entries of Mb to represent the result of that approximate matrix inversion. The matrix b is given as the vector that this operator is acting on and the void pointer data is for any user data structure that the operator may need.

---

5.8.3.8 `int cgs ( int (*)(const Matrix< double > &p, Matrix< double > &Ap, const void *data) matvec, int (*)(const Matrix< double > &r, Matrix< double > &z, const void *data) precon, Matrix< double > &b, CGS_DATA * cgs_dat, const void * matvec_data, const void * precon_data )`

Function to iteratively solve a non-symmetric, definite linear system with CGS.

This function iteratively solves a non-symmetric, definite linear system using the Conjugate Gradient Squared (CGS) method. This is an extremely efficient algorithm for solving non-symmetric problems, but will often breakdown and fail. Most common failures are caused by poor or no preconditioning. Works very well for grid-based linear systems.

#### Parameters

|                    |  |
|--------------------|--|
| <i>matvec</i>      | user supplied linear operator given as an int function                                 |
| <i>precon</i>      | user supplied preconditioning operator given as an int function                        |
| <i>b</i>           | matrix of boundary conditions in the linear system $Ax=b$                              |
| <i>cgs_dat</i>     | pointer to the <a href="#">CGS_DATA</a> data structure                                 |
| <i>matvec_data</i> | user supplied void pointer to a data structure needed for the linear operator          |
| <i>precon_data</i> | user supplied void pointer to a data structure needed for the preconditioning operator |

#### Note

`int (*matvec) (const Matrix<double> &v, Matrix<double> &Av, const void *data)`

This is a user supplied function for a linear operator. User's function must return an int of 0 upon success and anything else denotes a failure. The function accepts a matrix v that will act on the linear operator a modified the matrix entries of Av to form the result of a matrix-vector product. Void pointer data is used to pass any user data structure that the function may need in order to perform the linear operation.

`int (*precon) (const Matrix<double> &b, Matrix<double> &Mb, const void *data)`

This is a user supplied function for a preconditioning operator. It has the same form as the above linear operator function and should have all the same properties. The only difference is that this function must form an approximate matrix inversion on the original linear operator and modify the entries of Mb to represent the result of that approximate matrix inversion. The matrix b is given as the vector that this operator is acting on and the void pointer data is for any user data structure that the operator may need.

5.8.3.9 `int operatorTranspose ( int (*)(const Matrix< double > &v, Matrix< double > &Av, const void *data) matvec, Matrix< double > &r, Matrix< double > &u, OPTRANS_DATA * transpose_dat, const void * matvec_data )`

Function that is used to perform transposition of a linear operator and results in a new vector  $A^T r = u$ .

This function takes a user supplied linear operator and forms the result of that operator transposed and multiplied by a given vector r ( $A^T r = u$ ). Transposition is accomplished by reordering the transpose operator and multiplying the non-transposed operator by a complete set of orthonormal vectors. The end result gives the ith component of the vector u for each operation ( $u_i = r^T A * i$ ). Here, i is a vector made from the ith column of the identity matrix. If the linear system is sufficiently large, then this operation may take some time.

#### Parameters

|                      |   |
|----------------------|---|
| <i>matvec</i>        | user supplied linear operator given as an int function                        |
| <i>r</i>             | vector to be multiplied by the transpose of the operator                      |
| <i>u</i>             | vector to store the result of the operator transposition ( $u=A^T r$ )        |
| <i>transpose_dat</i> | pointer to the <a href="#">OPTRANS_DATA</a> data structure                    |
| <i>matvec_data</i>   | user supplied void pointer to a data structure needed for the linear operator |

## Note

```
int (*matvec) (const Matrix<double> & v, Matrix<double> &Av, const void *data)
```

-----

This is a user supplied function for a linear operator. User's function must return an int of 0 upon success and anything else denotes a failure. The function accepts a matrix v that will act on the linear operator a modified the matrix entries of Av to form the result of a matrix-vector product. Void pointer data is used to pass any user data structure that the function may need in order to perform the linear operation.

-----

**5.8.3.10** `int gcr ( int (*)(const Matrix< double > &x, Matrix< double > &Ax, const void *data) matvec, int (*)(const Matrix< double > &r, Matrix< double > &Mr, const void *data) precon, Matrix< double > & b, GCR\_DATA * gcr_dat, const void * matvec.data, const void * precon.data )`

Function to iteratively solve a non-symmetric, definite linear system with GCR.

This function iteratively solves a non-symmetric, definite linear system using the Generalized Conjugate Residual (GCR) method. Similar to GMRESRP, this algorithm will construct a growing orthonormal basis set that will eventually form the exact solution to the linear system. However, this algorithm is less efficient than GMRESRP and can suffer breakdowns if the linear system is indefinite.

## Parameters

|                    |  |
|--------------------|--|
| <i>matvec</i>      | user supplied linear operator given as an int function                                 |
| <i>precon</i>      | user supplied preconditioning operator given as an int function                        |
| <i>b</i>           | matrix of boundary conditions in the linear system Ax=b                                |
| <i>gcr_dat</i>     | pointer to the <a href="#">GCR_DATA</a> data structure                                 |
| <i>matvec_data</i> | user supplied void pointer to a data structure needed for the linear operator          |
| <i>precon_data</i> | user supplied void pointer to a data structure needed for the preconditioning operator |

## Note

```
int (*matvec) (const Matrix<double> & v, Matrix<double> &Av, const void *data)
```

-----

This is a user supplied function for a linear operator. User's function must return an int of 0 upon success and anything else denotes a failure. The function accepts a matrix v that will act on the linear operator a modified the matrix entries of Av to form the result of a matrix-vector product. Void pointer data is used to pass any user data structure that the function may need in order to perform the linear operation.

-----

```
int (*precon) (const Matrix<double> & b, Matrix<double> &Mb, const void *data)
```

-----

This is a user supplied function for a preconditioning operator. It has the same form as the above linear operator function and should have all the same properties. The only difference is that this function must form an approximate matrix inversion on the original linear operator and modify the entries of Mb to represent the result of that approximate matrix inversion. The matrix b is given as the vector that this operator is acting on and the void pointer data is for any user data structure that the operator may need.

-----

**5.8.3.11** `int gmresPreconditioner ( const Matrix< double > & r, Matrix< double > & Mr, const void * data )`

Function used in conjunction with GMRESR to apply GMRESRP iterations as a preconditioner.

This function is required to take the form of the user supplied preconditioning functions for other iterative methods. However, it cannot be used in conjunction with any other Krylov method. It is only called by the GMRESR function when the preconditioner needs to be applied.

## Parameters

|             |  |
|-------------|--|
| <i>r</i>    | vector supplied to the preconditioner to operate on            |
| <i>Mr</i>   | vector to hold the result of the preconditioning operation     |
| <i>data</i> | void pointer to the <a href="#">GMRESR_DATA</a> data structure |

5.8.3.12 `int gmresr ( int (*)(const Matrix< double > &x, Matrix< double > &Ax, const void *data) matvec, int (*)(const Matrix< double > &r, Matrix< double > &Mr, const void *data) terminal_precon, Matrix< double > &b, GMRESR_DATA * gmresr_dat, const void * matvec_data, const void * term_precon_data )`

Function to iteratively solve a non-symmetric, indefinite linear system with GMRESR.

This function iteratively solves a non-symmetric, indefinite linear system using the Generalized Minimum RESidual Recursive (GMRESR) method. This algorithm actually uses GCR at the outer iterations, but stabilizes GCR with GMRESRP inner iterations to implicitly form a variable preconditioner to the linear system. As such, this is the only linear method that inherently includes preconditioning, without any user supplied preconditioning operator. However, this algorithm is significantly more computationally expensive than GCR or GMRESRP separately. It should only be used for solving very large or very hard to solve linear systems.

## Parameters

|                         |  |
|-------------------------|--|
| <i>matvec</i>           | user supplied linear operator given as an int function                                 |
| <i>terminal_precon</i>  | user supplied preconditioning operator given as an int function                        |
| <i>b</i>                | matrix of boundary conditions in the linear system $Ax=b$                              |
| <i>gmresr_dat</i>       | pointer to the <a href="#">GMRESR_DATA</a> data structure                              |
| <i>matvec_data</i>      | user supplied void pointer to a data structure needed for the linear operator          |
| <i>term_precon_data</i> | user supplied void pointer to a data structure needed for the preconditioning operator |

## Note

`int (*matvec) (const Matrix<double> &v, Matrix<double> &Av, const void *data)`

-----  
This is a user supplied function for a linear operator. User's function must return an int of 0 upon success and anything else denotes a failure. The function accepts a matrix v that will act on the linear operator a modified the matrix entries of Av to form the result of a matrix-vector product. Void pointer data is used to pass any user data structure that the function may need in order to perform the linear operation.

`int (*terminal_precon) (const Matrix<double> &b, Matrix<double> &Mb, const void *data)`

-----  
This is a user supplied function for a preconditioning operator. It has the same form as the above linear operator function and should have all the same properties. The only difference is that this function must form an approximate matrix inversion on the original linear operator and modify the entries of Mb to represent the result of that approximate matrix inversion. The matrix b is given as the vector that this operator is acting on and the void pointer data is for any user data structure that the operator may need.

5.8.3.13 `int picard ( int (*)(const Matrix< double > &x, Matrix< double > &r, const void *data) res, int (*)(const Matrix< double > &x0, Matrix< double > &x, const void *data) evalx, Matrix< double > &x, PICARD_DATA * picard_dat, const void * res_data, const void * evalx_data )`

Function to iteratively solve a non-linear system using the Picard or Fixed-Point method.

This function iteratively solves a non-linear system using the Picard method. User supplies a residual function and a weak solution form function. The weak form function is used to approximate the next solution vector for the non-linear system and the residual function is used to determine convergence. User also supplies an initial guess to the non-linear system as a matrix x, which will also be used to store the solution. This algorithm is very simple and may not be sufficient to solve complex non-linear systems.



## Parameters

|                   |   |
|-------------------|---|
| <i>res</i>        | user supplied function for the non-linear residuals of the system               |
| <i>evalx</i>      | user supplied function for the weak form to estimate the next solution          |
| <i>x</i>          | user supplied matrix holding the initial guess to the non-linear system         |
| <i>picard_dat</i> | pointer to the <a href="#">PICARD_DATA</a> data structure                       |
| <i>res_data</i>   | user supplied void pointer to a data structure used for residual evaluations    |
| <i>evalx_data</i> | user supplied void pointer to a data structure used for evaluation of weak form |

## Note

```
int (*res) (const Matrix<double> & x, Matrix<double> &F, const void *data)
```

-----

This is a user supplied function for the non-linear residuals. User's function must return an int of 0 upon success and anything else denotes a failure. The function accepts a matrix x representing the current non-linear variables. Those variables are used to evaluate the users functions and return the residuals in the matrix F. The void pointer data is a data structure provided by the user to hold information the function may need in order to form the residuals.

```
int (*evalx) (const Matrix<double> & x0, Matrix<double> &x, const void *data)
```

-----

This is a user supplied function to approximate the next solution vector x based on the previous solution vector x0. The x0 matrix is passed to this function and must be used to edit the entries of x based on the weak form of the problem. The user is free to define any weak form approximation. Void pointer data is the users data structure that may be used to pass additional information into this function in order to evaluate the weak form.

Example Residual:  $F(x) = x^2 + x - 1$  Goal is to make this function equal zero

Example Weak Form:  $x = 1 - x0^2$  Rearrange residual to form a weak solution

```
5.8.3.14 int jacvec ( const Matrix< double > & v, Matrix< double > & Jv, const void * data )
```

Function to form a linear operator of a Jacobian matrix used along with the PJFNK method.

This function is used in conjunction with the PJFNK routine to form a linear operator that a Krylov method can operate on. This linear operator is formed from the current residual vector of the non-linear iteration in PJFNK using a finite difference approximation.

Jacobian Linear Operator:  $J*v = ( F(x_k + eps*v) - F(x_k) ) / eps$

## Parameters

|             |  |
|-------------|--|
| <i>v</i>    | vector to be multiplied by the Jacobian matrix   |
| <i>Jv</i>   | storage vector for the result of the Jacobi-vector product                               |
| <i>data</i> | void pointer to the <a href="#">PJFNK_DATA</a> data structure holding solver information |

```
5.8.3.15 int backtrackLineSearch ( int(*) (const Matrix< double > &x, Matrix< double > &F, const void *data)
    feval, Matrix< double > & Fkp1, Matrix< double > & xkp1, Matrix< double > & pk, double normFk,
    BACKTRACK\_DATA * backtrack_dat, const void * feval_data )
```

Function to perform a Backtracking Line Search operation to smooth out convergence of PJFNK.

This function performs a simple backtracking line search operation on the residuals from the PJFNK method. The step size of the non-linear iteration is checked against a level of tolerance for residual reduction, then adjusted down if necessary. This method always starts out with the maximum allowable step size. If the largest step size is fine, then the algorithm does nothing. Otherwise, it iteratively adjusts the step size down, until a suitable step is found. In the case that no suitable step is found, this algorithm will report failure to the PJFNK method and PJFNK will decide

whether to continue trying to find a global minimum or report that it is stuck in a local minimum.

#### Parameters

|                      |   |
|----------------------|---|
| <i>feval</i>         | user supplied residual function for the non-linear system                       |
| <i>Fkp1</i>          | vector holding the residuals for the next non-linear step                       |
| <i>xkp1</i>          | vector holding the solution for the next non-linear step                        |
| <i>pk</i>            | vector holding the current non-linear search direction                          |
| <i>normFk</i>        | value of the current non-linear residual  |
| <i>backtrack_dat</i> | pointer to the <a href="#">BACKTRACK_DATA</a> data structure                    |
| <i>feval_data</i>    | user supplied void pointer to the data structure needed for residual evaluation |

#### Note

```
int (*feval) (const Matrix<double> &x, Matrix<double> &F, const void *data)
```

-----  
This is a user supplied function for the non-linear residuals. User's function must return an int of 0 upon success and anything else denotes a failure. The function accepts a matrix x representing the current non-linear variables. Those variables are used to evaluate the users functions and return the residuals in the matrix F. The void pointer data is a data structure provided by the user to hold information the function may need in order to form the residuals.  
-----

**5.8.3.16** `int pjfnk ( int(*) (const Matrix< double > &x, Matrix< double > &F, const void *data) res, int(*) (const Matrix< double > &r, Matrix< double > &p, const void *data) precon, Matrix< double > &x, PJFNK\_DATA * pjfnk_dat, const void * res_data, const void * precon_data )`

Function to perform the PJFNK algorithm to solve a non-linear system of equations.

This function solves a non-linear system of equations using the Preconditioned Jacobian- Free Newton-Krylov (PJFNK) algorithm. Each non-linear step of this method results in a linear sub-problem that is solved iteratively with one of the Krylov methods in the krylov\_method enum. User must supplied a residual function that computes the non-linear residuals of the system given the current state of the variables x. Additionally, the user must also supplied an initial guess to the non-linear system. Optionally, the user may supply a preconditioning function for the linear sub-problem.

Basic Steps: (i) Calc  $F(x_k)$ , (ii) Solve  $J(x_k)s_k = -F(x_k)$  for  $s_k$ , (iii) Form  $x_{kp1} = x_k + s_k$

#### Parameters

|                    |   |
|--------------------|---|
| <i>res</i>         | user supplied residual function for the non-linear system                     |
| <i>precon</i>      | user supplied preconditioning function for the linear sub-problems            |
| <i>x</i>           | user supplied initial guess and storage location of the solution              |
| <i>pjfnk_dat</i>   | pointer to the <a href="#">PJFNK_DATA</a> data structure                      |
| <i>res_data</i>    | user supplied void pointer to data structure used in residual function        |
| <i>precon_data</i> | user supplied void pointer to data structure used in preconditioning function |

#### Note

```
int (*res) (const Matrix<double> &x, Matrix<double> &F, const void *data)
```

-----  
This is a user supplied function for the non-linear residuals. User's function must return an int of 0 upon success and anything else denotes a failure. The function accepts a matrix x representing the current non-linear variables. Those variables are used to evaluate the users functions and return the residuals in the matrix F. The void pointer data is a data structure provided by the user to hold information the function may need in order to form the residuals.  
-----

```
int (*precon) (const Matrix<double> &b, Matrix<double> &Mb, const void *data)
```

---

This is a user supplied function for a preconditioning operator. It has the same form as the linear operators from the Krylov methods and should have all the same properties. The only difference is that this function must form an approximate matrix inversion on the jacvec linear operator and modify the entries of Mb to represent the result of that approximate matrix inversion. The matrix b is given as the vector that this operator is acting on and the void pointer data is for any user data structure that the operator may need.

---

**5.8.3.17** `int NumericalJacobian ( int(*) (const Matrix< double > &x, Matrix< double > &F, const void *user_data) Func, const Matrix< double > &x, Matrix< double > &J, int Nx, int Nf, NUM_JAC_DATA * jac_dat, const void * user_data )`

Function to form a full numerical Jacobian matrix from a given non-linear function.

This function uses finite differences to form a full rank Jacobian matrix for a user supplied non-linear function. The Jacobian matrix will be formed at the current state of the non-linear variables x and stored in a full matrix J. Integers Nx and Nf are used to determine the size of the Jacobian matrix.

#### Parameters

|                  |  |
|------------------|--|
| <i>Func</i>      | user supplied function for evaluation of the non-linear system                 |
| <i>x</i>         | matrix holding the current value of the non-linear variables                   |
| <i>J</i>         | matrix that will store the numerical Jacobian result                           |
| <i>Nx</i>        | number of non-linear variables in the system                                   |
| <i>Nf</i>        | number of non-linear functions in the system                                   |
| <i>jac_dat</i>   | pointer to the <a href="#">NUM_JAC_DATA</a> data structure                     |
| <i>user_data</i> | user supplied void pointer to a data structure used in the non-linear function |

**5.8.3.18** `int LARK.TESTS ( )`

Function that runs a variety of tests on all the functions in LARK.

This function runs a variety of tests on the linear and non-linear methods developed in LARK. It can be called from the UI.

## 5.9 macaw.h File Reference

MAtrix CAlculation Workspace.

```
#include <stdio.h>
#include <math.h>
#include <iostream>
#include <fstream>
#include <stdlib.h>
#include <vector>
#include <time.h>
#include <float.h>
#include <string>
#include <exception>
#include "error.h"
```

#### Classes

- class [Matrix< T >](#)

Templated C++ [Matrix](#) Class Object (click [Matrix](#) to go to function definitions)

## Macros

- `#define M_PI 3.14159265358979323846264338327950288`  
*Value of PI with double precision.*

## Functions

- `int MACAW_TESTS ()`  
*Function to run the MACAW tests.*

### 5.9.1 Detailed Description

Matrix CAIculation Workspace. `macaw.cpp`

This is a small C++ library that facilitates the use and construction of real matrices using vector objects. The [Matrix](#) class is templated so that users are able to work with matrices of any type including, but not limited to: (i) doubles, (ii) ints, (iii) floats, and (iv) even other matrices! Routines and functions are defined for Dense matrix operations. As a result, we typically only use Column Matrices (or Vectors) when doing any actual simulations. However, the development of this class was integral to the development and testing of the Sparse matrix operators in [lark.h](#).

While the primary goal of this object was to define how to operate on real matrices, we could extend this idea to complex matrices as well. For this, we could develop objects that represent imaginary and complex numbers and then create a [Matrix](#) of those objects. For this reason, the matrix operations here are all templated to abstract away the specificity of the type of matrix being operated on.

#### Author

Austin Ladshaw

#### Date

01/07/2014

#### Copyright

This software was designed and built at the Georgia Institute of Technology by Austin Ladshaw for PhD research in the area of adsorption and surface science. Copyright (c) 2015, all rights reserved.

### 5.9.2 Macro Definition Documentation

5.9.2.1 `#define M_PI 3.14159265358979323846264338327950288`

Value of PI with double precision.

### 5.9.3 Function Documentation

5.9.3.1 `int MACAW_TESTS ( )`

Function to run the MACAW tests.

This function is callable from the UI and is used to run several algorithm tests for the [Matrix](#) objects. This test should never report any errors.

## 5.10 magpie.h File Reference

Multicomponent Adsorption Generalized Procedure for Isothermal Equilibria.

```
#include "lmcurve.h"
#include <stdio.h>
#include <math.h>
#include <iostream>
#include <fstream>
#include <stdlib.h>
#include <vector>
#include <time.h>
#include <float.h>
#include <string>
#include "error.h"
```

### Classes

- struct [GSTA\\_DATA](#)  
*GSTA Data Structure.*
- struct [mSPD\\_DATA](#)  
*MSPD Data Structure.*
- struct [GPAST\\_DATA](#)  
*GPAST Data Structure.*
- struct [SYSTEM\\_DATA](#)  
*System Data Structure.*
- struct [MAGPIE\\_DATA](#)  
*MAGPIE Data Structure.*

### Macros

- #define [DBL\\_EPSILON](#) 2.2204460492503131e-016  
*Machine precision value used for approximating gradients.*
- #define [Z](#) 10.0  
*Surface coordination number used in the MSPD activity model.*
- #define [A](#) 3.13E+09  
*Corresponding van der Waals standard area for our coordination number (cm<sup>2</sup>/mol)*
- #define [V](#) 18.92  
*Corresponding van der Waals standard volume for our coordination number (cm<sup>3</sup>/mol)*
- #define [Po](#) 100.0  
*Standard State Pressure - Units: kPa.*
- #define [R](#) 8.3144621  
*Gas Constant - Units: J/(K\*mol) = kB \* Na.*
- #define [Na](#) 6.0221413E+23  
*Avagadro's Number - Units: molecules/mol.*
- #define [kB](#) 1.3806488E-23  
*Boltzmann's Constant - Units: J/K.*
- #define [shapeFactor](#)(v\_i) ( ( (Z - 2) \* v\_i ) / ( Z \* V ) ) + ( 2 / Z )  
*This macro replaces all instances of shapeFactor(#) with the following single line calculation.*
- #define [lnKo](#)(H, S, T) -( H / ( R \* T ) ) + ( S / R )  
*This macro calculates the natural log of the dimensionless isotherm parameter.*
- #define [He](#)(qm, K1, m) ( qm \* K1 ) / ( m \* Po )  
*This macro calculates the Henry's Coefficient for the ith component.*

## Functions

- double [qo](#) (double po, const void \*data, int i)  
*Function computes the result of the GSTA isotherm for the ith species.*
- double [dq\\_dp](#) (double p, const void \*data, int i)  
*Function computes the derivative of the GSTA model with respect to partial pressure.*
- double [q\\_p](#) (double p, const void \*data, int i)  
*Function computes the ratio between the adsorbed amount and partial pressure for the GSTA isotherm.*
- double [PI](#) (double po, const void \*data, int i)  
*Function computes the spreading pressure integral of the ith species.*
- double [Qst](#) (double po, const void \*data, int i)  
*Function computes the heat of adsorption based on the ith species GSTA parameters.*
- double [eMax](#) (const void \*data, int i)  
*Function to approximate the maximum lateral energy term for the ith species.*
- double [lnact\\_mSPD](#) (const double \*par, const void \*data, int i, volatile double [PI](#))  
*Function to evaluate the MSPD activity coefficient for the ith species.*
- double [grad\\_mSPD](#) (const double \*par, const void \*data, int i)  
*Function to approximate the derivative of the MSPD activity model with spreading pressure.*
- double [qT](#) (const double \*par, const void \*data)  
*Function to calculate the total adsorbed amount (mol/kg) for the mixed surface phase.*
- void [initialGuess\\_mSPD](#) (double \*par, const void \*data)  
*Function to provide an initial guess to the unknown parameters being solved for in GPAST.*
- void [eval\\_po\\_PI](#) (const double \*par, int m\_dat, const void \*data, double \*fvec, int \*info)  
*Function used with Imfit to evaluate the reference state pressure of a species based on spreading pressure.*
- void [eval\\_po\\_qo](#) (const double \*par, int m\_dat, const void \*data, double \*fvec, int \*info)  
*Function used with Imfit to evaluate the reference state pressure of a species based on that species isotherm.*
- void [eval\\_po](#) (const double \*par, int m\_dat, const void \*data, double \*fvec, int \*info)  
*Function used with Imfit to evaluate the reference state pressure of a species based on a sub-system.*
- void [eval\\_eta](#) (const double \*par, int m\_dat, const void \*data, double \*fvec, int \*info)  
*Function used with Imfit to evaluate the binary interaction parameters for each unique species pair.*
- void [eval\\_GPAST](#) (const double \*par, int m\_dat, const void \*data, double \*fvec, int \*info)  
*Function used with Imfit to solve the GPAST system of equations.*
- int [MAGPIE](#) (const void \*data)  
*Function to call all sub-routines to solve a MAGPIE/GPAST problem at a given temperature and pressure.*
- int [MAGPIE\\_SCENARIOS](#) (const char \*inputFileName, const char \*sceneFileName)  
*Function to perform a series of MAGPIE simulations based on given input files.*

### 5.10.1 Detailed Description

Multicomponent Adsorption Generalized Procedure for Isothermal Equilibria. [magpie.cpp](#)

This file contains all functions and routines associated with predicting isothermal adsorption equilibria from only single component isotherm information. The basis of the model is the Adsorbed Solution Theory developed by Myers and Prausnitz (1965). Added to that base model is a procedure by which we can predict the non-idealities present at the surface phase by solving a closed system of equations involving the activity model.

For more details on this procedure, check out our publication in AIChE where we give a fully feature explanation of our Generalized Predictive Adsorbed Solution Theory (GPAST).

Reference: Ladshaw, A., Yiaccoumi, S., and Tsouris, C., "A generalized procedure for the prediction of multicomponent adsorption equilibria", AIChE J., vol. 61, No. 8, p. 2600-2610, 2015.

MAGPIE represents a special case of the more general GPAST procedure, wherein the isotherm for each species is represent by the GSTA isotherm (see [gsta\\_opt.h](#)) and the activity model for non-ideality at the adsorbent surface is a Modified Spreading Pressure Dependent (MSPD) model. See the above paper reference for more details.

**Author**

Austin Ladshaw

**Date**

12/17/2013

**Copyright**

This software was designed and built at the Georgia Institute of Technology by Austin Ladshaw for PhD research in the area of adsorption and surface science. Copyright (c) 2015, all rights reserved.

**5.10.2 Macro Definition Documentation****5.10.2.1 #define DBL\_EPSILON 2.2204460492503131e-016**

Machine precision value used for approximating gradients.

**5.10.2.2 #define Z 10.0**

Surface coordination number used in the MSPD activity model.

**5.10.2.3 #define A 3.13E+09**

Corresponding van der Waals standard area for our coordination number ( $\text{cm}^2/\text{mol}$ )

**5.10.2.4 #define V 18.92**

Corresponding van der Waals standard volume for our coordination number ( $\text{cm}^3/\text{mol}$ )

**5.10.2.5 #define Po 100.0**

Standard State Pressure - Units: kPa.

**5.10.2.6 #define R 8.3144621**

Gas Constant - Units:  $\text{J}/(\text{K} \cdot \text{mol}) = \text{kB} * \text{Na}$ .

**5.10.2.7 #define Na 6.0221413E+23**

Avagadro's Number - Units: molecules/mol.

**5.10.2.8 #define kB 1.3806488E-23**

Boltzmann's Constant - Units: J/K.

**5.10.2.9 #define shapeFactor( v\_i ) ( ( ( Z - 2 ) \* v\_i ) / ( Z \* V ) ) + ( 2 / Z )**

This macro replaces all instances of shapeFactor(#) with the following single line calculation.

5.10.2.10 `#define lnKo( H, S, T )-( H / ( R * T ) )+( S / R )`

This macro calculates the natural log of the dimensionless isotherm parameter.

5.10.2.11 `#define He( qm, K1, m ) ( qm * K1 ) / ( m * Po )`

This macro calculates the Henry's Coefficient for the ith component.

### 5.10.3 Function Documentation

5.10.3.1 `double qo ( double po, const void * data, int i )`

Function computes the result of the GSTA isotherm for the ith species.

This function just computes the result of the GSTA isotherm model for the ith species given the partial pressure po.

#### Parameters

|             |  |
|-------------|--|
| <i>po</i>   | partial pressure in kPa at which to evaluate the GSTA model          |
| <i>data</i> | void pointer to the <a href="#">MAGPIE_DATA</a> data structure       |
| <i>i</i>    | index of the gas species for which the GSTA model is being evaluated |

5.10.3.2 `double dq.dp ( double p, const void * data, int i )`

Function computes the derivative of the GSTA model with respect to partial pressure.

This function just computes the result of the derivative of GSTA isotherm model for the ith species at the given the partial pressure p.

#### Parameters

|             |  |
|-------------|--|
| <i>p</i>    | partial pressure in kPa at which to evaluate the GSTA model          |
| <i>data</i> | void pointer to the <a href="#">MAGPIE_DATA</a> data structure       |
| <i>i</i>    | index of the gas species for which the GSTA model is being evaluated |

5.10.3.3 `double q.p ( double p, const void * data, int i )`

Function computes the ratio between the adsorbed amount and partial pressure for the GSTA isotherm.

This function just computes the ratio between the adsorbed amount q (mol/kg) and the partial pressure p (kPa) at the given partial pressure. If p == 0, then this function returns the Henry's Law constant for the isotherm of the ith species.

#### Parameters

|             |  |
|-------------|--|
| <i>p</i>    | partial pressure in kPa at which to evaluate the GSTA model          |
| <i>data</i> | void pointer to the <a href="#">MAGPIE_DATA</a> data structure       |
| <i>i</i>    | index of the gas species for which the GSTA model is being evaluated |

5.10.3.4 `double PI ( double po, const void * data, int i )`

Function computes the spreading pressure integral of the ith species.

This function uses an analytical solution to the spreading pressure integral with the GSTA isotherm to evaluate and return the value computed by that integral equation.



