The Probabilistic Code Optimization Module PROCODE

**Knut O. Ronold**
Det Norske Veritas  
NO-1322 Høvik, Norway  
Knut.Ronold@dnv.com

**Rolf Skjong**
Det Norske Veritas  
NO-1322 Høvik, Norway  
Rolf.Skjong@dnv.com

**Abstract**
This paper presents the probabilistic code calibration module implemented with the structural reliability analysis program PROBAN. When a target reliability is specified, the module uses an optimization technique to solve for the requirements to the partial safety factors in a partial safety factor code format. The paper contains an introduction to reliability-based code calibration with due definition of the involved terminology. This is followed by a description of the code calibration module and its features, and two examples of application are briefly outlined.

**Keywords:** Structural Reliability, LRFD, Code Calibration, Optimization

---

1 Introduction

Probabilistic code calibration implies development of design codes that lead to structural designs whose resulting reliabilities exhibit minimum scatter about the target reliability and in particular do not entail overly safe nor too risky designs.

A probabilistic code calibration module PROCODE has been developed and is implemented as an extension of the structural reliability analysis program PROBAN. PROCODE is based on a code theory methodology as described by Hauge et al. (1992). For a description of PROBAN, reference is made to PROBAN (2002), Bjerager (1996) and Skjong et al. (1995).

In the following, a brief introduction to reliability-based code calibration is given together with definitions of some of the terminology, and the particular code calibration module PROCODE implemented with PROBAN is subsequently outlined.

2 Code Calibration

The purpose of a code calibration is to determine a vector of *partial safety factors* for use in the code checks that are executed during a structural design process in order to verify that the *design rules*, as set forth by a structural design code, are fulfilled. An example of such a design rule is the simple inequality

\[ R_D \geq L_D \]  

in which \( R_D = R_C / \gamma_m \) is the design resistance and \( L_D = L_C \gamma_l \) is the design load. The code specifies how to arrive at the characteristic resistance \( R_C \) and the characteristic load \( L_C \) and also specifies values of the partial safety factors \( \gamma_m \) and \( \gamma_l \) to be used. The *design equation* is a special case of the design rule obtained by turning the inequality into an equality, i.e., \( R_D = L_D \) in the example.
With structural reliability methods available, it is possible to determine sets of equivalent partial safety factors which, when applied with design rules in structural design codes, will lead to designs with a prescribed reliability.

One specific combination of environmental loading regime, types of material, and type and shape of structure form a design case. For a particular design case, which can be analyzed by a structural reliability method, a set of partial safety factors can thus be determined that will lead to a design which exactly meets the specified reliability for a given failure mode. Different sets of partial safety factors may result for different design cases and different failure modes.

A structural design code usually has a scope that covers an entire class of structures and is represented by a number of design cases, formed by combinations among multiple environmental loading regimes, several types and shapes of structures, and different structural materials. The design code will usually specify one common set of partial safety factors to be applied for all structures which are to be designed according to the code. This practical simplification implies that the prescribed reliability will usually not be met exactly, but only approximately, when structures are designed according to the code.

Hence, the goal of a reliability-based code calibration is to determine the common set of partial safety factors that reduces the scatter of the reliabilities, achieved by designing structures according to the code, to a minimum. This can be accomplished by means of an optimization technique, once a closeness measure for the achieved reliabilities has been defined, e.g., expressed in terms of a penalty function that penalizes deviations from the prescribed target reliability.

3 Scope of Code

As stated above, the scope of the code is represented by a number of design cases, each of which is a particular combination of environmental loading regime, type of material, and type and shape of structure. The environmental loading regime is usually site-specific and represents a condition that may be considered external to the structure. The type of material and the type and shape of structure are, almost by definition, structure-specific and may thus be considered internal to the structure. Other conditions to characterize the design cases may sometimes apply and can then also be referred to as being either external or internal to the structure. To reduce the amount of input for execution of a code calibration with PROCODE and to operate with a handy input format, the characteristic conditions for each of the design cases within the scope of the code are practically specified by distinction between external conditions and internal conditions. The external and internal conditions are dealt with separately in the following sections.

