

FEM - Optimization Module

and

SDESIGN

User Guides

Version 0.2

Kurt Maute

Michael Rauli

May 18, 2004

Contents

1	<i>FEM</i> - Optimization Module	1
1.1	Introduction	1
1.2	Input files	3
1.3	Structure of input data	4
1.4	Syntax of input data	5
1.4.1	General	5
1.4.2	Defining optimization criteria	5
1.4.3	Defining abstract optimization variables	9
1.4.4	Defining objective function	9
1.4.5	Defining constraints	10
1.4.6	Defining structural variables	10
1.4.7	Defining functions	12
1.4.8	Defining solution strategy	14
1.4.9	Defining explicit optimization variable – finite element relations	17
1.4.10	Defining structural analysis parameters	18
2	<i>FEM</i> - Reliability Module	21
2.1	Introduction	21
2.2	Input files	22
2.3	Structure of input data	23
2.4	Syntax of input data	24
2.4.1	General	24
2.4.2	Defining reliability criteria	24
2.4.3	Defining abstract random variables	28
2.4.4	Defining failure modes	28
2.4.5	Defining structural variations	29
2.4.6	Defining functions	30
2.4.7	Defining solution strategy	32

2.4.8	Defining explicit random variation – finite element relations	35
2.4.9	Defining structural analysis parameters	36
2.5	<i>FEM</i> Reliability example	37
2.5.1	2-Dimensional Cantilever	37
3	<i>SDESIGN</i> - Geometric Modeler	43
3.1	File names and command line options	44
3.2	Structure and syntax	46
3.2.1	General	46
3.2.2	Defining Optional Output Files for <i>SDESIGN</i>	50
3.2.3	Defining Output Files for <i>FEM</i>	50
3.2.4	Defining Inputs for <i>FEM</i>	52
3.2.5	Defining Inputs for <i>FEM</i> Optimization Module	52
3.2.6	Defining Input Variables from a File	53
3.2.7	Defining Accuracy, Iteration, Tolerance	53
3.2.8	Defining Variables	54
3.2.9	Defining Design Element Generation Order	55
3.2.10	Defining Duplicated Node Checking Strategy	55
3.2.11	Defining Node Scaling	56
3.2.12	Defining Nodes	56
3.2.13	Defining Edges	56
3.2.14	Defining Trusses	57
3.2.15	Defining Patches	58
3.2.16	Defining Volumes	61
3.2.17	Defining Attributes	62
3.2.18	Defining Search-boxes	63
3.2.19	Defining Match-boxes	64
3.2.20	Defining Boundary Conditions	64
3.2.21	Defining Structural Interfaces	67
3.2.22	Defining Mesh Motion Flags	68
3.2.23	Defining Variable Control Nodes	69
3.2.24	Defining Variable Element Properties	70
3.2.25	Defining Variable Node Properties	72
3.2.26	Defining Link Macros	73
3.2.27	Defining Initial Node and Element Numbers	74
3.2.28	Using a Pre-existing Fe-mesh	74
3.2.29	Using a Pre-existing Local Coordinate Mesh	74
3.3	Projecting a given fe-mesh onto design element mesh	75

4	Electro Mechanical Analysis, Optimization and Eigenvalues	79
4.1	Introduction	79
4.2	Structural Component	80
4.2.1	Analysis	80
4.2.2	Optimization	81
4.2.3	Eigenvalue	81
4.2.4	POD Decomposition	81
4.3	Electrostatic Component	82
4.3.1	Analysis	82
4.3.2	Optimization	85
4.3.3	Eigenvalue	85
4.3.4	POD Decomposition	85
4.4	Mesh-motion Component	86
4.4.1	Analysis	86
4.4.2	Optimization	87
4.4.3	Eigenvalue	87
4.4.4	POD Decomposition	88
4.5	Matching	88
4.6	Examples	88
4.6.1	Structural Input File	88
4.6.2	Electrostatic Input File	91
4.6.3	Mesh-motion Input File	93
4.6.4	Syntax for running problem	95
5	Examples	97
5.1	<i>FEM</i> Optimizatopn examples	97
5.1.1	2-Dimensional Cantilever	97
5.1.2	3-Dimensional Cantilever	101
5.2	<i>SDESIGN</i> Examples	106
5.2.1	FE-Mesh Generation using a Patch Design Element .	106
5.2.2	FE-Mesh Generation using a Volume Design Element	107
5.2.3	FE-Mesh Projection onto a Volume Design Element .	112

List of Figures

2.1	Example 1 - 2-Dimensional Cantilevered Beam	37
3.1	Edge definitions	57
3.2	Node input order for 16 node Bezier patch	58
3.3	Definition of orientation of triangular elements	59
5.1	Example 1 - 2-Dimensional Cantilever	97
5.2	Example 2 - 3-Dimensional Cantilever	102

List of Tables

2.1	Uncertainties for reliability analysis	37
3.1	Command line options	45
3.2	<i>SDESIGN</i> output files	46
3.3	Supported semantic blocks for <i>FEM</i>	47
3.4	Supported semantic blocks for optimization module	48
3.5	Available semantic blocks for <i>SDESIGN</i>	49
3.6	<i>SDESIGN</i> output file options	51
3.7	Design element type keywords and numbers	55
3.8	Types of 1-d finite elements available	57
3.9	Types of 2-d finite elements available	59
3.10	Available geometry types for boundary conditions	65
3.11	Boundary conditions for matching and wet edges and surfaces	68

Chapter 1

FEM - Optimization Module

1.1 Introduction

In general structural optimization means to minimize an objective function f with equality constraints \mathbf{h} and inequality constraints \mathbf{g} by varying the optimization variables \mathbf{s} ¹. The range of \mathbf{s} is restricted by lower and upper bounds $\underline{\mathbf{s}}$ and $\bar{\mathbf{s}}$.

$$\min_{\mathbf{s}} f(\mathbf{s}) \quad \mathbf{s} \in \mathbf{R}^{n_s} \quad (1.1)$$

$$\mathbf{h}(\mathbf{s}) = \mathbf{0} \quad \mathbf{h} \in \mathbf{R}^{n_h} \quad (1.2)$$

$$\mathbf{g}(\mathbf{s}) \geq \mathbf{0} \quad \mathbf{g} \in \mathbf{R}^{n_g} \quad (1.3)$$

$$\underline{\mathbf{s}} \leq \mathbf{s} \leq \bar{\mathbf{s}} \quad (1.4)$$

The number of optimization variables is denoted by n_s , the number of equality constraints by n_h and the number of inequality constraints by n_g .

Objective and constraints are in general composed of explicit functions of the optimization variables \mathbf{s} and so-called optimization criteria \mathbf{q} , like structural weight, strain energy etc. The abstract optimization variables \mathbf{s} are linked to so-called structural variables \mathbf{c} , like thickness, Young's modulus or parameters defining the shape of the structure. Or in other words the structural variables \mathbf{c} , are functions of the abstract optimization variables \mathbf{s}

¹The most dominant equilibrium constraints are the mechanical equilibrium conditions. Since the mechanical equilibrium is explicitly satisfied by analyzing the structure for each design by a finite element analysis during the optimization process, they are not explicitly specified.

². The optimization criteria \mathbf{q} directly depend on the structural variables \mathbf{c} .

$$f = f(\mathbf{s}, \mathbf{q}(\mathbf{c})) \quad (1.5)$$

$$\mathbf{h} = \mathbf{h}(\mathbf{s}, \mathbf{q}(\mathbf{c})) \quad (1.6)$$

$$\mathbf{g} = \mathbf{g}(\mathbf{s}, \mathbf{q}(\mathbf{c})) \quad (1.7)$$

with $\mathbf{c} = \mathbf{c}(\mathbf{s})$ and $\mathbf{q} \in \mathbf{R}^{n_q}$, where n_q is the number of design criteria. In general the design criteria \mathbf{q} are functions of the discretized state variables \mathbf{u} , like displacements, strains, stress, eigenfrequencies etc., which are in turn functions of the structural variables \mathbf{c} . Therefore, we write for the sake of completeness:

$$\mathbf{q} = \mathbf{q}(\mathbf{c}, \mathbf{u}(\mathbf{c})) \quad (1.8)$$

The in general nonlinear optimization problem can be solved by diverse iterative methods, like nonlinear programming or genetic and evolutionary algorithms. For some problems it may be efficient to apply subsequently different methods. For example, a robust evolutionary strategy determines roughly the position of the optimum. The final approach is done by a mathematical program method with a high convergence rate near the optimum.

Since the problems mainly arise in structural optimization are sufficiently smooth, gradient based optimization methods can be efficiently applied. The gradients of objective and constraints, i.e. of the design criteria $d\mathbf{q}/ds$, are determined by a sensitivity analysis. The gradients can be split up according to 1.5 - 1.7 into:

$$\frac{d\mathbf{q}}{ds} = \frac{\partial \mathbf{q}}{\partial \mathbf{s}} + \frac{\tilde{d}\mathbf{q}}{\tilde{d}\mathbf{c}} \frac{d\mathbf{c}}{ds} \quad (1.9)$$

where the derivatives of the structural variables with respect to the abstract variables $d\mathbf{c}/ds$ are called design velocities. Since the structural variables are most often explicit functions of the abstract variables, determining the design velocities is straight forward.

In order to stress the crucial point determining the derivatives of the design criteria with respect to the structural variables, once again this term is split up into:

$$\frac{\tilde{d}\mathbf{q}}{\tilde{d}\mathbf{c}} = \frac{\partial \mathbf{q}}{\partial \mathbf{c}} + \frac{\partial \mathbf{q}}{\partial \mathbf{u}} \frac{d\mathbf{u}}{d\mathbf{c}} \quad (1.10)$$

²For shape optimization the position of fe-nodes has to be defined as function of abstract optimization variables. In the appendix a simple geometric modeling program *SDESIGN* is described which generates theses functions based on a geometrical description of the structure.

Consequently, the derivatives of the design criteria with respect to the structural variables requires the derivatives of the state variables \mathbf{u} with respect to the structural variables \mathbf{c} . The derivations of the mechanical equilibrium conditions lead to the governing equations determining $d\mathbf{u}/d\mathbf{c}$.

There are diverse approaches to determine the derivatives of the design criteria with respect to the structural variables and the abstract variables, respectively. The most important approaches are:

1. Numerical sensitivity analysis: approximation of the derivatives $q\mathbf{u}/ds$ by means of finite differences.
2. Analytical sensitivity analysis: analytical derivation of the gradient terms based on the variational or discretized mechanical equilibrium conditions. Depending on the number of optimization variables n_s and design criteria n_q different approaches are slightly more efficient:
 - $n_s \leq n_q$ - direct method
 - $n_s \geq n_q$ - adjoint method

According to this short overview of modeling a structural optimization problem, the input for the optimization module has to be prepared. Subsequently, the structure and the syntax of the input is described and illustrated by some examples.

1.2 Input files

The initial structure is described in the main *FEM* input file as usual. For the sake of clarity the optimization problem is described in an own file. In order to run a structure optimization process, the keyword **STRUCTOPT** has to be added in the *FEM* input file, including the name of the optimization input file.

```
variant a: ...  
    STRUCTOPT <file>
```

```
variant b: ...  
    STRUCTOPT  
    <file>
```

For ease of creating the FEM input file, it can be completely created with *SDESIGN*. This can only be done if a new mesh is being created by *SDESIGN*, rather than pre-existing nodes being read.

The optimization process generates two files which contain information of the given problem and essential iteration data (opt-file) and a protocol of the different optimization algorithms (nlp-file). The name of the files is composed of the name of the input file and the suffix for the related file. An example:

```

optimization input file  example.oinp
opt-file                 example.opt
nlp-file                 example.nlp

```

Remark: In addition to the standard output file of *FEM*, like displacements, stress etc., different structural quantities can be plotted. The related keywords specifying the output type and file names are given in the *FEM* input file:

type	keyword
Thickness	THICKNES <file name>
Young's modulus	YMODULUS <file name>
Material density	MDENSITY <file name>
Shape (as attribute)	SHAPEATT <file name>
Shape (as load case)	SHAPESTC <file name>
Orientation of fibers (composite)	COMPOSIT <file name>

The fiber orientation is illustrated by crosses where the length of the bars indicates the values of the Young's modulus in the related direction. The pseudo structure for the crosses is in a file called <file name identifier>.cross, i.e. if the composite file name is e.g. out.comp the file name of the pseudo structure is out.cross.

1.3 Structure of input data

The optimization input is subdivided into different semantical blocks which are indicated by keywords:

semantic block	keyword
optimization criteria \mathbf{q}	CRITERIA
abstract optimization variables \mathbf{s}	ABSVAR
objective f	OBJECTIVE
constraints \mathbf{h} and \mathbf{g}	CONSTRAINT
structural optimization variables \mathbf{c}	STCVAR
solution strategies	SOLVER
structural analysis parameters	STCANALYSIS
end of file	END

As mentioned above objective and constraints are in general explicit functions of the abstract optimization variables and the design criteria. Analogously, structural variables are explicit functions of the abstract variables. These functional relations have to be defined in the optimization input. For this some kind of simple programming language is provided. The syntax is subsequently explained.

1.4 Syntax of input data

1.4.1 General

The following rules apply:

1. No blank lines are allowed.
2. Only capital letters are allowed.
3. Comment lines are mark by # in the first column.
4. No additional data is allowed in lines with keywords marking a semantic block.
5. The order of the semantic blocks is not arbitrary. You can use only optimization. criteria or optimization variables to build functions which are already defined.
6. The end of the input data is marked by the keyword END

1.4.2 Defining optimization criteria

CRITERIA <num> <crit> <nod/frq/dir> <disp/str/strvol-data> <time/loadcase> <gen>

- **<num>** User-defined id-number of optimization criteria. No gaps are allowed.
- **<crit>** Type of optimization criteria.

type of criteria	keyword	required data
structural mass	MASS	
moment of inertia	INERTIA	axis (RX,RY,RZ only)
strain energy	ENERGY	
kinetic energy	KINETIC	
dissipated energy	DAMPING	
control power	CTRLCOST	
stress volume	STRVOL	strvol-data
eigenfrequency	FRQ	id-number of eigenfrequency
nodal stress	STRESS	node id-number, stress type
nodal displacement	DISP	node id-number, displacement type
displacement level	DISLEVEL	displevel-data
internal force	INTFORCE	node id-number, displacement type
aeroelastic forces	AERO	direction
aeroelastic moments	MOMAER	rotation axis
sonic boom	SBOOM	
failure probability	FAILPROB	id-number of limit state function
reliability index	RELINDEX	id-number of limit state function
performance measure	PERFORM	id-number of limit state function

- If the mass or the strain energy of a subset of elements should be evaluated a list of the respective elemental id-numbers can be specified as follows:

{ <id-number> }

or

```
{
<id-number>
}
```

or any combination of the above alternatives.

- **<nod/frq/dir>** Id-number of FE-node, eigenfrequency or direction of aeroelastic force.

direction	keyword
x-direction	1
y-direction	2
z-direction	3

- `<disp/str>` Type of displacement or stress.

type of displacement	keyword
translation in x-direction	DX or 1
translation in y-direction	DY or 2
translation in z-direction	DZ or 3
rotation around x-axis	RX or 4
rotation around y-axis	RY or 5
rotation around z-axis	RZ or 6

type of nodal stress	keyword
von Mises - lower surface	VML
von Mises - middle surface	VMM
von Mises - upper surface	VMU
1. principal - lower surface	S1L
1. principal - middle surface	S1M
1. principal - upper surface	S1U
2. principal - lower surface	S2L
2. principal - middle surface	S2M
2. principal - upper surface	S2U

- `<strvol-data>` Definition of stress-volume function. The input of the `<strvol-data>` is

`<stress type> <reference stress> <exponent> <volume flag> ...`
`<NODAL/ELEMENTAL> { <list of elements/nodes> }`

The `<reference stress>` ($\bar{\sigma}$) and the `<exponent>` (p) are defined as follows:

$$\text{volumeflag}=1 \quad \text{strvol} = \left[\frac{1}{\|\Omega\|} \int_{\Omega} \left(\frac{\sigma_i}{\bar{\sigma}} \right)^p d\Omega \right]^{1/p}$$

$$\text{volumeflag}=0 \quad \text{strvol} = \left[\sum \left(\frac{\sigma_i}{\bar{\sigma}} \right)^p \right]^{1/p}$$

where $\|\Omega\|$ is the volume of the finite element domain; σ_i is either the elemental or the nodal stress of interest; $\bar{\sigma}$ a reference stress; p a exponent. The key words <NODAL/ELEMENTAL> indicate whether elemental or nodal stresses should be used. In the case of nodal stresses, the number of nodes rather than the volume of the elements is used to define the reference volume $\|\Omega\|$. This function can be used to extract the maximum/minimum stress in the finite element domain by using a large even/odd exponent.

• <displevel-data> Definition of displacement leveling function. This function is formulated as follows:

$$\begin{aligned} \text{node number flag=1 } dislevel &= \left[\frac{1}{N} \sum_N \left(\frac{u_i^j - u_i^{ref}}{\bar{u}_i} \right)^p \right]^{1/p} \\ \text{node number flag=0 } dislevel &= \left[\sum_N \left(\frac{u_i^j - u_i^{ref}}{\bar{u}_i} \right)^p \right]^{1/p} \end{aligned}$$

where N is the number of term to be considered, u_i^j the degree of freedom j of node i , u_i^{ref} a reference value, \bar{u}_i a scaling factor, and p is an exponent. The reference value can either be given or the average of the $\sum u_i^j/N$ can be used. The input of the <displevel-data> is

```
<average flag> <node number flag> <exponent> ...
{ <list of summand definitions> }
```

where the list of summands needs to be specified as follows:

```
<node-id> <displacement type> <reference value> <scaling factor>
```

If the averaging flag is turned on (average flag = 1) then the reference value is ignored.

- For reliability related criteria, the id-number of the limit state function corresponds to the order specified in the reliability input file.
- <time/loadcase> Time / loadcase to evaluate criteria.

keyword	data
TIME	<time for evaluating criteria>
LCASE	<loadcase for evaluating criteria>

If no time is specified, the criteria is evaluated at the end of the structural

analysis. If no loadcase is specified, the criteria is evaluated for loadcase #1.

- `<gen>` Generating optimization criteria.

```
GEN ( <first criteria>,<last criteria>,<step size> )
```

In the generation loop new criteria are defined interpolating the criteria quantities between the first and last criteria. The step size of the loop can be specified. If the step size is not specified, step size 1 is assumed.