## Parameters

|             |  |
|-------------|--|
| <i>po</i>   | partial pressure in kPa at which to evaluate the lumped spreading pressure |
| <i>data</i> | void pointer to the <a href="#">MAGPIE_DATA</a> data structure             |
| <i>i</i>    | index of the gas species for which the GSTA model is being evaluated       |

5.10.3.5 double Qst ( double *po*, const void \* *data*, int *i* )

Function computes the heat of adsorption based on the *i*th species GSTA parameters.

This function computes the isosteric heat of adsorption (J/mol) for the GSTA parameters of the *i*th species.

## Parameters

|             |  |
|-------------|--|
| <i>po</i>   | partial pressure in kPa at which to evaluate the heat of adsorption  |
| <i>data</i> | void pointer to the <a href="#">MAGPIE_DATA</a> data structure       |
| <i>i</i>    | index of the gas species for which the GSTA model is being evaluated |

5.10.3.6 double eMax ( const void \* *data*, int *i* )

Function to approximate the maximum lateral energy term for the *i*th species.

The function attempts to approximate the maximum lateral energy term for the *i*th species. This is not a true maximum, but a cheaper estimate. Value being computed is used to shift the geometric mean and formulate the average cross-lateral energy term between species *i* and *j*.

5.10.3.7 double lnact\_mSPD ( const double \* *par*, const void \* *data*, int *i*, volatile double *PI* )

Function to evaluate the MSPD activity coefficient for the *i*th species.

This function will return the natural log of the *i*th species activity coefficient using the Modified Spreading Pressure Dependent (MSPD) activity model. The *par* argument holds the variable values being solved for by GPAST and their contents will change depending on whether we are doing a forward or reverse evaluation. This function should not be called by the user and will only be called when needed in the GPAST routine.

## Parameters

|             |  |
|-------------|--|
| <i>par</i>  | list of parameters representing variables to be solved for in GPAST        |
| <i>data</i> | void pointer for the <a href="#">MAGPIE_DATA</a> data structure            |
| <i>i</i>    | <i>i</i> th species that we want to calculate the activity coefficient for |
| <i>PI</i>   | lumped spreading pressure term used in gradient estimations                |

5.10.3.8 double grad\_mSPD ( const double \* *par*, const void \* *data*, int *i* )

Function to approximate the derivative of the MSPD activity model with spreading pressure.

This function returns a 2nd order, finite different approximation of the derivative of the MSPD activity model with the spreading pressure. The *par* argument will either hold the current iterates estimate of spreading pressure or should be passed as null. User does not need to call this function. GPAST will call automatically when needed.

## Parameters

|             |   |
|-------------|---|
| <i>par</i>  | list of parameters representing variables to be solved for in GPAST           |
| <i>data</i> | void pointer for the <a href="#">MAGPIE_DATA</a> data structure               |
| <i>i</i>    | <i>i</i> th species for which we will approximate the activity model gradient |

### 5.10.3.9 double qT ( const double \* *par*, const void \* *data* )

Function to calculate the total adsorbed amount (mol/kg) for the mixed surface phase.

This function will use the obtained system parameters from *par* and estimate the total amount of gases adsorbed to the surface in mol/kg. The user does not need to call this function, since this result will be stored in the [SYSTE-M\\_DATA](#) structure.

#### Parameters

|             |   |
|-------------|---|
| <i>par</i>  | list of parameters representing variables to be solved for in GPAST |
| <i>data</i> | void pointer for the <a href="#">MAGPIE_DATA</a> data structure     |

### 5.10.3.10 void initialGuess\_mSPD ( double \* *par*, const void \* *data* )

Function to provide an initial guess to the unknown parameters being solved for in GPAST.

This function intends to provide an initial guess for the unknown values being solved for in the GPAST system. Depending on what type of solve is requested, this algorithm will provide a guess for the adsorbed or gas phase composition.

#### Parameters

|             |   |
|-------------|---|
| <i>par</i>  | list of parameters representing variables to be solved for in GPAST |
| <i>data</i> | void pointer for the <a href="#">MAGPIE_DATA</a> data structure     |

### 5.10.3.11 void eval\_po\_Pi ( const double \* *par*, int *m\_dat*, const void \* *data*, double \* *fvec*, int \* *info* )

Function used with *lmfit* to evaluate the reference state pressure of a species based on spreading pressure.

This function is used inside of the MSPD activity model to calculate the reference state pressure of a particular species at a given spreading pressure for the system. User does not need to call this function. GPAST will call automatically when needed.

#### Parameters

|              |   |
|--------------|---|
| <i>par</i>   | list of parameters representing variables to be solved for in GPAST |
| <i>m_dat</i> | number of functions/variables in the GPAST system of equations      |
| <i>data</i>  | void pointer for the <a href="#">MAGPIE_DATA</a> data structure     |
| <i>fvec</i>  | list of residuals formed by the functions in GPAST                  |
| <i>info</i>  | integer flag variable used in the <i>lmfit</i> routine              |

### 5.10.3.12 void eval\_po\_qo ( const double \* *par*, int *m\_dat*, const void \* *data*, double \* *fvec*, int \* *info* )

Function used with *lmfit* to evaluate the reference state pressure of a species based on that species isotherm.

This function is used to evaluate the partial pressure or reference state pressure for a particular species given single-component adsorbed amount. User does not need to call this function. GPAST will call automatically when needed.

#### Parameters

|              |   |
|--------------|---|
| <i>par</i>   | list of parameters representing variables to be solved for in GPAST |
| <i>m_dat</i> | number of functions/variables in the GPAST system of equations      |
| <i>data</i>  | void pointer for the <a href="#">MAGPIE_DATA</a> data structure     |
| <i>fvec</i>  | list of residuals formed by the functions in GPAST                  |
| <i>info</i>  | integer flag variable used in the <i>lmfit</i> routine              |

5.10.3.13 void eval\_po ( const double \* *par*, int *m\_dat*, const void \* *data*, double \* *fvec*, int \* *info* )

Function used with Imfit to evaluate the reference state pressure of a species based on a sub-system.

This function is used to approximate reference state pressures based on the spreading pressure of a sub-system in GPAST. The sub-system will be one of the unique binary systems that exist in the overall mixed gas system. User does not need to call this function. GPAST will call automatically when needed.

#### Parameters

|              |   |
|--------------|---|
| <i>par</i>   | list of parameters representing variables to be solved for in GPAST |
| <i>m_dat</i> | number of functions/variables in the GPAST system of equations      |
| <i>data</i>  | void pointer for the <a href="#">MAGPIE_DATA</a> data structure     |
| <i>fvec</i>  | list of residuals formed by the functions in GPAST                  |
| <i>info</i>  | integer flag variable used in the Imfit routine                     |

5.10.3.14 void eval\_eta ( const double \* *par*, int *m\_dat*, const void \* *data*, double \* *fvec*, int \* *info* )

Function used with Imfit to evaluate the binary interaction parameters for each unique species pair.

This function is used to estimate the binary interaction parameters for all species pairs in a given sub-system. Those parameters are then stored for later used when evaluating the activity coefficients for the overall mixture. User does not need to call this function. GPAST will call automatically when needed.

#### Parameters

|              |   |
|--------------|---|
| <i>par</i>   | list of parameters representing variables to be solved for in GPAST |
| <i>m_dat</i> | number of functions/variables in the GPAST system of equations      |
| <i>data</i>  | void pointer for the <a href="#">MAGPIE_DATA</a> data structure     |
| <i>fvec</i>  | list of residuals formed by the functions in GPAST                  |
| <i>info</i>  | integer flag variable used in the Imfit routine                     |

5.10.3.15 void eval\_GPAST ( const double \* *par*, int *m\_dat*, const void \* *data*, double \* *fvec*, int \* *info* )

Function used with Imfit to solve the GPAST system of equations.

This function is used after having calculated and stored all necessary information to solve a closed form GPAST system of equations. User does not need to call this function. GPAST will call automatically when needed.

#### Parameters

|              |   |
|--------------|---|
| <i>par</i>   | list of parameters representing variables to be solved for in GPAST |
| <i>m_dat</i> | number of functions/variables in the GPAST system of equations      |
| <i>data</i>  | void pointer for the <a href="#">MAGPIE_DATA</a> data structure     |
| <i>fvec</i>  | list of residuals formed by the functions in GPAST                  |
| <i>info</i>  | integer flag variable used in the Imfit routine                     |

5.10.3.16 int MAGPIE ( const void \* *data* )

Function to call all sub-routines to solve a MAGPIE/GPAST problem at a given temperature and pressure.

This is the function that a typical user will want to incorporate into their own codes when evaluating adsorption of a gas mixture. Prior to calling this function, all required structures and information in the [MAGPIE\\_DATA](#) structure must have been properly initialized. After this function has completed it's operations, it will return an integer used to denote a success or failure of the routine. Integers 0, 1, 2, and 3 all denote success. Anything else is considered a failure.

To setup the [MAGPIE\\_DATA](#) structure correctly, you must reserve space for all vector objects based on the number of gas species in the mixture. In general, you only need to reserve space for the adsorbing species. However, you can also reserve space for non-adsorbing species, but you **MUST** give a gas/adsorbed mole fraction of the non-adsorbing species 0.0 so that the routine knows to ignore them (very important)!

After setting up the memory for the vector objects, you can initialize information specific to the simulation you want to request. The number of species (N), total pressure (PT) and gas temperature (T) must always be given. You can neglect the non-idealities of the surface phase by setting the Ideal bool to true. This will result in faster calculations, because MAGPIE will just revert down to the Ideal Adsorbed Solution Theory (IAST).

The Recover bool will denote whether we are doing a forward or reverse GPAST evaluation. Forward evaluation is for solving for the composition of the adsorbed phase given the composition of the gas phase (Recover = false). Reverse evaluation is for solve for the composition of the gas phase given the composition of the adsorbed phase (Recover = true).

For a reverse evaluation (Recover = true) you will also need to stipulate whether or not there is a carrier gas (Carrier = true or false). A carrier gas is considered any non-adsorbing species that may be present in the gas phase and contributing to the total pressure in the system.

The parameters that must be initialized for all species include all [GSTA\\_DATA](#) parameters and the van der Waals volume parameter (v) in the [mSPD\\_DATA](#) structure. For non-adsorbing species, you can ignore these parameters, but need to set the sites (m) from [GSTA\\_DATA](#) to 1. GPAST cannot run any evaluations without these parameters being set properly AND set in the same order for all species (i.e., make sure that `gpast_dat[i].qmax` corresponds to `mspd_dat[i].v` and so on).

Lastly, you need to give either the gas phase or adsorbed phase mole fractions, depending on whether you are going to run a forward or reverse evaluation, respectively. For a forward evaluation, provide the gas mole fractions (y) in [GPAST\\_DATA](#) for each species (non-adsorbing species should have this value set to 0.0). For a reverse evaluation, provide the adsorbed mole fractions (x) in [GPAST\\_DATA](#) for each species, as well as the total adsorbed amount (qT) in [SYSTEM\\_DATA](#). Again, non-adsorbing species should have their respective phase mole fractions set to 0.0 to exclude them from the simulation. Additionally, if there are non-adsorbing species present, then the Carrier bool in [SYSTEM\\_DATA](#) must be set to true.

#### Parameters

|             |   |
|-------------|---|
| <i>data</i> | void pointer for the <a href="#">MAGPIE_DATA</a> data structure holding all necessary information |
|-------------|---|

#### 5.10.3.17 int MAGPIE\_SCENARIOS ( const char \* *inputFileName*, const char \* *sceneFileName* )

Function to perform a series of MAGPIE simulations based on given input files.

This function is callable from the UI and is used to perform a series of isothermal equilibria evaluations using the MAGPIE routines. There are two input files that must be provided: (i) *inputFileName* - containing parameter information for the species and (ii) *sceneFileName* - containing information for each MAGPIE simulation. Each of these files have a specific structure (see below). NOTE: this may change in future versions.

#### *inputFileName* Text File Structure:

Integer for Number of Adsorbing Species

van der Waals Volume ( $\text{cm}^3/\text{mol}$ ) of ith species

GSTA adsorption capacity (mol/kg) of ith species

Number of GSTA parameters of ith species

Enthalpy (J/mol) of nth site (tab) Entropy of nth site (J/K/mol) of ith species

(repeat above for all n sites in species i)

(repeat above for all species i)

**Example Input File:**

```
5
17.1
5.8797
1
-20351.9 -81.8369
16.2
5.14934
1
-16662.7 -74.4766
19.7
9.27339
4
-46597.5 -53.6994
-125024 -221.073
-193619 -356.728
-272228 -567.459
13.25
4.59144
1
-13418.5 -84.888
18.0
10.0348
1
-20640.4 -72.6119
```

(The above input file gives the parameter information for 5 adsorbing species)

**sceneFileName Text File Structure:**

Integer Flag to mark Forward (0) or { Reverse (1) evaluations }

Number of Simulations to Run

Total Pressure (kPa) (tab) Temperature (K) { (tab) Total Adsorption (mol/kg) (tab) Carrier Gas Flag (0=false, 1=true) }

Gas/Adsorbed Mole Fractions for each species in the order given in prior file (tab separated)

(repeat above for all simulations desired)

NOTE: only provide the Total Adsorption and Carrier Flag if doing Reverse evaluations!

**Example Scenario File 1:**

```
0
4
```

```
0.65 303.15
0.364 0.318 0.318
3.25 303.15
0.371 0.32 0.309
6.85 303.15
0.388 0.299 0.313
13.42 303.15
0.349 0.326 0.325
```

(The above scenario file is for 4 forward evaluations/simulations for a 3-adsorbing species system)

#### Example Scenario File 2:

```
1
4
0.65 303.15 5.4 0
0.364 0.318 0.318
3.25 303.15 7.7 0
0.371 0.32 0.309
6.85 303.15 9.8 0
0.388 0.299 0.313
13.42 303.15 10.4 0
0.349 0.326 0.325
```

(The above scenario file is for 4 reverse evaluations/simulations for a 3-adsorbing species system and no carrier gas)

## 5.11 mola.h File Reference

[Molecule](#) Object Library from Atoms.

```
#include <ctype.h>
#include "eel.h"
```

### Classes

- class [Molecule](#)

*C++ [Molecule](#) Object built from [Atom](#) Objects (click [Molecule](#) to go to function definitions)*

### Functions

- int [MOLA\\_TESTS](#) ()

*Function to run the MOLA tests.*

### 5.11.1 Detailed Description

**Molecule** Object Library from Atoms. mola.cpp

This file contains a C++ Class for creating **Molecule** objects from the **Atom** objects that were defined in **eel.h**. Molecules can be created and registered from basic information or can be registered from a growing list of pre-registered molecules that are accessible by name/formula.

Registered Molecules are known and defined prior to runtime. They have a charge, energy characteristics, phase, name, and formula that they are recognized by. The formula is used to create the atoms that they are made from. If some information is incomplete, it must be specified as to what information is missing (i.e. denote whether the formation energies are known).

Formation energies are used to determine stability/dissociation/acidity equilibrium constants during runtime. If the formation energies are unknown, then the equilibrium constants must be given to a reaction object on when it is initialized.

The molecule formula's are given as strings which are parsed in the constructor to determine what atoms from the EEL files will be registered and used. Note, you will be able to build molecules from an input file, but the library molecules here are ready to be used in applications and require no more input other than the molecule's formula.

#### List of Currently Registered Molecules

CO3 2- (aq)

Cl - (aq)

H2O (l)

H + (aq)

H2CO3 (aq)

HCO3 - (aq)

HNO3 (aq)

HCl (aq)

NaHCO3 (aq)

NaCO3 - (aq)

Na + (aq)

NaCl (aq)

NaOH (aq)

NO3 - (aq)

OH - (aq)

UO2 2+ (aq)

UO2NO3 + (aq)

UO2(NO3)2 (aq)

UO2OH + (aq)

UO2(OH)2 (aq)

UO2(OH)3 - (aq)

UO2(OH)4 2- (aq)

(UO2)2OH 3+ (aq)

(UO2)2(OH)2 2+ (aq)

(UO2)3(OH)4 2+ (aq)

$(\text{UO}_2)_3(\text{OH})_5 + (\text{aq})$

$(\text{UO}_2)_3(\text{OH})_7^- (\text{aq})$

$(\text{UO}_2)_4(\text{OH})_7 + (\text{aq})$

$\text{UO}_2\text{CO}_3 (\text{aq})$

$\text{UO}_2(\text{CO}_3)_2^{2-} (\text{aq})$

$\text{UO}_2(\text{CO}_3)_3^{4-} (\text{aq})$

Those registered molecules follow a strict naming convention by which they can be recognized (see below)...

#### Naming Convention

Plus (+) and minus (-) charges are denoted by the numeric value of the charge followed by a + or - sign, respectively ( e.g.  $\text{UO}_2(\text{CO}_3)_3^{4-} (\text{aq})$  )

The phase is always denoted last and will be marked as (l) for liquid, (s) for solid, (aq) for aqueous, and (g) for gas (see above).

When registering a molecule that is not in the library, you must also provide a linear formula during construction or registration. This is needed so that the string parsing is easier to handle when the molecule subsequently registers the necessary atoms. (e.g.  $\text{UO}_2(\text{CO}_3)_3 = \text{UO}_2\text{C}_3\text{O}_9$  or  $\text{UO}_{11}\text{C}_3$ ).

#### Author

Austin Ladshaw

#### Date

02/24/2014

#### Copyright

This software was designed and built at the Georgia Institute of Technology by Austin Ladshaw for PhD research in the area of adsorption and surface science. Copyright (c) 2015, all rights reserved.

## 5.11.2 Function Documentation

### 5.11.2.1 `int MOLA_TESTS ( )`

Function to run the MOLA tests.

This function is callable from the UI and is used to run several algorithm tests for the [Molecule](#) objects. This test should never report any errors.

## 5.12 monkfish.h File Reference

```
#include "dogfish.h"
```

#### Classes

- struct [MONKFISH\\_PARAM](#)
- struct [MONKFISH\\_DATA](#)



## Functions

- double [default\\_porosity](#) (int *i*, int *l*, const void \**user\_data*)
- double [default\\_density](#) (int *i*, int *l*, const void \**user\_data*)
- double [default\\_interparticle\\_diffusion](#) (int *i*, int *l*, const void \**user\_data*)
- double [default\\_monk\\_adsorption](#) (int *i*, int *l*, const void \**user\_data*)
- double [default\\_monk\\_equilibrium](#) (int *i*, int *l*, const void \**user\_data*)
- double [default\\_monkfish\\_retardation](#) (int *i*, int *l*, const void \**user\_data*)
- double [default\\_exterior\\_concentration](#) (int *i*, const void \**user\_data*)
- double [default\\_film\\_transfer](#) (int *i*, const void \**user\_data*)
- int [setup\\_MONKFISH\\_DATA](#) (FILE \**file*, double(\*eval\_porosity)(int *i*, int *l*, const void \**user\_data*), double(\*eval\_density)(int *i*, int *l*, const void \**user\_data*), double(\*eval\_ext\_diff)(int *i*, int *l*, const void \**user\_data*), double(\*eval\_adsorb)(int *i*, int *l*, const void \**user\_data*), double(\*eval\_retard)(int *i*, int *l*, const void \**user\_data*), double(\*eval\_ext\_conc)(int *i*, const void \**user\_data*), double(\*eval\_ext\_film)(int *i*, const void \**user\_data*), double(\*dog\_diffusion)(int *i*, int *l*, const void \**user\_data*), double(\*dog\_ext\_film)(int *i*, const void \**user\_data*), double(\*dog\_surf\_conc)(int *i*, const void \**user\_data*), const void \**user\_data*, [MONKFISH\\_DATA](#) \**monk\_dat*)
- int [MONKFISH\\_TESTS](#) ()

### 5.12.1 Function Documentation

5.12.1.1 double [default\\_porosity](#) ( int *i*, int *l*, const void \* *user\_data* )

5.12.1.2 double [default\\_density](#) ( int *i*, int *l*, const void \* *user\_data* )

5.12.1.3 double [default\\_interparticle\\_diffusion](#) ( int *i*, int *l*, const void \* *user\_data* )

5.12.1.4 double [default\\_monk\\_adsorption](#) ( int *i*, int *l*, const void \* *user\_data* )

5.12.1.5 double [default\\_monk\\_equilibrium](#) ( int *i*, int *l*, const void \* *user\_data* )

5.12.1.6 double [default\\_monkfish\\_retardation](#) ( int *i*, int *l*, const void \* *user\_data* )

5.12.1.7 double [default\\_exterior\\_concentration](#) ( int *i*, const void \* *user\_data* )

5.12.1.8 double [default\\_film\\_transfer](#) ( int *i*, const void \* *user\_data* )

5.12.1.9 int [setup\\_MONKFISH\\_DATA](#) ( FILE \* *file*, double(\*) (int *i*, int *l*, const void \**user\_data*) *eval\_porosity*, double(\*) (int *i*, int *l*, const void \**user\_data*) *eval\_density*, double(\*) (int *i*, int *l*, const void \**user\_data*) *eval\_ext\_diff*, double(\*) (int *i*, int *l*, const void \**user\_data*) *eval\_adsorb*, double(\*) (int *i*, int *l*, const void \**user\_data*) *eval\_retard*, double(\*) (int *i*, const void \**user\_data*) *eval\_ext\_conc*, double(\*) (int *i*, const void \**user\_data*) *eval\_ext\_film*, double(\*) (int *i*, int *l*, const void \**user\_data*) *dog\_diffusion*, double(\*) (int *i*, const void \**user\_data*) *dog\_ext\_film*, double(\*) (int *i*, const void \**user\_data*) *dog\_surf\_conc*, const void \* *user\_data*, [MONKFISH\\_DATA](#) \* *monk\_dat* )

5.12.1.10 int [MONKFISH\\_TESTS](#) ( )

## 5.13 sandbox.h File Reference

```
#include "flock.h"
#include "school.h"
```

## Classes

- struct [Speciation\\_Test01\\_Data](#)

## Functions

- int [Speciation\\_Test01\\_Function](#) (const [Matrix](#)< double > &x, [Matrix](#)< double > &F, const void \*res\_data)
- int [Speciation\\_Test01\\_Jacobian](#) (const [Matrix](#)< double > &x, [Matrix](#)< double > &J, const void \*precon\_data)
- int [Speciation\\_Test01\\_Guess](#) (const void \*user\_data)
- int [Speciation\\_Test01\\_MatVec](#) (const [Matrix](#)< double > &x, [Matrix](#)< double > &Ax, const void \*matvec\_data)
- int [RUN\\_SANDBOX](#) ()

### 5.13.1 Function Documentation

5.13.1.1 int [Speciation\\_Test01\\_Function](#) ( const [Matrix](#)< double > & x, [Matrix](#)< double > & F, const void \* *res\_data* )

5.13.1.2 int [Speciation\\_Test01\\_Jacobian](#) ( const [Matrix](#)< double > & x, [Matrix](#)< double > & J, const void \* *precon\_data* )

5.13.1.3 int [Speciation\\_Test01\\_Guess](#) ( const void \* *user\_data* )

5.13.1.4 int [Speciation\\_Test01\\_MatVec](#) ( const [Matrix](#)< double > & x, [Matrix](#)< double > & Ax, const void \* *matvec\_data* )

5.13.1.5 int [RUN\\_SANDBOX](#) ( )

## 5.14 school.h File Reference

```
#include "eel.h"
#include "mola.h"
#include "shark.h"
#include "dogfish.h"
#include "monkfish.h"
#include "yaml_wrapper.h"
```

## 5.15 scopsowl.h File Reference

```
#include "egret.h"
#include "skua.h"
```

## Classes

- struct [SCOPSOWL\\_PARAM\\_DATA](#)
- struct [SCOPSOWL\\_DATA](#)

## Macros

- #define [SCOPSOWL\\_HPP\\_](#)
- #define [Dp](#)(Dm, ep) (ep\*ep\*Dm)
- #define [Dk](#)(rp, T, MW) (9700.0\*rp\*pow((T/MW),0.5))
- #define [avgDp](#)(Dp, Dk) (pow(((1/Dp)+(1/Dk)),-1.0))

## Functions

- void [print2file\\_species\\_header](#) (FILE \*Output, [SCOPSOWL\\_DATA](#) \*owl\_dat, int i)
- void [print2file\\_SCOPSOWL\\_time\\_header](#) (FILE \*Output, [SCOPSOWL\\_DATA](#) \*owl\_dat, int i)
- void [print2file\\_SCOPSOWL\\_header](#) ([SCOPSOWL\\_DATA](#) \*owl\_dat)
- void [print2file\\_SCOPSOWL\\_result\\_old](#) ([SCOPSOWL\\_DATA](#) \*owl\_dat)
- void [print2file\\_SCOPSOWL\\_result\\_new](#) ([SCOPSOWL\\_DATA](#) \*owl\_dat)
- double [default\\_adsorption](#) (int i, int l, const void \*user\_data)
- double [default\\_retardation](#) (int i, int l, const void \*user\_data)
- double [default\\_pore\\_diffusion](#) (int i, int l, const void \*user\_data)
- double [default\\_surf\\_diffusion](#) (int i, int l, const void \*user\_data)
- double [default\\_effective\\_diffusion](#) (int i, int l, const void \*user\_data)
- double [const\\_pore\\_diffusion](#) (int i, int l, const void \*user\_data)
- double [default\\_filmMassTransfer](#) (int i, const void \*user\_data)
- double [const\\_filmMassTransfer](#) (int i, const void \*user\_data)
- int [setup\\_SCOPSOWL\\_DATA](#) (FILE \*file, double(\*eval\_sorption)(int i, int l, const void \*user\_data), double(\*eval\_retardation)(int i, int l, const void \*user\_data), double(\*eval\_pore\_diff)(int i, int l, const void \*user\_data), double(\*eval\_filmMT)(int i, const void \*user\_data), double(\*eval\_surface\_diff)(int i, int l, const void \*user\_data), const void \*user\_data, [MIXED\\_GAS](#) \*gas\_data, [SCOPSOWL\\_DATA](#) \*owl\_data)
- int [SCOPSOWL\\_Executioner](#) ([SCOPSOWL\\_DATA](#) \*owl\_dat)
- int [set\\_SCOPSOWL\\_ICs](#) ([SCOPSOWL\\_DATA](#) \*owl\_dat)
- int [set\\_SCOPSOWL\\_timestep](#) ([SCOPSOWL\\_DATA](#) \*owl\_dat)
- int [SCOPSOWL\\_preprocesses](#) ([SCOPSOWL\\_DATA](#) \*owl\_dat)
- int [set\\_SCOPSOWL\\_params](#) (const void \*user\_data)
- int [SCOPSOWL\\_postprocesses](#) ([SCOPSOWL\\_DATA](#) \*owl\_dat)
- int [SCOPSOWL\\_reset](#) ([SCOPSOWL\\_DATA](#) \*owl\_dat)
- int [SCOPSOWL](#) ([SCOPSOWL\\_DATA](#) \*owl\_dat)
- int [LARGE\\_CYCLE\\_TEST01](#) ([SCOPSOWL\\_DATA](#) \*owl\_dat)
- int [SMALL\\_CYCLE\\_TEST02](#) ([SCOPSOWL\\_DATA](#) \*owl\_dat)
- int [CURVE\\_TEST03](#) ([SCOPSOWL\\_DATA](#) \*owl\_dat)
- int [CURVE\\_TEST04](#) ([SCOPSOWL\\_DATA](#) \*owl\_dat)
- int [CURVE\\_TEST05](#) ([SCOPSOWL\\_DATA](#) \*owl\_dat)
- int [SCOPSOWL\\_SCENARIOS](#) (const char \*scene, const char \*sorberent, const char \*comp, const char \*sorbate)
- int [SCOPSOWL\\_TESTS](#) ()

### 5.15.1 Macro Definition Documentation

5.15.1.1 [#define SCOPSOWL\\_HPP\\_](#)

5.15.1.2 [#define Dp\( Dm, ep \)\(ep\\*ep\\*Dm\)](#)

5.15.1.3 [#define Dk\( rp, T, MW \)\(9700.0\\*rp\\*pow\(\(T/MW\),0.5\)\)](#)

5.15.1.4 [#define avgDp\( Dp, Dk \)\(pow\(\(\(1/Dp\)+\(1/Dk\)\), -1.0\)\)](#)

### 5.15.2 Function Documentation

5.15.2.1 [void print2file\\_species\\_header \( FILE \\* Output, SCOPSOWL\\_DATA \\* owl\\_dat, int i \)](#)

5.15.2.2 [void print2file\\_SCOPSOWL\\_time\\_header \( FILE \\* Output, SCOPSOWL\\_DATA \\* owl\\_dat, int i \)](#)

5.15.2.3 [void print2file\\_SCOPSOWL\\_header \( SCOPSOWL\\_DATA \\* owl\\_dat \)](#)

