**Topology Optimisation code description:**

*C files and its related functions:*

|  |
| --- |
| BLES5.c |
| **int** **main**(**int** argc, **char** \*argv[]) |

|  |
| --- |
| ABFG.c |
| **int** **closeNode**(mesh \*inMesh, **double** xp, **double** yp) |
| **void** **Find\_struct**(mesh \*inMesh, levSet \*levelset, boundary \*bound\_in, **double** \*alpha, **double** aMin) |
| **void** **AFG\_area**(mesh \*inMesh, **double** \*alpha, **short** \*NodeStat, **short** \*ElemStat, Coord \*AuxNodes,**int** NumBound, Bseg \*Boundary, **double** aMin) |
| **double** InArea(**int** eStat, **int** eNum, **int** \*Lnodes, **short** \*NodeStat, **int** NumNodes, Coord \*NodeCoord,Coord \*AuxNodes, **int** NumBound, Bseg \*Boundary) |
| **double** PolyArea(**int** N,Coord \*point) |
| **short** LineCross(Coord \*pts) |
| **double** det2(**double** a, **double** b, **double** c, **double** d) |
| **void** AFG\_Matrix(**int** mass, **double** \*\*KE, **double** \*\*ME, sp\_mat \*Kg, sp\_mat \*Mg, sp\_mat \*lump\_mass, **double** \*alpha, mesh \*inMesh, isoMat \*inMat, **double** aMin, **double** mMin) |
| **void** Gauss\_Coord(mesh \*inMesh, Coord \*gCoord) |
| **void** AFG\_Sens(mesh \*inMesh, boundary \*bound\_in, **double** \*alpha, isoMat \*inMat, **double** \*Nsens, **double** \*\*prim, **double** \*\*dual, **int** numDual, **int** numCase, **double** \*wgt, Coord \*gCoord, **double** aMin, **int** mode, **double** \*fact, bool sw, Coord \*acc) |
|  |

|  |
| --- |
| EMatrix.c |
| **double** Aodd(**int** i,**int** j,**double** e,**double** g) |
| **double** Aeven(**int** i,**int** j,**double** e,**double** g) |
| **void** KEMatrix(**double** \*KE, isoMat \*mat, **double** t) |
| **void** MEMatrix(**double** \*ME, **double** rho, **double** area, **double** t) |
| **void** Assemble2(**int** \*K\_begin, **int** \*M\_begin, **int** nNod, **int** nDof, **int** \*tnodes, **double** \*KE, **double** \*ME, sp\_mat \*Kg, sp\_mat \*Mg, **int** mass) |
| **double** HS\_mat(**double** alpha, **double** hs\_int, isoMat \*mat1, isoMat \*mat2) |
| **void** self\_weight(mesh \*inMesh, isoMat \*inMat, **double** aMin, **double** mMin, **double** \*alpha, **int** freeDof, **int** \*dofMap, **int** numCase, **double** \*load\_in, **double** \*load\_out, Coord \*acc) |

|  |
| --- |
| Input.c |
| **int** icmpfunc (**const** **void** \* p1, **const** **void** \* p2) |
| **int** read\_input(**char** \*datafile, mesh \*inMesh, **int** \*numMat, isoMat \*inMat, levSet \*levelset, prob \*lsprob, ctrl \*control, **int** \*\*map, **int** \*numCase, **double** \*\*load\_in, **int** \*freeDof, sp\_mat \*lump\_mass, bool \*sw, Coord \*\*acc) |

