

FINEL

Generated by Doxygen 1.8.12

Contents

1	Modules Index	1
1.1	Modules List	1
2	Data Type Index	3
2.1	Data Types List	3
3	File Index	5
3.1	File List	5
4	Module Documentation	7
4.1	meshstructure Module Reference	7
4.1.1	Detailed Description	7
4.1.2	Function/Subroutine Documentation	7
4.1.2.1	mallocelem()	7
4.1.2.2	mallocnodes()	8
4.2	minputreader Module Reference	8
4.2.1	Detailed Description	9
4.2.2	Function/Subroutine Documentation	10
4.2.2.1	analyzefile()	10
4.2.2.2	analyzefileinput()	10
4.2.2.3	createsimpleinputfile()	10
4.2.2.4	findinclude()	12
4.2.2.5	findkeyword()	12
4.2.2.6	mergeincludecontents()	13
4.2.2.7	preparefilelines()	13

4.2.2.8	<code>readboundaryconditions()</code>	14
4.2.2.9	<code>readinputfiles()</code>	15
4.2.2.10	<code>readintarraykeywordvalue()</code>	16
4.2.2.11	<code>readintegerarrayvalues()</code>	16
4.2.2.12	<code>readintegerkeywordvalue()</code>	17
4.2.2.13	<code>readrealkeywordvalue()</code>	18
4.2.2.14	<code>readrealmatrixvalues()</code>	18
4.2.2.15	<code>readstringkeywordvalue()</code>	19
4.2.3	Variable Documentation	20
4.2.3.1	<code>file_lines</code>	20
4.2.3.2	<code>number_of_lines</code>	20
4.3	mio Module Reference	20
4.3.1	Detailed Description	21
4.3.2	Function/Subroutine Documentation	21
4.3.2.1	<code>closefiles()</code>	21
4.3.2.2	<code>openfiles()</code>	22
4.3.2.3	<code>print_files()</code>	22
4.3.2.4	<code>print_sol()</code>	23
4.3.2.5	<code>print_sol_csv()</code>	24
4.3.2.6	<code>print_sol_vtk()</code>	24
4.3.2.7	<code>read_elems()</code>	25
4.3.2.8	<code>read_nodes()</code>	26
4.3.3	Variable Documentation	27
4.3.3.1	<code>file_format</code>	27
4.3.3.2	<code>iin</code>	27
4.3.3.3	<code>infile</code>	27
4.3.3.4	<code>iout</code>	27
4.3.3.5	<code>ioute</code>	27
4.3.3.6	<code>ioutn</code>	27
4.3.3.7	<code>isol</code>	27

4.3.3.8	isolcsv	27
4.3.3.9	isolvtk	28
4.3.3.10	outelem	28
4.3.3.11	outfile	28
4.3.3.12	outnodes	28
4.3.3.13	solfile	28
4.3.3.14	title	28
4.4	mprocessor Module Reference	28
4.4.1	Detailed Description	29
4.4.2	Function/Subroutine Documentation	29
4.4.2.1	applybc()	29
4.4.2.2	assmb()	30
4.4.2.3	drchlt()	31
4.4.2.4	formkf()	31
4.4.2.5	neumann()	32
4.4.2.6	processor()	33
4.4.2.7	solver()	34
4.5	mscalar Module Reference	35
4.5.1	Detailed Description	35
4.5.2	Function/Subroutine Documentation	36
4.5.2.1	fracelem()	36
4.5.2.2	localelem()	37
4.5.2.3	localelem2d()	37
4.6	msetup Module Reference	38
4.6.1	Detailed Description	38
4.6.2	Function/Subroutine Documentation	39
4.6.2.1	preprocessor()	39
4.6.2.2	setupphase()	40
4.7	mshapefunctions Module Reference	40
4.7.1	Detailed Description	41

4.7.2	Function/Subroutine Documentation	41
4.7.2.1	setint()	41
4.7.2.2	setint2()	42
4.7.2.3	shpf1d()	42
4.7.2.4	shpf2d()	43
4.7.3	Variable Documentation	43
4.7.3.1	quadext	43
4.7.3.2	w	43
4.7.3.3	wq	43
4.7.3.4	wt	44
4.7.3.5	xi	44
4.7.3.6	xiq	44
4.7.3.7	xit	44
4.8	msolver Module Reference	44
4.8.1	Detailed Description	44
4.8.2	Function/Subroutine Documentation	44
4.8.2.1	rsub()	44
4.8.2.2	tri()	45
4.9	intermo Module Reference	45
4.9.1	Detailed Description	46
4.9.2	Function/Subroutine Documentation	46
4.9.2.1	init_zcoef()	46
4.9.2.2	z_p()	46
4.9.3	Variable Documentation	47
4.9.3.1	zcoef	47
4.10	utilities Module Reference	47
4.10.1	Detailed Description	48
4.10.2	Function/Subroutine Documentation	48
4.10.2.1	check_conv()	48
4.10.2.2	f1()	48
4.10.2.3	factor_picard()	49
4.10.2.4	linspace()	50
4.10.2.5	print_matrix()	50
4.10.2.6	quad1()	50
4.10.2.7	scaling_picard()	51
4.10.2.8	test_shpf1d()	52
4.11	scalarstructure Module Reference	52
4.11.1	Detailed Description	52
4.11.2	Function/Subroutine Documentation	53
4.11.2.1	mallocelemkf()	53
4.11.2.2	mallocglobalkf()	53

5	Data Type Documentation	55
5.1	meshstructure::mesh Type Reference	55
5.1.1	Detailed Description	56
5.1.2	Member Data Documentation	56
5.1.2.1	ei	56
5.1.2.2	ej	57
5.1.2.3	ek	57
5.1.2.4	filename	57
5.1.2.5	flagnode	57
5.1.2.6	geokind	57
5.1.2.7	gnode	57
5.1.2.8	mat	57
5.1.2.9	nelem	57
5.1.2.10	nelems	58
5.1.2.11	nen	58
5.1.2.12	nintp	58
5.1.2.13	nnodes	58
5.1.2.14	nsd	58
5.1.2.15	numat	58
5.1.2.16	si	58
5.1.2.17	sj	58
5.1.2.18	sk	59
5.1.2.19	x	59
5.1.2.20	xv	59
5.1.2.21	yv	59
5.2	scalarstructure::scalarstructuresystem Type Reference	59
5.2.1	Detailed Description	60
5.2.2	Member Data Documentation	60
5.2.2.1	dt	60
5.2.2.2	kbc	60
5.2.2.3	lhelem	61
5.2.2.4	lhsys	61
5.2.2.5	linflag	61
5.2.2.6	mat	61
5.2.2.7	nsteps	61
5.2.2.8	rhelem	61
5.2.2.9	rhsys	61
5.2.2.10	tprint	61
5.2.2.11	transient	62
5.2.2.12	u	62
5.2.2.13	u_prev	62
5.2.2.14	u_prev_it	62
5.2.2.15	vbc	62

6 File Documentation	63
6.1 src/driver.F90 File Reference	63
6.1.1 Function/Subroutine Documentation	63
6.1.1.1 finel()	63
6.2 src/io.F90 File Reference	64
6.3 src/meshStructure.F90 File Reference	65
6.4 src/mInputReader.F90 File Reference	66
6.5 src/processor.F90 File Reference	67
6.6 src/scalar.F90 File Reference	67
6.7 src/scalarStructure.F90 File Reference	68
6.8 src/setup.F90 File Reference	68
6.9 src/shapeFunctions.F90 File Reference	69
6.10 src/solver.F90 File Reference	69
6.11 src/termo.F90 File Reference	70
6.12 src/utilities.F90 File Reference	70
Index	71

Chapter 1

Modules Index

1.1 Modules List

Here is a list of all modules with brief descriptions:

meshstructure	Module that contains the data structure of a mesh associate to a problem	7
minputreader	Modulo responsavel por reunir subrotinas para leitura do arquivo de entrada	8
mio	Contains input/output variables and routines	20
mprocessor	Processor module to compute, assemble and solve the system	28
mscalar	Contains variables and subroutine related to a general scalar problem	35
msetup	Module for setup phase by IO procedures	38
mshapefunctions	Module for shape functions computations and relate operations	40
msolver	Contains subroutine to compute numerical solution of linear systems $Ax=b$	44
mtermo	Module that contains thermodynamics function and parameters	45
mutilities	Module for auxiliar routines	47
scalarstructure	Module that contains the data structure of a general scalar problem	52

Chapter 2

Data Type Index

2.1 Data Types List

Here are the data types with brief descriptions:

meshstructure::mesh	
Data type for a mesh	55
scalarstructure::scalarstructuresystem	
Variables and characteristic data for a scalar problem	59

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

src/ driver.F90	63
src/ io.F90	64
src/ meshStructure.F90	65
src/ mInputReader.F90	66
src/ processor.F90	67
src/ scalar.F90	67
src/ scalarStructure.F90	68
src/ setup.F90	68
src/ shapeFunctions.F90	69
src/ solver.F90	69
src/ termo.F90	70
src/ utilities.F90	70

Chapter 4

Module Documentation

4.1 meshstructure Module Reference

Module that contains the data structure of a mesh associate to a problem.

Data Types

- type `mesh`
Data type for a mesh.

Functions/Subroutines

- subroutine `mallocnodes` (meshStrct)
Routine that allocate memory to node data.
- subroutine `mallocelem` (meshStrct)
Routine that allocate memory to element data.

4.1.1 Detailed Description

Module that contains the data structure of a mesh associate to a problem.

Author

Diego T. Volpatto

4.1.2 Function/Subroutine Documentation

4.1.2.1 mallocelem()

```
subroutine meshstructure::mallocelem (  
    type(mesh) meshStrct )
```

Routine that allocate memory to element data.

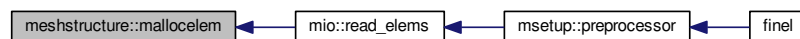
Parameters

<i>meshStrct</i>	[in/out] mesh structure to allocate
<i>n</i>	[in] number of elements

Author

Diego T. Volpatto

Here is the caller graph for this function:

**4.1.2.2 mallocnodes()**

```

subroutine meshstructure::mallocnodes (
    type(mesh) meshStrct )
  
```

Routine that allocate memory to node data.

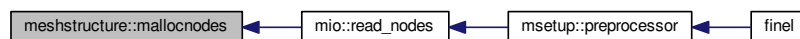
Parameters

<i>meshStrct</i>	[in/out] mesh structure to allocate
<i>n</i>	[in] number of nodes

Author

Diego T. Volpatto

Here is the caller graph for this function:

**4.2 minputreader Module Reference**

Modulo responsavel por reunir subrotinas para leitura do arquivo de entrada.

