Memorandum M- 690

USER’S GUIDE FOR ‘AMCYLAN’

- Computational package for asymptotic multi-mode analysis of (im-)perfect cylindrical shells -

Ir. G. Rebel

May, 1994

Delft University of Technology
faculty of Aerospace Engineering
Delft, The Netherlands
USER GUIDE FOR AMCYLAN - Computational package for asymptotic multi-mode analysis of (im-)perfect cylindrical shells

ir. G. Rebel
abstract

ABSTRACT

The computational package AMCYLAN consists of computational modules with which an asymptotic multi-mode analysis can be performed for (im-)perfect anisotropic cylindrical shells which may be stiffened in axial and/or circumferential direction. The stiffeners are taken into account as “smeared” stiffeners. The nonlinear Donnell-Mushtari-Vlasov or Sanders or Novozhilov type shell equations can be chosen for the analysis. The circumferential dependence is eliminated by Fourier decomposition. The resulting sets of ordinary differential equations are solved numerically using a finite difference formulation. Thus the specified boundary conditions can be enforced rigorously in the prebuckling, buckling and postbuckling problems. The prebuckling problem can be either membrane or nonlinear. Computational package AMCYLAN can take into account mode interaction of a number of buckling modes in the postbuckling problem as well as in the analysis of the imperfection sensitivity.
**TABLE OF CONTENTS**

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>i</td>
</tr>
<tr>
<td>LIST OF SYMBOLS</td>
<td>iv</td>
</tr>
<tr>
<td>INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1 COMPUTATIONAL PACKAGE AMCYLAN</td>
<td>2</td>
</tr>
<tr>
<td>1.1 Purpose</td>
<td>2</td>
</tr>
<tr>
<td>1.2 Problem notation</td>
<td>2</td>
</tr>
<tr>
<td>1.3 Computational stream diagram of package AMCYLAN</td>
<td>7</td>
</tr>
<tr>
<td>2 COMPUTATIONAL MODULE ANOVI</td>
<td>10</td>
</tr>
<tr>
<td>2.1 Standard input file : ANOVI.INP</td>
<td>10</td>
</tr>
<tr>
<td>2.2 Data input file (problem definition file)</td>
<td>11</td>
</tr>
<tr>
<td>2.2.1 Group 0 : SUBJECT</td>
<td>11</td>
</tr>
<tr>
<td>2.2.2 Group 1 : SHELL DATA</td>
<td>11</td>
</tr>
<tr>
<td>2.2.3 Group 2 : FINITE DIFFERENCE</td>
<td>14</td>
</tr>
<tr>
<td>2.2.4 Group 3 : DATA FOR DIMENSIONLESS WRITING</td>
<td>14</td>
</tr>
<tr>
<td>2.2.5 Group 4 &amp; 5 : SHELL THEORY CHOICE</td>
<td>14</td>
</tr>
<tr>
<td>2.2.6 Group 4 : PREBUCKLING PROBLEM</td>
<td>16</td>
</tr>
<tr>
<td>2.2.6.a MEMBRANE PREBUCKLING</td>
<td>16</td>
</tr>
<tr>
<td>2.2.6.b NONLINEAR PREBUCKLING</td>
<td>17</td>
</tr>
<tr>
<td>2.2.7 Group 5 : BUCKLING PROBLEM</td>
<td>23</td>
</tr>
<tr>
<td>2.2.7.1 Group 5.1 : Boundary Conditions (B.C.) buckling problem</td>
<td>25</td>
</tr>
<tr>
<td>2.2.7.2 Group 5.2 : Buckling problem parameters</td>
<td>27</td>
</tr>
<tr>
<td>3 COMPUTATIONAL MODULE POSTPR</td>
<td>32</td>
</tr>
<tr>
<td>3.1 Standard input file : POSTPR.INP</td>
<td>32</td>
</tr>
<tr>
<td>3.2 Mode identifying file &amp; post-buckling boundary condition file</td>
<td>34</td>
</tr>
<tr>
<td>3.2.1 Group 1 : POST-BUCKLING BOUNDARY CONDITIONS</td>
<td>35</td>
</tr>
<tr>
<td>3.2.2 Group 2 : BUCKLING MODE SPECIFYING</td>
<td>36</td>
</tr>
<tr>
<td>4 COMPUTATIONAL MODULE IMPSHP</td>
<td>39</td>
</tr>
<tr>
<td>4.1 Standard input file : IMPSHP.INP</td>
<td>39</td>
</tr>
<tr>
<td>5 COMPUTATIONAL MODULE IMPFAC</td>
<td>42</td>
</tr>
<tr>
<td>5.1 Standard input file : IMPFAC.INP</td>
<td>42</td>
</tr>
</tbody>
</table>
6 COMPUTATIONAL MODULE COHPAT
   6.1 Standard input file : COHPAT.INP
   6.2 Calculation control file
      6.2.1 Path-following parameters
      6.2.2 Imperfection amplitudes

REFERENCES

APPENDIX A - Input files for program ANOVSI
   A.1 Standard input file ANOVSLINP : example listing
   A.2 Data input file : example listing KHOT_11_13.INP

APPENDIX B - Input files for program POSTPR
   B.1 Standard input file POSTPR.INP : example listing
   B.2 Mode identifying file & post-buckling boundary condition file :
       example listing BCMODE.INP

APPENDIX C - Standard input file for program IMPSHIP

APPENDIX D - Standard input file for program IMPFAC

APPENDIX E - Input files for program COHPAT
   E.1 Standard input file COHPAT.INP : example listing
   E.2 Calculation control file : example listing CONTRL.INP
**LIST OF SYMBOLS**

\[ \begin{align*}
  a_i^L, b_{ijk}^L & \quad \text{postbuckling coefficients (see eq. 1.14)} \\
  c & \quad = \sqrt{3(1 - v^2)} \\
  C_s, C_r & \quad \text{eccentricity of the stringers and rings, respectively (see fig. 2-1)} \\
  D_s, D_r & \quad \text{spacing of the stringers and rings, respectively (see fig. 2-1)} \\
  \Delta \vec{D} & = \Delta \vec{D}_c + \Delta \vec{D}^* & \text{vector of prescribed displacements and rotations (see eq. 1.13)} \\
  E & \quad \text{arbitrarily chosen reference Young's modulus} \\
  \vec{f} & \quad \text{vector of equations } f_i \text{ (see eq. 6.8, 6.10 and 6.11)} \\
  \vec{F} = F_c + \Delta \vec{F}^* & \quad \text{load vector of forces, moments and external pressure (see eq. 1.12)} \\
  G & \quad \text{matrix in the Newton-iteration equations (see eq. 6.9)} \\
  h & \quad \text{finite difference stepsize (see fig. 2-2)} \\
  j^* & \quad \text{user defined target iteration count (see eq. 6.3)} \\
  J_l & \quad \text{Newton-iteration count for solution point } l \text{ (see eq. 6.3)} \\
  K & \quad \text{submatrix of matrix } G \text{ (see eq. 6.9)} \\
  L & \quad \text{cylinder length (see fig. 1-1)} \\
  \tilde{m} & \quad \text{nondimensional bending moment parameter (see eq. 1.18)} \\
  m_0 & \quad \text{prebuckling moment stress resultant function (see eq. 1.18)} \\
  m_{ki} & (k = 1,2) & \quad \text{buckling moment stress resultant functions (see eq. 1.22 and section 2.2.7.2)} \\
  m_{kij} & (k = 1,\ldots,4) & \quad \text{postbuckling moment stress resultant functions (see eq. 1.30)} \\
  M & \quad \text{moment stress resultant about the } y \text{-axis (see fig. 1-1)} \\
  \hat{M} & \quad \text{dimensionless } M \text{ (see eq. 1.4)} \\
  M_0, M_1, M_{ij} & \quad \text{zeroth order, first order and second order fields of } \hat{M} \text{ (see eq. 1.10)} \\
  \bar{M} & \quad \text{moment line load about the } y \text{-axis at } x=L \text{ (see fig. 1-2)} \\
  \bar{M} + \Delta \bar{M} & \quad \text{moment line load about the } y \text{-axis at } x=0 \text{ (see fig. 1-2)} \\
  n_l & \quad \text{number of full waves in circumferential direction} \\
  p_e & \quad \text{external dead load pressure (see fig. 1-2)} \\
  \hat{p}_e & \quad \text{dimensionless } p_e \text{ (see eq. 1.35 and fig. 2-3)} \\
  p_{fe} & \quad \text{external follower pressure load (see fig. 1-2)} \\
  \hat{p}_{fe} & \quad \text{dimensionless } p_{fe} \text{ (see eq. 1.35 and fig. 2-3)} \\
  p_0 & \quad \text{prebuckling axial stress resultant function (see eq. 1.15)} \\
  p_{ki} & (k = 1,2) & \quad \text{buckling axial stress resultant functions (see eq. 1.19 and section 2.2.7.2)} \\
  p_{kij} & (k = 1,\ldots,4) & \quad \text{postbuckling axial stress resultant functions (see eq. 1.27)} \\
  P & \quad \text{axial stress resultant (see fig. 1-1)} \\
  \hat{P} & \quad \text{dimensionless } P \text{ (see eq. 1.1)}
\end{align*} \]
list of symbols

\( P_0, P_1, P_{ij} \)  
zeroth order, first order and second order fields of \( \hat{P} \) (see eq. 1.7)

\( \hat{P} \)  
alxial line load (tension positive ; see fig. 1-2)

\( \hat{q} \)  
nondimensional transverse load parameter (see eq. 1.17)

\( q_0 \)  
prebuckling transverse stress resultant function (see eq. 1.17)

\( q_kl \)  
(k = 1, 2)

buckling transverse stress resultant functions
(see eq. 1.21 and section 2.2.7.2)

\( q_{klij} \)  
(k = 1, ..., 4)

postbuckling transverse stress resultant functions (see eq. 1.29)

\( \hat{Q} \)  
transverse stress resultant (see fig. 1-1)

\( \hat{\Omega} \)  
dimensionless \( Q \) (see eq. 1.3)

\( \hat{Q}_0, \hat{Q}_1, \hat{Q}_{ij} \)  
zeroth order, first order and second order fields of \( \hat{Q} \) (see eq. 1.9)

\( \overline{\hat{Q}} \)  
transverse line load at \( x=L \) (inward positive ; see fig. 1-2)

\( \overline{\hat{Q}} + \Delta \overline{\hat{Q}} \)  
transverse line load at \( x=0 \) (outward positive ; see fig. 1-2)

\( R \)  
cylinder radius (see fig. 1-1)

\( \hat{s} \)  
nondimensional torsional load parameter (see eq. 1.16)

\( s_0 \)  
prebuckling torsional stress resultant function (see eq. 1.16)

\( s_{kij} \)  
(k = 1, 2)

buckling torsional stress resultant functions
(see eq. 1.20 and section 2.2.7.2)

\( s_{klij} \)  
(k = 1, ..., 4)

postbuckling torsional stress resultant functions (see eq. 1.28)

\( \hat{S} \)  
torsional stress resultant (see fig. 1-1)

\( \hat{S} \)  
dimensionless \( S \) (see eq. 1.2)

\( S_0, S_1, S_{ij} \)  
zeroth order, first order and second order fields of \( \hat{S} \) (see eq. 1.8)

\( \overline{\hat{S}} \)  
torsional line load (counter clockwise positive ; see fig. 1-2)

\( t \)  
cylinder wall thickness (see fig. 1-1)

\( u \)  
displacement component in the \( x \)-direction (see fig. 1-1)

\( \hat{u} \)  
dimensionless \( u \) (see eq. 1.1)

\( \hat{u}_0 \)  
prescribed displacement in \( x \)-direction at \( x=0 \) (see eq. 1.13)

\( \hat{u}_L \)  
prescribed displacement in \( x \)-direction at \( x=L \) (see eq. 1.13)

\( u_0 \)  
prebuckling axial displacement function (see eq. 1.15)

\( u_{kij} \)  
(k = 1, 2)

buckling axial displacement functions
(see eq. 1.23 and section 2.2.7.2)

\( u_{klij} \)  
(k = 1, ..., 4)

postbuckling axial displacement functions (see eq. 1.31)

\( U_0, U_1, U_{ij} \)  
zeroth order, first order and second order fields of \( \hat{U} \) (see eq. 1.7)

\( v \)  
dimensionless \( v \) (see eq. 1.2)

\( \hat{v}_0 \)  
prescribed displacement in \( y \)-direction at \( x=0 \) (see eq. 1.13)

\( \hat{v}_L \)  
prescribed displacement in \( y \)-direction at \( x=L \) (see eq. 1.13)

\( v_0 \)  
prebuckling circumferential displacement function (see eq. 1.16)
list of symbols

$v_{ki}$ ($k = 1,2$) \hspace{2cm} buckling circumferential displacement functions
(see eq. 1.24 and section 2.2.7.2)

$v_{kij}$ ($k = 1,\ldots,4$) \hspace{2cm} postbuckling circumferential displacement functions (see eq. 1.32)

$V_0$, $V_1$, $V_{ij}$ \hspace{2cm} zeroth order, first order and second order fields of $\hat{v}$ (see eq. 1.8)

$w$ \hspace{2cm} displacement component in the z-direction (see fig. 1-1)

$\hat{w}$ \hspace{2cm} dimensionless $w$ (see eq. 1.3)

$\hat{w}_0$ \hspace{2cm} prescribed displacement in z-direction at $x=0$ (see eq. 1.13)

$\hat{w}_L$ \hspace{2cm} prescribed displacement in z-direction at $x=L$ (see eq. 1.13)

$w_0$ \hspace{2cm} prebuckling radial displacement function (see eq. 1.17)

$w_{ki}$ ($k = 1,2$) \hspace{2cm} buckling radial displacement functions
(see eq. 1.25 and section 2.2.7.2)

$w_{kij}$ ($k = 1,\ldots,4$) \hspace{2cm} postbuckling radial displacement functions (see eq. 1.33)

$\bar{w}$ \hspace{2cm} initial radial imperfection ($\bar{w} = t \bar{\xi}_m \overline{W}_m$)

$\Delta \bar{w}$ \hspace{2cm} dimensionless $\bar{w}$ (see eq. 1.6)

$\bar{w}_{ij}$ \hspace{2cm} modal initial radial imperfection function (see eq. 4.2)

$\overline{W}_m$ \hspace{2cm} imperfection shape function (see eq. 1.11)

$W_0$, $W_1$, $W_{ij}$ \hspace{2cm} zeroth order, first order and second order fields of $\hat{w}$ (see eq. 1.9)

$\hat{x}$ \hspace{2cm} vector of perturbation parameters augmented with the load factor (see eq. 6.8)

$x, y$ \hspace{2cm} axial and circumferential coordinates on the midsurface of the
cylinder, respectively (see fig. 1-1)

$\bar{x}, \bar{y}$ \hspace{2cm} nondimensional coordinates ($\bar{x} = x/R$, $\bar{y} = y/R$)

$z$ \hspace{2cm} coordinate normal to the midsurface of the cylinder
(positive inward; see fig. 1-1)

$\alpha$ \hspace{2cm} approach factor for computing the increase in $\Lambda_e$ (see fig. 2-4)

$\alpha_m^L$, $\beta_m^L$ \hspace{2cm} imperfection form factors (see eq. 1.14)

$\beta$, $\hat{\beta}$ \hspace{2cm} rotation about the y-axis (see fig. 1-2 and eq. 2.7; $\hat{\beta} = \beta$)

$\hat{\beta}_0$ \hspace{2cm} prescribed rotation about the y-axis at $x=0$ (see eq. 1.13)

$\hat{\beta}_L$ \hspace{2cm} prescribed rotation about the y-axis at $x=L$ (see eq. 1.13)

$\beta_0$ \hspace{2cm} prebuckling rotation function (see eq. 1.18)

$\beta_{ki}$ ($k = 1,2$) \hspace{2cm} buckling rotation functions (see eq. 1.26 and section 2.2.7.2)

$\beta_{kij}$ ($k = 1,\ldots,4$) \hspace{2cm} postbuckling rotation functions (see eq. 1.34)

$\beta_0$, $\beta_1$, $\beta_{ij}$ \hspace{2cm} zeroth order, first order and second order fields of $\hat{\beta}$ (see eq. 1.10)

$\gamma_{xy}$ \hspace{2cm} in-plane nonlinear shearing strain with respect to the cylinder
coordinate system (see eq. 2.3)
list of symbols

\( \gamma_{xy} \)
= \( v_x + u_y \) = shearing strain from linear shell theory

\( \delta \)
characteristic displacement (see fig. 2-4)

\( \varepsilon_{x}, \varepsilon_{y} \)
in-plane nonlinear normal strains with respect to the cylinder coordinate system (see eq. 2.1 and 2.2)

\( \eta \)
path parameter (see eq. 6.11)

\( \eta_I \)
path parameter \( \eta \) at point \( I \) (see fig. 6-1)

\( \eta_l \)
path parameter \( \eta \) at limit point \( I=l \) (see fig. 6-1)

\( \eta_s \)
start value for path parameter \( \eta \) (see fig. 6-1)

\( \eta_c^i \)
= \( \mu^i + \rho_c^i \) (see fig. 2-5)

\( \theta_k \)
angle of rotation of layer \( k \) with respect to the \( x \)-coordinate of the cylinder (see fig. 2-1)

\( \kappa_{x}, \kappa_{y}, \kappa_{xy} \)
curvature components (see eq. 2.4, 2.5 and 2.6)

\( \Lambda \)
nondimensional variable load factor (see eq. 1.12 and 1.13)

\( \Lambda_{BIF}^i \)
bifurcation loads computed from load level \( \Lambda_e^i \) (see fig. 2-5)

\( \Lambda_c \)
critical bifurcation load (see fig. 2-4)

\( \Lambda_c^i \)
critical bifurcation load computed from load level \( \Lambda_e^i \) (see fig. 2-5)

\( \Lambda_e^i \)
load level at which the prebuckling state is evaluated

\( \Lambda_e \)
load level \( \Lambda_e \) at which the \( i^{th} \) eigenvalue problem for computing bifurcation loads is formulated (see fig. 2-4 and 2-5)

\( \Lambda_L \)
bifurcation load of buckling mode \( L \) (see eq. 1.14)

\( \Lambda_s \)
starting value for load factor \( \Lambda \) for solving eq. (6.1) (see fig. 6-1)

\( \lambda \)
nondimensional axial load parameter (see eq. 1.15)

\( \mu^i \)
eigenvalue shift (see fig. 2-5)

\( \nu \)
arbitrarily chosen reference Poisson’s ratio

\( \varepsilon_{x}, \varepsilon_{y} \)
perturbation parameter (see eq. 1.7 to 1.10)

\( \varepsilon_{m} \)
vector of perturbation parameters (see eq. 6.5)

\( \rho_{c}^i \)
imperfection amplitude of \( \overrightarrow{W_m} \) (see eq. 1.11)

\( \phi_n \)
rotation about the shell normal (see eq. 2.9)

\( \psi \)
rotation about the \( x \)-axis (see fig. 1-2 and eq. 2.8)

\( ()^t \)
transpose

\( ()_{,x} \)
= \( \partial y / \partial x \)

\( ()_{,y} \)
= \( \partial y / \partial y \)

\( \| \hat{a} \| \)
square root of the inproduct of vector \( \hat{a} \)
INTRODUCTION

Computational package AMCYLAN consists of five separate computational modules. These modules are used to perform in several steps an asymptotic multimode analysis for (im-)perfect anisotropic cylindrical shells which may be stiffened in axial and/or circumferential direction. In the first step the prebuckling and buckling states are analyzed. Then in the second step a number of buckling modes are selected to investigate their mode interaction in the postbuckling state. In the third step a number of imperfection shapes are created for which imperfection form factors are calculated in the fourth step. Each of these imperfection shapes is multiplied with an imperfection amplitude. The sum of these products forms the initial imperfection of the shell. Finally, the imperfection sensitivity is investigated in the fifth step by computing the limit load for a given initial imperfection, i.e., for given imperfection amplitudes.