3.1 External Conditions

The external conditions primarily consist of the environmental conditions such as the environmental loading regimes set up by wind and ocean waves. These are conditions that are often site-specific, i.e., they may change from one location to another. For design of fixed structures founded on soils, soil conditions may also be considered part of the external conditions. For simplicity in the following, the external condi-
tions are considered to consist of environmental conditions. In general, the scope of a code will cover more than one environmental condition, e.g., if the code shall be applicable to a range of locations. Each environmental condition is described by a number of quantities that together define the environmental loads that are applied to a structure which is being designed. For example, when the environmental loading regime is a wave climate, then the significant wave height $H_S$ and the zero-upcrossing period $T_Z$ are two quantities that describe the environmental condition and define the environmental loads.

Therefore, a name set is defined for each environmental condition within the scope of the code. The name set consists of the names of the quantities that define an environmental condition. The purpose of the name set is to associate each physical quantity of the environmental condition with a PROBAN variable to describe it. For the wave climate example, a name set for an environmental condition may thus contain the two quantities $H_S$ and $T_Z$. If the scope of the code covers two environmental conditions, denoted COND1 and COND2, respectively, then the following table can be set up to specify the contents of the name set for each of them:

<table>
<thead>
<tr>
<th></th>
<th>HS</th>
<th>TZ</th>
</tr>
</thead>
<tbody>
<tr>
<td>COND1</td>
<td>VAR3</td>
<td>VAR5</td>
</tr>
<tr>
<td>COND2</td>
<td>VAR7</td>
<td>VAR12</td>
</tr>
</tbody>
</table>

Here, the environmental condition denoted COND1 has a significant wave height $H_S$ that is described by the PROBAN variable VAR3, and a zero-upcrossing period $T_Z$ that is described by the PROBAN variable VAR5. Similarly, $H_S$ and $T_Z$ of the environmental condition denoted COND2 are described by the PROBAN variables VAR7 and VAR12, respectively.

Note that only physical quantities that vary over the scope of the code need to be included in the name sets for the different environmental conditions.

### 3.2 Internal Conditions

The internal conditions consist of structure-related conditions, i.e., conditions which are characteristic for the structural design situation, such as type of material and type and shape of structure. For simplicity in the following, the internal conditions are referred to as design situations. In general, the scope of a code will cover more than one design situation. Each design situation is described by a number of quantities that together define the structure and the material from which it is being constructed. One example of such a quantity is a geometrical measure such as the length-to-depth ratio $L/D$, if a ship structure is considered. Another example is the material strength $R$, if more than one quality grade or more than one material type is to be covered by the scope of the code.

Therefore, a name set is defined for each design situation within the scope of the code. The name set consists of the names of the quantities that define a design situation. The purpose of the name set is to associate each physical quantity of the design situation with a PROBAN variable to describe it. For the ship structure example, a name set for a design situation may thus contain the two quantities $L/D$ and $R$. If the scope of the code covers two design situations, denoted DS1 and DS2, re-
spectively, then the following table can be set up to specify the contents of the name set for each of them

<table>
<thead>
<tr>
<th></th>
<th>LD</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>DS1</td>
<td>VAR2</td>
<td>VAR6</td>
</tr>
<tr>
<td>DS2</td>
<td>VAR8</td>
<td>VAR11</td>
</tr>
</tbody>
</table>

Here, the design situation denoted DS1 has a length-to-depth ratio LD which is described by the PROBAN variable VAR2, and a material strength R which is described by the PROBAN variable VAR6. Similarly, LD and R of the design situation denoted DS2 are described by the PROBAN variables VAR8 and VAR11, respectively.

Note that also for design situations, only physical quantities that vary over the scope of the code need to be included in the name sets for the different design situations.