Functions/Subroutines

- subroutine [readinputfileds](#) ()
Le arquivo de input e armazena seu conteudo em um array.
- subroutine [createsimpleinputfile](#) ()
Cria a estrutura de input usando um arquivo de entrada sem includes.
- subroutine [mergeincludecontents](#) (include_file, include_line)
Le o conteudo do arquivo de include e armazena no array principal.
- subroutine [preparefilelines](#) (include_indexes, include_number_of_lines, number_of_includes, original_file_lines)
Efetua a alocação da estrutura definitiva, preparando a linha dos arquivos originais para receber os includes.
- subroutine [analyzefileinput](#) (number_of_lines, number_of_includes)
Efetua algumas analises no arquivo recebido.
- subroutine [analyzefile](#) (file_name, number_of_lines, number_of_includes)
Efetua algumas analises no arquivo recebido.
- integer *4 function [findinclude](#) (position, file_lines, number_of_lines)
Procura a n-esima palavra-chave include.
- integer *4 function [findkeyword](#) (keyword)
Procura uma palavra-chave.
- subroutine [readintegerkeywordvalue](#) (keyword, target, default_value)
Efetua a leitura de uma palavra-chave to tipo inteiro. Se nao encontrado, associa o valor default fornecido.
- subroutine [readintarraykeywordvalue](#) (keyword, target, default_value)
Efetua a leitura de uma palavra-chave do tipo array de inteiro. Se nao encontrado, associa o valor default fornecido. Obs.: Atentar para o fato dessa sub-rotina ter um do "infinito".
- subroutine [readstringkeywordvalue](#) (keyword, target, default_value)
Efetua a leitura de uma palavra-chave to tipo string. Se nao encontrado, associa o valor default fornecido.
- subroutine [readrealkeywordvalue](#) (keyword, target, default_value)
Efetua a leitura de uma palavra-chave to tipo real. Se nao encontrado, associa o valor default fornecido.
- subroutine [readintegerarrayvalues](#) (keyword, target, default_value)
Efetua a leitura de uma palavra-chave do tipo de um array de inteiros. A leitura eh realizada linha por linha. A primeira linha informa o numero de valores a ser lido. Se nao encontrada a keyword, associa o valor default fornecido.
- subroutine [readrealmatrixvalues](#) (keyword, target, default_value)
Efetua a leitura de uma palavra-chave do tipo de um array bidimensional real. A leitura eh realizada linha por linha. Se nao encontrado, associa o valor default fornecido.
- subroutine [readboundaryconditions](#) (keyword, kbc, vbc, default_value)
Efetua a leitura de uma palavra-chave do tipo de um array bidimensional real. A leitura eh realizada linha por linha. Se nao encontrado, associa o valor default fornecido.

Variables

- character(len=200), dimension(:), allocatable [file_lines](#)
Armazena as linhas do arquivo de input.
- integer *4 [number_of_lines](#)
Armazena o numero de linhas no arquivo.

4.2.1 Detailed Description

Modulo responsavel por reunir subrotinas para leitura do arquivo de entrada.

4.2.2 Function/Subroutine Documentation

4.2.2.1 analyzefile()

```
subroutine minputreader::analyzefile (
    character(len=200) file_name,
    integer*4 number_of_lines,
    integer*4 number_of_includes )
```

Efetua algumas analises no arquivo recebido.

Parameters

<i>file_name</i>	O nome do arquivo.
<i>number_of_lines</i>	Numero de linhas.
<i>number_of_include</i>	Numero de ocorrencias da palavra include.

Here is the caller graph for this function:



4.2.2.2 analyzefileinput()

```
subroutine minputreader::analyzefileinput (
    integer*4 number_of_lines,
    integer*4 number_of_includes )
```

Efetua algumas analises no arquivo recebido.

Parameters

<i>number_of_lines</i>	Numero de linhas.
<i>number_of_include</i>	Numero de ocorrencias da palavra include.

Here is the caller graph for this function:



4.2.2.3 createsimpleinputfile()

```
subroutine minputreader::createsimpleinputfile ( )
```

Cria a estrutura de input usando um arquivo de entrada sem includes.

Parameters

<i>file_name</i>	Nome do arquivo a ser lido.
------------------	-----------------------------

Here is the caller graph for this function:

**4.2.2.4 findinclude()**

```
integer*4 function minputreader::findinclude (
    integer*4 position,
    character(len=200), dimension(:) file_lines,
    integer*4 number_of_lines )
```

Procura a n-esima palavra-chave include.

Parameters

<i>position</i>	Corresponde a posicao desejada.
<i>file_lines</i>	Linhas do arquivo.
<i>number_of_lines</i>	Numero de linhas atuais.

Returns

O indice da palavra-chave no array que contem as linhas do arquivo de entrada.

Here is the caller graph for this function:

**4.2.2.5 findkeyword()**

```
integer*4 function minputreader::findkeyword (
    character(50) keyword )
```

Procura uma palavra-chave.

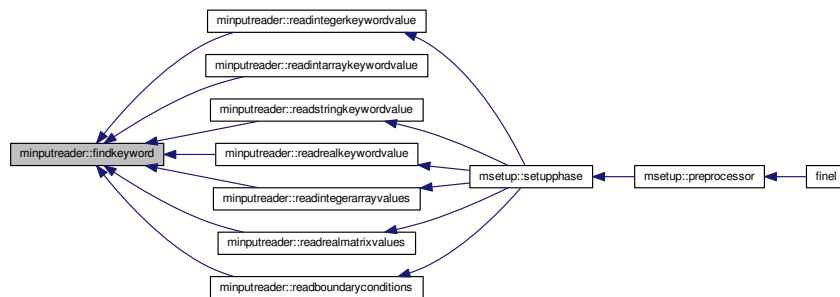
Parameters

<i>keyword</i>	A palavra-chave.
----------------	------------------

Returns

O indice da palavra-chave no array que contem as linhas do arquivo de entrada.

Here is the caller graph for this function:

**4.2.2.6 mergeincludecontents()**

```

subroutine minputreader::mergeincludecontents (
    character(len=200) include_file,
    integer*4 include_line )
  
```

Le o conteudo do arquivo de include e armazena no array principal.

Parameters

<i>include_index</i>	O index do include.
<i>include_files</i>	Array com includes.
<i>include_line</i>	A linha do include.

Here is the caller graph for this function:

**4.2.2.7 preparefilelines()**

```

subroutine minputreader::preparefilelines (
    integer*4, dimension(:) include_indexes,
    integer*4, dimension(:) include_number_of_lines,
    integer*4 number_of_includes,
    character(len=200), dimension(:) original_file_lines )
  
```

Efetua a alocação da estrutura definitiva, preparando a linha dos arquivos originais para receber os includes.

Parameters

<i>include_indexes</i>	Array os indices de ocorrencias dos includes.
<i>include_number_of_lines</i>	Array com o numero de linhas de cada include
<i>number_of_includes</i>	Numero de includes.
<i>original_file_lines</i>	Linhas do arquivo de entrada original.

Here is the caller graph for this function:



4.2.2.8 readboundaryconditions()

```

subroutine minputreader::readboundaryconditions (
    character(50) keyword,
    integer*4, dimension(:), allocatable kbc,
    real*8, dimension(:), allocatable vbc,
    real(8) default_value )
  
```

Efetua a leitura de uma palavra-chave do tipo de um array bidimensional real. A leitura eh realizada linha por linha. Se nao encontrado, associa o valor default fornecido.

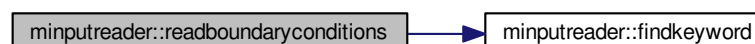
Parameters

<i>keyword</i>	A palavra-chave a ser encontrada.
<i>kbc</i>	Tipo da CC (1 = Dirichlet, 2 = Neumann).
<i>vbc</i>	Valor prescrito na CC.
<i>default_value</i>	Valor default.

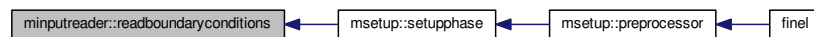
Author

Diego T. Volpatto

Here is the call graph for this function:



Here is the caller graph for this function:



4.2.2.9 readinputfiles()

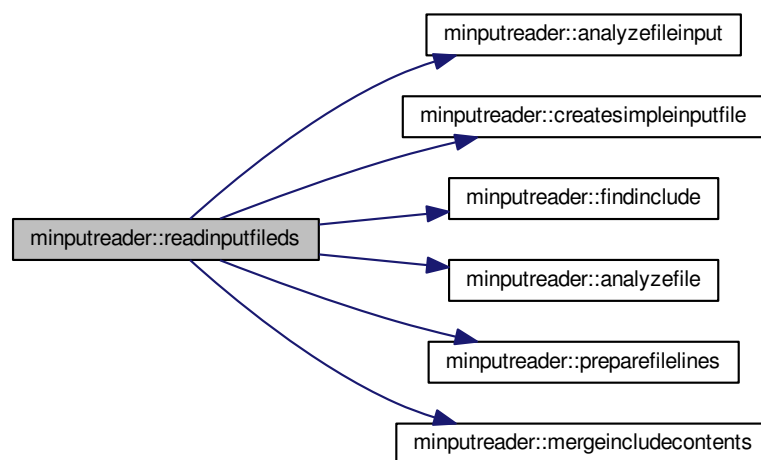
```
subroutine minputreader::readinputfiles ( )
```

Le arquivo de input e armazena seu conteudo em um array.

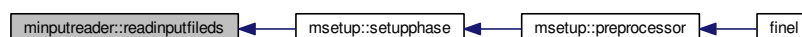
Parameters

<i>file_name</i>	Nome do arquivo a ser lido.
------------------	-----------------------------

Here is the call graph for this function:



Here is the caller graph for this function:



4.2.2.10 readintarraykeywordvalue()

```
subroutine minputreader::readintarraykeywordvalue (
    character(50) keyword,
    integer*4, dimension(:), allocatable target,
    integer*4 default_value )
```

Efetua a leitura de uma palavra-chave do tipo array de inteiro. Se nao encontrado, associa o valor default fornecido.
Obs.: Atentar para o fato dessa sub-rotina ter um do "infinito".

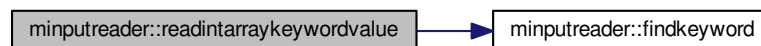
Parameters

<i>keyword</i>	A palavra-chave a ser encontrada.
<i>target</i>	Variavel onde o valor inteiro sera atribuido.
<i>default_value</i>	Valor default.

Author

Diego Volpatto

Here is the call graph for this function:



4.2.2.11 readintegerarrayvalues()

```
subroutine minputreader::readintegerarrayvalues (
    character(50) keyword,
    integer*4, dimension(:), allocatable target,
    integer(4) default_value )
```

Efetua a leitura de uma palavra-chave do tipo de um array de inteiros. A leitura eh realizada linha por linha. A primeira linha informa o numero de valores a ser lido. Se nao encontrada a keyword, associa o valor default fornecido.

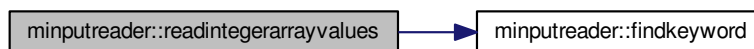
Parameters

<i>keyword</i>	A palavra-chave a ser encontrada.
<i>target</i>	Variavel onde os valores serao atribuido.
<i>default_value</i>	Valor default.

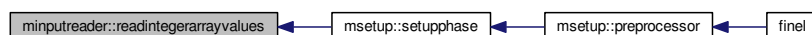
Author

Diego T. Volpatto

Here is the call graph for this function:



Here is the caller graph for this function:



4.2.2.12 readintegerkeywordvalue()

```

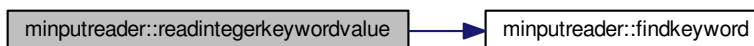
subroutine minputreader::readintegerkeywordvalue (
    character(50) keyword,
    target,
    integer*4, target default_value )
  
```

Efetua a leitura de uma palavra-chave to tipo inteiro. Se nao encontrado, associa o valor default fornecido.

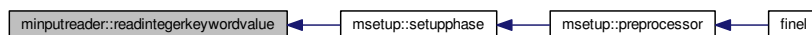
Parameters

<i>keyword</i>	A palavra-chave a ser encontrada.
<i>target</i>	Variavel onde o valor inteiro sera atribuido.
<i>default_value</i>	Valor default.

Here is the call graph for this function:



Here is the caller graph for this function:



4.2.2.13 readrealkeywordvalue()

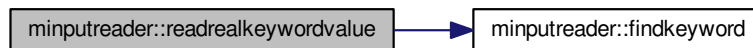
```
subroutine minputreader::readrealkeywordvalue (
    character(50) keyword,
    target,
    real(8), target default_value )
```

Efetua a leitura de uma palavra-chave to tipo real. Se nao encontrado, associa o valor default fornecido.

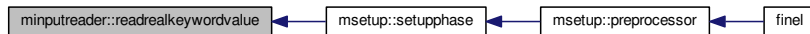
Parameters

<i>keyword</i>	A palavra-chave a ser encontrada.
<i>target</i>	Variavel onde o real sera atribuido.
<i>default_value</i>	Valor default.

Here is the call graph for this function:



Here is the caller graph for this function:



4.2.2.14 readrealmatrixvalues()

```
subroutine minputreader::readrealmatrixvalues (
    character(50) keyword,
    real*8, dimension(:,:), allocatable target,
    real(8) default_value )
```

Efetua a leitura de uma palavra-chave do tipo de um array bidimensional real. A leitura eh realizada linha por linha. Se nao encontrado, associa o valor default fornecido.