In chapter 1 the five computational modules of package AMCYLAN are introduced. This chapter also defines the notations used for the description of the cylinder problem and provides a stream diagram for using computational package AMCYLAN. The usage of each of its computational modules is described in the chapters 2 to 6.
1 COMPUTATIONAL PACKAGE AMCYLAN

Computational package AMCYLAN consists of 5 computational modules. These computational modules are:

1. ANOVSI : prebuckling and buckling analysis
2. POSTPR : postbuckling analysis and postbuckling coefficients computation
3. IMPSHP : imperfection input file creation module
4. IMPFAC : imperfection form factors computation
5. COHPAT : limit point calculation for a given imperfection

The input for these computational modules is discussed in the chapters 2 up to 6. The name AMCYLAN is a shortcut for Asymptotic Multi-mode CYLinder ANalysis. The purpose of chapter 1 is to introduce some notations and to provide a stream diagram for using computational package AMCYLAN.

1.1 Purpose

Asymptotic multi-mode analysis of (im-)perfect anisotropic cylindrical shells which may be stiffened in axial and/or circumferential direction. This analysis can be carried out using a Donnell, Sanders or Novozhilov type thin shell theory.

1.2 Problem notation

The package AMCYLAN uses a mixed formulation in which the unknowns are both forces and displacements. The definition for these unknowns is shown in

![Figure 1-1 Cylinder geometry and unknowns sign convention](image-url)
The unknowns are written dimensionless as follows

\[
\hat{P} = \frac{cR}{Et^2} P \quad ; \quad \hat{u} = \frac{1}{t} u
\]  

(1.1)

\[
\hat{S} = \frac{cR}{Et^2} S \quad ; \quad \hat{v} = \frac{1}{t} v
\]  

(1.2)

\[
\hat{Q} = \frac{4c^2R^3}{Et^4} Q \quad ; \quad \hat{w} = \frac{1}{t} w
\]  

(1.3)

\[
\hat{M} = \frac{4c^2R^2}{Et^4} M \quad ; \quad \hat{\beta} = \beta
\]  

(1.4)

in which \(E\) is the reference Young's modulus. The constant \(c\) is defined as

\[
c = \sqrt{3(1 - \nu^2)}
\]  

(1.5)

in which \(\nu\) is the reference poisson ratio. The initial imperfection \(\hat{w}\), measured from the undeformed perfect cylinder midsurface in the direction of the \(z\)-coordinate (see ref. [11]), is written dimensionless as follows

\[
\hat{\hat{w}} = \frac{1}{t} \bar{w}
\]  

(1.6)

In the asymptotic multi-mode analysis one begins with assuming a solution valid in the initial postbuckling region in the form of an asymptotic expansion

\[
\hat{P} = P_0 + \xi_i P_i + \xi_j \xi_j P_{ij} + \ldots \quad ; \quad \hat{u} = U_0 + \xi_i U_i + \xi_j \xi_j U_{ij} + \ldots
\]  

(1.7)

\[
\hat{S} = S_0 + \xi_i S_i + \xi_j \xi_j S_{ij} + \ldots \quad ; \quad \hat{v} = V_0 + \xi_i V_i + \xi_j \xi_j V_{ij} + \ldots
\]  

(1.8)

\[
\hat{Q} = Q_0 + \xi_i Q_i + \xi_j \xi_j Q_{ij} + \ldots \quad ; \quad \hat{w} = W_0 + \xi_i W_i + \xi_j \xi_j W_{ij} + \ldots
\]  

(1.9)

\[
\hat{M} = M_0 + \xi_i M_i + \xi_j \xi_j M_{ij} + \ldots \quad ; \quad \hat{\beta} = \beta_0 + \xi_i \beta_i + \xi_j \xi_j \beta_{ij} + \ldots
\]  

(1.10)

in which repeated lowercase indices imply summation. These indices range from \(I\) to \(N\) in which \(N\) indicates the number of modes used in the multi-mode analysis.

The imperfection \(\hat{\hat{w}}\) is written as

\[
\hat{\hat{w}} = \bar{\xi}_m \bar{W}_m \quad \text{with} : m = 1, \ldots, M
\]  

(1.11)

in which summation is implied over repeated lower case indices. The \(\bar{\xi}_m\) are the
imperfection amplitudes. The $\bar{W}_m$ are the imperfection shape functions. The total number of imperfection shape functions is indicated by $M$.

The loading of the cylinder is given in figure 1-2. The constant dead loads $\bar{P}$, $\bar{S}$, $\bar{Q} + \Delta \bar{Q}$ and $\bar{M} + \Delta \bar{M}$ applied at $x=0$ and the constant dead loads $\bar{P}$, $\bar{S}$, $\bar{Q}$ and $\bar{M}$ applied at $x=L$ are line loads per unit length in $y$-direction. Further, the cylinder can be loaded by two types of constant external pressure, namely a dead load $p_e$ and a follower load $p_{fe}$ per unit of area. The dead loads retain their work line from the undeformed state during deformation. The follower load $p_{fe}$ behaves quite differently since it remains normal to the cylinder surface during deformation, i.e., it follows the deformation of the cylinder. All these loads except follower load $p_{fe}$ are conservative and hence possess a potential. The load $p_{fe}$ is conservative if

at $x=0 : w + \bar{w} = 0$ or $u =$ prescribed
at $x=L : w + \bar{w} = 0$ or $u =$ prescribed

The load $p_{fe}$ cannot be used if these restrictions are not met. In other words, it can be taken into account only if it possesses a potential. Notice that the loading of the cylinder is such that it is in static equilibrium.

The cylinder can, apart from the line and surface loads given in figure 1-2, also be loaded by prescribed displacements at the cylinder edges $x=0$ and $x=L$. the loading is taken proportional to a load factor $\Lambda$ as follows

$$\hat{F} = \hat{F}_c + \Lambda \hat{F}^* \quad (\text{prescribed forces})$$

$$\hat{D} = \hat{D}_c + \Lambda \hat{D}^* \quad (\text{prescribed displacements})$$

where
\[ \hat{F}' = \begin{bmatrix} \tilde{P} & \tilde{S} & \tilde{Q} & \Delta \tilde{Q} & \tilde{M} & \Delta \tilde{M} & p_e & p_{pe} \end{bmatrix} \]

\[ \hat{D}' = \begin{bmatrix} \tilde{u}_0 & \tilde{v}_0 & \tilde{w}_0 & \beta_0 & \tilde{u}_L & \tilde{v}_L & \tilde{w}_L & \beta_L \end{bmatrix} \]

in which a superscript \( t \) denotes the transpose of a vector. The prescribed displacements at \( x=0 \) are represented by \( \tilde{u}_0, \tilde{v}_0, \tilde{w}_0, \beta_0 \) and the prescribed displacements at \( x=L \) are \( \tilde{u}_L, \tilde{v}_L, \tilde{w}_L, \beta_L \). Notice that the loading is not given in dimensionless form.

The dependance of the load factor \( \Lambda \) on the perturbation parameters \( \xi_i \) and the imperfection amplitudes \( \tilde{\xi}_m \) in the vicinity of bifurcation points having (nearly) simultaneous bifurcation loads \( \Lambda^\prime \) is described by (\( i, j, k, L = 1, \ldots, N ; m = 1, \ldots, M \))

\[
(\Lambda - \Lambda_i \xi_j \xi_k \delta_m \delta_m L \xi_j \xi_k) = \Lambda_i \tilde{a}_{ij} \tilde{L} \xi_j \xi_k + \Lambda_i \tilde{b}_{ij} \tilde{L} \xi_j \xi_k - \Lambda_i \tilde{e}_{ij} \tilde{L} \xi_j \xi_k - (\Lambda - \Lambda_i) \tilde{b}_{i} \tilde{L} \xi_j \xi_k \xi_m \xi_m \xi_m
\] (1.14)

in which \( \Lambda_i \) indicates the load level at which the prebuckling state is evaluated. Repeated lowercase indices imply summation, but repeated uppercase indices do not. The equations (1.14) are a set of \( N \) nonlinear simultaneous equations in the \( N+1 \) unknowns \( \xi_1, \ldots, \xi_N, \Lambda \).

After substitution of the perturbation expansions (1.7) to (1.10) into the governing equations describing the equilibrium of the perfect shell one obtains governing equation sets of the \( 0^{th} \), \( 1^{th} \) and \( 2^{th} \) order states by grouping all terms in powers of \( \xi_i \).

To solve the governing equations of the \( 0^{th} \) order state (prebuckling state) an axisymmetric prebuckling solution is introduced. This is possible since the loading and boundary conditions of the cylinder are axisymmetric. The axisymmetric prebuckling solution is given by

\[
P_0 = -\lambda + p_0(x) \quad ; \quad U_0 = u_0(x)
\]

\[
S_0 = \tilde{s} + s_0(x) \quad ; \quad V_0 = v_0(x)
\]

\[
Q_0 = \tilde{q} + q_0(x) \quad ; \quad W_0 = w_0(x)
\]

\[
M_0 = \tilde{m} + m_0(x) \quad ; \quad \beta_0 = \beta_0(x)
\]

where

\[
\lambda = -\frac{cR}{E \ell^2} \tilde{P} \quad ; \quad \tilde{s} = \frac{cR}{E \ell^2} \tilde{S} \quad ; \quad \tilde{q} = \frac{4c^2 R^3}{E \ell^4} \tilde{Q} \quad ; \quad \tilde{m} = \frac{4c^2 R^2}{E \ell^4} \tilde{M}
\]
To solve the governing equations of the 1\textsuperscript{st} order state (buckling state) the following fourier series are used for the buckling unknowns

\[
P_1 = p_{11}(x) \cos(n_1y/R) + p_{21}(x) \sin(n_2y/R) \quad (1.19)
\]

\[
S_1 = s_{11}(x) \sin(n_1y/R) + s_{21}(x) \cos(n_2y/R) \quad (1.20)
\]

\[
Q_1 = q_{11}(x) \cos(n_1y/R) + q_{21}(x) \sin(n_2y/R) \quad (1.21)
\]

\[
M_1 = m_{11}(x) \cos(n_1y/R) + m_{21}(x) \sin(n_2y/R) \quad (1.22)
\]

\[
U_1 = u_{11}(x) \cos(n_1y/R) + u_{21}(x) \sin(n_2y/R) \quad (1.23)
\]

\[
V_1 = v_{11}(x) \sin(n_1y/R) + v_{21}(x) \cos(n_2y/R) \quad (1.24)
\]

\[
W_1 = w_{11}(x) \cos(n_1y/R) + w_{21}(x) \sin(n_2y/R) \quad (1.25)
\]

\[
\beta_1 = \beta_{11}(x) \cos(n_1y/R) + \beta_{21}(x) \sin(n_2y/R) \quad (1.26)
\]

in which no summation is implied over repeated capital indices.

To solve the governing equations of the 2\textsuperscript{nd} order state (postbuckling state) the following fourier series are used for the postbuckling unknowns

\[
P_{ij} = p_{11j}(x) \cos((n_1-n_j)y/R) + p_{21j}(x) \sin((n_1-n_j)y/R) + p_{31j}(x) \cos((n_1+n_j)y/R) + p_{41j}(x) \sin((n_1+n_j)y/R) \quad (1.27)
\]

\[
S_{ij} = s_{11j}(x) \sin((n_1-n_j)y/R) + s_{21j}(x) \cos((n_1-n_j)y/R) + s_{31j}(x) \sin((n_1+n_j)y/R) + s_{41j}(x) \cos((n_1+n_j)y/R) \quad (1.28)
\]

\[
Q_{ij} = q_{11j}(x) \cos((n_1-n_j)y/R) + q_{21j}(x) \sin((n_1-n_j)y/R) + q_{31j}(x) \cos((n_1+n_j)y/R) + q_{41j}(x) \sin((n_1+n_j)y/R) \quad (1.29)
\]

\[
M_{ij} = m_{11j}(x) \cos((n_1-n_j)y/R) + m_{21j}(x) \sin((n_1-n_j)y/R) + m_{31j}(x) \cos((n_1+n_j)y/R) + m_{41j}(x) \sin((n_1+n_j)y/R) \quad (1.30)
\]

\[
U_{ij} = u_{11j}(x) \cos((n_1-n_j)y/R) + u_{21j}(x) \sin((n_1-n_j)y/R) + u_{31j}(x) \cos((n_1+n_j)y/R) + u_{41j}(x) \sin((n_1+n_j)y/R) \quad (1.31)
\]

\[
V_{ij} = v_{11j}(x) \sin((n_1-n_j)y/R) + v_{21j}(x) \cos((n_1-n_j)y/R) + v_{31j}(x) \sin((n_1+n_j)y/R) + v_{41j}(x) \cos((n_1+n_j)y/R) \quad (1.32)
\]
\[ W_{ij} = w_{1ij}(x) \cos((n_i-n_j)y/R) + w_{2ij}(x) \sin((n_i-n_j)y/R) + \\
 w_{3ij}(x) \cos((n_i+n_j)y/R) + w_{4ij}(x) \sin((n_i+n_j)y/R) \]  
\[ \beta_{ij} = \beta_{1ij}(x) \cos((n_i-n_j)y/R) + \beta_{2ij}(x) \sin((n_i-n_j)y/R) + \\
 \beta_{3ij}(x) \cos((n_i+n_j)y/R) + \beta_{4ij}(x) \sin((n_i+n_j)y/R) \] 

in which no summation is implied over repeated capital indices.

Finally, the dimensionless form of the surface loads is presented. The surface loads \( P_e \) and \( P_{fe} \) are written dimensionless as follows

\[ P_e = \frac{cR^2}{Et^2} P_e \quad ; \quad P_{fe} = \frac{cR^2}{Et^2} P_{fe} \]  

1.3 Computational stream diagram of package AMCYLAN

The computational stream diagram in figure 1-3 shows the input and output streams of the individual programs in computational package AMCYLAN. Its computational modules ANOVSI, POSTPR, IMPSHA, IMPFAC and COHPAT are represented by 5 rectangles. The input files needed by each module and the output files produced by each module are represented by incoming and outgoing arrows.

Each module has a so called standard input file. A superscript * is used to identify these files in figure 1-3. The name of such a file is compound of its modules name followed by the extension .INP. The names of the standard input files are fixed and cannot be changed.

The text strings that go with the incoming and outgoing arrows of one module other than the incoming arrow from its standard input file can be found as input fields in its standard input file. Therefore, the file names expected for these text strings are given in the standard input files. These files take care of the input and output streams of the modules.

All of the computational modules except ANOVSI take at least one input stream that is an output file from one of the modules. This is represented by lines connecting computational modules. These lines consist of arrows and dots. The text string that goes with an arrow pointing towards a dot can be found as input field in the standard input file of the module from which that arrow is pointing. The text string that goes with an arrow pointing away from a dot can be found as input field in the standard input file of the module to which that arrow is pointing.
The computational module ANOVSI solves the prebuckling and buckling states of the perfect shell for the unknowns listed in the equations (1.15) to (1.26). The input files needed to run computational module ANOVSI are described in chapter 2.

Figure 1-3 Computational stream diagram of package AMCYLAN

Computational module POSTPR solves the postbuckling state of the perfect
computational package AMCYLAN

shell for the unknowns listed in the eq. (1.27) to (1.34). It also computes the postbuckling coefficients $a_{ij}^L$ and $b_{ijk}^L$ needed in the eq. (1.14). The input files needed to run computational module POSTPR are described in chapter 3.

Computational module IMPSHP is used for setting up an imperfection input file for program IMPFAC. This file contains the $M$ imperfection shapes $\overline{W}_m$ in eq. (1.11). The input files needed to run computational module IMPSHP are described in chapter 4.

Computational module IMPFAC is used for computing the imperfection form factors $\alpha_m^L$ and $\beta_m^L$ needed in the eq. (1.14). The input files needed to run computational module IMPFAC are described in chapter 5.

The last computational module COHPAT is used for solving the eq. (1.14). The standard input file of this module has an input field with the label bifurcation loads, postbuckling coefficients & imperfection form factors file or restart file. This input file can be given the name of one of two different file types. In figure 1-3 this is indicated by the open circle with the word or next to it. Notice that the label part bifurcation loads, postbuckling coefficients & imperfection form factors file is abbreviated to bif. loads, post. coeff. & imp. form factors file in the figure. The input files needed to run computational module COHPAT are described in chapter 6.
2 COMPUTATIONAL MODULE ANOVSI

The computational module ANOVSI is designed as a tool to analyse the prebuckling and buckling states of an anisotropic (layered composite) cylindrical shell which may be stiffened in axial and/or circumferential direction.

The prebuckling state is axisymmetric which implies that only axisymmetric loading cases can be handled. This state can be either membrane or rigorous prebuckling.

The buckling problem is solved using subspace iteration. The computational module ANOVSI is therefore capable of computing a number of buckling modes with the smallest bifurcation loads per $n_1$ full waves in circumferential direction.

Computational module ANOVSI needs input from two files. These are:

1. **standard input file** ANOVSI.INP
2. **data input file** (also called: problem definition file)

The **standard input file** always has the name ANOVSI.INP. The name for the **data input file** can be chosen by the user. The example listing for the **data input file** in appendix A.2 is named KHORT_11_13.INP. An example listing for the **standard input file** ANOVSI.INP is given in appendix A.1. These listings show that the two input files contain comments describing all input fields. Notice that the last line of a comment block contains the sign $*$ in its first column indicating that input fields follow directly after that line. The variable set by the input field is always situated above the input field itself. Using these listings as a guideline a description of the two input files needed by computational module ANOVSI is given in the sections 2.1 and 2.2.

2.1 Standard input file: ANOVSI.INP

An example listing of ANOVSI's **standard input file** is given in appendix A.1. This file handles the input and output streams of ANOVSI.

**Input fields: IHIST, history output file**

The program produces calculation process information for two reasons. The first one is to indicate how much of the computation is already performed. Secondly to give error messages in case an error occurred. This information can be suppressed, written to the screen or written to a so called **history output file** by setting the flag IHIST (see the comment of the example listing for setting IHIST). If
the calculation process information is not written to the history output file one may leave the input field history output file empty, otherwise it must contain the name for the history output file. It is not allowed to leave one of the other input fields empty.

**Input field : data input file**

The input field for the data input file specifies the name of a problem definition file such as the one listed in appendix A.2. The cylinder problem under consideration is defined in the data input file.

**Input field : result output file**

The results of the computation are written to the file which name is given in the result output file input field.

2.2 Data input file (problem definition file)

An example listing of a data input file is given in appendix A.2. The input fields of the data input file are grouped in a number of subjects. Each of these subject groups is discussed in a separate section. Not all the input fields are discussed in these sections but only those for which additional explanation on top of the comments in the example data input file is helpful. Sufficient description of not discussed input fields can be found in the comments of the example data input file.

2.2.1 Group 0 : SUBJECT

The input given for the two input fields in this group is written unchanged to the result output file.

**Input field : SUBJEC**

Input field SUBJEC takes a string as input which can be used to give a short description of the problem under consideration.

**Input field : CRNAME**

The input field CRNAME is used to give the name of the person who created the data input file.

2.2.2 Group 1 : SHELL DATA

This group defines the geometry of the cylinder, the properties of the shell wall and whether the shell wall is stiffened. The shell wall of the cylindrical shell shown
in figure 2-1 is a layered anisotropic shell with \( K \) individual layers. The numbering of the layers starts at the outer surface as indicated in the figure. Notice that the angle of rotation \( \theta_k \) \((k = 1, ..., K)\) of the individual layers is defined with respect to the \( x \)-axis of the shell. The shell reference surface coincides with the midsurface of

\[ L = 1, 2 = \text{orthogonal reference frame of orthotropic lamina } k \]

Figure 2-1  Anisotropic cylindrical shell with ring and stringer stiffening

laminate. The cylindrical shell can be stiffened in axial and/or circumferential direction. The stiffening of the shell is shown for the dashed shell element in the backwall of the cylinder, which is drawn enlarged to the right of the cylinder in figure 2-1.

**Input fields :** \( L, R, t \)

The dimensions of the cylinder \((L, R, t)\) are specified in section 1.1 of the data input file.