### 3.3 Scope Matrix

A scope matrix is defined as a matrix with one row entry for each external condition or environmental condition covered by the code and one column entry for each internal condition or design situation covered by the code. For the example with two environmental conditions COND1 and COND2 and two design situations DS1 and DS2, the scope matrix becomes a 2x2 matrix as follows

\[
\begin{bmatrix}
(COND1, DS1) & (COND1, DS2) \\
(COND2, DS1) & (COND2, DS2)
\end{bmatrix}
\]

It follows that a design case is an entry in the scope matrix, i.e., a combination of a particular environmental condition and a particular design situation. In some cases, only a reduced scope is of interest. This is a scope that contains only a selection of entries from the scope matrix. This applies to cases for which at least one of the covered environmental conditions and one of the covered design situations do not combine to a design case of practical interest.

### 4 Failure Modes

For a particular design case, the structure is to be designed against failure in one or more failure modes, such as buckling, fatigue, etc. One such particular failure mode is considered in the following. The safety of the structure is governed by the vector \( X \) of stochastic variables, which are not different from one design case to another. The vector \( E \) of environmental condition variables, which may be different from one design case to another, the vector \( D \) of design situation variables, which may also be different from one design case to another, and the vector \( \theta \) of design parameters. Whereas the vectors \( X, E, \) and \( D \) all consist of stochastic and deterministic variables that cannot be controlled, the vector \( \theta \) of design parameters is a vector of parameters such as geometrical measures (length, wall thickness, cross-sectional area, section modulus), which can be controlled during the design, and which the designer is free to choose in order to make his design as safe as desirable.
4.1 Limit State Function
A limit state function \( G(X,E,D,\theta) \) for the considered failure mode is established in compliance with the requirements to such a function, namely
\[
G(X,E,D,\theta) > 0
\]  
(2)
in the safe set of \((X,E,D,\theta)\), and \( G(X,E,D,\theta) \leq 0 \) otherwise. The failure probability is accordingly defined as
\[
P_f = P[G(X,E,D,\theta) \leq 0]
\]  
(3)
and based on this the reliability index is defined as
\[
\beta = -\Phi^{-1}(P_f)
\]  
(4)

4.2 Code Check Function
When carrying out a design according to a code, one or more code checks are to be performed. A code check function \( h(x,e,d,\theta,\gamma) \) is defined for each code check that is to be performed, and is expressed in terms of a fixed realization \((x,e,d)\) of the variables \((X,E,D)\), the set of design parameters \(\theta\), and the set of partial safety factors \(\gamma\). The fixed realization \((x,e,d)\) is a set of prescribed characteristic values of the variables \((X,E,D)\), usually taken as a set of quantiles of the distributions of these variables. The code check function can be derived from the limit state function \( G(X,E,D,\theta) \), or it can be expressed in some other way. The code check function has to be defined in such a way that the code check consists in verifying that the code check function does not take on a negative value, hence
\[
h(x,e,d,\theta,\gamma) \geq 0
\]  
(5)
This inequality, which forms the requirement to be fulfilled by the design and verified by the code check, is sometimes also referred to as a design rule. The code check limit \( h(x,e,d,\theta,\gamma)=0 \) is correspondingly referred to as the design equation.

5 Code Optimization
A code is usually developed for structural design against a series of specific failure modes, such as buckling, fatigue, etc. The corresponding code calibration may therefore involve several failure modes. Let \( M \) denote the number of failure modes that pertain to a considered design case, and let \( N_k \) be the number of code check functions that pertain to the \( k \)th failure mode. Thus, for the considered design case, there are \( M \times N_k \) code checks with requirements
\[
h_{k,i}(x,e,d,\theta,\gamma) \geq 0, \quad k = 1,...,M \quad n = 1,...,N_k
\]  
(6)
and \( M \) limit state function requirements
\[
G_i(X,E_j,D_j,\theta,\gamma) \geq 0, \quad k = 1,...,M
\]  
(7)
in which \( ij \) denotes the entry in the scope matrix that corresponds to the design case, with index \( i \) referring to environmental condition and index \( j \) referring to design situation.
5.1 Optimal Code – One Design Case

A first implication of the code philosophy is that the code check limit
\[ h_{i,j}(x_i, e_j, d_j, \theta_j, \gamma) = 0 \] (8)
shall be fulfilled for at least one code check function for each failure mode, i.e., for at least one \( n \in [1, N_k] \) for each \( k \in [1, M] \) for the \( ij \)th design case.