Parameters

<i>keyword</i>	A palavra-chave a ser encontrada.
<i>target</i>	Variavel onde os valores serao atribuido.
<i>default_value</i>	Valor default.

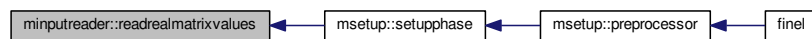
Author

Diego T. Volpatto

Here is the call graph for this function:



Here is the caller graph for this function:

**4.2.2.15 readstringkeywordvalue()**

```

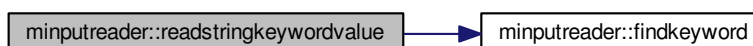
subroutine minputreader::readstringkeywordvalue (
    character(50) keyword,
    character(len=*) target,
    character(len=*) default_value )
  
```

Efetua a leitura de uma palavra-chave to tipo string. Se nao encontrado, associa o valor default fornecido.

Parameters

<i>keyword</i>	A palavra-chave a ser encontrada.
<i>target</i>	Variavel onde a string sera atribuido.
<i>default_value</i>	Valor default.

Here is the call graph for this function:



Here is the caller graph for this function:



4.2.3 Variable Documentation

4.2.3.1 file_lines

```
character(len=200), dimension(:), allocatable minputreader::file_lines
```

Armazena as linhas do arquivo de input.

4.2.3.2 number_of_lines

```
integer*4 minputreader::number_of_lines
```

Armazena o numero de linhas no arquivo.

4.3 mio Module Reference

Contains input/output variables and routines.

Functions/Subroutines

- subroutine [openfiles](#) ()
Open IO files.
- subroutine [closefiles](#) ()
Close IO files.
- subroutine [read_nodes](#) (mesh_)
Subroutine to read node data file generated by EasyMesh.
- subroutine [read_elems](#) (mesh_)
Subroutine to read element data file generated by EasyMesh.
- subroutine [print_sol](#) (mesh_, scalar_, tstep)
Prints the solution of scalar field.
- subroutine [print_sol_csv](#) (mesh_, scalar_, tstep)
Prints the solution of scalar field in the csv format aiming to compatibility with Paraview post-processing.
- subroutine [print_sol_vtk](#) (mesh_, scalar_, tstep)
Prints the solution of scalar field in the vtk "legacy" format aiming to compatibility with Paraview post-processing.
- subroutine [print_files](#) (mesh_, scalar_, tstep)
Record solution according to specified file kind in input.

Variables

- integer, parameter `iin` = 1110
Input file id.
- integer, parameter `iout` = 1120
Output file id.
- integer, parameter `isol` = 1130
Solution file id.
- integer, parameter `ioutn` = 1140
Node output file.
- integer, parameter `ioute` = 1150
Element output file.
- integer, parameter `isolcsv` = 1160
Solution csv file id.
- integer, parameter `isolvtk` = 1170
Solution vtk file id.
- character(len=20), parameter `infile` ='input.dat'
Input file name.
- character(len=20), parameter `outfile` ='output.dat'
Output file name.
- character(len=20), parameter `outnodes` ='outnodes.dat'
Output node's file name.
- character(len=20), parameter `outelem` ='outelem.dat'
Output elements' file name.
- character(len=50), parameter `solfile` ='solution00'
Solution file name.
- character(len=50) `title`
- character(len=50) `file_format`

4.3.1 Detailed Description

Contains input/output variables and routines.

Author

Diego T. Volpatto

4.3.2 Function/Subroutine Documentation

4.3.2.1 closefiles()

```
subroutine mio::closefiles ( )
```

Close IO files.

Author

Diego T. Volpatto

Here is the caller graph for this function:

**4.3.2.2 openfiles()**

```
subroutine mio::openfiles ( )
```

Open IO files.

Author

Diego T. Volpatto

Here is the caller graph for this function:

**4.3.2.3 print_files()**

```
subroutine mio::print_files (
    type(mesh) mesh_,
    type(scalarstructuresystem) scalar_,
    integer tstep )
```

Record solution according to specified file kind in input.

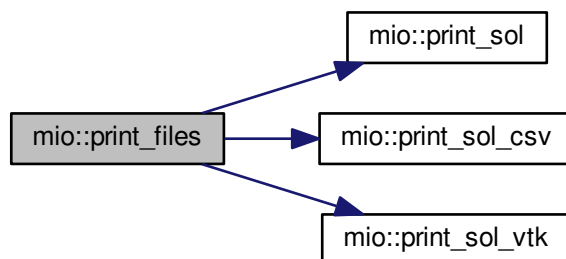
Parameters

<i>mesh</i> ↔	A mesh structure
—	
<i>scalar</i> ↔	A scalar structure
—	
<i>tstep</i>	Time step to print

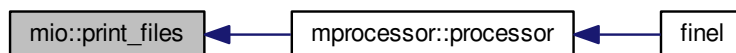
Author

Diego T. Volpatto

Here is the call graph for this function:



Here is the caller graph for this function:



4.3.2.4 print_sol()

```

subroutine mio::print_sol (
    type(mesh) mesh_,
    type(scalarstructuresystem) scalar_,
    integer tstep )
  
```

Prints the solution of scalar field.

Parameters

<i>mesh</i> ↔	A mesh structure
—	
<i>scalar</i> ↔	A scalar structure
—	
<i>tstep</i>	Time step index

Author

Diego T. Volpatto

Here is the caller graph for this function:

**4.3.2.5 print_sol_csv()**

```

subroutine mio::print_sol_csv (
    type(mesh) mesh_,
    type(scalarstructuresystem) scalar_,
    integer tstep )
  
```

Prints the solution of scalar field in the csv format aiming to compatibility with Paraview post-processing.

Parameters

<i>mesh</i> ↔	A mesh structure
—	
<i>scalar</i> ↔	A scalar structure
—	
<i>tstep</i>	Time step index

Author

Diego T. Volpatto

Here is the caller graph for this function:

**4.3.2.6 print_sol_vtk()**

```

subroutine mio::print_sol_vtk (
    type(mesh) mesh_,
    type(scalarstructuresystem) scalar_,
    integer tstep )
  
```

Prints the solution of scalar field in the vtk "legacy" format aiming to compatibility with Paraview post-processing.

Parameters

<i>mesh</i> ↔ —	A mesh structure
<i>scalar</i> ↔ —	A scalar structure
<i>tstep</i>	Current time step

Author

Diego T. Volpatto

Here is the caller graph for this function:



4.3.2.7 read_elems()

```

subroutine mio::read_elems (
    type(mesh) mesh_ )
  
```

Subroutine to read element data file generated by EasyMesh.

Parameters

<i>mesh</i> ↔ —	[in/out] a mesh structure
--------------------	---------------------------

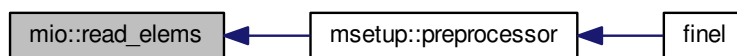
Author

Diego Volpatto

Here is the call graph for this function:



Here is the caller graph for this function:



4.3.2.8 read_nodes()

```

subroutine mio::read_nodes (
    type(mesh) mesh_ )
  
```

Subroutine to read node data file generated by EasyMesh.

Parameters

<i>mesh</i> ↔	[in/out] a mesh structure
—	

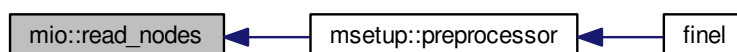
Author

Diego Volpatto

Here is the call graph for this function:



Here is the caller graph for this function:



4.3.3 Variable Documentation

4.3.3.1 file_format

```
character(len=50) mio::file_format
```

4.3.3.2 iin

```
integer, parameter mio::iin = 1110
```

Input file id.

4.3.3.3 infile

```
character(len=20), parameter mio::infile = 'input.dat'
```

Input file name.

4.3.3.4 iout

```
integer, parameter mio::iout = 1120
```

Output file id.

4.3.3.5 ioute

```
integer, parameter mio::ioute = 1150
```

Element output file.

4.3.3.6 ioutn

```
integer, parameter mio::ioutn = 1140
```

Node output file.

4.3.3.7 isol

```
integer, parameter mio::isol = 1130
```

Solution file id.

4.3.3.8 isolcsv

```
integer, parameter mio::isolcsv = 1160
```

Solution csv file id.

4.3.3.9 isolvtk

```
integer, parameter mio::isolvtk = 1170
```

Solution vtk file id.

4.3.3.10 outelem

```
character(len=20), parameter mio::outelem = 'outelem.dat'
```

Output elements' file name.

4.3.3.11 outfile

```
character(len=20), parameter mio::outfile = 'output.dat'
```

Output file name.

4.3.3.12 outnodes

```
character(len=20), parameter mio::outnodes = 'outnodes.dat'
```

Output node's file name.

4.3.3.13 solfile

```
character(len=50), parameter mio::solfile = 'solution00'
```

Solution file name.

4.3.3.14 title

```
character(len=50) mio::title
```

4.4 mprocessor Module Reference

Processor module to compute, assemble and solve the system.

Functions/Subroutines

- subroutine `formkf` (mesh_, scalar_, t)
Form and assemble $Ku = F$ system.
- subroutine `assmb` (mesh_, scalar_, nel)
Assemble element stiffness matrix and load vector to global stiffness matrix and load vector, respectively.
- subroutine `drchlt` (mesh_, scalar_, n)
Apply Dirichlet Boundary Condition.
- subroutine `neumann` (mesh_, scalar_, n)
Apply Neumann Boundary Condition – 1D. Prescribe $-k(x)u = vbc$.
- subroutine `applybc` (mesh_, scalar_)
Modify $Ku=F$ system to incorporate BC data.
- subroutine `solver` (mesh_, scalar_)
Executes Gauss reduction and forward substitution solving $Ku=F$.
- subroutine `processor` (mesh_, scalar_)
Processor routine phase.

4.4.1 Detailed Description

Processor module to compute, assemble and solve the system.

Author

Diego T. Volpatto

4.4.2 Function/Subroutine Documentation

4.4.2.1 `applybc()`

```
subroutine mprocessor::applybc (
    type(mesh) mesh_,
    type(scalarstructuresystem) scalar_ )
```

Modify $Ku=F$ system to incorporate BC data.

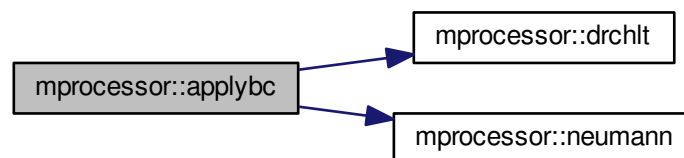
Parameters

<i>mesh</i> ↔ —	A mesh structure
<i>scalar</i> ↔ —	A scalar structure

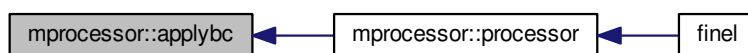
Author

Diego Volpatto

Here is the call graph for this function:



Here is the caller graph for this function:



4.4.2.2 assmb()

```

subroutine mprocessor::assmb (
    type(mesh) mesh_,
    type(scalarstructuresystem) scalar_,
    integer nel )
  
```

Assemble element stiffness matrix and load vector to global stiffness matrix and load vector, respectively.

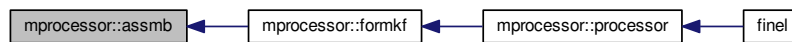
Parameters

<i>mesh</i> ↔	A mesh structure
—	
<i>scalar</i> ↔	A scalar structure
—	
<i>nel</i>	Index of current element

Author

Diego Volpatto

Here is the caller graph for this function:



4.4.2.3 drchlt()

```

subroutine mprocessor::drchlt (
    type(mesh) mesh_,
    type(scalarstructuresystem) scalar_,
    integer n )
  
```

Apply Dirichlet Boundary Condition.

