Using the ALC matlab toolbox with input files generated from Psi specifications

Systems Engineering Group
Department of Mechanical Engineering
Eindhoven University of Technology
PO Box 513
5600 MB Eindhoven
The Netherlands
http://se.wtb.tue.nl/

SE Report: Nr. 2009-05

Using the ALC matlab toolbox with input files generated from Ψ specifications

S. Tosserams, A.T. Hofkamp, L.F.P. Etman, J.E. Rooda

ISSN: 1872-1567

SE Report: Nr. 2009-05
Eindhoven, June 2009
SE Reports are available via http://se.wtb.tue.nl/sereports

Contents

Chapter 1
Introduction

Distributed optimization is a technique to partition a single, typically large system optimization problem into a number of smaller optimization subproblems. A coordination algorithm is used to drive the subproblem designs towards a solution that is optimal for the original problem.

Distributed optimization approaches are attractive for addressing the challenges that arise in the optimal design of advanced engineering systems. The main motivation for the use of distributed optimization is the organization of the design process itself. Since a single designer is not able to oversee each relevant aspect, the design process is commonly distributed over a number of design teams. Each team is responsible for a part of the system, and typically uses specialized analysis and design tools to solve its design subproblems. Distributed optimization methods apply naturally to such organizations since they provide a degree of decision autonomy to the different design teams. Full disciplinary autonomy can rarely be obtained completely since the disciplinary design subproblems involve some quantities from other disciplines related to the interdisciplinary interaction. A coordination method is required to address these interactions.

Augmented Lagrangian Coordination (ALC) [14] has been proposed as a method for coordination of these interaction. This user manual describes how the matlab toolbox of augmented Lagrangian coordination (ALC) can be used with input files generated from specifications in the Ψ format [15]. It includes a brief description of ALC, a description of the matlab toolbox for ALC, and how to generate matlab input files from partitioned problem specifications in Ψ. Several examples are included for illustration.

Chapter 2
Augmented Lagrangian coordination

The starting point for the augmented Lagrangian coordination (ALC) method [14] is the system optimization problem given by:

$$\begin{array}{ll} \min\limits_{\mathbf{y},\mathbf{x}_1,\ldots,\mathbf{x}_M} & f_0(\mathbf{y},\mathbf{x}_1,\ldots,\mathbf{x}_M)+\sum\limits_{j=1}^{M} f_j(\mathbf{y},\mathbf{x}_j) \\ \text{subject to} & \mathbf{g}_0(\mathbf{y},\mathbf{x}_1,\ldots,\mathbf{x}_M)\le\mathbf{0} \\ & \mathbf{h}_0(\mathbf{y},\mathbf{x}_1,\ldots,\mathbf{x}_M)=\mathbf{0} \\ & \mathbf{g}_j(\mathbf{y},\mathbf{x}_j)\le\mathbf{0} \qquad j=1,\ldots,M \\ & \mathbf{h}_j(\mathbf{y},\mathbf{x}_j)=\mathbf{0} \qquad j=1,\ldots,M \end{array} \tag{2.1}$$

where the system consists of M disciplines labeled $j = 1,\ldots,M$. The local variables $\mathbf{x}_j$ are associated exclusively with discipline j, and the linking variables denoted by $\mathbf{y}$ are relevant to two or more disciplines. Similarly, objective and constraint functions are divided into coupling functions f_0, $\mathbf{g}_0$, and $\mathbf{h}_0$ that depend on the local variables of more than one discipline, and local functions f_j, $\mathbf{g}_j$, and $\mathbf{h}_j$ that depend on only one subset of local variables.

1 Subproblem formulation

The augmented Lagrangian coordination algorithms require that all constraints are separable with respect to the variables of the individual disciplines (i.e. depend only on the variables of a single discipline); coupling through the objective function is allowed. To remove the coupling of the constraints through the linking variables $\mathbf{y}$ and coupling constraints $\mathbf{g}_0$ and $\mathbf{h}_0$, two problem transformation steps are used:

Step 1: Introduction of linking variable copies $\mathbf{y}_j$ as design variables at each discipline, and the introduction of consistency constraints $\mathbf{c}$ that force the introduced copies to take equal values $\mathbf{y}_1 = \mathbf{y}_2 = \ldots = \mathbf{y}_M$. The local constraints $\mathbf{g}_j$ and $\mathbf{h}_j$ then only depend on the variables $\mathbf{y}_j$ and $\mathbf{x}_j$ of discipline j. However, the introduced consistency constraints $\mathbf{c}(\mathbf{y}_1,\mathbf{y}_2,\ldots,\mathbf{y}_M)$ depend on the linking variables of more than one discipline, and are therefore nonseparable coupling constraints, similar to $\mathbf{g}_0$ and $\mathbf{h}_0$. In general, the linking

constraints between subproblem j and its neighbor n is given by:

$$\mathbf{c}_{jn} = S_{jn}\mathbf{y}_j - S_{nj}\mathbf{y}_n = \mathbf{0} \qquad j = 1, \ldots, M, \qquad n \in \mathcal{N}_j \tag{2.2}$$

where the binary selection matrix S_{jn} selects those copies of subproblem j that are linked to the selected copies of its neighbor n, and $\mathcal{N}_j$ are the set of neighbors to which subproblem j is coupled through the consistency constraints $\mathbf{c}_{jn}$.

Step 2: Relaxation of the linking constraints $\mathbf{q} = [\mathbf{c}, \mathbf{g}_0 + \mathbf{x}_0, \mathbf{h}_0]$ through an augmented Lagrangian penalty function $\phi(\mathbf{q}) = \mathbf{v}^T\mathbf{q} + \|\mathbf{w} \circ \mathbf{q}\|_2^2$. Here, $\mathbf{v}$ and $\mathbf{w}$ are the penalty parameters of the augmented Lagrangian functions, and $\mathbf{x}_0 \geq \mathbf{0}$ are slack variables that allow $\mathbf{g}_0$ to be treated as equality constraints. A penalty parameter update algorithm is required to set the penalty parameters such that the relaxation error becomes zero.

After relaxation of the linking constraints $\mathbf{q}$, the remaining local constraints $\mathbf{g}_j$ and $\mathbf{h}_j$ are separable with respect to the disciplinary variables $\mathbf{y}_j$ and $\mathbf{x}_j$. The objective is not separable due to the coupling objective f_0 and the augmented Lagrangian penalty ϕ.

Now that the constraints are separable, ALC defines M disciplinary subproblems P_j, $j = 1, \ldots, M$. The disciplinary optimization subproblem P_j is defined as solving the relaxed problem for one subset of variables $[\mathbf{x}_j, \mathbf{y}_j]$. Subproblem P_j only includes those functions that depend on $[\mathbf{x}_j, \mathbf{y}_j]$, and is therefore given by:

$$\begin{array}{ll} \min\limits_{\mathbf{x}_j, \mathbf{y}_j (, \mathbf{x}_0)} & f_j(\mathbf{y}_j, \mathbf{x}_j) + f_0(\mathbf{x}_1, \mathbf{y}_1, \ldots, \mathbf{x}_M, \mathbf{y}_M) + \phi(\mathbf{q}(\mathbf{x}_1, \mathbf{y}_1, \ldots, \mathbf{x}_M, \mathbf{y}_M), \mathbf{x}_0) \\ \text{subject to} & \mathbf{g}_j(\mathbf{y}_j, \mathbf{x}_j) \leq \mathbf{0} \\ & \mathbf{h}_j(\mathbf{y}_j, \mathbf{x}_j) = \mathbf{0} \\ & \mathbf{x}_0 \geq \mathbf{0} \end{array} \tag{2.3}$$

Where the slack variables $\mathbf{x}_0$ are included in the optimization variables of the first subproblem. Note that the solution to subproblem j depends on the solution of the other subproblems $i \neq j$, since these appear in the coupling objective f_0 and the penalty function ϕ.