**Input fields :** \( NMAT, NLAM \)

Section 1.2 of the data input file defines the properties of the shell wall. The number of materials is given in input field \( NMAT \). The number of laminae (layers) is given in input field \( NLAM \).

**Input fields :** \( mn, E11, E22, nu12, G12 \)

The properties of the materials \( 1 \) to \( NMAT \) are given in the input table in section 1.2.1 of the data input file. The number of materials specified in this table must equal the specified number of materials \( NMAT \). Each new material must be specified on a new line. Each material must have its own unique material number \( mn \)

12
which value must be chosen such that: \( 1 \leq mn \leq NMAT \). Empty lines are not allowed in the table.

**Input fields: ln, RH, TETA, lmat**

The input table in section 1.2.2 of the *data input file* defines the layers of the shell wall and must contain exactly \( NLAM \) lines. Each line specifies one layer. The first line defines layer 1 in figure 2-1. The last line (line \( NLAM \)) defines layer \( K \) in figure 2-1. Each layer must have its own unique layer number \( ln \) which value must be chosen such that: \( 1 \leq ln \leq NLAM \). Empty lines are not allowed in the table. The thickness of each layer \( RH \) is specified as a relative thickness. The actual thickness of a layer is then computed by dividing its value for \( RH \) by the total relative thickness and multiplying the result with the thickness of the shell wall \( t \). The total relative thickness is the sum of the entries \( RH \) for each layer. The entries \( TETA \) are the angles \( \theta_k \) in figure 2-1 for each layer. The entry \( lmat \) specifies the layer material and it must be chosen from the material numbers \( mn \) in the material table in section 1.2.1 of the *data input file*.

**Input fields: ISTRIN, Es, Gs, Ds, Cs, As, Is, Js, IRING, Er, Gr, Dr, Cr, Ar, Ir, Jr**

Section 1.3 of the *data input file* defines the stiffening of the shell wall. The stiffeners are treated as "smeared" stiffeners by the program (See ref. \([1]^{[1]}\)). The axial stiffening with stringers is specified in section 1.3.1 of the *data input file*. The flag \( ISTRIN \) is used for turning the axial stiffening on or off. The circumferential stiffening with rings is specified in section 1.3.2 of the *data input file*. The flag \( IRING \) is used for turning the circumferential stiffening on or off. The properties of the stringers and rings are always read in by ANOVS1 but used only in case the corresponding stiffening is turned on using one of the flags \( ISTRIN \) or \( IRING \). The moment of inertia \( Is \) is the moment of inertia of a stringer with respect to a line parallel to the y-axis through the centroid of the stringer (see figure 2-1). The moment of inertia \( Jr \) is the moment of inertia of a ring with respect to a line parallel to the x-axis through the centroid of the ring (see figure 2-1). The definition of the spacing (\( Ds, Dr \)) and eccentricity (\( Cs, Cr \)) of the stiffeners is illustrated in figure 2-1. The full description of the stiffener input data can be found in section 1.3 of the example *data input file*. 
2.2.3 Group 2: FINITE DIFFERENCE

**Input field: NIH**

Computational module ANOVSI solves a set of differential equations for each of the prebuckling and buckling states using a finite difference formulation with constant stepsize h. The number of finite difference intervals used is always

\[
\begin{array}{cccc}
0 & 1 & 2 & NIH \\
\frac{1}{2}L & L & 2*NIH
\end{array}
\]

Figure 2-2 Subdivision of the x-axis in finite difference intervals

specified by giving the number of finite difference intervals used for one half of the shell in input field NIH even if the whole shell is used for the computation.

2.2.4 Group 3: DATA FOR DIMENSIONLESS WRITING

**Input fields: DNU, ESTAR**

The reference poisson ratio \( \nu \) is given in input field DNU and is used to compute the constant \( c \) from: \( c = \sqrt{3} (1 - \nu^2) \). The reference Young's modulus \( E \) is given in input field ESTAR.

2.2.5 Group 4 & 5: SHELL THEORY CHOICE

**Input field: ITHERY**

This group has only one input field ITHERY. This flag is used to specify the shell theory to be used for the computation. It essentially determines the definition of the strain displacement relations used in the computation. Let the strain displacement relations be given as (See ref. [11])

\[
\begin{align*}
\varepsilon_x &= u_x + \frac{1}{2} \beta^2 + \frac{1}{2} \phi_n - \beta w_x \\
\varepsilon_y &= v_y - \frac{w}{R} + \frac{1}{2} \psi^2 + \frac{1}{2} \phi_n - \psi w_y \\
\gamma_{xy} &= u_y + v_x + \beta \psi - \beta w_y - \psi w_x \\
\kappa_x &= \beta_y \\
\end{align*}
\]

(2.1)  \hspace{2cm} (2.2)  \hspace{2cm} (2.3)  \hspace{2cm} (2.4)
\[ \kappa_y = \psi_y \]  
\[ \kappa_{xy} = \beta_y + \psi_x - \frac{1}{R} \phi_n - \frac{1}{2R} (v_x + u_y) \]  
(2.5)  
(2.6)

The rotations \( \beta, \psi \) and \( \phi_n \) are defined as follows

\[ \beta = -w_x \]  
(see fig. 1-2)  
(2.7)

\[ \psi = -w_y - \frac{1}{R} \nu \]  
(see fig. 1-2)  
(2.8)

\[ \phi_n = \frac{1}{2} (v_x - u_y) \]  
(normal rotation about z-axis)  
(2.9)

The strain displacement relations given here are derived using the assumption of small strains and moderately small rotations, i.e., the strains \( \varepsilon_x, \varepsilon_y, \gamma_{xy} \) are of order \( O(\varepsilon^2) \) while the rotations \( \beta, \psi, \phi_n \) are of order \( O(\varepsilon) \). Using these assumptions, it is possible to rewrite the rotation \( \phi_n \) as follows

\[ \phi_n = \frac{1}{2} (-2u_y + v_x + u_y) = -u_y + \frac{1}{2} \gamma_{xy} = -u_y + O(\varepsilon^2) \equiv -u_y \]  
(2.10)

with no loss in accuracy since \( \phi_n \) is assumed to be \( O(\varepsilon) \). The notation \( \gamma_{xy} \) is used to indicate the shear strain from linear theory. The form (2.10) for rotation \( \phi_n \) was introduced by Cohen [2].

Setting \( \text{ITHERY} \) equal to 1 leaves out rotation \( \phi_n \) from \( \varepsilon_x, \varepsilon_y \) and \( \kappa_{xy} \), leaves out the single underlined term from rotation \( \psi \) and leaves out the double underlined term from \( \kappa_{xy} \). This yields the so-called Donnell-Mushtari-Vlasov strain displacement relations.

Setting \( \text{ITHERY} \) equal to 2 leaves out rotation \( \phi_n \) from \( \varepsilon_x, \varepsilon_y \) and \( \kappa_{xy} \), and leaves out the double underlined term from \( \kappa_{xy} \). This yields Sanders strain displacement relations excluding the normal rotation \( \phi_n \).

Setting \( \text{ITHERY} \) equal to 3 leaves out rotation \( \phi_n \) from \( \varepsilon_x, \varepsilon_y \) and \( \kappa_{xy} \). This yields Novozhilov's strain displacement relations excluding the normal rotation \( \phi_n \).

Setting \( \text{ITHERY} \) equal to 4 leaves out rotation \( \phi_n \) from \( \varepsilon_x \) and \( \varepsilon_y \), and leaves out the double underlined term from \( \kappa_{xy} \). In this case, rotation \( \phi_n \) is included in twisting curvature \( \kappa_{xy} \) using its definition in eq. (2.9).

Setting \( \text{ITHERY} \) equal to 5 leaves out rotation \( \phi_n \) from \( \varepsilon_x \) and \( \varepsilon_y \). In this case, rotation \( \phi_n \) is included in twisting curvature \( \kappa_{xy} \) using its definition in eq. (2.9).

Setting \( \text{ITHERY} \) equal to 6 will include all terms except the double underlined term in \( \kappa_{xy} \) yielding Sanders strain displacement relations. Expression (2.9) is used
for $\phi_n$ in twisting curvature $\kappa_{xy}$. Expression (2.10) is used for $\phi_n$ in the normal strains $\varepsilon_x$ and $\varepsilon_y$.

Setting $ITHERY$ equal to 7 will include all terms yielding Novozhilov’s strain displacement relations. Expression (2.9) is used for $\phi_n$ in twisting curvature $\kappa_{xy}$. Expression (2.10) is used for $\phi_n$ in the normal strains $\varepsilon_x$ and $\varepsilon_y$.

**Warning**: Setting $ITHERY$ equal to 6 or 7 can lead to erroneous results due to using approximation (2.10) for $\phi_n$ in the normal strains $\varepsilon_x$ and $\varepsilon_y$. This can happen if the order of magnitude of the shear strain $\gamma_{xy}$ from linear theory is bigger than the order of magnitude of the normal strains $\varepsilon_x$ and $\varepsilon_y$. Such a situation is for example possible when the shell is heavily stiffened with both stringers and rings.

### 2.2.6 Group 4: PREBUCKLING PROBLEM

This group defines the loading and the boundary conditions for the prebuckling state.

**Input field**: $INONL$

The input read from group 4 depends on flag $INONL$. If set to zero the program will use a membrane prebuckling state as defined under point a) in group 4 of the *data input file*. The input under point b) in this group will be ignored for $INONL=0$ and the reading process is resumed at the start of input group 5. If flag $INONL$ is set to a nonzero value the program will use a nonlinear prebuckling state as defined under point b) in group 4 of the *data input file*. For a nonzero value of $INONL$ the program ignores the input under point a) in this group and the reading process is resumed at point b) in group 4.

### 2.2.6.a MEMBRANE PREBUCKLING

The membrane prebuckling input fields are read in only if flag $INONL$ is set equal to 0.

**Input fields**: $ISYM0, AU, AV$

The part of the shell used in the computation must be set by specifying flag $ISYM0$. If this flag is set equal to 0 then the whole shell is used for the computation in which case the program needs to know where the axial and circumferential displacement components $U_0$ and $V_0$ should be set equal to 0. This is specified by
computational module ANOVS

the values given for the input fields AU and AV. The values given for the input fields AU and AV are read in but not used if the half shell is used for the computation (ISYM0 is 1 or 2) for which case the axial and circumferential displacement components \( U_0 \) and \( V_0 \) are set equal to 0 at the midpoint \( x=0.5L \) of the shell. Details on setting the flags ISYM0, AU and AV can be found in the comments under point a) in group 4 of the example data input file.

Input fields: \( \lambda \text{mbdaC, stildeC, peC, lambdaSTAR, stildeSTAR, peSTAR} \)

The remaining input fields for the membrane prebuckling state specify the loading conditions as follows (see also eq. (1.15), (1.16) and (1.35))

\[
P_0 = - (\lambda \text{mbdaC + } \Lambda \text{*lambdaSTAR} ) \quad \text{(axial load)} \tag{2.11}
\]

\[
S_0 = \text{stildeC} + \Lambda \text{*stildeSTAR} \quad \text{(torsional load)} \tag{2.12}
\]

\[
p_e = \text{peC} + \Lambda \text{*peSTAR} \quad \text{(external pressure)} \tag{2.13}
\]

in which \( \Lambda \) represents the generalized load factor. The eq. (2.11) to (2.13) show that the loading consists of a constant part and a part that is proportional to load factor \( \Lambda \).

2.2.6.b NONLINEAR PREBUCKLING

The nonlinear prebuckling input fields are read in only if flag INONL is set to a nonzero value.

Input fields: \( ISYM0,\)

\[ \text{IBC00(1), IBC00(2), IBC00(3), IBC00(4),}\]

\[ \text{IBCL0(1), IBCL0(2), IBCL0(3), IBCL0(4)} \]

The input fields in section 4.1 of the data input file set the boundary conditions (B.C.) for the prebuckling state. The flag ISYM0 sets the part of the shell used for the computation. The whole shell is used if it is set equal to zero in which case boundary

<table>
<thead>
<tr>
<th>( i )</th>
<th>( \text{IBC00}(i) = 1 )</th>
<th>( \text{IBC00}(i) \neq 1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( P_0(0) = Pa(0) + \Lambda Pb(0) )</td>
<td>( U_0(0) = Ua(0) + \Lambda Ub(0) )</td>
</tr>
<tr>
<td>2</td>
<td>( S_0(0) = Sa(0) + \Lambda Sb(0) )</td>
<td>( V_0(0) = Va(0) + \Lambda Vb(0) )</td>
</tr>
<tr>
<td>3</td>
<td>( Q_0(0) = Qa(0) + \Lambda Qb(0) )</td>
<td>( W_0(0) = Wa(0) + \Lambda Wb(0) )</td>
</tr>
<tr>
<td>4</td>
<td>( M_0(0) = Ma(0) + \Lambda Mb(0) )</td>
<td>( \beta_0(0) = \beta a(0) + \Lambda \beta b(0) )</td>
</tr>
</tbody>
</table>

Table 1: Prebuckling boundary condition settings at \( x=0 \)
conditions are needed at both \( x=0 \) and \( x=L \). The boundary conditions at \( x=0 \) are set by the flags \( IBC00(i) \) \((i=1,...,4)\) as shown in table 1. The boundary conditions at \( x=L \) are set by the flags \( IBCLO(i) \) \((i=1,...,4)\) as shown in table 2. The part of the shell

<table>
<thead>
<tr>
<th>( i )</th>
<th>( IBCLO(i) = 1 )</th>
<th>( IBCLO(i) \neq 1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( P_0(L) = Pa(L) + \Lambda Pb(L) )</td>
<td>( U_0(L) = Ua(L) + \Lambda Ub(L) )</td>
</tr>
<tr>
<td>2</td>
<td>( S_0(L) = Sa(L) + \Lambda Sh(L) )</td>
<td>( V_0(L) = Va(L) + \Lambda Vb(L) )</td>
</tr>
<tr>
<td>3</td>
<td>( Q_0(L) = Qa(L) + \Lambda Qb(L) )</td>
<td>( W_0(L) = Wa(L) + \Lambda Wb(L) )</td>
</tr>
<tr>
<td>4</td>
<td>( M_0(L) = Ma(L) + \Lambda Mb(L) )</td>
<td>( \beta_0(L) = \beta a(L) + \Lambda \beta b(L) )</td>
</tr>
</tbody>
</table>

Table 2: Prebuckling boundary condition settings at \( x=L \)

between \( x=0 \) and \( x=0.5L \) is used if flag \( ISYM0 \) is set equal to 1 in which case the user must provide boundary conditions only at \( x=0 \). The flags \( IBC00(i) \) and \( IBCLO(i) \) are read in for \( ISYM0=1 \) but only the values for \( IBC00(i) \) are used. The part of the shell between \( x=0.5L \) and \( x=L \) is used if flag \( ISYM0 \) is set equal to 2 in which case the user must provide boundary conditions only at \( x=L \). The flags \( IBC00(i) \) and \( IBCLO(i) \) are read in for \( ISYM0=2 \) but only the values for \( IBCLO(i) \) are used. Using one half of the shell by setting \( ISYM0 \) equal to 1 or 2 automatically implies that the anti-symmetry conditions: \( U_0 = V_0 = Q_0 = \beta_0 = 0 \), are used as boundary conditions at \( x=0.5L \). These anti-symmetry conditions make \( W_0 \) symmetric with respect to \( x=0.5L \). The unknowns that can be set by the boundary condition flags are illustrated in figure 2-3.

**Input fields:** \( Pa(0), Sa(0), Qa(0), Ma(0), Pb(0), Sb(0), Qb(0), Mb(0), Pa(L), Sa(L), Qa(L), Ma(L), Pb(L), Sb(L), Qb(L), Mb(L), Ua(0), Va(0), Wa(0), BERTAa(0), Ub(0), Vb(0), Wb(0), BERTAb(0), Ua(L), Va(L), Wa(L), BERTAa(L), Ub(L), Vb(L), Wb(L), BERTAb(L) \)

After selecting the unknowns prescribed in the boundary conditions it is needed to specify their prescribed values. This is done by setting the input fields in section 4.2 of the data input file. Although all the input fields in section 4.2 of the data input file are read in always only a number of them are actually used by the program. The basic strategy for setting the prescribed values for the boundary

18
conditions is to select for each boundary condition flag $IBC00(i)$ and $IBCL0(i)$ that is actually used by the program the corresponding input fields in section 4.2 of the data input file and setting them to the desired values. If for example flag $IBC00(1)$ is used by the program and is set equal to 1 then axial load $P_0(0)$ is prescribed and the user must specify the values for the input fields $Pa(0)$ and $Pb(0)$ setting:

$$P_0(0) = Pa(0) + \Lambda \cdot Pb(0)$$ (tension positive). The input fields $Ua(0)$ and $Ub(0)$ are in such a case read in but not used by the program. If for example flag $IBC00(1)$ is used by the program and not set equal to 1 then axial displacement $U_0(0)$ is prescribed and the user must specify the values for the input fields $Ua(0)$ and $Ub(0)$ setting:

$$U_0(0) = Ua(0) + \Lambda \cdot Ub(0)$$ . The input fields $Pa(0)$ and $Pb(0)$ are in such a case read in but not used by the program. Notice that the prescribed value for a boundary condition always contains a constant part and a part that is proportional to the load factor $\Lambda$ (GLF in the data input file).

The input fields $Pa(L), Sa(L), Qa(L), Ma(L), Pb(L), Sb(L), Qb(L), Mb(L), Ua(L), Va(L), Wa(L), BERTAa(L), Ub(L), Vb(L), Wb(L)$ and $BERTAb(L)$ are read in but not used if $ISYM0=1$.

The input fields $Pa(0), Sa(0), Qa(0), Ma(0), Pb(0), Sb(0), Qb(0), Mb(0), Ua(0), Va(0), Wa(0), BERTAa(0), Ub(0), Vb(0), Wb(0)$ and $BERTAb(0)$ are read in but not used if $ISYM0=2$.

Setting the boundary conditions can be summarized as follows

1) specify the boundary condition flags in section 4.1 of the data input file
2) specify in section 4.2 of the data input file the input fields for:
   - line loads at $x=0$
   - line loads at $x=L$
   - displacements at $x=0$
   - displacements at $x=L$

that correspond to the settings of the flags $IBC00(i)$ and $IBCL0(i)$ for each
of these flags that is actually used by the program.

It is emphasized here that although not all input fields in section 4.2 of the data input file are used by the program to set boundary conditions they all must have values since they are all read in.

**Input fields :** \( pe_a, pfe_a \)
\( pe_b, pfe_b \)

Apart from loads introduced by setting boundary conditions it also possible to use surface loads. The surface load \( pe \_ \) in figure 2-3 represents external dead load pressure and is set by the input fields \( pe_a \) and \( pe_b \) whereby \( pe \_ \) becomes equal to \( pe_a = pe_a \Lambda pe_b \). The surface load \( pfe \_ \) in figure 2-3 represents follower external pressure and is set by the input fields \( pfe_a \) and \( pfe_b \) whereby \( pfe \_ \) becomes equal to \( pfe_a = pfe_a \Lambda pfe_b \). The surface loads \( pe \_ \) and \( pfe \_ \) are not allowed to be nonzero at the same time. Computational module ANOVI terminates the calculation if they are nonzero at the same time. Using a follower external pressure load \( pfe \_ \) in the loading conditions is only allowed if:

- at \( x=0 \): \( u(0) = \text{prescribed} \) or \( w(0) = 0 \)
- at \( x=L \): \( u(L) = \text{prescribed} \) or \( w(L) = 0 \)

Computational module ANOVI terminates the calculation if these conditions do not hold while including a follower external pressure load \( pfe \_ \) in the loading conditions.

**Input field :** \( GLFS \)

Input field \( GLFS \) is used to give the starting value for the load factor \( \Lambda \). An important point concerning this load factor is not yet mentioned. The computation of the bifurcation loads is programmed such that only positive values can be obtained for the bifurcation loads \( \Lambda_L \). This is not a serious drawback since it is always possible to ensure that the interesting bifurcation loads are positive by manipulating the sign of the prescribed loads.