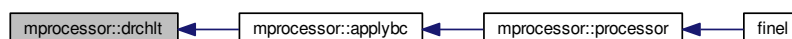
Parameters

<i>mesh</i> ↔	A mesh structure
—	
<i>scalar</i> ↔	A scalar structure
—	
<i>n</i>	Node index of BC

Author

Diego Volpatto

Here is the caller graph for this function:



4.4.2.4 formkf()

```

subroutine mprocessor::formkf (
    type(mesh) mesh_,
    type(scalarstructuresystem) scalar_,
    real*8 t )
  
```

Form and assemble $Ku = F$ system.

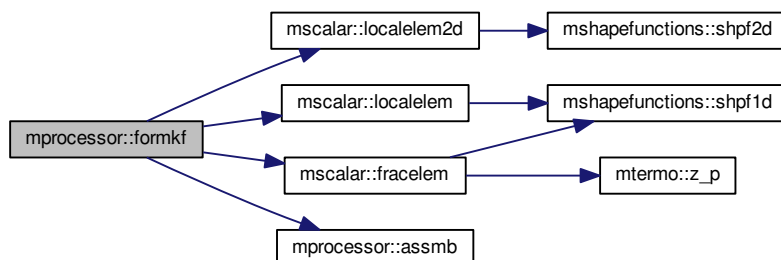
Parameters

<i>mesh</i> ↔	[in/out] A mesh structure
<i>scalar</i> ↔	[in/out] A scalar structure
<i>t</i>	[in] Current simulation time

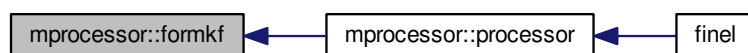
Author

Diego Volpatto

Here is the call graph for this function:



Here is the caller graph for this function:



4.4.2.5 neumann()

```

subroutine mprocessor::neumann (
    type(mesh) mesh_,
    type(scalarstructuresystem) scalar_,
    integer n )

```

Apply Neumann Boundary Condition – 1D. Prescribe $-k(x)u = vbc$.

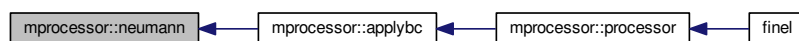
Parameters

<i>mesh</i> ↔ —	A mesh structure
<i>scalar</i> ↔ —	A scalar structure
<i>n</i>	Node index of BC

Author

Diego Volpatto

Here is the caller graph for this function:



4.4.2.6 processor()

```

subroutine mprocessor::processor (
    type(mesh) mesh_,
    type(scalarstructuresystem) scalar_ )
  
```

Processor routine phase.

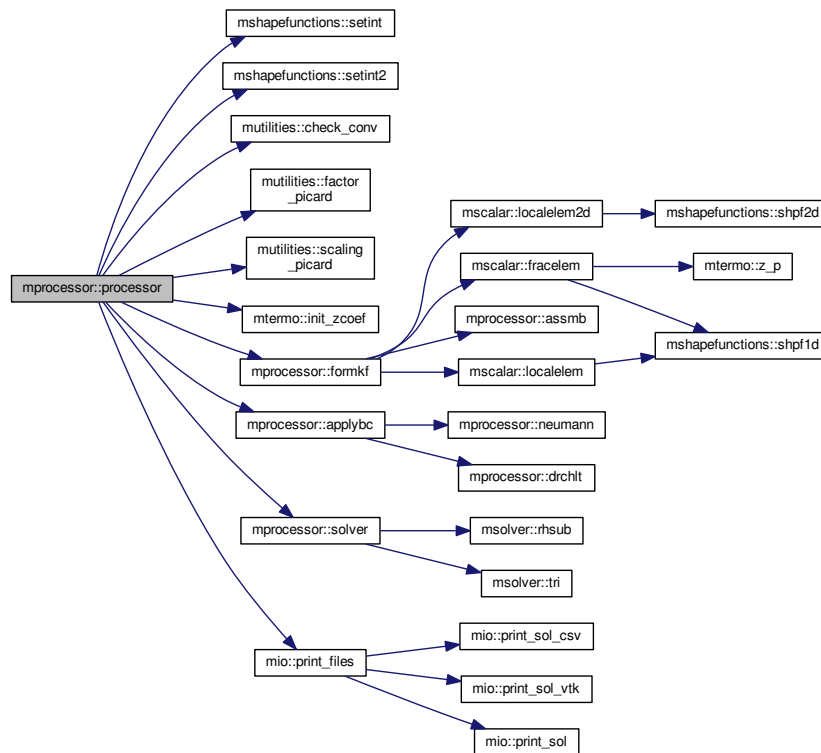
Parameters

<i>mesh</i> ↔ —	A mesh structure
<i>scalar</i> ↔ —	A scalar structure

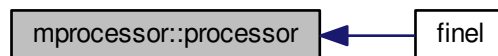
Author

Diego T. Volpatto

Here is the call graph for this function:



Here is the caller graph for this function:



4.4.2.7 solver()

```

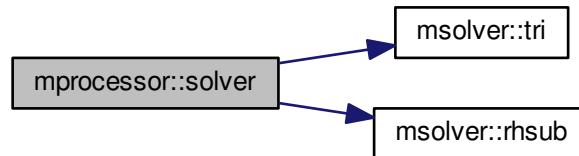
subroutine mprocessor::solver (
    type(mesh) mesh_,
    type(scalarstructuresystem) scalar_ )
  
```

Executes Gauss reduction and forward substitution solving $Ku=F$.

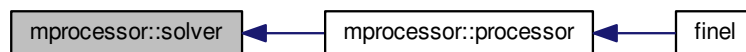
Parameters

<i>mesh</i> ↔ —	A mesh structure
<i>scalar</i> ↔ —	A scalar structure

Here is the call graph for this function:



Here is the caller graph for this function:



4.5 mscalar Module Reference

Contains variables and subroutine related to a general scalar problem.

Functions/Subroutines

- subroutine [localelem](#) (mesh_, scalar_, nel)
Computes a master element contribution – 1D.
- subroutine [localelem2d](#) (mesh_, scalar_, nel)
Computes a master element contribution – 2D.
- subroutine [fracelem](#) (mesh_, scalar_, nel, t)
Computes a master element contribution in fracture case – 1D.

4.5.1 Detailed Description

Contains variables and subroutine related to a general scalar problem.

Author

Diego T. Volpatto

4.5.2 Function/Subroutine Documentation

4.5.2.1 fracelem()

```
subroutine mscalar::fracelem (
    type(mesh) mesh_,
    type(scalarstructuresystem) scalar_,
    integer nel,
    real*8 t )
```

Computes a master element contribution in fracture case – 1D.

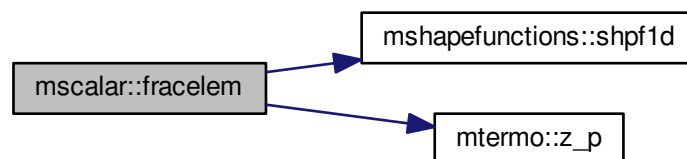
Parameters

<i>mesh</i> ↔	[in/out] A mesh structure
<i>scalar</i> ↔	[in/out] A scalar structure
<i>nel</i>	[in] Index of current element

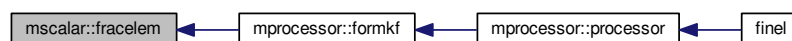
Author

Diego Volpatto

Here is the call graph for this function:



Here is the caller graph for this function:



4.5.2.2 localelem()

```

subroutine mscalar::localelem (
    type(mesh) mesh_,
    type(scalarstructuresystem) scalar_,
    integer nel )

```

Computes a master element contribution – 1D.

Parameters

<i>mesh</i> ↔ —	[in/out] A mesh structure
<i>scalar</i> ↔ —	[in/out] A scalar structure
<i>nel</i>	[in] Index of current element

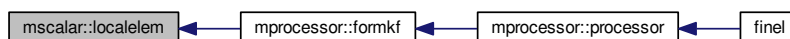
Author

Diego Volpatto

Here is the call graph for this function:



Here is the caller graph for this function:



4.5.2.3 localelem2d()

```

subroutine mscalar::localelem2d (
    type(mesh) mesh_,
    type(scalarstructuresystem) scalar_,
    integer nel )

```

Computes a master element contribution – 2D.

Parameters

<i>mesh</i> ↔	[in/out] A mesh structure
—	
<i>scalar</i> ↔	[in/out] A scalar structure
—	
<i>nel</i>	[in] Index of current element

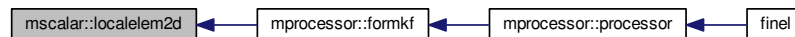
Author

Diego Volpatto

Here is the call graph for this function:



Here is the caller graph for this function:



4.6 msetup Module Reference

Module for setup phase by IO procedures.

Functions/Subroutines

- subroutine [setupphase](#) (mesh_, scalar_)
Reads parameters from input file.
- subroutine [preprocessor](#) (mesh_, scalar_)
Realizes preprocessor routines.

4.6.1 Detailed Description

Module for setup phase by IO procedures.

Author

Diego T. Volpatto

4.6.2 Function/Subroutine Documentation

4.6.2.1 preprocessor()

```
subroutine msetup::preprocessor (
    type(mesh) mesh_,
    type(scalarstructuresystem) scalar_ )
```

Realizes preprocessor routines.

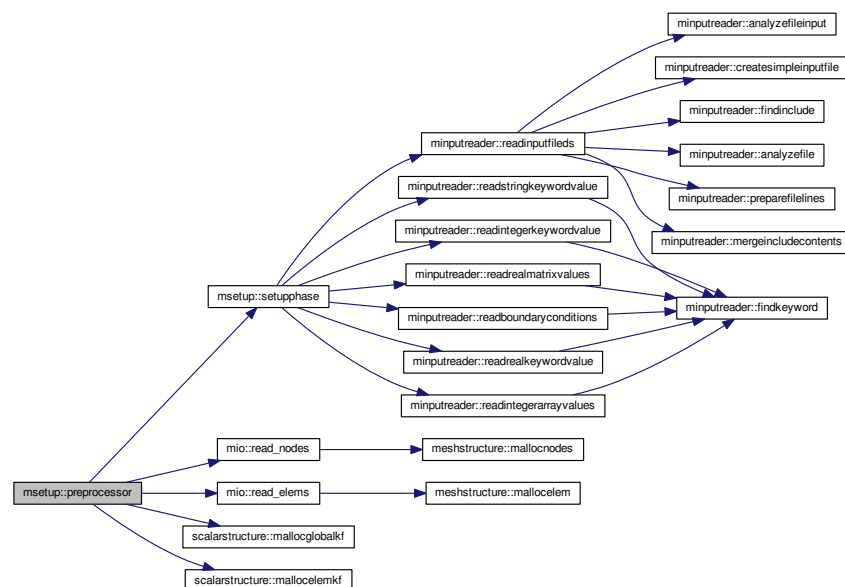
Parameters

<i>mesh</i> ↔	A mesh structure
—	
<i>scalar</i> ↔	A scalar structure
—	

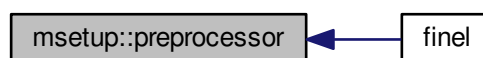
Author

Diego T. Volpatto

Here is the call graph for this function:



Here is the caller graph for this function:



4.6.2.2 setupphase()

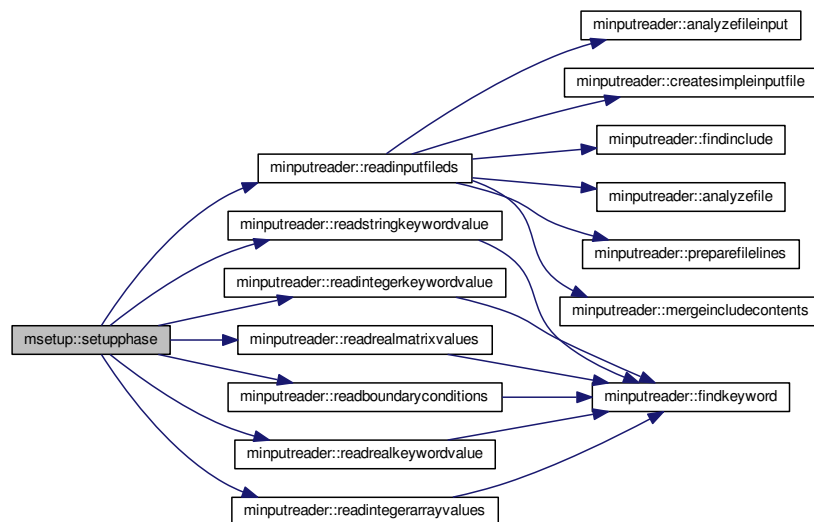
```
subroutine msetup::setupphase (
    type(mesh) mesh_,
    type(scalarstructuresystem) scalar_ )
```

Reads parameters from input file.