2 Coordination algorithm

The coordination algorithm for ALC has two tasks:

1. To select appropriate penalty parameters $\mathbf{v}$ and $\mathbf{w}$
2. To account for the coupling of subproblem objectives

The ALC coordination algorithm performs these tasks in a nested strategy that consists of inner and outer loops [14]. The method of multipliers [3] is used in the outer loop to set the penalty parameters, and an alternating minimization approach [4] that sequentially solves the subproblems accounts for the subproblem coupling in an inner loop. The coordination algorithm is illustrated in Figure 2.1.

Outer loop: method of multipliers

In the outer loops, the method of multipliers sets the penalty parameters $\mathbf{v}^{k+1}$ for outer iteration $k+1$ using the following update formula [2, 3]:

$$\mathbf{v}^{k+1} = \mathbf{v}^k + 2\mathbf{w}^k \circ \mathbf{w}^k \circ \mathbf{q}^k \tag{2.4}$$

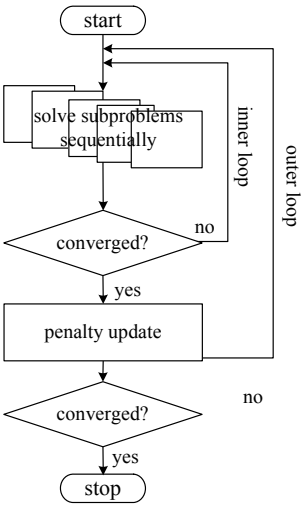

Figure 2.1: Illustration of the coordination algorithm for ALC

where $\mathbf{q}^k$ are the values of the linking constraints $\mathbf{q}$ at termination of the the k-th inner loop. Since large penalty weights slow down the coordination algorithms and introduce ill-conditioning of the subproblems, the penalty weights $\mathbf{w}$ are increased a factor β only when the reduction in the linking constraint value is smaller than some fraction γ. If the reduction is larger, the penalty weights are not updated. As a result, the penalty weights are only increased when the contribution of the Lagrange multiplier update (2.4) did not lead to a large enough reduction in the violation of the linking constraints. More formally, the penalty weight w_i for the ith linking constraint q_i is updated as [3]

$$w_i^{k+1} = \begin{cases} w_i^k & \text{if} \quad |q_i^k| \leq \gamma |q_i^{k-1}| \\ \beta w_i^k & \text{if} \quad |q_i^k| > \gamma |q_i^{k-1}| \end{cases} \tag{2.5}$$

where $\beta > 1$ and $0 < \gamma < 1$, and we observe that $\beta = 2.2$ and $\gamma = 0.4$ perform well in general.

The outer loop, and thus the solution procedure, is terminated when two conditions are satisfied. First, the change in the maximal linking constraint value for two consecutive outer loop iterations must be smaller than some user-defined termination tolerance $\varepsilon > 0$

$$\|\mathbf{q}^k - \mathbf{q}^{k-1}\|_\infty < \varepsilon \tag{2.6}$$

Second, the maximal linking constraint violation must also be smaller than tolerance $\varepsilon > 0$

$$\|\mathbf{q}^k\|_\infty < \varepsilon \tag{2.7}$$

Inner loop: alternating optimization

In the inner loop, subproblems are solved sequentially for fixed penalty parameters $\mathbf{v}$ and $\mathbf{w}$ using an alternating optimization approach [7, 4]. This procedure is terminated when the relative change in the objective function value F of the relaxed system design problem given by

$$\begin{aligned} F(\mathbf{x}_0, \mathbf{x}_1, \mathbf{y}_1, \ldots, \mathbf{x}_M, \mathbf{y}_M) &= \sum_{j=1}^{M} f_j(\mathbf{x}_j, \mathbf{y}_j) \\ &+ f_0(\mathbf{x}_1, \mathbf{y}_1, \ldots, \mathbf{x}_M, \mathbf{y}_M) + \phi(\mathbf{q}(\mathbf{x}_1, \mathbf{y}_1, \ldots, \mathbf{x}_M, \mathbf{y}_M), \mathbf{x}_0) \end{aligned}$$

for two consecutive inner loop iterations is smaller than some user-defined termination tolerance $\varepsilon_{\text{inner}} > 0$:

$$\frac{|F^\xi - F^{\xi-1}|}{1 + |F^\xi|} < \varepsilon_{\text{inner}} \tag{2.8}$$

where ξ denotes the inner loop iteration number. The division by $1 + |F^{\xi}|$ is used for proper scaling of the criterion for very large as well as very small objectives [5]. The termination tolerance $\varepsilon_{\text{inner}}$ should be smaller than the outer loop termination tolerance ε to assure sufficient accuracy of the inner loop solution. We use $\varepsilon_{\text{inner}} = \varepsilon/100$.

An alternative inner loop termination strategy is to cut off the inner loop before actual convergence during the first few iterations by using looser tolerances. More formally, such an inexact approach uses a different tolerance $\varepsilon^k_{\text{inner}}$ for each outer loop iteration. The main idea behind such a strategy is that costly inner loop iterations are avoided when the penalty parameters are still far from their optimal values. Convergence for the outer loop updates in case of an inexact inner loop can still be guaranteed, as long as the sequence of termination tolerances $\{\varepsilon^k_{\text{inner}}\}$ is non-increasing, and $\varepsilon^k_{\text{inner}} \to 0$ [3]. An extreme case of the above is the so-called alternating direction method of multipliers that performs only a single iteration for each inner loop. More moderate values for β (smaller) and γ (larger) are advised for these inexact approaches.

Initial weight selection

Although the above algorithms converge for any positive initial weight, the performance of the outer loop method of multipliers depends on the choice of the initial weight $\mathbf{w}$. To select the initial weights, the ALC toolbox includes an approach that chooses the weights such that the sum of the penalty terms is a fraction α of the objective function value: $\phi \approx \alpha|f|$.

This approach initially sets the Lagrange multipliers to $\mathbf{v} = \mathbf{0}$, and takes all weights equal $\mathbf{w} = w$, such that $\phi = w^2 \mathbf{q}^T \mathbf{q}$. The initial weights are then selected as

$$w = \sqrt{\frac{\alpha|\hat{f}|}{\hat{\mathbf{q}}^T \hat{\mathbf{q}}}} \tag{2.9}$$

where $\hat{f}$ and $\hat{\mathbf{q}}$ are estimates of typical objective function and the linking constraint values. For many engineering problems, a reasonable (order of magnitude) estimate of the objective function minimum in the optimum can often be given. ALC assumes that the estimate of the objective is non-zero, which is often the case in engineering design. However, if $\hat{f}$ happens to be zero, a non-zero, conservative "typical value" should be taken for this estimate (since we require the weights to be larger than zero).

The estimates for the linking constraints $\hat{\mathbf{q}}$ are obtained by solving the decomposed problem for a small weights $\mathbf{w} = w^0$, and zero Lagrange multipliers $\mathbf{v} = 0$. For these weights, the penalty term will be small when compared to the objective function value. As a consequence, the allowed linking constraint violations will be large, and the solution of the relaxed problem will produce an estimate $\hat{\mathbf{q}}$ for the size of the linking constraint values.

Chapter 3
Matlab toolbox for ALC

The ALC toolbox contains 13 files that together perform the coordination process for ALC. Table 3.1 provides a brief description of these files, and Figure 3.1 depicts the relations between the main toolbox files.

The general command to solve a partitioned problem using the ALC toolbox is

```
>> [Z, FVAL, EXITFLAG, OUTPUT] = alcsolve('name', Z0, ZLB, ZUB, ALG, par)
```

The required inputs are the name `'name'` of the Matlab input file (which can be generated from a Ψ specification), the initial guess `Z0` for the design variables, and the lower and upper bounds on the design variables, `ZLB` and `ZUB` respectively, where `Z0`, `ZLB`, and `ZUB` are all column vectors of appropriate size. The user input file should be of the same format as described in Section 1. The `ALG` and `par` arguments are optional, and can be used to set non-default algorithmic settings and to supply problem-specific fixed parameters, respectively. A description of the fields of `ALG` and their default values are given in Table 3.2.