A second implication of the code philosophy is that the scatter of the resulting reliabilities by the code about the prescribed target reliability shall be minimized. To facilitate this minimization, a penalty \( p \) is introduced, expressed as a function of the difference between achieved and target reliability index, and defined such that the function value takes on its minimum value when this reliability difference is zero. For a single design case governed by \( M \) failure modes, the sought-after optimal set of partial safety factors \( \gamma \) is achieved as the solution of the following optimization problem

\[ \min_{\theta_i, \gamma} \sum_{k=1}^{M} w_{\text{mode},k} p(\beta_k(X, E_i, D_j, \theta_j) - \beta_{T,k}) \] (9)
subject to \( \{h_{i,j}(x_i, e_j, d_j, \theta_j, \gamma) \geq 0, k = 1, \ldots M, n = 1, \ldots N_k \} \)
with equality fulfilled in the constraints for at least one \( n \) for each \( k \). The achieved reliability index for the \( k \)th failure mode is \( \beta_k(X, E_i, D_j, \theta_j) \) and the corresponding target reliability index is \( \beta_{T,k} \). The weighting factor \( w_{\text{mode},k} \) gives the importance assigned to the \( k \)th failure mode and should ideally reflect the expected future relative frequency of this failure mode. Reference is made to Hauge et al. (1992).

5.2 Optimal Code – Multiple Design Cases

For a class of design cases, represented by the scope matrix, the sought-after optimal set of partial safety factors \( \gamma \) is achieved as the solution of the following optimization problem

\[ \min_{\theta_i, \gamma} \sum_{i,j} \sum_{k=1}^{M} w_{\text{dc},ij} w_{\text{mode},k} p(\beta_k(X, E_i, D_j, \theta_j) - \beta_{T,k}) \] (10)
subject to \( \{h_{i,j}(x_i, e_j, d_j, \theta_j, \gamma) \geq 0, k = 1, \ldots M, n = 1, \ldots N_k \} \) for all selected design cases \( ij \)
with equality fulfilled in the constraints for at least one \( n \) for each \( k \) within each design case \( ij \).

The achieved reliability index for the \( k \)th failure mode is \( \beta_k(X, E_i, D_j, \theta_j) \) and the corresponding target reliability index is \( \beta_{T,k} \). As before, the weighting factor \( w_{\text{mode},k} \) gives the importance assigned to the \( k \)th failure mode. The weighting factor \( w_{\text{dc},ij} \) gives the importance assigned to the \( ij \)th design case and should ideally reflect the expected future demand of this design case. Note in this context that one should be careful not to use too small weighting factors for any design case, as the optimization then might result in acceptance of design cases whose reliability indices are actually unacceptable. A more flexible formulation of this optimization problem would be achieved if the
two vectors of weighting factors with entries $w_{\text{mode},k}$ and $w_{\text{dc},ij}$, respectively, were merged into one with entries $w_{ijk}$.

For the case that the two vectors of weighting factors are merged into one and only one failure mode with one code check function is considered, the optimization problem in Eq. (10) reduces to

$$\min_{\theta,y} \sum_{i,j} w_{ij} p(\beta(X,E_i,D_j,\theta_j) - \beta_1)$$  \hspace{1cm} (11)

subject to

$$\{h(x,e,d,\theta_j) = 0\} \text{ for all selected design cases } ij$$

Note that this formulation of the optimization problem with equality required in the constraints reflects the deterministic design procedure, in which the partial safety factors are to be used, and in which the designer will go for equality for his particular design in order to achieve an optimal design. In other words, the philosophy applied by the designer in the design procedure is used also in the formulation of the optimization problem for the safety factor calibration.