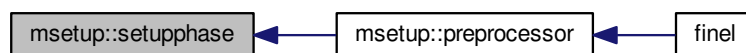
Parameters

<i>mesh</i> ↔ —	A mesh structure
<i>scalar</i> ↔ —	A scalar structure

Here is the call graph for this function:



Here is the caller graph for this function:



4.7 mshapefunctions Module Reference

Module for shape functions computations and relate operations.

Functions/Subroutines

- subroutine [setint](#)
Gauss quadrature data set routine - 1D.
- subroutine [setint2](#)
Gauss quadrature data set routine - 2D.
- subroutine [shpf1d](#) (xl, n, psi, dpsi)
Calculates the values of the shape functions and their derivatives - 1D.
- subroutine [shpf2d](#) (XL, N, PSI, DPSI)
Calculates the values of the shape functions and their derivatives - 2D.

Variables

- real *8, dimension(20, 20) [xi](#)
Gauss point integration 1D.
- real *8, dimension(20, 20) [w](#)
Gauss weights 1D.
- real *8, dimension(2, 9) [xiq](#)
Gauss points to rectangular element.
- real *8, dimension(9) [wq](#)
Gauss weights to rectangular element.
- real *8, dimension(2, 4) [xit](#)
Gauss points to triangle element.
- real *8, dimension(4) [wt](#)
Gauss weights to triangle element.
- integer [quadext](#)

4.7.1 Detailed Description

Module for shape functions computations and relate operations.

Author

Diego T. Volpatto

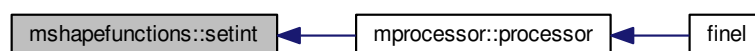
4.7.2 Function/Subroutine Documentation

4.7.2.1 [setint\(\)](#)

```
subroutine mshapefunctions::setint ( )
```

Gauss quadrature data set routine - 1D.

Here is the caller graph for this function:

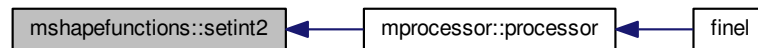


4.7.2.2 setint2()

```
subroutine mshapefunctions::setint2 ( )
```

Gauss quadrature data set routine - 2D.

Here is the caller graph for this function:



4.7.2.3 shpf1d()

```
subroutine mshapefunctions::shpf1d (
    real*8 xl,
    integer n,
    real*8, dimension(n) psi,
    real*8, dimension(n) dpsi )
```

Calculates the values of the shape functions and their derivatives - 1D.

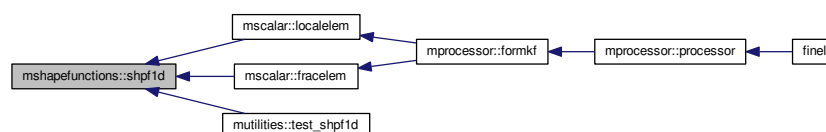
Parameters

<i>xl</i>	[in] specified value of master element coord
<i>n</i>	[in] number of element nodes
<i>psi</i>	[out] shape function values
<i>dpsi</i>	[out] derivatives shape functions values

Author

Diego Volpatto

Here is the caller graph for this function:



4.7.2.4 shpf2d()

```

subroutine mshapefunctions::shpf2d (
    real*8, dimension(2) XL,
    integer N,
    real*8, dimension(9) PSI,
    real*8, dimension(2,9) DPSI )

```

Calculates the values of the shape functions and their derivatives - 2D.

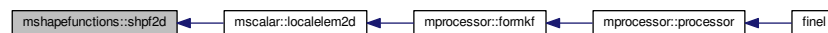
Parameters

<i>xl</i>	[in] specified value of master element coord
<i>n</i>	[in] number of element nodes
<i>psi</i>	[out] shape function values
<i>dpsi</i>	[out] derivatives shape functions values

Author

Diego Volpatto

Here is the caller graph for this function:



4.7.3 Variable Documentation

4.7.3.1 quadext

```
integer mshapefunctions::quadext
```

4.7.3.2 w

```
real*8, dimension(20,20) mshapefunctions::w
```

Gauss weights 1D.

4.7.3.3 wq

```
real*8, dimension(9) mshapefunctions::wq
```

Gauss weights to rectangular element.

4.7.3.4 wt

```
real*8, dimension(4) mshapefunctions::wt
```

Gauss weights to triangle element.

4.7.3.5 xi

```
real*8, dimension(20,20) mshapefunctions::xi
```

Gauss point integration 1D.

4.7.3.6 xiq

```
real*8, dimension(2,9) mshapefunctions::xiq
```

Gauss points to rectangular element.

4.7.3.7 xit

```
real*8, dimension(2,4) mshapefunctions::xit
```

Gauss points to triangle element.

4.8 msolver Module Reference

Contains subroutine to compute numerical solution of linear systems $Ax=b$.

Functions/Subroutines

- subroutine [tri](#) (A, n)
Applies Gauss reduction in $A(n,n)$ to obtain a superior triangular equivalent form.
- subroutine [rbsub](#) (A, x, b, n)
Does the forward substitution on the right-side-hand.

4.8.1 Detailed Description

Contains subroutine to compute numerical solution of linear systems $Ax=b$.

The present module has a general purpose such that the routines here intend to be independent of others module.

Author

Diego T. Volpatto

4.8.2 Function/Subroutine Documentation

4.8.2.1 rbsub()

```
subroutine msolver::rbsub (
    real*8, dimension(n,n) A,
    real*8, dimension(n) x,
    real*8, dimension(n) b,
    integer n )
```

Does the forward substitution on the right-side-hand.

Parameters

A	[in]A matrix $A(n,n)$
x	[out]Solution vector
b	[in/out]RHS-vector
n	[in]Number of solution points

Author

Diego T. Volpatto

Here is the caller graph for this function:



4.8.2.2 tri()

```

subroutine msolver::tri (
    real*8, dimension(n,n) A,
    integer n )
  
```

Applies Gauss reduction in $A(n,n)$ to obtain a superior triangular equivalent form.

Parameters

A	[in/out]A matrix $A(n,n)$
n	[in]Number of rows/columns of matrix A

Author

Diego T. Volpatto

Here is the caller graph for this function:



4.9 mtermo Module Reference

Module that contains thermodynamics function and parameters.

Functions/Subroutines

- subroutine `init_zcoef`
Initializes interpolation coefficients to compute compressibility factor.
- subroutine `z_p` (p, Z)
Computes compressibility factor.

Variables

- real *8, dimension(10) `zcoef`

4.9.1 Detailed Description

Module that contains thermodynamics function and parameters.

Author

Diego T. Volpatto

4.9.2 Function/Subroutine Documentation

4.9.2.1 `init_zcoef()`

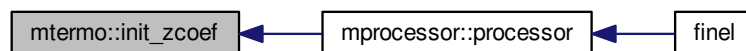
```
subroutine mtermo::init_zcoef ( )
```

Initializes interpolation coefficients to compute compressibility factor.

Author

Diego T. Volpatto

Here is the caller graph for this function:



4.9.2.2 `z_p()`

```
subroutine mtermo::z_p (
    real*8 p,
    real*8 Z )
```

Computes compressibility factor.

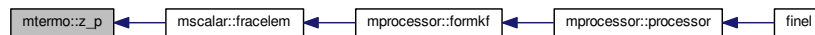
Parameters

p	[in] Pressure
Z	[out] Compressibility factor

Author

Diego T. Volpatto

Here is the caller graph for this function:



4.9.3 Variable Documentation

4.9.3.1 zcoef

```
real*8, dimension(10) mtermo::zcoef
```

4.10 mutilities Module Reference

Module for auxiliar routines.

Functions/Subroutines

- subroutine [linspace](#) ($x1$, $x2$, $nintv$, x)
Generate points between $x1$ and $x2$ equally spaced in $x(i)$. Same idea of numpy subroutine.
- real *8 function [f1](#) (x)
A function to test purpose.
- subroutine [quad1](#) (n , $x1$, $x2$)
Subroutine that computes gaussian quadrature of $f1$.
- subroutine [test_shpf1d](#) (n , $nelem$, x)
Check if shpf1d works properly.
- subroutine [print_matrix](#) (A , n , m)
Prints in the screen a matrix $A(n,m)$
- subroutine [check_conv](#) (u , $uprev$, $nnodes$, tol , $norm$, $flag$)
Routine to check if non-linear iteration converged.
- subroutine [factor_picard](#) ($alpha$, $delta$, eps , $omega_min$, $omega$)
Computes underrelaxation factor for Picard iteration.
- subroutine [scaling_picard](#) (i , $delta$, $deltap$, eps , rho , $omega$, $omega_min$, $alpha$)
Rescaling shape factor for underrelaxation.

4.10.1 Detailed Description

Module for auxiliar routines.

Author

Diego T. Volpatto

4.10.2 Function/Subroutine Documentation

4.10.2.1 `check_conv()`

```
subroutine mutilities::check_conv (
    real*8, dimension(nnodes) u,
    real*8, dimension(nnodes) uprev,
    integer nnodes,
    real*8 tol,
    real*8 norm,
    logical flag )
```

Routine to check if non-linear iteration converged.

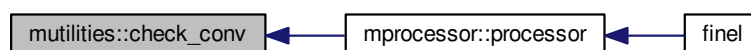
Parameters

<i>u</i>	Current solution vector
<i>uprev</i>	Previous iteration solution vector
<i>nnodes</i>	Number of nodes
<i>flag</i>	Flag for convergence checked

Author

Diego Volpatto

Here is the caller graph for this function:



4.10.2.2 `f1()`

```
real*8 function mutilities::f1 (
    real*8 x )
```

A function to test purpose.

Parameters

<i>x</i>	input coordinate
----------	------------------

Here is the caller graph for this function:



4.10.2.3 factor_picard()

```

subroutine mutilities::factor_picard (
    real*8 alpha,
    real*8 delta,
    real*8 eps,
    real*8 omega_min,
    real*8 omega )
  
```

Computes underrelaxation factor for Picard iteration.

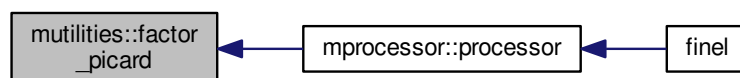
Parameters

<i>alpha</i>	[in] Shape factor for underrelaxation function
<i>delta</i>	[in] Error between two Picard iteration
<i>eps</i>	[in] Tolerance value (closure criterion)
<i>omega_min</i>	[in] Minimum value of underrelaxation function omega [out] Underrelaxation factor

Author

Diego T. Volpatto

Here is the caller graph for this function:



4.10.2.4 linspace()

```
subroutine mutilities::linspace (
    real*8 x1,
    real*8 x2,
    integer nintv,
    real*8, dimension(:), allocatable x )
```

Generate points between x1 and x2 equally spaced in x(i). Same idea of numpy subroutine.

Parameters

<i>x1</i>	interval lower bound
<i>x2</i>	interval upper bound
<i>nintv</i>	num of intervals
<i>x</i>	vector to assemble the values

4.10.2.5 print_matrix()

```
subroutine mutilities::print_matrix (
    real*8, dimension(n,m) A,
    integer n,
    integer m )
```

Prints in the screen a matrix A(n,m)

Parameters

<i>A</i>	A matrix
<i>n</i>	Number of lines of A
<i>m</i>	Number of columns of A

Author

Diego Volpatto

4.10.2.6 quad1()

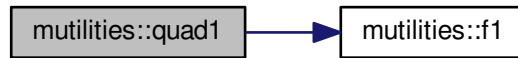
```
subroutine mutilities::quad1 (
    integer n,
    real*8 x1,
    real*8 x2 )
```

Subroutine that computes gaussian quadrature of f1.