The output `Z` contains the obtained solution, `FVAL` is the associated objective function value, `EXITFLAG` is a convergence message that is 1 if the process terminated correctly, 0 if the maximum number of iterations was reached, and -1 if the algorithms terminated at a design that violates one (or more) of the linking constraints. `OUTPUT` is a structure that includes detailed information on the final solution and the convergence history. For a more detailed description of `OUTPUT`, type `help alcsolve` at the Matlab command prompt.

Table 3.1: Description of the 13 files in the ALC toolbox.

Filename	Description
`alcsolve`	Main routine
`ALGdef`	Inserts default algorithmic settings
`setup_problem`	Initializes the problem and solution
`convert`	Converts Ψ generated input files into ALC input files
`update_penalty`	Updates penalty parameters $\mathbf{v}$ and $\mathbf{w}$
`solve_innerloop`	Inner loop routine
`solve_subproblem`	Solves individual subproblems
`sub_obj`	Determines the value of the objective of a subproblem
`sub_con`	Determines the values of the constraints of a subproblem
`get_z`	Computes intermediate solutions in the inner loop
`obj_relaxed`	Computes the objective of the relaxed problem required for determining termination of the inner loop
`check_cvrg_outer`	Outer loop convergence check
`none`	Placeholder used when an objective or constraint function is absent in an input files

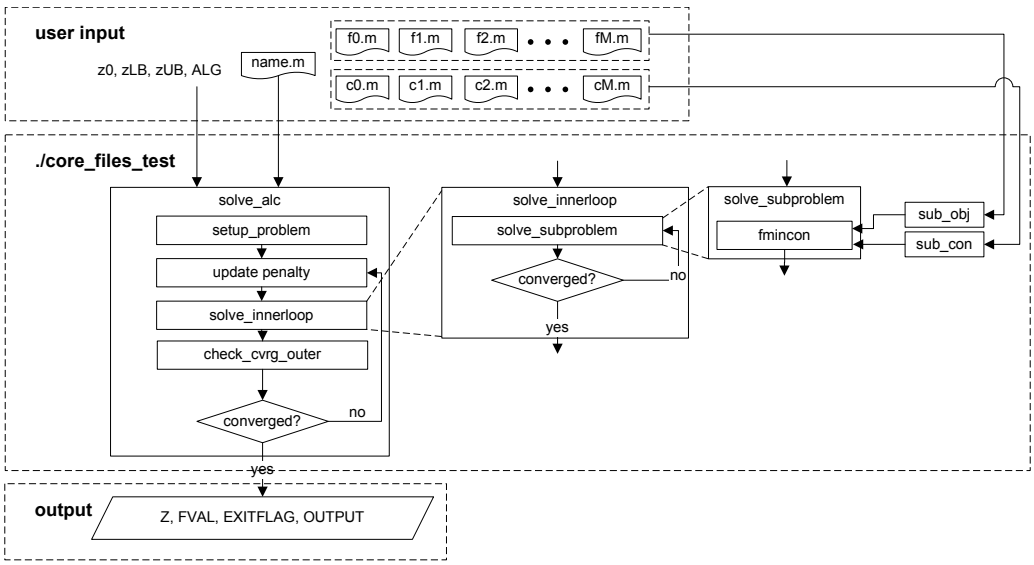

Figure 3.1: File organization of the ALC toolbox

Table 3.2: Description of the fields of ALG.

Field	Description	Default value
.maxiters	Maximum number of cumulative inner loop iterations	1000
.outer.tol	Outer loop termination tolerance ε	0.001
.outer.pen	Initial penalty parameters $\mathbf{v}^0$ and $\mathbf{w}^0$	[0, 0.001]
.outer.preset	Use initial weight selection strategy 'yes' or 'no'	'yes'
.outer.beta	Weight update parameter β	2.2 if ALG.inner.tol = 'fix' 2.2 if ALG.inner.tol = 'inexact' 1.2 if ALG.inner.tol = 'none'
.outer.gamma	Weight update parameter γ	0.4 if ALG.inner.tol = 'fix' 0.5 if ALG.inner.tol = 'inexact' 0.75 if ALG.inner.tol = 'none'
.inner.tol	Inner loop termination type. 'fix' for $\varepsilon_{\text{inner}} = \varepsilon/100$, 'inexact' for decreasing $\{\varepsilon^k_{\text{inner}}\}$, and 'none' for alternating direction method of multipliers with a single iteration inner loop.	'none'
.inner.iters	(if ALG.inner.tol = 'inexact'). Number of iterations in which the $\{\varepsilon^k_{\text{inner}}\}$ is reduced from 1 to $\varepsilon/100$	10
.inner.sequence	Sequence in which subproblems are solved in the inner loop 'ascend', 'descend', 'random'	'ascend'
.preset.alpha	Parameter α of the initial weight setting strategy	0.1
.preset.typical_f	Parameter $\hat{f}$ of the initial weight setting strategy	1

Chapter 4

Generating input files from Ψ specifications

Generating input files for the ALC toolbox from Ψ specification is very straightforward. The first step is to generate a normalized partition from a Ψ specification. In the second step, a matlab file is generated from this normalized partition.

To illustrate the generation of input files, consider the example optimization problem taken from [12]

$$\begin{array}{rl} \min\limits_{z_1,\ldots,z_7} & f(z_1)+f(z_2)=z_1^2+z_2^2 \\ \text{subject to} & g_1(z_3,z_4,z_5)=(z_3^{-2}+z_4^2)z_5^{-2}-1\leq 0 \\ & g_2(z_5,z_6,z_7)=(z_5^2+z_6^{-2})z_7^{-2}-1\leq 0 \\ & h_1(z_1,z_3,z_4,z_5)=(z_3^2+z_4^{-2}+z_5^2)z_1^{-2}-1=0 \\ & h_2(z_2,z_5,z_6,z_7)=(z_5^2+z_6^2+z_7^2)z_2^{-2}-1=0 \\ & 0.1\leq z_i\leq 10 \qquad\qquad i=1,\ldots,7 \end{array} \tag{4.1}$$

The problem is partitioned into two subproblems. The first subproblem has local variables $\mathbf{x}_1 = [z_1, z_3, z_4]$, local objective $f(z_1)$, and local constraints g_1, h_1. The second subproblem has local variables $\mathbf{x}_2 = [z_2, z_6, z_7]$, local objective $f(z_2)$, and local constraints g_2, h_2. The subproblems are coupled through the linking variable $\mathbf{y} = [z_5]$. The Ψ specification for this partition is given below:

comp *First* =
|[extvar z_5
intvar z_1, z_3, z_4
objfunc $f(z_1)$
confunc $g_1(z_3, z_4, z_5)$, $h_1(z_1, z_3, z_4, z_5)$
]|

comp *Second* =
|[extvar z_5
intvar z_2, z_6, z_7
objfunc $f(z_2)$
confunc $g_2(z_5, z_6, z_7)$, $h_2(z_2, z_5, z_6, z_7)$
]|

```
syst Geo1 =
|[  sub A: First, B: Second
    link A.z5 -- B.z5
]|

topsyst Geo1
```

The first step is to generate the normalized partition for this specification by running the command

```
$check-psi geo1.psi geo1.np
```

where `geo1.psi` contains the Ψ specification, and `geo1.np` is the file to which the normalized partition is written. The second step is to generate a matlab input file from this normalized partition with the command:

```
$np2ml geo1.np geo1.m
```

where `geo1.m` is the name of the generated matlab input file.

In its current implementation, the generator only allows only partitioned problems that do not have response functions, and that contain a single system. Since ALC allows only scalar objective functions, the total objective function is taken as the sum of the specified objective functions.

1 Input files

The generated Matlab input file is of a simple structure and appends information of the partitioned problem to a structure `PR` initialized by the ALC toolbox. Since the input file `name` is called as a function within the ALC toolbox, its first line is

```
PR = geo1(PR)
```

where `geo1` refers to the name of the Matlab input file itself (`geo1.m` for this example).