**RIGID BODY MOTIONS :** When using the whole shell for the computation by setting \( ISYM0 \) equal to 0 one needs to suppress rigid body motions in the axial and circumferential directions. To suppress the rigid body motion in axial direction one needs to set at least one of the flags \( IBC00(1) \) and \( IBCLO(1) \) not equal to 1. To suppress the rigid body motion in circumferential direction
one needs to set at least one of the flags \textit{IBC00(2)}
and \textit{IBCLO(2)} not equal to 1. Program ANOVSI
terminates as soon as a rigid body motion is detected.

\textbf{ANILISA SIMULATION} : To obtain the same results as the program ANILISA
\cite{3} one needs to set the boundary condition flags such
that the prebuckling axial and torsional loads $P_0$ and
$S_0$ are prescribed at the boundaries $x=0$ and $x=L$.
This is needed since program ANILISA cannot
prescribe the axial and circumferential displacement
components $U_0$ and $V_0$ at these boundaries in the
prebuckling state but only the corresponding loads
$P_0$ and $S_0$.

\textbf{NOTATION FOR $\beta$} : The notation BERTA is used for rotation $\beta$ in the
\textit{data input file}.

\textbf{Prebuckling computational strategy}

The nonlinear prebuckling input fields in section 4.3 of the \textit{data input file}
direct the prebuckling calculation process. This calculation process is nonlinear in
the unknowns and is solved using a path following method. To calculate the
prebuckling solution at load level $\Lambda_{e}^{i+1}$ from the known prebuckling solution at load
level $\Lambda_{e}^{i}$ (see fig. 2-4) the program makes a prediction for the solution at load level
$\Lambda_{e}^{i+1}$ in one step from load level $\Lambda_{e}^{i}$ and then applies a Newton-iteration as
corrector. If this step is too big for obtaining a converging Newton-iteration the
program reduces the step size and performs a number of predictor-corrector steps
starting from load level $\Lambda_{e}^{i}$ in order to find the prebuckling solution at load level
$\Lambda_{e}^{i+1}$.

\textbf{Input fields : IBORD, IASFCZ}

The flag \textit{IBORD} is used for selecting the definition for the path parameter. The
flag \textit{IASFCZ} selects a modified or a standard Newton-iteration. After making a
prediction for a solution a Newton-iteration process is used to correct the prediction
until a converged solution is found. In a modified Newton-iteration the system
matrix is constructed and factorized only at the prediction and not at the successive
corrected solutions that lead up to the converged solution. In a standard Newton-
iteration the system matrix is constructed and factorized not only at the prediction but also at each corrected solution obtained from one Newton-iteration step. Compared to the standard method, the modified method takes more iteration steps to obtain a converged solution, but its iteration steps are far less time consuming except for the first step. In general it can be stated that the modified method takes less time for finding a converged solution than the standard method.

**Input fields: MAXFAC, MAXSTP, MAXCUT, MAXITE**

The total number of matrix factorizations, needed in the predictor-corrector steps to compute the prebuckling solution at load level $\Lambda_e^{i+1}$ starting from the known prebuckling solution at load level $\Lambda_e^i$, cannot exceed its maximum set in input field MAXFAC. The maximum number of corrector steps that is permitted for finding a converged solution from its prediction is set to a maximum in input field MAXITE. If more corrector steps are needed to find a converged solution or if the corrector steps diverge than the stepsize along the prebuckling path is cut in half. This is done until the prediction is close enough to the desired prebuckling solution so that converging corrector steps are obtained that lead to a converged solution within MAXITE corrector steps. The maximum number of times the stepsize in the path parameter is permitted to be cut in half in total for following the prebuckling path from load level $\Lambda_e^i$ to load level $\Lambda_e^{i+1}$ is set to a maximum in input field MAXCUT. The number of steps to be taken along the prebuckling path from load level $\Lambda_e^i$ to load level $\Lambda_e^{i+1}$ to compute the prebuckling solution at load level $\Lambda_e^{i+1}$ increases every time the stepsize is cut into half. The maximum number of predictor-corrector steps that is permitted along the prebuckling path in between the load levels $\Lambda_e^i$ and $\Lambda_e^{i+1}$ is set to a maximum in input field MAXSTP.

**Input fields: TOLEQS, TOLUNK**

The last two input fields in section 4.3 of the data input file are tolerances needed by the corrector (Newton-iteration) to decide whether a converged solution is found or not. The corrected solution obtained from a Newton-iteration step is accepted as converged solution as soon as the norm of the right hand side vector (or residual vector) of the system of Newton-iteration equations becomes smaller than the absolute value of TOLEQS and the norm of the change in solution vector divided by the norm of the corrected solution vector becomes smaller than the absolute value of TOLUNK.
2.2.7 Group 5: BUCKLING PROBLEM

Group 5 is the last input group of the data input file for computational module ANOVI and directs the computations for solving the buckling problem.

Input field: IBUCK

The flag IBUCK is used to turn the computations for solving the buckling problem on and off. If not set equal to 1 the program only solves the prebuckling problem. If set to 1 the program solves both the prebuckling and buckling problems.

Computational strategy

Before discussing the input fields needed to solve the buckling problem a general description of the computational strategy used for solving this problem is given. Figure 2-4 illustrates the calculation procedure when using a nonlinear prebuckling state. It is important for the following discussion to emphasize that the program is designed to take into account only positive eigenvalues. A superindex $i$ used with a variable in the following indicates that such a variable corresponds to a buckling calculation in which the prebuckling solution evaluated at load level $\Lambda^i_c$ is used.

First, the prebuckling state is computed at the starting load level $\Lambda^1_c$ specified in input field GLFS in section 4.2 of the data input file. Secondly, using an eigenvalue shift $\mu^1$ a number of eigenvalues $\rho^1$ are computed for each wavenumber $n_1$ in the interval of wavenumbers NWFRS to NWLAST taken into account. This is illustrated in figure 2-5. In the case of membrane prebuckling no further calculations
are required and program ANOVSI terminates.

In the case of nonlinear prebuckling further calculations are needed to take account of the prebuckling nonlinearity, i.e., to compute the critical bifurcation load \( \Lambda_c \) for which \( \Lambda_c^1 \) is a first approximation more accurately. To do this the program continues by searching for the smallest of the computed eigenvalues \( \rho^1 \). This eigenvalue is called the critical eigenvalue \( \rho^1_c \). The load level \( \Lambda^1_c \) is increased by \( \alpha \eta^1_c \) obtaining load level \( \Lambda^2_e \). The calculation process for calculating the critical eigenvalue is then repeated from load level \( \Lambda^2_e \) yielding the critical eigenvalue \( \rho^2_c \).

This process is repeated until the difference between to successive values of the critical bifurcation load \( \Lambda^{i+1}_c - \Lambda^i_c \) becomes in absolute value smaller than a specified tolerance or the preset maximum number of increases in load level \( \Lambda_e \) is reached. In the first case the calculation process is executed one more time whereby the load level \( \Lambda^{i+2}_e \) is set equal to \( \Lambda^{i+1}_c \). The factor \( \alpha \) (\( 0 < \alpha < 1 \)) used in increasing the load level \( \Lambda_e \) is a damping factor introduced to prevent the new load level \( \Lambda^{i+1}_e \) from being bigger than approximation \( \Lambda^{i+1}_c \) for the critical bifurcation load. The shift \( \mu^1 \) for the first eigenvalue calculation from load level \( \Lambda^1_e \) must be specified by the user. Starting from the second eigenvalue calculation the shift is set as: \( \mu^i = -\gamma \Lambda^i_c \) (\( i=2, \ldots \)). For \( \gamma \) positive this represents a downshift that becomes more and more important as \( \Lambda^i_c \) approaches \( \Lambda_c \). This is because the system matrix of the eigenvalue problem becomes singular for \( \Lambda^i_c = \Lambda_c \) if no eigenvalue shift is used. It is
computational module ANOVS

essential for the calculation process that the load level \( \Lambda^i_c + \mu^i \) never overestimates the approximation \( \Lambda^i_c \) for the critical bifurcation load. If it does the critical eigenvalue \( \rho^i_c \) becomes negative and is ignored by the program since the program only takes into account positive eigenvalues.

2.2.7.1 Group 5.1: Boundary Conditions (B.C.) buckling problem

Input fields: ISYM1,  
IBC01(1), IBC01(2), IBC01(3), IBC01(4),  
IBCLI(1), IBCLI(2), IBCLI(3), IBCLI(4)

The input flag ISYM1 sets the part of the shell used for solving the governing equations of the buckling state. The whole shell is used if it is set equal to zero in which case boundary conditions are needed at both \( x=0 \) and \( x=L \). The boundary conditions at \( x=0 \) are set by the flags IBC01(i) for \( i=1,...,4 \) as shown in table 3. The

<table>
<thead>
<tr>
<th>( i )</th>
<th>( IBC01 (i) = 1 )</th>
<th>( IBC01 (i) \neq 1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( P_1(x=0,y)=0 )</td>
<td>( U_1(x=0,y)=0 )</td>
</tr>
<tr>
<td>2</td>
<td>( S_1(x=0,y)=0 )</td>
<td>( V_1(x=0,y)=0 )</td>
</tr>
<tr>
<td>3</td>
<td>( Q_1(x=0,y)=0 )</td>
<td>( W_1(x=0,y)=0 )</td>
</tr>
<tr>
<td>4</td>
<td>( M_1(x=0,y)=0 )</td>
<td>( \beta_1(x=0,y)=0 )</td>
</tr>
</tbody>
</table>

Table 3: Buckling boundary condition settings at \( x=0 \)

boundary conditions at \( x=L \) are set by the flags IBCLI(i) for \( i=1,...,4 \) as shown in table 4. The part of the shell between \( x=0 \) and \( x=0.5L \) is used if flag ISYM1 is set

<table>
<thead>
<tr>
<th>( i )</th>
<th>( IBCLI (i) = 1 )</th>
<th>( IBCLI (i) \neq 1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( P_1(x=L,y)=0 )</td>
<td>( U_1(x=L,y)=0 )</td>
</tr>
<tr>
<td>2</td>
<td>( S_1(x=L,y)=0 )</td>
<td>( V_1(x=L,y)=0 )</td>
</tr>
<tr>
<td>3</td>
<td>( Q_1(x=L,y)=0 )</td>
<td>( W_1(x=L,y)=0 )</td>
</tr>
<tr>
<td>4</td>
<td>( M_1(x=L,y)=0 )</td>
<td>( \beta_1(x=L,y)=0 )</td>
</tr>
</tbody>
</table>

Table 4: Buckling boundary condition settings at \( x=L \)
equal to 1 or 2 in which case the user must provide boundary conditions only at \( x=0 \)
(set by the flags $IBC01(i)$). The flags $IBC01(i)$ and $IBCL1(i)$ are read in both for
$ISYM1$ equal to 1 or 2 but only the values for $IBC01(i)$ are used. The part of the shell
between $x=0.5L$ and $x=L$ is used if the flag $ISYM1$ is set equal to 3 or 4 in which case
the user must provide boundary conditions only at $x=L$ (set by the flags $IBCL1(i)$).
The flags $IBC01(i)$ and $IBCL1(i)$ are read in both for $ISYM1$ equal to 3 or 4 but only
the values for $IBCL1(i)$ are used. The settings $ISYM1=1$ and $ISYM1=3$ make $w_{11}$
symmetric and $w_{21}$ anti-symmetric with respect to $x=0.5L$. The settings $ISYM1=2$
and $ISYM1=4$ make $w_{11}$ anti-symmetric and $w_{21}$ symmetric with respect to $x=0.5L$.
It is important to notice that not all combinations of the prebuckling boundary
condition flag $ISYM0$ and the buckling boundary condition flag $ISYM1$ are allowed.
The possible combinations are listed in the example data input file. Program
ANOVS1 terminates the calculation if the user tries to enter a non-possible
combination for the flags $ISYM0$ and $ISYM1$.

**Input field: $IRBMN0$**

The input flag $IRBMN0$ is always read in but used only when computing
axisymmetric buckling modes whereby using the whole shell for the computation.
Rigid body motions can be present in the axial and circumferential directions when
computing axisymmetric buckling modes. The flag $IRBMN0$ sets the location where
these rigid body motions must be suppressed if they exist when using the whole shell
for the computation. Existing rigid body motions are suppressed at the midpoint
$x=0.5L$ of the shell when computing axisymmetric buckling modes using only one
half of the shell. It is mentioned here that rigid body motions never exist when
computing asymmetric buckling modes. Flag $IRBMN0$ comes in handy when
computing axisymmetric and asymmetric buckling modes in the same run using the
whole shell for the computation. The flag $IRBMN0$ is then used to specify where to
suppress existing rigid body motions in the computation of the axisymmetric
buckling modes without affecting the boundary condition settings for the
computation of the asymmetric buckling modes. When computing only
axisymmetric buckling modes using the whole shell, it is of course also possible to
suppress the rigid body motion in axial direction by setting at least one of the flags
$IBC01(1),IBCL1(1)$ not equal to 1 and to suppress the rigid body motion in
circumferential direction by setting at least one of the flags $IBC01(2),IBCL1(2)$ not
equal to 1.
2.2.7.2 Group 5.2: Buckling problem parameters

Group 5.2.1: Problem size

Input field: IBASFC

The program can use 4 different definitions for the buckling unknowns depending on the value of flag IBASFC and depending on whether or not axisymmetric buckling modes are computed. The part of the shell used for the computation of axisymmetric buckling modes also affects the definition used for the buckling unknowns. The 4 definitions for the buckling unknowns and the conditions under which they are used are:

1. - Axi-symmetric buckling modes \((n_1 = 0)\) using the whole shell (value for IBASFC is ignored for this case)

- Axi-symmetric buckling modes using the half shell while IBASFC is not equal to 2

\[
\begin{align*}
P_1 &= p_{11}(x) \quad ; \quad U_1 = u_{11}(x) \\
S_1 &= s_{21}(x) \quad ; \quad V_1 = v_{21}(x) \\
Q_1 &= q_{11}(x) \quad ; \quad W_1 = w_{11}(x) \\
M_1 &= m_{11}(x) \quad ; \quad \beta_1 = \beta_{11}(x)
\end{align*}
\]

2. - Axi-symmetric buckling modes \((n_1 = 0)\) using the half shell while \(IBASFC=2\)

\[
\begin{align*}
P_1 &= p_{11}(x) + p_{21}(x) \quad ; \quad U_1 = u_{11}(x) + u_{21}(x) \\
S_1 &= s_{21}(x) + s_{11}(x) \quad ; \quad V_1 = v_{21}(x) + v_{11}(x) \\
Q_1 &= q_{11}(x) + q_{21}(x) \quad ; \quad W_1 = w_{11}(x) + w_{21}(x) \\
M_1 &= m_{11}(x) + m_{21}(x) \quad ; \quad \beta_1 = \beta_{11}(x) + \beta_{21}(x)
\end{align*}
\]

One of the radial displacement functions \(w_{11}\) and \(w_{21}\) will be symmetric and the other one will be anti-symmetric with respect to \(x=0.5L\) depending on the setting of flag ISYM1.
3. - A-symmetric buckling modes \( (n_1 \neq 0) \) while IBASFC is not equal to 2

\[
P_1 = p_{11}(x) \cos(n_1y/R) \quad ; \quad U_1 = u_{11}(x) \cos(n_1y/R)
\]

\[
S_1 = s_{11}(x) \sin(n_1y/R) \quad ; \quad V_1 = v_{11}(x) \sin(n_1y/R)
\]

\[
Q_1 = q_{11}(x) \cos(n_1y/R) \quad ; \quad W_1 = w_{11}(x) \cos(n_1y/R)
\]

\[
M_1 = m_{11}(x) \cos(n_1y/R) \quad ; \quad \beta_1 = \beta_{11}(x) \cos(n_1y/R)
\]

This choice for the definition of the buckling unknowns must be made for computing the asymmetric buckling modes of orthotropic shells using the full shell length in the computation combined with a torsionless prebuckling state. This choice can be made for computing the asymmetric buckling modes of orthotropic shells using the half shell length in the computation combined with a torsionless prebuckling state.

4. - A-symmetric buckling modes \( (n_1 \neq 0) \) while IBASFC=2

\[
P_1 = p_{11}(x) \cos(n_1y/R) + p_{21}(x) \sin(n_1y/R)
\]

\[
S_1 = s_{11}(x) \sin(n_1y/R) + s_{21}(x) \cos(n_1y/R)
\]

\[
Q_1 = q_{11}(x) \cos(n_1y/R) + q_{21}(x) \sin(n_1y/R)
\]

\[
M_1 = m_{11}(x) \cos(n_1y/R) + m_{21}(x) \sin(n_1y/R)
\]

\[
U_1 = u_{11}(x) \cos(n_1y/R) + u_{21}(x) \sin(n_1y/R)
\]

\[
V_1 = v_{11}(x) \sin(n_1y/R) + v_{21}(x) \cos(n_1y/R)
\]

\[
W_1 = w_{11}(x) \cos(n_1y/R) + w_{21}(x) \sin(n_1y/R)
\]

\[
\beta_1 = \beta_{11}(x) \cos(n_1y/R) + \beta_{21}(x) \sin(n_1y/R)
\]

This choice for the definition of the buckling unknowns must be made for computing the asymmetric buckling modes of shells with an anisotropic shell wall or whenever the prebuckling state is not torsionless.

Notice that it is always possible to compute both symmetric and anti-symmetric buckling modes at the same time by setting IBASFC equal to 2 while using the half shell for the computation. If the definition of the buckling unknowns is as given in
the points 1 and 3 then it depends on flag ISYM1 if only symmetric or only anti-
symmetric buckling modes are computed while using the half shell for the
computation.

**Input field: IQUAD**

The input field IQUAD lets the user select the linear or quadratic version of the
eigenvalue problem. One should keep in mind that not using the quadratic
eigenvalue problem is actually an approximation as soon as a shell theory other than
the Donnell-Mushtari-Vlasov theory is selected. Therefore, it is recommended to
select the quadratic eigenvalue problem as soon as theory choice flag IHERY is not
set equal to 1.

**Group 5.2.2: Interval of wavenumbers**

**Input fields: NWFIRS, NWLAST**

Input field NWFIRS gives the starting value for the interval of full waves in
circumferential direction that is taken into account. Input field NWLAST gives the
ending value of this wavenumber interval (see fig. 2-5).

**Group 5.2.3: Approach of critical bifurcation load**

**Input field: APPFAC**

The absolute value of input field APPFAC is used as the damping factor \( \alpha \)
discussed in the explanation of the computational strategy (see fig. 2-4). The value
for this input field must be in between 0 and 1 and is usually set equal to 0.5.

**Input field: NSTGLF**

The maximum number of times that load level \( \Lambda_e \) can be increased in order to
compute the critical bifurcation load more accurately when using nonlinear
prebuckling is given in input field NSTGLF. This flag is read in but has no effect
when using membrane prebuckling, because the bifurcation loads are then computed
only once using the membrane prebuckling state evaluated at the starting load level
specified in input field GLFS. Notice that if one needs to compute bifurcation loads
using nonlinear prebuckling at a specific load level one can specify that load level as
starting load \( \Lambda_e^1 \) in input field GLFS and set NSTGLF equal to zero. The program
will not increase load level \( \Lambda_e \) from starting load level \( \Lambda_e^1 \) for NSTGLF=0.
Input field: **TOLGLF**

The absolute value of input field TOLGLF is used as tolerance in the lowest bifurcation load when using nonlinear prebuckling. As soon as the absolute difference between two approximations \( \Lambda^i_c \) and \( \Lambda^{i+1}_c \) (see fig. 2-4) for the critical bifurcation load \( \Lambda_c \) becomes smaller than TOLGLF then \( \Lambda^{i+1}_c \) is accepted as \( \Lambda_c \). As soon as this is the case the bifurcation loads are computed one more time for all the wavenumbers \( n_1 \) taken into account using nonlinear prebuckling at load level \( \Lambda^{i+1}_c \).

Input field: **DOWNSH**

The absolute value of input field DOWNSH is used to specify constant \( \gamma \) used in computing the downshift. The function of this downshift is explained in the discussion of the computational strategy (see fig. 2-4).