---

```

5.15.2.4 void print2file_SCOPSOWL_result_old ( SCOPSOWL_DATA * owl_dat )

5.15.2.5 void print2file_SCOPSOWL_result_new ( SCOPSOWL_DATA * owl_dat )

5.15.2.6 double default_adsorption ( int i, int l, const void * user_data )

5.15.2.7 double default_retardation ( int i, int l, const void * user_data )

5.15.2.8 double default_pore_diffusion ( int i, int l, const void * user_data )

5.15.2.9 double default_surf_diffusion ( int i, int l, const void * user_data )

5.15.2.10 double default_effective_diffusion ( int i, int l, const void * user_data )

5.15.2.11 double const_pore_diffusion ( int i, int l, const void * user_data )

5.15.2.12 double default_filmMassTransfer ( int i, const void * user_data )

5.15.2.13 double const_filmMassTransfer ( int i, const void * user_data )

5.15.2.14 int setup_SCOPSOWL_DATA ( FILE * file, double(*) (int i, int l, const void *user_data) eval_sorption, double(*) (int i,
int l, const void *user_data) eval_retardation, double(*) (int i, int l, const void *user_data) eval_pore_diff, double(*) (int
i, const void *user_data) eval_filmMT, double(*) (int i, int l, const void *user_data) eval_surface_diff, const void *
user_data, MIXED_GAS * gas_data, SCOPSOWL_DATA * owl_data )

5.15.2.15 int SCOPSOWL_Executioner ( SCOPSOWL_DATA * owl_dat )

5.15.2.16 int set_SCOPSOWL_ICs ( SCOPSOWL_DATA * owl_dat )

5.15.2.17 int set_SCOPSOWL_timestep ( SCOPSOWL_DATA * owl_dat )

5.15.2.18 int SCOPSOWL_preprocesses ( SCOPSOWL_DATA * owl_dat )

5.15.2.19 int set_SCOPSOWL_params ( const void * user_data )

5.15.2.20 int SCOPSOWL_postprocesses ( SCOPSOWL_DATA * owl_dat )

5.15.2.21 int SCOPSOWL_reset ( SCOPSOWL_DATA * owl_dat )

5.15.2.22 int SCOPSOWL ( SCOPSOWL_DATA * owl_dat )

5.15.2.23 int LARGE_CYCLE_TEST01 ( SCOPSOWL_DATA * owl_dat )

5.15.2.24 int SMALL_CYCLE_TEST02 ( SCOPSOWL_DATA * owl_dat )

5.15.2.25 int CURVE_TEST03 ( SCOPSOWL_DATA * owl_dat )

5.15.2.26 int CURVE_TEST04 ( SCOPSOWL_DATA * owl_dat )

5.15.2.27 int CURVE_TEST05 ( SCOPSOWL_DATA * owl_dat )

5.15.2.28 int SCOPSOWL_SCENARIOS ( const char * scene, const char * sorbent, const char * comp, const char * sorbate )

5.15.2.29 int SCOPSOWL_TESTS ( )

```

---

## 5.16 scopsowl\_opt.h File Reference

```
#include "scopsowl.h"
```

### Classes

- struct [SCOPSOWL\\_OPT\\_DATA](#)

### Functions

- int [SCOPSOWL\\_OPT\\_set\\_y](#) ([SCOPSOWL\\_OPT\\_DATA](#) \*owl\_opt)
- int [initial\\_guess\\_SCOPSOWL](#) ([SCOPSOWL\\_OPT\\_DATA](#) \*owl\_opt)
- void [eval\\_SCOPSOWL\\_Uptake](#) (const double \*par, int m\_dat, const void \*data, double \*fvec, int \*info)
- int [SCOPSOWL\\_OPTIMIZE](#) (const char \*scene, const char \*sorbent, const char \*comp, const char \*sorbate, const char \*data)

#### 5.16.1 Function Documentation

5.16.1.1 int [SCOPSOWL\\_OPT\\_set\\_y](#) ( [SCOPSOWL\\_OPT\\_DATA](#) \* owl\_opt )

5.16.1.2 int [initial\\_guess\\_SCOPSOWL](#) ( [SCOPSOWL\\_OPT\\_DATA](#) \* owl\_opt )

5.16.1.3 void [eval\\_SCOPSOWL\\_Uptake](#) ( const double \* par, int m\_dat, const void \* data, double \* fvec, int \* info )

5.16.1.4 int [SCOPSOWL\\_OPTIMIZE](#) ( const char \* scene, const char \* sorbent, const char \* comp, const char \* sorbate, const char \* data )

## 5.17 shark.h File Reference

```
#include "mola.h"  
#include "macaw.h"  
#include "lark.h"  
#include "yaml_wrapper.h"
```

### Classes

- class [MasterSpeciesList](#)
- class [Reaction](#)
- class [MassBalance](#)
- class [UnsteadyReaction](#)
- class [Mechanism](#)
- class [Precipitation](#)
- class [UnsteadyPrecipitation](#)
- struct [SHARK\\_DATA](#)

### Macros

- #define [Rstd](#) 8.3144621

## Typedefs

- typedef struct [SHARK\\_DATA](#) [SHARK\\_DATA](#)

## Enumerations

- enum [valid\\_act](#) {  
[IDEAL](#), [DAVIES](#), [DEBYE\\_HUCKEL](#), [DAVIES\\_LADSHAW](#),  
[SIT](#), [PITZER](#) }

## Functions

- void [print2file\\_shark\\_info](#) ([SHARK\\_DATA](#) \*shark\_dat)
- void [print2file\\_shark\\_header](#) ([SHARK\\_DATA](#) \*shark\_dat)
- void [print2file\\_shark\\_results\\_new](#) ([SHARK\\_DATA](#) \*shark\_dat)
- void [print2file\\_shark\\_results\\_old](#) ([SHARK\\_DATA](#) \*shark\_dat)
- int [ideal\\_solution](#) (const [Matrix](#)< double > &x, [Matrix](#)< double > &F, const void \*data)
- int [Davies\\_equation](#) (const [Matrix](#)< double > &x, [Matrix](#)< double > &F, const void \*data)
- int [DebyeHuckel\\_equation](#) (const [Matrix](#)< double > &x, [Matrix](#)< double > &F, const void \*data)
- int [DaviesLadshaw\\_equation](#) (const [Matrix](#)< double > &x, [Matrix](#)< double > &F, const void \*data)
- int [act\\_choice](#) (const std::string &input)
- bool [linesearch\\_choice](#) (const std::string &input)
- int [linearsolve\\_choice](#) (const std::string &input)
- int [Convert2LogConcentration](#) (const [Matrix](#)< double > &x, [Matrix](#)< double > &logx)
- int [Convert2Concentration](#) (const [Matrix](#)< double > &logx, [Matrix](#)< double > &x)
- int [read\\_scenario](#) ([SHARK\\_DATA](#) \*shark\_dat)
- int [read\\_options](#) ([SHARK\\_DATA](#) \*shark\_dat)
- int [read\\_species](#) ([SHARK\\_DATA](#) \*shark\_dat)
- int [read\\_massbalance](#) ([SHARK\\_DATA](#) \*shark\_dat)
- int [read\\_equilrxn](#) ([SHARK\\_DATA](#) \*shark\_dat)
- int [read\\_unsteadyrxn](#) ([SHARK\\_DATA](#) \*shark\_dat)
- int [setup\\_SHARK\\_DATA](#) (FILE \*file, int(\*residual)(const [Matrix](#)< double > &x, [Matrix](#)< double > &res, const void \*data), int(\*activity)(const [Matrix](#)< double > &x, [Matrix](#)< double > &gama, const void \*data), int(\*precond)(const [Matrix](#)< double > &r, [Matrix](#)< double > &p, const void \*data), [SHARK\\_DATA](#) \*dat, const void \*activity\_data, const void \*residual\_data, const void \*precon\_data, const void \*other\_data)
- int [shark\\_add\\_customResidual](#) (int i, double(\*other\_res)(const [Matrix](#)< double > &x, [SHARK\\_DATA](#) \*shark\_dat, const void \*other\_data), [SHARK\\_DATA](#) \*shark\_dat)
- int [shark\\_parameter\\_check](#) ([SHARK\\_DATA](#) \*shark\_dat)
- int [shark\\_energy\\_calculations](#) ([SHARK\\_DATA](#) \*shark\_dat)
- int [shark\\_temperature\\_calculations](#) ([SHARK\\_DATA](#) \*shark\_dat)
- int [shark\\_pH\\_finder](#) ([SHARK\\_DATA](#) \*shark\_dat)
- int [shark\\_guess](#) ([SHARK\\_DATA](#) \*shark\_dat)
- int [shark\\_initial\\_conditions](#) ([SHARK\\_DATA](#) \*shark\_dat)
- int [shark\\_executioner](#) ([SHARK\\_DATA](#) \*shark\_dat)
- int [shark\\_timestep\\_const](#) ([SHARK\\_DATA](#) \*shark\_dat)
- int [shark\\_timestep\\_adapt](#) ([SHARK\\_DATA](#) \*shark\_dat)
- int [shark\\_preprocesses](#) ([SHARK\\_DATA](#) \*shark\_dat)
- int [shark\\_solver](#) ([SHARK\\_DATA](#) \*shark\_dat)
- int [shark\\_postprocesses](#) ([SHARK\\_DATA](#) \*shark\_dat)
- int [shark\\_reset](#) ([SHARK\\_DATA](#) \*shark\_dat)
- int [shark\\_residual](#) (const [Matrix](#)< double > &x, [Matrix](#)< double > &F, const void \*data)
- int [SHARK](#) ([SHARK\\_DATA](#) \*shark\_dat)
- int [SHARK\\_SCENARIO](#) (const char \*yaml\_input)
- int [SHARK\\_TESTS](#) ()

### 5.17.1 Macro Definition Documentation

5.17.1.1 `#define Rstd 8.3144621`

### 5.17.2 Typedef Documentation

5.17.2.1 `typedef struct SHARK_DATA SHARK_DATA`

### 5.17.3 Enumeration Type Documentation

5.17.3.1 `enum valid_act`

Enumerator

***IDEAL***

***DAVIES***

***DEBYE\_HUCKEL***

***DAVIES\_LADSHAW***

***SIT***

***PITZER***

### 5.17.4 Function Documentation

5.17.4.1 `void print2file_shark_info ( SHARK_DATA * shark_dat )`

5.17.4.2 `void print2file_shark_header ( SHARK_DATA * shark_dat )`

5.17.4.3 `void print2file_shark_results_new ( SHARK_DATA * shark_dat )`

5.17.4.4 `void print2file_shark_results_old ( SHARK_DATA * shark_dat )`

5.17.4.5 `int ideal_solution ( const Matrix< double > & x, Matrix< double > & F, const void * data )`

5.17.4.6 `int Davies_equation ( const Matrix< double > & x, Matrix< double > & F, const void * data )`

5.17.4.7 `int DebyeHuckel_equation ( const Matrix< double > & x, Matrix< double > & F, const void * data )`

5.17.4.8 `int DaviesLadshaw_equation ( const Matrix< double > & x, Matrix< double > & F, const void * data )`

5.17.4.9 `int act_choice ( const std::string & input )`

5.17.4.10 `bool linesearch_choice ( const std::string & input )`

5.17.4.11 `int linearsolve_choice ( const std::string & input )`

5.17.4.12 `int Convert2LogConcentration ( const Matrix< double > & x, Matrix< double > & logx )`

5.17.4.13 `int Convert2Concentration ( const Matrix< double > & logx, Matrix< double > & x )`

5.17.4.14 `int read_scenario ( SHARK_DATA * shark_dat )`

5.17.4.15 `int read_options ( SHARK_DATA * shark_dat )`

5.17.4.16 `int read_species ( SHARK_DATA * shark_dat )`

- 5.17.4.17 int read\_massbalance ( SHARK\_DATA \* shark\_dat )
- 5.17.4.18 int read\_equilrxn ( SHARK\_DATA \* shark\_dat )
- 5.17.4.19 int read\_unsteadyrxn ( SHARK\_DATA \* shark\_dat )
- 5.17.4.20 int setup\_SHARK\_DATA ( FILE \* file, int(\*) (const Matrix< double > &x, Matrix< double > &res, const void \*data) residual, int(\*) (const Matrix< double > &x, Matrix< double > &gama, const void \*data) activity, int(\*) (const Matrix< double > &r, Matrix< double > &p, const void \*data) precon, SHARK\_DATA \* dat, const void \* activity\_data, const void \* residual\_data, const void \* precon\_data, const void \* other\_data )
- 5.17.4.21 int shark\_add\_customResidual ( int i, double(\*) (const Matrix< double > &x, SHARK\_DATA \* shark\_dat, const void \*other\_data) other\_res, SHARK\_DATA \* shark\_dat )
- 5.17.4.22 int shark\_parameter\_check ( SHARK\_DATA \* shark\_dat )
- 5.17.4.23 int shark\_energy\_calculations ( SHARK\_DATA \* shark\_dat )
- 5.17.4.24 int shark\_temperature\_calculations ( SHARK\_DATA \* shark\_dat )
- 5.17.4.25 int shark\_pH\_finder ( SHARK\_DATA \* shark\_dat )
- 5.17.4.26 int shark\_guess ( SHARK\_DATA \* shark\_dat )
- 5.17.4.27 int shark\_initial\_conditions ( SHARK\_DATA \* shark\_dat )
- 5.17.4.28 int shark\_executioner ( SHARK\_DATA \* shark\_dat )
- 5.17.4.29 int shark\_timestep\_const ( SHARK\_DATA \* shark\_dat )
- 5.17.4.30 int shark\_timestep\_adapt ( SHARK\_DATA \* shark\_dat )
- 5.17.4.31 int shark\_preprocesses ( SHARK\_DATA \* shark\_dat )
- 5.17.4.32 int shark\_solver ( SHARK\_DATA \* shark\_dat )
- 5.17.4.33 int shark\_postprocesses ( SHARK\_DATA \* shark\_dat )
- 5.17.4.34 int shark\_reset ( SHARK\_DATA \* shark\_dat )
- 5.17.4.35 int shark\_residual ( const Matrix< double > &x, Matrix< double > &F, const void \* data )
- 5.17.4.36 int SHARK ( SHARK\_DATA \* shark\_dat )
- 5.17.4.37 int SHARK\_SCENARIO ( const char \* yaml\_input )
- 5.17.4.38 int SHARK\_TESTS ( )

## 5.18 skua.h File Reference

```
#include "finch.h"
#include "magpie.h"
#include "egret.h"
```



## Classes

- struct [SKUA\\_PARAM](#)
- struct [SKUA\\_DATA](#)

## Macros

- `#define SKUA_HPP_`
- `#define D_inf(Dref, Tref, B, p, T) ( Dref * pow(p+sqrt(DBL_EPSILON),(Tref/T)-B) )`
- `#define D_o(Diff, E, T) ( Diff * exp(-E/(Rstd*T)) )`
- `#define D_c(Diff, phi) ( Diff * (1.0/((1.0+1.1E-6)-phi)) )`

## Functions

- void [print2file\\_species\\_header](#) (FILE \*Output, [SKUA\\_DATA](#) \*skua\_dat, int i)
- void [print2file\\_SKUA\\_time\\_header](#) (FILE \*Output, [SKUA\\_DATA](#) \*skua\_dat, int i)
- void [print2file\\_SKUA\\_header](#) ([SKUA\\_DATA](#) \*skua\_dat)
- void [print2file\\_SKUA\\_results\\_old](#) ([SKUA\\_DATA](#) \*skua\_dat)
- void [print2file\\_SKUA\\_results\\_new](#) ([SKUA\\_DATA](#) \*skua\_dat)
- double [default\\_Dc](#) (int i, int l, const void \*data)
- double [default\\_kf](#) (int i, const void \*data)
- double [const\\_Dc](#) (int i, int l, const void \*data)
- double [simple\\_darken\\_Dc](#) (int i, int l, const void \*data)
- double [theoretical\\_darken\\_Dc](#) (int i, int l, const void \*data)
- double [empirical\\_kf](#) (int i, const void \*data)
- double [const\\_kf](#) (int i, const void \*data)
- int [molefractionCheck](#) ([SKUA\\_DATA](#) \*skua\_dat)
- int [setup\\_SKUA\\_DATA](#) (FILE \*file, double(\*eval\_Dc)(int i, int l, const void \*user\_data), double(\*eval\_Kf)(int i, const void \*user\_data), const void \*user\_data, [MIXED\\_GAS](#) \*gas\_data, [SKUA\\_DATA](#) \*skua\_dat)
- int [SKUA\\_Executioner](#) ([SKUA\\_DATA](#) \*skua\_dat)
- int [set\\_SKUA\\_ICs](#) ([SKUA\\_DATA](#) \*skua\_dat)
- int [set\\_SKUA\\_timestep](#) ([SKUA\\_DATA](#) \*skua\_dat)
- int [SKUA\\_preprocesses](#) ([SKUA\\_DATA](#) \*skua\_dat)
- int [set\\_SKUA\\_params](#) (const void \*user\_data)
- int [SKUA\\_postprocesses](#) ([SKUA\\_DATA](#) \*skua\_dat)
- int [SKUA\\_reset](#) ([SKUA\\_DATA](#) \*skua\_dat)
- int [SKUA](#) ([SKUA\\_DATA](#) \*skua\_dat)
- int [SKUA\\_CYCLE\\_TEST01](#) ([SKUA\\_DATA](#) \*skua\_dat)
- int [SKUA\\_CYCLE\\_TEST02](#) ([SKUA\\_DATA](#) \*skua\_dat)
- int [SKUA\\_LOW\\_TEST03](#) ([SKUA\\_DATA](#) \*skua\_dat)
- int [SKUA\\_MID\\_TEST04](#) ([SKUA\\_DATA](#) \*skua\_dat)
- int [SKUA\\_SCENARIOS](#) (const char \*scene, const char \*sorbent, const char \*comp, const char \*sorbate)
- int [SKUA\\_TESTS](#) ()

### 5.18.1 Macro Definition Documentation

#### 5.18.1.1 `#define SKUA_HPP_`

#### 5.18.1.2 `#define D_inf( Dref, Tref, B, p, T ) ( Dref * pow(p+sqrt(DBL_EPSILON),(Tref/T)-B) )`

#### 5.18.1.3 `#define D_o( Diff, E, T ) ( Diff * exp(-E/(Rstd*T)) )`

#### 5.18.1.4 `#define D_c( Diff, phi ) ( Diff * (1.0/((1.0+1.1E-6)-phi)) )`

## 5.18.2 Function Documentation

- 5.18.2.1 void print2file\_species\_header ( FILE \* *Output*, SKUA\_DATA \* *skua\_dat*, int *i* )
- 5.18.2.2 void print2file\_SKUA\_time\_header ( FILE \* *Output*, SKUA\_DATA \* *skua\_dat*, int *i* )
- 5.18.2.3 void print2file\_SKUA\_header ( SKUA\_DATA \* *skua\_dat* )
- 5.18.2.4 void print2file\_SKUA\_results\_old ( SKUA\_DATA \* *skua\_dat* )
- 5.18.2.5 void print2file\_SKUA\_results\_new ( SKUA\_DATA \* *skua\_dat* )
- 5.18.2.6 double default\_Dc ( int *i*, int *l*, const void \* *data* )
- 5.18.2.7 double default\_kf ( int *i*, const void \* *data* )
- 5.18.2.8 double const\_Dc ( int *i*, int *l*, const void \* *data* )
- 5.18.2.9 double simple\_darken\_Dc ( int *i*, int *l*, const void \* *data* )
- 5.18.2.10 double theoretical\_darken\_Dc ( int *i*, int *l*, const void \* *data* )
- 5.18.2.11 double empirical\_kf ( int *i*, const void \* *data* )
- 5.18.2.12 double const\_kf ( int *i*, const void \* *data* )
- 5.18.2.13 int molefractionCheck ( SKUA\_DATA \* *skua\_dat* )
- 5.18.2.14 int setup\_SKUA\_DATA ( FILE \* *file*, double(\*) (int *i*, int *l*, const void \* *user\_data*) *eval\_Dc*, double(\*) (int *i*, const void \* *user\_data*) *eval\_Kf*, const void \* *user\_data*, MIXED\_GAS \* *gas\_data*, SKUA\_DATA \* *skua\_dat* )
- 5.18.2.15 int SKUA\_Executioner ( SKUA\_DATA \* *skua\_dat* )
- 5.18.2.16 int set\_SKUA\_ICs ( SKUA\_DATA \* *skua\_dat* )
- 5.18.2.17 int set\_SKUA\_timestep ( SKUA\_DATA \* *skua\_dat* )
- 5.18.2.18 int SKUA\_preprocesses ( SKUA\_DATA \* *skua\_dat* )
- 5.18.2.19 int set\_SKUA\_params ( const void \* *user\_data* )
- 5.18.2.20 int SKUA\_postprocesses ( SKUA\_DATA \* *skua\_dat* )
- 5.18.2.21 int SKUA\_reset ( SKUA\_DATA \* *skua\_dat* )
- 5.18.2.22 int SKUA ( SKUA\_DATA \* *skua\_dat* )
- 5.18.2.23 int SKUA\_CYCLE\_TEST01 ( SKUA\_DATA \* *skua\_dat* )
- 5.18.2.24 int SKUA\_CYCLE\_TEST02 ( SKUA\_DATA \* *skua\_dat* )
- 5.18.2.25 int SKUA\_LOW\_TEST03 ( SKUA\_DATA \* *skua\_dat* )
- 5.18.2.26 int SKUA\_MID\_TEST04 ( SKUA\_DATA \* *skua\_dat* )
- 5.18.2.27 int SKUA\_SCENARIOS ( const char \* *scene*, const char \* *sorbent*, const char \* *comp*, const char \* *sorbate* )

5.18.2.28 int SKUA\_TESTS ( )

## 5.19 skua\_opt.h File Reference

```
#include "skua.h"
```

### Classes

- struct [SKUA\\_OPT\\_DATA](#)

### Functions

- int [SKUA\\_OPT\\_set\\_y](#) ([SKUA\\_OPT\\_DATA](#) \*skua\_opt)
- int [initial\\_guess\\_SKUA](#) ([SKUA\\_OPT\\_DATA](#) \*skua\_opt)
- void [eval\\_SKUA\\_Uptake](#) (const double \*par, int m\_dat, const void \*data, double \*fvec, int \*info)
- int [SKUA\\_OPTIMIZE](#) (const char \*scene, const char \*sorbent, const char \*comp, const char \*sorbate, const char \*data)

### 5.19.1 Function Documentation

5.19.1.1 int SKUA\_OPT\_set\_y ( [SKUA\\_OPT\\_DATA](#) \* skua\_opt )

5.19.1.2 int initial\_guess\_SKUA ( [SKUA\\_OPT\\_DATA](#) \* skua\_opt )

5.19.1.3 void eval.SKUA\_Uptake ( const double \* par, int m\_dat, const void \* data, double \* fvec, int \* info )

5.19.1.4 int SKUA\_OPTIMIZE ( const char \* scene, const char \* sorbent, const char \* comp, const char \* sorbate, const char \* data )

## 5.20 Trajectory.h File Reference

```
#include "macaw.h"
#include <random>
#include <chrono>
```

### Classes

- struct [TRAJECTORY\\_DATA](#)

### Functions

- double [Magnetic\\_R](#) (const [Matrix](#)< double > &dX, const [Matrix](#)< double > &dY, int i, double b, double mu\_0, double chi\_p, double M, double H0, double a)
- double [Magnetic\\_T](#) (const [Matrix](#)< double > &dX, const [Matrix](#)< double > &dY, int i, double b, double mu\_0, double chi\_p, double M, double H0, double a)
- double [Grav\\_R](#) (const [Matrix](#)< double > &dX, int i, double b, double rho\_p, double rho\_f)
- double [Grav\\_T](#) (const [Matrix](#)< double > &dX, int i, double b, double rho\_p, double rho\_f)
- double [Van\\_R](#) (const [Matrix](#)< double > &dX, const [Matrix](#)< double > &dY, int i, double Hamaker, double b, double a)

- double **V\_RAD** (const **Matrix**< double > &dX, const **Matrix**< double > &dY, int i, double V0, double rho\_f, double a, double eta)
- double **V\_THETA** (const **Matrix**< double > &dX, const **Matrix**< double > &dY, int i, double V0, double rho\_f, double a, double eta)
- double **Brown\_RAD** (double n\_rand, double m\_rand, double sigma\_n, double sigma\_m)
- double **Brown\_THETA** (double s\_rand, double t\_rand, double sigma\_n, double sigma\_m)
- int **POLAR** (**Matrix**< double > &POL, const **Matrix**< double > &dX, const **Matrix**< double > &dY, const void \*data, int i)
- double **RADIAL\_FORCE** (const **Matrix**< double > &POL, double eta, double b, double mp, double t, double a)
- double **TANGENTIAL\_FORCE** (const **Matrix**< double > &POL, const **Matrix**< double > &dY, double eta, double b, double mp, double t, double a, int i)
- int **CARTESIAN** (const **Matrix**< double > &POL, **Matrix**< double > &H, const **Matrix**< double > &dY, double i, const void \*data)
- int **DISPLACEMENT** (**Matrix**< double > &dX, **Matrix**< double > &dY, const **Matrix**< double > &H, int i)
- int **LOCATION** (const **Matrix**< double > &dY, const **Matrix**< double > &dX, **Matrix**< double > &X, **Matrix**< double > &Y, int i)
- double **Removal\_Efficiency** (double Sum\_Cap, const void \*data)
- int **Trajectory\_SetupConstants** (**TRAJECTORY\_DATA** \*dat)
- int **Number\_Generator** (**TRAJECTORY\_DATA** \*dat)
- int **Run\_Trajectory** ()

### 5.20.1 Function Documentation

- 5.20.1.1 double **Magnetic\_R** ( const **Matrix**< double > &dX, const **Matrix**< double > &dY, int i, double b, double mu\_0, double chi\_p, double M, double H0, double a )
- 5.20.1.2 double **Magnetic\_T** ( const **Matrix**< double > &dX, const **Matrix**< double > &dY, int i, double b, double mu\_0, double chi\_p, double M, double H0, double a )
- 5.20.1.3 double **Grav\_R** ( const **Matrix**< double > &dX, int i, double b, double rho\_p, double rho\_f )
- 5.20.1.4 double **Grav\_T** ( const **Matrix**< double > &dX, int i, double b, double rho\_p, double rho\_f )
- 5.20.1.5 double **Van\_R** ( const **Matrix**< double > &dX, const **Matrix**< double > &dY, int i, double Hamaker, double b, double a )
- 5.20.1.6 double **V\_RAD** ( const **Matrix**< double > &dX, const **Matrix**< double > &dY, int i, double V0, double rho\_f, double a, double eta )
- 5.20.1.7 double **V\_THETA** ( const **Matrix**< double > &dX, const **Matrix**< double > &dY, int i, double V0, double rho\_f, double a, double eta )
- 5.20.1.8 double **Brown\_RAD** ( double n\_rand, double m\_rand, double sigma\_n, double sigma\_m )
- 5.20.1.9 double **Brown\_THETA** ( double s\_rand, double t\_rand, double sigma\_n, double sigma\_m )
- 5.20.1.10 int **POLAR** ( **Matrix**< double > &POL, const **Matrix**< double > &dX, const **Matrix**< double > &dY, const void \* data, int i )
- 5.20.1.11 double **RADIAL\_FORCE** ( const **Matrix**< double > &POL, double eta, double b, double mp, double t, double a )
- 5.20.1.12 double **TANGENTIAL\_FORCE** ( const **Matrix**< double > &POL, const **Matrix**< double > &dY, double eta, double b, double mp, double t, double a, int i )

- 5.20.1.13 `int CARTESIAN ( const Matrix< double > & POL, Matrix< double > & H, const Matrix< double > & dY, double i, const void * data )`
- 5.20.1.14 `int DISPLACEMENT ( Matrix< double > & dX, Matrix< double > & dY, const Matrix< double > & H, int i )`
- 5.20.1.15 `int LOCATION ( const Matrix< double > & dY, const Matrix< double > & dX, Matrix< double > & X, Matrix< double > & Y, int i )`
- 5.20.1.16 `double Removal_Efficiency ( double Sum_Cap, const void * data )`
- 5.20.1.17 `int Trajectory_SetupConstants ( TRAJECTORY_DATA * dat )`
- 5.20.1.18 `int Number_Generator ( TRAJECTORY_DATA * dat )`
- 5.20.1.19 `int Run_Trajectory ( )`

## 5.21 ui.h File Reference

User Interface for Ecosystem.

```
#include <fstream>
#include <string>
#include <iostream>
#include "error.h"
#include "yaml_wrapper.h"
#include "flock.h"
#include "school.h"
#include "sandbox.h"
#include "Trajectory.h"
```

### Classes

- struct [UI\\_DATA](#)  
*Data structure holding the UI arguments.*

### Macros

- #define [UI\\_HPP\\_](#)
- #define [ECO\\_VERSION](#) "0.0 alpha"  
*Macro expansion for executable current version number.*
- #define [ECO\\_EXECUTABLE](#) "eco0"  
*Macro expansion for executable current name.*

### Enumerations

- enum [valid\\_options](#) {  
[TEST](#), [EXECUTE](#), [EXIT](#), [CONTINUE](#),  
[HELP](#), [dogfish](#), [eel](#), [egret](#),  
[finch](#), [lark](#), [macaw](#), [mola](#),  
[monkfish](#), [sandbox](#), [scopsowl](#), [shark](#),  
[skua](#), [gsta\\_opt](#), [magpie](#), [scops\\_opt](#),  
[skua\\_opt](#), [trajectory](#) }  
*Valid options available upon execution of the code.*

## Functions

- void `au_i_help` ()  
*Function to display help for Advanced User Interface.*
- void `bui_help` ()  
*Function to display help for Basic User Interface.*
- std::string `allLower` (const std::string &input)  
*Function to return an all lower case string based on the passed argument.*
- bool `exit` (const std::string &input)  
*Function returns true if user requests exit.*
- bool `help` (const std::string &input)  
*Function returns true if the user requests help.*
- bool `version` (const std::string &input)  
*Function returns true if user requests to know the executable version.*
- bool `test` (const std::string &input)  
*Function returns true if user requests to run a test.*
- bool `exec` (const std::string &input)  
*Function returns true if the user requests to run a simulation/executable.*
- bool `path` (const std::string &input)  
*Function returns true if the user indicates that input files share a common path.*
- bool `input` (const std::string &input)  
*Function returns true if the user indicates that the next arguments are input files.*
- bool `valid_test_string` (const std::string &input, UI\_DATA \*ui\_dat)  
*Function returns true if the user gave a valid test option.*
- bool `valid_exec_string` (const std::string &input, UI\_DATA \*ui\_dat)  
*Function returns true if the user gave a valid execution option.*
- int `number_files` (UI\_DATA \*ui\_dat)  
*Function returns the number of expected input files for the user's run option.*
- bool `valid_addon_options` (UI\_DATA \*ui\_dat)  
*Function returns true if the user has chosen a valid additional runtime option.*
- void `display_help` (UI\_DATA \*ui\_dat)  
*Function to call the appropriate help menu based on type of interface.*
- void `display_version` (UI\_DATA \*ui\_dat)  
*Function to display ecosystem version information to the console.*
- int `invalid_input` (int count, int max)  
*Function returns a CONTINUE or EXIT when invalid input is given.*
- bool `valid_input_main` (UI\_DATA \*ui\_dat)  
*Function returns true if user gave valid input in Basic UI.*
- bool `valid_input_tests` (UI\_DATA \*ui\_dat)  
*Function returns true if user gave a valid test function to run.*
- bool `valid_input_execute` (UI\_DATA \*ui\_dat)  
*Function returns true if user gave a valid executable function to run.*
- int `test_loop` (UI\_DATA \*ui\_dat)  
*Function that loops the Basic UI until a valid test option was selected.*
- int `exec_loop` (UI\_DATA \*ui\_dat)  
*Function that loops the Basic UI until a valid executable option was selected.*
- int `run_test` (UI\_DATA \*ui\_dat)  
*Function will call the user requested test function.*
- int `run_exec` (UI\_DATA \*ui\_dat)  
*Function will call the user requested executable function.*
- int `run_executable` (int argc, const char \*argv[])  
*Function called by the main and runs both user interfaces for the program.*

### 5.21.1 Detailed Description

User Interface for Ecosystem. ui.cpp

These routines define how the user will interface with the software

**Author**

Austin Ladshaw

**Date**

08/25/2015

**Copyright**

This software was designed and built at the Georgia Institute of Technology by Austin Ladshaw for PhD research in the area of adsorption and surface science. Copyright (c) 2015, all rights reserved.