1.4.3 Defining abstract optimization variables

```
ABSVAR  
<num> <ival> <scl> <low> <upp> <gen>
```

- `<num>` User-defined id-number of abstract variable. No gaps are allowed.
- `<ival>` Initial value of optimization variable.
- `<scl>` Scaling factor. Variable scaling gets important when range and influence of variables considerably differ, e.g. combined shape optimization and sizing. If no value is specified, scaling factor is set to 1.0.
- `<low>` Lower bound of optimization variable. If no value is specified, lower bound is set to -10^{10} .
- `<upp>` Upper bound of optimization variable. If no value is specified, lower bound is set to 10^{10} . • `<gen>` Generating optimization criteria.

```
GEN ( <first variable>,<last variable>,<step size> )
```

In the generation loop new abstract variables are defined interpolating the variable quantities between the first and last variable. The step size of the loop can be specified. If the step size is not specified, step size 1 is assumed.

1.4.4 Defining objective function

```
OBJECTIVE  
<scl> * <func>
```

- `<sc1>` Scaling factor for objective function. Depending on the applied optimization algorithm this parameter is a powerful tool to trim the convergence of the optimization process, especially in the first few optimization steps.

- `<func>` Function of abstract variables and optimization criteria defining the objective (\rightarrow 1.4.7). This way multi-criteria optimization problems can be formulated based on scalar projection approaches.

Remark: By definition the optimization problem is formulated as a minimization problem. In order to define a maximization problem, use a negative scaling factor.

1.4.5 Defining constraints

```
CONSTRAINT
<num> <typ> <sc1> * <func>
```

- `<num>` User-defined id-number of constraint. No gaps are allowed.

- `<typ>` Type of constraint.

type of constraint	definition	keyword
equality constraint	$\mathbf{h}_j = 0$	EQL
inequality constraint	$\mathbf{g}_j \geq 0$	IEQ

- `<sc1>` Scaling factor for constraint.

- `<func>` Function of abstract variables and optimization criteria defining the constraint (\rightarrow 1.4.7).

1.4.6 Defining structural variables

```
STCVAR
<num> <typ> <nod/set> <dof/attr/par> <func>
```

NOTE: When the design input file is run with *SDESIGN*, an '.opi' file will be created; the information in this file can be pasted under the STCVAR heading and this will provide all of the necessary information for the optimizer. Additionally, the full '.oinp' file can be created in *SDESIGN*, which eliminates the need for copying and pasting.

- **<num>** User-defined id-number of structural variable. No gaps are allowed.
- **<typ>** Type of variable:

type of variable	keyword
Elemental attribute	ATTR
Coordinate of fe-node	COORD
Nodal Force	NODF
Composite material	COMP
Fluid parameter	FLUID
Electrostatic parameter	ELCATT

- **<nod/set>** Id-number of fe-node or attribute / material set.
- **<dof/attr/par>** Id-number defining move/force direction, attribute type, or fluid/thermal/electrostatic parameter.

move/force direction	keyword
Global x-direction	DX or 1
Global y-direction	DY or 2
Global z-direction	DZ or 3
Rotation about x-direction	RX or 4
Rotation about y-direction	RY or 5
Rotation about z-direction	RZ or 6

The id-number for the attribute and composite type corresponds to the column number of the material set definition in the *FEM* input. Elemental attributes are:

type of attribute	keyword
Cross-sectional area	AREA or 1
Young's modulus	YMOD or 2
Material density	DENS or 4
Electrostatic permittivity	PERM or 6
Thermal conductivity	TCOND or 6
Thickness	THICK or 7
Thermal expansion	CTE or 11

Composite specification is:

LAYER <id-number> <attribute> The variable composite attributes are:

type of attribute	keyword
Young's modulus 1	EMOD1 or 2
Young's modulus 2	EMOD2 or 3
Shear modulus	GMOD or 5
Density	DENS or 8
Thickness	THICK or 9
Orientation	PHI or 10
Thermal expansion 1	CTE1 or 11
Thermal expansion 2	CTE1 or 12

The following fluid parameters are variable:

type of attribute	keyword
Free stream Mach number	MACH
Angle of attack (pitch)	AOA
Yaw angle	YAW

The following electrostatic parameters are variable:

type of attribute	keyword
Constrained nodal voltage	VOLT

- <func> Function of abstract variables and optimization criteria defining the constraint (→ 1.4.7).

1.4.7 Defining functions

<type> { <description> }

- <type> Type of function:

type	definition	keyword
Sum	$\sum a_i * x_i^{p_i} + b_i$	SUM
Product	$\prod a_i * x_i^{p_i} + b_i$	MUL
Sinus	$\sin(a * x^p + b)$	SIN
Cosines	$\cos(a * x^p + b)$	COS
KS function	$\ln \sum e^{a_i * x_i^{p_i} + b_i}$	KSF

• <description> The parameters a_i, p_i, b_i and x_i have to be specified, where x_i can either be an optimization criteria CRT[i], an abstract optimization variable VAR[i] or an locally defined function OPR[i]. Local function have to be defined first before they can be addressed. Local function definition are marked by the keyword DEF_OPR[i]. The structure and syntax of an function definition is for example:

```
SUM {
    DEF_OPR[1] = KSF {  2.0 * CRT[1] ^ 1.5 - 3.0
                      2.5 * CRT[2]           - 4.0 }
    1.5 * CRT[3] ^ 0.5 - 1.0
    VAR[1]
    3.0 * OPR[1]
}
```

Each line within a function description defines one summand. Incomplete description are allowed, but the order of the parameters is to be kept. Summands can also be generated. In this case the number of the summands has to be specified:

```
GEN ( <first summand>, <last summand>, <step size> )
```

```
SUM {  1:  CRT[1] - 1.0
      10: CRT[10] - 10.0 GEN (1,10,1) }
```

In the generation loop new summands are defined interpolating the parameter a_i, p_i, b_i and the number of x_i between the first and last summand. The type of x_i has to be the same. The step size of the loop can be specified. If the step size is not specified, step size 1 is assumed.

1.4.8 Defining solution strategy

```

SOLVER
<num> <type>
    <solver specifications>
    <specifications for sensitivity analysis>
<num> <type>
    <solver specifications>
    <specifications for sensitivity analysis>
...

```

The overall solution procedure can consist of several different optimization algorithms which are applied in succession.

- **<num>** Id-number of optimization algorithm
- **<type>** Type of optimization algorithm

type of algorithm	keyword
Sequential Quadratic Programming (Schittkowsky)	NLP
Sequential Linear Programming	SLP
Sequential Augmented Lagrange Method	SAL
Method of Moving Asymptotes	MMA
Optimality Criteria Method	OCM

- **<specification>** Specification of parameters for each algorithm. If necessary for each algorithm the sensitivity analysis method (SA) has also to be specified. Only the parameters that are differ from the default values have to specified.

NLP - parameter	keyword	default
Accuracy - convergence	NLPACC	10^{-4}
Maximum number of iterations	NLPITR	10
Maximum number of function calls in line search	NLPFCL	10
Upper bound for scaling functions	NLPSCB	10^6
Solver for quadratic subproblem	NLPLDL	1
Merit function (augmented L2=1, L1=0)	NLPFIT	1
Output level (0/1/2)	NLPPRN	2

SLP - parameter	keyword	default
Maximum iterations	SLPITR	NA
Accuracy - convergence in KKT	SLPACC	NA
Accuracy - convergence in objective	SLPOTL	NA
Type of convergence criteria 0: SLPOTL; 1: SLPACC	SLPICV	NA
Initial move limit $(s_U - s_L)/SLPQMV$	SLPQMV	NA
Treatment of move limits in case of inconsistency 1: doubling of move limits; 2: additional variable	SLPICN	NA
Treatment of move limits in case of cycling 0: $stepsize = SLPFMV * stepsize$ 1: $newstep = SLPFBK * oldstep$ 2: line search	SLPCYC	NA
Adaption factor for move-limits	SLPFMV	NA
Factor for a step backwards in case of cycling	SLPFBK	NA
Factor of Amijo check in line search	SLPAMJ	NA
Adaptation of move limits 0: no adaptation 1: $\alpha_{k+1} = \alpha_k / (\alpha_k + 1)$ 2: $\alpha_{k+1} = \alpha_k / (\alpha_k^2 + 1)$ 3: $\alpha_{k+1} = \alpha_k / 2$	SLPADP	NA
Type of line search (2: quadratic; 3: cubic)	SLPLIN	NA
Output mode (1/2/3)	SLPRT	NA

MMA - parameter	keyword	default
Maximum number of iterations	MMAMXI	500
Accuracy - convergence	MMAACC	$1.0e - 1$
Type of adaption rule	MMASVN	1
0 : rule of Fleury/Kuritz		
1 : rule of Svanberg		
2 : linear adaption $s_u = s/MMASVN$; $s_l = s * MMASVN$		
3 : fixed asymptotes		
4 : mix of all rules		
Rule for upper asymptotes	MMAMXU	1
Rule for lower asymptotes	MMAMXL	1
Type of line search (2: quadratic; 3: cubic)	MMAAPR	3
Maximum number of func calls in line search	MMAMXF	20
Maximum number of cycles in subproblem	MMASUB	20
Output mode (1/2/3)	MMAPRT	2
Upper bound for scaling functions	MMASCB	$1.0e + 6$
Factor for line search adjustment	MMAALM	$1.0e + 0$
Asymptotes adaption factor if iter $\neq 2$	MMASSA	0.5
Asymptotes adaption factor if iter $= 2$	MMASSB	0.7
Upper asymptotes adaption factor if iter $\neq 2$	MMASAU	0.1
Upper asymptotes adaption factor if iter $= 2$	MMASBU	0.1
Lower asymptotes adaption factor if iter $\neq 2$	MMASAL	0.1
Lower asymptotes adaption factor if iter $= 2$	MMASBL	0.2
Linear adaption factor for all asymptotes	MMASCL	0.5
Linear adaption factor for upper asymptotes	MMACLU	0.5
Linear adaption factor for lower asymptotes	MMACLL	0.5
Fix value for upper asymptote	MMAFXU	1.0
Fix value for lower asymptote	MMAFXL	0.01

SAL - parameter	keyword	default
Maximum number of iterations	SALITR	10
Maximum number of iterations in subproblem	SALFCL	10
Size of memory in LBFGS	SALBFG	100
Output mode	SALPRT	1
Accuracy - convergence	SALACC	1.0e - 6
Accuracy - convergence in subproblem	SALACL	1.0e - 6
Initial penalty factor	SALPEN	1.0
Adaptation of penalty factor $r = SALPFC * r$	SALPFC	0.1
Initial values for Lagrange multipliers	SALLAM	0.0
Scaling of Lagrange function	SALSCL	1.0e2
Move limits in subproblem	SALRBO	1.0

OCM - parameter	keyword	default
Accuracy - convergence	OCMACC	1.0 ⁻⁴
Maximum iterations	OCMITR	10
Exponent in power-law	OCMPOW	0.7
Maximum step size	OCMSTP	0.1
Gradient shifting (-1/0/1)	OCMSHT	-1
Correction of mass constraints	OCMCOR	0

Remarks to OCM: The *OCM* algorithm is a special optimizer for topology optimization problems. Gradient shifting can be turned on with and without correcting the mass constraints for orthotropic composite materials. If isotropic material are used gradient shifting is turned on by setting *OCMSHT* to -1.

SA - parameter	keyword	options
Type of gradient evaluation	GRDTYP	ANALYTIC - NUMERIC
Analytic Method : adjoint or direct	GRDMTH	DIRECT - ADJOINT
Numeric Method : forward or central	GRDMTH	FORWARD - CENTRAL
Relative increment (NUMERIC)	GRDRELINC	<value>
Absolute increment (NUMERIC)	GRDABSINC	<value>

1.4.9 Defining explicit optimization variable – finite element relations

In order to speed up problems with a large number of optimization variables and where one optimization variable affects only one (few) finite elements the dependency of a finite element or a elemental property can be explicitly

indicated. For example, this is recommended for large topology optimization problems.

```
STCELV
<id-number of abstract variable> <id-number of finite element>
```

1.4.10 Defining structural analysis parameters

```
STCANALYSIS
<options> <specification>
```

If the structural analysis is an iterative procedure or a transient simulation additional parameters can be specified adjusting the structural analysis to the outer optimization loop. For transient simulation the following options and specifications are available:

option	specification
STCDYNAMIC	INIT \ CONTINUE
STCDAMP	FRQ1 FRQ2 or AUTO
STCTRANS	MINSTEP MAXSTEP DESIGNVEL FORCEVEL

- **STCDYNAMIC** specifies whether the initial conditions are set for each transient simulations or whether the overall optimization process is simulated as a quasi-transient process (e.g. optimization in static aeroelasticity).

INIT initial conditions are set.

CONTINUE conditions of previous simulation are kept. In case of quasi-static simulations velocities and accelerations are set to zero.

- **STCDAMP** adjusts viscous damping factors to critical damping. This option can be applied for quasi-static simulations to improve numerical efficiency.

FRQ1 first frequency.

FRQ2 second frequency.

AUTO frequencies are automatically selected according to a modal decomposition of the structural response at the end of the previous simulation.

- **STCTRANS** specifies a smooth design change if the overall optimization process is simulated as a quasi-transient process.

MINSTEP minimum number of time step for design change.

MAXSTEP maximum number of time step for design change.

DESIGNVEL maximum design velocity (see reference).

FORCEVEL maximum force velocity (see reference).

Chapter 2

FEM - Reliability Module

2.1 Introduction

In general, a reliability analysis quantifies the probability, or likelihood, of a particular event occurring. Often the event in question is the failure of a system due to a particular failure mode. However, the event in question does not always have to represent the failure of the system.

The reliability analysis method chosen to compute the probabilities of failure is the First-Order Reliability Method (FORM). All uncertainties are represented as random variables \mathbf{r} . If the random variables are correlated, they must first be transformed to become uncorrelated. FORM requires a transformation of the random variables \mathbf{r} into standard normal space \mathbf{u}

$$\mathbf{u} = \mathbf{T}(\mathbf{r}) \quad (2.1)$$

where \mathbf{T} is generally a nonlinear mapping that depends on the type of random distribution of \mathbf{r} . In standard normal space the Most Probable Point (MPP), or the closest point from the origin to the failure surface, is to be found by solving an optimization problem with one equality constraint

$$\begin{aligned} \min z &= (\mathbf{u}^T \mathbf{u})^{\frac{1}{2}} & (2.2) \\ \text{subject to: } & g_i(\mathbf{x}(\mathbf{u})) = 0 \end{aligned}$$

where $g_i(\mathbf{x}(\mathbf{u})) = 0$ describes the failure surface. In this study, the optimization problem in (2.2) is solved by a Sequential Quadratic Programming (SQP) method [1, 2]. Once the MPP \mathbf{u}^* is identified, the reliability is found from

$$\beta = (\mathbf{u}^{*T} \mathbf{u}^*)^{\frac{1}{2}} \quad (2.3)$$

where β is the Reliability Index. The probability of failure is then found as

$$P_f = \Phi(-\beta) \quad (2.4)$$

where Φ is the standard normal cumulative distribution function (CDF). The feasibility of the mean-value point is checked to identify on which side of the failure curve the mean design point lies.

FORM is chosen due to the relatively low number of deterministic analyses required, as long as the number of uncertainty variables is reasonably small (i.e. ≤ 100). FORM does have inherent errors in approximating the failure curve as linear. As a result, the failure probability of nonlinear failure curves that are concave to the origin will be underestimated, while convex curves will be overestimated. This error could be reduced by the Second-Order Reliability Method (SORM), which would require the second derivatives of the failure surface with respect to the uncertainty parameters. Although FORM can create problems with convergence of the MPP search, and gives no guarantee to finding the global optimum in the MPP search, at this point the authors consider FORM to be the only practical choice when high-fidelity simulation is applied. As a check to FORM, and an alternative for computationally less expensive problems, Monte Carlo sampling has also been implemented.

According to this short overview of the reliability analysis problem, the input for the reliability module has to be prepared. Subsequently, the structure and the syntax of the input is described and illustrated by some examples. As the reliability analysis method implemented requires an optimization procedure, the setup and running of a single reliability problem is similar to an optimization problem. These similarities are apparent in the input and output files described in the following sections.

2.2 Input files

The initial structure, and the mean or expected structure in a reliability problem, is described in the main *FEM* input file as usual. For the sake of clarity the reliability problem is described in its own file. In order to run a reliability problem, the keyword **STRUCTREL** has to be added in the *FEM* input file, including the name of the reliability input file.

```
variant a: ...  
    STRUCTREL <file>
```

variant b: ...
 STRUCTREL
 <file>

This keyword is denoted similar to the STRUCTOPT <file> is for optimization problems.

The reliability process generates two files which contain information of the given problem, essential iteration data in the MPP search (rel-file), and a protocol of the different optimization algorithms used in the MPP search (rpo-file). The name of the files is composed of the name of the input file and the suffix for the related file. An example:

```
reliability input file  example.rinp
rel-file                example.rel
rpo-file                example.rpo
```

2.3 Structure of input data

The reliability input (rinp-file) is subdivided into different semantical blocks which are indicated by keywords:

semantic block	keyword
reliability criteria q	CRITERIA
abstract random variables r	RNDVAR
failure modes g	FAILURE
structural random variables c	STCRND
solution strategies	SOLVER
structural analysis parameters	STCANALYSIS
end of file	END

The reader may recall that each reliability analysis with FORM requires an optimization with a single equality constraint. Each failure mode identified will produce an optimization problem where the objective is to minimize the sum of the random variables.

As mentioned above failure modes are in general explicit functions of the abstract random variables and the design criteria. Analogously, structural variables are explicit functions of the abstract variables. These functional relations have to be defined in the optimization input. For this some kind of

simple programming language is provided. The syntax, similar to that for optimization input files, is subsequently explained.

2.4 Syntax of input data

2.4.1 General

The following rules apply:

1. No blank lines are allowed.
2. Only capital letters are allowed.
3. Comment lines are mark by # in the first column.
4. No additional data is allowed in lines with keywords marking a semantic block.
5. The order of the semantic blocks is not arbitrary. You can use only reliability variables or criteria to build functions which are already defined.
6. The end of the input data is marked by the keyword END

2.4.2 Defining reliability criteria

```
CRITERIA  
<num> <crit> <nod/frq/dir> <disp/str/strvol-data> <time/loadcase> <gen>
```

- <num> User-defined id-number of optimization criteria. No gaps are allowed.
- <crit> Type of reliability criteria.

type of criteria	keyword	required data
structural mass	MASS	
moment of inertia	INERTIA	axis (RX,RY,RZ only)
strain energy	ENERGY	
kinetic energy	KINETIC	
dissipated energy	DAMPING	
control power	CTRLCOST	
stress volume	STRVOL	strvol-data
eigenfrequency	FRQ	id-number of eigenfrequency
nodal stress	STRESS	node id-number, stress type
nodal displacement	DISP	node id-number, displacement type
displacement level	DISLEVEL	displevel-data
internal force	INTFORCE	node id-number, displacement type
aeroelastic forces	AERO	direction
aeroelastic moments	MOMAER	rotation axis
sonic boom	SBOOM	
failure probability	FAILPROB	id-number of limit state function
reliability index	RELINDEX	id-number of limit state function
performance measure	PERFORM	id-number of limit state function

- If the mass or the strain energy of a subset of elements should be evaluated a list of the respective elemental id-numbers can be specified as follows:

{ <id-number> }

or

{
<id-number>
}

or any combination of the above alternatives.