An input file has to add two fields to the `PR` structure generated by the ALC toolbox: the field `PR.main` for system-level properties, and the vector field `PR.sub`, whose j-th element `PR.sub(`j`)` includes the properties for component `comp_`j of the normalized partition.

Both the system-level and subproblem fields are based on the formulation of the decomposed problem with subproblems (2.3). The decomposed formulation includes copies of the shared variables, and has separable local objectives and constraints that depend only on the variables of one of the subproblems. The system-wide functions may however depend on the variables of two or more subproblems. Note that the decomposed formulation is different from the system optimization problem (2.1) without shared variable copies.

Field `PR.main` has a number of sub-fields associated with the description of the problem, its components, and their variables, together with sub-fields that define the coupling objectives $\mathbf{f}_0$ and coupling constraints $\mathbf{g}_0$ and $\mathbf{h}_0$.

The field PR.main.name simply contains the name of the top-level system (i.e. the value associated with the name key of section syst_1). The sub-field PR.main.comp_names contains an array of names of the instantiated components, such that the operation PR.main.comp_names(j) yields the value of the name key of component comp_j of the normalized partition. Similarly, PR.main.z_names is an array that contains the names of the variables such that PR.main.z_names(i) yields the name of variable var_i. To distinguish between variables that have the same name but belong to different components (which can occur for shared variable copies), the field PR.main.z_comps is introduced and contains the names of the instantiated components to which the variables belong (i.e. the value of the name field of the component section that includes var_i as one of the values of its variable keys). Hence, the operation PR.main.z_comps(i) yields the name of the component to which variable PR.main.z_names(i) belongs. For the geometric programming problem and its given partition, these main fields become[1]

```
PR.main.name = 'Geo1'
PR.main.comp_names = {'A','B'};
PR.main.z_names = {'z5','z1','z3','z4','z5','z2','z6','z7'};
PR.main.z_comps = {'A','A','A','A','B','B','B','B'};
```

The properties of the coupling objectives and constraints are listed in the fields PR.main.obj and PR.main.con. Each of these fields is a vector whose length equals the number of coupling objective and coupling constraint functions, respectively. Each element of this vector is a structure with two sub-fields. For example, the sub-fields for the first coupling objective are PR.main.obj(1).name that defines its name, and PR.main.obj(1).args that includes the indices of its arguments. The argument indices operate on the list of variables PR.main.z_names such that the operation PR.main.z_names(PR.main.obj(1).args) yields the function's arguments in the correct order. The name of a function refers to the name of a Matlab function that takes the vector of arguments as inputs, and returns a scalar value of the objective function as an output. Following Matlab's fmincon preferences, constraint functions yield two outputs upon evaluation: the first is a vector that contains the values of the inequality constraints, and the second is a vector that contains the values of the equality constraints. If a system does not have a coupling objective or a coupling constraint, then the none function (part of the toolbox) is used with an empty list of arguments. Example (4.1) does not have coupling functions, such that

```
PR.main.obj(1).name = 'none';
PR.main.obj(1).args = [];
PR.main.con(1).name = 'none';
PR.main.con(1).args = [];
```

which completes the PR.main field for this example.

The component fields PR.sub(j), $j = 1, \ldots, M$, contain definitions of a component's variables, its functions, and its variable couplings. The fields PR.sub(j).y_id and PR.sub(j).x_id contain the vectors of indices associated with the coupling variables $\mathbf{y}_j$ and local variables $\mathbf{x}_j$, respectively, of component comp_j. The coupling variables are given by the values of the coupling_var key, and the local variables are the values of the local_var key of section comp_j. Note that each variable can only be assigned to only one subproblem. The fields PR.sub(j).obj and PR.sub(j).con for the local objective and constraint functions are defined similar to the coupling functions in PR.main.obj and PR.main.con.

Couplings between the coupling variables of component comp_j and its neighbors comp_n, $n \in \mathcal{N}_j$ are specified by selection matrices S_{jn} that appear in the definition of the consistency

[1] All names are placed between quotes to comply with Matlab's notational conventions for strings.

constraints $\mathbf{c}_{jn}$ of (2.2). These matrices are generated from the link sections of the normalized partition. For component comp_j the matrix S_{jn} is included in field `PR.sub(`j`).to(`n`).Sij`. The selection matrices for components that are not neighbors of j are defined as empty zero matrices of appropriate size.

The generated Matlab definitions of the two components of Example (4.1) are given by

```
PR.sub(1).y_id = [1];
PR.sub(1).x_id = [2, 3, 4];
PR.sub(1).obj(1).name = 'f';
PR.sub(1).obj(1).args = [2];
PR.sub(1).con(1).name = 'g1';
PR.sub(1).con(1).args = [3, 4, 1];
PR.sub(1).con(2).name = 'h1';
PR.sub(1).con(2).args = [2, 3, 4, 1];
PR.sub(1).to(2).Sij = [1];

PR.sub(2).y_id = [5];
PR.sub(2).x_id = [6, 7, 8];
PR.sub(2).obj(1).name = 'f';
PR.sub(2).obj(1).args = [6];
PR.sub(2).con(1).name = 'g2';
PR.sub(2).con(1).args = [5, 7, 8];
PR.sub(2).con(2).name = 'h2';
PR.sub(2).con(2).args = [6, 5, 7, 8];
PR.sub(2).to(1).Sij = [1];
```

Assuming that the objective and constraint functions and the toolbox files are in the current directory or in Matlab's path, the solution of Example (4.1) with ALC under default settings is obtained by typing

```
>> Z0 = ones(8,1); ZLB = .1*Z0, ZUB = 10*Z0;
>> [Z, FVAL] = alcsolve('geo1', Z0, ZLB, ZUB)
```

in the Matlab command prompt. Note that the length of the initial guess and variable bounds is larger than the number of original variables of Problem (4.1) since the initial guess and bounds also include entries for each copy of the linking variables $\mathbf{y}$.

Chapter 5
Examples

In this section, we demonstrate the use of the input file generator on a number of example problems and partitions taken from the Ψ user manual [15]. Since the input file generator does not allow response functions or multiple systems, the examples that do not satisfy these requirements are reformulated appropriately. The first example is a geometric programming problems, the second example is Golinski's speed reducer design problem [6], the third example is the portal frame design problem introduced by [11], and the fourth is the supersonic business jet example introduced by [1]. The vehicle chassis design problem of [9] is not included here since its analysis models are not available at this point. The partition specification files and generated ALC input files are included in Appendix A.

1 Geometric programming

The first example is the geometric programming problem taken from [8, 12, 13]. The problem is given by:

$$
\begin{aligned}
\min_{z_1,\ldots,z_{14}} \quad & f(z_1) + f(z_2) = z_1^2 + z_2^2 \\
\text{subject to} \quad & g_1(z_3, z_4, z_5) = (z_3^{-2} + z_4^2) z_5^{-2} - 1 \leq 0 \\
& g_2(z_5, z_6, z_7) = (z_5^2 + z_6^{-2}) z_7^{-2} - 1 \leq 0 \\
& g_3(z_8, z_9, z_{11}) = (z_8^2 + z_9^2) z_{11}^{-2} - 1 \leq 0 \\
& g_4(z_8, z_{10}, z_{11}) = (z_8^{-2} + z_{10}^2) z_{11}^{-2} - 1 \leq 0 \\
& g_5(z_{11}, z_{12}, z_{13}) = (z_{11}^2 + z_{12}^{-2}) z_{13}^{-2} - 1 \leq 0 \\
& g_6(z_{11}, z_{12}, z_{14}) = (z_{11}^2 + z_{12}^2) z_{14}^{-2} - 1 \leq 0 \\
& h_1(z_1, z_3, z_4, z_5) = (z_3^2 + z_4^{-2} + z_5^2) z_1^{-2} - 1 = 0 \\
& h_2(z_2, z_5, z_6, z_7) = (z_5^2 + z_6^2 + z_7^2) z_2^{-2} - 1 = 0 \\
& h_3(z_3, z_8, z_9, z_{10}, z_{11}) = (z_8^2 + z_9^{-2} + z_{10}^{-2} + z_{11}^2) z_3^{-2} - 1 = 0 \\
& h_4(z_6, z_{11}, z_{12}, z_{13}, z_{14}) = (z_{11}^2 + z_{12}^2 + z_{13}^2 + z_{14}^2) z_6^{-2} - 1 = 0 \\
& 0.1 \leq z_i \leq 10 \qquad i = 1, \ldots, 14
\end{aligned}
\tag{5.1}
$$

For this problem, we take the third partition of the Ψ reference manual, illustrated in Figure 5.1. The partition has four components that are coupled through the linking variables z_3, z_5, z_6, z_{11}.