|  |
| --- |
| Levels.c |
| **void** initialLsf(mesh \*inMesh, levSet \*levelset, **int** NumHole, CirH \*holes, **int** NumRect, Coord \*Rect, **double** lBand) |
| **void** RectHole(**int** NumNodes, Coord \*NodeCoord, **int** NumRect, Coord \*Rect, **double** \*lsf) |
| **void** NarBand(mesh \*inMesh, levSet \*levelset, **double** lBand) |
| **void** BsegWgt(boundary \*bound\_in, mesh \*inMesh) |
| **void** BoundInt(mesh \*inMesh, levSet \*levelset, boundary \*bound\_in, **int** numFunc, **double** \*\*Nsens, **int** \*Lbound\_nums, **int** \*numLbound, **double** \*Lbound) |
| **void** Vext(mesh \*inMesh, levSet \*levelset, boundary \*bound\_in, **double** \*Vnorm) |
| **void** LocalVext2(**int** NodeX, **int** NodeY, **int** \*\*Nodes, **double** \*Vnorm, **double** \*Vtemp, **double** \*lsf, **double** \*lsf\_temp, **double** tol,  bool \*known, bool \*trial, boundary \*bound\_in, **double** h, **int** NumNodes, Coord \*NodeCoord, **int** sign) |
| **double** GradWENO(**int** Xi, **int** Yj, **int** num, **double** \*lsf, mesh \*inMesh, **int** sign, **double** Vn, **double** dt) |
| **double** GWsub(**double** v1,**double** v2,**double** v3,**double** v4,**double** v5) |
| **void** ReInt(mesh \*inMesh, levSet \*levelset) |
| **void** LocalInt(**int** NodeX, **int** NodeY, **int** \*\*Nodes, **double** \*lsf, **double** \*lsf\_temp, bool \*known, bool \*trial, **double** h, **double** tol, **int** sign) |
| **void** get\_delD(**int** n, **int** m, **double** \*x, **double** \*lam, **double** \*s, **double** \*up\_lim, **double** \*low\_lim) |
| **void** get\_slpGrad(**int** n, **int** m, **int** numVar, **double** \*lam, **double** \*s, **double** \*c, **double** \*up\_lim, **double** \*low\_lim, **double** \*max\_lam, **double** \*min\_lam, **double** \*grad, **int** pinfo) |
| **int** dcmpfunc (**const** **void** \* p1, **const** **void** \* p2) |
| **void** getLamLim(**int** n, **int** m, **double** \*inmax, **double** \*inmin, **double** \*s, **double** \*c, **double** \*up\_lim, **double** \*low\_lim, **int** pinfo) |
| **int** SLPsubSol4(mesh \*inMesh, levSet \*levelset, boundary \*bound\_in, **double** alt, **double** \*delCon, **int** numCon, **double** \*\*sens, **double** \*cA, **int** n, **int** \*bound, **double** \*lam\_in, **int** \*active, **double** \*Vnorm, **double** \*pred, **int** numAdd, **double** \*add\_sens, **double** \*add\_min, **double** \*add\_max, **double** \*add\_change, **int** pinfo) |
| **int** trust\_sub(**int** n, **int** m, **int** nu, **double** \*x, **double** \*c, **double** \*A, **double** \*b, **double** \*u, **int** pinfo) |
| **int** con\_min3(**int** n, **int** m, **int** numCon, **double** \*lam, **double** \*s, **double** \*A, **double** \*b, **double** \*lam\_min, **double** \*lam\_max, **double** \*up\_lim, **double** \*low\_lim, **int** pinfo) |
| **void** get\_lam0(**int** n, **int** m, **int** numCon, **double** \*lam, **double** \*s, **double** \*cA, **double** \*b, **double** \*lam\_min, **double** \*lam\_max) |

|  |
| --- |
| lipsol\_func.c |
| **double** divZero(**double** num, **double** denom) |
| **int** initialize(**int** n, **int** m, **int** nu, **double** \*\*v, **double** \*A, **double** \*b, **double** \*u, **double** \*c) |
| **int** initialize2(**int** n, **int** m, **int** nu, **double** \*\*v, **double** \*A, **double** \*b, **double** \*u, **double** \*c) |
| **int** predictor(**int** n, **int** m, **int** nu, **double** \*\*v, **double** \*\*delP, **double** \*A, **double** \*b, **double** \*u, **double** \*c) |
| **void** corrector(**int** n, **int** nu, **double** mu, **double** \*\*v, **double** \*\*delP) |
| **double** centering(**int** n, **int** m, **int** nu, **double** lim, **double** \*\*v, **double** \*\*delP) |
| **double** dual\_gap(**int** n, **int** nu, **double** \*\*v) |
| **void** update(**int** n, **int** nu, **int** m, **double** \*\*v, **double** \*\*delP) |
| **double** stopping(**int** n, **int** m, **int** nu, **double** \*\*v, **double** \*A, **double** lenb, **double** \*b, **double** lenu, **double** \*u, **double** lenc, **double** \*c) |
| **void** report(**int** n, **int** m, **int** nu, **double** \*\*v) |