**Group 5.2.4: Subspace iteration parameters**

This group consists of 8 input fields. These input fields direct the subspace iterations.

Input fields: **IJ, KL**

The two input fields **IJ** and **KL** are seeds for the random number generator. These seeds are used to initialize the random number generator used to generate a random starting subspace spanned by NMODE vectors.

Input fields: **NMODE, NCONV**

Input field NMODE is used to specify the size of the subspace, i.e., gives the number of vectors that span the subspace. Input field NCONV specifies how many converged lowest positive eigenvalues must be computed for each wavenumber \( n_1 \). It is recommended to choose input field NMODE at least twice as big as NCONV.

Input fields: **MAXITR, TOLSUB**

Input field MAXITR specifies the maximum number of subspace iteration steps that are allowed for computing NCONV converged lowest positive eigenvalues. As soon as the change in an eigenvalue \( (\Delta \rho) \) is relative to the eigenvalue \( \rho \) itself smaller than TOLSUB, i.e., \( |\Delta \rho/\rho| < |TOLSUB| \), such an eigenvalue is considered to be converged. The subspace iteration ends as soon as MAXITR subspace iteration steps are performed or NCONV converged lowest positive eigenvalues are found.
Input fields: ISHIFT, SHIFT

The last two input fields control the eigenvalue shift used in the first eigenvalue calculation from the starting load level GLFS. Flag ISHIFT is used to turn the eigenvalue shift for the first eigenvalue calculation on and off. If turned off the eigenvalue shift in input field SHIFT is not used. If turned on input field SHIFT is used as eigenvalue shift $\mu^1$ (see fig. 2-4). These two input fields are not used anymore as soon as the load level $\Lambda_e$ in increased once for computing the critical bifurcation load more accurately when using nonlinear prebuckling. Using an eigenvalue shift for the first eigenvalue calculation is extremely important if one sets the starting load level GLFS equal to a bifurcation load, since this prevents the system matrix of the buckling problem from being singular. Furthermore, if this bifurcation load is the critical bifurcation load and one wants to include it in the output, this shift must be negative since program ANOVSI is designed to take into account only positive eigenvalues.
3 COMPUTATIONAL MODULE POSTPR

The computational module POSTPR is designed to solve the response problem of the postbuckling state whereby taking into account buckling mode interaction. It also computes the postbuckling coefficients $a_{ij}^L$ and $b_{ijk}^L$.

Computational module POSTPR takes 3 types of input files. These are:

1. *standard input file* POSTPR.INP
2. *mode identifying file & post-buckling boundary condition file*
3. *data input file(s) & data input file* of program POSTPR must be a *result output file* of program ANOVSI (see fig. 1–3)

The *standard input file* always has the name POSTPR.INP. The name for the *mode identifying file & post-buckling boundary condition file* can be chosen by the user. An example listing for the *standard input file* is given in appendix B.1. An example listing of the *mode identifying file & post-buckling boundary condition file* is given in appendix B.2. The name BCMODE.INP for this last file is chosen for convenient reference but can be changed by the user. An example listing of the third type of input file is omitted since it does not contain input fields that should be set by the user and must be provided by running computational module ANOVSI.

The listings in appendix B contain comments describing input fields. Notice that the last line of a comment block starts with the symbol $\$ $ in its first column. This indicates that input fields follow directly after that line. The variable set by the input field is always situated above the input field itself. Using the example listings in appendix B as a guideline a description of the input files needed to run computational module POSTPR is given in the sections 3.1 and 3.2.

3.1 Standard input file: POSTPR.INP

An example listing of the *standard input file* of program POSTPR is given in appendix B.1. This file handles the input and output streams of program POSTPR and must be named POSTPR.INP.

**Input fields:** *IHIST, history output file*

The input field *IHIST* determines whether or not the program produces calculation process information indicating the progress of the computation. This information can be suppressed, written to the screen or written to a so called *history*
output file by setting flag IHIST (see the comment of the example listing for setting IHIST). If the calculation process information is written to the history output file then its filename must be specified in the history output file input field (the last input field of file POSTPR.INP). The history output file input field can be left empty whenever the calculation process information is not written to the history output file. It is not allowed to leave one of the other input fields empty.

Input fields: number of data input files, data input file(s), IDATA (input table)

The number of data input files input field is given the number of data input files from which information should be read in concerning the cylinder problem for which the postbuckling computation must be performed. This information includes:

1. shell geometry, shell wall properties, discretization and data for dimensionless writing
2. shell theory choice
3. prebuckling solution
4. buckling mode(s)

The files from which this information is read must be result output files from program ANOVSI. The first 3 information items listed above are read in from one data input file only. The fourth one is read from all the specified data input files. The names of the data input files must be specified in the data input file(s) input fields of the input table that follows directly after the number of data input files input field. Each name is given on a new line in this table. The number of names specified must equal the amount given in the number of data input files input field. Empty lines are not allowed in the table. The input field IDATA to the right of the data input file filename directs what information is read from that data input file. Only buckling modes are read in if IDATA=1. Setting this flag to a value bigger then 1 causes the program to read all four information items listed above. Only one of the IDATA input fields in the table should be set bigger then 1. All the others should be set equal to 1.

The information stored in the given data input files must be the same for the following points:

1. The first 3 information points listed above. For the case of nonlinear prebuckling it is essential to ensure that load level $\Lambda_e$ at which the prebuckling solution is evaluated is the same in all the data input files.
2. The buckling boundary conditions at the shell edges $x=0$ and $x=L$.  

33
computational module POSTPR

The differences in the given data input files should only concern the buckling modes. The buckling modes in different data input files may differ only for the following points:

1. interval of wavenumbers NWFIRS to NWLAST taken into account
2. boundary conditions at the midpoint x=0.5L of the shell when using only one half of the shell for solving the buckling problem
3. the part of the shell used for solving the prebuckling and buckling problems

The possibility to give more than one data input file is designed for reading different buckling modes of the same shell problem from separate output files of program ANOVSI. This is useful for example when computing symmetric and anti-symmetric buckling modes separately in two runs with program ANOVSI and then computing their mode interaction using program POSTPR. Another example is the case that one wants to compute the interaction of two buckling modes of which one has n_1 and the other has 2n_1 full waves in circumferential direction. One can compute the buckling modes for these two wavenumbers separately in two runs with program ANOVSI without having to compute the buckling modes for the intermediate wavenumbers in between n_1 and 2n_1. Then these two buckling modes can be read in by program POSTPR from separate result output files of program ANOVSI in order to compute their mode interaction in the postbuckling state.

Input field: mode identifying file & post-buckling boundary condition file

The mode identifying file & postbuckling boundary condition file input field must be given the name of the file in which mode selection parameters are given to select modes from each data input file and in which the postbuckling boundary conditions are specified. The form of such a file is described in section 3.2. An example listing of this file named BCMODE.INP is given in appendix B.2.

Input field: result output file

The result output file input field is used to give the filename of the file to which the read in data, the solution of the postbuckling response problem and the computed postbuckling coefficients a_{ij}^L and b_{ijk}^L are written.

3.2 Mode identifying file & post-buckling boundary condition file

An example listing of the mode identifying file & post-buckling boundary condition file is given in appendix B.2. This file has two input groups. The first one
provides input fields for specifying the postbuckling boundary conditions. The second input field group is used to select the buckling modes that are taken into account from the data input file(s).

3.2.1 Group 1 : POST-BUCKLING BOUNDARY CONDITIONS

There are 10 input fields in this group. They are used to specify the boundary conditions for the postbuckling unknowns in the response problem.

Input field : IANSA

Input field IANSA switches the ANILISA simulation on and off. If set to 1 the program uses the ANILISA simulation for setting the postbuckling boundary conditions. This means that the axisymmetric solution part of the postbuckling forces $P_{11}$ and $S_{11}$ is set to zero at $x=0$ and $x=L$ regardless of the settings for the flags $IBC02(1)$, $IBC02(2)$, $IBCL2(1)$ and $IBCL2(2)$. Setting this flag equal to 1 is needed if results computed with program ANILISA are to be reproduced since this program cannot set the axisymmetric part of the postbuckling displacement components in axial and circumferential direction equal to zero at the shell edges $x=0$ and $x=L$. Program ANILISA always sets the axisymmetric part of the axial and circumferential postbuckling forces equal to zero at the shell edges. For reproducing results obtained with program ANILISA one should use the ANILISA simulation not only in the response problem but also for solving the prebuckling problem (see section 2.2.6.b). The input field IANSA has no influence on the boundary condition settings of the asymmetric parts of the postbuckling unknowns.

Input field : IRGD

Input field IRGD lets the user specify where to suppress existing rigid body motions when computing axisymmetric parts of postbuckling unknowns. By default program POSTPR uses the whole shell length for formulating and solving the postbuckling response problem. When computing axisymmetric parts of postbuckling unknowns rigid body motions can exist. These can be suppressed at $x=0$, $x=0.5L$ or $x=L$ by setting flag IRGD equal to 1, 0 or 2. Suppressing them at $x=0.5L$ (IRGD=0) is accepted only by the program if the postbuckling boundary conditions are symmetric, i.e., $IBC02(i) = IBCL2(i)$ for $i = 1,..,4$ if IANSA $\neq 1$ or for $i = 3,4$ if IANSA = 1. Program POSTPR uses the half shell $(0 < x < 0.5L)$ for computing axisymmetric parts of postbuckling unknowns in case existing rigid body motions are suppressed at the midpoint $x=0.5L$ of the shell. The program always writes the solution for the postbuckling unknowns from $x=0$ to $x=L$ to the result
output file, even if only one half of the shell was used in the computation.

Input fields: \(IBC02(1), IBC02(2), IBC02(3), IBC02(4), IBC2(1), IBC2(2), IBC2(3), IBC2(4)\)

The flags \(IBC02(i)\) and \(IBCL2(i)\) are used to specify the boundary conditions for the postbuckling unknowns at \(x=0\) and \(x=L\). The postbuckling boundary conditions at \(x=0\) are set by the flags \(IBC02(i)\) for \(i=1,...,4\) as shown in Table 5.

<table>
<thead>
<tr>
<th>(i)</th>
<th>(IBC02(i) = 1)</th>
<th>(IBC02(i) \neq 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(P_{11}(x=0,y)=0)</td>
<td>(U_{11}(x=0,y)=0)</td>
</tr>
<tr>
<td>2</td>
<td>(S_{11}(x=0,y)=0)</td>
<td>(V_{11}(x=0,y)=0)</td>
</tr>
<tr>
<td>3</td>
<td>(Q_{11}(x=0,y)=0)</td>
<td>(W_{11}(x=0,y)=0)</td>
</tr>
<tr>
<td>4</td>
<td>(M_{11}(x=0,y)=0)</td>
<td>(\beta_{11}(x=0,y)=0)</td>
</tr>
</tbody>
</table>

**Table 5:** Postbuckling boundary condition settings at \(x=0\)

The postbuckling boundary conditions at \(x=L\) are set by the flags \(IBCL2(i)\) for \(i=1,...,4\) as shown in Table 6.

<table>
<thead>
<tr>
<th>(i)</th>
<th>(IBCL2(i) = 1)</th>
<th>(IBCL2(i) \neq 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(P_{11}(x=L,y)=0)</td>
<td>(U_{11}(x=L,y)=0)</td>
</tr>
<tr>
<td>2</td>
<td>(S_{11}(x=L,y)=0)</td>
<td>(V_{11}(x=L,y)=0)</td>
</tr>
<tr>
<td>3</td>
<td>(Q_{11}(x=L,y)=0)</td>
<td>(W_{11}(x=L,y)=0)</td>
</tr>
<tr>
<td>4</td>
<td>(M_{11}(x=L,y)=0)</td>
<td>(\beta_{11}(x=L,y)=0)</td>
</tr>
</tbody>
</table>

**Table 6:** Postbuckling boundary condition settings at \(x=L\)

The postbuckling boundary conditions actually used for the axisymmetric part of the unknowns \(P_{11}, S_{11}, U_{11}\) and \(V_{11}\) can differ from the settings specified by the flags \(IBC02(1), IBC02(2), IBCL2(1)\) and \(IBCL2(2)\) depending on the settings of the flags \(IANSA\) and \(IRGD\).

**3.2.2 Group 2: BUCKLING MODE SPECIFYING**

The second input group lets the user specify the buckling modes that should be read in from each *data input file*.  

36
Input block(s): number of modes.
NWI, mode number (input table)
(repeated number of data input files times)

To select buckling modes from a data input file the mode identifying file & post-buckling boundary condition file must contain an input block like:

```
FILE : KHOT_5_7.OUT

number of modes
.................
3

NWI  mode number
.................
5    3
6    1
7    2
```

for each of the data input files. These input blocks are written directly below one another in the order in which the names of the data input files, from which these input blocks select buckling modes, are given in the data input file(s) input fields in standard input file POSTPR.INP. The buckling modes taken into account from data input file i (i counted from the top) are selected in input block i (i counted from the top) of input group 2 of the mode identifying file & post-buckling boundary condition file. Input block i contains a number of modes input field followed by a table with on each line the two input fields NWI and mode number that select one buckling mode from data input file i. The total number of lines in this table must equal the specified number of modes to read in from data input file i. The parameters used to select a certain buckling mode are the number of full waves NWI in circumferential direction and the mode number for the specified wavenumber NWI. For each wavenumber NWI a number of buckling modes are computed with the lowest bifurcation loads. The mode numbers of these buckling modes increase with increasing bifurcation loads. The buckling mode with the lowest bifurcation load for NWI waves has mode number 1. Figure 2-5 illustrates this mode numbering.

It is recommended to specify the buckling modes per data input file in the order of increasing wavenumbers NWI. The numbering of the buckling modes in the result output file of program POSTPR will then be the same as the order in which they were specified in the successive input blocks. If not the program will renumber the buckling modes per data input file to increasing wavenumbers NWI. The numbering of the buckling modes in the result output file of program POSTPR starts with number 1.

It is important to emphasize that all selected buckling modes must be independent from one another. It is the users responsibility to make sure that no
duplicate buckling modes are specified.

The buckling modes read in from the data input file(s) can have one of the four definitions listed in section 2.2.7.2 for the buckling unknowns. Each of these definitions can be transformed to the form given in the eq. (1.19) to (1.26). This transformation is carried out by program POSTPR whereby not used unknowns in the eq. (1.19) to (1.26) are set equal to zero. In case the prebuckling and buckling unknowns are listed in a data input file for one half of the shell, program POSTPR constructs them for the whole shell using symmetry and anti-symmetry conditions at \( x=0.5L \).

Example mode identifying file & post-buckling boundary condition file BCMODE.INP illustrates the selection of buckling modes from the two data input files KHOT_5_7.OUT and KHOT_11_13.OUT. Notice the order of the mode selection input blocks in file BCMODE.INP and the order of the filenames in the data input file(s) input fields in example standard input file POSTPR.INP. The files KHOT_5_7.OUT and KHOT_11_13.OUT are result output files of program ANOSSI. They both are concerned with the same shell problem but the interval of wavenumbers for which buckling modes are stored differs. In file KHOT_5_7.OUT four buckling modes are stored for each of the wavenumbers 5, 6 and 7. In file KHOT_11_13.OUT four buckling modes are stored for each of the wavenumbers 11, 12 and 13. All of these buckling modes are computed using nonlinear prebuckling at the same load level \( \Lambda_c \).
4 COMPUTATIONAL MODULE IMPSHP

The computational module IMPSHP is designed to produce an imperfection input file for computational module IMPFAC. The computational module IMPFAC computes imperfection form factors $\alpha_m^L$ and $\beta_m^L$ based on the imperfection shape functions $W_m$ (see eq. (1.11)) stored in such an imperfection input file.

Computational module IMPSHP reads data from the following files:

1. standard input file IMPSHP.INP
2. result output file(s) from computational module POSTPR from which buckling modes are read to which imperfection shapes are affined (such a file is called postbuckling result file in the following)
3. dimensions & gridpoints file which must be a result output file of computational module POSTPR

The standard input file always has the name IMPSHP.INP. An example listing for this file is given in appendix C. An example listing of the other input files is omitted since they do not contain input fields that should be set by the user and must be provided by running computational module POSTPR.

The listing in appendix C contains comments describing input fields. Notice that the last line of a comment block starts with the symbol $\$ $ in its first column. This indicates that input fields follow directly after that line. The variable set by the input field is always situated above the input field itself. Using the example listing in appendix C as a guideline a description of the standard input file needed to run computational module IMPSHP is given in section 4.1.

4.1 Standard input file : IMPSHP.INP

Input fields : IHIST, history output file

Input field IHIST determines whether or not the program produces calculation process information indicating the progress of the computation. This information can be suppressed, written to the screen or written to a so called history output file by setting flag IHIST (see the comment of the example listing for setting IHIST). If the calculation process information is written to the history output file then its filename must be given in the history output file input field (the last input field of file IMPSHP.INP). The history output file input field can be left empty whenever the calculation process information is not written to the history output file.
Input field: number of imperfection shapes

This input field asks for the total number of imperfection shapes $W_m$ to be computed, i.e., the maximum value $M$ of subindex $m$ in eq. (1.11).

Input fields: imperfection shape no., type,

read shape from post-buckling result file, buckling mode no.,
$\bar{n}_1$, $\bar{m}_1$

(input table)

These input fields are the columns of a table. Each line of this table defines one imperfection shape. The number of lines in this table must equal the specified number of imperfection shapes. Empty lines are not allowed in the table.

The program numbers the imperfection shapes in the order they are defined in the table. The imperfection defined in the first line of the table gets number 1. The imperfection defined in the second line of the table gets number 2, etc. Although the table contains an imperfection shape no. input field that cannot be left empty, the value of this input field is not actually used by program IMPSHP to number the imperfection shapes.

Two types of imperfection shapes can be chosen. The first one is affine to a buckling mode. The second one is a so called modal imperfection shape. They are defined as follows:

type 1: This type specifies an imperfection shape that is affine to a buckling mode. Such an imperfection shape is defined as follows

$$
\overline{W}_t(x,y) = w_{11}(x) \cos(n_1 y/R) + w_{21}(x) \sin(n_1 y/R)
$$

(4.1)

in which the right hand side is identical to the radial displacement component $W_t$ of buckling mode 1 to which the imperfection is affine. Imperfection shape index $m$ is replaced by 1 to indicate that no summations are implied over repeated indices. If the type input field in the table is set to 1 then one must also specify the input fields read shape from post-buckling result file and buckling mode no. on the same line. The read shape from post-buckling result file input field is given the filename of the postbuckling result file in which the buckling mode can be found to which the imperfection is affine. The input field buckling mode no. is given the number of that buckling mode in the postbuckling result file. The other two input fields $\bar{n}_1$ and $\bar{m}_1$ are not used if input field type is set equal to 1 and can be omitted.
type 2: This type specifies a modal imperfection shape. Such an imperfection shape is defined as follows

\[ W_1(x,y) = w_{11}(x) \cos\left(\frac{n_y}{R}\right) ; \quad w_{11}(x) = \sin\left(\frac{m_x}{L}\right) \] (4.2)

Imperfection shape index \( m \) is replaced by 1 to indicate that no summations are implied over repeated indices. If the type input field in the table is set equal to 2 then one must specify the input fields \( n_y \) and \( m_x \) on the same line. The other input fields read shape from postbuckling result file and buckling mode no. must be omitted when specifying a modal imperfection shape. The program computes a modal imperfection at finite difference gridpoints.

Input field: imperfection shape output file

The input field imperfection shape output file is used to give the filename of the file to which the constructed imperfection shapes are written. This file is the imperfection input file for computational module IMPFAC.