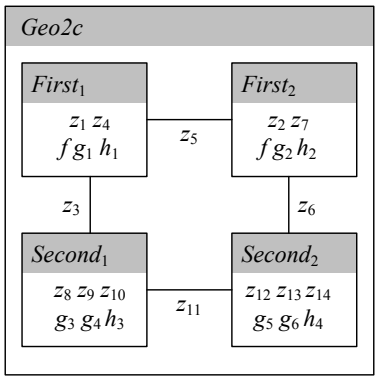

Figure 5.1: Partition of geometric programming problem

The Ψ specification and the generated input file for the ALC toolbox are given in Appendix 2.

The problem's solution can be invoked by entering

```
>> Z0 = ones(18,1); ZLB = .1*Z0, ZUB = 10*Z0;
>> [Z, FVAL] = alcsolve('geo2c', Z0, ZLB, ZUB)
```

in the matlab command prompt. Again, the initial guesses and bound values are also included for the four variable copies introduced for the linking variables.

2 Speed reducer

The second example is the speed reducer design problem taken from [6, 10, 13]. The objective of this problem is to minimize the volume of a speed reducer, subjected to stress, deflection, and geometric constraints. The design variables are the dimensions of the gear itself (x_1, x_2, x_3), and

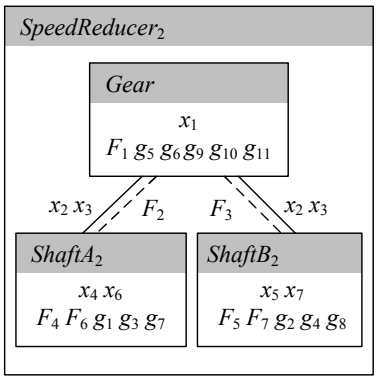

Figure 5.2: Partition for the speed reducer problem.

both the shafts (x_4, x_6 and x_5, x_7). The design problem for the speed reducer is defined by:

$$
\begin{aligned}
\min_{x_1,\ldots,x_7} \quad & [F_1(x_1,x_2,x_3), F_2(x_1,x_6), F_3(x_1,x_7), F_4(x_6), F_5(x_7), F_6(x_4,x_6), F_7(x_5,x_7)] \\
\text{subject to} \quad & g_1(x_2,x_3,x_4,x_6) = \frac{1}{110x_6^3}\sqrt{\left(\frac{745x_4}{x_2x_3}\right)^2 + 1.69\cdot 10^7} - 1 \le 0 \\
& g_2(x_2,x_3,x_5,x_7) = \frac{1}{85x_7^3}\sqrt{\left(\frac{745x_5}{x_2x_3}\right)^2 + 1.575\cdot 10^8} - 1 \le 0 \\
& g_3(x_4,x_6) = \frac{1.5x_6+1.9}{x_4} - 1 \le 0 \\
& g_4(x_5,x_7) = \frac{1.1x_7+1.9}{x_5} - 1 \le 0 \\
& g_5(x_1,x_2,x_3) = \frac{27}{x_1x_2^2x_3} - 1 \le 0 \\
& g_6(x_1,x_2,x_3) = \frac{397.5}{x_1x_2^2x_3^2} - 1 \le 0 \\
& g_7(x_2,x_3,x_4,x_6) = \frac{1.93x_4^3}{x_2x_3x_6^4} - 1 \le 0 \\
& g_8(x_2,x_3,x_5,x_7) = \frac{1.93x_5^3}{x_2x_3x_7^4} - 1 \le 0 \\
& g_9(x_2,x_3) = \frac{x_2x_3}{40} - 1 \le 0 \\
& g_{10}(x_1,x_2) = \frac{5x_2}{x_1} - 1 \le 0 \\
& g_{11}(x_1,x_2) = \frac{x_1}{12x_2} - 1 \le 0 \\
& 2.6 \le x_1 \le 3.6 \;\; 7.3 \le x_5 \le 8.3 \\
& 0.7 \le x_2 \le 0.8 \;\; 2.9 \le x_6 \le 3.9 \\
& 17 \le x_3 \le 28 \;\; 5.0 \le x_7 \le 5.5 \\
& 7.3 \le x_4 \le 8.3 \\
\text{where} \quad & F_1 = 0.7854x_1x_2^2(3.3333x_3^2 + 14.9335x_3 - 43.0934) \\
& F_2 = -1.5079x_1x_6^2,\; F_3 = -1.5079x_1x_7^2,\; F_4 = 7.477x_6^3 \\
& F_5 = 7.477x_7^3,\; F_6 = 0.7854x_4x_6^2,\; F_7 = 0.7854x_5x_7^2
\end{aligned}
\tag{5.2}
$$

The second partition of the Ψ reference manual is taken for this problem. This partition is illustrated in Figure 5.2 and has three components coupled through the linking variables x_2 and x_3 and the coupling objectives F_2 and F_3. The Ψ specification and matlab input file are given in Appendix 3.

The problem's solution can be invoked by entering

```
>> ZLB = [ 0.7, 17, 2.6,   0.7, 17, 7.3, 2.9,   0.7, 17, 7.3, 5.0]';
>> ZUB = [ 0.8, 28, 3.6,   0.8, 28, 8.3, 3.9,   0.8, 17, 8.3, 5.5]';
>> Z0 = (ZLB + ZUB)./2;
>> [Z, FVAL] = alcsolve('speed2', Z0, ZLB, ZUB)
```

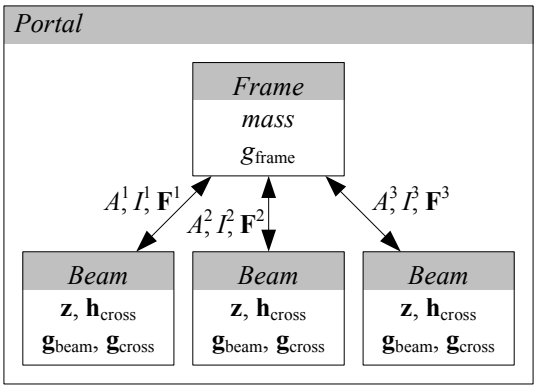

Figure 5.3: Partition for the portal frame example

in the matlab command prompt. Again, the initial guesses and bound values are also included for the four variable copies introduced for the linking variables.