|  |
| --- |
| Numbering.c |
| **void** Numbering(mesh \*inMesh) |
| **void** Coordinates(mesh \*inMesh) |
| **void** NodeNums2(mesh \*inMesh) |
| **void** Bar\_numbering(mesh \*inMesh) |
| **void** free\_sp\_mat(sp\_mat \*m) |
| **void** set\_sp\_mat(sp\_mat \*m) |
| **void** rem\_sp\_mat(sp\_mat \*m, **int** \*map, **int** inc) |

|  |
| --- |
| Output.c |
| **void** OutNumber(mesh \*inMesh, **char** \*datafile) |
| **void** OutNodeCoord(mesh \*inMesh, **char** \*datafile) |
| **void** OutPLotShapeVTK2(mesh \*inMesh, **double** \*lsf, **double** \*alpha, **int** pinfo, **int** itt, **char** \*datafile) |
| **void** OutBoundVTK(mesh \*inMesh, boundary \*bound\_in, **int** num\_sens, **double** \*\*Sens, **int** itt, **char** \*datafile) |
| **void** OutBoundInt(**int** numFunc, **int** numLbound, **int** \*Lbound\_nums, **double** \*Lbound, **int** itt, **char** \*datafile) |
| **void** OutHJVTK(mesh \*inMesh, **double** \*Vnorm, **double** \*Grad, **int** itt, **char** \*datafile) |
| **void** OutDispVTK(mesh \*inMesh, **int** numCase, **double** \*disp, **int** num\_eig, **double** \*vec, **int** itt, **char** \*datafile) |
| **void** OutConv(**int** itt, prob \*lsprob, **double** \*Obj, **double** \*constr, **char** \*datafile) |
| **void** OutFreq(**int** itt, **int** num\_eig, **double** \*freq, **char** \*datafile) |
| **void** OutBars(mesh \*inMesh, **int** numFunc, **double** \*sens, **int** pinfo, **int** itt, **char** \*datafile) |
| **void** OutDesBC(mesh \*inMesh, **double** \*sens, **int** pinfo, **int** itt, **char** \*datafile) |
| **void** OutDesMat(mesh \*inMesh, **double** \*alpha, **double** aMin, **int** num\_sens, **double** \*sens, **int** pinfo, **int** itt, **char** \*datafile) |

|  |
| --- |
| Sens.c |
| **int** Lsens(Coord \*pt, **int** xMax, **int** xMin, **int** yMax, **int** yMin, **double** aMin, **double** \*alpha, **double** r2, Coord \*gCoord, **double** \*gSens, Elem \*\*Number, **int** wFlag, **int** numDual, **double** \*out) |
| **void** GaSens\_Q4(**int** \*tnodes, **double** \*\*prim, **double** \*\*dual, **double** alpha, **double** h, **double** t, isoMat \*inMat, **int** Gcount, **double** \*gSens, **int** numCase, **int** numDual, **double** \*wgt, bool sw, Coord \*acc) |
| **void** GaEigSens\_Q4(**int** \*tnodes, **double** \*\*prim, **double** \*\*dual, **double** alpha, **double** h, **double** t, isoMat \*inMat, **int** Gcount, **double** \*gSens, **int** num\_eig, **double** \*eig) |
| **void** barSens(mesh \*inMesh, **double** \*bar\_sens, **double** \*\*prim, **int** numCase, **double** \*wgt) |
| **void** bcSens(mesh \*inMesh, **double** \*bc\_sens, **double** \*\*prim, **int** numCase, **double** \*wgt) |
| **void** matSens\_comp(mesh \*inMesh, isoMat \*inMat, **double** \*KE, **double** \*mat\_sens, **int** numCase, **double** \*wgt, **double** \*\*prim, **double** \*\*dual, **double** \*alpha, **double** aMin, bool sw, Coord \*acc) |
| **void** matSens\_eig(mesh \*inMesh, isoMat \*inMat, **double** \*KE, **double** \*ME, **double** \*mat\_sens, **int** numEig, **double** \*eig\_vals, **double** \*\*eig\_vecs, **double** \*alpha, **double** aMin) |
| **void** HS\_Sens\_eig(mesh \*inMesh, isoMat \*inMat, **double** \*KE, **double** \*ME, **double** \*mat\_sens, **int** numEig, **double** \*eig\_vals, **double** \*\*eig\_vecs, **double** \*alpha, **double** aMin) |
| **double** dE\_dalpha(**double** alpha, **double** hs\_int, isoMat \*mat1, isoMat \*mat2) |