- <nod/frq/dir> Id-number of FE-node, eigenfrequency or direction of aeroelastic force.

direction	keyword
x-direction	1
y-direction	2
z-direction	3

- <disp/str> Type of displacement or stress.

type of displacement	keyword
translation in x-direction	DX or 1
translation in y-direction	DY or 2
translation in z-direction	DZ or 3
rotation around x-axis	RX or 4
rotation around y-axis	RY or 5
rotation around z-axis	RZ or 6

type of nodal stress	keyword
von Mises - lower surface	VML
von Mises - middle surface	VMM
von Mises - upper surface	VMU
1. principal - lower surface	S1L
1. principal - middle surface	S1M
1. principal - upper surface	S1U
2. principal - lower surface	S2L
2. principal - middle surface	S2M
2. principal - upper surface	S2U

- `<strvol-data>` Definition of stress-volume function. The input of the `<strvol-data>` is

```
<stress type> <reference stress> <exponent> <volume flag> ...
<NODAL/ELEMENTAL> { <list of elements/nodes> }
```

The `<reference stress>` ($\bar{\sigma}$) and the `<exponent>` (p) are defined as follows:

$$\text{volumeflag}=1 \quad \text{strvol} = \left[\frac{1}{\|\Omega\|} \int_{\Omega} \left(\frac{\sigma_i}{\bar{\sigma}} \right)^p d\Omega \right]^{1/p}$$

$$\text{volumeflag}=0 \quad \text{strvol} = \left[\sum \left(\frac{\sigma_i}{\bar{\sigma}} \right)^p \right]^{1/p}$$

where $\|\Omega\|$ is the volume of the finite element domain; σ_i is either the elemental or the nodal stress of interest; $\bar{\sigma}$ a reference stress; p a exponent. The key words `<NODAL/ELEMENTAL>` indicate whether elemental or nodal stresses should be used. In the case of nodal stresses, the number of nodes rather than the volume of the elements is used to define the reference volume $\|\Omega\|$. This function can be used to extract the maximum/minimum stress in the finite element domain by using a large even/odd exponent.

- `<displevel-data>` Definition of displacement leveling function. This function is formulated as follows:

$$\begin{aligned} \text{node number flag}=1 \quad \text{dislevel} &= \left[\frac{1}{N} \sum_N \left(\frac{u_i^j - u_i^{ref}}{\bar{u}_i} \right)^p \right]^{1/p} \\ \text{node number flag}=0 \quad \text{dislevel} &= \left[\sum_N \left(\frac{u_i^j - u_i^{ref}}{\bar{u}_i} \right)^p \right]^{1/p} \end{aligned}$$

where N is the number of term to be considered, u_i^j the degree of freedom j of node i , u_i^{ref} a reference value, \bar{u}_i a scaling factor, and p is an exponent. The reference value can either be given or the average of the $\sum u_i^j/N$ can be used. The input of the `<displevel-data>` is

```
<average flag> <node number flag> <exponent> ...
{ <list of summand definitions> }
```

where the list of summands needs to be specified as follows:

```
<node-id> <displacement type> <reference value> <scaling factor>
```

If the averaging flag is turned on (`average flag = 1`) then the reference value is ignored.

- For reliability related criteria, the id-number of the limit state function corresponds to the order specified in the reliability input file.
- `<time/loadcase>` Time / loadcase to evaluate criteria.

keyword	data
TIME	<time for evaluating criteria>
LCASE	<loadcase for evaluating criteria>

If no time is specified, the criteria is evaluated at the end of the structural analysis. If no loadcase is specified, the criteria is evaluated for loadcase #1.

- `<gen>` Generating optimization criteria.

```
GEN ( <first criteria>,<last criteria>,<step size> )
```

In the generation loop new criteria are defined interpolating the criteria

quantities between the first and last criteria. The step size of the loop can be specified. If the step size is not specified, step size 1 is assumed.

2.4.3 Defining abstract random variables

ABSVAR <num> <dist> <mean> <low> <upp>

- <num> User-defined id-number of abstract random variable. No gaps are allowed.
- <dist> Distribution type: NORMAL, UNIFORM, or LOGNORMAL.
- <mean> Mean value of abstract random variable. Since the abstract variable usually represents the variation of an existing parameter, the mean of the abstract variable is often 0.0
- <sdev> Standard deviation of abstract random variable. Since the true change of the physical random parameter is defined in STCRND, the value here can either represent the actual standard deviation of the physical parameter or the coefficient of variance (COV), if multiplied by the mean in STCRND.
- <low> Lower bound of random variable in real space. If no value is specified, lower bound is set to -10^{10} . (* bound is not in standard normal space)
- <upp> Upper bound of random variable in real space. If no value is specified, lower bound is set to 10^{10} . (* bound is not in standard normal space)

2.4.4 Defining failure modes

FAILURE <num> <pma> <func>

- <num> User-defined id-number of constraint. No gaps are allowed.
- <pma> Target reliability index if the performance measure approach is to be used. If the standard reliability index approach is to be used, then the pma is to be left out.

- **<func>** Function of abstract variables and reliability criteria defining the failure mode or limit state function (→ 2.4.6).

2.4.5 Defining structural variations

STCRND
 <num> <typ> <nod/set> <dof/attr> <func>

NOTE: When the design input file is run with SDESIGN, an '.opi' file will be created; the information in this file can be pasted under the STCRND heading and this will provide all of the necessary information for the optimizer.

- **<num>** User-defined id-number of structural variation. No gaps are allowed.
- **<typ>** Type of variation:

type of variable	keyword
Elemental attribute	ATTR
Coordinate of fe-node	COOR
Nodal Force	NODF
Composite material	COMP
Fluid parameter	FLUID

- **<nod/set>** Id-number of fe-node or attribute / material set.
- **<dof/attr>** Id-number defining move/force direction or attribute type.

move/force direction	keyword
Global x-direction	DX or 1
Global y-direction	DY or 2
Global z-direction	DZ or 3
Rotation about x-direction	RX or 4
Rotation about y-direction	RY or 5
Rotation about z-direction	RZ or 6

The id-number for the attribute and composite type corresponds to the column number of the material set definition in the *FEM* input. Elemental attributes are:

type of attribute	keyword
Cross-sectional area	AREA or 1
Young's modulus	YMOD or 2
Material density	DENS or 4
Thickness	THICK or 7
Thermal expansion	CTE or 11

Composite specification is:

LAYER <id-number> <attribute> The variable composite attributes are:

type of attribute	keyword
Young's modulus 1	EMOD1 or 2
Young's modulus 2	EMOD2 or 3
Shear modulus	GMOD or 5
Density	DENS or 8
Thickness	THICK or 9
Orientation	PHI or 10
Thermal expansion 1	CTE1 or 11
Thermal expansion 2	CTE1 or 12

The following fluid parameters are variable:

type of attribute	keyword
Free stream Mach number	MACH
Angle of attack (pitch)	AOA
Yaw angle	YAW

- <func> Function of abstract variables and optimization criteria defining the constraint (\rightarrow 2.4.6).

2.4.6 Defining functions

<type> { <description> }

- <type> Type of function:

type	definition	keyword
Sum	$\sum a_i * x_i^{p_i} + b_i$	SUM
Product	$\prod a_i * x_i^{p_i} + b_i$	MUL
Sinus	$\sin(a * x^p + b)$	SIN
Cosines	$\cos(a * x^p + b)$	COS
KS function	$\ln \sum e^{a_i * x_i^{p_i} + b_i}$	KSF

• <description> The parameters a_i, p_i, b_i and x_i have to be specified, where x_i can either be an optimization criteria CRT[i], an abstract optimization variable VAR[i] or an locally defined function OPR[i]. Local function have to be defined first before they can be addressed. Local function definition are marked by the keyword DEF_OPR[i]. The structure and syntax of an function definition is for example:

```
SUM {
  DEF_OPR[1] = KSF {  2.0 * CRT[1] ^ 1.5 - 3.0
                    2.5 * CRT[2]          - 4.0 }
  1.5 * CRT[3] ^ 0.5 - 1.0
  VAR[1]
  3.0 * OPR[1]
}
```

Each line within a function description defines one summand. Incomplete description are allowed, but the order of the parameters is to be kept. Summands can also be generated. In this case the number of the summands has to be specified:

```
GEN ( <first summand>, <last summand>, <step size> )
SUM {  1:  CRT[1] - 1.0
      10: CRT[10] - 10.0 GEN (1,10,1) }
```

In the generation loop new summands are defined interpolating the parameter a_i, p_i, b_i and the number of x_i between the first and last summand. The type of x_i has to be the same. The step size of the loop can be specified. If the step size is not specified, step size 1 is assumed.

2.4.7 Defining solution strategy

```

SOLVER
<num> <type>
      <solver specifications>
      <specifications for sensitivity analysis>
<num> <type>
      <solver specifications>
      <specifications for sensitivity analysis>
...

```

As a reliability analysis is required for each failure mode, the user is given the flexibility of choosing the appropriate solution strategy for each failure mode.

- **<num>** Id-number of failure mode
- **<type>** Type of reliability analysis

type of reliability analysis	keyword
First order reliability method	FORM
Monte Carlo sampling	MONTE

If Monte Carlo is chosen only the number of samples to be run is required.

MONTE	
SAMPLES	number of samples

If FORM is chosen, the optimization algorithm to be used in the MPP search is chosen from the following list:

type of algorithm	keyword
Sequential Quadratic Programming (Schittkowsky)	NLP
Sequential Linear Programming	SLP
Sequential Augmented Lagrange Method	SAL
Method of Moving Asymptotes	MMA
Optimality Criteria Method	OCM

- **<specification>** Specification of parameters for each algorithm. If necessary for each algorithm the sensitivity analysis method (SA) has also to be

specified. Only the parameters that are differ from the default values have to specified.

NLP - parameter	keyword	default
Accuracy - convergence	NLPACC	10^{-4}
Maximum number of iterations	NLPITR	10
Maximum number of function calls in line search	NLPFCL	10
Upper bound for scaling functions	NLPSCB	10^6
Solver for quadratic subproblem	NLPLDL	1
Merit function (augmented L2=1, L1=0)	NLPFIT	1
Output level (0/1/2)	NLPPRN	2

SLP - parameter	keyword	default
Maximum iterations	SLPITR	NA
Accuracy - convergence in KKT	SLPACC	NA
Accuracy - convergence in objective	SLPOTL	NA
Type of convergence criteria 0: SLPOTL; 1: SLPACC	SLPICV	NA
Initial move limit $(s_U - s_L)/SLPQMV$	SLPQMV	NA
Treatment of move limits in case of inconsistency 1: doubling of move limits; 2: additional variable	SLPICN	NA
Treatment of move limits in case of cycling 0: $stepsize = SLPFMV * stepsize$ 1: $newstep = SLPFBK * oldstep$ 2: line search	SLPCYC	NA
Adaption factor for move-limits	SLPFMV	NA
Factor for a step backwards in case of cycling	SLPFBK	NA
Factor of Amijo check in line search	SLPAMJ	NA
Adaptation of move limits 0: no adaptation 1: $\alpha_{k+1} = \alpha_k / (\alpha_k + 1)$ 2: $\alpha_{k+1} = \alpha_k / (\alpha_k^2 + 1)$ 3: $\alpha_{k+1} = \alpha_k / 2$	SLPADP	NA
Type of line search (2: quadratic; 3: cubic)	SLPLIN	NA
Output mode (1/2/3)	SLPPRT	NA

MMA - parameter	keyword	default
Maximum number of iterations	MMAMXI	500
Accuracy - convergence	MMAACC	$1.0e - 1$
Type of adaption rule	MMASVN	1
0 : rule of Fleury/Kuritz		
1 : rule of Svanberg		
2 : linear adaption $s_u = s/MMASVN$; $s_l = s * MMASVN$		
3 : fixed asymptotes		
4 : mix of all rules		
Rule for upper asymptotes	MMAMXU	1
Rule for lower asymptotes	MMAMXL	1
Type of line search (2: quadratic; 3: cubic)	MMAAPR	3
Maximum number of func calls in line search	MMAMXF	20
Maximum number of cycles in subproblem	MMASUB	20
Output mode (1/2/3)	MMAPRT	2
Upper bound for scaling functions	MMASCB	$1.0e + 6$
Factor for line search adjustment	MMAALM	$1.0e + 0$
Asymptotes adaption factor if iter $\neq 2$	MMASSA	0.5
Asymptotes adaption factor if iter $= 2$	MMASSB	0.7
Upper asymptotes adaption factor if iter $\neq 2$	MMASAU	0.1
Upper asymptotes adaption factor if iter $= 2$	MMASBU	0.1
Lower asymptotes adaption factor if iter $\neq 2$	MMASAL	0.1
Lower asymptotes adaption factor if iter $= 2$	MMASBL	0.2
Linear adaption factor for all asymptotes	MMASCL	0.5
Linear adaption factor for upper asymptotes	MMACLU	0.5
Linear adaption factor for lower asymptotes	MMACLL	0.5
Fix value for upper asymptote	MMAFXU	1.0
Fix value for lower asymptote	MMAFXL	0.01

SAL - parameter	keyword	default
Maximum number of iterations	SALITR	10
Maximum number of iterations in subproblem	SALFCL	10
Size of memory in LBFGS	SALBFG	100
Output mode	SALPRT	1
Accuracy - convergence	SALACC	1.0e - 6
Accuracy - convergence in subproblem	SALACL	1.0e - 6
Initial penalty factor	SALPEN	1.0
Adaptation of penalty factor $r = SALPFC * r$	SALPFC	0.1
Initial values for Lagrange multipliers	SALLAM	0.0
Scaling of Lagrange function	SALSCL	1.0e2
Move limits in subproblem	SALRBO	1.0

OCM - parameter	keyword	default
Accuracy - convergence	OCMACC	1.0 ⁻⁴
Maximum iterations	OCMITR	10
Exponent in power-law	OCMPOW	0.7
Maximum step size	OCMSTP	0.1
Gradient shifting (-1/0/1)	OCMSHT	-1
Correction of mass constraints	OCMCOR	0

Remarks to OCM: The *OCM* algorithm is a special optimizer for topology optimization problems. Gradient shifting can be turned on with and without correcting the mass constraints for orthotropic composite materials. If isotropic material are used gradient shifting is turned on by setting *OCMSHT* to -1.

SA - parameter	keyword	options
Type of gradient evaluation	GRDTYP	ANALYTIC - NUMERIC
Analytic Method : adjoint or direct	GRDMTH	DIRECT - ADJOINT
Numeric Method : forward or central	GRDMTH	FORWARD - CENTRAL
Relative increment (NUMERIC)	GRDRELINC	<value>
Absolute increment (NUMERIC)	GRDABSINC	<value>

2.4.8 Defining explicit random variation – finite element relations

In order to speed up problems with a large number of random variables and where one random variable affects only one (few) finite elements the dependency of a finite element or a elemental property can be explicitly indicated.

STCELV <id-number of abstract variable> <id-number of finite element>
--

2.4.9 Defining structural analysis parameters

STCANALYSIS <options> <specification>
--

If the structural analysis is an iterative procedure or a transient simulation additional parameters can be specified adjusting the structural analysis to the outer optimization loop. For transient simulation the following options and specifications are available:

option	specification
STCDYNAMIC	INIT \ CONTINUE
STCDAMP	FRQ1 FRQ2 or AUTO
STCTRANS	MINSTEP MAXSTEP DESIGNVEL FORCEVEL

- **STCDYNAMIC** specifies whether the initial conditions are set for each transient simulations or whether the overall optimization process is simulated as a quasi-transient process (e.g. optimization in static aeroelasticity).

INIT initial conditions are set.

CONTINUE conditions of previous simulation are kept. In case of quasi-static simulations velocities and accelerations are set to zero.

- **STCDAMP** adjusts viscous damping factors to critical damping. This option can be applied for quasi-static simulations to improve numerical efficiency.

FRQ1 first frequency.

FRQ2 second frequency.

AUTO frequencies are automatically selected according to a modal decomposition of the structural response at the end of the previous simulation.

- **STCTRANS** specifies a smooth design change if the overall optimization process is simulated as a quasi-transient process.

MINSTEP minimum number of time step for design change.

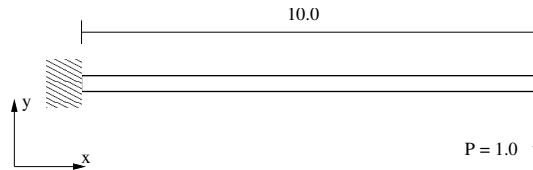


Figure 2.1: Example 1 - 2-Dimensional Cantilevered Beam

Uncertainties	mean	COV	Standard deviation
Modulus of elasticity	1.0×10^8	0.1	1.0×10^7
Beam length	10.0	0.1	1.0

Table 2.1: Uncertainties for reliability analysis

MAXSTEP maximum number of time step for design change.

DESIGNVEL maximum design velocity (see reference).

FORCEVEL maximum force velocity (see reference).

2.5 *FEM* Reliability example

For some simple examples, input files of *FEM* and the reliability module are subsequently given.

2.5.1 2-Dimensional Cantilever

The a reliability analysis is performed on the two-dimensional beam pictured in Fig. (2.1). The uncertainties accounted for are summarized in Table (2.1).

The failure mode, or the event for which the probability of occurring is desired, is downward displacement of the tip exceeding 0.1 units. The single equality constraint for the MPP optimization problem then becomes

$$1.0 * u_{tip} + 0.1 = 0.0 \quad (2.5)$$

In this example failure mode, the actual displacement of the tip u_{tip} is called load of the system. The set level of displacement that the system can handle, in this case 0.1 units, is called the resistance of the system. As the reesistance of a system is often an uncertainty in itself, such as variation in the yield

stress of a material, uncertainty in the resistance can be incorporated as follows

$$1.0 * u_{tip} + 0.1 + U_R * \sigma_R = 0.0 \quad (2.6)$$

where U_R is the uncertainty variable for the resistance and σ_R is the standard deviation of the resistance.