3 Portal frame

The third example is the structural optimization of a portal frame subjected to an external load. The portal frame consists of three I-beams $i = 1, 2, 3$, each with six cross-sectional dimensions $\mathbf{z}^i = [h, w_1, w_2, b, t_1, t_2]^i$ as design variables. As response variables, the are A^i and moment of inertia I^i are introduced for each beam, as well as the reaction forces $\mathbf{F}^i = [X_1, Y_1, M_1, X_2, Y_2, M_2]^i$. The portal frame optimization problem is defined by

$$
\begin{array}{lll}
\text{find} & \mathbf{z}^1, \mathbf{z}^2, \mathbf{z}^3, A^1, A^2, A^3, I^1, I^2, I^3, & \\
& u, \mathbf{F}^1, \mathbf{F}^2, \mathbf{F}^3, \boldsymbol{\sigma}^1, \boldsymbol{\sigma}^2, \boldsymbol{\sigma}^3 & \\
\min & \mathit{mass}(A^1, A^2, A^3) & \\
\text{s.t.} & g_{\text{frame}}(\mathbf{F}^1, \mathbf{F}^2, \mathbf{F}^3, A^1, A^2, A^3, I^1, I^2, I^3) \leq \mathbf{0} & \\
& \mathbf{g}^i_{\text{beam}}(\mathbf{F}^i, \mathbf{z}^i) \leq \mathbf{0} & i = 1, 2, 3 \\
& \mathbf{g}^i_{\text{cross}}(\mathbf{z}^i) \leq \mathbf{0} & i = 1, 2, 3 \\
& \mathbf{h}_{\text{cross}}(A^i, I^i, \mathbf{z}^i) = (A^i, I^i) - \mathbf{a}_{\text{cross}}(\mathbf{z}^i) = \mathbf{0} & i = 1, 2, 3
\end{array}
\tag{5.3}
$$

where both design and response variables are included as optimization variables. Here, the constraints $\mathbf{g}_{\text{beam}}$ are stress constraints, $\mathbf{g}_{\text{cross}}$ are cross-sectional constraints, and the constraints $\mathbf{h}_{\text{cross}}$ are introduced to reformulate the response functions for A and I as equality constraints. Note that the intermediate stress and frame analysis functions $\mathbf{a}_{\text{beam}}$ and $\mathbf{a}_{\text{frame}}$ are integrated in the constraints $\mathbf{g}_{\text{beam}}$ and $\mathbf{g}_{\text{frame}}$, respectively.

For the portal frame example, we take a partition similar to the one described in the Ψ user manual. The partition has one system-level component and one component for each beam (Figure 5.3). The specification in Ψ and the generated input file are given in Appendix 4.

The problem's solution can be invoked by running the script `examples/portal/run_portalalc.m` from the Matlab command line. The script includes the definition of problem-specific parameters, an appropriate initial guess, and lower and upper bounds on the optimization variables. To prevent ill-conditioning of the problem, the variables are scaled such that they have the same order of magnitude.

Bibliography

[1] J. S. Agte, J. Sobieszczanski-Sobieski, and R. R. Jr. Sandusky. Supersonic business jet design through bi-level integrated system synthesis. In *Proceedings of the World Aviation Conference*, San Fransisco, CA, 1999. MCB Press, SAE paper 1999-01-5622.

[2] D. P. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods*. Academic Press, New York, NY, 1982.

[3] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, MA, 2nd edition, 2003. 2nd printing.

[4] J. C. Bezdek and R.J. Hathaway. Some notes on alternating optimization. *Lecture Notes in Computer Science*, 2275:288–300, January 2002.

[5] P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization*. Academic Press, London, UK, 1981.

[6] J. Golinski. Optimal synthesis problems solved by means of nonlinear programming and random methods. *Journal of Mechanisms*, 5:287–309, 1970.

[7] L. Grippo and M. Sciandrone. On the convergence of the block nonlinear Gauss-Seidel method under convex constraints. *Operations Research Letters*, 26(3):127–136, 2000.

[8] H. M. Kim. *Target Cascading in Optimal System Design*. PhD thesis, University of Michigan, 2001.

[9] H. M. Kim, N. F. Michelena, P. Y. Papalambros, and T. Jiang. Target cascading in optimal system design. *ASME Journal of Mechanical Design*, 125(3):474–480, 2003.

[10] S. L. Padula, N. M. Alexandrov, and L. L. Green. MDO test suite at nasa langley research center. In *Proceedings of the 6th AIAA/USAF/NASA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Bellevue, WA, 4-6 September 1996. AIAA paper 1996-4028. Website http://www.eng.buffalo.edu/Research/MODEL/mdotestsuite.html.

[11] J. Sobieszczanski-Sobieski, B. B. James, and A. R. Dovi. Structural optimization by multilevel decomposition. AIAA paper 1983-0832, May 1983.

[12] S. Tosserams, L. F. P. Etman, P. Y. Papalambros, and J. E. Rooda. An augmented Lagrangian relaxation for analytical target cascading using the alternating direction method of multipliers. *Structural and Multidisciplinary Optimization*, 31(3):176–189, 2006.

[13] S. Tosserams, L. F. P. Etman, and J. E. Rooda. An augmented Lagrangian decomposition method for quasi-separable problems in MDO. *Structural and Multidisciplinary Optimization*, 34(3):211–227, 2007.

[14] S. Tosserams, L. F. P. Etman, and J. E. Rooda. Augmented Lagrangian coordination for distributed optimal design in MDO. *International Journal for Numerical Methods in Engineering*, 73(13):1885–1910, 2008.

[15] S. Tosserams, A. T. Hofkamp, and L. F. P. Etman. Ψ reference manual. SE-report, Eindhoven University of Technology, 2009.

Appendix A
Specification and generated outputs for examples

This appendix includes the actual input files and generated outputs for the examples presented in the user manual.

1 Example (4.1)

Partition specification in Ψ:

```
# Geometric programming problem 1
#
# vars: z1 z2 z3 z4 z5 z6 z7
#
# objs: f(z1), f(z2)
# cons: g1(z3,z4,z5), g2(z5,z6,z7), h1(z1,z3,z4,z5), h2(z2,z5,z6,z7)


comp First =
|[  extvar z5
    intvar z1, z3, z4
    objfunc f(z1)
    confunc g1(z3,z4,z5)
          , h1(z1,z3,z4,z5)
]|

comp Second =
|[  extvar z5
    intvar z2, z6, z7
    objfunc f(z2)
    confunc g2(z5,z6,z7)
```

```
          , h2(z2,z5,z6,z7)
]|

syst Geo1 =
|[  sub A: First, B: Second
    link A.z5 -- B.z5
]|

topsyst Geo1
```

Generated matlab file:

```
%
% Generated by np2alc
%
function PR = geo1(PR)

PR.main.z_id = [1, 2, 3, 4, 5, 6, 7, 8];
PR.main.z_names = {'z1', 'z3', 'z4', 'z5', 'z2', 'z6', 'z7', 'z5'};
PR.main.z_comps = {'First', 'First', 'First', 'First', 'Second', 'Second', 'Second', 'Second'};

PR.main.m = 2;
PR.main.comp_id = [1, 2];
PR.main.comp_names = {'A', 'B'};

PR.main.obj(1).name = 'none';
PR.main.obj(1).args = [];

PR.main.con(1).name = 'none';
PR.main.con(1).args = [];

PR.sub(1).x_id = [1, 2, 3];
PR.sub(1).y_id = [4];

PR.sub(1).obj(1).name = 'f';
PR.sub(1).obj(1).args = [1];

PR.sub(1).con(1).name = 'g1';
PR.sub(1).con(1).args = [2, 3, 4];
PR.sub(1).con(2).name = 'h1';
PR.sub(1).con(2).args = [1, 2, 3, 4];

PR.sub(1).to(2).Sij = [1];

PR.sub(2).x_id = [5, 6, 7];
PR.sub(2).y_id = [8];

PR.sub(2).obj(1).name = 'f';
PR.sub(2).obj(1).args = [5];

PR.sub(2).con(1).name = 'g2';
PR.sub(2).con(1).args = [8, 6, 7];
PR.sub(2).con(2).name = 'h2';
PR.sub(2).con(2).args = [5, 8, 6, 7];

PR.sub(2).to(1).Sij = [1];
```

---

2 Geometric programming problem

Partition specification in Ψ:

```
comp First1 =
|[  extvar z3, z5
    intvar z1, z4
```

```
    objfunc f(z1)
    confunc g1(z3,z4,z5)
          , h1(z1,z3,z4,z5)
]|

comp First2 =
|[  extvar z5, z6
    intvar z2, z7
    objfunc f(z2)
    confunc g2(z5,z6,z7)
          , h2(z2,z5,z6,z7)
]|

comp Second1 =
|[  extvar z3, z11
    intvar z8, z9, z10
    confunc g3(z8,z9,z11)
          , g4(z8,z10,z11)
          , h3(z3,z8,z9,z10,z11)
]|

comp Second2 =
|[  extvar z6, z11
    intvar z12, z13, z14
    confunc g5(z11,z12,z13)
          , g6(z11,z12,z14)
          , h4(z6,z11,z12,z13,z14)
]|

syst Geo2c =
|[  sub A1: First1, A2: First2, B1: Second1, B2: Second2
    link A1.z3 -- B1.z3, A2.z6 -- B2.z6, B1.z11 -- B2.z11, A1.z5 -- A2.z5
]|

topsyst Geo2c
```