Input field: dimensions & gridpoints file

The input field dimensions & gridpoints file is given the filename of a result output file of computational module POSTPR. From this file the program reads the shell dimensions \( L, R, t \) and the number of finite difference intervals used. These data must be the same for all the postbuckling result files specified in the read shape from post-buckling result file and dimensions & gridpoints file input fields. The program IMPHSP terminates as soon as a difference is detected. The program uses the data read from the dimensions & gridpoints file for the computation of modal imperfection shapes. It is not allowed to leave this input field empty, even if modal imperfection shapes are not defined.
5 COMPUTATIONAL MODULE IMPFAC

The computational module IMPFAC is designed to compute the imperfection form factors $\alpha_m^L$ and $\beta_m^L$ for given imperfection shapes $\bar{W}_m$. Further, it produces a bifurcation loads, postbuckling coefficients & imperfection form factors file needed for running computational module COHPAT. Program IMPFAC stores bifurcation loads ($\Lambda_L$), postbuckling coefficients ($a_{ij}^L, b_{ijk}^L$) and imperfection form factors ($\alpha_m^L, \beta_m^L$) in this file. Using these data computational module COHPAT can compute the limit load for an imperfect shell with initial imperfection: $\bar{w} = t \xi_m \bar{W}_m$ (summation is implied over repeated index $m$ from $1$ to $M$).

Computational module IMPFAC takes 3 input files. These are:

1. standard input file IMPFAC.INP
2. postbuckling result file which must be a result output file of computational module POSTPR
3. imperfection input file which must be an imperfection shape output file of computational module IMPSHP

The standard input file always has the name IMPFAC.INP. An example listing for this file is given in appendix D. Example listings of the second and third input file are omitted since they do not contain input fields that should be set by the user and must be provided by running the computational modules POSTPR and IMPSHP.

The listing in appendix D contains comments describing all input fields. Notice that the last line of a comment block starts with the symbol $*$ in its first column. This indicates that input fields follow directly after that line. The variable set by the input field is always situated above the input field itself. Using the example listing in appendix D as a guideline a description of standard input file IMPFAC.INP is given in section 5.1.

5.1 Standard input file: IMPFAC.INP

Input fields : IHist, history output file

Input field IHist determines whether or not the program produces calculation process information indicating the progress of the computation. This information can be suppressed, written to the screen or written to a so called history output file by setting flag IHist (see the comment of the example listing for setting IHist). If the calculation process information is written to the history output file then its
computational module IMPFAC

filename must be given in the *history output file* input field (the last input field of file IMPFAC.INP). The *history output file* input field can be left empty whenever the calculation process information is not written to the *history output file*. It is not allowed to leave one of the other input fields empty.

**Input field: postbuckling result file**

The *postbuckling result file* input field is given the name of a *result output file* of program POSTPR. The following data are read in from this file:

1. shell geometry and shell wall properties
2. shell theory choice
3. prebuckling solution
4. selected buckling modes (see the *buckling mode number* input table)
5. postbuckling coefficients $a_{ij}^L$ and $b_{ijk}^L$ (for the selected buckling modes)

**Input field: buckling mode number (input table)**

The buckling modes taken into account are selected from the buckling modes stored in the specified *postbuckling result file*. Each buckling mode in this file has a *buckling mode number*. To select a buckling mode from this file its *buckling mode number* in it is given in the *buckling mode number* table. Only one *buckling mode number* is given on each line of this table. The last line of this table must be followed directly by a line with the character C in its first column. Empty lines are not allowed in the table.

**Input field: imperfection input file**

The *imperfection input file* input field is given the filename of the file in which the imperfection shapes $W_m$ are stored for which the imperfection form factors are to be computed. This file must be an *imperfection shape output file* created with computational module IMPSHP.

**Input field: result output file**

The file which filename is given in the *result output file* input field is created by program IMPFAC. Program IMPFAC stores all the read in data from the specified *postbuckling result file* and *imperfection input file* in its *result output file* together with the computed imperfection form factors $\alpha_m^L$ and $\beta_m^L$.

**Input field: path following input file**

The file which filename is given in the *path following input file* input field is created by program IMPFAC. The bifurcation loads $(\Lambda_L)$, postbuckling coefficients
computational module IMPFAC

$(a^L_{ij}, b^L_{ijk})$ and imperfection form factors $(\alpha^L_m, \beta^L_m)$ are stored in this file for all the buckling modes and imperfection shapes taken into account. It is used as bifurcation loads, postbuckling coefficients & imperfection form factors file in runs with program COHPAT.
6 COMPUTATIONAL MODULE COHPAT

The computational module COHPAT is designed to solve the equations that describe the dependence of the load factor $\Lambda$ on a number of perturbation parameters $\xi_j$ in the vicinity of bifurcation points having (nearly) simultaneous bifurcation loads $\Lambda_L$. These equations are given by $(i, j, k, L = 1, \ldots, N; m = 1, \ldots, M)$

$$(\Lambda - \Lambda_L) \xi_L = \Lambda_L a_{ij}^L \xi_i \xi_j + \Lambda_L b_{ij}^L \xi_i \xi_k \xi_k - \Lambda_c \alpha_m^L \xi_m - (\Lambda - \Lambda_c) \beta_m^L \xi_m$$  \hspace{1cm} (6.1)

in which $\Lambda_c$ indicates the load level at which the prebuckling state is evaluated. Repeated lowercase indices imply summation, but repeated uppercase indices do not. The equations (6.1) are a set of $N$ nonlinear simultaneous equations in the $N+1$ unknowns $\xi_1, \ldots, \xi_N, \Lambda$. This set of equations is solved using a path-following technique as described by Riks [4]. Capital $N$ is used for the number of buckling modes taken into account. Capital $M$ is used for the total number of imperfection shapes $\bar{W}_m$. The equations (6.1) describe the behavior of a perfect shell if one sets the imperfection form factors $\alpha_m^L$, $\beta_m^L$ or the imperfection amplitudes $\xi_m$ equal to zero. Otherwise they describe the behavior of an imperfect shell with initial imperfection $\bar{w} = t \bar{\xi}_m \bar{W}_m$.

Computational module COHPAT reads input from 3 files. These are:

1. standard input file COHPAT.INP
2. calculation control file
3. one of the two files:
   - bifurcation loads, postbuckling coefficients & imperfection form factors file
     which must be a path following input file created by program IMPFAC
   - result output file of computational module COHPAT

The standard input file always has the name COHPAT.INP. The name for the calculation control file can be chosen by the user. An example listing of standard input file COHPAT.INP is given in appendix E.1. An example listing of the calculation control file is given in appendix E.2. The name CONTRL.INP for this last file is chosen for convenient reference but can be changed by the user. Example listings of the 2 types of input file that can be used as third input file are omitted since they do not contain input fields that should be set by the user and must be provided by running the computational modules IMPFAC or COHPAT.

The listings in appendix E contain comments describing input fields. Notice
that the last line of a comment block starts with the symbol $ in its first column. This indicates that input fields follow directly after that line. The variable set by the input field is always situated above the input field itself. Using the example listings in appendix E as a guideline a description of the input files needed to run computational module COHPAT is given in the sections 6.1 and 6.2.

6.1 Standard input file: COHPAT.INP

The standard input file of computational module COHPAT handles its input and output streams and must be named COHPAT.INP. An example listing of this file is given in appendix E.1.

Input fields: IHIST, history output file

The input field IHIST determines whether or not the program produces calculation process information indicating the progress of the computation. This information also includes error messages. The calculation process information can be suppressed, written to the screen or written to a so called history output file by setting flag IHIST (see the comment of the example listing for setting IHIST). If the calculation process information is written to the history output file then its filename must be given in the history output file input field (the last input field of file COHPAT.INP). The history output file input field can be left empty whenever the calculation process information is not written to the history output file. It is not allowed to leave one of the other input fields empty.

Input field: calculation control file

The calculation control file input field must be given the name of a file such as the one listed in appendix E.2. Its input fields are discussed in section 6.2. This file is used to direct the path-following solution procedure and to give the imperfection amplitudes $\tilde{\xi}_m$.

Input field: bifurcation loads, postbuckling coefficients & imperfection form factors file

or

restart file

This input field is given the name of the bifurcation loads, postbuckling coefficients & imperfection form factors file or the name of the restart file.

The bifurcation loads, postbuckling coefficients & imperfection form factors file must be a path following input file created by program IMPFAC. Such a file is
used as input when the computation of the path is started from a specified load level $\Lambda_s$ (see fig. 6-1). The input field $ISTART$ in the calculation control file must not be set equal to 1 or 2 for this case.

The restart file is a result output file from an earlier run with program COHPAT. Such a file is used as input if the calculation of the solution branch that is listed in the restart file is restarted from one of its solution points in order to compute a part of the same solution branch (input field $ISTART$ in the calculation control file is set equal to 2 for this case). A restart file is also used as input if the calculation of an intersecting branch is started from a bifurcation point that is listed in the restart file (input field $ISTART$ in the calculation control file is set equal to 1 for this case).

Input field: result output file

The result output file input field is given the name of the file to which the calculation results are written. This file is created by computational module COHPAT. The read in prebuckling load level $\Lambda_s$, bifurcation loads $\Lambda_L$, postbuckling coefficients $a_{ij}^L, b_{ijk}^L$ and imperfection form factors $\alpha_m^L, \beta_m^L$ are written to this file together with the computed points of the computed solution branch of eq. (6.1). Each point is classified as PATH, LIMIT or BIFURCATION point. For a LIMIT point the load factor $\Lambda$ reaches an extreme value. At a BIFURCATION point the solution path is intersected by another solution branch. All points that are neither LIMIT nor BIFURCATION point are classified as PATH points. The program automatically detects and calculates limit and bifurcation points whenever they exist along the part of the solution path (primary branch) that is computed. For each bifurcation point that is detected along the primary branch the program also computes the tangent to the intersecting secondary branch.

6.2 Calculation control file

The file CONTRL.INP listed in appendix E.2 is an example listing of the calculation control file. This file is divided into two main parts. Its path-following parameters part is discussed in section 6.2.1. Its imperfection amplitudes part is discussed in section 6.2.2.

6.2.1 Path-following parameters

The path-following parameters are the parameters that direct the computation
of a solution branch of the equations (6.1) using a path following method.

**Input fields**: ISTART, NOSBIF, STLOAD

The input field ISTART specifies the way in which the calculation of a solution path is started. Figure 6-1 illustrates the possibilities. This figure shows two solution branches for a two-mode analysis. These are labeled as primary and secondary branch. The dashed lines are the projections of the secondary branch on the $\xi_2, \Lambda$ and $\xi_1, \xi_2$ planes. Points on the primary branch are labeled with index $I$. The path parameter at point $I$ is $\eta_I$.

Setting input field ISTART not equal to 1 or 2 starts the calculation of the primary branch at the starting load level $\Lambda_s$. This starting load level is given in input field STLOAD and is usually set equal to 0. For this case a path following input file created by program IMPFAC is needed as bifurcation loads, postbuckling coefficients & imperfection form factors file. Input field NOSBIF is read in but not used for ISTART not equal to 1 or 2.

Setting input field ISTART equal to 1 restarts the calculation at the bifurcation point with point number NOSBIF ($I=b$ is figure 6-1) in the restart file for calculating one part of the secondary branch. This is the branch switching option of program COHPAT. The restart file must be a result output file of program COHPAT containing the part of the primary branch that includes both bifurcation point NOSBIF and the tangent to the secondary branch at this point. Input field STLOAD is read in but not used for ISTART equal to 1.

Setting input field ISTART equal to 2 restarts the calculation of solution points on the primary branch at point NOSBIF of that branch. This restart option is used for computing another part of the same solution branch. The restart file must be a result output file of program COHPAT containing solution points of a part of the primary
branch including point NOSBIF. It is not allowed to choose the first solution point
listed in the restart file as point NOSBIF. Input field STLOAD is read in but not used
for ISTART equal to 2.

input fields : NUMSTR, STPATH, STSTEP

Input field NUMSTR is used to specify the starting value for the point numbers
I (s in figure 6-1). Input field STPATH is used to give the starting value for the path
parameter $\eta$ at starting point NUMSTR ($\eta_s$ in figure 6-1). Input field STSTEP is
used to give the starting value for the increase in path parameter used for
constructing the first prediction from the first solution point NUMSTR ($\Delta \eta_s$ in
figure 6-1). The direction in which a branch is computed is influenced by the sign of
input field STSTEP as follow:

1. If ISTART not 1 or 2 : + $\rightarrow$ $\Lambda$ increases from STLOAD
   - $\rightarrow$ $\Lambda$ decreases from STLOAD
2. If ISTART = 1 : + $\rightarrow$ in direction of tangent to secondary branch
   - $\rightarrow$ opposite to tangent to secondary branch
3. If ISTART = 2 : + $\rightarrow$ in same direction as in restart file
   - $\rightarrow$ opposite to direction in restart file

The direction in which the path parameter $\eta$ changes for increasing point numbers I
is not influenced by the sign of input field STSTEP. Path parameter $\eta$ always
increases for increasing point numbers I.

input fields : NUMMAX, ETAMAX, FACLIM

The input field NUMMAX specifies the maximum number of solution points I
that are allowed to be computed. The maximum value allowed for the path
parameter $\eta_I$ is specified in input field ETAMAX. One must make sure that the
relation ETAMAX $>$ STPATH holds. The input field FACLIM determines the
maximum increase in path parameter that is allowed after finding a limit point. This
maximum increase in path parameter is limited by the relation

$$\eta_I - \eta_1 \leq (\eta_I - \eta_s) \cdot FACLIM \quad \text{with : } FACLIM > 0.0 \quad (6.2)$$
in which $\eta_I$ is used for the path parameter at the limit point (See figure 6-1).

input fields : ITARGT, DAMPIN, STPMIN, STPMAX

These 4 input fields are used to control the computation of the increase in path
parameter $\Delta \eta_I$ used for computing the prediction for solution point I+1. In first
instance the increase in path parameter $\Delta \eta_I$ is determined from
\[ \Delta \eta_s = |\text{STSTEP}| \quad ; \quad \Delta \eta_I = \left( \frac{J^*}{J_I} \right)^\gamma \Delta \eta_{I-1} \quad \text{for } I = s+1, s+2, \ldots \] (6.3)

in which \( J^* \) is a user defined target iteration count, \( \gamma \) is a damping factor and \( J_I \) is the number of Newton-iteration steps needed to compute converged solution point \( I \) from its prediction. Target iteration count \( J^* \) is given in input field \( \text{ITARGT} \).

Damping factor \( \gamma \) is given in input field \( \text{DAMPIN} \). Notice that the increase in path parameter is always positive. After computing \( \Delta \eta_I \), the program checks whether it is smaller than the minimum allowed increase in path parameter given in input field \( \text{STPMIN} \). If \( \Delta \eta_I < \text{STPMIN} \) then \( \Delta \eta_I \) is set equal to \( \text{STPMIN} \). The program also checks \( \Delta \eta_I \) for exceeding the maximum allowed increase in path parameter given in input field \( \text{STPMAX} \). If \( \Delta \eta_I > \text{STPMAX} \) then \( \Delta \eta_I \) is set equal to \( \text{STPMAX} \). The possible values for the increase in path parameter thus become

\[ \text{STPMIN} \leq \Delta \eta_I \leq \text{STPMAX} \quad \text{with : } I = s, s+1, s+2, \ldots \] (6.4)

The values given for \( \text{STPMIN} \) and \( \text{STPMAX} \) must be positive and must be such that the relation \( \text{STPMIN} \leq \text{STPMAX} \) holds.

**Newton-iteration**

Before continuing with the discussion of the remaining input fields it is needed to introduce the notation used for the Newton-iteration. The vector \( \hat{\xi} \) of perturbation parameters is given by

\[ \hat{\xi}^t = \left[ \xi_1 \ldots \xi_N \right] \] (6.5)

The equations (6.1) are represented by

\[ f_L(\hat{\xi}, \Lambda) = 0 \quad \text{with : } L = I, \ldots, N \] (6.6)

The additional equation needed to solve these equations for a particular point is represented by

\[ f_{N+1}(\hat{\xi}, \Lambda; \eta) = 0 \] (6.7)

A vector of unknowns \( \hat{x} \) and a vector of residual forces \( \hat{f} \) are introduced as follows

\[ \hat{x}^t = \left[ \xi_1 \ldots \xi_N \Lambda \right] \quad ; \quad \hat{f}^t = \left[ f_1 \ldots f_N \quad f_{N+1} \right] \] (6.8)

Using the notations just introduced the set of Newton-iteration equations for obtaining solution point \( I \) is given as \( (J = I, \ldots) \).
\[ \hat{x}^{i+1} = \hat{x}^i - G^{-1}(\hat{x}^i; \eta_i) \hat{f}(\hat{x}^i; \eta_i) \]  

(standard Newton-iteration) \hfill (6.9)

where

\[ G = \begin{bmatrix} K & \hat{b} \\ n^t & n_{N+1} \end{bmatrix} ; \quad K = \begin{bmatrix} K_{11} & \cdots & K_{1N} \\ \vdots & \ddots & \vdots \\ K_{N1} & \cdots & K_{NN} \end{bmatrix} \]

\[ K_{Lj} = \frac{\partial f_L}{\partial \xi_j} \text{ for } Lj = 1, \ldots, N \]

\[ \hat{b}^t = \begin{bmatrix} b_1 \\ \vdots \\ b_N \end{bmatrix} \text{ with } b_L = \frac{\partial f_L}{\partial \Lambda} \text{ for } L = 1, \ldots, N \]

\[ n^t = \begin{bmatrix} n_1 \\ \vdots \\ n_N \end{bmatrix} \text{ with } n_j = \frac{\partial f_{N+1}}{\partial \xi_j} \text{ for } j = 1, \ldots, N ; \quad n_{N+1} = \frac{\partial f_{N+1}}{\partial \Lambda} \]

The definitions of the functions \( f_L(\xi, \Lambda) \) for \( L=1, \ldots, N \) and \( f_{N+1}(\xi, \Lambda; \eta) \) used for computing the \( j \)th solution point are given by

\[ f_L = (\Lambda - \Lambda_L) \xi_L - \Lambda_L a_{ij} \xi_i \xi_j - \Lambda_L L_{ij} \xi_i \xi_j + \Lambda_L e_m \xi_m^L + (\Lambda - \Lambda_e) B_m^L \xi_m \] \hfill (6.10)

\[ f_{N+1} = (\xi_k - \xi_k^{(i-1)}) (\xi_k - \xi_k^{(i-1)}) + (\Lambda - \Lambda^{(i-1)})^2 - (\eta_i - \eta_{i-1})^2 \] \hfill (6.11)

in which summation is implied by repeated lowercase subindices. The indices \( L, i, j, \) and \( k \) range from \( 1 \) to \( N \). Index \( m \) ranges from \( 1 \) to \( M \).

**input fields**: `MAXCUT`, `CUTFAC`, `OVERSH`

The prediction for solution point \( I+1 \), computed using an increase in path parameter \( \Delta \eta_I \) obtained from eq. (6.3) and that fulfills condition (6.4), does not always result in a converging Newton-iteration in which this prediction is used as first iterate (prediction is used as iterate \( J=1 \) in the equations (6.9)). The stepsize \( \Delta \eta_I \) is multiplied with `CUTFAC` in case a converging Newton-iteration could not be obtained using this stepsize. The value for input field `CUTFAC` must be in between 0 and 1 and is usually set equal to 0.5. After
multiplication of stepsize $\Delta \eta_i$ with $CUTFAC$ a new stepsize $\Delta \eta_i^*$ is obtained. The new prediction calculated with stepsize $\Delta \eta_i^*$ is closer to solution point $I+1^*$ than the prediction computed with stepsize $\Delta \eta_i$ was to point $I+1$. This is illustrated in figure 6-2. Therefore, the chance is increased that the new prediction is located in the convergence area of the Newton-iteration for convergence to point $I+1^*$. If convergence is still not possible with the new prediction than the stepsise is decreased another time by multiplying $\Delta \eta_i^*$ with $CUTFAC$. This process is repeated until a converging Newton-iteration is obtained or until $MAXCUT$ decreases in stepsise are performed without obtaining a converging Newton-iteration for finding the next solution point from point $I$. The program terminates if more than $MAXCUT$ reduces in stepsise are needed to obtain a converging Newton-iteration process. The program also terminates if stepsise reductions make the stepsise smaller than its preset minimum value in input field $STPMIN$.