|  |
| --- |
| Solve.c |
| **void** FE\_Solve(sp\_mat \*Kg, **int** \*dofMap, **double** \*disp, **int** freeDof, **int** order, **int** numRhs) |
| **void** solve(**int** n, **int** ne, **int** \*irn, **int** \*jcn, **double** \*A, **int** n\_rhs, **double** \*rhs, **int** pinfo) |
| **void** d\_lsLPK(**int** N, **int** M, **int** NRHS, **double** \*A, **double** \*b) |
| **int** dun\_slvLPK(**char** trans, **int** N, **int** NRHS, **double** \*A, **double** \*B) |
| **int** dsy\_slvLPK(**int** N, **int** NRHS, **double** \*A, **double** \*B) |
| **void** din\_LPK(**double**\* A, **int** N) |
| **int** lp\_simplex(**int** n, **int** m, **int** nu, **double** \*x, **double** \*c, **double** \*A, **double** \*b, **double** \*u, **int** pinfo) |
| **int** LPsolve(**int** n, **int** m, **int** nu, **double** \*x, **double** \*c, **double** \*A, **double** \*b, **double** \*u, **int** pinfo) |
| **int** eig\_solve(**int** nev\_in, sp\_mat \*Kg, sp\_mat \*Mg, **int** n, **int** order, **int** \*dofMap, **double** \*vals, **double** \*vecs, **int** pinfo) |
| **void** Msp\_Vec(**int** n, sp\_mat \*mat, **double** \*vin, **double** \*vout) |

*C files and its related functions descriptions:*

|  |  |
| --- | --- |
| **BLES5.c** | |
| main() | Main function |

|  |  |
| --- | --- |
| **ABFG.c** | |
| closeNode() | Function to find the nearest grid node number to a set of co-ordinates. |
| Find\_struct() | Function to extract the structure from the lsf  1. determines the node and element status  2. discretizes the boundary  3. computes area ratios for AFG method |
| AFG\_area() | function to compute area ratio for all elements |
| InArea() | function to compute a cut element area |
| PolyArea() | Function that calculates the area of any Polygon |
| LineCross() | function to determine if two lines (p1 -> p2 & p3 -> p4) cross |
| det2() | compute determinant of a 2x2 matrix |
| AFG\_Matrix() | Assembles global stiffness (& maybe mass) matrix for AFG method in triplet format (for MA57 solver) |
| Gauss\_Coord() | function to compute gauss point coords |
| AFG\_Sens() | calculate sensitivies using least squares of integration points for AFG method |

|  |  |
| --- | --- |
| **EMatrix.c** | |
| Aodd() | calculate the values in each 2 x 2 block in the Element Stiffness Matrix, For even (i + j =odd) Matrix enrites |
| Aeven() | calculate the values in each 2 x 2 block in the Element Stiffness Matrix, For odd (i + j = even) Matrix enrites |
| KEMatrix() | Complete square IN element stiffness matrix computation |
| MEMatrix() | consistent mass matrix for a 2D plane element |
| Assemble2() | Function that assembles the element matrices into the global matrix (in triplet form) |
| HS\_mat() | function to compute elastic modulus from two materials |
| self\_weight() | function to compute self-weight load vector |

|  |  |
| --- | --- |
| **Input.c** | |
| icmpfunc () |  |
| read\_input() | function to read input file |