Parameters

<i>n</i>	quadrature order
<i>x1</i>	integral lower bound
<i>x2</i>	integral upper bound

Here is the call graph for this function:



4.10.2.7 scaling_picard()

```

subroutine mutilities::scaling_picard (
    integer i,
    real*8 delta,
    real*8 deltap,
    real*8 eps,
    real*8 rho,
    real*8 omega,
    real*8 omega_min,
    real*8 alpha )
  
```

Rescaling shape factor for underrelaxation.

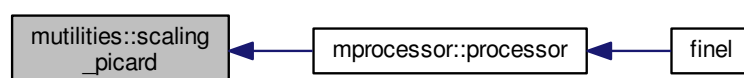
Parameters

<i>i</i>	[in] Current Picard iteration
<i>delta</i>	[in] Error between two Picard iteration
<i>deltap</i>	[in] Previous error between two Picard iteration
<i>eps</i>	[in] Tolerance value (closure criterion)
<i>rho</i>	[in] Scaling factor omega [in] Underrelaxation factor
<i>omega_min</i>	[in/out] Minimum value of underrelaxation function
<i>alpha</i>	[out] Shape factor for underrelaxation function

Author

Diego T. Volpatto

Here is the caller graph for this function:



4.10.2.8 test_shpf1d()

```
subroutine mutilities::test_shpf1d (
    integer n,
    integer nelem,
    real*8, dimension(nelem+1) x )
```

Check if shpf1d works properly.

Parameters

<i>n</i>	element node numbers
<i>nelem</i>	num of discrete intervals
<i>x</i>	master element's coordinates

Here is the call graph for this function:



4.11 scalarstructure Module Reference

Module that contains the data structure of a general scalar problem.

Data Types

- type [scalarstructuresystem](#)
Variables and characteristic data for a scalar problem.

Functions/Subroutines

- subroutine [mallocglobalkf](#) (scalar_, n)
Routine to allocate and clear the $Ku = F$ system.
- subroutine [mallocelemkf](#) (scalar_, n)
Routine to allocate and clear the element KF .

4.11.1 Detailed Description

Module that contains the data structure of a general scalar problem.

Author

Diego T. Volpatto

4.11.2 Function/Subroutine Documentation

4.11.2.1 mallocelemkf()

```
subroutine scalarstructure::mallocelemkf (
    type(scalarstructuresystem) scalar_,
    integer n )
```

Routine to allocate and clear the element KF.

Parameters

<i>scalar_</i> ↔	[in/out] A general scalar structure
<i>n</i>	[in] Number of element nodes

Here is the caller graph for this function:



4.11.2.2 mallocglobalkf()

```
subroutine scalarstructure::mallocglobalkf (
    type(scalarstructuresystem) scalar_,
    integer n )
```

Routine to allocate and clear the $Ku = F$ system.

Parameters

<i>scalar_</i> ↔	[in/out] A general scalar structure
<i>n</i>	[in] Number of global nodes

Here is the caller graph for this function:



Chapter 5

Data Type Documentation

5.1 meshstructure::mesh Type Reference

Data type for a mesh.

Collaboration diagram for meshstructure::mesh:

```
meshstructure::mesh
+ numat
+ nsd
+ nintp
+ nnodes
+ nelems
+ nen
+ x
+ flagnode
+ nele
+ xv
and 11 more...
```

Public Attributes

- integer **numat**
Number of materials.
- integer **nsd**
Number of spatial.
- integer **nintp**
Number of integration points.

- integer [nnodes](#)
Number of nodes.
- integer [nelems](#)
Number of elements.
- integer [nen](#)
Number of element's nodes.
- real *8, dimension(:,:), allocatable [x](#)
nodes coordinates
- integer *4, dimension(:), allocatable [flagnode](#)
boundary flag
- integer [nelem](#)
Number of elements.
- real *8, dimension(:), allocatable [xv](#)
Circumcenter Elem xcoord.
- real *8, dimension(:), allocatable [yv](#)
Circumcenter Elem ycoord.
- integer *4, dimension(:,:), allocatable [gnode](#)
Global node.
- integer *4, dimension(:), allocatable [mat](#)
Element material kind.
- integer *4, dimension(:), allocatable [ei](#)
i-opposite element
- integer *4, dimension(:), allocatable [ej](#)
j-opposite element
- integer *4, dimension(:), allocatable [ek](#)
k-opposite element
- integer *4, dimension(:), allocatable [si](#)
Opposite i-side.
- integer *4, dimension(:), allocatable [sj](#)
Opposite j-side.
- integer *4, dimension(:), allocatable [sk](#)
Opposite k-side.
- character(50) [geokind](#)
- character(50) [filename](#)

5.1.1 Detailed Description

Data type for a mesh.

5.1.2 Member Data Documentation

5.1.2.1 [ei](#)

```
integer*4, dimension(:), allocatable meshstructure::mesh::ei
```

i-opposite element

5.1.2.2 ej

`integer*4, dimension(:), allocatable meshstructure::mesh::ej`

j-opposite element

5.1.2.3 ek

`integer*4, dimension(:), allocatable meshstructure::mesh::ek`

k-opposite element

5.1.2.4 filename

`character(50) meshstructure::mesh::filename`

5.1.2.5 flagnode

`integer*4, dimension(:), allocatable meshstructure::mesh::flagnode`

boundary flag

5.1.2.6 geokind

`character(50) meshstructure::mesh::geokind`

5.1.2.7 gnode

`integer*4, dimension(:,:), allocatable meshstructure::mesh::gnode`

Global node.

5.1.2.8 mat

`integer*4, dimension(:), allocatable meshstructure::mesh::mat`

Element material kind.

5.1.2.9 nelem

`integer meshstructure::mesh::nelem`

Number of elements.

5.1.2.10 nelems

```
integer meshstructure::mesh::nelems
```

Number of elements.

5.1.2.11 nen

```
integer meshstructure::mesh::nen
```

Number of element's nodes.

5.1.2.12 nintp

```
integer meshstructure::mesh::nintp
```

Number of integration points.

5.1.2.13 nnodes

```
integer meshstructure::mesh::nnodes
```

Number of nodes.

5.1.2.14 nsd

```
integer meshstructure::mesh::nsd
```

Number of spatial.

5.1.2.15 numat

```
integer meshstructure::mesh::numat
```

Number of materials.

5.1.2.16 si

```
integer*4, dimension(:), allocatable meshstructure::mesh::si
```

Opposite i-side.

5.1.2.17 sj

```
integer*4, dimension(:), allocatable meshstructure::mesh::sj
```

Opposite j-side.

5.1.2.18 sk

```
integer*4, dimension(:), allocatable meshstructure::mesh::sk
```

Opposite k-side.

5.1.2.19 x

```
real*8, dimension(:,:), allocatable meshstructure::mesh::x
```

nodes coordinates

5.1.2.20 xv

```
real*8, dimension(:), allocatable meshstructure::mesh::xv
```

Circumcenter Elem xcoor.

5.1.2.21 yv

```
real*8, dimension(:), allocatable meshstructure::mesh::yv
```

Circumcenter Elem ycoor.

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

- [src/meshStructure.F90](#)

5.2 scalarstructure::scalarstructuresystem Type Reference

Variables and characteristic data for a scalar problem.

Collaboration diagram for scalarstructure::scalarstructuresystem:



Public Attributes

- real *8, dimension(:), allocatable [u](#)
Solution vector.
- real *8, dimension(:), allocatable [u_prev](#)
Previous time step solution vector.
- real *8, dimension(:), allocatable [u_prev_it](#)
Previous non-linear solution vector.
- real *8, dimension(:, :), allocatable [lhelem](#)
Element left-hand system.
- real *8, dimension(:), allocatable [rhelem](#)
Element right-hand system.
- real *8, dimension(:, :), allocatable [lhsys](#)
Global left-hand system.
- real *8, dimension(:), allocatable [rhsys](#)
Global right-hand system.
- real *8, dimension(:), allocatable [vbc](#)
BC values vector.
- real *8, dimension(:, :), allocatable [mat](#)
Material properties values.
- integer *4, dimension(:), allocatable [kbc](#)
BC kind.
- integer *4, dimension(:), allocatable [tprint](#)
- integer [transient](#)
Transient's flag.
- integer [nsteps](#)
Number of time steps.
- real *8 [dt](#)
Time step.
- integer [linflag](#)
Flag to indicate if problem is linear or not.

5.2.1 Detailed Description

Variables and characteristic data for a scalar problem.

5.2.2 Member Data Documentation

5.2.2.1 dt

```
real*8 scalarstructure::scalarstructuresystem::dt
```

Time step.

5.2.2.2 kbc

```
integer*4, dimension(:), allocatable scalarstructure::scalarstructuresystem::kbc
```

BC kind.

5.2.2.3 lhelem

```
real*8, dimension(:,:), allocatable scalarstructure::scalarstructuresystem::lhelem
```

Element left-hand system.

5.2.2.4 lhsys

```
real*8, dimension(:,:), allocatable scalarstructure::scalarstructuresystem::lhsys
```

Global left-hand system.

5.2.2.5 linflag

```
integer scalarstructure::scalarstructuresystem::linflag
```

Flag to indicate if problem is linear or not.

5.2.2.6 mat

```
real*8, dimension(:,:), allocatable scalarstructure::scalarstructuresystem::mat
```

Material properties values.

5.2.2.7 nsteps

```
integer scalarstructure::scalarstructuresystem::nsteps
```

Number of time steps.

5.2.2.8 rhelem

```
real*8, dimension(:), allocatable scalarstructure::scalarstructuresystem::rhelem
```

Element right-hand system.

5.2.2.9 rhsys

```
real*8, dimension(:), allocatable scalarstructure::scalarstructuresystem::rhsys
```

Global right-hand system.

5.2.2.10 tprint

```
integer*4, dimension(:), allocatable scalarstructure::scalarstructuresystem::tprint
```

5.2.2.11 transient

```
integer scalarstructure::scalarstructuresystem::transient
```

Transient's flag.

5.2.2.12 u

```
real*8, dimension(:), allocatable scalarstructure::scalarstructuresystem::u
```

Solution vector.

5.2.2.13 u_prev

```
real*8, dimension(:), allocatable scalarstructure::scalarstructuresystem::u_prev
```

Previous time step solution vector.

5.2.2.14 u_prev_it

```
real*8, dimension(:), allocatable scalarstructure::scalarstructuresystem::u_prev_it
```

Previous non-linear solution vector.

5.2.2.15 vbc

```
real*8, dimension(:), allocatable scalarstructure::scalarstructuresystem::vbc
```

BC values vector.

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

- [src/scalarStructure.F90](#)

Chapter 6

File Documentation

6.1 src/driver.F90 File Reference

Functions/Subroutines

- program [finel](#)

A FINite ELement program for general purpose problems. The present code is based in the book "Finite Elements: An Introduction" wrote by Eric Becker, Graham Carey and Tinsley Oden.

6.1.1 Function/Subroutine Documentation

6.1.1.1 finel()

```
program finel ( )
```

A FINite ELement program for general purpose problems. The present code is based in the book "Finite Elements: An Introduction" wrote by Eric Becker, Graham Carey and Tinsley Oden.