### 5.21.2 Macro Definition Documentation

#### 5.21.2.1 `#define UI_HPP_`

#### 5.21.2.2 `#define ECO_VERSION "0.0 alpha"`

Macro expansion for executable current version number.

#### 5.21.2.3 `#define ECO_EXECUTABLE "eco0"`

Macro expansion for executable current name.

### 5.21.3 Enumeration Type Documentation

#### 5.21.3.1 `enum valid_options`

Valid options available upon execution of the code.

Enumeration of valid options for executing the ecosystem code. More options become available as the code updates. Some options that appear here may not be viewable in the "help" screen of the executable. Those options are hidden, but are still valid entries.

**Enumerator**

***TEST***

***EXECUTE***

***EXIT***

***CONTINUE***

***HELP***

***dogfish***

***eel***

***egret***

***finch***

***lark***

***macaw***

*mola*  
*monkfish*  
*sandbox*  
*scopsowl*  
*shark*  
*skua*  
*gsta\_opt*  
*magpie*  
*scops\_opt*  
*skua\_opt*  
*trajectory*

#### 5.21.4 Function Documentation

##### 5.21.4.1 void aui\_help ( )

Function to display help for Advanced User Interface.

The Advanced User Interface help screen is accessed by including run option -h or --help when executing the program from command line.

##### 5.21.4.2 void bui\_help ( )

Function to display help for Basic User Interface.

The Basic User Interface help screen is accessed by running the executable, then typing "help" at any point during the console prompts. Exception to this occurs when the console prompts you to provide input files for your chosen routine. In this circumstance, the executable always assumes that what the user types in will be an input file.

##### 5.21.4.3 std::string allLower ( const std::string & input )

Function to return an all lower case string based on the passed argument.

This function will copy the input parameter and convert that copy to all lower case. The copy is then returned and can be checked against valid or allowed strings.

###### Parameters

|              |  |
|--------------|--|
| <i>input</i> | string to copy and convert to lower case |
|--------------|--|

##### 5.21.4.4 bool exit ( const std::string & input )

Function returns true if user requests exit.

This function will check the input string for "exit" or "quit" and terminate the executable. Only checked if using the Basic User Interface.

###### Parameters

|              |  |
|--------------|--|
| <i>input</i> | input string user gives to the console |
|--------------|--|



#### 5.21.4.5 bool help ( const std::string & *input* )

Function returns true if the user requests help.

This function will check the input string for "help", "-h", or "&ndash;help" and will tell the executable to display the help menu. The help menu that gets displayed depends on how the executable was run to begin with.

##### Parameters

|              |  |
|--------------|--|
| <i>input</i> | input string user gives to the console |
|--------------|--|

#### 5.21.4.6 bool version ( const std::string & *input* )

Function returns true if user requests to know the executable version.

This function will check the input string for "version", "-v", or "&ndash;version" and will tell the executable to display version information about the executable.

##### Parameters

|              |  |
|--------------|--|
| <i>input</i> | input string user gives to the console |
|--------------|--|

#### 5.21.4.7 bool test ( const std::string & *input* )

Function returns true if user requests to run a test.

This function will check the input string for "-t" or "&ndash;test" and determine whether or not the user requests to run an ecosystem test function.

##### Parameters

|              |  |
|--------------|--|
| <i>input</i> | input string user gives to the console |
|--------------|--|

#### 5.21.4.8 bool exec ( const std::string & *input* )

Function returns true if the user requests to run a simulation/executable.

This function will check the input string for "-e" or "&ndash;execute" and determine whether or not the user requests to run an ecosystem executable function.

##### Parameters

|              |  |
|--------------|--|
| <i>input</i> | input string the user gives to the console |
|--------------|--|

#### 5.21.4.9 bool path ( const std::string & *input* )

Function returns true if the user indicates that input files share a common path.

This function will check the input string for "-p" or "&ndash;path" and determine whether or not the user will give a common path to all input files needed for the specified simulation. Only used in Advanced User Interface.

##### Parameters

|              |  |
|--------------|--|
| <i>input</i> | input string the user gives to the console |
|--------------|--|

#### 5.21.4.10 `bool input ( const std::string & input )`

Function returns true if the user indicates that the next arguments are input files.

This function will check the input string for "-i" or "&ndash;input" and determine whether or not the user's next arguments are input files for a specific simulation. Only used in Advanced User Interface.

##### Parameters

|              |  |
|--------------|--|
| <i>input</i> | input string the user gives to the console |
|--------------|--|

#### 5.21.4.11 `bool valid_test_string ( const std::string & input, UI_DATA * ui_dat )`

Function returns true if the user gave a valid test option.

This function will check the input string given by the user and determine whether that string denotes a valid test. Then, it will mark the option variable in *ui\_dat* with the appropriate option from the *valid\_options* enum.

##### Parameters

|               |   |
|---------------|---|
| <i>input</i>  | input string the user gives to the console      |
| <i>ui_dat</i> | pointer to the data structure for the ui object |

#### 5.21.4.12 `bool valid_exec_string ( const std::string & input, UI_DATA * ui_dat )`

Function returns true if the user gave a valid execution option.

This function will check the input string given by the user and determine whether that string denotes a valid execution option. Then, it will mark the option variable in *ui\_dat* with the appropriate option from the *valid\_options* enum.

##### Parameters

|               |   |
|---------------|---|
| <i>input</i>  | input string the user gives to the console      |
| <i>ui_dat</i> | pointer to the data structure for the ui object |

#### 5.21.4.13 `int number_files ( UI_DATA * ui_dat )`

Function returns the number of expected input files for the user's run option.

This function will check the option variable in the *ui\_dat* structure to determine the number of input files that is expected to be given. Running different executable functions in ecosystem may require various number of input files.

##### Parameters

|               |   |
|---------------|---|
| <i>ui_dat</i> | pointer to the data structure for the ui object |
|---------------|---|

#### 5.21.4.14 `bool valid_addon_options ( UI_DATA * ui_dat )`

Function returns true if the user has chosen a valid additional runtime option.

This function will check all additional input options in the *user\_input* variable of *ui\_dat* to determine if the user requests any additional options during runtime. Valid additional options are -p or -path and -i or -input.

## Parameters

|               |   |
|---------------|---|
| <i>ui_dat</i> | pointer to the data structure for the ui object |
|---------------|---|

**5.21.4.15 void display\_help ( UI\_DATA \* ui\_dat )**

Function to call the appropriate help menu based on type of interface.

This function looks at the *ui\_dat* structure and the user's OS files to determine what help menu to display and how to display it. There are two different types of help menus that can be displayed: (i) Advanced Help and (ii) Basic Help. Additionally, this function checks the OS file system for the existence of installed help files. If it finds those files, then it instructs the command terminal to read the contents of those files with the "less" command. Otherwise, it will just print the appropriate help menu to the console window.

## Parameters

|               |   |
|---------------|---|
| <i>ui_dat</i> | pointer to the data structure for the ui object |
|---------------|---|

**5.21.4.16 void display\_version ( UI\_DATA \* ui\_dat )**

Function to display ecosystem version information to the console.

This function will check the *ui\_dat* structure to see which type of interface the user is using, then print out the version information for the executable being run.

## Parameters

|               |   |
|---------------|---|
| <i>ui_dat</i> | pointer to the data structure for the ui object |
|---------------|---|

**5.21.4.17 int invalid\_input ( int count, int max )**

Function returns a CONTINUE or EXIT when invalid input is given.

This function looks at the current count and the max iterations and determines whether or not to force the executable to terminate. If the user provides too many incorrect options during the Basic User Interface, then the executable will force quit.

## Parameters

|              |  |
|--------------|--|
| <i>count</i> | number of times the user has provided a bad option |
| <i>max</i>   | maximum allowable bad options before force quit    |

**5.21.4.18 bool valid\_input\_main ( UI\_DATA \* ui\_dat )**

Function returns true if user gave valid input in Basic UI.

This function is only called if the user is running the Basic UI. It checks the given console argument stored in *user\_input* of *ui\_dat* for a valid option. If no valid option is given, then this function returns false.

## Parameters

|               |   |
|---------------|---|
| <i>ui_dat</i> | pointer to the data structure for the ui object |
|---------------|---|

**5.21.4.19 bool valid\_input\_tests ( UI\_DATA \* ui\_dat )**

Function returns true if user gave a valid test function to run.

This function checks the user\_input argument of ui\_dat for a valid test option. If no valid test was given, then this function returns false.

**Parameters**

|               |   |
|---------------|---|
| <i>ui_dat</i> | pointer to the data structure for the ui object |
|---------------|---|

**5.21.4.20 bool valid\_input\_execute ( UI\_DATA \* ui\_dat )**

Function returns true if user gave a valid executable function to run.

This function checks the user\_input argument of ui\_dat for a valid executable option. If no valid executable was given, then this function returns false.

**Parameters**

|               |   |
|---------------|---|
| <i>ui_dat</i> | pointer to the data structure for the ui object |
|---------------|---|

**5.21.4.21 int test\_loop ( UI\_DATA \* ui\_dat )**

Function that loops the Basic UI until a valid test option was selected.

This function loops the Basic UI menu for running a test until a valid test is selected by the user. If a valid test is not selected, and the maximum number of loops has been reached, then this function will cause the program to force quit.

**Parameters**

|               |   |
|---------------|---|
| <i>ui_dat</i> | pointer to the data structure for the ui object |
|---------------|---|

**5.21.4.22 int exec\_loop ( UI\_DATA \* ui\_dat )**

Function that loops the Basic UI until a valid executable option was selected.

This function loops the Basic UI menu for running an executable until a valid executable is selected by the user. If a valid executable is not selected, and the maximum number of loops has been reached, then this function will cause the program to force quit.

**Parameters**

|               |   |
|---------------|---|
| <i>ui_dat</i> | pointer to the data structure for the ui object |
|---------------|---|

**5.21.4.23 int run\_test ( UI\_DATA \* ui\_dat )**

Function will call the user requested test function.

This function checks the option variable of the ui\_dat structure and runs the corresponding test function.

**Parameters**

|               |   |
|---------------|---|
| <i>ui_dat</i> | pointer to the data structure for the ui object |
|---------------|---|

## 5.21.4.24 int run\_exec ( UI\_DATA \* ui\_dat )

Function will call the user requested executable function.

This function checks the option variable of the ui\_dat structure and runs the corresponding executable function.

## Parameters

|               |   |
|---------------|---|
| <i>ui_dat</i> | pointer to the data structure for the ui object |
|---------------|---|

## 5.21.4.25 int run\_executable ( int argc, const char \* argv[] )

Function called by the main and runs both user interfaces for the program.

This function is called in the main.cpp file and passes the console arguments given at run time.

## Parameters

|             |  |
|-------------|--|
| <i>argc</i> | number of arguments provided by the user at the time of execution        |
| <i>argv</i> | list of C-strings that was provided by the user at the time of execution |

## 5.22 yaml\_wrapper.h File Reference

```
#include "yaml.h"
#include "error.h"
#include <map>
#include <string>
#include <iostream>
#include <utility>
#include <stdexcept>
```

## Classes

- class [ValueTypePair](#)
- class [KeyValueMap](#)
- class [SubHeader](#)
- class [Header](#)
- class [Document](#)
- class [YamlWrapper](#)
- class [yaml\\_cpp\\_class](#)

## Typedefs

- typedef enum [data\\_type](#) [data\\_type](#)
- typedef enum [header\\_state](#) [header\\_state](#)

## Enumerations

- enum [data\\_type](#) {  
  [STRING](#), [BOOLEAN](#), [DOUBLE](#), [INT](#),  
  [UNKNOWN](#) }
- enum [header\\_state](#) { [ANCHOR](#), [ALIAS](#), [NONE](#) }

## Functions

- int [YAML\\_WRAPPER\\_TESTS](#) ()
- int [YAML\\_CPP\\_TEST](#) (const char \*file)

### 5.22.1 Typedef Documentation

5.22.1.1 typedef enum data\_type data\_type

5.22.1.2 typedef enum header\_state header\_state

### 5.22.2 Enumeration Type Documentation

5.22.2.1 enum data\_type

Enumerator

***STRING***  
***BOOLEAN***  
***DOUBLE***  
***INT***  
***UNKNOWN***

5.22.2.2 enum header\_state

Enumerator

***ANCHOR***  
***ALIAS***  
***NONE***

### 5.22.3 Function Documentation

5.22.3.1 int [YAML\\_WRAPPER\\_TESTS](#) ( )

5.22.3.2 int [YAML\\_CPP\\_TEST](#) ( const char \* *file* )

# Index

- ~Atom
  - Atom, [11](#)
- ~Document
  - Document, [23](#)
- ~Header
  - Header, [59](#)
- ~KeyValueMap
  - KeyValueMap, [62](#)
- ~MassBalance
  - MassBalance, [64](#)
- ~MasterSpeciesList
  - MasterSpeciesList, [66](#)
- ~Matrix
  - Matrix, [70](#)
- ~Molecule
  - Molecule, [81](#)
- ~PeriodicTable
  - PeriodicTable, [95](#)
- ~Reaction
  - Reaction, [105](#)
- ~SubHeader
  - SubHeader, [122](#)
- ~UnsteadyReaction
  - UnsteadyReaction, [132](#)
- ~ValueTypePair
  - ValueTypePair, [135](#)
- ~YamlWrapper
  - YamlWrapper, [138](#)
- ~yaml\_cpp\_class
  - yaml\_cpp\_class, [136](#)
- A
  - magpie.h, [183](#)
- a
  - TRAJECTORY\_DATA, [127](#)
- ALIAS
  - yaml\_wrapper.h, [214](#)
- ANCHOR
  - yaml\_wrapper.h, [214](#)
- A\_separator
  - TRAJECTORY\_DATA, [127](#)
- A\_wire
  - TRAJECTORY\_DATA, [127](#)
- ARNOLDI\_DATA, [7](#)
  - beta, [8](#)
  - e1, [8](#)
  - Hkp1, [8](#)
  - hp1, [8](#)
  - iter, [8](#)
  - k, [8](#)
  - Output, [8](#)
  - sum, [9](#)
  - v, [8](#)
  - Vk, [8](#)
  - w, [8](#)
  - yk, [8](#)
- abs\_tol\_bias
  - SCOPSOWL\_OPT\_DATA, [111](#)
  - SKUA\_OPT\_DATA, [119](#)
- act\_choice
  - shark.h, [199](#)
- act\_fun
  - SHARK\_DATA, [114](#)
- activation\_energy
  - SCOPSOWL\_PARAM\_DATA, [112](#)
  - SKUA\_PARAM, [119](#)
  - UnsteadyReaction, [134](#)
- activity\_data
  - SHARK\_DATA, [115](#)
- activity\_new
  - SHARK\_DATA, [115](#)
- activity\_old
  - SHARK\_DATA, [115](#)
- addDockKey
  - YamlWrapper, [139](#)
- addHeadKey
  - Document, [24](#)
- addKey
  - KeyValueMap, [62](#)
- addPair
  - Document, [24](#)
  - Header, [60](#)
  - KeyValueMap, [62](#)
  - SubHeader, [122](#)
- addSubKey
  - Header, [60](#)
- adjoint
  - Matrix, [72](#)
- adsorb\_index
  - SCOPSOWL\_OPT\_DATA, [110](#)
  - SKUA\_OPT\_DATA, [118](#)
- Adsorbable
  - SCOPSOWL\_PARAM\_DATA, [112](#)
  - SKUA\_PARAM, [120](#)
- affinity
  - SCOPSOWL\_PARAM\_DATA, [112](#)
  - SKUA\_PARAM, [120](#)
- Ai
  - OPTRANS\_DATA, [91](#)

- alias
  - SubHeader, [123](#)
- alkalinity
  - MasterSpeciesList, [67](#)
- all\_pars
  - GSTA\_OPT\_DATA, [58](#)
- allLower
  - ui.h, [208](#)
- alpha
  - BACKTRACK\_DATA, [15](#)
  - BiCGSTAB\_DATA, [17](#)
  - CGS\_DATA, [20](#)
  - GCR\_DATA, [44](#)
  - PCG\_DATA, [93](#)
- anchor\_alias\_dne
  - error.h, [152](#)
- Ap
  - PCG\_DATA, [94](#)
- arg
  - GMRESR\_DATA, [50](#)
- arg\_matrix\_same
  - error.h, [151](#)
- argc
  - UI\_DATA, [130](#)
- argv
  - UI\_DATA, [130](#)
- arnoldi
  - lark.h, [169](#)
- arnoldi\_dat
  - GMRESLP\_DATA, [47](#)
- As
  - SYSTEM\_DATA, [124](#)
- assertType
  - KeyValueMap, [63](#)
  - ValueTypePair, [135](#)
- Atom, [9](#)
  - ~Atom, [11](#)
  - Atom, [11](#)
  - AtomCategory, [13](#)
  - AtomName, [12](#)
  - AtomState, [13](#)
  - AtomSymbol, [12](#)
  - atomic\_number, [14](#)
  - atomic\_weight, [13](#)
  - AtomicNumber, [13](#)
  - AtomicWeight, [12](#)
  - BondingElectrons, [12](#)
  - Category, [14](#)
  - DisplayInfo, [13](#)
  - editAtomicWeight, [11](#)
  - editElectrons, [11](#)
  - editNeutrons, [11](#)
  - editOxidationState, [11](#)
  - editProtons, [11](#)
  - editValence, [12](#)
  - Electrons, [12](#)
  - electrons, [13](#)
  - Name, [13](#)
  - NaturalState, [14](#)
  - Neutrons, [12](#)
  - neutrons, [13](#)
  - oxidation\_state, [13](#)
  - OxidationState, [12](#)
  - Protons, [12](#)
  - protons, [13](#)
  - Register, [11](#)
  - removeElectron, [12](#)
  - removeNeutron, [12](#)
  - removeProton, [12](#)
  - Symbol, [13](#)
  - valence\_e, [13](#)
- AtomCategory
  - Atom, [13](#)
- AtomName
  - Atom, [12](#)
- AtomState
  - Atom, [13](#)
- AtomSymbol
  - Atom, [12](#)
- atomic\_number
  - Atom, [14](#)
- atomic\_weight
  - Atom, [13](#)
- AtomicNumber
  - Atom, [13](#)
- AtomicWeight
  - Atom, [12](#)
- atoms
  - Molecule, [85](#)
- au\_help
  - ui.h, [208](#)
- avg\_fiber\_density
  - MONKFISH\_DATA, [87](#)
- avg\_norm
  - SYSTEM\_DATA, [125](#)
- avg\_sorption
  - MONKFISH\_PARAM, [89](#)
- avg\_sorption\_old
  - MONKFISH\_PARAM, [89](#)
- avgDp
  - scopsowl.h, [195](#)
- avgPar
  - gsta\_opt.h, [162](#)
- avgValue
  - gsta\_opt.h, [163](#)
- b
  - TRAJECTORY\_DATA, [127](#)
- B0
  - TRAJECTORY\_DATA, [127](#)
- BOOLEAN
  - yaml\_wrapper.h, [214](#)
- BACKTRACK\_DATA, [14](#)
  - alpha, [15](#)
  - constRho, [15](#)
  - Fk, [15](#)
  - lambdaMin, [15](#)



- normFkp1, [15](#)
- rho, [15](#)
- xk, [15](#)
- backtrack\_dat
  - PJFNK\_DATA, [102](#)
- backtrackLineSearch
  - lark.h, [177](#)
- BasicUI
  - UI\_DATA, [130](#)
- begin
  - Document, [24](#)
  - Header, [60](#)
  - KeyValueMap, [62](#)
  - YamlWrapper, [138](#)
- best\_par
  - GSTA\_OPT\_DATA, [57](#)
- bestres
  - BiCGSTAB\_DATA, [18](#)
  - CGS\_DATA, [21](#)
  - GCR\_DATA, [44](#)
  - GMRESLP\_DATA, [47](#)
  - GMRESRP\_DATA, [52](#)
  - PCG\_DATA, [93](#)
  - PICARD\_DATA, [97](#)
- bestx
  - BiCGSTAB\_DATA, [18](#)
  - CGS\_DATA, [21](#)
  - GCR\_DATA, [44](#)
  - GMRESLP\_DATA, [47](#)
  - GMRESRP\_DATA, [53](#)
  - PCG\_DATA, [93](#)
  - PICARD\_DATA, [97](#)
  - PJFNK\_DATA, [101](#)
- beta
  - ARNOLDI\_DATA, [8](#)
  - BiCGSTAB\_DATA, [17](#)
  - CGS\_DATA, [20](#)
  - FINCH\_DATA, [37](#)
  - GCR\_DATA, [44](#)
  - PCG\_DATA, [93](#)
  - TRAJECTORY\_DATA, [127](#)
- BiCGSTAB
  - lark.h, [169](#)
- BiCGSTAB\_DATA, [15](#)
  - alpha, [17](#)
  - bestres, [18](#)
  - bestx, [18](#)
  - beta, [17](#)
  - breakdown, [17](#)
  - iter, [17](#)
  - maxit, [17](#)
  - omega, [17](#)
  - omega\_old, [17](#)
  - Output, [18](#)
  - p, [18](#)
  - r, [18](#)
  - r0, [18](#)
  - relres, [17](#)
  - relres\_base, [18](#)
  - res, [17](#)
  - rho, [17](#)
  - rho\_old, [17](#)
  - s, [18](#)
  - t, [19](#)
  - tol\_abs, [17](#)
  - tol\_rel, [17](#)
  - v, [18](#)
  - x, [18](#)
  - y, [18](#)
  - z, [18](#)
- bicgstab
  - lark.h, [173](#)
- bicgstab\_dat
  - PJFNK\_DATA, [101](#)
- binary\_diffusion
  - MIXED\_GAS, [78](#)
- binder\_fraction
  - SCOPSOWL\_DATA, [108](#)
- binder\_poresize
  - SCOPSOWL\_DATA, [108](#)
- binder\_porosity
  - SCOPSOWL\_DATA, [108](#)
- BondingElectrons
  - Atom, [12](#)
- Bounce
  - PJFNK\_DATA, [101](#)
- breakdown
  - BiCGSTAB\_DATA, [17](#)
  - CGS\_DATA, [20](#)
  - GCR\_DATA, [44](#)
- Brown\_RAD
  - Trajectory.h, [204](#)
- Brown\_THETA
  - Trajectory.h, [204](#)
- bui\_help
  - ui.h, [208](#)
- C
  - Speciation\_Test01\_Data, [121](#)
- c
  - CGS\_DATA, [22](#)
  - GCR\_DATA, [45](#)
- CGS
  - lark.h, [169](#)
- CONTINUE
  - ui.h, [207](#)
- c\_temp
  - GCR\_DATA, [45](#)
- CARTESIAN
  - Trajectory.h, [204](#)
- CC\_E
  - FINCH\_DATA, [37](#)
- CC\_I
  - FINCH\_DATA, [37](#)
- CE3
  - egret.h, [148](#)
- CGS\_DATA, [19](#)

- alpha, [20](#)
- bestres, [21](#)
- bestx, [21](#)
- beta, [20](#)
- breakdown, [20](#)
- c, [22](#)
- iter, [20](#)
- maxit, [20](#)
- Output, [21](#)
- p, [22](#)
- r, [21](#)
- r0, [21](#)
- relres, [21](#)
- relres\_base, [21](#)
- res, [21](#)
- rho, [20](#)
- sigma, [20](#)
- tol\_abs, [21](#)
- tol\_rel, [21](#)
- u, [21](#)
- v, [22](#)
- w, [22](#)
- x, [21](#)
- z, [22](#)
- CL\_E
  - FINCH\_DATA, [37](#)
- CL\_I
  - FINCH\_DATA, [37](#)
- CN
  - FINCH\_DATA, [36](#)
- CR\_E
  - FINCH\_DATA, [37](#)
- CR\_I
  - FINCH\_DATA, [37](#)
- CT
  - Speciation\_Test01\_Data, [121](#)
- CURVE\_TEST03
  - scopsowl.h, [196](#)
- CURVE\_TEST04
  - scopsowl.h, [196](#)
- CURVE\_TEST05
  - scopsowl.h, [196](#)
- calculate\_properties
  - egret.h, [149](#)
- calculateAvgOxiState
  - Molecule, [83](#)
- calculateEnergies
  - Reaction, [106](#)
  - UnsteadyReaction, [133](#)
- calculateEquilibrium
  - Reaction, [106](#)
  - UnsteadyReaction, [133](#)
- calculateRate
  - UnsteadyReaction, [133](#)
- callroutine
  - FINCH\_DATA, [41](#)
- CanCalcG
  - Reaction, [106](#)
- CanCalcHS
  - Reaction, [106](#)
- Cap
  - TRAJECTORY\_DATA, [128](#)
- Carrier
  - SYSTEM\_DATA, [125](#)
- Cartesian
  - finch.h, [155](#)
- Category
  - Atom, [14](#)
- cgs
  - lark.h, [173](#)
- cgs\_dat
  - PJFNK\_DATA, [101](#)
- changeKey
  - Document, [24](#)
  - Header, [60](#)
  - YamlWrapper, [138](#)
- char\_length
  - MIXED\_GAS, [78](#)
- char\_macro
  - SCOPSOWL\_DATA, [108](#)
- char\_measure
  - SKUA\_DATA, [117](#)
- char\_micro
  - SCOPSOWL\_DATA, [108](#)
- Charge
  - Molecule, [83](#)
- charge
  - MasterSpeciesList, [66](#)
  - Molecule, [85](#)
- check\_Mass
  - finch.h, [156](#)
- CheckMass
  - FINCH\_DATA, [36](#)
- CheckMolefractions
  - MIXED\_GAS, [77](#)
- checkSpeciesEnergies
  - Reaction, [106](#)
  - UnsteadyReaction, [133](#)
- chi\_p
  - TRAJECTORY\_DATA, [127](#)
- cleanup
  - yaml\_cpp\_class, [137](#)
- clear
  - Document, [24](#)
  - Header, [60](#)
  - KeyValueMap, [62](#)
  - SubHeader, [122](#)
  - YamlWrapper, [138](#)
- cofactor
  - Matrix, [71](#)
- columnExtend
  - Matrix, [75](#)
- columnExtract
  - Matrix, [75](#)
- columnProjection
  - Matrix, [74](#)

columnReplace  
     Matrix, 75  
 columnShrink  
     Matrix, 75  
 columnVectorFill  
     Matrix, 74  
 columns  
     Matrix, 71  
 CompareFile  
     SCOPSOWL\_OPT\_DATA, 111  
     SKUA\_OPT\_DATA, 119  
 Conc\_new  
     SHARK\_DATA, 115  
 Conc\_old  
     SHARK\_DATA, 115  
 Console\_Output  
     SHARK\_DATA, 115  
 const\_Dc  
     skua.h, 202  
 const\_filmMassTransfer  
     scopsowl.h, 196  
 const\_kf  
     skua.h, 202  
 const\_pH  
     SHARK\_DATA, 115  
 const\_pore\_diffusion  
     scopsowl.h, 196  
 constRho  
     BACKTRACK\_DATA, 15  
 ConstantICFill  
     Matrix, 73  
 Contains\_pH  
     SHARK\_DATA, 115  
 Contains\_pOH  
     SHARK\_DATA, 115  
 Converged  
     SHARK\_DATA, 115  
 Convert2Concentration  
     shark.h, 199  
 Convert2LogConcentration  
     shark.h, 199  
 coord  
     SKUA\_DATA, 116  
 coord\_macro  
     SCOPSOWL\_DATA, 108  
 coord\_micro  
     SCOPSOWL\_DATA, 108  
 copyAnchor2Alias  
     Document, 24  
     Header, 60  
     YamlWrapper, 139  
 count  
     UI\_DATA, 129  
 crystal\_radius  
     SCOPSOWL\_DATA, 108  
 Cstd  
     egret.h, 148  
 current\_equil  
     SCOPSOWL\_OPT\_DATA, 110  
     SKUA\_OPT\_DATA, 118  
 current\_points  
     SCOPSOWL\_OPT\_DATA, 110  
     SKUA\_OPT\_DATA, 118  
 current\_press  
     SCOPSOWL\_OPT\_DATA, 110  
     SKUA\_OPT\_DATA, 118  
 current\_temp  
     SCOPSOWL\_OPT\_DATA, 110  
     SKUA\_OPT\_DATA, 118  
 current\_token  
     yaml\_cpp\_class, 137  
 Cylindrical  
     finch.h, 155  
  
 d  
     FINCH\_DATA, 34  
 DAVIES  
     shark.h, 199  
 DAVIES\_LADSHAW  
     shark.h, 199  
 DEBYE\_HUCKEL  
     shark.h, 199  
 DOUBLE  
     yaml\_wrapper.h, 214  
 D\_c  
     skua.h, 201  
 D\_ii  
     egret.h, 148  
 D\_ij  
     egret.h, 148  
 D\_inf  
     skua.h, 201  
 D\_o  
     skua.h, 201  
 DBL\_EPSILON  
     magpie.h, 183  
 dHo  
     GSTA\_DATA, 56  
 DIC  
     FINCH\_DATA, 35  
 DISPLACEMENT  
     Trajectory.h, 205  
 DOGFISH  
     dogfish.h, 144  
 DOGFISH\_DATA, 25  
     DirichletBC, 27  
     end\_time, 27  
     eval\_DI, 28  
     eval\_R, 27  
     eval\_kf, 28  
     eval\_qs, 28  
     fiber\_diameter, 27  
     fiber\_length, 27  
     finch\_dat, 28  
     NonLinear, 27  
     NumComp, 27  
     OutputFile, 27