6 Application

PROCODE is developed as separate application software that links with the latest version of PROBAN. The commands and numeric facilities of PROBAN are therefore accessible also in PROCODE. Additional commands required for the description of the code calibration problem are exclusive PROCODE commands and are in general not available with PROBAN.

6.1 Input Requirements

The PROBAN analyses and the code check function calculations, which are performed several times during the execution of a code optimization by PROCODE, have different input from one analysis to the next. This is so, because different design cases with different environmental conditions and different design situations are covered, and because the design parameters and the partial safety factors become changed as the optimization proceeds.

As thus several sets of stochastic variables are needed one after another for input to PROBAN, one particular such set cannot be specified directly as one would normally do in a standard PROBAN analysis. Instead, one must specify a generic variable set as input. Such variables are assigned names instead of numerical values or distributions, and these names then serve as pointers to allow for picking the particular set of stochastic variables that is needed for a particular design-case-specific PROBAN analysis during the PROCODE execution.

To facilitate this, to reduce the amount of input required, and to minimize the management of this input, four generic variable types are introduced into PROBAN. These generic variable types allow for specification of all variables that need to be assigned values during the execution of the code optimization. The generic variable types are accessible only through PROCODE. The generic variable types, which are
used to create variables needed by limit state functions and code check functions, are listed below:

ENVIRONMENT
DESIGN-SITUATION
DESIGN-PARAMETER
SAFETY-FACTOR

The event that a structural element fails in a specific failure mode is part of the code calibration problem. In a probabilistic formulation such as in PROBAN, this event is represented by a negative outcome of the limit state function associated with the failure mode. In a deterministic formulation as in a design code, it is represented by negative outcomes of one or more code check functions.

A set of code check functions is associated with each failure mode, expressed in terms of characteristic values of the governing variables, of the design parameters, and of a set of partial safety factors. For valid designs, each code check function within this set needs to be positive valued. Such a positive valued code check function implies a code check that is fulfilled. The fulfilled code checks appear as constraints in the code optimization problem. The code check functions are specified by the CODE-FUNCTION functionality, which is similar to the EVENT functionality that is used for representation of negative-valued limit state functions in a probabilistic formulation. By creating a CODE-FUNCTION, a variable which is created as a function of fixed-valued quantities is specified to be a code check function with a specific name, and PROCODE will automatically create the code check inequality and apply the correspondingly fulfilled code check as an optimization constraint.

In addition, an importance value, a target reliability index, and a penalty function are associated with each failure mode. The importance value is a weighting factor for the failure mode and represents the expected future demand of this failure mode. The penalty function is specified by the PENALTY-FUNCTION functionality, which is similar to the CODE-FUNCTION functionality. By creating a PENALTY-FUNCTION, a variable that is created as a function of fixed-valued quantities and the deviation between achieved and target reliability index, is specified to be a penalty function with a specific name.

A failure mode is specified by CREATE FAILURE-MODE. The failure mode is given a name, and an event, a set of code check functions, a penalty function, a target reliability index, and a weighting factor are assigned to it.

A set of failure modes is specified by CREATE MODE-SET. The failure mode set is given a common name, and a number of failure modes are assigned to it by reference to their names.

6.2 Results
The results from a PROCODE code calibration consist of the optimized set of partial safety factors $\gamma$ and the corresponding optimal design parameters $\theta$. Print and display of the reliability scatter inherent in the code are available. Also the values of the code check functions corresponding to the optimized parameters are available for each design case.
The results from a PROCODE code evaluation are the same as for a PROCODE code calibration, except that the safety factors are preset and not optimized. The code evaluation feature of PROCODE is handy for evaluation of existing codes and is not nearly as computation intensive as a full code calibration by optimization.