Generated matlab file:

```
%
% Generated by np2alc
%
function PR = geo2c(PR)

PR.main.z_id = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18];
PR.main.z_names = {'z1', 'z4', 'z3', 'z5', 'z2', 'z7', 'z5', 'z6', 'z8', 'z9', 'z10', 'z3', 'z11', 'z12', 'z13', 'z14', 'z6', 'z11'};
PR.main.z_comps = {'First1', 'First1', 'First1', 'First1', 'First2', 'First2', 'First2', 'First2', 'Second1', 'Second1', 'Second1', 'Second1', 'Second1', 'Secon

PR.main.m = 4;
PR.main.comp_id = [1, 2, 3, 4];
PR.main.comp_names = {'A1', 'A2', 'B1', 'B2'};

PR.main.obj(1).name = 'none';
PR.main.obj(1).args = [];

PR.main.con(1).name = 'none';
PR.main.con(1).args = [];

PR.sub(1).x_id = [1, 2];
PR.sub(1).y_id = [3, 4];

PR.sub(1).obj(1).name = 'f';
PR.sub(1).obj(1).args = [1];

PR.sub(1).con(1).name = 'g1';
PR.sub(1).con(1).args = [3, 2, 4];
PR.sub(1).con(2).name = 'h1';
```

```
PR.sub(1).con(2).args = [1, 3, 2, 4];

PR.sub(1).to(2).Sij = [0,1];
PR.sub(1).to(3).Sij = [1,0];
PR.sub(1).to(4).Sij = zeros(0, 2);

PR.sub(2).x_id = [5, 6];
PR.sub(2).y_id = [7, 8];

PR.sub(2).obj(1).name = 'f';
PR.sub(2).obj(1).args = [5];

PR.sub(2).con(1).name = 'g2';
PR.sub(2).con(1).args = [7, 8, 6];
PR.sub(2).con(2).name = 'h2';
PR.sub(2).con(2).args = [5, 7, 8, 6];

PR.sub(2).to(1).Sij = [1,0];
PR.sub(2).to(3).Sij = zeros(0, 2);
PR.sub(2).to(4).Sij = [0,1];

PR.sub(3).x_id = [9, 10, 11];
PR.sub(3).y_id = [12, 13];

PR.sub(3).obj(1).name = 'none';
PR.sub(3).obj(1).args = [];

PR.sub(3).con(1).name = 'g3';
PR.sub(3).con(1).args = [9, 10, 13];
PR.sub(3).con(2).name = 'g4';
PR.sub(3).con(2).args = [9, 11, 13];
PR.sub(3).con(3).name = 'h3';
PR.sub(3).con(3).args = [12, 9, 10, 11, 13];

PR.sub(3).to(1).Sij = [1,0];
PR.sub(3).to(2).Sij = zeros(0, 2);
PR.sub(3).to(4).Sij = [0,1];

PR.sub(4).x_id = [14, 15, 16];
PR.sub(4).y_id = [17, 18];

PR.sub(4).obj(1).name = 'none';
PR.sub(4).obj(1).args = [];

PR.sub(4).con(1).name = 'g5';
PR.sub(4).con(1).args = [18, 14, 15];
PR.sub(4).con(2).name = 'g6';
PR.sub(4).con(2).args = [18, 14, 16];
PR.sub(4).con(3).name = 'h4';
PR.sub(4).con(3).args = [17, 18, 14, 15, 16];

PR.sub(4).to(1).Sij = zeros(0, 2);
PR.sub(4).to(2).Sij = [1,0];
PR.sub(4).to(3).Sij = [0,1];
```

3 Speed reducer

Partition specification in Ψ:

```
comp Gear =
|[  extvar x1, x2, x3
    objfunc F1(x1,x2,x3)
    confunc g5(x1,x2,x3)
          , g6(x1,x2,x3)
          , g9(x2,x3)
          , g10(x1,x2)
          , g11(x1,x2)
]|

comp ShaftA2 =
|[  extvar x2, x3, x6
    intvar x4
```

```
    objfunc F4(x6)
          , F6(x4,x6)
    confunc g1(x2,x3,x4,x6)
          , g3(x4,x6)
          , g7(x2,x3,x4,x6)
]|

comp ShaftB2 =
|[  extvar x2, x3, x7
    intvar x5
    objfunc F5(x7)
          , F7(x5,x7)
    confunc g2(x2,x3,x5,x7)
          , g4(x5,x7)
          , g8(x2,x3,x5,x7)
]|

syst SpeedReducer2 =
|[  sub G: Gear, S1: ShaftA2, S2: ShaftB2
    objfunc F2(G.x1,S1.x6)
          , F3(G.x1,S2.x7)
    link G.x2 -- {S1.x2, S2.x2}
       , G.x3 -- {S1.x3, S2.x3}
]|

topsyst SpeedReducer2
```

Generated matlab file:

```
%
% Generated by np2alc
%
function PR = speed2(PR)

PR.main.z_id = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11];
PR.main.z_names = {'x1', 'x2', 'x3', 'x4', 'x2', 'x3', 'x6', 'x5', 'x2', 'x3', 'x7'};
PR.main.z_comps = {'Gear', 'Gear', 'Gear', 'ShaftA2', 'ShaftA2', 'ShaftA2', 'ShaftA2', 'ShaftB2', 'ShaftB2', 'ShaftB2', 'ShaftB2'};

PR.main.m = 3;
PR.main.comp_id = [1, 2, 3];
PR.main.comp_names = {'G', 'S1', 'S2'};

PR.main.obj(1).name = 'F2';
PR.main.obj(1).args = [1, 7];
PR.main.obj(2).name = 'F3';
PR.main.obj(2).args = [1, 11];

PR.main.con(1).name = 'none';
PR.main.con(1).args = [];

PR.sub(1).x_id = [1];
PR.sub(1).y_id = [2, 3];

PR.sub(1).obj(1).name = 'F1';
PR.sub(1).obj(1).args = [1, 2, 3];

PR.sub(1).con(1).name = 'g5';
PR.sub(1).con(1).args = [1, 2, 3];
PR.sub(1).con(2).name = 'g6';
PR.sub(1).con(2).args = [1, 2, 3];
PR.sub(1).con(3).name = 'g9';
PR.sub(1).con(3).args = [2, 3];
PR.sub(1).con(4).name = 'g10';
PR.sub(1).con(4).args = [1, 2];
PR.sub(1).con(5).name = 'g11';
PR.sub(1).con(5).args = [1, 2];

PR.sub(1).to(2).Sij = [1,0; 0,1];
PR.sub(1).to(3).Sij = [0,1; 1,0];

PR.sub(2).x_id = [4, 7];
PR.sub(2).y_id = [5, 6];

PR.sub(2).obj(1).name = 'F4';
```

```
PR.sub(2).obj(1).args = [7];
PR.sub(2).obj(2).name = 'F6';
PR.sub(2).obj(2).args = [4, 7];

PR.sub(2).con(1).name = 'g1';
PR.sub(2).con(1).args = [5, 6, 4, 7];
PR.sub(2).con(2).name = 'g3';
PR.sub(2).con(2).args = [4, 7];
PR.sub(2).con(3).name = 'g7';
PR.sub(2).con(3).args = [5, 6, 4, 7];

PR.sub(2).to(1).Sij = [1,0; 0,1];
PR.sub(2).to(3).Sij = zeros(0, 2);

PR.sub(3).x_id = [8, 11];
PR.sub(3).y_id = [9, 10];

PR.sub(3).obj(1).name = 'F5';
PR.sub(3).obj(1).args = [11];
PR.sub(3).obj(2).name = 'F7';
PR.sub(3).obj(2).args = [8, 11];

PR.sub(3).con(1).name = 'g2';
PR.sub(3).con(1).args = [9, 10, 8, 11];
PR.sub(3).con(2).name = 'g4';
PR.sub(3).con(2).args = [8, 11];
PR.sub(3).con(3).name = 'g8';
PR.sub(3).con(3).args = [9, 10, 8, 11];

PR.sub(3).to(1).Sij = [0,1; 1,0];
PR.sub(3).to(2).Sij = zeros(0, 2);
```