- param\_dat, 28
- Print2Console, 26
- Print2File, 26
- t\_counter, 27
- t\_print, 27
- time, 26
- time\_old, 26
- total\_sorption, 27
- total\_sorption\_old, 27
- total\_steps, 26
- user\_data, 28
- DOGFISH\_Executioner
  - dogfish.h, 143
- DOGFISH\_PARAM, 28
  - film\_transfer\_coeff, 29
  - initial\_sorption, 29
  - intraparticle\_diffusion, 29
  - sorbed\_molefraction, 29
  - species, 29
  - surface\_concentration, 29
- DOGFISH\_TESTS
  - dogfish.h, 144
- DOGFISH\_postprocesses
  - dogfish.h, 144
- DOGFISH\_preprocesses
  - dogfish.h, 144
- DOGFISH\_reset
  - dogfish.h, 144
- dSo
  - GSTA\_DATA, 56
- dX
  - TRAJECTORY\_DATA, 128
- dY
  - TRAJECTORY\_DATA, 128
- Data
  - Matrix, 76
- Data\_Map
  - SubHeader, 123
- data\_type
  - yaml\_wrapper.h, 214
- Davies\_equation
  - shark.h, 199
- DaviesLadshaw\_equation
  - shark.h, 199
- DebyeHuckel\_equation
  - shark.h, 199
- default\_Dc
  - skua.h, 202
- default\_FilmMTCoeff
  - dogfish.h, 143
- default\_IntraDiffusion
  - dogfish.h, 143
- default\_Retardation
  - dogfish.h, 143
- default\_SurfaceConcentration
  - dogfish.h, 143
- default\_adsorption
  - scopsowl.h, 196
- default\_bcs
  - finch.h, 158
- default\_density
  - monkfish.h, 193
- default\_effective\_diffusion
  - scopsowl.h, 196
- default\_execution
  - finch.h, 157
- default\_exterior\_concentration
  - monkfish.h, 193
- default\_film\_transfer
  - monkfish.h, 193
- default\_filmMassTransfer
  - scopsowl.h, 196
- default\_ic
  - finch.h, 157
- default\_interparticle\_diffusion
  - monkfish.h, 193
- default\_kf
  - skua.h, 202
- default\_monk\_adsorption
  - monkfish.h, 193
- default\_monk\_equilibrium
  - monkfish.h, 193
- default\_monkfish\_retardation
  - monkfish.h, 193
- default\_params
  - finch.h, 158
- default\_pore\_diffusion
  - scopsowl.h, 196
- default\_porosity
  - monkfish.h, 193
- default\_postprocess
  - finch.h, 159
- default\_precon
  - finch.h, 159
- default\_preprocess
  - finch.h, 157
- default\_res
  - finch.h, 158
- default\_reset
  - finch.h, 159
- default\_retardation
  - scopsowl.h, 196
- default\_solve
  - finch.h, 158
- default\_surf\_diffusion
  - scopsowl.h, 196
- default\_timestep
  - finch.h, 157
- Delta
  - MassBalance, 65
- density
  - PURE\_GAS, 104
- determinate
  - Matrix, 71
- diagonalSolve
  - Matrix, 74

- dielectric\_const
  - SHARK\_DATA, 114
- diffusion\_type
  - SCOPSOWL\_OPT\_DATA, 110
  - SKUA\_OPT\_DATA, 118
- dim\_mis\_match
  - error.h, 151
- Dirichlet
  - FINCH\_DATA, 36
- DirichletBC
  - DOGFISH\_DATA, 27
  - MONKFISH\_DATA, 87
  - SCOPSOWL\_DATA, 108
  - SKUA\_DATA, 117
- dirichletBCFill
  - Matrix, 74
- discretize
  - FINCH\_DATA, 41
- Display
  - Matrix, 72
- Display\_Info
  - MassBalance, 65
  - Reaction, 106
  - UnsteadyReaction, 132
- display\_help
  - ui.h, 211
- display\_version
  - ui.h, 211
- DisplayAll
  - MasterSpeciesList, 66
- DisplayConcentrations
  - MasterSpeciesList, 66
- DisplayContents
  - Document, 24
  - Header, 60
  - SubHeader, 122
  - yaml\_cpp\_class, 137
  - YamlWrapper, 139
- DisplayInfo
  - Atom, 13
  - MasterSpeciesList, 66
  - Molecule, 84
- DisplayMap
  - KeyValueMap, 63
- DisplayPair
  - ValueTypePair, 135
- DisplayTable
  - PeriodicTable, 95
- Dk
  - scopsowl.h, 195
- Dn
  - FINCH\_DATA, 39
- Dnp1
  - FINCH\_DATA, 39
- Do
  - FINCH\_DATA, 35
- Doc\_Map
  - YamlWrapper, 139
- Document, 22
  - ~Document, 23
  - addHeadKey, 24
  - addPair, 24
  - begin, 24
  - changeKey, 24
  - clear, 24
  - copyAnchor2Alias, 24
  - DisplayContents, 24
  - Document, 23, 24
  - end, 24
  - getAlias, 25
  - getAnchoredHeader, 25
  - getDataMap, 24
  - getHeadFromSubAlias, 25
  - getHeadMap, 24
  - getHeader, 24
  - getName, 25
  - getState, 25
  - Head\_Map, 25
  - isAlias, 25
  - isAnchor, 25
  - operator(), 24
  - operator=, 24
  - resetKeys, 24
  - revalidateAllKeys, 24
  - setAlias, 24
  - setName, 24
  - setNameAliasPair, 24
  - setState, 24
  - size, 24
- dog\_dat
  - MONKFISH\_DATA, 88
- dogfish
  - ui.h, 207
- dogfish.h, 141
  - DOGFISH, 144
  - DOGFISH\_Executioner, 143
  - DOGFISH\_TESTS, 144
  - DOGFISH\_postprocesses, 144
  - DOGFISH\_preprocesses, 144
  - DOGFISH\_reset, 144
  - default\_FilmMTCoeff, 143
  - default\_IntraDiffusion, 143
  - default\_Retardation, 143
  - default\_SurfaceConcentration, 143
  - print2file\_DOGFISH\_header, 142
  - print2file\_DOGFISH\_result\_new, 143
  - print2file\_DOGFISH\_result\_old, 142
  - print2file\_species\_header, 142
  - set\_DOGFISH\_ICs, 143
  - set\_DOGFISH\_params, 144
  - set\_DOGFISH\_timestep, 144
  - setup\_DOGFISH\_DATA, 143
- domain\_diameter
  - MONKFISH\_DATA, 87
- Dp
  - scopsowl.h, 195

- Dp\_ij
  - egret.h, 148
- dq\_dc
  - SCOPSOWL\_PARAM\_DATA, 112
- dq\_dco
  - SCOPSOWL\_PARAM\_DATA, 112
- dq\_dp
  - magpie.h, 184
- dt
  - FINCH\_DATA, 34
  - SHARK\_DATA, 114
  - TRAJECTORY\_DATA, 127
- dt\_min
  - SHARK\_DATA, 114
- dt\_old
  - FINCH\_DATA, 34
- duplicate\_variable
  - error.h, 152
- dxj
  - NUM\_JAC\_DATA, 90
- dynamic\_viscosity
  - PURE\_GAS, 104
- dz
  - FINCH\_DATA, 34
- e0
  - GMRESRP\_DATA, 53
- e0\_bar
  - GMRESRP\_DATA, 53
- e1
  - ARNOLDI\_DATA, 8
- EXECUTE
  - ui.h, 207
- EXIT
  - ui.h, 207
- e\_norm
  - SCOPSOWL\_OPT\_DATA, 110
  - SKUA\_OPT\_DATA, 118
- e\_norm\_old
  - SCOPSOWL\_OPT\_DATA, 110
  - SKUA\_OPT\_DATA, 118
- ECO\_EXECUTABLE
  - ui.h, 207
- ECO\_VERSION
  - ui.h, 207
- EEL\_TESTS
  - eel.h, 146
- EGRET\_TESTS
  - egret.h, 149
- eMax
  - magpie.h, 185
  - mSPD\_DATA, 89
- edit
  - Matrix, 70
- editAllOxidationStates
  - Molecule, 83
- editAtomicWeight
  - Atom, 11
- editCharge
  - Molecule, 82
- editElectrons
  - Atom, 11
- editEnergy
  - Molecule, 83
- editEnthalpy
  - Molecule, 83
- editEntropy
  - Molecule, 83
- editHS
  - Molecule, 83
- editNeutrons
  - Atom, 11
- editOneOxidationState
  - Molecule, 82
- editOxidationState
  - Atom, 11
- editPair
  - ValueTypePair, 135
- editProtons
  - Atom, 11
- editValence
  - Atom, 12
- editValue
  - ValueTypePair, 135
- editValue4Key
  - KeyValueMap, 62
- eduGuess
  - gsta\_opt.h, 164
- eel
  - ui.h, 207
- eel.h, 145
  - EEL\_TESTS, 146
- egret
  - ui.h, 207
- egret.h, 146
  - CE3, 148
  - calculate\_properties, 149
  - Cstd, 148
  - D\_ii, 148
  - D\_ij, 148
  - Dp\_ij, 148
  - EGRET\_TESTS, 149
  - FilmMTCoeff, 149
  - initialize\_data, 149
  - Mu, 148
  - Nu, 148
  - PE3, 148
  - PSI, 148
  - Po, 148
  - Pstd, 148
  - RE3, 148
  - ReNum, 149
  - Rstd, 148
  - ScNum, 149
  - set\_variables, 149
- Electrons
  - Atom, 12

- electrons
  - Atom, [13](#)
- empirical\_kf
  - skua.h, [202](#)
- empty\_matrix
  - error.h, [151](#)
- end
  - Document, [24](#)
  - Header, [60](#)
  - KeyValueMap, [62](#)
  - YamlWrapper, [138](#)
- end\_time
  - DOGFISH\_DATA, [27](#)
  - MONKFISH\_DATA, [87](#)
- Energy
  - Molecule, [84](#)
- energy
  - Reaction, [106](#)
- Enthalpy
  - Molecule, [84](#)
- enthalpy
  - Reaction, [106](#)
- Entropy
  - Molecule, [84](#)
- entropy
  - Reaction, [106](#)
- eps
  - NUM\_JAC\_DATA, [90](#)
  - PJFNK\_DATA, [100](#)
- Equilibrium
  - Reaction, [106](#)
- error
  - error.h, [152](#)
- error.h
  - anchor\_alias\_dne, [152](#)
  - arg\_matrix\_same, [151](#)
  - dim\_mis\_match, [151](#)
  - duplicate\_variable, [152](#)
  - empty\_matrix, [151](#)
  - file\_dne, [151](#)
  - generic\_error, [151](#)
  - indexing\_error, [151](#)
  - initial\_error, [152](#)
  - invalid\_atom, [152](#)
  - invalid\_boolean, [151](#)
  - invalid\_components, [151](#)
  - invalid\_console\_input, [152](#)
  - invalid\_electron, [152](#)
  - invalid\_fraction, [151](#)
  - invalid\_gas\_sum, [151](#)
  - invalid\_molefraction, [151](#)
  - invalid\_neutron, [152](#)
  - invalid\_norm, [152](#)
  - invalid\_proton, [152](#)
  - invalid\_size, [151](#)
  - invalid\_solid\_sum, [151](#)
  - invalid\_species, [152](#)
  - invalid\_type, [152](#)
  - invalid\_valence, [152](#)
  - key\_not\_found, [152](#)
  - magpie\_reverse\_error, [151](#)
  - matrix\_too\_small, [151](#)
  - matvec\_mis\_match, [151](#)
  - missing\_information, [152](#)
  - negative\_mass, [151](#)
  - negative\_time, [151](#)
  - no\_diffusion, [151](#)
  - non\_real\_edge, [152](#)
  - non\_square\_matrix, [151](#)
  - not\_a\_token, [152](#)
  - nullptr\_error, [152](#)
  - nullptr\_func, [151](#)
  - opt\_no\_support, [151](#)
  - ortho\_check\_fail, [151](#)
  - out\_of\_bounds, [151](#)
  - read\_error, [152](#)
  - rxn\_rate\_error, [152](#)
  - scenario\_fail, [151](#)
  - simulation\_fail, [151](#)
  - singular\_matrix, [151](#)
  - string\_parse\_error, [152](#)
  - tensor\_out\_of\_bounds, [152](#)
  - unregistered\_name, [152](#)
  - unstable\_matrix, [151](#)
  - vector\_out\_of\_bounds, [152](#)
  - zero\_vector, [152](#)
- error.h, [150](#)
  - error, [152](#)
  - error\_type, [151](#)
  - mError, [151](#)
- error\_type
  - error.h, [151](#)
- eta
  - mSPD\_DATA, [89](#)
  - TRAJECTORY\_DATA, [127](#)
- eval\_Cex
  - MONKFISH\_DATA, [88](#)
- Eval\_ChargeResidual
  - MasterSpeciesList, [67](#)
- eval\_DI
  - DOGFISH\_DATA, [28](#)
- eval\_Dex
  - MONKFISH\_DATA, [87](#)
- eval\_GPAST
  - magpie.h, [187](#)
- eval\_GSTA
  - gsta\_opt.h, [164](#)
- Eval\_IC\_Residual
  - UnsteadyReaction, [134](#)
- eval\_R
  - DOGFISH\_DATA, [27](#)
- Eval\_ReactionRate
  - UnsteadyReaction, [134](#)
- Eval\_Residual
  - MassBalance, [65](#)
  - Reaction, [106](#)

- UnsteadyReaction, [134](#)
- eval\_Ret
  - MONKFISH\_DATA, [88](#)
- eval\_SCOPSOWL\_Uptake
  - scopsowl\_opt.h, [197](#)
- eval\_SKUA\_Uptake
  - skua\_opt.h, [203](#)
- eval\_ads
  - MONKFISH\_DATA, [87](#)
  - SCOPSOWL\_DATA, [109](#)
- eval\_diff
  - SCOPSOWL\_DATA, [109](#)
  - SKUA\_DATA, [117](#)
- eval\_eps
  - MONKFISH\_DATA, [87](#)
- eval\_eta
  - magpie.h, [187](#)
- eval\_kf
  - DOGFISH\_DATA, [28](#)
  - MONKFISH\_DATA, [88](#)
  - SCOPSOWL\_DATA, [109](#)
  - SKUA\_DATA, [117](#)
- eval\_po
  - magpie.h, [187](#)
- eval\_po\_PI
  - magpie.h, [186](#)
- eval\_po\_qo
  - magpie.h, [186](#)
- eval\_qs
  - DOGFISH\_DATA, [28](#)
- eval\_retard
  - SCOPSOWL\_DATA, [109](#)
- eval\_rho
  - MONKFISH\_DATA, [87](#)
- eval\_surfDiff
  - SCOPSOWL\_DATA, [109](#)
- EvalActivity
  - SHARK\_DATA, [115](#)
- evalprecon
  - FINCH\_DATA, [41](#)
- evalres
  - FINCH\_DATA, [41](#)
- evaluation
  - SCOPSOWL\_OPT\_DATA, [110](#)
  - SKUA\_OPT\_DATA, [118](#)
- exec
  - ui.h, [209](#)
- exec\_loop
  - ui.h, [212](#)
- executeYamlRead
  - yaml\_cpp\_class, [137](#)
- exit
  - ui.h, [208](#)
- Explicit\_Eval
  - UnsteadyReaction, [134](#)
- ExplicitFlux
  - FINCH\_DATA, [36](#)
- exterior\_concentration
  - MONKFISH\_PARAM, [88](#)
- exterior\_transfer\_coeff
  - MONKFISH\_PARAM, [88](#)
- F
  - PJFNK\_DATA, [101](#)
- FINCH\_Picard
  - finch.h, [155](#)
- FOM
  - lark.h, [169](#)
- f\_bias
  - SCOPSOWL\_OPT\_DATA, [110](#)
  - SKUA\_OPT\_DATA, [118](#)
- f\_bias\_old
  - SCOPSOWL\_OPT\_DATA, [110](#)
  - SKUA\_OPT\_DATA, [118](#)
- fC\_E
  - FINCH\_DATA, [38](#)
- fC\_I
  - FINCH\_DATA, [38](#)
- FINCH\_DATA, [29](#)
  - beta, [37](#)
  - CC\_E, [37](#)
  - CC\_I, [37](#)
  - CL\_E, [37](#)
  - CL\_I, [37](#)
  - CN, [36](#)
  - CR\_E, [37](#)
  - CR\_I, [37](#)
  - callroutine, [41](#)
  - CheckMass, [36](#)
  - d, [34](#)
  - DIC, [35](#)
  - Dirichlet, [36](#)
  - discretize, [41](#)
  - Dn, [39](#)
  - Dnp1, [39](#)
  - Do, [35](#)
  - dt, [34](#)
  - dt\_old, [34](#)
  - dz, [34](#)
  - evalprecon, [41](#)
  - evalres, [41](#)
  - ExplicitFlux, [36](#)
  - fC\_E, [38](#)
  - fC\_I, [38](#)
  - fL\_E, [38](#)
  - fL\_I, [38](#)
  - fR\_E, [38](#)
  - fR\_I, [38](#)
  - Fn, [40](#)
  - Fnp1, [40](#)
  - gE, [40](#)
  - gl, [40](#)
  - Iterative, [36](#)
  - kIC, [35](#)
  - kfn, [35](#)
  - kfnp1, [36](#)
  - kn, [40](#)



- knP1, [40](#)
- ko, [35](#)
- L, [34](#)
- LN, [36](#)
- lambda\_E, [36](#)
- lambda\_I, [36](#)
- ME, [38](#)
- MI, [38](#)
- max\_iter, [37](#)
- NE, [38](#)
- NI, [38](#)
- nl\_method, [37](#)
- NormTrack, [36](#)
- OE, [38](#)
- OI, [38](#)
- param\_data, [42](#)
- picard\_dat, [42](#)
- pjfnk\_dat, [42](#)
- pres, [40](#)
- RIC, [35](#)
- res, [40](#)
- resettime, [41](#)
- Rn, [40](#)
- Rnp1, [40](#)
- Ro, [35](#)
- s, [34](#)
- setbcs, [41](#)
- setic, [41](#)
- setparams, [41](#)
- setpostprocess, [41](#)
- setpreprocess, [41](#)
- settime, [41](#)
- Sn, [40](#)
- SnP1, [40](#)
- solve, [41](#)
- SteadyState, [36](#)
- T, [34](#)
- t, [34](#)
- t\_old, [34](#)
- tol\_abs, [37](#)
- tol\_rel, [37](#)
- total\_iter, [37](#)
- u\_star, [39](#)
- uAvg, [34](#)
- uAvg\_old, [35](#)
- uIC, [35](#)
- uT, [34](#)
- uT\_old, [34](#)
- ubest, [39](#)
- un, [39](#)
- unm1, [39](#)
- unp1, [39](#)
- uo, [35](#)
- Update, [36](#)
- uz\_I\_E, [39](#)
- uz\_I\_I, [39](#)
- uz\_lm1\_E, [39](#)
- uz\_lm1\_I, [39](#)
- uz\_lp1\_E, [39](#)
- uz\_lp1\_I, [39](#)
- vIC, [35](#)
- vn, [39](#)
- vnp1, [39](#)
- vo, [35](#)
- FINCH\_TESTS
  - finch.h, [159](#)
- fL\_E
  - FINCH\_DATA, [38](#)
- fL\_I
  - FINCH\_DATA, [38](#)
- fR\_E
  - FINCH\_DATA, [38](#)
- fR\_I
  - FINCH\_DATA, [38](#)
- fiber\_diameter
  - DOGFISH\_DATA, [27](#)
- fiber\_length
  - DOGFISH\_DATA, [27](#)
- file\_dne
  - error.h, [151](#)
- File\_Output
  - SHARK\_DATA, [115](#)
- file\_name
  - yaml\_cpp\_class, [137](#)
- Files
  - UI\_DATA, [130](#)
- film\_transfer
  - SCOPSOWL\_PARAM\_DATA, [112](#)
  - SKUA\_PARAM, [120](#)
- film\_transfer\_coeff
  - DOGFISH\_PARAM, [29](#)
  - MONKFISH\_PARAM, [88](#)
- FilmMTCoeff
  - egret.h, [149](#)
- finch
  - ui.h, [207](#)
- finch.h
  - Cartesian, [155](#)
  - Cylindrical, [155](#)
  - FINCH\_Picard, [155](#)
  - LARK\_PJFNK, [155](#)
  - LARK\_Picard, [155](#)
  - Spherical, [155](#)
- finch.h, [152](#)
  - check\_Mass, [156](#)
  - default\_bcs, [158](#)
  - default\_execution, [157](#)
  - default\_ic, [157](#)
  - default\_params, [158](#)
  - default\_postprocess, [159](#)
  - default\_precon, [159](#)
  - default\_preprocess, [157](#)
  - default\_res, [158](#)
  - default\_reset, [159](#)
  - default\_solve, [158](#)
  - default\_timestep, [157](#)

- FINCH\_TESTS, 159
- finch\_coord\_type, 155
- finch\_solve\_type, 155
- l\_direct, 156
- lark\_picard\_step, 156
- max, 155
- min, 155
- minmod, 156
- minmod\_discretization, 158
- nl\_picard, 156
- ospre\_discretization, 158
- print2file\_dim\_header, 157
- print2file\_newline, 157
- print2file\_result\_new, 157
- print2file\_result\_old, 157
- print2file\_tab, 157
- print2file\_time\_header, 157
- setup\_FINCH\_DATA, 156
- uAverage, 156
- uTotal, 156
- vanAlbada\_discretization, 158
- finch\_coord\_type
  - finch.h, 155
- finch\_dat
  - DOGFISH\_DATA, 28
  - MONKFISH\_DATA, 88
  - SCOPSOWL\_DATA, 109
  - SKUA\_DATA, 117
- finch\_solve\_type
  - finch.h, 155
- findAllTypes
  - KeyValueMap, 63
- findType
  - KeyValueMap, 63
  - ValueTypePair, 135
- Fk
  - BACKTRACK\_DATA, 15
- flock.h, 159
- Fn
  - FINCH\_DATA, 40
- Fnp1
  - FINCH\_DATA, 40
- Fobj
  - GSTA\_OPT\_DATA, 57
- fom
  - lark.h, 171
- formation\_energy
  - Molecule, 85
- formation\_enthalpy
  - Molecule, 85
- formation\_entropy
  - Molecule, 85
- Formula
  - Molecule, 85
- forward\_rate
  - UnsteadyReaction, 134
- forward\_ref\_rate
  - UnsteadyReaction, 134
- funeval
  - PJFNK\_DATA, 102
- Fv
  - PJFNK\_DATA, 101
- Fx
  - NUM\_JAC\_DATA, 90
- Fxp
  - NUM\_JAC\_DATA, 90
- GCR
  - lark.h, 169
- GMRESLP
  - lark.h, 169
- GMRESR
  - lark.h, 169
- GMRESRP
  - lark.h, 169
- GCR\_DATA, 42
  - alpha, 44
  - bestres, 44
  - bestx, 44
  - beta, 44
  - breakdown, 44
  - c, 45
  - c\_temp, 45
  - iter\_inner, 43
  - iter\_outer, 43
  - maxit, 43
  - Output, 44
  - r, 45
  - relres, 44
  - relres\_base, 44
  - res, 44
  - restart, 43
  - tol\_abs, 44
  - tol\_rel, 44
  - total\_iter, 43
  - transpose\_dat, 45
  - u, 45
  - u\_temp, 45
  - x, 44
- GCR\_Output
  - GMRESR\_DATA, 49
- gE
  - FINCH\_DATA, 40
- gl
  - FINCH\_DATA, 40
- GMRES\_Output
  - GMRESR\_DATA, 49
- GMRESLP\_DATA, 45
  - arnoldi\_dat, 47
  - bestres, 47
  - bestx, 47
  - iter, 46
  - maxit, 46
  - Output, 47
  - r, 47
  - relres, 47
  - relres\_base, 47

- res, [47](#)
- restart, [46](#)
- steps, [46](#)
- tol\_abs, [46](#)
- tol\_rel, [46](#)
- x, [47](#)
- GMRESR\_DATA, [47](#)
  - arg, [50](#)
  - GCR\_Output, [49](#)
  - GMRES\_Output, [49](#)
  - gcr\_abs\_tol, [49](#)
  - gcr\_dat, [50](#)
  - gcr\_maxit, [49](#)
  - gcr\_rel\_tol, [49](#)
  - gcr\_restart, [49](#)
  - gmres\_dat, [50](#)
  - gmres\_maxit, [49](#)
  - gmres\_restart, [49](#)
  - gmres\_tol, [49](#)
  - iter\_inner, [49](#)
  - iter\_outer, [49](#)
  - matvec, [50](#)
  - matvec\_data, [50](#)
  - N, [49](#)
  - term\_precon, [50](#)
  - terminal\_precon, [50](#)
  - total\_iter, [49](#)
- GMRESRP\_DATA, [50](#)
  - bestres, [52](#)
  - bestx, [53](#)
  - e0, [53](#)
  - e0\_bar, [53](#)
  - H, [53](#)
  - H\_bar, [53](#)
  - iter\_inner, [52](#)
  - iter\_outer, [52](#)
  - iter\_total, [52](#)
  - maxit, [52](#)
  - Output, [52](#)
  - r, [53](#)
  - relres, [52](#)
  - relres\_base, [52](#)
  - res, [52](#)
  - restart, [52](#)
  - sum, [53](#)
  - tol\_abs, [52](#)
  - tol\_rel, [52](#)
  - v, [53](#)
  - Vk, [53](#)
  - w, [53](#)
  - x, [52](#)
  - y, [53](#)
- GPAST\_DATA, [54](#)
  - gama\_inf, [55](#)
  - He, [54](#)
  - Plo, [55](#)
  - po, [55](#)
  - poi, [55](#)
  - present, [55](#)
  - q, [54](#)
  - qo, [55](#)
  - x, [54](#)
  - y, [54](#)
- GSTA\_DATA, [55](#)
  - dHo, [56](#)
  - dSo, [56](#)
  - m, [56](#)
  - qmax, [56](#)
- GSTA\_OPT\_DATA, [56](#)
  - all\_pars, [58](#)
  - best\_par, [57](#)
  - Fobj, [57](#)
  - iso, [57](#)
  - Kno, [58](#)
  - n\_par, [57](#)
  - norms, [58](#)
  - opt\_qmax, [58](#)
  - P, [57](#)
  - q, [57](#)
  - qmax, [57](#)
  - total\_eval, [57](#)
- gama
  - mSPD\_DATA, [90](#)
- gama\_inf
  - GPAST\_DATA, [55](#)
- gas\_dat
  - SCOPSOWL\_DATA, [109](#)
  - SKUA\_DATA, [117](#)
- gas\_temperature
  - MIXED\_GAS, [78](#)
  - SCOPSOWL\_DATA, [108](#)
- gas\_velocity
  - SCOPSOWL\_DATA, [108](#)
  - SKUA\_DATA, [116](#)
- gcr
  - lark.h, [175](#)
- gcr\_abs\_tol
  - GMRESR\_DATA, [49](#)
- gcr\_dat
  - GMRESR\_DATA, [50](#)
  - PJFNK\_DATA, [102](#)
- gcr\_maxit
  - GMRESR\_DATA, [49](#)
- gcr\_rel\_tol
  - GMRESR\_DATA, [49](#)
- gcr\_restart
  - GMRESR\_DATA, [49](#)
- generic\_error
  - error.h, [151](#)
- Get\_ActivationEnergy
  - UnsteadyReaction, [133](#)
- Get\_Affinity
  - UnsteadyReaction, [134](#)
- Get\_Delta
  - MassBalance, [65](#)
- Get\_Energy