7 List of Applications
PROCODE has been used extensively by Det Norske Veritas since the first version was developed in 1990. Many of the studies relate to the development of ship rules. Example codes that have been published are
- design of jack-ups spudcans to prevent punching and tubular members to prevent buckling
- design of offshore tension piles against pull-out
- design of wind-turbine rotor blades against fatigue
- design of ship structures against hull girder collapse
Reference is made to Hauge et al. (1992), Ronold (1999), Ronold and Christensen (2001), and Skjong and Bitner-Gregersen (2002), Bitner-Gregersen et al. (2002).

8 Examples
Results from a couple of applications of PROCODE are presented briefly in the following two sections.

8.1 Fatigue Design of Wind-Turbine Rotor Blades
A code for the design of wind-turbine rotor blades against fatigue failure in flapwise bending is considered. The rotor blades are constructed from fibre-reinforced polyester laminate. The critical location for accumulation of fatigue damage is assumed to be at the blade root. The loading consists of a history of bending moment ranges owing to wind exposure. The capacity is calculated on the basis of a Miner’s rule approach, based on a conventional S-N curve formulation.

The scope of code is represented by 18 design cases established as the possible combinations of 3 wind turbines, 3 locations, and 2 blade materials. The 3 locations represent 3 wind climates, i.e. 3 “environmental conditions”, each of which is represented by a set of distributions of mean wind speed and turbulence intensity. The 3 wind turbines represent 3 blade structures, each of which has its own bending moment response distribution at the blade root, conditioned on the wind climate parameters. The two materials are characterized by two different S-N curves, thus representing two different material strengths. The 3x2=6 combinations of wind turbine and material form 6 “design situations” in the PROCODE terminology. There is one failure mode and one associated code check function per design case.

The section modulus at the blade root, which is used to transform bending moment ranges to stress ranges, is used as the design parameter. A load factor $\gamma_l$ is applied as a factor on all stress ranges in some specified characteristic distribution of stress ranges over the design lifetime. In similar manner, a materials factor $\gamma_m$ is applied as a divisor on all stress ranges in the characteristic S-N curve. A rather detailed format for $\gamma_l$ and $\gamma_m$ is adopted,
\[
\gamma_f = 1.0 + d \frac{R^b}{n_{10}^c}, \quad \gamma_m = d + e \sigma_c^f
\]

in which \(a, b, c, d, e, \) and \(f\) are coefficients which are optimized by PROCODE. Further, \(R\) is the rotor radius of the wind turbine, \(n_{10}\) is the number of 10-minute series of observed bending moment ranges on the wind turbine, used to establish the characteristic load distribution, and \(\sigma_c\) is the standard deviations of the residuals in log \(N\) about the mean \(S-N\) curve. The detailed format of the partial safety factors serves to adequately reflect the variability and uncertainty in the governing variables. In particular, by the dependency on \(n_{10}\), the format serves to reflect the load data quality and honors the designer who is willing to spend more on testing his structure and become more certain about his load distributions.

The target annual failure probability for this calibration is set to \(10^{-6}\) with a corresponding target reliability index \(\beta_T=4.265\). This applies to the last year of service during a 20-year design life. The results of the calibration by means of PROCODE are reproduced in Table 1. The table gives the optimized safety factors and the achieved reliability indices for the 18 design cases modeled to represent the scope of code. The resulting scatter in the achieved reliability indices from the target is also given. For further details, reference is made to Ronold and Christensen (2001).