4 Portal frame

Partition specification in Ψ:

```
comp Frame =
|[  extvar A1, A2, A3, I1, I2, I3
          , X11, Y11, M11, X12, Y12, M12
          , X21, Y21, M21, X22, Y22, M22
          , X31, Y31, M31, X32, Y32, M32
    objfunc mass(A1, A2, A3)
    confunc gframe( X11, Y11, M11, X12, Y12, M12
                  , X21, Y21, M21, X22, Y22, M22
                  , X31, Y31, M31, X32, Y32, M32
                  , A1, A2, A3, I1, I2, I3)
]|

comp Beam =
|[  extvar A, I, X1, Y1, M1, X2, Y2, M2
    intvar h, w1, w2, d, t1, t2
    confunc gbeam(X1, Y1, M1, X2, Y2, M2, h, w1, w2, d, t1, t2)
          , gcross(h, w1, w2, d, t1, t2)
          , hcross(A, I, h, w1, w2, d, t1, t2)
]|

syst Portal =
|[  sub F: Frame, B1,B2,B3: Beam
    link F.A1  -- B1.A  , F.I1  -- B1.I
       , F.X11 -- B1.X1 , F.X12 -- B1.X2
       , F.Y11 -- B1.Y1 , F.Y12 -- B1.Y2
       , F.M11 -- B1.M1 , F.M12 -- B1.M2
```

```
       , F.A2  -- B2.A  , F.I2  -- B2.I
       , F.X21 -- B2.X1 , F.X22 -- B2.X2
       , F.Y21 -- B2.Y1 , F.Y22 -- B2.Y2
       , F.M21 -- B2.M1 , F.M22 -- B2.M2
       , F.A3  -- B3.A  , F.I3  -- B3.I
       , F.X31 -- B3.X1 , F.X32 -- B3.X2
       , F.Y31 -- B3.Y1 , F.Y32 -- B3.Y2
       , F.M31 -- B3.M1 , F.M32 -- B3.M2
]|

topsyst Portal
```

Generated matlab file:

```
%
% Generated by np2alc
%
function PR = portal1alc(PR)

PR.main.z_id = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 3
PR.main.z_names = {'A1', 'A2', 'A3', 'I1', 'I2', 'I3', 'X11', 'Y11', 'M11', 'X12', 'Y12', 'M12', 'X21', 'Y21', 'M21', 'X22', 'Y22', 'M22', 'X31', 'Y31', 'M31',
PR.main.z_comps = {'Frame', 'Frame', 'Frame', 'Frame', 'Frame', 'Frame', 'Frame', 'Frame', 'Frame', 'Frame', 'Frame', 'Frame', 'Frame', 'Frame', 'Frame', 'Frame

PR.main.m = 4;
PR.main.comp_id = [1, 2, 3, 4];
PR.main.comp_names = {'F', 'B1', 'B2', 'B3'};

PR.main.obj(1).name = 'none';
PR.main.obj(1).args = [];

PR.main.con(1).name = 'none';
PR.main.con(1).args = [];

PR.sub(1).x_id = [];
PR.sub(1).y_id = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24];

PR.sub(1).obj(1).name = 'mass';
PR.sub(1).obj(1).args = [1, 2, 3];

PR.sub(1).con(1).name = 'gframe';
PR.sub(1).con(1).args = [7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 1, 2, 3, 4, 5, 6];

PR.sub(1).to(2).Sij = [0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0; 0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0; 0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
PR.sub(1).to(3).Sij = [0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0; 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0; 0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0
PR.sub(1).to(4).Sij = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0; 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0; 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

PR.sub(2).x_id = [25, 26, 27, 28, 29, 30];
PR.sub(2).y_id = [31, 32, 33, 34, 35, 36, 37, 38];

PR.sub(2).obj(1).name = 'none';
PR.sub(2).obj(1).args = [];

PR.sub(2).con(1).name = 'gbeam';
PR.sub(2).con(1).args = [33, 34, 35, 36, 37, 38, 25, 26, 27, 28, 29, 30];
PR.sub(2).con(2).name = 'gcross';
PR.sub(2).con(2).args = [25, 26, 27, 28, 29, 30];
PR.sub(2).con(3).name = 'hcross';
PR.sub(2).con(3).args = [31, 32, 25, 26, 27, 28, 29, 30];

PR.sub(2).to(1).Sij = [0,0,0,0,0,0,0,1; 0,0,1,0,0,0,0,0; 0,1,0,0,0,0,0,0; 1,0,0,0,0,0,0,0; 0,0,0,0,1,0,0,0; 0,0,0,0,0,0,1,0; 0,0,0,1,0,0,0,0; 0,0,0,0,0,1,0,0];
PR.sub(2).to(3).Sij = zeros(0, 8);
PR.sub(2).to(4).Sij = zeros(0, 8);

PR.sub(3).x_id = [39, 40, 41, 42, 43, 44];
PR.sub(3).y_id = [45, 46, 47, 48, 49, 50, 51, 52];

PR.sub(3).obj(1).name = 'none';
PR.sub(3).obj(1).args = [];

PR.sub(3).con(1).name = 'gbeam';
PR.sub(3).con(1).args = [47, 48, 49, 50, 51, 52, 39, 40, 41, 42, 43, 44];
PR.sub(3).con(2).name = 'gcross';
PR.sub(3).con(2).args = [39, 40, 41, 42, 43, 44];
PR.sub(3).con(3).name = 'hcross';
PR.sub(3).con(3).args = [45, 46, 39, 40, 41, 42, 43, 44];

PR.sub(3).to(1).Sij = [0,0,0,1,0,0,0,0; 0,0,0,0,0,1,0,0; 0,0,1,0,0,0,0,0; 0,1,0,0,0,0,0,0; 0,0,0,0,0,0,0,1; 0,0,0,0,1,0,0,0; 0,0,0,0,0,0,1,0; 1,0,0,0,0,0,0,0];
PR.sub(3).to(2).Sij = zeros(0, 8);
PR.sub(3).to(4).Sij = zeros(0, 8);

PR.sub(4).x_id = [53, 54, 55, 56, 57, 58];
PR.sub(4).y_id = [59, 60, 61, 62, 63, 64, 65, 66];

PR.sub(4).obj(1).name = 'none';
PR.sub(4).obj(1).args = [];
```

```
PR.sub(4).con(1).name = 'gbeam';
PR.sub(4).con(1).args = [61, 62, 63, 64, 65, 66, 53, 54, 55, 56, 57, 58];
PR.sub(4).con(2).name = 'gcross';
PR.sub(4).con(2).args = [53, 54, 55, 56, 57, 58];
PR.sub(4).con(3).name = 'hcross';
PR.sub(4).con(3).args = [59, 60, 53, 54, 55, 56, 57, 58];

PR.sub(4).to(1).Sij = [0,0,0,0,0,0,1,0; 0,0,0,0,1,0,0,0; 0,0,0,0,0,1,0,0; 0,0,0,1,0,0,0,0; 0,0,0,0,0,0,0,1; 1,0,0,0,0,0,0,0; 0,0,1,0,0,0,0,0; 0,1,0,0,0,0,0,0];
PR.sub(4).to(2).Sij = zeros(0, 8);
PR.sub(4).to(3).Sij = zeros(0, 8);
```