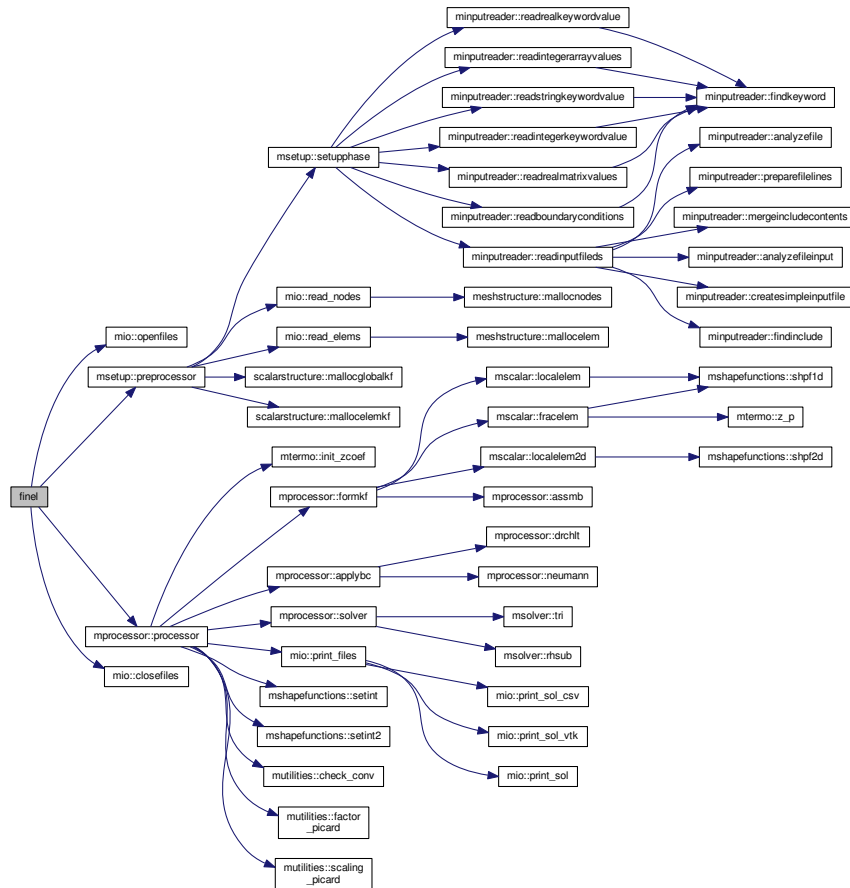
Due to the evolution of Fortran programming language, the code developed here incorporates several changes comparing to the original given in the book cited before. Modular paradigm was employed, as well a little of derived data structure.

Implementations by Diego T. Volpatto. email: volpatto@lncc.br or dtvolpatto@gmail.com

Author

Diego Tavares Volpatto

Here is the call graph for this function:



6.2 src/io.F90 File Reference

Modules

- module [mio](#)
Contains input/output variables and routines.

Functions/Subroutines

- subroutine [mio::openfiles](#) ()
Open IO files.
- subroutine [mio::closefiles](#) ()
Close IO files.
- subroutine [mio::read_nodes](#) (mesh_)
Subroutine to read node data file generated by EasyMesh.

- subroutine `mio::read_elems` (mesh_)
Subroutine to read element data file generated by EasyMesh.
- subroutine `mio::print_sol` (mesh_, scalar_, timestep)
Prints the solution of scalar field.
- subroutine `mio::print_sol_csv` (mesh_, scalar_, timestep)
Prints the solution of scalar field in the csv format aiming to compatibility with Paraview post-processing.
- subroutine `mio::print_sol_vtk` (mesh_, scalar_, timestep)
Prints the solution of scalar field in the vtk "legacy" format aiming to compatibility with Paraview post-processing.
- subroutine `mio::print_files` (mesh_, scalar_, timestep)
Record solution according to specified file kind in input.

Variables

- integer, parameter `mio::iin` = 1110
Input file id.
- integer, parameter `mio::iout` = 1120
Output file id.
- integer, parameter `mio::isol` = 1130
Solution file id.
- integer, parameter `mio::ioutn` = 1140
Node output file.
- integer, parameter `mio::ioute` = 1150
Element output file.
- integer, parameter `mio::isolcsv` = 1160
Solution csv file id.
- integer, parameter `mio::isolvtk` = 1170
Solution vtk file id.
- character(len=20), parameter `mio::infile` = 'input.dat'
Input file name.
- character(len=20), parameter `mio::outfile` = 'output.dat'
Output file name.
- character(len=20), parameter `mio::outnodes` = 'outnodes.dat'
Output node's file name.
- character(len=20), parameter `mio::outelem` = 'outelem.dat'
Output elements' file name.
- character(len=50), parameter `mio::solfile` = 'solution00'
Solution file name.
- character(len=50) `mio::title`
- character(len=50) `mio::file_format`

6.3 src/meshStructure.F90 File Reference

Data Types

- type `meshstructure::mesh`
Data type for a mesh.

Modules

- module [meshstructure](#)

Module that contains the data structure of a mesh associate to a problem.

Functions/Subroutines

- subroutine [meshstructure::mallocnodes](#) (meshStrct)
Routine that allocate memory to node data.
- subroutine [meshstructure::mallocelelem](#) (meshStrct)
Routine that allocate memory to element data.

6.4 src/mInputReader.F90 File Reference

Modules

- module [minputreader](#)

Modulo responsavel por reunir subrotinas para leitura do arquivo de entrada.

Functions/Subroutines

- subroutine [minputreader::readinputfileds](#) ()
Le arquivo de input e armazena seu conteudo em um array.
- subroutine [minputreader::createsimpleinputfile](#) ()
Cria a estrutura de input usando um arquivo de entrada sem includes.
- subroutine [minputreader::mergeincludecontents](#) (include_file, include_line)
Le o conteudo do arquivo de include e armazena no array principal.
- subroutine [minputreader::preparefilelines](#) (include_indexes, include_number_of_lines, number_of_includes, original_file_lines)
Efetua a alocação da estrutura definitiva, preparando a linha dos arquivos originais para receber os includes.
- subroutine [minputreader::analyzefileinput](#) (number_of_lines, number_of_includes)
Efetua algumas análises no arquivo recebido.
- subroutine [minputreader::analyzefile](#) (file_name, number_of_lines, number_of_includes)
Efetua algumas análises no arquivo recebido.
- integer *4 function [minputreader::findinclude](#) (position, file_lines, number_of_lines)
Procura a n-esima palavra-chave include.
- integer *4 function [minputreader::findkeyword](#) (keyword)
Procura uma palavra-chave.
- subroutine [minputreader::readintegerkeywordvalue](#) (keyword, target, default_value)
Efetua a leitura de uma palavra-chave to tipo inteiro. Se nao encontrado, associa o valor default fornecido.
- subroutine [minputreader::readintarraykeywordvalue](#) (keyword, target, default_value)
Efetua a leitura de uma palavra-chave do tipo array de inteiro. Se nao encontrado, associa o valor default fornecido.
Obs.: Atentar para o fato dessa sub-rotina ter um do "infinito".
- subroutine [minputreader::readstringkeywordvalue](#) (keyword, target, default_value)
Efetua a leitura de uma palavra-chave to tipo string. Se nao encontrado, associa o valor default fornecido.
- subroutine [minputreader::readrealkeywordvalue](#) (keyword, target, default_value)
Efetua a leitura de uma palavra-chave to tipo real. Se nao encontrado, associa o valor default fornecido.
- subroutine [minputreader::readintegerarrayvalues](#) (keyword, target, default_value)

Efetua a leitura de uma palavra-chave do tipo de um array de inteiros. A leitura eh realizada linha por linha. A primeira linha informa o numero de valores a ser lido. Se nao encontrada a keyword, associa o valor default fornecido.

- subroutine [minputreader::readrealmatrixvalues](#) (keyword, target, default_value)

Efetua a leitura de uma palavra-chave do tipo de um array bidimensional real. A leitura eh realizada linha por linha. Se nao encontrado, associa o valor default fornecido.

- subroutine [minputreader::readboundaryconditions](#) (keyword, kbc, vbc, default_value)

Efetua a leitura de uma palavra-chave do tipo de um array bidimensional real. A leitura eh realizada linha por linha. Se nao encontrado, associa o valor default fornecido.

Variables

- character(len=200), dimension(:), allocatable [minputreader::file_lines](#)

Armazena as linhas do arquivo de input.

- integer *4 [minputreader::number_of_lines](#)

Armazena o numero de linhas no arquivo.

6.5 src/processor.F90 File Reference

Modules

- module [mprocessor](#)

Processor module to compute, assemble and solve the system.

Functions/Subroutines

- subroutine [mprocessor::formkf](#) (mesh_, scalar_, t)

Form and assemble $Ku = F$ system.

- subroutine [mprocessor::assmb](#) (mesh_, scalar_, nel)

Assemble element stiffness matrix and load vector to global stiffness matrix and load vector, respectively.

- subroutine [mprocessor::drchlt](#) (mesh_, scalar_, n)

Apply Dirichlet Boundary Condition.

- subroutine [mprocessor::neumann](#) (mesh_, scalar_, n)

Apply Neumann Boundary Condition – 1D. Prescribe $-k(x)u = vbc$.

- subroutine [mprocessor::applybc](#) (mesh_, scalar_)

Modify $Ku=F$ system to incorporate BC data.

- subroutine [mprocessor::solver](#) (mesh_, scalar_)

Executes Gauss reduction and forward substitution solving $Ku=F$.

- subroutine [mprocessor::processor](#) (mesh_, scalar_)

Processor routine phase.

6.6 src/scalar.F90 File Reference

Modules

- module [mscalar](#)

Contains variables and subroutine related to a general scalar problem.

Functions/Subroutines

- subroutine `mscalar::localelem` (mesh_, scalar_, nel)
Computes a master element contribution – 1D.
- subroutine `mscalar::localelem2d` (mesh_, scalar_, nel)
Computes a master element contribution – 2D.
- subroutine `mscalar::fracelem` (mesh_, scalar_, nel, t)
Computes a master element contribution in fracture case – 1D.

6.7 src/scalarStructure.F90 File Reference

Data Types

- type `scalarstructure::scalarstructuresystem`
Variables and characteristic data for a scalar problem.

Modules

- module `scalarstructure`
Module that contains the data structure of a general scalar problem.

Functions/Subroutines

- subroutine `scalarstructure::mallocglobalkf` (scalar_, n)
Routine to allocate and clear the $Ku = F$ system.
- subroutine `scalarstructure::mallocelemkf` (scalar_, n)
Routine to allocate and clear the element KF .

6.8 src/setup.F90 File Reference

Modules

- module `msetup`
Module for setup phase by IO procedures.

Functions/Subroutines

- subroutine `msetup::setupphase` (mesh_, scalar_)
Reads parameters from input file.
- subroutine `msetup::preprocessor` (mesh_, scalar_)
Realizes preprocessor routines.

6.9 src/shapeFunctions.F90 File Reference

Modules

- module [mshapefunctions](#)
Module for shape functions computations and relate operations.

Functions/Subroutines

- subroutine [mshapefunctions::setint](#)
Gauss quadrature data set routine - 1D.
- subroutine [mshapefunctions::setint2](#)
Gauss quadrature data set routine - 2D.
- subroutine [mshapefunctions::shpf1d](#) (xl, n, psi, dpsl)
Calculates the values of the shape functions and their derivatives - 1D.
- subroutine [mshapefunctions::shpf2d](#) (XL, N, PSI, DPSI)
Calculates the values of the shape functions and their derivatives - 2D.

Variables

- real *8, dimension(20, 20) [mshapefunctions::xi](#)
Gauss point integration 1D.
- real *8, dimension(20, 20) [mshapefunctions::w](#)
Gauss weights 1D.
- real *8, dimension(2, 9) [mshapefunctions::xiq](#)
Gauss points to rectangular element.
- real *8, dimension(9) [mshapefunctions::wq](#)
Gauss weights to rectangular element.
- real *8, dimension(2, 4) [mshapefunctions::xit](#)
Gauss points to triangle element.
- real *8, dimension(4) [mshapefunctions::wt](#)
Gauss weights to triangle element.
- integer [mshapefunctions::quadext](#)

6.10 src/solver.F90 File Reference

Modules

- module [msolver](#)
Contains subroutine to compute numerical solution of linear systems $Ax=b$.