- Reaction, [106](#)
- UnsteadyReaction, [133](#)
- Get\_Enthalpy
  - Reaction, [106](#)
  - UnsteadyReaction, [133](#)
- Get\_Entropy
  - Reaction, [106](#)
  - UnsteadyReaction, [133](#)
- Get\_Equilibrium
  - Reaction, [106](#)
  - UnsteadyReaction, [133](#)
- Get\_Forward
  - UnsteadyReaction, [133](#)
- Get\_ForwardRef
  - UnsteadyReaction, [133](#)
- Get\_InitialValue
  - UnsteadyReaction, [133](#)
- Get\_MaximumValue
  - UnsteadyReaction, [133](#)
- Get\_Name
  - MassBalance, [65](#)
- Get\_Reverse
  - UnsteadyReaction, [133](#)
- Get\_ReverseRef
  - UnsteadyReaction, [133](#)
- Get\_Species\_Index
  - UnsteadyReaction, [133](#)
- Get\_Stoichiometric
  - Reaction, [106](#)
  - UnsteadyReaction, [133](#)
- Get\_TimeStep
  - UnsteadyReaction, [134](#)
- Get\_TotalConcentration
  - MassBalance, [65](#)
- get\_index
  - MasterSpeciesList, [66](#)
- get\_species
  - MasterSpeciesList, [66](#)
- getAlias
  - Document, [25](#)
  - Header, [60](#)
  - SubHeader, [123](#)
- getAnchoredDoc
  - YamlWrapper, [139](#)
- getAnchoredHeader
  - Document, [25](#)
- getAnchoredSub
  - Header, [61](#)
- getBool
  - KeyValueMap, [63](#)
  - ValueTypePair, [135](#)
- getDataMap
  - Document, [24](#)
  - Header, [60](#)
- getDocFromHeadAlias
  - YamlWrapper, [139](#)
- getDocFromSubAlias
  - YamlWrapper, [139](#)
- getDocMap
  - YamlWrapper, [138](#)
- getDocument
  - YamlWrapper, [138](#)
- getDouble
  - KeyValueMap, [63](#)
  - ValueTypePair, [136](#)
- getHeadFromSubAlias
  - Document, [25](#)
- getHeadMap
  - Document, [24](#)
- getHeader
  - Document, [24](#)
- getInt
  - KeyValueMap, [63](#)
  - ValueTypePair, [136](#)
- getMap
  - KeyValueMap, [62](#)
  - SubHeader, [122](#)
- getName
  - Document, [25](#)
  - Header, [60](#)
  - SubHeader, [122](#)
- getPair
  - KeyValueMap, [63](#)
  - ValueTypePair, [136](#)
- getState
  - Document, [25](#)
  - Header, [60](#)
  - SubHeader, [123](#)
- getString
  - KeyValueMap, [63](#)
  - ValueTypePair, [135](#)
- getSubHeader
  - Header, [60](#)
- getSubMap
  - Header, [60](#)
- getType
  - KeyValueMap, [63](#)
  - ValueTypePair, [136](#)
- getValue
  - KeyValueMap, [63](#)
  - ValueTypePair, [136](#)
- getYamlWrapper
  - yaml\_cpp\_class, [137](#)
- gmres\_dat
  - GMRESR\_DATA, [50](#)
- gmres\_maxit
  - GMRESR\_DATA, [49](#)
- gmres\_restart
  - GMRESR\_DATA, [49](#)
- gmres\_tol
  - GMRESR\_DATA, [49](#)
- gmresLeftPreconditioned
  - lark.h, [170](#)
- gmresPreconditioner
  - lark.h, [175](#)
- gmresRightPreconditioned

- lark.h, [171](#)
- gmreslp\_dat
  - PJFNK\_DATA, [101](#)
- gmresr
  - lark.h, [176](#)
- gmresr\_dat
  - PJFNK\_DATA, [102](#)
- gmresrp\_dat
  - PJFNK\_DATA, [102](#)
- gpast\_dat
  - MAGPIE\_DATA, [64](#)
- grad\_mSPD
  - magpie.h, [185](#)
- Grav\_R
  - Trajectory.h, [204](#)
- Grav\_T
  - Trajectory.h, [204](#)
- gsta\_opt
  - ui.h, [208](#)
- gsta\_dat
  - MAGPIE\_DATA, [64](#)
- gsta\_opt.h, [160](#)
  - avgPar, [162](#)
  - avgValue, [163](#)
  - eduGuess, [164](#)
  - eval\_GSTA, [164](#)
  - gsta\_optimize, [164](#)
  - gstaFunc, [164](#)
  - gstaObjFunc, [164](#)
  - isSmooth, [163](#)
  - minIndex, [162](#)
  - minValue, [162](#)
  - Na, [162](#)
  - orderMag, [162](#)
  - orthoLinReg, [163](#)
  - Po, [162](#)
  - R, [162](#)
  - rSq, [163](#)
  - roundIt, [162](#)
  - twoFifths, [162](#)
  - weightedAvg, [163](#)
- gsta\_optimize
  - gsta\_opt.h, [164](#)
- gstaFunc
  - gsta\_opt.h, [164](#)
- gstaObjFunc
  - gsta\_opt.h, [164](#)
- H
  - GMRESRP\_DATA, [53](#)
  - TRAJECTORY\_DATA, [128](#)
- H0
  - TRAJECTORY\_DATA, [127](#)
- HELP
  - ui.h, [207](#)
- H\_bar
  - GMRESRP\_DATA, [53](#)
- Hamaker
  - TRAJECTORY\_DATA, [127](#)
- HaveEnergy
  - Molecule, [84](#)
- HaveEquil
  - Reaction, [107](#)
- haveEquilibrium
  - Reaction, [106](#)
  - UnsteadyReaction, [133](#)
- HaveForRef
  - UnsteadyReaction, [134](#)
- HaveForward
  - UnsteadyReaction, [134](#)
- HaveG
  - Reaction, [106](#)
- haveG
  - Molecule, [85](#)
- HaveHS
  - Molecule, [84](#)
  - Reaction, [106](#)
- haveHS
  - Molecule, [85](#)
- haveMinMax
  - MONKFISH\_DATA, [87](#)
- haveRate
  - UnsteadyReaction, [133](#)
- HaveRevRef
  - UnsteadyReaction, [134](#)
- HaveReverse
  - UnsteadyReaction, [134](#)
- He
  - GPAST\_DATA, [54](#)
  - magpie.h, [184](#)
- Head\_Map
  - Document, [25](#)
- Header, [58](#)
  - ~Header, [59](#)
  - addPair, [60](#)
  - addSubKey, [60](#)
  - begin, [60](#)
  - changeKey, [60](#)
  - clear, [60](#)
  - copyAnchor2Alias, [60](#)
  - DisplayContents, [60](#)
  - end, [60](#)
  - getAlias, [60](#)
  - getAnchoredSub, [61](#)
  - getDataMap, [60](#)
  - getName, [60](#)
  - getState, [60](#)
  - getSubHeader, [60](#)
  - getSubMap, [60](#)
  - Header, [59](#)
  - isAlias, [60](#)
  - isAnchor, [61](#)
  - operator(), [60](#)
  - operator=, [59](#)
  - resetKeys, [60](#)
  - setAlias, [60](#)
  - setName, [60](#)

- setNameAliasPair, 60
- setState, 60
- size, 60
- Sub\_Map, 61
- header\_state
  - yaml\_wrapper.h, 214
- help
  - ui.h, 208
- Heterogeneous
  - SCOPSOWL\_DATA, 108
- Hkp1
  - ARNOLDI\_DATA, 8
- hp1
  - ARNOLDI\_DATA, 8
- I
  - SYSTEM\_DATA, 125
- IDEAL
  - shark.h, 199
- INT
  - yaml\_wrapper.h, 214
- Ideal
  - SYSTEM\_DATA, 125
- ideal\_solution
  - shark.h, 199
- li
  - OPTRANS\_DATA, 91
- indexing\_error
  - error.h, 151
- initial\_error
  - error.h, 152
- initial\_guess\_SCOPSOWL
  - scopsowl\_opt.h, 197
- initial\_guess\_SKUA
  - skua\_opt.h, 203
- initial\_sorption
  - DOGFISH\_PARAM, 29
  - MONKFISH\_PARAM, 88
- initial\_value
  - UnsteadyReaction, 134
- initialGuess\_mSPD
  - magpie.h, 186
- Initialize\_List
  - MassBalance, 64
  - Reaction, 105
  - UnsteadyReaction, 132
- initialize\_data
  - egret.h, 149
- inner\_product
  - Matrix, 71
- input
  - ui.h, 209
- input\_file
  - yaml\_cpp\_class, 137
- input\_files
  - UI\_DATA, 129
- IntegralAvg
  - Matrix, 73
- IntegralTotal
  - Matrix, 73
- interparticle\_diffusion
  - MONKFISH\_PARAM, 88
- intraparticle\_diffusion
  - DOGFISH\_PARAM, 29
  - MONKFISH\_PARAM, 88
- invalid\_atom
  - error.h, 152
- invalid\_boolean
  - error.h, 151
- invalid\_components
  - error.h, 151
- invalid\_console\_input
  - error.h, 152
- invalid\_electron
  - error.h, 152
- invalid\_fraction
  - error.h, 151
- invalid\_gas\_sum
  - error.h, 151
- invalid\_molefraction
  - error.h, 151
- invalid\_neutron
  - error.h, 152
- invalid\_norm
  - error.h, 152
- invalid\_proton
  - error.h, 152
- invalid\_size
  - error.h, 151
- invalid\_solid\_sum
  - error.h, 151
- invalid\_species
  - error.h, 152
- invalid\_type
  - error.h, 152
- invalid\_valence
  - error.h, 152
- invalid\_input
  - ui.h, 211
- inverse
  - Matrix, 72
- isAlias
  - Document, 25
  - Header, 60
  - SubHeader, 123
- isAnchor
  - Document, 25
  - Header, 61
  - SubHeader, 123
- isRegistered
  - Molecule, 84
- isSmooth
  - gsta\_opt.h, 163
- iso
  - GSTA\_OPT\_DATA, 57
- iter
  - ARNOLDI\_DATA, 8

- BiCGSTAB\_DATA, [17](#)
- CGS\_DATA, [20](#)
- GMRESLP\_DATA, [46](#)
- PCG\_DATA, [93](#)
- PICARD\_DATA, [97](#)
- iter\_inner
  - GCR\_DATA, [43](#)
  - GMRESR\_DATA, [49](#)
  - GMRESRP\_DATA, [52](#)
- iter\_outer
  - GCR\_DATA, [43](#)
  - GMRESR\_DATA, [49](#)
  - GMRESRP\_DATA, [52](#)
- iter\_total
  - GMRESRP\_DATA, [52](#)
- Iterative
  - FINCH\_DATA, [36](#)
- J
  - SYSTEM\_DATA, [125](#)
- Jacobian
  - Speciation\_Test01\_Data, [121](#)
- jacvec
  - lark.h, [177](#)
- K
  - SYSTEM\_DATA, [125](#)
- k
  - ARNOLDI\_DATA, [8](#)
  - TRAJECTORY\_DATA, [127](#)
- kB
  - magpie.h, [183](#)
- kIC
  - FINCH\_DATA, [35](#)
- key\_not\_found
  - error.h, [152](#)
- Key\_Value
  - KeyValueMap, [63](#)
- KeyValueMap, [61](#)
  - ~KeyValueMap, [62](#)
  - addKey, [62](#)
  - addPair, [62](#)
  - assertType, [63](#)
  - begin, [62](#)
  - clear, [62](#)
  - DisplayMap, [63](#)
  - editValue4Key, [62](#)
  - end, [62](#)
  - findAllTypes, [63](#)
  - findType, [63](#)
  - getBool, [63](#)
  - getDouble, [63](#)
  - getInt, [63](#)
  - getMap, [62](#)
  - getPair, [63](#)
  - getString, [63](#)
  - getType, [63](#)
  - getValue, [63](#)
  - Key\_Value, [63](#)
  - KeyValueMap, [62](#)
  - KeyValueMap, [62](#)
  - operator=, [62](#)
  - size, [63](#)
- kfn
  - FINCH\_DATA, [35](#)
- kfnp1
  - FINCH\_DATA, [36](#)
- kinematic\_viscosity
  - MIXED\_GAS, [78](#)
- kn
  - FINCH\_DATA, [40](#)
- Kno
  - GSTA\_OPT\_DATA, [58](#)
- knp1
  - FINCH\_DATA, [40](#)
- ko
  - FINCH\_DATA, [35](#)
- krylov\_method
  - lark.h, [169](#)
- L
  - FINCH\_DATA, [34](#)
  - TRAJECTORY\_DATA, [127](#)
- LARK\_PJFNK
  - finch.h, [155](#)
- LARK\_Picard
  - finch.h, [155](#)
- L\_Output
  - PJFNK\_DATA, [101](#)
- L\_direct
  - finch.h, [156](#)
- L\_iter
  - PJFNK\_DATA, [99](#)
- L\_wire
  - TRAJECTORY\_DATA, [127](#)
- LARGE\_CYCLE\_TEST01
  - scopsowl.h, [196](#)
- LARK\_TESTS
  - lark.h, [179](#)
- LN
  - FINCH\_DATA, [36](#)
- LOCATION
  - Trajectory.h, [205](#)
- ladshawSolve
  - Matrix, [72](#)
- lambda\_E
  - FINCH\_DATA, [36](#)
- lambda\_I
  - FINCH\_DATA, [36](#)
- lambdaMin
  - BACKTRACK\_DATA, [15](#)
- lark
  - ui.h, [207](#)
- lark.h
  - BiCGSTAB, [169](#)
  - CGS, [169](#)
  - FOM, [169](#)
  - GCR, [169](#)

- GMRESLP, [169](#)
- GMRESR, [169](#)
- GMRESRP, [169](#)
- PCG, [169](#)
- lark.h, [165](#)
  - arnoldi, [169](#)
  - backtrackLineSearch, [177](#)
  - bicgstab, [173](#)
  - cgs, [173](#)
  - fom, [171](#)
  - gcr, [175](#)
  - gmresLeftPreconditioned, [170](#)
  - gmresPreconditioner, [175](#)
  - gmresRightPreconditioned, [171](#)
  - gmresr, [176](#)
  - jacvec, [177](#)
  - krylov\_method, [169](#)
  - LARK\_TESTS, [179](#)
  - NumericalJacobian, [179](#)
  - operatorTranspose, [174](#)
  - pcg, [172](#)
  - picard, [176](#)
  - pjfnk, [178](#)
  - update\_arnoldi\_solution, [169](#)
- lark\_picard\_step
  - finch.h, [156](#)
- level
  - MONKFISH\_DATA, [87](#)
  - SCOPSOWL\_DATA, [108](#)
- lin\_precon
  - SHARK\_DATA, [115](#)
- lin\_tol\_abs
  - PJFNK\_DATA, [100](#)
- lin\_tol\_rel
  - PJFNK\_DATA, [100](#)
- LineSearch
  - PJFNK\_DATA, [101](#)
- linear\_solver
  - PJFNK\_DATA, [100](#)
- linearsolve\_choice
  - shark.h, [199](#)
- linsearch\_choice
  - shark.h, [199](#)
- List
  - MassBalance, [65](#)
  - Mechanism, [76](#)
  - Reaction, [106](#)
- list\_size
  - MasterSpeciesList, [66](#)
- InKo
  - magpie.h, [183](#)
- Inact\_mSPD
  - magpie.h, [185](#)
- logC
  - Speciation\_Test01\_Data, [121](#)
- logKa1
  - Speciation\_Test01\_Data, [120](#)
- logKa2
  - Speciation\_Test01\_Data, [120](#)
- logKw
  - Speciation\_Test01\_Data, [120](#)
- lowerHessenberg2Triangular
  - Matrix, [74](#)
- lowerHessenbergSolve
  - Matrix, [75](#)
- lowerTriangularSolve
  - Matrix, [74](#)
- M
  - TRAJECTORY\_DATA, [127](#)
- m
  - GSTA\_DATA, [56](#)
- M\_PI
  - macaw.h, [180](#)
- m\_rand
  - TRAJECTORY\_DATA, [128](#)
- MACAW\_TESTS
  - macaw.h, [180](#)
- MAGPIE
  - magpie.h, [187](#)
- MAGPIE\_DATA, [63](#)
  - gpast\_dat, [64](#)
  - gsta\_dat, [64](#)
  - mspd\_dat, [64](#)
  - sys\_dat, [64](#)
- MAGPIE\_SCENARIOS
  - magpie.h, [188](#)
- ME
  - FINCH\_DATA, [38](#)
- mError
  - error.h, [151](#)
- MI
  - FINCH\_DATA, [38](#)
- MIXED\_GAS, [76](#)
  - binary\_diffusion, [78](#)
  - char\_length, [78](#)
  - CheckMolefractions, [77](#)
  - gas\_temperature, [78](#)
  - kinematic\_viscosity, [78](#)
  - molefraction, [78](#)
  - N, [77](#)
  - Reynolds, [78](#)
  - species\_dat, [78](#)
  - total\_density, [78](#)
  - total\_dyn\_vis, [78](#)
  - total\_molecular\_weight, [78](#)
  - total\_pressure, [77](#)
  - total\_specific\_heat, [78](#)
  - velocity, [78](#)
- MOLA\_TESTS
  - mola.h, [192](#)
- MONKFISH\_DATA, [86](#)
  - avg\_fiber\_density, [87](#)
  - DirichletBC, [87](#)
  - dog\_dat, [88](#)
  - domain\_diameter, [87](#)
  - end\_time, [87](#)



- eval\_Cex, [88](#)
- eval\_Dex, [87](#)
- eval\_Ret, [88](#)
- eval\_ads, [87](#)
- eval\_eps, [87](#)
- eval\_kf, [88](#)
- eval\_rho, [87](#)
- finch\_dat, [88](#)
- haveMinMax, [87](#)
- level, [87](#)
- max\_fiber\_density, [87](#)
- max\_porosity, [87](#)
- min\_fiber\_density, [87](#)
- min\_porosity, [87](#)
- MultiScale, [87](#)
- NonLinear, [87](#)
- NumComp, [87](#)
- Output, [87](#)
- param\_dat, [88](#)
- Print2Console, [87](#)
- Print2File, [87](#)
- single\_fiber\_density, [87](#)
- t\_counter, [87](#)
- t\_print, [87](#)
- time, [87](#)
- time\_old, [87](#)
- total\_sorption, [87](#)
- total\_sorption\_old, [87](#)
- total\_steps, [87](#)
- user\_data, [88](#)
- MONKFISH\_PARAM, [88](#)
  - avg\_sorption, [89](#)
  - avg\_sorption\_old, [89](#)
  - exterior\_concentration, [88](#)
  - exterior\_transfer\_coeff, [88](#)
  - film\_transfer\_coeff, [88](#)
  - initial\_sorption, [88](#)
  - interparticle\_diffusion, [88](#)
  - intraparticle\_diffusion, [88](#)
  - sorbed\_molefraction, [88](#)
  - sorption\_bc, [88](#)
  - species, [89](#)
- MONKFISH\_TESTS
  - monkfish.h, [193](#)
- mSPD\_DATA, [89](#)
  - eMax, [89](#)
  - eta, [89](#)
  - gama, [90](#)
  - s, [89](#)
  - v, [89](#)
- macaw
  - ui.h, [207](#)
- macaw.h, [179](#)
  - M\_PI, [180](#)
  - MACAW\_TESTS, [180](#)
- Magnetic\_R
  - Trajectory.h, [204](#)
- Magnetic\_T
  - Trajectory.h, [204](#)
- magpie
  - ui.h, [208](#)
- magpie.h, [181](#)
  - A, [183](#)
  - DBL\_EPSILON, [183](#)
  - dq\_dp, [184](#)
  - eMax, [185](#)
  - eval\_GPAST, [187](#)
  - eval\_eta, [187](#)
  - eval\_po, [187](#)
  - eval\_po\_PI, [186](#)
  - eval\_po\_qo, [186](#)
  - grad\_mSPD, [185](#)
  - He, [184](#)
  - initialGuess\_mSPD, [186](#)
  - kB, [183](#)
  - InKo, [183](#)
  - Inact\_mSPD, [185](#)
  - MAGPIE, [187](#)
  - MAGPIE\_SCENARIOS, [188](#)
  - Na, [183](#)
  - PI, [184](#)
  - Po, [183](#)
  - q\_p, [184](#)
  - qT, [185](#)
  - qo, [184](#)
  - Qst, [185](#)
  - R, [183](#)
  - shapeFactor, [183](#)
  - V, [183](#)
  - Z, [183](#)
- magpie\_reverse\_error
  - error.h, [151](#)
- magpie\_dat
  - SCOPSOWL\_DATA, [109](#)
  - SKUA\_DATA, [117](#)
- MassBalance, [64](#)
  - ~MassBalance, [64](#)
  - Delta, [65](#)
  - Display\_Info, [65](#)
  - Eval\_Residual, [65](#)
  - Get\_Delta, [65](#)
  - Get\_Name, [65](#)
  - Get\_TotalConcentration, [65](#)
  - Initialize\_List, [64](#)
  - List, [65](#)
  - MassBalance, [64](#)
  - MassBalance, [64](#)
  - Name, [65](#)
  - Set\_Delta, [65](#)
  - Set\_Name, [65](#)
  - Set\_TotalConcentration, [65](#)
  - Sum\_Delta, [65](#)
  - TotalConcentration, [65](#)
- MassBalanceList
  - SHARK\_DATA, [114](#)
- MasterList

- SHARK\_DATA, 114
- MasterSpeciesList, 65
  - ~MasterSpeciesList, 66
  - alkalinity, 67
  - charge, 66
  - DisplayAll, 66
  - DisplayConcentrations, 66
  - DisplayInfo, 66
  - Eval\_ChargeResidual, 67
  - get\_index, 66
  - get\_species, 66
  - list\_size, 66
  - MasterSpeciesList, 66
  - MasterSpeciesList, 66
  - operator=, 66
  - residual\_alkalinity, 67
  - set\_alkalinity, 66
  - set\_list\_size, 66
  - set\_species, 66
  - size, 67
  - species, 67
  - speciesName, 67
- Matrix
  - ~Matrix, 70
  - adjoint, 72
  - cofactor, 71
  - columnExtend, 75
  - columnExtract, 75
  - columnProjection, 74
  - columnReplace, 75
  - columnShrink, 75
  - columnVectorFill, 74
  - columns, 71
  - ConstantICFill, 73
  - Data, 76
  - determinate, 71
  - diagonalSolve, 74
  - dirichletBCFill, 74
  - Display, 72
  - edit, 70
  - inner\_product, 71
  - IntegralAvg, 73
  - IntegralTotal, 73
  - inverse, 72
  - ladshawSolve, 72
  - lowerHessenberg2Triangular, 74
  - lowerHessenbergSolve, 75
  - lowerTriangularSolve, 74
  - Matrix, 70
  - naturalLaplacian3D, 72
  - norm, 71
  - num\_cols, 76
  - num\_rows, 76
  - operator\*, 71
  - operator(), 70
  - operator+, 71
  - operator-, 71
  - operator/, 71
  - operator=, 70
  - rowExtend, 75
  - rowExtract, 75
  - rowReplace, 75
  - rowShrink, 75
  - rows, 71
  - set\_size, 70
  - SolnTransform, 73
  - sphericalAvg, 73
  - sphericalBCFill, 72
  - sum, 71
  - transpose, 72
  - transpose\_multiply, 72
  - tridiagonalFill, 72
  - tridiagonalSolve, 72
  - tridiagonalVectorFill, 73
  - upperHessenberg2Triangular, 74
  - upperHessenbergSolve, 75
  - upperTriangularSolve, 74
  - zeros, 70
- Matrix< T >, 67
- matrix\_too\_small
  - error.h, 151
- matvec
  - GMRESR\_DATA, 50
- matvec\_mis\_match
  - error.h, 151
- matvec\_data
  - GMRESR\_DATA, 50
- max
  - finch.h, 155
  - UI\_DATA, 129
- max\_bias
  - SCOPSOWL\_OPT\_DATA, 110
  - SKUA\_OPT\_DATA, 118
- max\_fiber\_density
  - MONKFISH\_DATA, 87
- max\_guess\_iter
  - SCOPSOWL\_OPT\_DATA, 110
  - SKUA\_OPT\_DATA, 118
- max\_iter
  - FINCH\_DATA, 37
- max\_norm
  - SYSTEM\_DATA, 125
- max\_porosity
  - MONKFISH\_DATA, 87
- max\_value
  - UnsteadyReaction, 134
- maxit
  - BiCGSTAB\_DATA, 17
  - CGS\_DATA, 20
  - GCR\_DATA, 43
  - GMRESLP\_DATA, 46
  - GMRESRP\_DATA, 52
  - PCG\_DATA, 93
  - PICARD\_DATA, 97
- Mechanism, 76
  - List, 76

- reactions, 76
- species\_index, 76
- weight, 76
- min
  - finch.h, 155
- min\_bias
  - SCOPSOWL\_OPT\_DATA, 110
  - SKUA\_OPT\_DATA, 118
- min\_fiber\_density
  - MONKFISH\_DATA, 87
- min\_porosity
  - MONKFISH\_DATA, 87
- minIndex
  - gsta\_opt.h, 162
- minValue
  - gsta\_opt.h, 162
- minmod
  - finch.h, 156
- minmod\_discretization
  - finch.h, 158
- missing\_information
  - error.h, 152
- MissingArg
  - UI\_DATA, 130
- mola
  - ui.h, 207
- mola.h, 190
  - MOLA\_TESTS, 192
- molar\_weight
  - Molecule, 85
- MolarWeight
  - Molecule, 84
- molecular\_diffusion
  - PURE\_GAS, 104
- molecular\_weight
  - PURE\_GAS, 104
- MolecularFormula
  - Molecule, 84
- Molecule, 79
  - ~Molecule, 81
  - atoms, 85
  - calculateAvgOxiState, 83
  - Charge, 83
  - charge, 85
  - DisplayInfo, 84
  - editAllOxidationStates, 83
  - editCharge, 82
  - editEnergy, 83
  - editEnthalpy, 83
  - editEntropy, 83
  - editHS, 83
  - editOneOxidationState, 82
  - Energy, 84
  - Enthalpy, 84
  - Entropy, 84
  - formation\_energy, 85
  - formation\_enthalpy, 85
  - formation\_entropy, 85
  - Formula, 85
  - HaveEnergy, 84
  - haveG, 85
  - HaveHS, 84
  - haveHS, 85
  - isRegistered, 84
  - molar\_weight, 85
  - MolarWeight, 84
  - MolecularFormula, 84
  - Molecule, 81
  - MoleculeName, 84
  - MoleculePhase, 84
  - Name, 85
  - Phase, 85
  - recalculateMolarWeight, 82
  - Register, 82
  - registered, 85
  - removeAllAtoms, 83
  - removeOneAtom, 83
  - setFormula, 82
  - setMolarWeight, 82
- MoleculeName
  - Molecule, 84
- MoleculePhase
  - Molecule, 84
- molefraction
  - MIXED\_GAS, 78
- molefractionCheck
  - skua.h, 202
- monkfish
  - ui.h, 208
- monkfish.h, 192
  - default\_density, 193
  - default\_exterior\_concentration, 193
  - default\_film\_transfer, 193
  - default\_interparticle\_diffusion, 193
  - default\_monk\_adsorption, 193
  - default\_monk\_equilibrium, 193
  - default\_monkfish\_retardation, 193
  - default\_porosity, 193
  - MONKFISH\_TESTS, 193
  - setup\_MONKFISH\_DATA, 193
- mp
  - TRAJECTORY\_DATA, 127
- Ms
  - TRAJECTORY\_DATA, 127
- mspd\_dat
  - MAGPIE\_DATA, 64
- Mu
  - egret.h, 148
- mu\_0
  - TRAJECTORY\_DATA, 127
- MultiScale
  - MONKFISH\_DATA, 87
- N
  - GMRESR\_DATA, 49
  - MIXED\_GAS, 77
  - Speciation\_Test01\_Data, 120

- SYSTEM\_DATA, 124
- NONE
  - yaml\_wrapper.h, 214
- n\_par
  - GSTA\_OPT\_DATA, 57
- n\_rand
  - TRAJECTORY\_DATA, 128
- NE
  - FINCH\_DATA, 38
- NI
  - FINCH\_DATA, 38
- NL\_Output
  - PJFNK\_DATA, 100
- NUM\_JAC\_DATA, 90
  - dxj, 90
  - eps, 90
  - Fx, 90
  - Fxp, 90
- Na
  - gsta\_opt.h, 162
  - magpie.h, 183
- NaT
  - Speciation\_Test01\_Data, 121
- Name
  - Atom, 13
  - MassBalance, 65
  - Molecule, 85
- name
  - SubHeader, 123
- naturalLaplacian3D
  - Matrix, 72
- NaturalState
  - Atom, 14
- negative\_mass
  - error.h, 151
- negative\_time
  - error.h, 151
- Neutrons
  - Atom, 12
- neutrons
  - Atom, 13
- Newton\_data
  - SHARK\_DATA, 115
- nl\_bestres
  - PJFNK\_DATA, 100
- nl\_iter
  - PJFNK\_DATA, 99
- nl\_maxit
  - PJFNK\_DATA, 100
- nl\_method
  - FINCH\_DATA, 37
- nl\_picard
  - finch.h, 156
- nl\_relres
  - PJFNK\_DATA, 100
- nl\_res
  - PJFNK\_DATA, 100
- nl\_res\_base
  - PJFNK\_DATA, 100
- nl\_tol\_abs
  - PJFNK\_DATA, 100
- nl\_tol\_rel
  - PJFNK\_DATA, 100
- no\_diffusion
  - error.h, 151
- non\_real\_edge
  - error.h, 152
- non\_square\_matrix
  - error.h, 151
- NonLinear
  - DOGFISH\_DATA, 27
  - MONKFISH\_DATA, 87
  - SCOPSOWL\_DATA, 108
  - SKUA\_DATA, 117
- Norm
  - SHARK\_DATA, 114
- norm
  - Matrix, 71
- normFkp1
  - BACKTRACK\_DATA, 15
- NormTrack
  - FINCH\_DATA, 36
- norms
  - GSTA\_OPT\_DATA, 58
- not\_a\_token
  - error.h, 152
- Nu
  - egret.h, 148
- nullptr\_error
  - error.h, 152
- nullptr\_func
  - error.h, 151
- num\_cols
  - Matrix, 76
- num\_curves
  - SCOPSOWL\_OPT\_DATA, 110
  - SKUA\_OPT\_DATA, 118
- num\_mbe
  - SHARK\_DATA, 114
- num\_other
  - SHARK\_DATA, 114
- num\_params
  - SCOPSOWL\_OPT\_DATA, 110
  - SKUA\_OPT\_DATA, 118
- num\_rows
  - Matrix, 76
- num\_ssr
  - SHARK\_DATA, 114
- num\_usr
  - SHARK\_DATA, 114
- NumComp
  - DOGFISH\_DATA, 27
  - MONKFISH\_DATA, 87
- NumJac
  - Speciation\_Test01\_Data, 121
- Number\_Generator