The input file describing the mean design for *FEM* is:

```
*
*-----
*
*   FEM -Input-file: Example 1
*
*-----
*
CONTROL
relex1
1
nset
eset
*
STRUCTREL relmonte.rinp
*
RENUM
rcm
*
*-----
*
OUTPUT
gdisplac relex1.dis 1
stressvm relex1.von 1
thicknes relex1.thk 1
shapeatt relex1.sha 1
shapestc relex1.shc 1
energies relex1.ene 1
*
*-----
*
MATERIALS
1 0.02 1.0e+8 0.30 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 6.67e-5 6.67e-5 6.67e-5
*
*-----
*
NODES
1 0.0 0.0 0.0
2 10. 0.0 0.0
*
*-----
*
TOPOLOGY
1 7 1 2
*
*-----
```

```

*
ATTRIBUTES
1 1 1
*
*-----
*
EFRAME
*
1 1 0 0 0 1 0 0 0 1
*
*-----
*
DISPLACEMENTS
*
1 1 0.000000
1 2 0.000000
1 3 0.000000
1 4 0.000000
1 5 0.000000
1 6 0.000000
*
*-----
*
FORCE
*
2 2 -1.000000
*
*-----
*
END
*

```

The input file *relex1.rinp*, without resistive uncertainty, for the reliability module is:

```

#-----
#
# Input-File for Reliability Module
#
# Example 1: Cantilevered Beam
#
# Uncertainties: Modulus of elasticity and Beam length
# Failure Mode: Tip Displacement exceeding 0.1 units
#
#-----
#-----
#
CRITERIA
#
# number      type      node/frq.    dof
#
#      1      DISP          2           DY
#
#-----
#

```

```

RNDVAR
#
# number  distribution  mean  sdev  lower  upper
#
#       1   NORMAL      0.0   0.1   -1.0   1.0
#       2   NORMAL      0.0   0.1   -1.0   1.0
#
#-----
#
FAILURE
#
# number  PMA (optional)      func-type
#
#       1                               SUM { 1.0 * CRT[1] + 0.1 }
#
#-----
#
STCRND
#
# number  type      nod  dof      func-type
#
#       1  ATTR      1    YMOD  SUM { 1.0e+8*VAR[1] }
#       2  COOR      2    DX    SUM { 10.0*VAR[2] }
#
#-----
#
SOLVER
#
# number  type
#
#       1  MONTE      (Monte Carlo not selected)
#       SAMPLES 1e4
#
#       1  FORM
#       NLP
#       NLPITR 100
#       NLPFCL 10
#       NLPSCE 1.0D+4
#       NLPLDL 1
#       NLPACC 1.0D-6
#       NLPPRN 2
#       NLPFIT 1
#
#       GRDTYP  NUMERIC
#       GRDMTH  CENTRAL
#       GRDABSINC 1.0E-4
#
#       GRDTYP  ANALYTIC      (Analitical gradients not selected)
#       GRDMTH  DIRECT
#       GRDMTH  ADJOINT
#
#-----
#
END

```

#

Remarks:

- FORM is chosen as the reliability analysis. For a structural problem of this simplicity, Monte Carlo sampling would be a feasible option and is therefore included, but commented out.
- As with optimization problems, FORM requires sensitivities for the gradient-based algorithm. In this example numerical sensitivities are used. However, analytical direct and adjoint sensitivities are available and are more efficient for computationally more expensive problems.

Chapter 3

SDESIGN - Geometric Modeler

The program *SDESIGN* is a geometric modeler based on the design element concept. Analogously to finite elements, the geometry of the structure is piecewise approximated by so-called design or macro elements. Each design element is defined by the position of the so-called control nodes and a specific shape function.

Currently, *SDESIGN* provides the possibility of describing 3-dimensional shell and plate structures by 2-dimensional Coons-elements, Bezièr patches and surfaces of revolution and 3-dimensional Coons-volumes. In general, different edge types can be mixed within a given design element definition. Linear, quadratic and cubic Lagrange, cubic Bézier splines, b-splines and spiral edges are implemented. Also, 2-dimensional Coons-elements can be constructed into a 3-dimensional Coons volume. Boundary conditions for structural, heat transfer and electrostatic problems may also be defined. *SDESIGN* can be used for two type of problems:

1. Based on the defined design element model, a structured fe-mesh with triangular or quad elements (2-d) or brick or tetrahedra elements (3-d) is generated. The node, topology (element), boundary condition and miscellaneous data can be automatically inserted into a *FEM* input file. Additionally, the functional dependencies between the position of the finite element nodes and the position of the control nodes of the design elements (*design velocity field*) can be generated and printed in the format of the input file for the *FEM* optimization or reliability module.

2. A finite element mesh is given (only fe-nodes have to be given). These finite element nodes are projected onto the defined design element model. The design velocity field for the projected fe-nodes is generated and printed in the format of the input file for the optimization module. See section 3.3 for details about projecting a given finite element mesh onto design element model.

3.1 File names and command line options

SDESIGN must be given at least one input file in order to run, however as many input files as desired may be run at one time. Each file may have its own set of options associated with it. A given set of options corresponds *only* to the input file directly following them. The form is given below:

```
sdesign <options 1> <input file 1> ... <options N> <input file N>
```

- **<options>** options must be designated with the '-' character, with the options immediately following (no space in between). Multiple options may be concatenated following one '-', but this is not required. Generally, a lowercase option signifies that only a certain thing will be processed and an uppercase options signifies that a certain thing will not be processed. The lowercase and uppercase option refer to the same thing. See table 3.1 for a list of command line options. There are several output files that may be generated by *SDESIGN*. The names of the output files are composed of the name of the input file and the suffix for the related file, unless otherwise specified:

design element input file	<code>example.dsg</code>
fe-file	<code>example.input</code>
opt-file	<code>example.opi</code>

There are standard output files and optional output files. The manner in which optional output files are specified is detailed in 3.2.2. Table 3.2 summarizes the output files:

Option	Function
'Only' options	
-c	only generate design elements
-d	only generate displacement boundary conditions
-e	only generate voltage boundary conditions
-f	only generate force boundary conditions
-h	only generate temperature boundary conditions
-p	only use patches, regardless of design file
-r	only use trusses, regardless of design file
-t	only generate or project topology
-v	only use volumes, regardless of design file
-w	print all screen output to file "screen.out"
'Not used' options	
-B	do not generate any boundary conditions
-C	do not generate design elements
-D	do not generate displacement boundary conditions
-E	do not generate voltage boundary conditions
-F	do not generate force boundary conditions
-H	do not generate temperature boundary conditions
-I	do not generate electrical-structure interfaces
-L	do not output local coordinates
-M	do not generate mesh motion flags
-O	do not generate design velocity field
-P	do not use patches, regardless of design file
-R	do not use trusses, regardless of design file
-T	do not generate or project topology
-V	do not use volumes, regardless of design file
-W	print all screen output to file specified next

Table 3.1: Command line options

suffix	contents of file
Standard	
.loc	Local coordinates of fe-nodes within design elements
Optional	
.cnbc	matching control file for <i>MATCHER</i>
.del	design element data, for <i>TOPDOMDEC</i>
.input	input file for <i>FEM</i>
.match	element and node matching
.mod	“modal” animation of opt variables, match ABSVAR def
.oinp/.opi	input file/dvf for optimization module
.sin	topology and b.c. info for matching (FORTRAN fluid code)
.sgi	topology and b.c. info for matching (C++ fluid code)
.stl	stereoscopy file for 3-D viewing
.top	node and element data for <i>TOPDOMDEC</i>
.vmo	mesh motion nodal freedoms

Table 3.2: *SDESIGN* output files

3.2 Structure and syntax

3.2.1 General

The following rules apply:

1. Blank lines are allowed everywhere except at the beginning of the file.
2. Comment lines are marked by a # in the first column.
3. All keywords are case sensitive, so be careful to note the case in which they are given in the manual.
4. Continuous integer lists may be given in the form a:b
5. The 5 basic arithmetic operations (+, -, *, /, ^) can be used any place a floating point number is expected. Standard order of operations and parantheses are followed. Additionally, defined variables may be used in the arithmetic operations. If any arithmetic operations are being used somewhere that consecutive floating point values are expected, the fields must be separated by commas.
6. The design element input file is divided into semantic blocks, which are marked by keywords, defined in tables 3.3, 3.4 and 3.5. Though

Semantic block	Keyword
aeroelastic algorithm	AERO
composite frames for finite elements	CFRAME
composite properties of finite elements	COMPOSITE
control parameters	CONTROL
dynamic solver	DYNAMIC
electro-mechanical eigenvalue algorithm	EIGELECSTRC
mesh-motion eigenvalue algorithm	EIGMESHALG
electro-mechanical algorithm	ELECSTRC
fit algorithm	FITALG
material properties of finite elements	MATERIAL
mesh-motion algorithm	MESHALG
mesh motion scheme	MESHMOTSCHEME
number of Laplace mesh motion steps	MESHMOTSTEP
non-linear solver	NONLINEAR
electro-mechanical optimization algorithm	OPTELECSTRC
mesh-motion optimization algorithm	OPTMESHALG
electro-mechanical POD decomposition algorithm	PODELECSTRC
mesh-motion POD decomposition algorithm	PODMESHALG
renumbering method	RENUM
static solver	STATIC
tolerance of rigid body motions	TRBM

Table 3.3: Supported semantic blocks for *FEM*

the semantic blocks do not have an exact required order, there are several instances in which one must be defined before another. The list of semantic keywords in the three tables is a valid ordering. The user should mimic this order, omitting the fields that are not desired (assuming they are optional).

7. Unless otherwise noted, the following is required for semantic blocks:
 - (a) A newline must follow any semantic block keyword
 - (b) All fields in a given definition must be on the same line in the input file. For example, when a patch is defined, the number, type, mesh properties and bounding edges for a given patch must all be entered on the same line.
8. When defining items in the input file, there are several instances in which either a keyword *or* a number can be used. Where this is pos-

Semantic block	Keyword
abstract design variables	ABSVAR
constraints	CONSTRAINT
optimization criteria	CRITERIA
failure function	FAILURE
objective function	OBJECTIVE
random design variables	RNDVAR
solver definition and parameters	SOLVER
structural analysis parameters	STCANALYSIS

Table 3.4: Supported semantic blocks for optimization module
For reliability input, any keyword that is the same for optimization must be prefixed by ‘REL’

sible, both options are given. Either the number, the keyword or a combination of both will work.

9. There are several items that can be specified for the finite element code used in the Center for Aerospace Structures (*FEM*). The reasoning for this is that *SDESIGN* optionally generates a *FEM* input file with defaults. *SDESIGN* is set up so that it can process input for *FEM* and print it into the *FEM* input file, so that the user can specify everything desired in one place - the *SDESIGN* input file. Not every input available for *FEM* is set up to be understood by *SDESIGN*, so if an error is encountered when trying to use a specific *FEM* input, it may be that it is not implemented. The one difference in input is that instead of the keyword OUTPUT being used to designate *FEM* output files, FEMOUTPUT is used. All fields corresponding to *FEM* are optional. Definitions for the input file for the optimization module and reliability module of *FEM* can also be made in the *SDESIGN* input file, with the above information applying. Do not use the END keyword for any of the above definitions.

The design element geometry needs to be defined in the *SDESIGN* input file (‘.dsg’ file). Additionally, variable control nodes have to be marked. If the position of a control node is a function of other control nodes this function can be defined by so-called macros. The same general rules given in section 1.4.1 apply. The following sections layout particular aspects of the *SDESIGN* input file.

Semantic block	Keyword
optional output files for <i>SDESIGN</i>	SDOUTPUT
optional output files for <i>FEM</i>	FEMOUTPUT
input variables to be used in input file from a file	INPUTVAR
tolerance of node projection convergence	ACCURACY
max number of newton iterations for node projection	ITERATION
tolerance for node location within design element	TOLERANCE
heading under which variable names can be assigned	DEFINE
order in which to generate design element types	DEORDER
strategy used to check for duplicated nodes	BNDNODESTRAT
scale the value of all nodal coordinates	SCALE
control nodes	NODE
edges	EDGE
1-D design elements (with 1-D finite elements)	TRUSS
2-D design elements (with 2-D finite elements)	PATCH
3-D design elements (with 3-D finite elements)	VOLUME
material attributes assigned to design elements	ATTRIBUTE
limiting box for node projection	SEARCHBOX
limiting box for node matching	MATCHBOX
force boundary conditions	FORCE
displacement boundary conditions	DISPLACEMENT
temperature boundary conditions	TEMPERATURE
voltage boundary conditions	VOLTAGE
soft voltage boundary conditions	SOFTVOLT
interface between two physical domains	INTERFACE
nodal flags for freedom of movement in mesh motion	MESHMOTION
variable control nodes for shape variations	SHAPEDSGVAR
optimization variables for elements in a design element	ELED SGVAR
optimization variables for nodes in a design element	NODALDSGVAR
macros that link variable control nodes for optimization	LINK
macros that link variable control nodes for reliability	RELINK
initial node/element numbers	INIT
existing FE-mesh for projection	FEMESH
existing local coordinate mesh	LCMESH
end of file	END

Table 3.5: Available semantic blocks for *SDESIGN*

3.2.2 Defining Optional Output Files for *SDESIGN*

```
SDOUTPUT
<output file keyword> <file name (optional)>
<output option keyword (optional)> <output option value (optional)>
```

- **<output file keyword>** Specifies which file to output. See table 3.2 for the contents of the output files.

Keyword	Corresponding output file
deselement	.del
feminput	.input
flucontrol	.cnbc
fluidtop	.sgi
match	.match
meshmotion	.mmo
optinput	.oinp/.opi
stereo	.stl
sinus	.sin
topology	.top
varmode	.vmo

- **<file name>** If an optional file name is not specified, the file will be given the base name of the input file with a default suffix. This name must be entered *before* any output file options in order to be recognized as a filename.

- **<output option keyword> <output option value>** A keyword to indicate an option associated with a particular output file. A keyword is required to differentiate this field from a specified file name. Table 3.6 lists the available options and keywords, the corresponding output file and the function. Note that the maximum string length of the output option value is 100 characters.

3.2.3 Defining Output Files for *FEM*

```
FEMOUTPUT
<file keyword> <file name (optional)> <frequency (optional)>
<option (optional)>
```

See the *FEM* manual for all possible output files. All the same keywords are

File	Keyword	Value	Function
deselement	deltyp " eleset nodset patset volset	allnode corner any string " " "	connect all control nodes with lines connect corner nodes with quads or bricks prefixes string to all Xpost ElementSet indicate name for Xpost NodeSet indicate name for patch elements ElementSet indicate name for volume elements ElementSet
feminput	outprec	integer	min # of decimal places for scientific notation output
flucontrol	locate	yes/no	match or do not match nodes in file (default 'no')
local	noloc	–	do not produce a '.loc' file
optinput	addextvar endextvar externvar onlyopi outprec usexo	filename filename – – integer yes/no	write STCVAR to the given file and allow more write STCVAR to the given file but allow no more allow more STCVAR to be written from other files output dvf only, not entire input file for opt. module min # of decimal places for scientific notation output whether to output intial mesh coord in dvf (def: 'no')
sinus	sindim	2 or 3	dimension of sinus file produced (default: 3)
topology	split eleset ifcset patset nodset truset volset	– any string " " " " "	different Xpost ElementSet for all design elements prefixes string to all Xpost ElementSet indicate name for interface elements ElementSet indicate name for patch elements ElementSet indicate name for Xpost NodeSet indicate name for truss elements ElementSet indicate name for volume elements ElementSet

Table 3.6: *SDESIGN* output file options

used in *SDESIGN*.

Note: The option may only be specified in *SDESIGN* if the 3 previous fields are also defined.

3.2.4 Defining Inputs for *FEM*

The MATERIAL semantic block has special inputs for *SDESIGN*, defined here, for all others, see *FEM* manual.

```
MATERIAL
<standard FEM input>
or
<material type> <units> <area/thickness (optional)>
```

- <material type> The following standard material types are implemented:

material type	type keyword
Aluminum	alumi
Silicon	silic
Silicon with half of elastic modulus	silic2
Steel	steel
Vacuum	vacuum

- <units> To designate metric units, the keyword **metric** should be used. The keyword **english** is used for English units.
- <area/thickness> If truss or plane element are being generated an area or thickness is required. The value in this field will be assigned to the area *and* the thickness as both are not required for the same element. This field is optional, with the default being zero.

3.2.5 Defining Inputs for *FEM* Optimization Module

The CRITERIA semantic block has special inputs for *SDESIGN*, defined here, for all others, see the beginning of this manual. When defining which node a criteria corresponds to, for example, stress and displacement, it may be desired to refer to a control node, edge or patch, instead of a finite element node, as is currently the standard input.

If it desired to refer to a defined geometrical entity in the *SDESIGN* input file, the fe-node number is replaced by

```
<geometry type> <geometry number>
```

- <geometry type> Only node, edge and surf of table 3.10 are available for use.
- <geometry number> The corresponding number of the geometry, as defined in the input file for *SDESIGN*.

If a criteria is defined on an edge or a patch, with multiple nodes, as many criteria as nodes will be printed. In turn, if such a criteria is used in a constraint, as many constraints as nodes will be printed. If this happens with the objective, only the first one will be printed, since only one objective is allowed.

3.2.6 Defining Input Variables from a File

This field may be used in place of the **DEFINE** section, in order to define variables that can be used throughout the *SDESIGN* input file. In order to use a variable, this file must have been read, therefore, if a node position is given using variables defined in the **INPUTVAR** file, this section must precede the usage of the variable.

```
INPUTVAR                                <file name of input variables>  
or  
INPUTVAR  
<file name of input variables>
```

- The syntax of the input file is as follows:

```
VARIABLE  
<number of input variables>  
<list of variable names>  
<list of variable values>  
END
```

A newline must be started for each field listed above. The variable names and values are all given on the same line in the file, and separated by spaces. The keywords **VARIABLE** and **END** must be given.

3.2.7 Defining Accuracy, Iteration, Tolerance

Using **ACCURACY** as an example of all three ...

ACCURACY	<accuracy value>
or	
ACCURACY	<accuracy value>

Note: All three of these fields are optional. If they are not defined, the default values will be used. Also, these options are only applicable when a pre-existing fe-mesh is being projected onto a design element model.

- **ACCURACY:** This determines the tolerance within which a node is considered to be located by the projection algorithm, when projecting nodes onto a design element. Changing this value does not significantly alter the speed of the program. The default value is 1.0×10^{-8} .

- **ITERATION:** This determines the maximum number of iterations in the Newton process that locates nodes in a design element. If a node is not located within the maximum number of iterations, a node is determined to not be in the design element. The default value is 20. In general, this value does not need to be changed.

- **TOLERANCE:** This determines the tolerance within which a node is considered to be within a given design element, when projecting nodes onto a design element. This is a very important parameter if the user is trying to project nodes of a fe-mesh onto a design element with very curved geometry. If the user is projecting nodes onto a design element and certain nodes are determined not to be within a design element that should be, this is the second parameter that the user should change in order to find all appropriate nodes. The user should be cautioned not to hastily change this parameter without being sure that the input file is error free. Note that the first parameter to be adjusted is the number of elements in the local directions of the design element in question (discussed in section 3.3). The default value is 1.0×10^{-4} . Do not change this parameter unless you are sure!