The input field $OVERSH$ determines a so called overshoot value. It is used in case stepsise $\Delta \eta_i$ is such that solution point $I+1$ is a bifurcation point. Matrix $G$ is in such a point singular, causing poor behaviour of the Newton-iteration in its vicinity. If this is detected by the program the Newton-iteration is stopped and the increase in path parameter $\Delta \eta_i$ is multiplied by overshoot value $OVERSH$. This will move solution point $I+1$ passed the bifurcation point if the value of $OVERSH$ is set bigger then 1.

**input fields**: $IBLMAX$, $EPS$, $TOLETA$, $TOLDK$

As soon as a limit or a bifurcation point is passed, the program automatically starts to compute these points up to the desired accuracy.

A limit point is detected by a sign change of the determinant of matrix $K$. As soon as this happens between two solution points $I$ and $I+1$ these points become the starting values for a regula falsi iteration\(^5\) for finding the limit point. This iteration temporarily governs the change in path parameter until the limit point is found or $IBLMAX$ regula falsi iteration steps are performed. The limit point is found as soon as the absolute value of the determinant of matrix $K$ becomes smaller than $TOLDK$ and the absolute change in the path parameter becomes smaller than $TOLETA$ at some point in the regula falsi iteration. The tolerances given in the input fields $TOLDK$ and $TOLETA$ must be positive.

A bifurcation point is detected by a sign change of the determinant of matrix $G$. As soon as this happens between two solution points $I$ and $I+1$ these points become the starting values for a regula falsi iteration for finding the bifurcation point. This iteration temporarily governs the change in path parameter until the
bifurcation point is found or $IBLMAX$ regula falsi iteration steps are performed. The bifurcation point is found as soon as the absolute change in the path parameter becomes smaller than $TOLETA$ at some point in the regula falsi iteration.

The input field $EPS$ is used by the program to decide whether a matrix is singular or not. It should be set equal to some small value. See the comment in the example listing of the calculation control file for details.

**input fields: $EPSEIG$, $ITRMAX$**

The two input fields $EPSEIG$ and $ITRMAX$ are used in the computation of the tangent to the secondary branch at a bifurcation point. To compute this tangent the program performs an inverse power iteration. The input field $EPSEIG$ is used for specifying the tolerance in the eigenvalue computed with this iteration. As soon as the error in the eigenvalue becomes smaller than $EPSEIG$ the eigenvalue and corresponding eigenmode are considered to be found. However, no more than $ITRMAX$ inverse power iteration steps will be performed for finding the converged eigenvalue.

**input fields: $IFULLN$, $INTMAX$, $MAXFAC$**

Input field $IFULLN$ specifies the type of Newton-iteration to be used. Setting flag $IFULLN$ equal to 0 selects the standard Newton-iteration process described by the equations (6.9) in which the system matrix $G$ is assembled and factorized at each Newton-iterate $\tilde{x}^J$ (see figure 6-2). Setting flag $IFULLN$ not equal to 0 selects the modified Newton-iteration process described by ($J = 1,...$)

$$
\tilde{x}^{J+1} = \tilde{x}^J - G^{-1}(\tilde{x}^P;\eta_I) \dot{f}(\tilde{x}^J;\eta_I)
$$

(modified Newton-iteration) (6.12)

in which the system matrix $G$ is assembled and factorized only at the points $\tilde{x}^P$. One of the points $\tilde{x}^P$ is the prediction. If divergence is detected in the Newton-iteration after initial convergence then the last Newton-iterate for which the Newton-iteration was converging is used for $\tilde{x}^P$. Although the modified version takes more Newton-iteration steps $J$ than the standard version does, it usually is far less time consuming since each Newton-iteration step in the modified version takes less time than in the standard version except for the steps with $J=P$.

Input field $INTMAX$ specifies the maximum number of Newton-iteration steps that will be performed to find a converged solution from its prediction, i.e., Newton-iteration index $J$ cannot exceed $INTMAX$.

Input field $MAXFAC$ gives the maximum number of times that matrix $G$ is allowed to be assembled and factorized in total for the whole run with computational module COHPAT.
computational module COHPAT

input fields: TOLRES, TOLUNK

The input fields TOLRES and TOLUNK specify tolerances from 0 used in the Newton-iteration to see if a converged solution point is found in which case the Newton-iteration can be stopped. A Newton-iterate \( \hat{x}^J \) is considered to be converged as soon as

\[
\|f(\hat{x}^J; \eta_J)\| < \text{TOLRES} \quad \text{and} \quad \|\hat{x}^J - \hat{x}^{J-1}\| < \text{TOLUNK} \cdot \|\hat{x}^J\| \quad (6.13)
\]

In case \(\|\hat{x}^J\|=0\) the program sets \(\|\hat{x}^J\|\) equal to unity whereby transforming TOLUNK into an absolute tolerance instead of a relative one.

6.2.2 Imperfection amplitudes

The imperfection amplitudes input part of the calculation control file is only needed if the calculation is started from starting load level \(\Lambda_1\) (see fig. 6-1), i.e., if flag ISTART is not set equal to 1 or 2. If a restart is made from a bifurcation (ISTART=1) or a limit (ISTART=2) point then the imperfection amplitudes input part can be omitted, since it is not read in for ISTART equal to 1 or 2. The imperfection amplitudes are for ISTART equal to 1 or 2 read from the restart file.

input field: IFACCH

Input flag IFACCH is used to select which imperfection form factors should be taken into account. Setting it equal to 0 leaves out all the imperfection form factors \(\alpha_m^L\) and \(\beta_m^L\) by setting them equal to 0. The equations (6.1) describe for IFACCH=0 the behaviour of the perfect shell. Setting IFACCH equal to 1 includes the imperfection form factors \(\alpha_m^L\), but leaves out the imperfection form factors \(\beta_m^L\) by setting \(\beta_m^L\) equal to 0. Setting IFACCH not equal to 0 or 1 includes all the imperfection form factors \(\alpha_m^L\) and \(\beta_m^L\).

input fields: IMPERFECTION, AMPLITUDE (input table)

These input fields actually represent a table in which an imperfection amplitude \(\tilde{\xi}_m\) must be given for each of the \(M\) imperfection shapes present in the imperfection input file that was used as input file for program IMPFAC to compute the imperfection form factors stored in the specified bifurcation loads, postbuckling coefficients & imperfection form factors file. On each line of the table an imperfection shape number is given in the IMPERFECTION column followed by the amplitude \(\tilde{\xi}_m\) of the corresponding imperfection shape in the AMPLITUDE column. The imperfection shape number is the number of an imperfection shape in the just
mentioned *imperfection input file* and must be chosen from the interval: \(1, \ldots, M\). Empty lines are not allowed in this table and the total number of lines must be equal to \(M\). Each imperfection shape number in the table must be unique, i.e., duplicate imperfection shape numbers are not allowed.

Generally the *IMPERFECTION,AMPLITUDE* table must be filled as described above. The only exception is the case that input field *IFACCH* is set equal to 0. If *IFACCH* is set equal to 0 this input table can be left completely empty.
REFERENCES


appendix A

APPENDIX A
Input files for program ANOVI

Example listings of the input files needed to run program ANOVI are given in this appendix. Program ANOVI uses two input files. One of them is its standard input file that must be named ANOVI.INP. It handles the input and output streams of the program. The other one is the data input file containing the problem definition. Its name is given in file ANOVI.INP. The name KHOT_11_13.INP is used in the following for the name of the data input file, but can be changed by the user into some other appropriate name for the particular problem under consideration.

A.1 Standard input file ANOVI.INP : example listing

```
C.............................................................................C
C
C STANDARD INPUT FILE for program ANOVI
C
C
C purpose : - definition of data input file
- definition of result output file
- definition of history output file

IHIST = flag to specify the destination of the
calculation process information :
not 6 or 9  --> no calculation process
information
   = 6   --> to screen
   = 9   --> to history output file

data input file = name of problem definition file
result output file = name of file to which the calculation
results for the prebuckling problem and
the buckling problem are written
history output file = name of file to which information about
the calculation process of the prebuckling
and buckling problems is written
(only needed if IHIST=9)

Notice : Input fields for file names start at column 1
and end at column 70

IHIST
$....................
   6
C
C data input file
$....................
KHOT_11_13.INP
C
C result output file
$....................
KHOT_11_13.OUT
C
```

57
A.2 Data input file: example listing KHOT_11_13.INP

```
... DATA INPUT FILE of program ANOVSI ...

SUBJECT = text string of maximum 80 characters in which the
subject of the calculation can be indicated briefly
CRNAME = name of person who created this input file
SUBJECT (Maximum 80 characters starting in the 1-th column.
Spaces and tabs preceding the text are removed.)

KTH shell
CRNAME (Maximum 80 characters starting in the 1-th column.
user name)

1 SHELL DATA :

1.1 Dimensions of the cylinder:
L = cylinder length
R = cylinder radius
t = wall thickness

L       R       t
12.50   6       0.036

1.2 Shell wall data:
NMAT = number of materials
NLAM = number of laminae

NMAT   NLAM
2       3

1.2.1 Materials:
mm = material number (each material is specified on a
new line up to a total of NMAT materials)
E11 = modulus of elasticity in the direction of axe 1
E22 = modulus of elasticity in the direction of axe 2
nu12 = poisson ratio 12
G12 = shear modulus

mm   E11  E22  nu12  G12
1    7.5E+06 3.5E+06 0.25   1.25E+06
2    40.0E+06 4.5E+06 0.25   1.50E+06
...```
appendix A

1.2.2 Laminae:

ln = lamina number (Each lamina is specified on a new line up to a total of NLAM laminae. The numbering starts at the outside surface of the cylinder wall)

RH = relative lamina thickness (the actual lamina thickness HL is calculated from:

\[ HL = RH/\text{Total} \times t \]

in which Total represents the sum of all lamina thicknesses and t is the wall thickness)

TTESTA = angle of lamina with x-axis (positive according to right hand side rule in z-direction)

lmat = material number of lamina

<table>
<thead>
<tr>
<th>ln</th>
<th>RH</th>
<th>TTESTA</th>
<th>lmat</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>40</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>40</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

1.3 Stiffener data:

ISTRIN = not 1 -> no stringers
1 --> stringers presents

IRING = not 1 --> no rings
1 --> rings present

Es, Er = modulus of elasticity for stringers, rings
Gs, Gr = shear modulus for stringers, rings
As, Ar = cross section area of stringers, rings
Is, Ir = moment of inertia with respect to centroid of stringers, rings
Js, Jr = torsional constant of stringers, rings
Dc, Dr = spacing of stringers, rings
Cc, Cr = distance from shell-midsurface to the centroid of the stringers, rings (positive inward)

1.3.1 Stringers:

ISTRIN

<table>
<thead>
<tr>
<th></th>
<th>Es</th>
<th>Gs</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Gs</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>As</th>
<th>Is</th>
<th>Js</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

1.3.2 Rings:

IRING

<table>
<thead>
<tr>
<th></th>
<th>Er</th>
<th>Gr</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Dr</th>
<th>Cr</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Ar</th>
<th>Ir</th>
<th>Jr</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

2 FINITE DIFFERENCE:

NIH = number of finite difference intervals used for one half of the shell

NIH

<table>
<thead>
<tr>
<th>NIH</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
</tr>
</tbody>
</table>

59
Appendix A

3 DATA FOR DIMENSIONLESS WRITING:

- DNU = reference value poison ratio
- ESTAR = reference value modulus of elasticity

DNU   ESTAR
- 0.250 7.5E+06

4 & 5 SHELL THEORY CHOICE:

- Itheory = flag to specify the shell theory which is to be used:
  1 -> Donnell-Mushtari-Vlasov
  2 -> Sanders without rotation about the shell normal
  3 -> Novozhilov without rotation about the shell normal
  4 -> Sanders including rotation about the shell normal only in the twisting curvature
  5 -> Novozhilov including rotation about the shell normal only in the twisting curvature
  6 -> Sanders including rotation about the shell normal (*)
  7 -> Novozhilov including rotation about the shell normal (*)

(*) using Cohen's approximation for the normal rotation in the definition of the axial and circumferential normal strains

Itheory

- 1

4 PREBUCKLING PROBLEM:

- INONL = flag to specify membrane or nonlinear prebuckling:
  0 -> membrane prebuckling (requires the under point a listed input)
  not 0 -> nonlinear prebuckling (requires the under point b listed input)

INONL

- 1

a) MEMBRANE PREBUCKLING:

- ISYMO = prebuckling boundary condition flag:
  0 -> use whole shell for calculation
  1 -> symmetry at x=0.5L (U0=V0=Q0=BERTA0=0)
  use part of shell: 0 < x < 0.5L
  2 -> symmetry at x=0.5L (U0=V0=Q0=BERTA0=0)
  use part of shell: 0.5L < x < L

- AU = coefficient which specifies the x station where axial displacement component u0 is zero: u0 = 0 at x = AU * L, in case ISYMO=0.

- AV = coefficient which specifies the x station where tangential displacement component v0 is zero: v0 = 0 at x = AV * L, in case ISYMO=0.

ISYMO   AU   AV

- 1 0.50 0.50
appendix A

\[
\begin{align*}
\text{lambda} &= \text{lambdaC} + \text{GLF} \times \text{lambdaSTAR} = \text{axial load} \\
\text{stilde} &= \text{stildeC} + \text{GLF} \times \text{stildeSTAR} = \text{torsional load} \\
\text{pe} &= \text{peC} + \text{GLF} \times \text{peSTAR} = \text{external pressure} \\
&\text{(GLF is generalized load factor)} \\
\text{lambdaC} &= 0 \quad \text{stildeC} = 0 \quad \text{peC} = 0 \\
\text{lambdaSTAR} = 1 \quad \text{stildeSTAR} = 0 \quad \text{peSTAR} = 0 \\
\hline
\end{align*}
\]

b) NONLINEAR PREBUCKLING :

4.1 Boundary Conditions (B.C.) prebuckling problem :

\[
\text{ISYM} = \text{prebuckling boundary condition flag}:
\]

\[
0 \rightarrow \text{use whole shell for calculation} \\
1 \rightarrow \text{symmetry at x=0.5L (U0=V0=Q0=BETA0=0)} \\
\text{use part of shell: } 0 < x < 0.5L \quad \text{(B.C. are only needed at x=0)} \\
2 \rightarrow \text{symmetry at x=0.5L (U0=V0=Q0=BETA0=0)} \\
\text{use part of shell: } 0.5L < x < L \quad \text{(B.C. are only needed at x=L)}
\]

\[
\begin{align*}
\text{IBCO}0(1) &= \text{not 1} \rightarrow U0 \text{ prescribed at } x=0 \\
1 \rightarrow V0 \text{ prescribed at } x=0 \\
\text{IBCO}0(2) &= \text{not 1} \rightarrow U0 \text{ prescribed at } x=0 \\
1 \rightarrow V0 \text{ prescribed at } x=0 \\
\text{IBCO}0(3) &= \text{not 1} \rightarrow W0 \text{ prescribed at } x=0 \\
1 \rightarrow Q0 \text{ prescribed at } x=0 \\
\text{IBCO}0(4) &= \text{not 1} \rightarrow BETA0 \text{ prescribed at x=0} \\
1 \rightarrow M0 \text{ prescribed at } x=0 \\
\text{IBCLO}(1) &= \text{not 1} \rightarrow U0 \text{ prescribed at } x=L \\
1 \rightarrow V0 \text{ prescribed at } x=L \\
\text{IBCLO}(2) &= \text{not 1} \rightarrow W0 \text{ prescribed at } x=L \\
1 \rightarrow Q0 \text{ prescribed at } x=L \\
\text{IBCLO}(3) &= \text{not 1} \rightarrow W0 \text{ prescribed at } x=L \\
1 \rightarrow Q0 \text{ prescribed at } x=L \\
\text{IBCLO}(4) &= \text{not 1} \rightarrow BETA0 \text{ prescribed at x=L} \\
1 \rightarrow M0 \text{ prescribed at } x=L \\
\text{ISYM} &= 1 \\
\hline
\end{align*}
\]

4.2 Loading :

\[
\text{GLFS} = \text{starting value of generalized load factor GLF}
\]

line loads at x=0:

\[
\begin{align*}
\text{P}(0) &= \text{P}(0) + \text{GLF} \times \text{p}(0) = \text{axial load (tension positive)} \\
\text{S}(0) &= \text{S}(0) + \text{GLF} \times \text{S}(0) = \text{counterclockwise torsion} \\
\text{Q}(0) &= \text{Q}(0) + \text{GLF} \times \text{Q}(0) = \text{transverse load in negative z-direction} \\
\text{M}(0) &= \text{M}(0) + \text{GLF} \times \text{M}(0) = \text{bending moment around y-axis positive according to right hand rule in negative y-direction} \\
\hline
\end{align*}
\]

61
appendix A

line loads at x=L :

\[ \begin{align*}
F_0(L) &= F_a(L) + GLF * F_b(L) = \text{axial load (tension positive)} \\
S_0(L) &= S_a(L) + GLF * S_b(L) = \text{counter clockwise torsion} \\
Q_0(L) &= Q_a(L) + GLF * Q_b(L) = \text{transverse load in positive x-direction} \\
M_0(L) &= M_a(L) + GLF * M_b(L) = \text{bending moment around y-axis positive according to right hand rule in positive y-direction}
\end{align*} \]

surface loads :

\[ \begin{align*}
pe_\_ &= \text{pe_a} + GLF * \text{pe_b} = \text{external dead load pressure} \\
\text{pe}_\_ &= \text{(CR}^2/8\text{E}^2\text{)} \text{pe} \\
\text{pfe}_\_ &= \text{pfe_a} + GLF * \text{pfe_b} = \text{external follower pressure load} \\
\text{pfe}_\_ &= \text{(CR}^2/8\text{E}^2\text{)} \text{pfe}
\end{align*} \]

displacements at x=0 :

\[ \begin{align*}
U_0(0) &= U_a(0) + GLF * U_b(0) = \text{axial displacement (u/t)} \\
V_0(0) &= V_a(0) + GLF * V_b(0) = \text{circumferential displacement (v/t)} \\
W_0(0) &= W_a(0) + GLF * W_b(0) = \text{normal displacement (w/t)} \\
\text{BERTa0}(0) &= \text{BERTa(0)} + GLF * \text{BERTab}(0) = \text{rotation about y-axis positive according to right hand rule in positive y-direction}
\end{align*} \]

displacements at x=L :

\[ \begin{align*}
U_0(L) &= U_a(L) + GLF * U_b(L) = \text{axial displacement (u/t)} \\
V_0(L) &= V_a(L) + GLF * V_b(L) = \text{circumferential displacement (v/t)} \\
W_0(L) &= W_a(L) + GLF * W_b(L) = \text{normal displacement (w/t)} \\
\text{BERTa0}(L) &= \text{BERTa(L)} + GLF * \text{BERTab(L)} = \text{rotation about y-axis positive according to right hand rule in positive y-direction}
\end{align*} \]

GLFS

0.000000

surface loads :

\[ \begin{align*}
\text{pe_a} & \quad \text{pfe_a} \\
0 & \quad 0 \\
\text{pe_b} & \quad \text{pfe_b} \\
0 & \quad 0
\end{align*} \]

line loads at x=0 :

\[ \begin{align*}
F_a(0) & \quad S_a(0) & \quad Q_a(0) & \quad M_a(0) \\
0 & \quad 0 & \quad 0 & \quad 0 \\
F_b(0) & \quad S_b(0) & \quad Q_b(0) & \quad M_b(0) \\
-1 & \quad 0 & \quad 0 & \quad 0
\end{align*} \]

line loads at x=L :

\[ \begin{align*}
F_a(L) & \quad S_a(L) & \quad Q_a(L) & \quad M_a(L) \\
0 & \quad 0 & \quad 0 & \quad 0 \\
F_b(L) & \quad S_b(L) & \quad Q_b(L) & \quad M_b(L) \\
0 & \quad 0 & \quad 0 & \quad 0
\end{align*} \]

62
displacements at x=0 :

\[
\begin{align*}
U_a(0) & \quad V_a(0) & \quad W_a(0) & \quad \text{BERTAa}(0) \\
0 & \quad 0 & \quad 0 & \quad 0 \\
U_b(0) & \quad V_b(0) & \quad W_b(0) & \quad \text{BERTB}(0) \\
0 & \quad 0 & \quad 0 & \quad 0
\end{align*}
\]

displacements at x=L :

\[
\begin{align*}
U_a(L) & \quad V_a(L) & \quad W_a(L) & \quad \text{BERTAa}(L) \\
0 & \quad 0 & \quad 0 & \quad 0 \\
U_b(L) & \quad V_b(L) & \quad W_b(L) & \quad \text{BERTB}(L) \\
0 & \quad 0 & \quad 0 & \quad 0
\end{align*}
\]

4.3 Processing parameters for path following method :

IBORD = not 1 --> generalized load factor used as path parameter
1 --> approximation for path length used as path parameter
IASPCZ = not 1 --> block tridiagonal matrix is not factorized for every Newton iteration step
(modified Newton iteration)
1 --> block tridiagonal matrix is factorized for every Newton iteration step (standard Newton iteration)
MAXFAC = maximum number of times that the block tri-diagonal matrix can be factorized in total
MAXSTP = maximum number of steps along the prebuckling path
MAXCUT = maximum number of times that the stepsize is cut in half
MAXITE = maximum number of Newton iterations per step along the pre-buckling path
TOLBQS = tolerance in the residual vector
TOLINK = relative tolerance in the vector of unknowns

IBORD  IASPCZ
0 0

MAXFAC  MAXSTP  MAXCUT  MAXITE
10 11 12 20

TOLBQS  TOLINK
1E-9 1E-9

5 BUCKLING PROBLEM :

IBUCK = buckling problem flag :
not 1 --> calculate only the prebuckling problem
1 --> calculate the prebuckling problem and the buckling problem

IBUCK
1
5.1 Boundary Conditions (B.C.) buckling problem:
---

(only needed if IBUCK=1)

**ISYM1** = buckling boundary condition flag:

- **0** -> use whole shell for calculation
- **1** -> use part of shell: \( 0 < x < 0.5L \)
  - **select**: \( w1 \) -- symmetric mode
  - \( w2 \) -- anti-symmetric mode
  - B.C. at \( x=0.5L \): \( u1i=m1i=0 \)
    - symmetric -- \( p1i,w1i,w1i,miI \)
    - anti-symmetric -- \( p1i,w2i,w2i,miI \)
    - (B.C. are only needed at \( x=0 \))

- **2** -> use part of shell: \( 0 < x < 0.5L \)
  - **select**: \( w1 \) -- anti-symmetric mode
  - \( w2 \) -- symmetric mode
  - B.C. at \( x=0.5L \): \( u1i=m1i=0 \)
    - symmetric -- \( p1i,w2i,w1i,miI \)
    - anti-symmetric -- \( p1i,w1i,w2i,miI \)
    - (B.C. are only needed at \( x=0 \))

- **3** -> use part of shell: \( 0.5L < x < L \)
  - **select**: \( w1 \) -- symmetric mode
  - \( w2 \) -- anti-symmetric mode
  - B.C. at \( x=0.5L \): \( u1i=m1i=0 \)
    - symmetric -- \( p1i,w1i,w2i,miI \)
    - anti-symmetric -- \( p1i,w2i,w2i,miI \)
    - (B.C. are only needed at \( x=L \))

- **4** -> use part of shell: \( 0.5L < x < L \)
  - **select**: \( w1 \) -- anti-symmetric mode
  - \( w2 \) -- symmetric mode
  - B.C. at \( x=0.5L \): \( u1i=m1i=0 \)
    - symmetric -- \( p1i,w2i,w2i,miI \)
    - anti-symmetric -- \( p1i,w1i,w1i,miI \)
    - (B.C. are only needed at \( x=L \))

The following table shows the allowed combinations for the prebuckling boundary condition flag ISYM0 and the buckling boundary condition flag ISYM1:

<table>
<thead>
<tr>
<th>ISYM0</th>
<th>ISYM1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0,1,2</td>
</tr>
<tr>
<td>2</td>
<td>0,3,4</td>
</tr>
</tbody>
</table>

**IBEMNO** = flag to specify where to suppress existing rigid body motions when calculating axisymmetric buckling modes (i.e. for NW=0) whereby using the whole shell length for the computation:

- **0** -> suppress existing rigid body motions at \( x=0 \)
- **nonzero** -> suppress existing rigid body motions at \( x=L \)

(This flag is read always but used only if NW=0 is included in the wave number interval from NWTEST to NWWLAST and the buckling mode computation is performed using the whole shell length.)

- **IBC01(1)** = not 1 --> \( u1i=0 \); \( u2i=0 \) at \( x=0 \)
  - 1 --> \( p1i=0 \); \( p2i=0 \) at \( x=0 \)
- **IBC01(2)** = not 1 --> \( v1i=0 \); \( v2i=0 \) at \( x=0 \)
  - 1 --> \( s1i=0 \); \( s2i=0 \) at \( x=0 \)
- **IBC01(3)** = not 1 --> \( w1i=0 \); \( w2i=0 \) at \( x=0 \)
  - 1 --> \( q1i=0 \); \( q2i=0 \) at \( x=0 \)
- **IBC01(4)** = not 1 --> \( berta1i=0 \); \( berta2i=0 \) at \( x=0 \)
  - 1 --> \( m1i=0 \); \( m2i=0 \) at \( x=0 \)

64
IBCL1(1) = not 1 --> u1I=0 ; u2I=0 at x=L
1 --> p1I=0 ; p2I=0 at x=L
IBCL1(2) = not 1 --> v1I=0 ; v2I=0 at x=L
1 --> s1I=0 ; s2I=0 at x=L
IBCL1(3) = not 1 --> w1I=0 ; w2I=0 at x=L
1 --> q1I=0 ; q2I=0 at x=L
IBCL1(4) = not 1 --> beta1II=0 ; beta2II=0 at x=L
1 --> m1I=0 ; m2I=0 at x=L

ISYMI IREMNO
1 0

B.C. at x=0 : (for ISYMI=0 or ISYMI=1 or ISYMI=2)

IBC01(1) IBC01(2) IBC01(3) IBC01(4)
1 0 0 1

B.C. at x=L : (for ISYMI=0 or ISYMI=3 or ISYMI=4)

IBC1(1) IBC1(2) IBC1(3) IBC1(4)
0 0 0 0

5.2 Buckling problem parameters:

5.2.1 Problem size:

IBASFC = flag to specify how many basis functions are used in the circumferential direction:
not 2 --> 1 basis function is used:
unknowns : p1I, q1I (**), w1I, beta1I,
(*) --> or s2I if NW=0
(**) --> or v2I if NW=0
2 --> 2 basis functions are used:
unknowns : p1I, q1I, q2I, m2I,
       u1I, v1I, w1I, beta1I,
p2I, s2I, q2I, m2I,
u2I, v2I, w2I, beta2I

The value for IBASFC is ignored by the program for computing axisymmetric buckling modes (NW=0) using the whole shell length for the computation in which case the buckling unknowns are:
p1I, s2I, q1I, m1I, u1I, v2I, w1I, beta1I

IQUD = flag used to specify which to solve the linear or the quadratic eigenvalue problem:
not 1 --> solve linear eigenvalue problem
1 --> solve quadratic eigenvalue problem

IBASFC IQUD
2 1

5.2.2 Interval of wavenumbers:

NWFRS = first wavenumber in the interval of wavenumbers for which the buckling problem is executed:
NW : [NWFRS ... NWLAST]
(NW = number of full waves in the circumferential direction)

NWLAST = last wavenumber in the interval of wavenumbers for which the buckling problem is executed:
NW : [NWFRS ... NWLAST]
(NW = number of full waves in the circumferential direction)

NWFRS NWLAST
11 13
5.2.3 Approach of critical bifurcation load:

(only needed if IBUCK=1)

The critical bifurcation load is defined as the lowest bifurcation load found for a certain wavenumber in the interval of wavenumbers: [NWFRS ... NWLAST]

APPFAC = approach factor
0 < APPFAC < 1; usually equal to 0.5
The new load level GLFnew from which the critical bifurcation load can be computed more accurately is obtained from:

GLFnew = GLFold + APPFAC * (GLFbic - GLFold)

where:
GLFold = load level from which GLFbic is computed
GLFbic = approximation for the critical bifurcation load calculated from prebuckling load level GLFold

These load factor increases are carried out until the absolute change in the critical bifurcation load is less than TOLGLF or until the maximum number of critical bifurcation load approaches NSTGLF is reached.

The first load level from which the critical bifurcation load is calculated is GLFS, which is the starting value for the generalized load factor specified at the prebuckling loading.

NSTGLF = maximum number of times that the generalized load factor can be increased to approach the critical bifurcation load in order to calculate it more accurately when using nonlinear prebuckling. This flag is read in but has no effect when using membrane prebuckling. (For membrane prebuckling the bifurcation loads are computed only once using the membrane prebuckling solution at starting load level GLFS)

TOLGLF = absolute tolerance in the critical bifurcation load. The critical bifurcation load is approached from below until the absolute change in the critical bifurcation load is less than the absolute value of TOLGLF or until the maximum number of approaches NSTGLF is reached.

DOWNSH = factor with which the negative of the current load factor is multiplied to obtain an eigenvalue shift along the linearized prebuckling state. The eigenvalue problem is in fact shifted away from the lowest eigenvalue to prevent singularity of the subspace iteration process for the case that the prebuckling problem is linearized at a load level which is (nearly) equal to the lowest bifurcation load. It is not used for the first eigenvalue calculation for which the eigenvalue problem is formulated at the starting load level GLFS, but takes effect as soon as the load factor is increased once to calculate the lowest bifurcation load more accurately. (DOWNSH > 0)

APPFAC  NSTGLF  TOLGLF  DOWNSH
0.7    20       1E-8    0.10

5.2.4 Subspace iteration parameters:

(only needed if IBUCK=1)

IJ = first seed for the initializing routine of the random number generator. The random number generator is used to construct NMODE 'random' starting vectors for the subspace iteration.
(0 <= IJ <= 31328)

KL = second seed for the initializing routine of the random number generator. The random number generator is used to construct NMODE 'random' starting vectors for the subspace iteration.
(0 <= KL <= 30081)
appendix A

NMODE = subspace size, i.e., number of vectors that span the subspace
NCONV = number of converged lowest positive eigenvalues that must be calculated; minimum value = 1
maximum value = NMODE
MAXITR = maximum number of subspace iteration steps that are allowed per eigenvalue calculation for computing NCONV converged lowest positive eigenvalues
TOLSUB = relative tolerance in the eigenvalues. The subspace iteration is converged as soon as NCONV of the lowest positive eigenvalues are calculated with relative accuracy TOLSUB
ISHIFT = flag to specify whether or not an eigenvalue shift is used:
          0 not 1 -> eigenvalue shift not used
          1 -> eigenvalue shift used
(only used for eigenvalue problem formulated at the starting load level GLFS)
SHIFT = shift in eigenvalue (only used if ISHIFT=1)
(only used for eigenvalue problem formulated at the starting load level GLFS)

IJ       KL
         1000    1000

NMODE NCONV
         12      4

MAXITR TOLSUB
         30  1.0E-6

ISHIFT SHIFT
         0.0

*****************************************************************************
END of DATA INPUT FILE of program ANOFSI
*****************************************************************************
APPENDIX B
Input files for program POSTPR

To run program POSTPR two input files are needed containing input fields that must be provided by the user. Example listings of these input files are given in this appendix. One of them is the so called standard input file of program POSTPR and must be named POSTPR.INP. It handles the input and output streams of program POSTPR. An example of this input file is given in appendix B.1. The other one is the so called mode identifying file & post-buckling boundary condition file of which file BCMODE.INP listed in appendix B.2 is an example. The name BCMODE.INP for this file is chosen for convenient reference and can be changed by the user.

B.1 Standard input file POSTPR.INP : example listing