- Trajectory.h, 205
- number\_elements
  - PeriodicTable, 95
- number\_files
  - ui.h, 210
- NumericalJacobian
  - lark.h, 179
- numvar
  - SHARK\_DATA, 114
- OE
  - FINCH\_DATA, 38
- OI
  - FINCH\_DATA, 38
- OPTRANS\_DATA, 91
  - Ai, 91
  - Ii, 91
- omega
  - BiCGSTAB\_DATA, 17
- omega\_old
  - BiCGSTAB\_DATA, 17
- operator\*
  - Matrix, 71
- operator()
  - Document, 24
  - Header, 60
  - Matrix, 70
  - YamlWrapper, 138
- operator+
  - Matrix, 71
- operator-
  - Matrix, 71
- operator/
  - Matrix, 71
- operator=
  - Document, 24
  - Header, 59
  - KeyValuePair, 62
  - MasterSpeciesList, 66
  - Matrix, 70
  - SubHeader, 122
  - ValueTypePair, 135
  - YamlWrapper, 138
- operatorTranspose
  - lark.h, 174
- opt\_no\_support
  - error.h, 151
- opt\_qmax
  - GSTA\_OPT\_DATA, 58
- Optimize
  - SCOPSOWL\_OPT\_DATA, 110
  - SKUA\_OPT\_DATA, 118
- option
  - UI\_DATA, 130
- orderMag
  - gsta\_opt.h, 162
- ortho\_check\_fail
  - error.h, 151
- orthoLinReg
  - gsta\_opt.h, 163
- ospre\_discretization
  - finch.h, 158
- other\_data
  - SHARK\_DATA, 115
- OtherList
  - SHARK\_DATA, 114
- out\_of\_bounds
  - error.h, 151
- Output
  - ARNOLDI\_DATA, 8
  - BiCGSTAB\_DATA, 18
  - CGS\_DATA, 21
  - GCR\_DATA, 44
  - GMRESLP\_DATA, 47
  - GMRESRP\_DATA, 52
  - MONKFISH\_DATA, 87
  - PCG\_DATA, 93
  - PICARD\_DATA, 97
  - SYSTEM\_DATA, 125
- OutputFile
  - DOGFISH\_DATA, 27
  - SCOPSOWL\_DATA, 109
  - SHARK\_DATA, 115
  - SKUA\_DATA, 117
- owl\_dat
  - SCOPSOWL\_OPT\_DATA, 111
- oxidation\_state
  - Atom, 13
- OxidationState
  - Atom, 12
- P
  - GSTA\_OPT\_DATA, 57
- p
  - BiCGSTAB\_DATA, 18
  - CGS\_DATA, 22
  - PCG\_DATA, 94
- PCG
  - lark.h, 169
- PITZER
  - shark.h, 199
- PCG\_DATA, 91
  - alpha, 93
  - Ap, 94
  - bestres, 93
  - bestx, 93
  - beta, 93
  - iter, 93
  - maxit, 93
  - Output, 93
  - p, 94
  - r, 94
  - r\_old, 94
  - relres, 93
  - relres\_base, 93
  - res, 93
  - tol\_abs, 93
  - tol\_rel, 93

- x, [93](#)
  - z, [94](#)
  - z\_old, [94](#)
- PE3
  - egret.h, [148](#)
- pH
  - SHARK\_DATA, [114](#)
- pH\_index
  - SHARK\_DATA, [114](#)
- PI
  - magpie.h, [184](#)
  - SYSTEM\_DATA, [124](#)
- PICARD\_DATA, [96](#)
  - bestres, [97](#)
  - bestx, [97](#)
  - iter, [97](#)
  - maxit, [97](#)
  - Output, [97](#)
  - r, [97](#)
  - relres, [97](#)
  - relres\_base, [97](#)
  - res, [97](#)
  - tol\_abs, [97](#)
  - tol\_rel, [97](#)
  - x0, [97](#)
- Plo
  - GPAST\_DATA, [55](#)
- PJFNK\_DATA, [98](#)
  - backtrack\_dat, [102](#)
  - bestx, [101](#)
  - bicgstab\_dat, [101](#)
  - Bounce, [101](#)
  - cgs\_dat, [101](#)
  - eps, [100](#)
  - F, [101](#)
  - funeval, [102](#)
  - Fv, [101](#)
  - gcr\_dat, [102](#)
  - gmreslp\_dat, [101](#)
  - gmresr\_dat, [102](#)
  - gmresrp\_dat, [102](#)
  - L\_Output, [101](#)
  - l\_iter, [99](#)
  - lin\_tol\_abs, [100](#)
  - lin\_tol\_rel, [100](#)
  - LineSearch, [101](#)
  - linear\_solver, [100](#)
  - NL\_Output, [100](#)
  - nl\_bestres, [100](#)
  - nl\_iter, [99](#)
  - nl\_maxit, [100](#)
  - nl\_relres, [100](#)
  - nl\_res, [100](#)
  - nl\_res\_base, [100](#)
  - nl\_tol\_abs, [100](#)
  - nl\_tol\_rel, [100](#)
  - pcg\_dat, [101](#)
  - precon, [102](#)
  - precon\_data, [102](#)
  - res\_data, [102](#)
  - v, [101](#)
  - x, [101](#)
- pOH\_index
  - SHARK\_DATA, [114](#)
- POL
  - TRAJECTORY\_DATA, [128](#)
- POLAR
  - Trajectory.h, [204](#)
- PSI
  - egret.h, [148](#)
- PT
  - SYSTEM\_DATA, [124](#)
- PURE\_GAS, [103](#)
  - density, [104](#)
  - dynamic\_viscosity, [104](#)
  - molecular\_diffusion, [104](#)
  - molecular\_weight, [104](#)
  - Schmidt, [104](#)
  - specific\_heat, [104](#)
  - Sutherland\_Const, [104](#)
  - Sutherland\_Temp, [104](#)
  - Sutherland\_Viscosity, [104](#)
- Par
  - SYSTEM\_DATA, [125](#)
- param\_dat
  - DOGFISH\_DATA, [28](#)
  - MONKFISH\_DATA, [88](#)
  - SCOPSOWL\_DATA, [109](#)
  - SKUA\_DATA, [117](#)
- param\_data
  - FINCH\_DATA, [42](#)
- param\_guess
  - SCOPSOWL\_OPT\_DATA, [110](#)
  - SKUA\_OPT\_DATA, [118](#)
- param\_guess\_old
  - SCOPSOWL\_OPT\_DATA, [111](#)
  - SKUA\_OPT\_DATA, [119](#)
- ParamFile
  - SCOPSOWL\_OPT\_DATA, [111](#)
  - SKUA\_OPT\_DATA, [119](#)
- Path
  - UI\_DATA, [130](#)
- path
  - ui.h, [209](#)
  - UI\_DATA, [129](#)
- pcg
  - lark.h, [172](#)
- pcg\_dat
  - PJFNK\_DATA, [101](#)
- pellet\_density
  - SCOPSOWL\_DATA, [108](#)
- pellet\_radius
  - SCOPSOWL\_DATA, [108](#)
  - SKUA\_DATA, [116](#)
- PeriodicTable, [94](#)
  - ~PeriodicTable, [95](#)

- DisplayTable, [95](#)
- number\_elements, [95](#)
- PeriodicTable, [95](#)
- PeriodicTable, [95](#)
- Table, [95](#)
- Phase
  - Molecule, [85](#)
- pi
  - SYSTEM\_DATA, [124](#)
- picard
  - lark.h, [176](#)
- picard\_dat
  - FINCH\_DATA, [42](#)
- pjfnk
  - lark.h, [178](#)
- pjfnk\_dat
  - FINCH\_DATA, [42](#)
- Po
  - egret.h, [148](#)
  - gsta\_opt.h, [162](#)
  - magpie.h, [183](#)
- po
  - GPAST\_DATA, [55](#)
- poi
  - GPAST\_DATA, [55](#)
- pore\_diffusion
  - SCOPSOWL\_PARAM\_DATA, [112](#)
- porosity
  - TRAJECTORY\_DATA, [127](#)
- Precipitation, [102](#)
- precon
  - PJFNK\_DATA, [102](#)
- precon\_data
  - PJFNK\_DATA, [102](#)
  - SHARK\_DATA, [115](#)
- pres
  - FINCH\_DATA, [40](#)
- present
  - GPAST\_DATA, [55](#)
- previous\_token
  - yaml\_cpp\_class, [137](#)
- Print2Console
  - DOGFISH\_DATA, [26](#)
  - MONKFISH\_DATA, [87](#)
  - SCOPSOWL\_DATA, [108](#)
  - SKUA\_DATA, [116](#)
- Print2File
  - DOGFISH\_DATA, [26](#)
  - MONKFISH\_DATA, [87](#)
  - SCOPSOWL\_DATA, [108](#)
  - SKUA\_DATA, [116](#)
- print2file\_DOGFISH\_header
  - dogfish.h, [142](#)
- print2file\_DOGFISH\_result\_new
  - dogfish.h, [143](#)
- print2file\_DOGFISH\_result\_old
  - dogfish.h, [142](#)
- print2file\_SCOPSOWL\_header
  - scopsowl.h, [195](#)
- print2file\_SCOPSOWL\_result\_new
  - scopsowl.h, [196](#)
- print2file\_SCOPSOWL\_result\_old
  - scopsowl.h, [195](#)
- print2file\_SCOPSOWL\_time\_header
  - scopsowl.h, [195](#)
- print2file\_SKUA\_header
  - skua.h, [202](#)
- print2file\_SKUA\_results\_new
  - skua.h, [202](#)
- print2file\_SKUA\_results\_old
  - skua.h, [202](#)
- print2file\_SKUA\_time\_header
  - skua.h, [202](#)
- print2file\_dim\_header
  - finch.h, [157](#)
- print2file\_newline
  - finch.h, [157](#)
- print2file\_result\_new
  - finch.h, [157](#)
- print2file\_result\_old
  - finch.h, [157](#)
- print2file\_shark\_header
  - shark.h, [199](#)
- print2file\_shark\_info
  - shark.h, [199](#)
- print2file\_shark\_results\_new
  - shark.h, [199](#)
- print2file\_shark\_results\_old
  - shark.h, [199](#)
- print2file\_species\_header
  - dogfish.h, [142](#)
  - scopsowl.h, [195](#)
  - skua.h, [202](#)
- print2file\_tab
  - finch.h, [157](#)
- print2file\_time\_header
  - finch.h, [157](#)
- Protons
  - Atom, [12](#)
- protons
  - Atom, [13](#)
- Pstd
  - egret.h, [148](#)
- q
  - GPAST\_DATA, [54](#)
  - GSTA\_OPT\_DATA, [57](#)
- q\_bar
  - TRAJECTORY\_DATA, [128](#)
- q\_data
  - SCOPSOWL\_OPT\_DATA, [111](#)
  - SKUA\_OPT\_DATA, [119](#)
- Q\_in
  - TRAJECTORY\_DATA, [127](#)
- q\_p
  - magpie.h, [184](#)
- q\_sim

- SCOPSOWL\_OPT\_DATA, 111
- SKUA\_OPT\_DATA, 119
- qAvg
  - SCOPSOWL\_PARAM\_DATA, 112
- qAvg\_old
  - SCOPSOWL\_PARAM\_DATA, 112
- qIntegralAvg
  - SCOPSOWL\_PARAM\_DATA, 112
- qIntegralAvg\_old
  - SCOPSOWL\_PARAM\_DATA, 112
- qT
  - magpie.h, 185
  - SYSTEM\_DATA, 124
- qTn
  - SKUA\_DATA, 116
- qTnp1
  - SKUA\_DATA, 116
- qmax
  - GSTA\_DATA, 56
  - GSTA\_OPT\_DATA, 57
- qo
  - GPAST\_DATA, 55
  - magpie.h, 184
  - SCOPSOWL\_PARAM\_DATA, 112
- Qst
  - magpie.h, 185
  - SCOPSOWL\_PARAM\_DATA, 112
- Qst\_old
  - SCOPSOWL\_PARAM\_DATA, 112
- QstAvg
  - SCOPSOWL\_PARAM\_DATA, 112
- QstAvg\_old
  - SCOPSOWL\_PARAM\_DATA, 112
- Qstn
  - SKUA\_PARAM, 120
- Qstnp1
  - SKUA\_PARAM, 120
- Qsto
  - SCOPSOWL\_PARAM\_DATA, 112
- R
  - gsta\_opt.h, 162
  - magpie.h, 183
- r
  - BiCGSTAB\_DATA, 18
  - CGS\_DATA, 21
  - GCR\_DATA, 45
  - GMRESLP\_DATA, 47
  - GMRESRP\_DATA, 53
  - PCG\_DATA, 94
  - PICARD\_DATA, 97
- r0
  - BiCGSTAB\_DATA, 18
  - CGS\_DATA, 21
- r\_old
  - PCG\_DATA, 94
- RADIAL\_FORCE
  - Trajectory.h, 204
- RE3
  - egret.h, 148
- RIC
  - FINCH\_DATA, 35
- rSq
  - gsta\_opt.h, 163
- RUN\_SANDBOX
  - sandbox.h, 194
- ReNum
  - egret.h, 149
- Reaction, 104
  - ~Reaction, 105
  - calculateEnergies, 106
  - calculateEquilibrium, 106
  - CanCalcG, 106
  - CanCalcHS, 106
  - checkSpeciesEnergies, 106
  - Display\_Info, 106
  - energy, 106
  - enthalpy, 106
  - entropy, 106
  - Equilibrium, 106
  - Eval\_Residual, 106
  - Get\_Energy, 106
  - Get\_Enthalpy, 106
  - Get\_Entropy, 106
  - Get\_Equilibrium, 106
  - Get\_Stoichiometric, 106
  - HaveEquil, 107
  - haveEquilibrium, 106
  - HaveG, 106
  - HaveHS, 106
  - Initialize\_List, 105
  - List, 106
  - Reaction, 105
  - Set\_Energy, 106
  - Set\_Enthalpy, 106
  - Set\_EnthalpyANDEntropy, 106
  - Set\_Entropy, 106
  - Set\_Equilibrium, 106
  - Set\_Stoichiometric, 106
  - Stoichiometric, 106
- ReactionList
  - SHARK\_DATA, 114
- reactions
  - Mechanism, 76
- read\_error
  - error.h, 152
- read\_equilrxn
  - shark.h, 200
- read\_massbalance
  - shark.h, 199
- read\_options
  - shark.h, 199
- read\_scenario
  - shark.h, 199
- read\_species
  - shark.h, 199
- read\_unsteadyrxn



- shark.h, [200](#)
- readInputFile
  - yaml\_cpp\_class, [137](#)
- recalculateMolarWeight
  - Molecule, [82](#)
- Recover
  - SYSTEM\_DATA, [125](#)
- ref\_diffusion
  - SCOPSOWL\_PARAM\_DATA, [112](#)
  - SKUA\_PARAM, [119](#)
- ref\_pressure
  - SCOPSOWL\_PARAM\_DATA, [112](#)
  - SKUA\_PARAM, [120](#)
- ref\_temperature
  - SCOPSOWL\_PARAM\_DATA, [112](#)
  - SKUA\_PARAM, [119](#)
- Register
  - Atom, [11](#)
  - Molecule, [82](#)
- registered
  - Molecule, [85](#)
- rel\_tol\_norm
  - SCOPSOWL\_OPT\_DATA, [111](#)
  - SKUA\_OPT\_DATA, [119](#)
- relres
  - BiCGSTAB\_DATA, [17](#)
  - CGS\_DATA, [21](#)
  - GCR\_DATA, [44](#)
  - GMRESLP\_DATA, [47](#)
  - GMRESRP\_DATA, [52](#)
  - PCG\_DATA, [93](#)
  - PICARD\_DATA, [97](#)
- relres\_base
  - BiCGSTAB\_DATA, [18](#)
  - CGS\_DATA, [21](#)
  - GCR\_DATA, [44](#)
  - GMRESLP\_DATA, [47](#)
  - GMRESRP\_DATA, [52](#)
  - PCG\_DATA, [93](#)
  - PICARD\_DATA, [97](#)
- Removal\_Efficiency
  - Trajectory.h, [205](#)
- removeAllAtoms
  - Molecule, [83](#)
- removeElectron
  - Atom, [12](#)
- removeNeutron
  - Atom, [12](#)
- removeOneAtom
  - Molecule, [83](#)
- removeProton
  - Atom, [12](#)
- res
  - BiCGSTAB\_DATA, [17](#)
  - CGS\_DATA, [21](#)
  - FINCH\_DATA, [40](#)
  - GCR\_DATA, [44](#)
  - GMRESLP\_DATA, [47](#)
  - GMRESRP\_DATA, [52](#)
  - PCG\_DATA, [93](#)
  - PICARD\_DATA, [97](#)
- res\_data
  - PJFNK\_DATA, [102](#)
- resetKeys
  - Document, [24](#)
  - Header, [60](#)
  - YamlWrapper, [138](#)
- resetime
  - FINCH\_DATA, [41](#)
- Residual
  - SHARK\_DATA, [115](#)
- residual\_alkalinity
  - MasterSpeciesList, [67](#)
- residual\_data
  - SHARK\_DATA, [115](#)
- restart
  - GCR\_DATA, [43](#)
  - GMRESLP\_DATA, [46](#)
  - GMRESRP\_DATA, [52](#)
- revalidateAllKeys
  - Document, [24](#)
  - YamlWrapper, [138](#)
- reverse\_rate
  - UnsteadyReaction, [134](#)
- reverse\_ref\_rate
  - UnsteadyReaction, [134](#)
- Reynolds
  - MIXED\_GAS, [78](#)
- rho
  - BACKTRACK\_DATA, [15](#)
  - BiCGSTAB\_DATA, [17](#)
  - CGS\_DATA, [20](#)
- rho\_f
  - TRAJECTORY\_DATA, [127](#)
- rho\_old
  - BiCGSTAB\_DATA, [17](#)
- rho\_p
  - TRAJECTORY\_DATA, [127](#)
- Rn
  - FINCH\_DATA, [40](#)
- Rnp1
  - FINCH\_DATA, [40](#)
- Ro
  - FINCH\_DATA, [35](#)
- Rough
  - SCOPSOWL\_OPT\_DATA, [110](#)
  - SKUA\_OPT\_DATA, [118](#)
- roundIt
  - gsta\_opt.h, [162](#)
- rowExtend
  - Matrix, [75](#)
- rowExtract
  - Matrix, [75](#)
- rowReplace
  - Matrix, [75](#)
- rowShrink

- Matrix, [75](#)
- rows
  - Matrix, [71](#)
- Rs
  - TRAJECTORY\_DATA, [127](#)
- Rstd
  - egret.h, [148](#)
  - shark.h, [199](#)
- Run\_Trajectory
  - Trajectory.h, [205](#)
- run\_exec
  - ui.h, [212](#)
- run\_executable
  - ui.h, [213](#)
- run\_test
  - ui.h, [212](#)
- rxn\_rate\_error
  - error.h, [152](#)
- s
  - BiCGSTAB\_DATA, [18](#)
  - FINCH\_DATA, [34](#)
  - mSPD\_DATA, [89](#)
- SIT
  - shark.h, [199](#)
- STRING
  - yaml\_wrapper.h, [214](#)
- s\_rand
  - TRAJECTORY\_DATA, [128](#)
- SCOPSOWL
  - scopsowl.h, [196](#)
- SCOPSOWL\_DATA, [107](#)
  - binder\_fraction, [108](#)
  - binder\_poresize, [108](#)
  - binder\_porosity, [108](#)
  - char\_macro, [108](#)
  - char\_micro, [108](#)
  - coord\_macro, [108](#)
  - coord\_micro, [108](#)
  - crystal\_radius, [108](#)
  - DirichletBC, [108](#)
  - eval\_ads, [109](#)
  - eval\_diff, [109](#)
  - eval\_kf, [109](#)
  - eval\_retard, [109](#)
  - eval\_surfDiff, [109](#)
  - finch\_dat, [109](#)
  - gas\_dat, [109](#)
  - gas\_temperature, [108](#)
  - gas\_velocity, [108](#)
  - Heterogeneous, [108](#)
  - level, [108](#)
  - magpie\_dat, [109](#)
  - NonLinear, [108](#)
  - OutputFile, [109](#)
  - param\_dat, [109](#)
  - pellet\_density, [108](#)
  - pellet\_radius, [108](#)
  - Print2Console, [108](#)
  - Print2File, [108](#)
  - sim\_time, [108](#)
  - skua\_dat, [109](#)
  - SurfDiff, [108](#)
  - t, [108](#)
  - t\_counter, [108](#)
  - t\_old, [108](#)
  - t\_print, [108](#)
  - tempy, [108](#)
  - total\_pressure, [108](#)
  - total\_steps, [108](#)
  - user\_data, [109](#)
  - y, [108](#)
- SCOPSOWL\_Executioner
  - scopsowl.h, [196](#)
- SCOPSOWL\_HPP\_
  - scopsowl.h, [195](#)
- SCOPSOWL\_OPT\_DATA, [109](#)
  - adsorb\_index, [110](#)
  - CompareFile, [111](#)
  - current\_equil, [110](#)
  - current\_points, [110](#)
  - current\_press, [110](#)
  - current\_temp, [110](#)
  - diffusion\_type, [110](#)
  - e\_norm, [110](#)
  - evaluation, [110](#)
  - f\_bias, [110](#)
  - max\_bias, [110](#)
  - min\_bias, [110](#)
  - num\_curves, [110](#)
  - num\_params, [110](#)
  - Optimize, [110](#)
  - owl\_dat, [111](#)
  - param\_guess, [110](#)
  - ParamFile, [111](#)
  - q\_data, [111](#)
  - q\_sim, [111](#)
  - Rough, [110](#)
  - simulation\_equil, [110](#)
  - t, [111](#)
  - total\_eval, [110](#)
  - y\_base, [111](#)
- SCOPSOWL\_OPT\_set\_y
  - scopsowl\_opt.h, [197](#)
- SCOPSOWL\_OPTIMIZE
  - scopsowl\_opt.h, [197](#)
- SCOPSOWL\_PARAM\_DATA, [111](#)
  - Adsorbable, [112](#)
  - affinity, [112](#)
  - qAvg, [112](#)
  - qo, [112](#)
  - Qst, [112](#)
  - QstAvg, [112](#)
  - Qsto, [112](#)
  - speciesName, [112](#)
- SCOPSOWL\_SCENARIOS
  - scopsowl.h, [196](#)

SCOPSOWL\_TESTS  
  scopsowl.h, 196  
SCOPSOWL\_postprocesses  
  scopsowl.h, 196  
SCOPSOWL\_preprocesses  
  scopsowl.h, 196  
SCOPSOWL\_reset  
  scopsowl.h, 196  
SHARK  
  shark.h, 200  
SHARK\_DATA, 112  
  act\_fun, 114  
  activity\_data, 115  
  activity\_new, 115  
  activity\_old, 115  
  Conc\_new, 115  
  Conc\_old, 115  
  Console\_Output, 115  
  const\_pH, 115  
  Contains\_pH, 115  
  Contains\_pOH, 115  
  Converged, 115  
  dielectric\_const, 114  
  dt, 114  
  dt\_min, 114  
  EvalActivity, 115  
  File\_Output, 115  
  lin\_precon, 115  
  MassBalanceList, 114  
  MasterList, 114  
  Newton\_data, 115  
  Norm, 114  
  num\_mbe, 114  
  num\_other, 114  
  num\_ssr, 114  
  num\_usr, 114  
  numvar, 114  
  other\_data, 115  
  OtherList, 114  
  OutputFile, 115  
  pH, 114  
  pH\_index, 114  
  pOH\_index, 114  
  precon\_data, 115  
  ReactionList, 114  
  Residual, 115  
  residual\_data, 115  
  shark.h, 199  
  simulationtime, 114  
  SpeciationCurve, 115  
  steadystate, 114  
  t\_count, 114  
  t\_out, 114  
  temperature, 114  
  time, 114  
  time\_old, 114  
  TimeAdaptivity, 114  
  timesteps, 114  
  totalsteps, 114  
  UnsteadyList, 114  
  X\_new, 115  
  X\_old, 115  
  yaml\_object, 115  
SHARK\_SCENARIO  
  shark.h, 200  
SHARK\_TESTS  
  shark.h, 200  
SKUA  
  skua.h, 202  
SKUA\_CYCLE\_TEST01  
  skua.h, 202  
SKUA\_CYCLE\_TEST02  
  skua.h, 202  
SKUA\_DATA, 115  
  char\_measure, 117  
  coord, 116  
  DirichletBC, 117  
  eval\_diff, 117  
  eval\_kf, 117  
  finch\_dat, 117  
  gas\_dat, 117  
  gas\_velocity, 116  
  maggie\_dat, 117  
  NonLinear, 117  
  OutputFile, 117  
  param\_dat, 117  
  pellet\_radius, 116  
  Print2Console, 116  
  Print2File, 116  
  qTn, 116  
  qTnp1, 116  
  sim\_time, 116  
  t, 116  
  t\_counter, 116  
  t\_old, 116  
  t\_print, 116  
  total\_steps, 116  
  user\_data, 117  
  y, 117  
SKUA\_Executioner  
  skua.h, 202  
SKUA\_HPP\_  
  skua.h, 201  
SKUA\_LOW\_TEST03  
  skua.h, 202  
SKUA\_MID\_TEST04  
  skua.h, 202  
SKUA\_OPT\_DATA, 117  
  abs\_tol\_bias, 119  
  adsorb\_index, 118  
  CompareFile, 119  
  current\_equil, 118  
  current\_points, 118  
  current\_press, 118  
  current\_temp, 118  
  diffusion\_type, 118

- e\_norm, 118
- e\_norm\_old, 118
- evaluation, 118
- f\_bias, 118
- f\_bias\_old, 118
- max\_bias, 118
- max\_guess\_iter, 118
- min\_bias, 118
- num\_curves, 118
- num\_params, 118
- Optimize, 118
- param\_guess, 118
- param\_guess\_old, 119
- ParamFile, 119
- q\_data, 119
- q\_sim, 119
- rel\_tol\_norm, 119
- Rough, 118
- simulation\_equil, 118
- skua\_dat, 119
- t, 119
- total\_eval, 118
- y\_base, 119
- SKUA\_OPT\_set\_y
  - skua\_opt.h, 203
- SKUA\_OPTIMIZE
  - skua\_opt.h, 203
- SKUA\_PARAM, 119
  - activation\_energy, 119
  - Adsorbable, 120
  - affinity, 120
  - film\_transfer, 120
  - Qstn, 120
  - Qstnp1, 120
  - ref\_diffusion, 119
  - ref\_pressure, 120
  - ref\_temperature, 119
  - speciesName, 120
  - xC, 120
  - xn, 120
  - xnp1, 120
  - y\_eff, 120
- SKUA\_SCENARIOS
  - skua.h, 202
- SKUA\_TESTS
  - skua.h, 202
- SKUA\_postprocesses
  - skua.h, 202
- SKUA\_preprocesses
  - skua.h, 202
- SKUA\_reset
  - skua.h, 202
- SMALL\_CYCLE\_TEST02
  - scopsowl.h, 196
- SYSTEM\_DATA, 123
  - As, 124
  - avg\_norm, 125
  - Carrier, 125
  - I, 125
  - Ideal, 125
  - J, 125
  - K, 125
  - max\_norm, 125
  - N, 124
  - Output, 125
  - PI, 124
  - PT, 124
  - Par, 125
  - pi, 124
  - qT, 124
  - Recover, 125
  - Sys, 125
  - T, 124
  - total\_eval, 125
- sandbox
  - ui.h, 208
- sandbox.h, 193
  - RUN\_SANDBOX, 194
  - Speciation\_Test01\_Function, 194
  - Speciation\_Test01\_Guess, 194
  - Speciation\_Test01\_Jacobian, 194
  - Speciation\_Test01\_MatVec, 194
- ScNum
  - egret.h, 149
- scenario\_fail
  - error.h, 151
- Schmidt
  - PURE\_GAS, 104
- school.h, 194
- scops\_opt
  - ui.h, 208
- scopsowl
  - ui.h, 208
- scopsowl.h, 194
  - avgDp, 195
  - CURVE\_TEST03, 196
  - CURVE\_TEST04, 196
  - CURVE\_TEST05, 196
  - const\_filmMassTransfer, 196
  - const\_pore\_diffusion, 196
  - default\_adsorption, 196
  - default\_effective\_diffusion, 196
  - default\_filmMassTransfer, 196
  - default\_pore\_diffusion, 196
  - default\_retardation, 196
  - default\_surf\_diffusion, 196
  - Dk, 195
  - Dp, 195
  - LARGE\_CYCLE\_TEST01, 196
  - print2file\_SCOPSOWL\_header, 195
  - print2file\_SCOPSOWL\_result\_new, 196
  - print2file\_SCOPSOWL\_result\_old, 195
  - print2file\_SCOPSOWL\_time\_header, 195
  - print2file\_species\_header, 195
  - SCOPSOWL, 196
  - SCOPSOWL\_Executioner, 196