3.2.8 Defining Variables

DEFINE
<variable name> = <value>

A variable can be defined anywhere in the *SDESIGN* input file, so long as it comes under the **DEFINE** heading, which can be used any number of times.

element type	type number	type keyword
patch	0	patch
volume	1	volume
truss	2	truss
interface	3	interface

Table 3.7: Design element type keywords and numbers

The variable name can then be inserted in place of a numeric floating point value after it has been defined.

3.2.9 Defining Design Element Generation Order

```
DEORDER <1st d.e.> <2nd d.e. (optional)> <3rd d.e. (opt.)>
or
DEORDER
<1st d.e.> <2nd d.e. (optional)> <3rd d.e. (optional)>
```

The standard order of design element generation is: truss, patch, volume. If the user would like to change this order, than the keywords or numbers of table 3.7 can be used to specify the order. If only one or two design element types are specified, than the default order will be used for the remaining ones. Note that ‘interface’ is not considered in this field as it is treated as a boundary condition.

3.2.10 Defining Duplicated Node Checking Strategy

```
BNDNODESTRAT <strategy>
or
BNDNODESTRAT
<strategy>
```

<strategy> Integer or keyword that defines the node checking strategy to be used, when checking for duplicate nodes.

Strategy	Number	Keyword
Do not check nodes	0	nobnd
Check all boundary nodes	1	allbnd
Check boundary nodes on already used interfaces	2	usedbnd
Check every node, boundary or not	3	everynode

The default value is allbnd.

3.2.11 Defining Node Scaling

SCALE	<scaling factor>
or	
SCALE	<scaling factor>

<scaling factor> The nodal coordinates will all be scaled by this number.

3.2.12 Defining Nodes

NODE	<number>	<x>	<y>	<z>
------	----------	-----	-----	-----

The z-coordinate may be omitted for two dimensional problems.

3.2.13 Defining Edges

EDGE	<number>	<type>	<number of nodes>	<list of nodes>	<edge refinement (optional)>
------	----------	--------	-------------------	-----------------	------------------------------

- <type> The following edge types are implemented:

edge type	type keyword	# nodes
linear Lagrange	linear	2
quadratic Lagrange	quadratic	3
cubic Lagrange	cubic	4
spiral	spiral	4
cubic Bézier	bezier	4
b-spline	bspline	variable

In figure 3.1, the internal node numbering for edges is given.

- <edge refinement> In order to refine the mesh along a certain edge, a refinement factor can be specified in the following manner: the keyword **eref** followed by the refinement factor. The factor should be ≤ 1 (the closer to zero, the more favor is given to refinement), with the sign of the number indicating direction in which to refine.

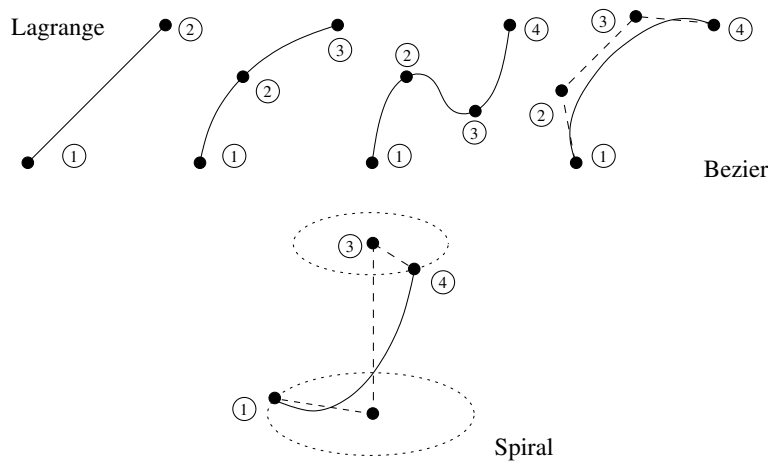


Figure 3.1: Edge definitions

element physics	element type	type keyword
structural	2-node bar	bar
structural	2-node Bernoulli beam	bernbeam
structural	2-node Timoshenko beam	timobeam
electrostatic	2-node electrostatic bar	estattruss
thermal	2-node thermal bar	thermtruss

Table 3.8: Types of 1-d finite elements available

3.2.14 Defining Trusses

```

TRUSS
<number> <fe-type> <number of elements in local x-direction>
          <list of edges>

```

- **<fe-type>** The types of 1-d finite elements available for use with trusses, corresponding to *FEM*. The possibilities are listed in table 3.8
- **<number of elements in local x-direction>** Number of truss elements along a given edge.
- **<list of edges>** Every edge in this list will have the specified number of 2-node linear elements generated along its trajectory.

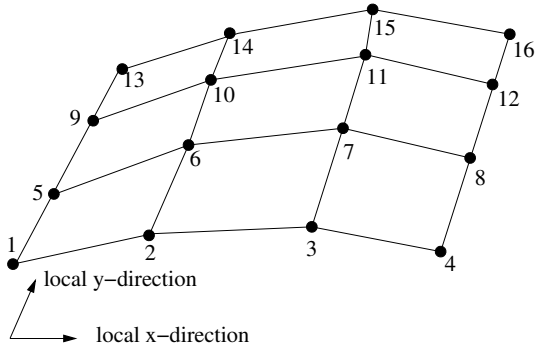


Figure 3.2: Node input order for 16 node Bezier patch

3.2.15 Defining Patches

```

PATCH
<number> <type> <fe-type> <triangle (optional)>
          <number of elements in local x-direction>
          <number of elements in local y-direction>
          <list of edges or nodes>
          <starting point (opt)> <bound. node check (opt)>
          <mesh transition (opt)>

```

- **<type>** The following patch types are implemented:

patch type	type number	type keyword	# of edges or nodes
coons	0	coons	4
revolution	1	revol	3
bezier	2	bezier16	16

note: Only coons patches support node projection. See figure 3.2 for input of nodes of the 16 node Bezier patch.

- **<fe-type>** The types of 2-d finite elements available for use with patches, corresponding to *FEM*. The possibilities are listed in table 3.9

- **<triangle>** The orientation of triangular elements. Possible options are shown in figure 3.3, with the keywords **left** and **right**. If a finite element type is used that does not require a triangle orientation, such as a quad element, an entry is not required. This is an optional field.

element physics	element type	type keyword
structural	2-node bar	bar
structural	2-node bar, overlapped	bar2
structural	2-node Bernoulli beam	bernbeam
structural	2-node Timoshenko beam	timobeam
structural	3-node tri, isotropic	isotri
structural	3-node tri, composite	comptri
structural	3-node tri, overlapped isotropic	isotri2
structural	3-node tri, overlapped composite	comptri2
structural	4-node quad	quad4
electrostatic	4-node quad	estatquad4
heat transfer	3-node tri	thermtri
heat transfer	4-node quad	thermquad4

Table 3.9: Types of 2-d finite elements available

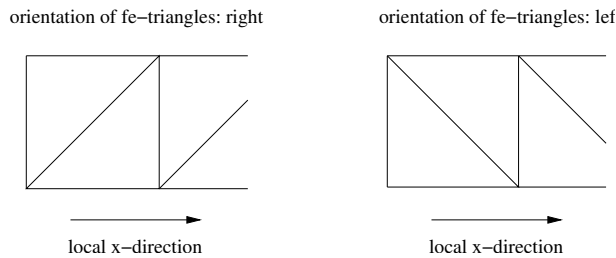


Figure 3.3: Definition of orientation of triangular elements

- **<number of elements in local x\y-direction>** The number of elements generated along the related edge of the design element. If the user is generating volumes out of the patches and does not wish to have two-dimensional elements and nodes generated, the number of elements should be set to zero. It is possible to have a combination of patches, with and without elements, and volumes.

When generating nodes, these values are applied to the *local* x and y directions. Since the user will not know what *SDESIGN* will use as local directions, it is recommended that the user first test this using only a small number of elements, so as not to waste time generating a fine mesh, only to have it be incorrect. Once the local directions are properly determined, these fields can be set to the desired values.

When projecting pre-defined nodes, rather than generating nodes, if the number of elements for a given patch is set to zero, then there will be no projection onto that patch. If projection of nodes onto a patch is desired, these fields must be set greater than zero. For detailed information about the importance of this field in projecting a pre-existing mesh, see section 3.3.

- **<list of edges>** Any combination of edge types may be used. In regards to edge ordering, the edges must be listed in order, however, the direction of patch traverse is unimportant. For a patch of revolution, the first and third edges determine the starting and stopping edges for the surface, as well as the direction of revolution (from edge one to three). The second edge is the axis of revolution.

- **<starting point>** This field is only applicable to the projection of a pre-existing fe-mesh onto a patch. The purpose of this field is to allow the user to choose whether or not a good starting point for a node's location on a patch is automatically generated, or if one is only generated upon failed convergence when a standard starting point is used.

If the projection is being performed in a patch with all linear (2 node) edges, convergence will most likely occur without a good starting point, so it is more efficient to only generate a starting point on failed convergence. This is done by the keyword **nostartpt**, following all other fields in the given patch definition.

However, if even one edge is non-linear, a good starting point is most likely required, therefore it is faster to require a good starting point before the projection process begins. This is done by the keyword **startpt**. This

is an optional field, with the default being **startpt**. If the user is unsure of which is the best choice, use the default, as it is usually faster. This field can only be used in the absence of the boundary node check definition, see below. This is an optional field.

- **<boundary node check>** This will only have an effect if the boundary node strategy **usedbnd** is employed, see section 3.2.10. Normally, if this option is used and two volumes share a patch, then the nodes on this surface for the second volume are not generated, so as not to create duplicate nodes. If it is desired that duplicated nodes should be created on a given surface interface, the keyword **nobnd** should be written here. If the interface patches are defined as different design elements in the input file, though they are still equivalent, apply this definition to all of them, or else it will be ignored. This field can only be used in the absence of the starting point definition, see above. This is an optional field.

- **<mesh transition>** This field is for the purpose of generating a mesh with varying fineness across a design element. Currently, there can only be a transition in mesh size in the local x-direction. The syntax is as follows:

trans <number of elements in opposite local-x direction>

The keyword **trans** and the number of elements must appear on the same line as the rest of the patch input. Also, the number given here must be smaller than the initial number of elements in the local x-direction given. This is an optional field.

3.2.16 Defining Volumes

```
VOLUME
<number> <type> <fe-type>
          <number of elements in local x-direction>
          <number of elements in local y-direction>
          <number of elements in local z-direction>
          <list of patches> <starting point (optional)>
          <exclude projected nodes (optional)>
```

Note: Duplicating volumes may cause problems in the design velocity field, therefore the user should be careful if volume duplication is desired.

- **<type>** So far, only a 3-D Coons Element is implemented:

type	type #	type keyword	# nodes	# edges	# patches
coons	0	coons	8	12	6

- `<fe-type>` The type of finite element corresponding to *FEM*.

element physics	element type	type keyword
structural	8-node brick	brick8
structural	20-node brick	brick20
structural	4-node tetrahedron	tetra4
electrostatic	8-node brick	estatbrick8

- `<number of elements in local x\y\z-direction>` Number of elements generated along the related face of the design volume. If this number is zero, there will be no generation or projection of nodes in this design volume. See this field in section 3.2.15 for all relevant information. Also, see section 3.3 for detailed information about this field regarding projection of a pre-existing fe-mesh into a volume.

- `<list of patches>` List of patches that define a volume, exactly six are required. There is no specific ordering of patches required.

- `<starting point>` See the information regarding this field in section 3.2.15. All information applies to volumes. This is an optional field.

- `<exclude projected nodes>` This field is used to specify that all nodes considered to be in a volume during node projection will actually be excluded from parameterization, regardless of whether or not they are contained within another design volume. This option is specified by writing the keyword **excludenodes**. This is an optional field with the default being **includenodes**, which can be explicitly specified if desired.

3.2.17 Defining Attributes

This field allows a material property and/or composite property and frame (defined in the **MATERIAL**, **COMPOSITE** or **CFRAME** semantic blocks, see section 3.2.4) to be applied to an entire design element. If an attribute is not specified for every design element employed, then by default, material property 1, with no composite information, will be assigned to the design element.

ATTRIBUTE

```
<number> <design element type> <design element number>  
<material number> <composite number (optional)>  
<cframe number (optional)>  
<one material/attribute per element (block) (optional)>
```

- **<design element type>** This field specifies which type of design element the attribute corresponds to. See table 3.7 for a list of design element definitions.
- **<design element number>** Number of design element, as defined in *SDESIGN* input file.
- **<material\composite\cframe number>** Corresponds to the number of material, composite or cframe defined in the input file.
- **<one material/attribute per element>** *SDESIGN* will output one material property (with a unique number), which will be a copy of the material property assigned in the **<material number>** field, to each finite element in a design element. The attribute of each finite element will be listed individually. This is accomplished by the keyword **sepmat** placed at the end of an attribute input line. Additionally, if it is desired that the same material should correspond to an element block (this is more than one element only for triangles) the keyword **blockmat** should be used. This field is useful for a topology optimization problem where each element needs a separate material property. This is an optional field.

3.2.18 Defining Search-boxes

SEARCHBOX

```
<number> <design element type> <number of design element>  
<min x> <max x> <min y> <max y> <min z> <max z>  
<inside or outside> (optional)
```

When projecting a pre-existing mesh onto a design element, the most time consuming part of the process is the attempted projection of a node that is not contained within the design element (patch or volume). The use of a searchbox allows *SDESIGN* to skip the projection of any node located inside or outside of the user-defined searchbox, thus greatly speeding up the projection process when not all nodes are located within a design element.

Any number of searchbox definitions may be applied to a given design element.

- `<design element type>` See table in 3.2.17 for design element types.
- `<number of design element>` This is the number of the design element that the searchbox corresponds to. A searchbox definition only applies to one design element.
- `<min/max x/y/z>` This defines the boundaries of the searchbox in global coordinates. Any node that does not lie within these confines will not be located in the projection process. The user must make sure to enter the fields in the proper order, or else the searchbox will not be properly formed.
- `<inside or outside>` This field tells whether to project nodes that are located either inside or outside the searchbox. If the keyword **inside** is put here, only nodes inside the searchbox will be projected. If the keyword **outside** is put here, nodes inside the searchbox will be skipped. Note that nodes on the edges of a searchbox are considered to be inside the searchbox. This is an optional field, with the default being **inside**.

3.2.19 Defining Match-boxes

```
MATCHBOX  
<number> <min x> <max x> <min y> <max y> <min z> <max z>
```

If the program *MATCHER* will be used on the mesh generated by *SDESIGN*, the user may desire to include or exclude certain nodes from matching. Mesh nodes that lie within the designated match-box will be output to a control file (.cnbc) that will be read by *MATCHER*. See section 3.2.2 for more information regarding this output file.

- `<min/max x/y/z>` This defines the boundaries of the matchbox in global coordinates. The user must make sure to enter the fields in the proper order, or else the matchbox will not be properly formed.

3.2.20 Defining Boundary Conditions

The following boundary conditions are implemented: force, displacement, temperature, voltage. All require the same input, described below, except

description	type keyword
b.c. on a control node	node
b.c. on an edge for 2-D matching	edge
b.c. on an edge	matchededge
b.c. on a patch	surf
b.c. on a patch (2-d interface problems)	surf2d
b.c. on a patch for 3-D matching	matchsurf
b.c. on a patch for 3-D wet surface b.c	wetsurf
b.c. in a volume (only Dirichlet b.c.)	hexa

Table 3.10: Available geometry types for boundary conditions

for temperature and voltage boundary conditions, which do not require a degree of freedom. This field must be omitted when defining a temperature boundary condition, while all others remain the same.

```
FORCE, DISPLACEMENT, TEMPERATURE, VOLTAGE or SOFTVOLT
<number> <geometry type> <number of geometrical entity>
      <magnitude> <dof list> (except temp and volt )
      <patch nodes (optional)>
```

- **<geometry type>** There are 5 geometry types to which a boundary condition can be applied, summarized in the table 3.10. Note that a boundary condition cannot be applied to a control node without a corresponding finite element node, for example, an interior node on a non-straight b-spline edge does not correspond to a finite element node.
- **<number of geometrical entity>** This is the number in the design input file of the control node, edge or patch to which the boundary condition is to be applied.
- **<magnitude>** The numerical value of the boundary condition, positive or negative, given with regards to the proper units. For Dirichlet boundary conditions, the magnitude specified will simply be applied to every finite element node on the associated control node, edge or patch. For Neumann boundary conditions, the magnitude should be the distributed magnitude. The units are as follows, using a force boundary condition as an example:

description	units
b.c on a control node	force
b.c on an edge	force/length
b.c on a patch	force/length ²

- `<dof list>` A list of the degrees of freedom to which the boundary condition applies. This field must be skipped for temperature and voltage boundary conditions. The available options are as follows:

description	dof number	dof keyword
x-translational	1	xtran
y-translational	2	ytran
z-translational	3	ztran
x-rotational	4	xrot
y-rotational	5	yrot
z-rotational	6	zrot

Both numbers and keyword or a combination of both may be listed. However, if numbers are used, a comma must follow the magnitude specification. In addition to listing degrees of freedom individually, there are keywords for combinations of degrees of freedom, which cannot be listed, only one per boundary condition definition. These keywords are summarized below:

description	keyword
all x-dofs	allx
all y-dofs	ally
all z-dofs	allz
all translational dof	alltran
all rotational dof	allrot
all dofs	alldof

- `<patch nodes>` If the user encounters a situation where there are meshes from patches and volumes overlapping, the fe-nodes produced by the volume will be the ones to which the boundary conditions are applied if the patch and volume nodes are different. However, the user may for some reason find it desirable that the patch nodes, rather than the volume ones, should be used. If this is the case, the keyword **patchnode** should be used for this field. If the user would like to explicitly specify the default condition, the keyword **volumenode** should be used. This is an optional field so if this

situation does not apply or the user is satisfied with the default, than this field should be left blank.

3.2.21 Defining Structural Interfaces

```
INTERFACE
<number> <geometry type> <number of geometrical entity>
      <outer normal direction/boundary flag>
      <patch nodes (optional)>
```

- <geometry type>,<number of geometrical entity>,<patch nodes>
See information in 3.2.20.