Functions/Subroutines

- subroutine [msolver::tri](#) (A, n)
Applies Gauss reduction in $A(n,n)$ to obtain a superior triangular equivalent form.
- subroutine [msolver::rsub](#) (A, x, b, n)
Does the forward substitution on the right-side-hand.

6.11 src/termo.F90 File Reference

Modules

- module `mtermo`
Module that contains thermodynamics function and parameters.

Functions/Subroutines

- subroutine `mtermo::init_zcoef`
Initializes interpolation coefficients to compute compressibility factor.
- subroutine `mtermo::z_p` (p, Z)
Computes compressibility factor.

Variables

- real *8, dimension(10) `mtermo::zcoef`

6.12 src/utilities.F90 File Reference

Modules

- module `mutilities`
Module for auxiliar routines.

Functions/Subroutines

- subroutine `mutilities::linspace` (x1, x2, nintv, x)
Generate points between x1 and x2 equally spaced in x(i). Same idea of numpy subroutine.
- real *8 function `mutilities::f1` (x)
A function to test purpose.
- subroutine `mutilities::quad1` (n, x1, x2)
Subroutine that computes gaussian quadrature of f1.
- subroutine `mutilities::test_shpf1d` (n, nelem, x)
Check if shpf1d works properly.
- subroutine `mutilities::print_matrix` (A, n, m)
Prints in the screen a matrix A(n,m)
- subroutine `mutilities::check_conv` (u, uprev, nnodes, tol, norm, flag)
Routine to check if non-linear iteration converged.
- subroutine `mutilities::factor_picard` (alpha, delta, eps, omega_min, omega)
Computes underrelaxation factor for Picard iteration.
- subroutine `mutilities::scaling_picard` (i, delta, deltap, eps, rho, omega, omega_min, alpha)
Rescaling shape factor for underrelaxation.

Index

analyzefile
 minputreader, 10
analyzefileinput
 minputreader, 10
applybc
 mprocessor, 29
assmb
 mprocessor, 30

check_conv
 mutilities, 48
closefiles
 mio, 21
createsimpleinputfile
 minputreader, 10

drchlt
 mprocessor, 31
driver.F90
 finel, 63
dt
 scalarstructure::scalarstructuresystem, 60

ei
 meshstructure::mesh, 56
ej
 meshstructure::mesh, 56
ek
 meshstructure::mesh, 57

f1
 mutilities, 48
factor_picard
 mutilities, 49
file_format
 mio, 27
file_lines
 minputreader, 20
filename
 meshstructure::mesh, 57
findinclude
 minputreader, 12
findkeyword
 minputreader, 12
finel
 driver.F90, 63
flagnode
 meshstructure::mesh, 57
formkf
 mprocessor, 31

fracelem
 mscalar, 36

geokind
 meshstructure::mesh, 57
gnode
 meshstructure::mesh, 57

iin
 mio, 27
infile
 mio, 27
init_zcoef
 mtermo, 46
iout
 mio, 27
ioute
 mio, 27
ioutn
 mio, 27
isol
 mio, 27
isolcsv
 mio, 27
isolvtk
 mio, 27

kbc
 scalarstructure::scalarstructuresystem, 60

lhelem
 scalarstructure::scalarstructuresystem, 60
lhsys
 scalarstructure::scalarstructuresystem, 61
linflag
 scalarstructure::scalarstructuresystem, 61
linspace
 mutilities, 49
localelem
 mscalar, 36
localelem2d
 mscalar, 37

mallocelem
 meshstructure, 7
mallocelemkf
 scalarstructure, 53
mallocglobalkf
 scalarstructure, 53
mallocnodes
 meshstructure, 8

- mat
 - meshstructure::mesh, [57](#)
 - scalarstructure::scalarstructuresystem, [61](#)
- mergeincludecontents
 - minputreader, [13](#)
- meshstructure, [7](#)
 - mallocelelem, [7](#)
 - mallocnodes, [8](#)
- meshstructure::mesh, [55](#)
 - ei, [56](#)
 - ej, [56](#)
 - ek, [57](#)
 - filename, [57](#)
 - flagnode, [57](#)
 - geokind, [57](#)
 - gnode, [57](#)
 - mat, [57](#)
 - nelem, [57](#)
 - nelems, [57](#)
 - nen, [58](#)
 - nintp, [58](#)
 - nnodes, [58](#)
 - nsd, [58](#)
 - numat, [58](#)
 - si, [58](#)
 - sj, [58](#)
 - sk, [58](#)
 - x, [59](#)
 - xv, [59](#)
 - yv, [59](#)
- minputreader, [8](#)
 - analyzefile, [10](#)
 - analyzefileinput, [10](#)
 - createsimpleinputfile, [10](#)
 - file_lines, [20](#)
 - findinclude, [12](#)
 - findkeyword, [12](#)
 - mergeincludecontents, [13](#)
 - number_of_lines, [20](#)
 - preparefilelines, [13](#)
 - readboundaryconditions, [14](#)
 - readinputfileds, [15](#)
 - readintarraykeywordvalue, [15](#)
 - readintegerarrayvalues, [16](#)
 - readintegerkeywordvalue, [17](#)
 - readrealkeywordvalue, [18](#)
 - readrealmatrixvalues, [18](#)
 - readstringkeywordvalue, [19](#)
- mio, [20](#)
 - closefiles, [21](#)
 - file_format, [27](#)
 - iin, [27](#)
 - infile, [27](#)
 - iout, [27](#)
 - ioute, [27](#)
 - ioutn, [27](#)
 - isol, [27](#)
 - isolcsv, [27](#)
 - isolvtk, [27](#)
 - openfiles, [22](#)
 - outelem, [28](#)
 - outfile, [28](#)
 - outnodes, [28](#)
 - print_files, [22](#)
 - print_sol, [23](#)
 - print_sol_csv, [24](#)
 - print_sol_vtk, [24](#)
 - read_elems, [25](#)
 - read_nodes, [26](#)
 - solfile, [28](#)
 - title, [28](#)
- mprocessor, [28](#)
 - applybc, [29](#)
 - assmb, [30](#)
 - drchlt, [31](#)
 - formkf, [31](#)
 - neumann, [32](#)
 - processor, [33](#)
 - solver, [34](#)
- mscalar, [35](#)
 - fracelem, [36](#)
 - localelem, [36](#)
 - localelem2d, [37](#)
- msetup, [38](#)
 - preprocessor, [39](#)
 - setupphase, [40](#)
- mshapefunctions, [40](#)
 - quadext, [43](#)
 - setint, [41](#)
 - setint2, [41](#)
 - shpf1d, [42](#)
 - shpf2d, [42](#)
 - w, [43](#)
 - wq, [43](#)
 - wt, [43](#)
 - xi, [44](#)
 - xiq, [44](#)
 - xit, [44](#)
- msolver, [44](#)
 - rhsb, [44](#)
 - tri, [45](#)
- mtermo, [45](#)
 - init_zcoef, [46](#)
 - z_p, [46](#)
 - zcoef, [47](#)
- mutilities, [47](#)
 - check_conv, [48](#)
 - f1, [48](#)
 - factor_picard, [49](#)
 - linspace, [49](#)
 - print_matrix, [50](#)
 - quad1, [50](#)
 - scaling_picard, [51](#)
 - test_shpf1d, [51](#)
- nelem
 - meshstructure::mesh, [57](#)

- nelems
 - meshstructure::mesh, [57](#)
- nen
 - meshstructure::mesh, [58](#)
- neumann
 - mprocessor, [32](#)
- nintp
 - meshstructure::mesh, [58](#)
- nnodes
 - meshstructure::mesh, [58](#)
- nsd
 - meshstructure::mesh, [58](#)
- nsteps
 - scalarstructure::scalarstructuresystem, [61](#)
- numat
 - meshstructure::mesh, [58](#)
- number_of_lines
 - minputreader, [20](#)
- openfiles
 - mio, [22](#)
- outelem
 - mio, [28](#)
- outfile
 - mio, [28](#)
- outnodes
 - mio, [28](#)
- preparefilelines
 - minputreader, [13](#)
- preprocessor
 - msetup, [39](#)
- print_files
 - mio, [22](#)
- print_matrix
 - mutilities, [50](#)
- print_sol
 - mio, [23](#)
- print_sol_csv
 - mio, [24](#)
- print_sol_vtk
 - mio, [24](#)
- processor
 - mprocessor, [33](#)
- quad1
 - mutilities, [50](#)
- quadext
 - mshapefunctions, [43](#)
- read_elems
 - mio, [25](#)
- read_nodes
 - mio, [26](#)
- readboundaryconditions
 - minputreader, [14](#)
- readinputfiles
 - minputreader, [15](#)
- readintarraykeywordvalue
 - minputreader, [15](#)
- readintegerarrayvalues
 - minputreader, [16](#)
- readintegerkeywordvalue
 - minputreader, [17](#)
- readrealkeywordvalue
 - minputreader, [18](#)
- readrealmatrixvalues
 - minputreader, [18](#)
- readstringkeywordvalue
 - minputreader, [19](#)
- rhelem
 - scalarstructure::scalarstructuresystem, [61](#)
- rhsb
 - msolver, [44](#)
- rhsys
 - scalarstructure::scalarstructuresystem, [61](#)
- scalarstructure, [52](#)
 - mallocelembf, [53](#)
 - mallocglobalkf, [53](#)
- scalarstructure::scalarstructuresystem, [59](#)
 - dt, [60](#)
 - kbc, [60](#)
 - lhelem, [60](#)
 - lhsys, [61](#)
 - linflag, [61](#)
 - mat, [61](#)
 - nsteps, [61](#)
 - rhelem, [61](#)
 - rhsys, [61](#)
 - tprint, [61](#)
 - transient, [61](#)
 - u, [62](#)
 - u_prev, [62](#)
 - u_prev_it, [62](#)
 - vbc, [62](#)
- scaling_picard
 - mutilities, [51](#)
- setint
 - mshapefunctions, [41](#)
- setint2
 - mshapefunctions, [41](#)
- setupphase
 - msetup, [40](#)
- shpf1d
 - mshapefunctions, [42](#)
- shpf2d
 - mshapefunctions, [42](#)
- si
 - meshstructure::mesh, [58](#)
- sj
 - meshstructure::mesh, [58](#)
- sk
 - meshstructure::mesh, [58](#)
- solfile
 - mio, [28](#)
- solver
 - mprocessor, [34](#)

- src/driver.F90, [63](#)
- src/io.F90, [64](#)
- src/mInputReader.F90, [66](#)
- src/meshStructure.F90, [65](#)
- src/processor.F90, [67](#)
- src/scalar.F90, [67](#)
- src/scalarStructure.F90, [68](#)
- src/setup.F90, [68](#)
- src/shapeFunctions.F90, [69](#)
- src/solver.F90, [69](#)
- src/termo.F90, [70](#)
- src/utilities.F90, [70](#)

- test_shpf1d
 - mutilities, [51](#)
- title
 - mio, [28](#)
- tprint
 - scalarstructure::scalarstructuresystem, [61](#)
- transient
 - scalarstructure::scalarstructuresystem, [61](#)
- tri
 - msolver, [45](#)

- u
 - scalarstructure::scalarstructuresystem, [62](#)
- u_prev
 - scalarstructure::scalarstructuresystem, [62](#)
- u_prev_it
 - scalarstructure::scalarstructuresystem, [62](#)

- vbc
 - scalarstructure::scalarstructuresystem, [62](#)

- w
 - mshapefunctions, [43](#)
- wq
 - mshapefunctions, [43](#)
- wt
 - mshapefunctions, [43](#)

- x
 - meshstructure::mesh, [59](#)
- xi
 - mshapefunctions, [44](#)
- xiq
 - mshapefunctions, [44](#)
- xit
 - mshapefunctions, [44](#)
- xv
 - meshstructure::mesh, [59](#)

- yv
 - meshstructure::mesh, [59](#)

- z_p
 - mtermo, [46](#)
- zcoef
 - mtermo, [47](#)