<table>
<thead>
<tr>
<th>Table 1</th>
<th>Results of Code Optimization for Wind-Turbine Rotor Blades</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partial safety factor</td>
<td>Optimized safety factor</td>
</tr>
<tr>
<td>(\gamma_f)</td>
<td>(1.0 + 0.0590 \frac{R^{0.956}}{n_{10}^{0.250}})</td>
</tr>
<tr>
<td>(\gamma_m)</td>
<td>(0.998 + 0.437\sigma_c^{1.444})</td>
</tr>
<tr>
<td>Design case</td>
<td>Achieved reliability index</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>11</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td>4</td>
<td>13</td>
</tr>
<tr>
<td>5</td>
<td>14</td>
</tr>
<tr>
<td>6</td>
<td>15</td>
</tr>
<tr>
<td>7</td>
<td>16</td>
</tr>
<tr>
<td>8</td>
<td>17</td>
</tr>
<tr>
<td>9</td>
<td>18</td>
</tr>
</tbody>
</table>

### 8.2 Buckling Design of Ship Hull Girders

A code for the design of ship hull girders against buckling is considered. Four different failure modes in buckling are considered, each of which has one associated code check function. The four failure modes are plate buckling, lateral stiffener buckling, torsional stiffener buckling, and web stiffener buckling. The loading consists of bend-
ing moments in the extreme sagging condition in heavy seas. The capacity is formed by the critical buckling stress in the buckling mode of interest.

The scope of code is represented by 24 design cases established as the possible combinations of 1 ocean environment, 1 ship structure, 12 different local designs, and 2 different stiffener profiles. The ship structure considered is a VLCC tanker. The stiffener profiles considered are L-profiles and flat bar profiles. Thus, in PROCODE terminology, there is one “environmental condition” and 24 “design situations”.

The plate thickness and the web thickness are used as design parameters for the code calibration. One partial safety factor is defined for each of the four buckling modes. Each factor is applied as a factor on the critical buckling stress in the associated code check function.

The code evaluation feature of PROCODE is used to obtain the plate and web thicknesses for the 24 design cases when their designs are carried out according to the design criteria for buckling in the current DNV rules for classification of ships. The corresponding 24 achieved reliability indices are also calculated.

The code calibration feature of PROCODE is used to calibrate the four partial safety factors that minimize the deviations of the achieved reliability indices from target. This is done for four different assumptions of the target reliability index, corresponding to four different levels of safety, and the implied plate and web thicknesses for the 24 design cases are calculated for each of the four target safety levels. The resulting plate and web thicknesses directly influence the cost of the ship structure, and the results of this PROCODE application have been used to study the trade-off between cost and safety. For further details, reference is made to Skjong and Bitner-Gregersen (2002).

9 Conclusions and Discussion

The methodology implemented in PROCODE for calibration of partial safety factors implies minimization of a penalty function. The penalty function penalizes deviations of the achieved reliabilities from the target reliability over the design cases, which represent the scope of the code. The minimization is carried out with constraints. There is one constraint per code check function for each failure for each defined design case. Each constraint consists of the fulfilled design inequality for the corresponding code check. For each design case, it is further required that the design equality is fulfilled for at least one code check function per failure mode. This corresponds to practical design where one code check will be determining the dimensions. When there is only one code check function per failure mode, it appears that all constraints consist of fulfilled design equalities.

The partial safety factors, which are determined by the constrained minimization of the penalty function, appear only in the constraints, where they are coefficients in the code check functions, but they do not appear in the penalty function itself. It is noted that the methodology implemented in PROCODE ensures that the design inequality is fulfilled for all code checks and that the design equality is fulfilled for at least one code check per failure mode per design case. This reflects the philosophy of the design procedure, where the partial safety factors are used, and by which the designer
will optimize his design by fulfilling the design inequalities for his particular case and ensuring that also at least one design equality is fulfilled for each failure mode considered.

The proposed calibration procedure makes it easy to implement the extract of advanced reliability analysis into simple design rules. The referenced studies also allow for studying cost and safety tradeoffs. This is documented in the reference papers. Savings are documented both in moving from existing codes to calibrated (optimized) codes, and in the case of Hauge et al. (1992) the further saving in moving from optimized codes to probabilistic design is demonstrated.

10 References