- <outer normal direction/boundary flag>

- **2D**: Part of the calculation of electrical forces involves an outer normal. The orientation of the normal depends on the orientation of the interface elements. The normal will be calculated as 90° counter clockwise from the vector going from node 1 to node 2 of the interface element. The direction of node 1 to node 2 is determined by the direction from the first control node to the last control node of the edge that is defined as an interface. This field specifies whether the interface elements will be oriented from the first to the last control node, or vice versa. The following table gives the options:

description	direction keyword
reverse existing edge definition	backward
use existing edge definition	forward

- **3D**: The normal will be calculated as normal to the specified face. The direction of this normal is determined by using either the right or left hand rule on the orientation of defined edges for this specific patch. The following table gives the options:

description	direction keyword
left hand rule	lefthand
right hand rule	righthand

- **matchedgedge, matchsurf or wetsurf**: Instead of a normal direction, a boundary condition value must be given here. Table 3.11 lists the possibilities:

description	keyword	number
Internal face	internal	0
Symmetry condition for a non-moving face	symmetryfixed	1
Symmetry condition for a moving face	symmetryfree	-1
Slip condition for a non-moving face	slipfixed	2
Slip condition for a moving face	slipfree	-2
Stick condition for a non-moving face	stickfixed	3
Stick condition for a moving face	stickfree	-3
Inlet condition for a non-moving face	inletfixed	4
Inlet condition for a moving face	inletfree	-4
Outlet condition for a non-moving face	outletfixed	5
Outlet condition for a moving face	outletfree	-5

Table 3.11: Boundary conditions for matching and wet edges and surfaces

- `<patch nodes>` See this field in section 3.2.20.

3.2.22 Defining Mesh Motion Flags

```
MESHMOTION
<number> <geometry type> <number of geometrical entity>
      <nodal flag> <patch nodes (optional)>
```

- `<geometry type>`, `<number of geometrical entity>`, `<patch nodes>`
See information in 3.2.20.

- `<nodal flag>` This flag tells what freedom of movement a node has during mesh motion. Note that the nodal flag specified by an electric-structure interface is -2 , which means prescribed. This will supercede any definition made in this field. Also, if nodes are shared by multiple definitions, the last mesh motion definition will overwrite the previous ones. A list of options can be found in the following table.

description	freedom #	freedom keyword
no movement allowed	-5	fixed
prescribed	-2	prescribed
no constraint	0	free
x direction only	1	xonly
y direction only	2	yonly
z direction only	3	zonly
x and y directions only	4	xandy
x and z directions only	5	xandz
y and z directions only	6	yandz

3.2.23 Defining Variable Control Nodes

```

DSGVAR
<node number> <dx: number of variable>
                <dy: number of variable>
                <dz: number of variable>

```

- **<node number>** The control node number which is being defined as variable.
- **<dx\y\z: number of variable>** For each control node listed, it has to be specified whether the node is variable in global x- or y- or z-direction by assigning a unique variable number > 0 . Skipping variable numbers is not allowed. If the node is not movable, this is indicated by zero. For two-dimensional problems, the z-definition may be omitted.

3.2.24 Defining Variable Element Properties

```
ELEDSGVAR
<number> <design element type> <design element number>
<variable properties>
<element variable dependencies (optional)>
<one variable for all elements> (optional)
<no increment of variable/attribute> (optional)
<different design element var/att> (optional)
<use the variable specified> (optional)
<offset STCVAR value> (optional)
<number of elements per block> (optional)
<first element in a design element> (optional)
<maximum variable number> (optional)
<function>
```

Note: Any number and ordering of the optional fields is valid.

- **<design element type> <design element number>** This information takes care of whether or not this is an ATTR, COMP, ELCATT or THERMATT and the corresponding number, so only the part of the attribute that varies is necessary to input. For example, EMOD or LAYER 1 EMOD2. In order for this to work properly, the design element must be explicitly assigned attributes, see section 3.2.17
- **<variable properties>** The element property that should be varied for finite elements on this design element. A new definition is required for each additional property on this design element. See the information on element attributes in section 1.4.6 of the *FEM* - Optimization Module Users Guide for possible entries.
- **<element variable dependencies>** If the keyword **stcelv** is inserted here, the element variable dependencies will be generated for the optimization input file. See section 1.4.9 of the *FEM* - Optimization Module Users Guide for details. This is an optional field.
- **<one variable for all elements>** The default procedure is to assign each element to a different optimization variable, as is needed for topology optimization. If it desired to have all finite elements in the given design element dependent on the same variable, the keyword **samevar** should be

inserted. This is an optional field.

- `<no increment of variable/attribute>` Since all variable and attribute numbering is handled internally, the values will automatically be incremented for different design elements. If this is not desired, the keyword **noincr** should be used. Note that this will only effect the variable/attribute number of the next design element, not the current one. The result is that the next design element will have the same variable/attribute starting point as the one with **noincr** specified. If it is desired to not increment either variables or attributes individually, the keywords **noincrvar** or **noincratt** should be used, respectively. This is an optional field.

- `<different design element var/att>` By default, if different element attributes of the same design element are specified, they are both linked to the same attribute(s) and variable number(s). If this is not desired, the keyword **diffdel** should be used. If it is desired that only the attributes or variables should be different, while keeping the other the same, the keywords **diffdelatt** and **diffdelvar** should be used, respectively. This is an optional field.

- `<use the variable specified>` Since numbering is handled internally, the variable number specified is only a starting point. If it is desired that the actual variable number specified is used, the keyword **actvar** should be used. This is an optional field.

- `<offset STCVAR value>` If the **addextvar** option is being used to print variables to an external optimization input file, it may be desired to offset the numbering of the STCVAR section such that it allows for previously written variations. The syntax of this option is '**offstcvar** Integer'. This is an optional field.

- `<number of elements per block>` If the **blockelem** option is being used in the ATTRIBUTE section in order to link the element attributes of all elements in a 'block', for example, two tris that make a quad, the user needs to define how many elements are in this block such that the variable numbers are correctly output. The syntax is '**numblockelem** Integer'. This is an optional field.

- `<first element in a design element>` If it is desired that the param-

eterization of the finite element properties in a given design element should not start with the first element, the keyword **firstelenum** should be given, followed by the first element to be printed into the optimization input file. This is an optional field.

- **<maximum variable number>** This field limits the parameterization of element properties, such that a given variable number is not exceeded. The syntax is **maxnumvar**, followed by the maximum variable number desired. This is an optional field.

- **<function>** defines a function according to section 1.4.7 of the *FEM - Optimization Module Users Guide* where x_i can be only abstract variables VAR[i]. Note that unless otherwise specified through an option, that the variable number given here defines the starting point for the ELED SGVAR section and *SDESIGN* keeps track of all numbering internally. Generally, the same variable number will be given for all definitions in the ELED SGVAR section.

3.2.25 Defining Variable Node Properties

```
NODALDSGVAR
<number>      <geometry type> <number of geometrical entity>
               <variable properties>
               <different variable for all elements> (optional)
               <no increment of variable> (optional)
               <use the variable specified> (optional)
               <offset STCVAR value> (optional)
               <function>
```

Note: Any number and ordering of the optional fields is valid.

- **<geometry type>**, **<number of geometrical entity>** See information in 3.2.20. This does not work for match and wet geometry. This information takes care of whether or not this is an ATTR, COMP, ELCATT or THERMATT, and the corresponding number, so only the part of the material or composite attribute that varies is necessary to input. For example, EMOD or LAYER 1 EMOD2

- **<variable properties>** The element property that should be varied for finite elements on this design element. A new definition is required for each

additional property on this design element. See the information on element attributes in section 1.4.6 of the *FEM* - Optimization Module Users Guide for possible entries.

- **<different variable for all nodes>** The default procedure is to assign the variation of each nodal parameter to the same optimization variable. If it desired to give each node its own variable, the keyword **diffvar** should be inserted. This is an optional field.

- **<no increment of variable>** Since all variable and attribute numbering is handled internally, the values will automatically be incremented for different entries. If this is not desired, the keyword **noincr** should be used. Note that this will only effect the variable number of the next entry, not the current one. The result is that the next entry will have the same variable starting point as the one with **noincr** specified. This is an optional field.

- **<use the variable specified>** Since numbering is handled internally, the variable number specified is only a starting point. If it is desired that the actual variable number specified is used, the keyword **actvar** should be used. This is an optional field.

- **<offset STCVAR value>** If the **addextvar** option is being used to print variables to an external optimization input file, it may be desired to offset the numbering of the STCVAR section such that it allows for previously written variations. The syntax of this option is '**offstcvar** Integer'. This is an optional field.

- **<function>** defines a function according to section 1.4.7 of the *FEM* - Optimization Module Users Guide where x_i can be only abstract variables VAR[i]. Note that unless otherwise specified through an option, that the variable number given here defines the starting point for the NODALDSGVAR section and *SDESIGN* keeps track of all numbering internally. Generally, the same variable number will be given for all definitions in the NODALDSGVAR section.

3.2.26 Defining Link Macros

LINK <number of variable> <function>
--

Macros, or linking rules, are used to define the motion of the control nodes as functions of abstract optimization variables \mathbf{s} (see introduction of *FEM* - Optimization Module Users Guide). They have to be defined for *each* non-zero variable defined in the `DSGVAR` block. It should be noted that the keyword `OPTRELLINK` instead of `LINK` will allow this link to be used for both the reliability and optimization input file.

- `<function>` defines a function according to section 1.4.7 of the *FEM* - Optimization Module Users Guide where x_i can be only abstract variables `VAR[i]`.

3.2.27 Defining Initial Node and Element Numbers

```
INIT <id-number of first fe-node to be generated>  
      <id-number of first fe-element to be generated>
```

3.2.28 Using a Pre-existing Fe-mesh

```
FEMESH <filename of fe-mesh>
```

- The syntax of the fe-mesh file is:

```
FENODE  
<id-number fe-node> <coordinates x,y,z fe--node>  
...  
END
```

See section 3.3 for detailed information about projecting a pre-existing fe-mesh onto a design element model.

3.2.29 Using a Pre-existing Local Coordinate Mesh

```
LCMESH <filename of lc-mesh>
```

- A local coordinate mesh file is generated, by default, every time *SDESIGN* runs. The syntax of the lc-mesh file is:

```

LCNODE
<id-number fe-node>          <coordinates x,y,z fe--node>
...
LCLOC
<id-number local coordinate> <local coordinates r,s,t>
...
END

```

Use of the local coordinate mesh would be advantageous in the following instance. If the user runs *SDESIGN* in order to create the structural variable definitions (design velocity field) for the optimization module ('.opi') file and then realizes after that there was an error in the design velocity field, i.e. either the DSGVAR or LINK section of the input file. The most time consuming part of *SDESIGN* is the generation or projection of nodes in local coordinates. If the user needs to run *SDESIGN* again, and is not changing the definition of any design elements, using the local coordinate mesh that was output by the previous running of *SDESIGN* will allow the subsequent running to occur in a fraction of the time. There does not need to be any modification of the '.loc' file by the user, only the LCMESH definition in the input file.

3.3 Projecting a given fe-mesh onto design element mesh

Projecting existing fe-nodes onto design elements is sometimes tricky. The items below contain all the relevant information pertaining to this this topic.

- Existing nodes can be projected onto both patch and volume elements. Whether or not a given patch or volume is activated for projection depends on the input of that design element. Both design elements require the input of the number of elements along their local axes. In order for a design element to be activated for projection, these numbers must be set > 0 . Of course, all patches or volumes may be deactivated by using the -v and -p flags, respectively, on the command line.
- When projecting nodes onto a design element, every node in the input file will be searched for, which can significantly slow the process. In order to only project nodes within the desired design element, the user may define a searchbox (see 3.2.18), which will limit the projection area to the interior of the defined searchbox. This definition will greatly

increase the speed of node projection if many are located outside the scope of the desired design elements.

- When projecting nodes onto a patch, not just nodes in the plane of the patch are projected. All nodes within the normal projection of the patch are located and assigned local x and y-coordinates corresponding to those of the patch. The local z-coordinate is assigned to be zero because of the dimensionality of the patch element.
- When projecting nodes into a volume, only those contained within the volume will be found, nodes will not be projected onto the volume from outside.
- The local coordinates of the nodes being projected are located with a Newton solver. This problem is not generally smooth and therefore a random starting point does not always converge to the solution. The correction for this problem involves the use of a starting point for the Newton solver.

The starting point is determined by the generation of nodes on the design element of interest and a subsequent search through these nodes to find the node closest to the node being projected. The local coordinates of this node are used as the starting point for the Newton solver. The point of all this information is that a small amount of user input affects this starting point.

When the user inputs the number of elements in the local direction for either a patch or a volume, this information is used to generate the temporary nodes used for starting points. The more elements generated along each local coordinate, the better the starting point will be, with the trade-off being more nodes to search through and longer run-time. There must be enough nodes such that the starting point will allow convergence to the correct solution, but not too many to unnecessarily slow down the overall process.

A recommendation is 10 elements along all local axes, this should be sufficient without drastically affecting performance. If all nodes are not properly projected, *SDESIGN* should be run again with a greater number of elements along the local axes.

- If all nodes that should be projected still are not after the grid is sufficiently refined, there is one more solution. When a node position is located within a design element, it is given a local coordinate. If this

coordinate is within the design element, than it is projected. There is a tolerance for whether or not the local coordinate is considered within a design element. The default value is 1.0×10^{-4} , meaning that if a local coordinate was (1.000001, 0.5, 0.2) it would be projected onto a design element whose local coordinates are from 0 – 1, but a node with a local coordinate of (1.001, 0.5, 0.2) would not be.

If the design element onto which fe-nodes are being projected has a very strange shape, this tolerance may need to be made larger in order to capture all of the correct nodes. This is a result of the interpolation of Coons elements. See the discussion on TOLERANCE for information on how to do this.

- There are two other fields that relate to projection: ITERATION and ACCURACY. There is little need to change these unless the user has a specific purpose. Lowering the number of iterations, or making the convergence less accurate will speed run time, but at the risk of improper results. The current default values of these fields should be sufficient to obtain accurate results for most conceivable cases.

Chapter 4

Electro Mechanical Analysis, Optimization and Eigenvalues

4.1 Introduction

This manual will give an overview of the syntax required to run a coupled electro-mechanical analysis, optimization, eigenvalue or POD decomposition problem using *FEM*. The eigenvalue problem returns either a “load-factor” lambda (which should equal one at instability) or the lowest absolute value eigenvalue of the entire linearized system. The structural analysis and electrostatic analysis are both carried out using *FEM*. In the coupled process, two separate executables are run simultaneously, one for the structure and one for the electrostatic. The two programs communicate with each other throughout the process, therefore it is necessary to run them using the mpi environment.

In addition to the two previously mentioned analysis programs that will be run, a third one can be inserted as well: elastic mesh-motion. It is possible to use a simplistic Laplace smoothing technique for mesh-motion. In this case the electrostatic code will perform the mesh-motion. However, if an elastic stiffness is used to represent the “stiffness” of the mesh, this mesh-motion needs to be performed with a third copy of *FEM*.

For all *FEM*, *MATCHER* or *SDESIGN* commands that are referenced, see the respective manuals for documentation on their usage, as it will not be covered herein.

4.2 Structural Component

4.2.1 Analysis

This section will give an overview of the setup for the structural input file. All the normal inputs are valid for an electro-mechanical problem. The extra inputs necessary for electro-mechanical analysis are in the specification of the dynamic and “aeroelastic” algorithms and a matching file.

“Aeroelastic” algorithm

The A6 algorithm for coupled problems is the only one set up to work for an electro-mechanical problem. The specification is as follows:

```
AERO A6
```

Dynamic algorithm

An electro-mechanical problem is treated as a dynamic problem. Under the dynamic heading, either the quasi-static or newmark algorithm works. Also, the time integration and steady-state criteria need to be entered. A valid specification is as follows:

```
DYNAMIC
qstatic      0.75      0.75001  100000
time         0.500     0.500   1000.00
steady       1.0e-8    2        500
```

In order to run the coupled problem with a dynamic structure, the ‘qstatic’ line should be changed to

```
newmark       $\beta$        $\gamma$ 
```

The values for β and γ will be set to Newmark defaults if they are omitted.

Matching file

A file called “ELMATCHER,” which is produced by *MATCHER*, must exist. This file contains information that tells the structure which elements are matched to the electrostatic nodes.

These are the only additional necessary specifications for the structural input file in an electro-mechanical problem. Also, note that no FORCE specification is required, as the forces come from the electrostatic field.

4.2.2 Optimization

In order to optimize an electro-mechanical system, all of the normal optimization specifications are valid. The only additional requirement that is not present in a static optimization problem is the specification of parameters that define how the structure transfers displacements to the electrostatic field.

```
STCANALYSIS
STCTRANS 1 50 1.0 1.0 0
```

See the optimization module manual for information on the above field. Either the direct or adjoint method may be used for analytical sensitivities. Also, numerical sensitivities may be used if desired.

4.2.3 Eigenvalue

The only aspect that changes to run an eigen problem is the specification of the dynamic algorithm. The algorithm should be modified to 'eigenem.' Also, it should be stated which eigenvalue problem to run. This is done by putting either 'P1' (load factor for structural problem, recommended), 'IP' or 'IP0' (full linearized system about the analyzed or initial configuration, respectively) at the end of the line. A valid specification of the dynamic algorithm is below:

```
DYNAMIC

eigenem      0.5      0.5001  100000  P1
time         0.500    0.500   2000.00
steady       1.0e-8    2       3000
```

All else remains the same in comparison to pure analysis.

4.2.4 POD Decomposition

The only aspect that changes to run a POD decomposition problem is the specification of the dynamic algorithm. The algorithm should be modified to either '**qspodbasis**' or '**nmpodbasis**' (for quasi-static or newmark analysis, respectively). Also, it should be stated which decomposition is being run. This is done by putting 'DC' at the end of the line. This is currently the only decomposition implemented. A valid specification of the dynamic algorithm is below:

DYNAMIC

```
qspodbasis    0.75      0.75001  100000  DC
time          0.500     0.500   2000.00
steady        1.0e-8    2        3000
```

Also, the unconstrained basis vectors need to be specified in an input file that *FEM* will read. The file for the structural basis vectors must be called “STRCBASIS”. The file will look as follows:

```
3 2 0 360
1
structural basis vector 1
...

...
2
structural basis vector 2
...

...
3
structural basis vector 3
...

...
```

The first line of the file will contain four integers: the number of structural basis vectors, the number of electrostatic basis vectors and the number of fluid basis vectors (zero for an electro-mechanical problem) and the number of unconstrained degrees of freedom for the structure. For each vector, there must be a line containing the number of the structural basis vector and then the vector of the length specified follows, with all entries on separate lines. The structural decomposition will be output in the file “strcDecomp.m” in Matlab format. Any number of basis vectors can be used.

4.3 Electrostatic Component

4.3.1 Analysis

An electrostatic problem is driven by the application of voltage to the field. In general, voltage is applied at the electrode and where the electrostatic field

interfaces with the structure. The voltage is a Dirichlet boundary condition for the electrostatic problem.