- SCOPSOWL\_HPP\_, 195
- SCOPSOWL\_SCENARIOS, 196
- SCOPSOWL\_TESTS, 196
- SCOPSOWL\_postprocesses, 196
- SCOPSOWL\_preprocesses, 196
- SCOPSOWL\_reset, 196
- SMALL\_CYCLE\_TEST02, 196
- set\_SCOPSOWL\_ICs, 196
- set\_SCOPSOWL\_params, 196
- set\_SCOPSOWL\_timestep, 196
- setup\_SCOPSOWL\_DATA, 196
- scopsowl\_opt.h, 197
  - eval\_SCOPSOWL\_Uptake, 197
  - initial\_guess\_SCOPSOWL, 197
  - SCOPSOWL\_OPT\_set\_y, 197
  - SCOPSOWL\_OPTIMIZE, 197
- Set\_ActivationEnergy
  - UnsteadyReaction, 133
- Set\_Affinity
  - UnsteadyReaction, 133
- set\_DOGFISH\_ICs
  - dogfish.h, 143
- set\_DOGFISH\_params
  - dogfish.h, 144
- set\_DOGFISH\_timestep
  - dogfish.h, 144
- Set\_Delta
  - MassBalance, 65
- Set\_Energy
  - Reaction, 106
  - UnsteadyReaction, 132
- Set\_Enthalpy
  - Reaction, 106
  - UnsteadyReaction, 132
- Set\_EnthalpyANDEntropy
  - Reaction, 106
  - UnsteadyReaction, 132
- Set\_Entropy
  - Reaction, 106
  - UnsteadyReaction, 132
- Set\_Equilibrium
  - Reaction, 106
  - UnsteadyReaction, 132
- Set\_Forward
  - UnsteadyReaction, 133
- Set\_ForwardRef
  - UnsteadyReaction, 133
- Set\_InitialValue
  - UnsteadyReaction, 133
- Set\_MaximumValue
  - UnsteadyReaction, 133
- Set\_Name
  - MassBalance, 65
- Set\_Reverse
  - UnsteadyReaction, 133
- Set\_ReverseRef
  - UnsteadyReaction, 133
- set\_SCOPSOWL\_ICs
  - scopsowl.h, 196
- set\_SCOPSOWL\_params
  - scopsowl.h, 196
- set\_SCOPSOWL\_timestep
  - scopsowl.h, 196
- set\_SKUA\_ICs
  - skua.h, 202
- set\_SKUA\_params
  - skua.h, 202
- set\_SKUA\_timestep
  - skua.h, 202
- Set\_Species\_Index
  - UnsteadyReaction, 132
- Set\_Stoichiometric
  - Reaction, 106
  - UnsteadyReaction, 132
- Set\_TimeStep
  - UnsteadyReaction, 133
- Set\_TotalConcentration
  - MassBalance, 65
- set\_alkalinity
  - MasterSpeciesList, 66
- set\_list\_size
  - MasterSpeciesList, 66
- set\_size
  - Matrix, 70
- set\_species
  - MasterSpeciesList, 66
- set\_variables
  - egret.h, 149
- setAlias
  - Document, 24
  - Header, 60
  - SubHeader, 122
- setFormula
  - Molecule, 82
- setInputFile
  - yaml\_cpp\_class, 137
- setMolarWeighth
  - Molecule, 82
- setName
  - Document, 24
  - Header, 60
  - SubHeader, 122
- setNameAliasPair
  - Document, 24
  - Header, 60
  - SubHeader, 122
- setState
  - Document, 24
  - Header, 60
  - SubHeader, 122
- setbcs
  - FINCH\_DATA, 41
- setic
  - FINCH\_DATA, 41
- setparams
  - FINCH\_DATA, 41

- setpostprocess
  - FINCH\_DATA, 41
- setpreprocess
  - FINCH\_DATA, 41
- settime
  - FINCH\_DATA, 41
- setup\_DOGFISH\_DATA
  - dogfish.h, 143
- setup\_FINCH\_DATA
  - finch.h, 156
- setup\_MONKFISH\_DATA
  - monkfish.h, 193
- setup\_SCOPSOWL\_DATA
  - scopsowl.h, 196
- setup\_SHARK\_DATA
  - shark.h, 200
- setup\_SKUA\_DATA
  - skua.h, 202
- shapeFactor
  - magpie.h, 183
- shark
  - ui.h, 208
- shark.h
  - DAVIES, 199
  - DAVIES\_LADSHAW, 199
  - DEBYE\_HUCKEL, 199
  - IDEAL, 199
  - PITZER, 199
  - SIT, 199
- shark.h, 197
  - act\_choice, 199
  - Convert2Concentration, 199
  - Convert2LogConcentration, 199
  - Davies\_equation, 199
  - DaviesLadshaw\_equation, 199
  - DebyeHuckel\_equation, 199
  - ideal\_solution, 199
  - linearsolve\_choice, 199
  - linesearch\_choice, 199
  - print2file\_shark\_header, 199
  - print2file\_shark\_info, 199
  - print2file\_shark\_results\_new, 199
  - print2file\_shark\_results\_old, 199
  - read\_equilrxn, 200
  - read\_massbalance, 199
  - read\_options, 199
  - read\_scenario, 199
  - read\_species, 199
  - read\_unsteadyrxn, 200
  - Rstd, 199
  - SHARK, 200
  - SHARK\_DATA, 199
  - SHARK\_SCENARIO, 200
  - SHARK\_TESTS, 200
  - setup\_SHARK\_DATA, 200
  - shark\_add\_customResidual, 200
  - shark\_energy\_calculations, 200
  - shark\_executioner, 200
  - shark\_guess, 200
  - shark\_initial\_conditions, 200
  - shark\_pH\_finder, 200
  - shark\_parameter\_check, 200
  - shark\_postprocesses, 200
  - shark\_preprocesses, 200
  - shark\_reset, 200
  - shark\_residual, 200
  - shark\_solver, 200
  - shark\_temperature\_calculations, 200
  - shark\_timestep\_adapt, 200
  - shark\_timestep\_const, 200
  - valid\_act, 199
- shark\_add\_customResidual
  - shark.h, 200
- shark\_energy\_calculations
  - shark.h, 200
- shark\_executioner
  - shark.h, 200
- shark\_guess
  - shark.h, 200
- shark\_initial\_conditions
  - shark.h, 200
- shark\_pH\_finder
  - shark.h, 200
- shark\_parameter\_check
  - shark.h, 200
- shark\_postprocesses
  - shark.h, 200
- shark\_preprocesses
  - shark.h, 200
- shark\_reset
  - shark.h, 200
- shark\_residual
  - shark.h, 200
- shark\_solver
  - shark.h, 200
- shark\_temperature\_calculations
  - shark.h, 200
- shark\_timestep\_adapt
  - shark.h, 200
- shark\_timestep\_const
  - shark.h, 200
- sigma
  - CGS\_DATA, 20
- sigma\_m
  - TRAJECTORY\_DATA, 128
- sigma\_n
  - TRAJECTORY\_DATA, 128
- sigma\_v
  - TRAJECTORY\_DATA, 128
- sigma\_vz
  - TRAJECTORY\_DATA, 128
- sigma\_z
  - TRAJECTORY\_DATA, 128
- sim\_time
  - SCOPSOWL\_DATA, 108
  - SKUA\_DATA, 116

- simple\_darken\_Dc
  - skua.h, [202](#)
- simulation\_fail
  - error.h, [151](#)
- simulation\_equil
  - SCOPSOWL\_OPT\_DATA, [110](#)
  - SKUA\_OPT\_DATA, [118](#)
- simulationtime
  - SHARK\_DATA, [114](#)
- single\_fiber\_density
  - MONKFISH\_DATA, [87](#)
- singular\_matrix
  - error.h, [151](#)
- size
  - Document, [24](#)
  - Header, [60](#)
  - KeyValueMap, [63](#)
  - MasterSpeciesList, [67](#)
  - YamlWrapper, [139](#)
- skua
  - ui.h, [208](#)
- skua.h, [200](#)
  - const\_Dc, [202](#)
  - const\_kf, [202](#)
  - D\_c, [201](#)
  - D\_inf, [201](#)
  - D\_o, [201](#)
  - default\_Dc, [202](#)
  - default\_kf, [202](#)
  - empirical\_kf, [202](#)
  - molefractionCheck, [202](#)
  - print2file\_SKUA\_header, [202](#)
  - print2file\_SKUA\_results\_new, [202](#)
  - print2file\_SKUA\_results\_old, [202](#)
  - print2file\_SKUA\_time\_header, [202](#)
  - print2file\_species\_header, [202](#)
  - SKUA, [202](#)
  - SKUA\_CYCLE\_TEST01, [202](#)
  - SKUA\_CYCLE\_TEST02, [202](#)
  - SKUA\_Executioner, [202](#)
  - SKUA\_HPP, [201](#)
  - SKUA\_LOW\_TEST03, [202](#)
  - SKUA\_MID\_TEST04, [202](#)
  - SKUA\_SCENARIOS, [202](#)
  - SKUA\_TESTS, [202](#)
  - SKUA\_postprocesses, [202](#)
  - SKUA\_preprocesses, [202](#)
  - SKUA\_reset, [202](#)
  - set\_SKUA\_ICs, [202](#)
  - set\_SKUA\_params, [202](#)
  - set\_SKUA\_timestep, [202](#)
  - setup\_SKUA\_DATA, [202](#)
  - simple\_darken\_Dc, [202](#)
  - theoretical\_darken\_Dc, [202](#)
- skua\_opt
  - ui.h, [208](#)
- skua\_dat
  - SCOPSOWL\_DATA, [109](#)
  - SKUA\_OPT\_DATA, [119](#)
- skua\_opt.h, [203](#)
  - eval\_SKUA\_Uptake, [203](#)
  - initial\_guess\_SKUA, [203](#)
  - SKUA\_OPT\_set\_y, [203](#)
  - SKUA\_OPTIMIZE, [203](#)
- Sn
  - FINCH\_DATA, [40](#)
- Snp1
  - FINCH\_DATA, [40](#)
- SolnTransform
  - Matrix, [73](#)
- solve
  - FINCH\_DATA, [41](#)
- sorbed\_molefraction
  - DOGFISH\_PARAM, [29](#)
  - MONKFISH\_PARAM, [88](#)
- sorption\_bc
  - MONKFISH\_PARAM, [88](#)
- Speciation\_Test01\_Data, [120](#)
  - C, [121](#)
  - CT, [121](#)
  - Jacobian, [121](#)
  - logC, [121](#)
  - logKa1, [120](#)
  - logKa2, [120](#)
  - logKw, [120](#)
  - N, [120](#)
  - NaT, [121](#)
  - NumJac, [121](#)
  - x, [121](#)
- Speciation\_Test01\_Function
  - sandbox.h, [194](#)
- Speciation\_Test01\_Guess
  - sandbox.h, [194](#)
- Speciation\_Test01\_Jacobian
  - sandbox.h, [194](#)
- Speciation\_Test01\_MatVec
  - sandbox.h, [194](#)
- SpeciationCurve
  - SHARK\_DATA, [115](#)
- species
  - DOGFISH\_PARAM, [29](#)
  - MasterSpeciesList, [67](#)
  - MONKFISH\_PARAM, [89](#)
- species\_dat
  - MIXED\_GAS, [78](#)
- species\_index
  - Mechanism, [76](#)
  - UnsteadyReaction, [134](#)
- speciesName
  - MasterSpeciesList, [67](#)
  - SCOPSOWL\_PARAM\_DATA, [112](#)
  - SKUA\_PARAM, [120](#)
- specific\_heat
  - PURE\_GAS, [104](#)
- Spherical
  - finch.h, [155](#)

- sphericalAvg
  - Matrix, [73](#)
- sphericalBCFill
  - Matrix, [72](#)
- state
  - SubHeader, [123](#)
- SteadyState
  - FINCH\_DATA, [36](#)
- steadystate
  - SHARK\_DATA, [114](#)
- steps
  - GMRESLP\_DATA, [46](#)
- Stoichiometric
  - Reaction, [106](#)
- string\_parse\_error
  - error.h, [152](#)
- Sub\_Map
  - Header, [61](#)
- SubHeader, [121](#)
  - ~SubHeader, [122](#)
  - addPair, [122](#)
  - alias, [123](#)
  - clear, [122](#)
  - Data\_Map, [123](#)
  - DisplayContents, [122](#)
  - getAlias, [123](#)
  - getMap, [122](#)
  - getName, [122](#)
  - getState, [123](#)
  - isAlias, [123](#)
  - isAnchor, [123](#)
  - name, [123](#)
  - operator=, [122](#)
  - setAlias, [122](#)
  - setName, [122](#)
  - setNameAliasPair, [122](#)
  - setState, [122](#)
  - state, [123](#)
  - SubHeader, [122](#)
  - SubHeader, [122](#)
- sum
  - ARNOLDI\_DATA, [9](#)
  - GMRESRP\_DATA, [53](#)
  - Matrix, [71](#)
- Sum\_Delta
  - MassBalance, [65](#)
- SurfDiff
  - SCOPSOWL\_DATA, [108](#)
- surface\_concentration
  - DOGFISH\_PARAM, [29](#)
- Sutherland\_Const
  - PURE\_GAS, [104](#)
- Sutherland\_Temp
  - PURE\_GAS, [104](#)
- Sutherland\_Viscosity
  - PURE\_GAS, [104](#)
- Symbol
  - Atom, [13](#)
- Sys
  - SYSTEM\_DATA, [125](#)
- sys\_dat
  - MAGPIE\_DATA, [64](#)
- T
  - FINCH\_DATA, [34](#)
  - SYSTEM\_DATA, [124](#)
- t
  - BiCGSTAB\_DATA, [19](#)
  - FINCH\_DATA, [34](#)
  - SCOPSOWL\_DATA, [108](#)
  - SCOPSOWL\_OPT\_DATA, [111](#)
  - SKUA\_DATA, [116](#)
  - SKUA\_OPT\_DATA, [119](#)
- TEST
  - ui.h, [207](#)
- t\_count
  - SHARK\_DATA, [114](#)
- t\_counter
  - DOGFISH\_DATA, [27](#)
  - MONKFISH\_DATA, [87](#)
  - SCOPSOWL\_DATA, [108](#)
  - SKUA\_DATA, [116](#)
- t\_old
  - FINCH\_DATA, [34](#)
  - SCOPSOWL\_DATA, [108](#)
  - SKUA\_DATA, [116](#)
- t\_out
  - SHARK\_DATA, [114](#)
- t\_print
  - DOGFISH\_DATA, [27](#)
  - MONKFISH\_DATA, [87](#)
  - SCOPSOWL\_DATA, [108](#)
  - SKUA\_DATA, [116](#)
- t\_rand
  - TRAJECTORY\_DATA, [128](#)
- TANGENTIAL\_FORCE
  - Trajectory.h, [204](#)
- TRAJECTORY\_DATA, [126](#)
  - a, [127](#)
  - A\_separator, [127](#)
  - A\_wire, [127](#)
  - b, [127](#)
  - B0, [127](#)
  - beta, [127](#)
  - Cap, [128](#)
  - chi\_p, [127](#)
  - dX, [128](#)
  - dY, [128](#)
  - dt, [127](#)
  - eta, [127](#)
  - H, [128](#)
  - H0, [127](#)
  - Hamaker, [127](#)
  - k, [127](#)
  - L, [127](#)
  - L\_wire, [127](#)
  - M, [127](#)



- m\_rand, 128
- mp, 127
- Ms, 127
- mu\_0, 127
- n\_rand, 128
- POL, 128
- porosity, 127
- q\_bar, 128
- Q\_in, 127
- rho\_f, 127
- rho\_p, 127
- Rs, 127
- s\_rand, 128
- sigma\_m, 128
- sigma\_n, 128
- sigma\_v, 128
- sigma\_vz, 128
- sigma\_z, 128
- t\_rand, 128
- Temp, 127
- V0, 127
- V\_separator, 127
- V\_wire, 127
- X, 128
- Y, 128
- Y\_initial, 127
- Table
  - PeriodicTable, 95
- Temp
  - TRAJECTORY\_DATA, 127
- temperature
  - SHARK\_DATA, 114
- temperature\_affinity
  - UnsteadyReaction, 134
- tempy
  - SCOPSOWL\_DATA, 108
- tensor\_out\_of\_bounds
  - error.h, 152
- term\_precon
  - GMRESR\_DATA, 50
- terminal\_precon
  - GMRESR\_DATA, 50
- test
  - ui.h, 209
- test\_loop
  - ui.h, 212
- theoretical\_darken\_Dc
  - skua.h, 202
- time
  - DOGFISH\_DATA, 26
  - MONKFISH\_DATA, 87
  - SHARK\_DATA, 114
- time\_old
  - DOGFISH\_DATA, 26
  - MONKFISH\_DATA, 87
  - SHARK\_DATA, 114
- time\_step
  - UnsteadyReaction, 134
- TimeAdaptivity
  - SHARK\_DATA, 114
- timesteps
  - SHARK\_DATA, 114
- token\_parser
  - yaml\_cpp\_class, 137
- tol\_abs
  - BiCGSTAB\_DATA, 17
  - CGS\_DATA, 21
  - FINCH\_DATA, 37
  - GCR\_DATA, 44
  - GMRESLP\_DATA, 46
  - GMRESRP\_DATA, 52
  - PCG\_DATA, 93
  - PICARD\_DATA, 97
- tol\_rel
  - BiCGSTAB\_DATA, 17
  - CGS\_DATA, 21
  - FINCH\_DATA, 37
  - GCR\_DATA, 44
  - GMRESLP\_DATA, 46
  - GMRESRP\_DATA, 52
  - PCG\_DATA, 93
  - PICARD\_DATA, 97
- total\_density
  - MIXED\_GAS, 78
- total\_dyn\_vis
  - MIXED\_GAS, 78
- total\_eval
  - GSTA\_OPT\_DATA, 57
  - SCOPSOWL\_OPT\_DATA, 110
  - SKUA\_OPT\_DATA, 118
  - SYSTEM\_DATA, 125
- total\_iter
  - FINCH\_DATA, 37
  - GCR\_DATA, 43
  - GMRESR\_DATA, 49
- total\_molecular\_weight
  - MIXED\_GAS, 78
- total\_pressure
  - MIXED\_GAS, 77
  - SCOPSOWL\_DATA, 108
- total\_sorption
  - DOGFISH\_DATA, 27
  - MONKFISH\_DATA, 87
- total\_sorption\_old
  - DOGFISH\_DATA, 27
  - MONKFISH\_DATA, 87
- total\_specific\_heat
  - MIXED\_GAS, 78
- total\_steps
  - DOGFISH\_DATA, 26
  - MONKFISH\_DATA, 87
  - SCOPSOWL\_DATA, 108
  - SKUA\_DATA, 116
- TotalConcentration
  - MassBalance, 65
- totalsteps

- SHARK\_DATA, 114
- trajectory
  - ui.h, 208
- Trajectory.h, 203
  - Brown\_RAD, 204
  - Brown\_THETA, 204
  - CARTESIAN, 204
  - DISPLACEMENT, 205
  - Grav\_R, 204
  - Grav\_T, 204
  - LOCATION, 205
  - Magnetic\_R, 204
  - Magnetic\_T, 204
  - Number\_Generator, 205
  - POLAR, 204
  - RADIAL\_FORCE, 204
  - Removal\_Efficiency, 205
  - Run\_Trajectory, 205
  - TANGENTIAL\_FORCE, 204
  - Trajectory\_SetupConstants, 205
  - V\_RAD, 204
  - V\_THETA, 204
  - Van\_R, 204
- Trajectory\_SetupConstants
  - Trajectory.h, 205
- transpose
  - Matrix, 72
- transpose\_dat
  - GCR\_DATA, 45
- transpose\_multiply
  - Matrix, 72
- tridiagonalFill
  - Matrix, 72
- tridiagonalSolve
  - Matrix, 72
- tridiagonalVectorFill
  - Matrix, 73
- twoFifths
  - gsta\_opt.h, 162
- type
  - ValueTypePair, 136
- u
  - CGS\_DATA, 21
  - GCR\_DATA, 45
- UNKNOWN
  - yaml\_wrapper.h, 214
- u\_star
  - FINCH\_DATA, 39
- u\_temp
  - GCR\_DATA, 45
- uAverage
  - finch.h, 156
- uAvg
  - FINCH\_DATA, 34
- uAvg\_old
  - FINCH\_DATA, 35
- UI\_DATA, 128
  - argc, 130
  - argv, 130
  - BasicUI, 130
  - count, 129
  - Files, 130
  - input\_files, 129
  - max, 129
  - MissingArg, 130
  - option, 130
  - Path, 130
  - path, 129
  - user\_input, 129
  - value\_type, 129
- UI\_HPP\_
  - ui.h, 207
- uIC
  - FINCH\_DATA, 35
- uT
  - FINCH\_DATA, 34
- uT\_old
  - FINCH\_DATA, 34
- uTotal
  - finch.h, 156
- ubest
  - FINCH\_DATA, 39
- ui.h
  - CONTINUE, 207
  - dogfish, 207
  - EXECUTE, 207
  - EXIT, 207
  - eel, 207
  - egret, 207
  - finch, 207
  - gsta\_opt, 208
  - HELP, 207
  - lark, 207
  - macaw, 207
  - maggie, 208
  - mola, 207
  - monkfish, 208
  - sandbox, 208
  - scops\_opt, 208
  - scopsowl, 208
  - shark, 208
  - skua, 208
  - skua\_opt, 208
  - TEST, 207
  - trajectory, 208
- ui.h, 205
  - allLower, 208
  - aii\_help, 208
  - bui\_help, 208
  - display\_help, 211
  - display\_version, 211
  - ECO\_EXECUTABLE, 207
  - ECO\_VERSION, 207
  - exec, 209
  - exec\_loop, 212
  - exit, 208

- help, 208
- input, 209
- invalid\_input, 211
- number\_files, 210
- path, 209
- run\_exec, 212
- run\_executable, 213
- run\_test, 212
- test, 209
- test\_loop, 212
- UI\_HPP\_, 207
- valid\_addon\_options, 210
- valid\_exec\_string, 210
- valid\_input\_execute, 212
- valid\_input\_main, 211
- valid\_input\_tests, 211
- valid\_options, 207
- valid\_test\_string, 210
- version, 209
- un
  - FINCH\_DATA, 39
- unm1
  - FINCH\_DATA, 39
- unp1
  - FINCH\_DATA, 39
- unregistered\_name
  - error.h, 152
- unstable\_matrix
  - error.h, 151
- UnsteadyList
  - SHARK\_DATA, 114
- UnsteadyPrecipitation, 130
- UnsteadyReaction, 131
  - ~UnsteadyReaction, 132
  - activation\_energy, 134
  - calculateEnergies, 133
  - calculateEquilibrium, 133
  - calculateRate, 133
  - checkSpeciesEnergies, 133
  - Display\_Info, 132
  - Eval\_IC\_Residual, 134
  - Eval\_ReactionRate, 134
  - Eval\_Residual, 134
  - Explicit\_Eval, 134
  - forward\_rate, 134
  - forward\_ref\_rate, 134
  - Get\_ActivationEnergy, 133
  - Get\_Affinity, 134
  - Get\_Energy, 133
  - Get\_Enthalpy, 133
  - Get\_Entropy, 133
  - Get\_Equilibrium, 133
  - Get\_Forward, 133
  - Get\_ForwardRef, 133
  - Get\_InitialValue, 133
  - Get\_MaximumValue, 133
  - Get\_Reverse, 133
  - Get\_ReverseRef, 133
  - Get\_Species\_Index, 133
  - Get\_Stoichiometric, 133
  - Get\_TimeStep, 134
  - haveEquilibrium, 133
  - HaveForRef, 134
  - HaveForward, 134
  - haveRate, 133
  - HaveRevRef, 134
  - HaveReverse, 134
  - initial\_value, 134
  - Initialize\_List, 132
  - max\_value, 134
  - reverse\_rate, 134
  - reverse\_ref\_rate, 134
  - Set\_ActivationEnergy, 133
  - Set\_Affinity, 133
  - Set\_Energy, 132
  - Set\_Enthalpy, 132
  - Set\_EnthalpyANDEntropy, 132
  - Set\_Entropy, 132
  - Set\_Equilibrium, 132
  - Set\_Forward, 133
  - Set\_ForwardRef, 133
  - Set\_InitialValue, 133
  - Set\_MaximumValue, 133
  - Set\_Reverse, 133
  - Set\_ReverseRef, 133
  - Set\_Species\_Index, 132
  - Set\_Stoichiometric, 132
  - Set\_TimeStep, 133
  - species\_index, 134
  - temperature\_affinity, 134
  - time\_step, 134
  - UnsteadyReaction, 132
  - UnsteadyReaction, 132
- uo
  - FINCH\_DATA, 35
- Update
  - FINCH\_DATA, 36
- update\_arnoldi\_solution
  - lark.h, 169
- upperHessenberg2Triangular
  - Matrix, 74
- upperHessenbergSolve
  - Matrix, 75
- upperTriangularSolve
  - Matrix, 74
- user\_data
  - DOGFISH\_DATA, 28
  - MONKFISH\_DATA, 88
  - SCOPSOWL\_DATA, 109
  - SKUA\_DATA, 117
- user\_input
  - UI\_DATA, 129
- uz\_I\_E
  - FINCH\_DATA, 39
- uz\_I\_I
  - FINCH\_DATA, 39

- uz\_lm1\_E
  - FINCH\_DATA, 39
- uz\_lm1\_l
  - FINCH\_DATA, 39
- uz\_lp1\_E
  - FINCH\_DATA, 39
- uz\_lp1\_l
  - FINCH\_DATA, 39
- V
  - magpie.h, 183
- v
  - ARNOLDI\_DATA, 8
  - BiCGSTAB\_DATA, 18
  - CGS\_DATA, 22
  - GMRESRP\_DATA, 53
  - mSPD\_DATA, 89
  - PJFNK\_DATA, 101
- V0
  - TRAJECTORY\_DATA, 127
- V\_RAD
  - Trajectory.h, 204
- V\_THETA
  - Trajectory.h, 204
- V\_separator
  - TRAJECTORY\_DATA, 127
- V\_wire
  - TRAJECTORY\_DATA, 127
- vIC
  - FINCH\_DATA, 35
- valence\_e
  - Atom, 13
- valid\_act
  - shark.h, 199
- valid\_addon\_options
  - ui.h, 210
- valid\_exec\_string
  - ui.h, 210
- valid\_input\_execute
  - ui.h, 212
- valid\_input\_main
  - ui.h, 211
- valid\_input\_tests
  - ui.h, 211
- valid\_options
  - ui.h, 207
- valid\_test\_string
  - ui.h, 210
- Value\_Type
  - ValueTypePair, 136
- value\_type
  - UI\_DATA, 129
- ValueTypePair, 134
  - ~ValueTypePair, 135
  - assertType, 135
  - DisplayPair, 135
  - editPair, 135
  - editValue, 135
  - findType, 135
  - getBool, 135
  - getDouble, 136
  - getInt, 136
  - getPair, 136
  - getString, 135
  - getType, 136
  - getValue, 136
  - operator=, 135
  - type, 136
  - Value\_Type, 136
  - ValueTypePair, 135
  - ValueTypePair, 135
- Van\_R
  - Trajectory.h, 204
- vanAlbada\_discretization
  - finch.h, 158
- vector\_out\_of\_bounds
  - error.h, 152
- velocity
  - MIXED\_GAS, 78
- version
  - ui.h, 209
- Vk
  - ARNOLDI\_DATA, 8
  - GMRESRP\_DATA, 53
- vn
  - FINCH\_DATA, 39
- vnp1
  - FINCH\_DATA, 39
- vo
  - FINCH\_DATA, 35
- w
  - ARNOLDI\_DATA, 8
  - CGS\_DATA, 22
  - GMRESRP\_DATA, 53
- weight
  - Mechanism, 76
- weightedAvg
  - gsta\_opt.h, 163
- X
  - TRAJECTORY\_DATA, 128
- x
  - BiCGSTAB\_DATA, 18
  - CGS\_DATA, 21
  - GCR\_DATA, 44
  - GMRESLP\_DATA, 47
  - GMRESRP\_DATA, 52
  - GPAST\_DATA, 54
  - PCG\_DATA, 93
  - PJFNK\_DATA, 101
  - Speciation\_Test01\_Data, 121
- x0
  - PICARD\_DATA, 97
- X\_new
  - SHARK\_DATA, 115
- X\_old
  - SHARK\_DATA, 115

- xIC
  - SCOPSOWL\_PARAM\_DATA, 112
  - SKUA\_PARAM, 120
- xk
  - BACKTRACK\_DATA, 15
- xn
  - SKUA\_PARAM, 120
- xnp1
  - SKUA\_PARAM, 120
- Y
  - TRAJECTORY\_DATA, 128
- y
  - BiCGSTAB\_DATA, 18
  - GMRESRP\_DATA, 53
  - GPAST\_DATA, 54
  - SCOPSOWL\_DATA, 108
  - SKUA\_DATA, 117
- y\_base
  - SCOPSOWL\_OPT\_DATA, 111
  - SKUA\_OPT\_DATA, 119
- y\_eff
  - SKUA\_PARAM, 120
- Y\_initial
  - TRAJECTORY\_DATA, 127
- YAML\_CPP\_TEST
  - yaml\_wrapper.h, 214
- YAML\_WRAPPER\_TESTS
  - yaml\_wrapper.h, 214
- yaml\_wrapper.h
  - ALIAS, 214
  - ANCHOR, 214
  - BOOLEAN, 214
  - DOUBLE, 214
  - INT, 214
  - NONE, 214
  - STRING, 214
  - UNKNOWN, 214
- yaml\_cpp\_class, 136
  - ~yaml\_cpp\_class, 136
  - cleanup, 137
  - current\_token, 137
  - DisplayContents, 137
  - executeYamlRead, 137
  - file\_name, 137
  - getYamlWrapper, 137
  - input\_file, 137
  - previous\_token, 137
  - readInputFile, 137
  - setInputFile, 137
  - token\_parser, 137
  - yaml\_cpp\_class, 136
  - yaml\_wrapper, 137
  - yaml\_cpp\_class, 136
- yaml\_object
  - SHARK\_DATA, 115
- yaml\_wrapper
  - yaml\_cpp\_class, 137
- yaml\_wrapper.h, 213
  - data\_type, 214
  - header\_state, 214
  - YAML\_CPP\_TEST, 214
- YamlWrapper, 137
  - ~YamlWrapper, 138
  - addDocKey, 139
  - begin, 138
  - changeKey, 138
  - clear, 138
  - copyAnchor2Alias, 139
  - DisplayContents, 139
  - Doc\_Map, 139
  - end, 138
  - getAnchoredDoc, 139
  - getDocFromHeadAlias, 139
  - getDocFromSubAlias, 139
  - getDocMap, 138
  - getDocument, 138
  - operator(), 138
  - operator=, 138
  - resetKeys, 138
  - revalidateAllKeys, 138
  - size, 139
  - YamlWrapper, 138
  - YamlWrapper, 138
- yk
  - ARNOLDI\_DATA, 8
- Z
  - magpie.h, 183
- z
  - BiCGSTAB\_DATA, 18
  - CGS\_DATA, 22
  - PCG\_DATA, 94
- z\_old
  - PCG\_DATA, 94
- zero\_vector
  - error.h, 152
- zeros
  - Matrix, 70