|  |  |
| --- | --- |
| **Levels.c** | |
| initialLsf() | function to create the initial signed distance function - inc holes |
| RectHole() | Function to calculate lsf around a rectangular hole |
| NarBand() | Function to calculate active and mine nodes for the narrow band method |
| BsegWgt() | function to weight boundary segments so that lengths are more even |
| BoundInt() | Function to perfrom boundary intergration of objective and constraint shape sens |
| Vext() | Function to calculate extension velocities using the fast marching method |
| LocalVext2() | function that works out Velocity for nodes close to the boundary |
| GradWENO() | Function to calcualte gradient using WENO scheme |
| GWsub() | sub-function for GradWENO |
| ReInt() | Function to re-initalise the lsf as a signed distance function - similar to Vext above |
| LocalInt() | Function to reinitalise lsf for inital set of trial nodes |
| get\_delD() | obtain a boundary move vector from input of lam, s and move limits |
| get\_slpGrad() | function to obtain gradients by finite difference |
| dcmpfunc () | sub-function for sorting in ascending order |
| getLamLim() | function to get limits for lambda |
| SLPsubSol4() | function to set velocity (or move dist) using SLP - filter method |
| trust\_sub() | sub-solve function for the trust region method |
| con\_min3() | function to minimise constraint violation using Newtons method |
| get\_lam0() | function to find an initial set of lambda values |

|  |  |
| --- | --- |
| **lipsol\_func.c** | |
| divZero() | function to handle divide by zero |
| initialize() | function to find an initial point |
| initialize2() | function to find an initial point |
| predictor() | function to compute delP(v) - predictor |
| corrector() | function to compute delC(v) - corrector |
| centering() | function to compute the centering parameter |
| dual\_gap() | function to find the duality gap |
| update() | update the variables |
| stopping() | compute the stopping criterion |
| report() | function to print solution to screen |

|  |  |
| --- | --- |
| **Numbering.c** | |
| Numbering() | Function that numbers all elements and nodes in the FG domain |
| Coordinates() | Node Co-ordinate calculation function |
| NodeNums2() | function that orders node numbers into a 2D based on their relative positions |
| Bar\_numbering() | function to number bars elements |
| free\_sp\_mat() | function to free memory for a sparse matrix |
| set\_sp\_mat() | function to create memory for a sparse matrix |
| rem\_sp\_mat() | function to remove dof from a sparse matrix |

|  |  |
| --- | --- |
| Output.c | |
| OutNumber() |  |
| OutNodeCoord() |  |
| OutPLotShapeVTK2() |  |
| OutBoundVTK() |  |
| OutBoundInt() |  |
| OutHJVTK() |  |
| OutDispVTK() |  |
| OutConv() |  |
| OutFreq() |  |
| OutBars() |  |
| OutDesBC() |  |
| OutDesMat() |  |

|  |  |
| --- | --- |
| **Sens.c** | |
| Lsens() | Function that calculates the sensitivity of a node by a least squares |
| GaSens\_Q4() | Function to calculate sensitivity values for an element at 4 gauss points |
| GaEigSens\_Q4() | Function to calculate additional sensitivity part for eigenvalues |
| barSens() | function that computes bar senstivites for compliance |
| bcSens() | function to compute designable bc sensitvities for compliance |
| matSens\_comp() | function to compute designable material design varibles for compliance |
| matSens\_eig() | function to compute designable material design varibles for eigenvalues |
| HS\_Sens\_eig() | function to compute designable material design H-S varibles for eigenvalues |
| dE\_dalpha() | derivative of E for H-S bound material model |

|  |  |
| --- | --- |
| **Solve.c** | |
| FE\_Solve() | Function that sets up and calls the MA57 to solve Ku = f |
| solve() | Function that sets up and runs the MA57 Multifrontal Solver |
| d\_lsLPK() | function to solve a linear least squares problem (by factorization) |
| dun\_slvLPK() | fucntion to solve a non-symmetric system (LU decomposition) |
| dsy\_slvLPK() | function to solve a symmetric system (Cholesky factorization) |
| din\_LPK() | function to invert a matrix |
| lp\_simplex() | function to solve a linear program using HSL revised Simplex method |
| LPsolve() | function to solve linear prog using interior point method |
| eig\_solve() | function to use ARPACK reverse communication to solve generalized eigenvalue problem (mode 3) |
| Msp\_Vec() | multiply a sparse matrix by a vector |