The input file for the electrostatic allows all of the same inputs as for a structural problem. The following list will highlight the different inputs, as well as give the additional inputs beyond what is required for a pure electrostatic problem:

Input files

In addition to the *FEM* input file, the following files are needed for an electro-mechanical problem:

1. Sinus - this file is used by *MATCHER* in order to match the computational meshes of the structural and electrostatic domains. Specify whether or not this is a 2 or 3-d problem when requesting *SDESIGN* to output a sinus file.
2. Meshmotion - this output file is only required if Laplace smoothing is being used for mesh-motion. If it is required, it should be named "BCDATA" because that is the file that *FEM* searches for.
3. elc+00001 - this file informs the electrostatic field which of its nodes are matched to the structure. If the electro-mechanical analysis is parallelized, files named elc+xxxxxx will be allowed.

Output files

The following output files are available for electrostatics:

Physical quantity	Basename	Variants
Voltage	gvoltage	–
Electric field	efield_	x,y,z,g,n
Electric force	efrcstr_	x,y,z,g,n
Mesh-motion	elmshrlx	–

For electric field and electric forces, the underscore is replaced with one of the letters in the adjacent column. The letters refer to the direction (x,y,z), 'g' is global (all three) and 'n' is Euclidian "length" of the vector.

Element type

It is necessary to use electrostatic elements, either 4-node electrostatic quads (for 2-d problems) or 8-node electrostatic bricks (for 3-d problems).

Voltage b.c.

Currently, voltage is the only boundary condition that will drive an electrostatic problem. It is recommended that the voltage is applied in the following manner: for a voltage differential of 100 volts, for example, apply 100 volts to the electrostatic boundaries with the electrode and 0 volts to the interface with the structure. All other boundaries should be left free.

Structural interfaces

In order to compute electrostatic forces, which are transferred to the structure in order to compute structural displacements, it is necessary to define where the electrostatic field interfaces with the structure. The voltage specification of zero, discussed above, does not satisfy this requirement. An interface must be explicitly defined, using the `INTERFACE` command in *SDESIGN*. This will generate interface elements used in the computation of electrostatic forces.

Matching interfaces

In order to correctly output the sinus file, it needs to be specified which surfaces are matching surfaces. This is done in the same way as the structural interface command above, except ‘edge’ and ‘surf’ are replaced with ‘matchededge’ and ‘matchsurf.’

Material

The only important material properties for the electrostatic are the permittivity and the thickness (for 2-d problems). The permittivity is input in the same position as ‘k’, the heat conduction coefficient. There is a default material for vacuum in *SDESIGN*

Algorithm

The algorithm for electro-mechanical analysis is specified as follows:

```
ELECSTRC A6
```

Mesh-motion

First, the electrostatic fields needs to know whether Laplace smoothing or elastic mesh-motion will be used, this is done as follows:

```
MESHMOTSHEME laplace or elastic
```

If elastic mesh-motion is being used, this is the only necessary mesh-motion specification in the electrostatic. If Laplace smoothing is used, more information is required. When displacements are applied to the

mesh, they are broken up so that they are applied incrementally. The number of steps must be specified as follows (the default is 50 if nothing specified):

```
MESHMOTSTEP 10
```

Also, the boundary conditions for the mesh must be specified in the BCDATA file (output file 2 above). This specifies if any parts of the mesh are not free to move under mesh-motion, for example, the part of the mesh where the electrode is. This can be done using the MESH-MOTION command in *SDESIGN*.

4.3.2 Optimization

The only field that changes for electro-mechanical optimization is the algorithm specification. It becomes

```
OPTELECSTRC A6
```

All else remains the same. Note that if an adjoint sensitivity analysis is being used, elastic mesh-motion *must* be used.

4.3.3 Eigenvalue

The only field that changes for electro-mechanical eigenvalue analysis is the algorithm specification. It becomes

```
EIGELECSTRC P1 or IP or IPO
```

‘P1’ should produce an eigenvalue of 1 at the instability point (recommended). All else remains the same.

4.3.4 POD Decomposition

The only field that changes for electro-mechanical POD decomposition problem is the algorithm specification. It becomes

```
PODELECSTRC DC
```

Also, the unconstrained electrostatic basis vectors need to be specified in an input file that *FEM* will read. The file for the electrostatic basis vectors must be called “ELEC BASIS”. The file will look as follows:

```
3 2 0 192
```

```
1
```

```
electrostatic basis vector 1
```

```
...  
...  
2  
electrostatic basis vector 2  
...  
...
```

The first line of the file will contain four integers: the number of structural basis vectors, the number of electrostatic basis vectors, the number of fluid basis vectors and the number of unconstrained degrees of freedom for the electrostatic. For each vector, there must be a line containing the number of the electrostatic basis vector and then the vector of the length specified follows, with all entries on separate lines. The electrostatic decomposition will be output in the file “elecDecomp.m” in Matlab format. Any number of basis vectors can be used.

4.4 Mesh-motion Component

4.4.1 Analysis

If Laplace smoothing is being used for the electrostatic mesh-motion, this section may be ignored, as a separate input file for the mesh is not required. When creating the finite element mesh for the elastic mesh-motion, the *identical* geometry as the electrostatic problem should be used. The only parameter that needs to be changed is the element type. The element type needs to be a structural element, instead of an electrostatic element. All the standard *FEM* input is acceptable. The following list highlights the extra inputs for mesh-motion.

Geometry

As stated above, use the same geometry as the electrostatic mesh, but change the finite element type to a structural element.

Boundary conditions

Only displacement boundary conditions need to be specified. The displacement boundary conditions need to be specified for two different reasons:

1. The mesh is desired to not change shape, for example, where the electrode is located. This is the equivalent of the MESHMOTION

designation for Laplace smoothing. This should be done with the DISPLACEMENT command.

2. There is an interface with the structure. This interface must be specified with a Dirichlet boundary condition, so that the finite element problem is properly formed. This boundary condition can be specified with either the DISPLACEMENT or IFCDISPLACEMENT command. The actual magnitude is irrelevant, as it will be continuously overwritten internally. Note that if the adjoint method id being used, the IFCDISPLACEMENT designation *must* be given to the interfaces, or else the problem will not run properly. For all other sensitivity analysis types, it is irrelevant.

Algorithm

The algorithm that tells *FEM* do run a mesh-motion problem is specified as follows:

```
MESHALG A6
```

Matching

No additional files are needed for matching, as the mesh-motion uses the same matching files output by *MATCHER* for the electrostatic.

4.4.2 Optimization

In order to use mesh-motion with optimization, only the algorithm needs to be changed. It becomes

```
OPTMESHALG A6
```

It should be noted that for adjoint sensitivity analysis, elastic mesh-motion must be used. All else remains the same.

4.4.3 Eigenvalue

The only field that changes for electro-mechanical eigenvalue analysis is the algorithm specification. It becomes

```
EIGMESHALG P1 or IP or IPO
```

‘P1’ should produce an eigenvalue of 1 at the instability point (recommended). All else remains the same.

4.4.4 POD Decomposition

The only field that changes for electro-mechanical POD decomposition is the algorithm specification. It becomes

```
PODMESHALG DC
```

All else remains the same.

4.5 Matching

In order to run the coupled problem, the computational domains need to be matched, using *MATCHER*. The sinus file from the electrostatic and the structural *FEM* input file are matched. The files *ELMATCHER* (section 4.2.1) and *elc+00001* (section 4.3.1) are produced by *MATCHER* when a ‘-e’ flag is placed at the end of the input.

4.6 Examples

This section will give an example containing the following:

1. Structural, electrostatic and mesh-motion *SDESIGN* input files.
2. Syntax for running *SDESIGN*, *MATCHER* and *FEM*

4.6.1 Structural Input File

This following file is the structural input file for *SDESIGN*: “strc.dsg.”

```
#-----  
#  
#           Structural SDESIGN input file for testing  
#           electrostatic-structure interaction  
#           -6/13/02  
#  
#           FEM units - meters,kg  
#           dimensions - 500 x 2 x 50 microns  
#-----  
#  
SDOUTPUT  
  
topology  
feminput  
optinput  
local      noloc
```

NODE

1	0.0	2.0e-6	0.0
2	500.0e-6	2.0e-6	0.0
3	500.0e-6	2.0e-6	50.0e-6
4	0.0	2.0e-6	50.0e-6
5	0.0	4.0e-6	0.0
6	500.0e-6	4.0e-6	0.0
7	500.0e-6	4.0e-6	50.0e-6
8	0.0	4.0e-6	50.0e-6

EDGE

1	linear	2	1	2
2	linear	2	2	3
3	linear	2	3	4
4	linear	2	4	1
5	linear	2	5	6
6	linear	2	6	7
7	linear	2	7	8
8	linear	2	8	5
9	linear	2	1	5
10	linear	2	2	6
11	linear	2	3	7
12	linear	2	4	8

PATCH

1	coons	4	quad4	none	0	0	1	10	5	9
2	coons	4	quad4	none	0	0	2	11	6	10
3	coons	4	quad4	none	0	0	3	11	7	12
4	coons	4	quad4	none	0	0	4	12	8	9
5	coons	4	isotri	right	0	0	1	2	3	4
6	coons	4	isotri	none	50	10	5	6	7	8

DISPLACEMENT

1	edge	8	0.0	alldof
---	------	---	-----	--------

DSGVAR

5	1	0	0
6	2	0	0
7	3	0	0
8	4	0	0

LINK

```
1 SUM { 1.0 * VAR[1] }
2 SUM { -1.0 * VAR[1] }
3 SUM { -1.0 * VAR[2] }
4 SUM { 1.0 * VAR[2] }
```

```
#-----
# Info for the FEM code
#-----
```

```
FEMOUTPUT
gdisplac
```

```
MATERIAL
```

```
# A      E  nu rho      c  k      h      P  Ta  q  w  Ixx Iyy Izz
1 0.0 1.65e11 0.23 2230.0 0.0 0.0  2.0e-6  0.0 0.0 0.0 0.0 0.0 0.0 0.0
```

```
TRBM 0.0
```

```
AERO A6
```

```
DYNAMIC
```

```
qstatic 1.00000      1.0      1
time    0.125000    0.125000 1000.000000
steady  1.000000e-08 1      100
```

```
#-----
# Optimization info
#-----
```

```
CRITERIA
```

```
1 DISP node 7 DY
2 DISP node 6 DY
```

```
ABSVAR
```

```
1 0.0 1.0 -1.0 1.0 2
```

```
OBJECTIVE
```

```
1.0 * SUM { 1.75e2 * CRT[2] }
```

```
CONSTRAINT
```

```
1 IEQ 1.0 * SUM { -1.4e0 * CRT[1] + 1.0 }
```

```
SOLVER
```

```

1      NLP

NLPITR  1
NLPFCL  10
NLPSCB  1.0e4
NLPLDL  1
NLPACC  1.0e-8
NLPPRN  2
NLPFIT  0

GRDTYP  ANALYTIC
GRDMTH  DIRECT

STCANALYSIS

STCTRANS  1  50  0.25  0.25  0

```

```

#-----
END

```

4.6.2 Electrostatic Input File

This following file is the electrostatic input file for *SDESIGN*: "elec.dsg."

```

#-----
#
#      Electrostatic SDESIGN input file for
#      analytical beam 3-D electroelastic test problem
#      -5/30/02
#
#      FEM units - meters,kg
#      dimensions - 500 x 2 x 50 microns
#-----
#
SDOUTPUT

topology  nodset  enset  eleset  e
feminput
local     noloc
meshmotion  BCDATA
sinus     sindim  3

NODE

1      0.0      0.0e-6  0.0
2      500.0e-6  0.0e-6  0.0

```

```
3  500.0e-6  0.0e-6  50.0e-6
4    0.0      0.0e-6  50.0e-6
5    0.0      2.0e-6   0.0
6  500.0e-6  2.0e-6   0.0
7  500.0e-6  2.0e-6  50.0e-6
8    0.0      2.0e-6  50.0e-6
```

EDGE

```
1  linear  2  1  2
2  linear  2  2  3
3  linear  2  3  4
4  linear  2  4  1
5  linear  2  5  6
6  linear  2  6  7
7  linear  2  7  8
8  linear  2  8  5
9  linear  2  1  5
10 linear  2  2  6
11 linear  2  3  7
12 linear  2  4  8
```

PATCH

```
1  coons  4  quad4 none  0  0  1 10 5 9
2  coons  4  quad4 none  0  0  2 11 6 10
3  coons  4  quad4 none  0  0  3 11 7 12
4  coons  4  quad4 none  0  0  4 12 8 9
5  coons  4  quad4 none  0  0  1 2 3 4
6  coons  4  quad4 none  0  0  5 6 7 8
```

VOLUME

```
1  coons  6 12 8  estatbrick8  10  50 3  1 2 3 4 5 6
```

DEFINE

```
vlt = 0.5
```

VOLTAGE

```
1  surf  5  vlt
2  surf  6  -vlt
```

ESINTERFACE

```
1  surf      6  righthand
2  matchsurf 6  slipfree
```

MESHMOTION

```

1 surf 5 fixed

#-----
# FEM info
#-----

CONTROL
elec
1
enset
easet

FEMOUTPUT
efrcstrn
gvoltage
elmshrlx
efieldg
efieldn

MATERIAL
1 vacuum metric

OPTELECSTRC A6

MESHMOTSCHHEME elastic

#-----

END

```

4.6.3 Mesh-motion Input File

This following file is the mesh-motion input file for *SDESIGN*: "mesh.dsg."

```

#-----
#
#           Electrostatic SDESIGN input file for
#           analytical beam 3-D electroelastic test problem
#           -5/30/02
#
#           FEM units - meters,kg
#           dimensions - 500 x 2 x 50 microns
#
#-----
#
SDOUTPUT

topology      nodset mnset eleset m

```

feminput
local noloc

NODE

1	0.0	0.0e-6	0.0
2	500.0e-6	0.0e-6	0.0
3	500.0e-6	0.0e-6	50.0e-6
4	0.0	0.0e-6	50.0e-6
5	0.0	2.0e-6	0.0
6	500.0e-6	2.0e-6	0.0
7	500.0e-6	2.0e-6	50.0e-6
8	0.0	2.0e-6	50.0e-6

EDGE

1	linear	2	1	2
2	linear	2	2	3
3	linear	2	3	4
4	linear	2	4	1
5	linear	2	5	6
6	linear	2	6	7
7	linear	2	7	8
8	linear	2	8	5
9	linear	2	1	5
10	linear	2	2	6
11	linear	2	3	7
12	linear	2	4	8

PATCH

1	coons	4	quad4	none	0	0	1	10	5	9
2	coons	4	quad4	none	0	0	2	11	6	10
3	coons	4	quad4	none	0	0	3	11	7	12
4	coons	4	quad4	none	0	0	4	12	8	9
5	coons	4	quad4	none	0	0	1	2	3	4
6	coons	4	quad4	none	0	0	5	6	7	8

VOLUME

1 coons 6 12 8 brick8 10 50 3 1 2 3 4 5 6

DISPLACEMENT

1 surf 5 0.0 alltran

IFCDISPLACEMENT

1 surf 6 0.0 alltran

```

#-----
# FEM info
#-----

CONTROL
mesh
1
mnset
meset

FEMOUTPUT
gdisplac

MATERIAL

1 silic metric

OPTMESHALG A6

#-----

END

```

4.6.4 Syntax for running problem

The input files above represent all of the necessary information to run the problem. The following sequence of commands will do so:

1. Run *SDESIGN* with all three input files:

```
sdesign strc.dsg elec.dsg mesh.dsg
```

2. Run *MATCHER*:

```
matcher elec.sin strc.input -d 3 -e
```

The ‘-e’ option will output the “ELMATCHER” and “elc+00001” files with the appropriate names. If the version of *MATCHER* being used does not support this option, then the files “strc.match00” and “flu+00001” must be copied to the above names, respectively.

3. Run the *FEM* files.

On Linux machines, this can be done by creating a file called “scheme,” for example. For this problem, scheme will look as follows:

```
-c 1 n0 fem.stc_mp_opt strc.input
```

```
-c 1 n0 fem.elc_mp_opt elec.input  
-c 1 n0 fem.msh_mp_opt mesh.input
```

Once this file is created, use the following command:

```
mpirun scheme
```

First make sure that the lam environment is running with the lamboot command in Linux.

Chapter 5

Examples

5.1 FEM Optimizaton examples

For some simple examples, input files of *FEM* and the optimization module are subsequently given.

5.1.1 2-Dimensional Cantilever

Objective	Structural Mass
Constraints	Vertical displacement in the lower right corner ≥ -0.02
Variables	Shape of upper edge and overall thickness

The geometrical and material data of the initial structure is given in fig.5.1 and the subsequent table.

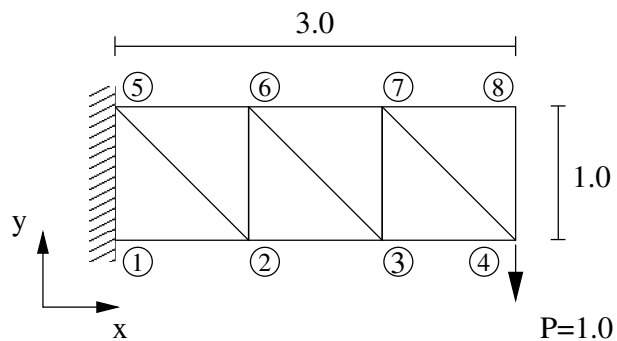


Figure 5.1: Example 1 - 2-Dimensional Cantilever

Young's modulus = 10^5
Poisson ration = 0.3
thickness = 0.1
density = 1.0

The input file describing the initial design for *FEM* is:

```
*
*-----
*
*   FEM -Input-file: Example 1
*
*-----
*
CONTROL
example1
1
nset
eset
*
STRUCTOPT example1.oinp
*
STATICS
skyline
*
RENUM
rcm
*
*-----
*
OUTPUT
gdisplac example1.dis 1
stressvm example1.von 1
thicknes example1.thk 1
shapeatt example1.sha 1
shapestc example1.shc 1
*
*-----
*
MATERIALS
1 0.0 1.0e+5 0.30 1.0 0.0 0.0 0.1 0.0 0.0 0.0 0.0 0.0 0.0 0.0
*
*-----
*
NODES
1 0.0 0.0 0.0
2 1.0 0.0 0.0
3 2.0 0.0 0.0
4 3.0 0.0 0.0
5 0.0 1.0 0.0
6 1.0 1.0 0.0
7 2.0 1.0 0.0
8 3.0 1.0 0.0
*
*-----
```

```

*
TOPOLOGY
1 8 1 2 5
2 8 2 6 5
3 8 2 3 6
4 8 3 7 6
5 8 3 4 7
6 8 4 8 7
*
*-----
*
ATTRIBUTES
1 6 1
*
*-----
*
DISPLACEMENT
*
1 1 0.000000
1 2 0.000000
1 3 0.000000
1 4 0.000000
1 5 0.000000
1 6 0.000000
5 1 0.000000
5 2 0.000000
5 3 0.000000
5 4 0.000000
5 5 0.000000
5 6 0.000000
*
*-----
*
FORCE
4 2 -1.000000
*
*-----
*
END

```

The input file *example1.oinp* for the optimization module is:

```

#
#-----
#
# Input-File for Optimization - Module
#
# Example 1: Cantilever
#
# Objective: Mass
# Constraint: Displacement
# Variable: Shape + Thickness
#
#-----
#
CRITERIA

```

```

#
# number      type      node/frq.    dof
#
#       1      MASS
#       2      DISP      4          DY
#
#-----
#
# ABSVAR
#
# number  value    scl    lower    upper
#
#       1     0.0    1.0    -0.90    2.00
#       2     0.0    1.0    -0.90    2.00
#       3     0.0    1.0    -0.05    0.05
#
#-----
#
# OBJECTIVE
#
# factor function
#
# 1.0 * SUM { CRT[1] }
#
#-----
#
# CONSTRAINT
#
# number  type  factor  func-type
#
#       1   IEQ   1.0 *   SUM { CRT[2] + 0.02 }
#
#-----
#
# STCVAR
#
# number  type      nod  dof    func-type
#
#       1   COOR     5   DY    SUM {          VAR[1] + 1.0 }
#       2   COOR     6   DY    SUM { 0.66 * VAR[1] + 1.0
#                               0.33 * VAR[2]
#                               }
#       3   COOR     7   DY    SUM { 0.33 * VAR[1] + 1.0
#                               0.66 * VAR[2]
#                               }
#       4   COOR     8   DY    SUM {          VAR[2] + 1.0 }
#
#       5   ATTR     1   THICK SUM {          VAR[3] + 0.1 }
#
#-----
#
# SOLVER
#
# number  type
#
#       1   NLP
#
# NLPITR 100

```

```

NLPFCL 10
NLPSCB 1.0D+4
NLPLDL 1
NLPACC 1.0D-8
NLPPRN 2
NLPFIT 0
#
GRDTYP ANALYTIC
GRDMTH DIRECT
#
#-----
#
END

```

Remarks:

- For the sake of simplicity the optimization problem is reduced to three abstract optimization variables. Two are linked to the vertical positions of the fe-nodes. The third corresponds to the overall thickness.
- The upper edge is assumed to be a linear Lagrange curve. This simple geometric model leads to the following linking between the abstract and structural variables,i.e. the vertical position of the fe-nodes on the upper edge:

$$\begin{aligned}
\Delta y_5 &= s_1 \\
\Delta y_6 &= \frac{2}{3}s_1 + \frac{1}{3}s_2 \\
\Delta y_7 &= \frac{1}{3}s_1 + \frac{2}{3}s_2 \\
\Delta y_8 &= s_2
\end{aligned}$$

5.1.2 3-Dimensional Cantilever

Objective	Structural Mass
Constraints	Vertical displacement in the lower right corners ≥ -0.1 Von Mises stress at lower left corners ≤ 1000 Lengthwise edges connecting to supports must have horizontal tangents
Variables	Shape of all surfaces

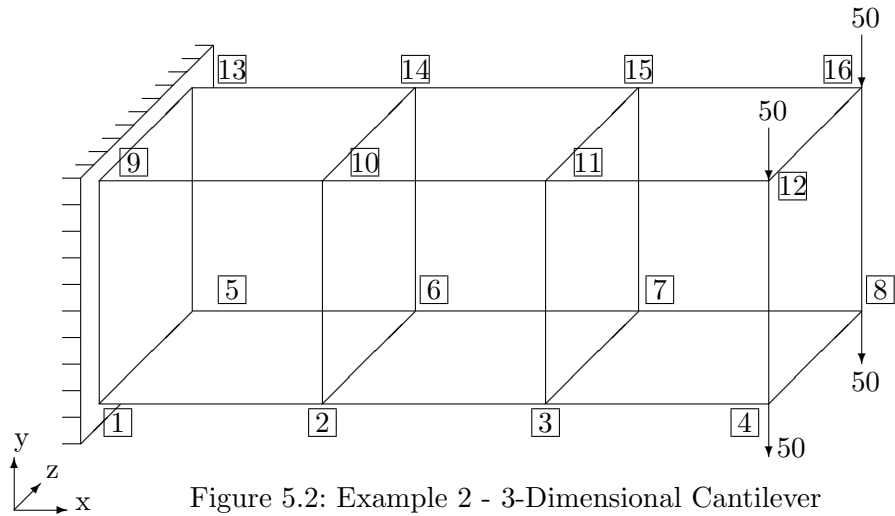


Figure 5.2: Example 2 - 3-Dimensional Cantilever

The geometrical and material data of the initial structure is given in fig.5.2 and the subsequent table.

Young's modulus = 10^5
Poisson ration = 0.3
density = 1.0

The design file *cbeam.dsg* executed by *SDESIGN* is located in chapter 5.2.2. The *cbeam.input* file created by *SDESIGN* was used by *FEM*. The *cbeam.oinp* file created by *SDESIGN* was used the *FEM* optimization module. The input file *cbeam.input* describing the initial design for *FEM* follows:

```
*
*-----
*
*   FEM -Input-file: Example 2
*
*-----
*
CONTROL
cbeam
1
nset
eset
*
STRUCTOPT cbeam.oinp
*
STATICS
skyline
*
RENUM
```

rcm

*

*

OUTPUT

gdisplac cbeam.dis 1

stressvm cbeam.von 1

shapeatt cbeam.sha 1

shapestc cbeam.shc 1

*

*

MATERIALS

1 0.0 1.0e+5 0.30 1.0 0.0 0.0 0.0 0.1 0.0 0.0 0.0 0.0 0.0 0.0

*

*

NODES

1	0.000000	0.000000	0.000000
2	3.303704	0.000000	0.000000
3	6.629630	0.000000	0.000000
4	10.000000	0.000000	0.000000
5	0.000000	0.000000	3.000000
6	3.303704	0.000000	3.000000
7	6.629630	0.000000	3.000000
8	10.000000	0.000000	3.000000
9	0.000000	3.000000	0.000000
10	3.303704	3.000000	0.000000
11	6.629630	3.000000	0.000000
12	10.000000	3.000000	0.000000
13	0.000000	3.000000	3.000000
14	3.303704	3.000000	3.000000
15	6.629630	3.000000	3.000000
16	10.000000	3.000000	3.000000

*

*

TOPOLOGY

1 17 1 5 6 2 9 13 14 10
2 17 2 6 7 3 10 14 15 11
3 17 3 7 8 4 11 15 16 12

*

*

ATTRIBUTES

1 3 1

*

*

DISPLACEMENTS

1	1	0.000000
1	2	0.000000
1	3	0.000000
5	1	0.000000
5	2	0.000000
5	3	0.000000

```

9   1 0.000000
9   2 0.000000
9   3 0.000000
13  1 0.000000
13  2 0.000000
13  3 0.000000
*
*-----
*
FORCE
4   2 -50.000000
8   2 -50.000000
12  2 -50.000000
16  2 -50.000000
*
*-----
*
END

```

The input file *cbeam.oinp* for the optimization module is:

```

#
#-----
#
# Input-File for Optimization - Module
#
# Example 2: 3-Dimensional Cantilever
#
# Objective: Mass
# Constraint: Displacement
# Variable: Shape
#
#-----
#
CRITERIA
  1  MASS
  2  DISP   4  DY
  3  DISP   8  DY
  4  STRESS  1  VMM
  5  STRESS  5  VMM
#
#-----
#
ABSVAR
#
# number  value    scl    lower    upper
#
  1      0.0      1.0    -1.5     1.5
  2      0.0      1.0    -1.5     1.5
  3      0.0      1.0    -1.5     1.5
  4      0.0      1.0    -1.5     1.5
  5      0.0      1.0    -1.5     1.5
  6      0.0      1.0    -1.5     1.5
#
#
#-----

```

```

#
OBJECTIVE
#
# factor      function
#
0.0005555 * SUM { CRT[1] }
#
#-----
#
CONSTRAINT
#
# number  type  factor  func-type
#
1      IEQ    1.0 *    SUM { 10.0 * CRT[2] + 1.0 }
2      IEQ    1.0 *    SUM { 10.0 * CRT[3] + 1.0 }
3      IEQ    1.0 *    SUM { -0.001 * CRT[4] + 1.0 }
4      IEQ    1.0 *    SUM { -0.001 * CRT[5] + 1.0 }
#
#-----
#
STCVAR
1      COOR  1  2  SUM {
DEF_OPR[1] = SUM {
1.00000000e+00 * VAR[1]^ 1.00000000e+00 + 0.00000000e+00
}
1.00000000e+00 * OPR[1] + 0.00000000e+00
}

... omitted for space considerations

16     COOR  16  2  SUM {
DEF_OPR[1] = SUM {
1.00000000e+00 * VAR[4]^ 1.00000000e+00 + 0.00000000e+00
}
1.00000000e+00 * OPR[1] + 3.00000000e+00
}
#
#-----
#
SOLVER
#
# number  type
#
1      NLP
#
NLPITR  50
NLPFCL  10
NLPSCB  1.0D+4
NLPLDL  1
NLPACC  1.0D-8
NLPPRN  2
NLPFIT  0
#
GRDTYP  ANALYTIC
GRDMTH  ADJOINT

```

```
#-----  
#  
#  
END
```

Remarks:

- For simplicity, the beam's shape is allowed to be modified only in the global y-direction. This requirement is stated in the DSGVAR section of the *cbeam.dsg* file.
- A horizontal tangent at the support is forced by requiring that the two control nodes of the lengthwise edges closest to the supports are modified identically. This functionality is allowed by the use of Bézier splines.

5.2 *SDESIGN* Examples

5.2.1 FE-Mesh Generation using a Patch Design Element

The following is an example of the use of *SDESIGN* to generate a structured finite element mesh with triangular shell elements, as well as generate the functional dependencies of the finite element nodes on the control nodes (design velocity field), in order to formulate an optimization problem.

```
# Optional output files for sdesign  
#  
SDOUTPUT  
  
topology  
feminput  
deselement  
  
NODE  
  
# Note that z-coordinates may be omitted  
  
1  0.0  0.0  0.0  
2  3.0  0.0  
3  3.0  1.0  0.0  
4  0.0  1.0
```

EDGE

```
1 linear 2 1 2
2 linear 2 2 3
3 linear 2 3 4
4 linear 2 4 1
```

PATCH

```
1 coons isotri right 3 1 1 2 3 4
```

DSGVAR

```
# nodes 1 & 2 omitted because they are not variable
```

```
3 0 1 0
4 0 2
```

LINK

```
1 SUM { VAR[2] }
2 SUM { VAR[1] }
```

END

5.2.2 FE-Mesh Generation using a Volume Design Element

The following is an example of the use of *SDESIGN* to generate a structured finite element mesh with 3-D brick elements, as well as generate the functional dependencies of the finite element nodes on the control nodes, in order to formulate an optimization problem. This is the input file used for the cantilever beam shape optimization problem described in section 5.1.2.

```
# Optional output files for sdesign
SDOUTPUT
```

```
topology
feminput
optinput
```

NODE

```
1  0.0  0.0  0.0
2  10.0  0.0  0.0
3  10.0  0.0  3.0
4   0.0  0.0  3.0
5   0.0  3.0  0.0
6  10.0  3.0  0.0
7  10.0  3.0  3.0
8   0.0  3.0  3.0
9   3.3  0.0  0.0
10  6.6  0.0  0.0
11  6.6  0.0  3.0
12  3.3  0.0  3.0
13  3.3  3.0  0.0
14  6.6  3.0  0.0
15  6.6  3.0  3.0
16  3.3  3.0  3.0
```

EDGE

```
1  bezier  4  1  9  10  2
2  linear  2  2  3
3  bezier  4  3  11  12  4
4  linear  2  4  1
5  linear  2  2  6
6  linear  2  3  7
7  linear  2  4  8
8  linear  2  1  5
9  bezier  4  5  13  14  6
10 linear  2  6  7
11 bezier  4  7  15  16  8
12 linear  2  8  5
```

PATCH

```
# Note that the number of elements is set to zero
# because a volume is being used and only brick
# elements are desired
# also, right or left tri definition is omitted because
# quad elements are being used
```

```
1 coons quad4 0 0 4 7 12 8
2 coons quad4 0 0 2 6 10 5
3 coons quad4 0 0 1 5 9 8
4 coons quad4 0 0 3 6 11 7
5 coons quad4 0 0 1 2 3 4
6 coons quad4 0 0 9 10 11 12
```

VOLUME

```
# note the shorthand 1:6 means 1 2 3 4 5 6
```

```
1 coons brick8 3 1 1 1:6
```

ATTRIBUTE

```
1 volume 1 2
```

```
#-----
# define boundary conditions
#-----
```

FORCE

```
1 edge 1 100.0 xtran zrot
2 surf 2 -32.4 1 2 3 5 6
3 node 6 81.0 3
```

DISPLACEMENT

```
1 node 5 85.1 allx
2 edge 2 20.0 alldof
3 2 1 10.0 1 xrot 4 ztran
```

```
#-----
# parameterize geometry
#-----
```

DSGVAR

```
1 0 1
2 0 2
```

```
3 0 3
4 0 4
5 0 5
6 0 6
7 0 7
8 0 8
9 0 9
10 0 10
11 0 11
12 0 12
13 0 13
14 0 14
15 0 15
16 0 16
```

LINK

```
1 SUM { VAR[1] }
2 SUM { VAR[2] }
3 SUM { VAR[2] }
4 SUM { VAR[1] }
5 SUM { VAR[3] }
6 SUM { VAR[4] }
7 SUM { VAR[4] }
8 SUM { VAR[3] }
9 SUM { VAR[1] }
10 SUM { VAR[5] }
11 SUM { VAR[5] }
12 SUM { VAR[1] }
13 SUM { VAR[3] }
14 SUM { VAR[6] }
15 SUM { VAR[6] }
16 SUM { VAR[3] }
```

```
#-----
# Some information for FEM
#-----
```

FEMOUTPUT
gdisplac

MATERIAL

1 0.0 3e7 0.30 1.0 0.0 0.0 0.1 0.0 0.0 0.0 0.0 0.0 0.0
2 0.0 2e5 0.25 1.0 0.0 0.0 0.1 0.0 0.0 0.0 0.0 0.0 0.0

#-----
Some information for FEM - optimization module
#-----

CRITERIA

1	MASS		
2	DISP	4	DY
3	DISP	8	DY
4	STRESS	1	VMM
5	STRESS	5	VMM

ABSVAR

#	number	value	scl	lower	upper
1	0.0	1.0	-1.5	1.5	
2	0.0	1.0	-1.5	1.5	
3	0.0	1.0	-1.5	1.5	
4	0.0	1.0	-1.5	1.5	
5	0.0	1.0	-1.5	1.5	
6	0.0	1.0	-1.5	1.5	

OBJECTIVE

0.0005555 * SUM { CRT[1] }

CONSTRAINT

1	IEQ	1.0 *	SUM { 10.0 * CRT[2] + 1.0 }
2	IEQ	1.0 *	SUM { 10.0 * CRT[3] + 1.0 }
3	IEQ	1.0 *	SUM { -0.001 * CRT[4] + 1.0 }
4	IEQ	1.0 *	SUM { -0.001 * CRT[5] + 1.0 }

SOLVER

1 NLP

NLPITR 50
NLPFCL 10
NLPSCB 1.0D+4
NLPLDL 1
NLPACC 1.0D-8
NLPPRN 2
NLPFIT 0

GRDTYP ANALYTIC
GRDMTH ADJOINT

#-----
END

5.2.3 FE-Mesh Projection onto a Volume Design Element

The following is an example of the use of *SDESIGN* to project a given finite element mesh onto a volume design element in order as generate the functional dependencies of the finite element nodes on the control nodes of the design element.

SDOUTPUT
optinput

NODE

1	0.0	0.0	0.0
2	1.0	0.0	0.0
3	1.0	1.0	0.0
4	0.0	1.0	0.0
5	0.0	0.0	1.0
6	1.0	0.0	1.0
7	1.0	1.0	1.0
8	0.0	1.0	1.0
9	0.4	0.0	0.0
10	1.0	0.3	0.0
11	0.6	1.0	0.0

12	0.0	0.7	0.0
13	1.0	0.0	0.4
14	1.0	1.0	0.6
15	0.0	1.0	0.5
16	0.0	0.0	0.7
17	0.3	0.0	1.0
18	1.0	0.55	1.0
19	0.45	1.0	1.0
20	0.0	0.4	1.0

EDGE

1	quadratic	3	1	9	2
2	quadratic	3	2	10	3
3	quadratic	3	3	11	4
4	quadratic	3	4	12	1
5	quadratic	3	2	13	6
6	quadratic	3	3	14	7
7	quadratic	3	4	15	8
8	quadratic	3	1	16	5
9	quadratic	3	5	17	6
10	quadratic	3	6	18	7
11	quadratic	3	7	19	8
12	quadratic	3	8	20	5

PATCH

1	coons	quad4	0	0	4	7	12	8
2	coons	quad4	0	0	2	6	10	5
3	coons	quad4	0	0	1	5	9	8
4	coons	quad4	0	0	3	6	11	7
5	coons	quad4	0	0	1	2	3	4
6	coons	quad4	0	0	9	10	11	12

VOLUME

1	coons	brick8	5	5	5	1	3	5	2	4	6
---	-------	--------	---	---	---	---	---	---	---	---	---

SEARCHBOX

This limits projection to nodes located in one

quadrant of the volume

1 volume 1 0 .5 0 .5 0 .5

FEMESH test.nodes

END

The following is the file test.nodes:

FENODE

1	0.00	0.0	0.00
2	0.00	0.4	0.50
3	0.60	0.0	0.75
4	0.40	0.0	0.21
5	0.70	0.3	0.00
6	0.40	0.0	0.10
7	0.10	0.0	0.00
8	0.01	0.4	0.50
9	0.62	0.9	0.75
10	0.14	0.0	0.21
11	0.71	0.3	0.00
12	0.14	0.0	0.10

END

Bibliography

- [1] Schittkowski, K., “NLPQL: A FORTRAN subroutine for solving constrained nonlinear programming problems,” *Annals of Operations Research*, Vol. 5, 1985, pp. 485–500.
- [2] Gill, P., Saunders, M., and Murray, W., “SNOPT: An SQP algorithm for large-scale constrained optimization,” Report na 97-2m department of mathematics, university of california, san diego, 1997.

optguide