```
PROGRAM POSTPR

  * STANDARD INPUT FILE for program POSTPR

  purpose:
  * definition of data input file(s)
    (they are output files of program ANOVA)
  * definition of what data should be read from each data input file
  * definition of the mode identifying and post-
    buckling boundary condition file
  * definition of result output file
  * definition of history output file

  INIST
  * flag to specify the destination of the calculation process information:
    not 6 or 7 --> no calculation process information
    = 6     --> to screen
    = 7     --> to history output file

  data input file(s) * name(s) of the output file(s) of pro-
                      gram ANOVA containing the prebuckling solution together with the (interac-
                      ting) buckling mode(s) for which the postbuckling response problem must be solved

  IDATA
  * flag for controlling which data should be read from each data input file:
    IDATA > 0 --> read only buckling modes
    IDATA = 1 --> read general data, prebuckling solution and buckling modes

  result output file * name of file to which the calculation results obtained for the post-buckling problem are written

  history output file * name of file to which information about the calculation process of the postbuck-
                       ling problem is written
                       (only needed if INIST=7)
```

68
Notice: Input fields for file names start at column 1
and end at column 70

IHIST

number of data input files

data input file(s) IDATA

KNOT_5_7.OUT 1
KNOT_13_13.OUT 2

mode identifying file &
post-buckling boundary
condition file

result output file

KNOT.OUT

history output file
(only needed if IHIST=?)

SCREEN.OUT

END of STANDARD INPUT FILE for program POSTPR

B.2 Mode identifying file & post-buckling boundary condition file:
example listing BCMODE.INP

MODE IDENTIFYING FILE & POST-BUCKLING BOUNDARY CONDITION FILE
of program POSTPR

1 POST-BUCKLING BOUNDARY CONDITIONS:

IANS A = ANILISA simulation flag used for constructing
boundary conditions which must be satisfied
by the computed axi-symmetric post-buckling
solutions:
not 1 --> don't use the ANILISA simulation
1 --> use the ANILISA simulation

IRGD = flag used to suppress rigid body motions, if
any, when calculating axi-symmetric post-
buckling solutions:
0 --> suppress rigid body motions at the
midpoint of the shell at x=0.5L
(only possible for symmetric post-
buckling boundary conditions so
that the calculation can be per-
formed using only one half of the
shell (*))
1 --> suppress rigid body motions at x=0
2 --> suppress rigid body motions at x=L

(*) : The choice IRGD=0 is only accepted by the
program if one makes sure that:
if IANS A not 1 --> IBC02(i)=IBCL1(i) for i=1,2,3,4
if IANS A = 1 --> IBC02(i)=IBCL2(i) for i= 3,4)
appendix B

IBC02(1) = not 1 --> UIJ=0 at x=0
1 --> PIJ=0 at x=0

IBC02(2) = not 1 --> VIJ=0 at x=0
1 --> SIJ=0 at x=0

IBC02(3) = not 1 --> WIJ=0 at x=0
1 --> QIJ=0 at x=0

IBC02(4) = not 1 --> BERTA1J=0 at x=0
1 --> MIZ=0 at x=0

IBC02(1) = not 1 --> UIJ=0 at x=L
1 --> PIJ=0 at x=L

IBC02(2) = not 1 --> VIJ=0 at x=L
1 --> SIJ=0 at x=L

IBC02(3) = not 1 --> WIJ=0 at x=L
1 --> QIJ=0 at x=L

IBC02(4) = not 1 --> BERTA1J=0 at x=L
1 --> MIZ=0 at x=L

ANLISA simulation:

IANSA
....

0

Suppression of rigid body motions:

IRGD
....

0

B.C. at x=0:

IBC02(1)  IBC02(2)  IBC02(3)  IBC02(4)
1   0   0   1

B.C. at x=L:

IBC02(1)  IBC02(2)  IBC02(3)  IBC02(4)
1   0   0   1

FILE : KNOT_5_7.OUT

number of modes
..........
3

NWI   mode number
..........
6   3
7   2

FILE : KNOT_11_13.OUT

number of modes
..........
2

NWI   mode number
..........
12   1
12   3

END of
MODE IDENTIFYING FILE & POST-BUCKLING BOUNDARY CONDITION FILE of program POSTPR

--------------------------------------------------------------------------------------

70
APPENDIX C
Standard input file for program IMPSHP

To run program IMPSHP one input file is needed of which the input fields must be provided by the user. This input file is the so called standard input file of program IMPSHP and must always be named IMPSHP.INP. An example listing of this file is given in appendix C.

```
C..............................................C
C
C STANDARD INPUT FILE for program IMPSHP
C
C..............................................C

purpose : Specifying the imperfection shape(s) that should be produced by program IMPSHP. This program saves the constructed imperfection shape(s) in the specified imperfection shape output file.

IHIST = flag to specify the destination of the calculation process information :
not 6 or 7 --> no calculation process information
   = 6   --> to screen
   = 7   --> to history output file

type : 1 --> AFFINE to buckling mode -->
One must give the name of the post-buckling result file and the number of the buckling mode in that file to which the imperfection is affine. (n1_bar and m1_bar can be omitted)

2 --> MODAL : WI_bar = WI_bar cos(n1_bar*y/H)
   where : WI_bar = sin(n1_bar*pi*x/L)
One must give n1_bar and m1_bar and MUST omit the other input fields (i.e., the name of the post-buckling result file and the buckling mode number must be omitted)

dimensions & gridpoints file = name of the postbuckling result file from which the cylinder length L, radius R and wall thickness t as well as the number of finite difference intervals used in axial direction for the whole shell are read.

IHIST

..............................................
6

number of imperfection shapes
..............................................
2

imperfection shape no.   type   read shape from post- buckling n   m   buckling result file mode no.  I   I
..............................................
1       1      KHOT.POS               1   I
2       2

imperfection shape output file
..............................................

KNOT_IMP.SHP

dimensions & gridpoints file
..............................................

KNOT.POS

```
appendix C

history output file (only needed if IHIST=7)

SCREEN.OUT

END of STANDARD INPUT FILE for program IMPUSH
APPENDIX D

Standard input file for program IMPFAC

To run program IMPFAC one input file is needed of which the input fields must be provided by the user. This input file is the so called *standard input file* of program IMPFAC and must always be named IMPFAC.INP. An example listing of this file is given in appendix D.

```plaintext
STANDARD INPUT FILE for program IMPFAC

purpose: specification of the postbuckling result file
    - selection of the buckling modes that are taken into account
    (from the specified postbuckling result file)
    - specification of the imperfection input file
    - specification of the result output file of program IMPFAC
    - specification of the path following input file
      produced by program IMPFAC
    - specification of the history output file

IHIST = flag to specify the destination of the calculation process information:
    not 6 or 7 --> no calculation process information
    = 6 --> to screen
    = 7 --> to history output file

postbuckling result file = name of postbuckling result file from which the shell data, the loading, the choice for the shell theory, the prebuckling solution and the selected buckling mode(s) are read (must be a result output file of program POSTPR)

buckling mode number = input table used to select the buckling modes that are taken into account from the given postbuckling result file. One buckling mode is selected on each line of the table by giving its buckling mode number in the postbuckling result file. The first line after the last buckling mode number must start with character C in its first column. Empty lines are not allowed in the table.

imperfection input file = name of the file from which the imperfection shape functions are read in (must be an imperfection shape output file of program IMPRFIP)

result output file = name of file to which the calculated imperfection form factors are written together with the read in data from the postbuckling result file and the imperfection input file

path following input file = name of the file in which bifurcation loads, postbuckling coefficients and imperfection form factors will be stored. It is created by program IMPFAC to serve as bifurcation loads, postbuckling coefficients & imperfection form factors file for program COMPAT
```
appendix D

history output file = name of file to which information about the calculation process of the imperfection form factors computation is written (only needed if IHIST=7)

Notice: Input fields for file names start at column 1 and end at column 70

IHIST
$ ............................................

6

postbuckling result file
$ ............................................

KNOT.POS

buckling mode number
$ ............................................

1

imperfection input file
$ ............................................

KHO7_IMP.SHF

result output file
$ ............................................

KNOT.PAC

path following input file
$ ............................................

KNOT.CFF

history output file (only needed if IHIST=7)
$ ............................................

SCREEN.OUT

-------------------------------------------------------------------C

END of STANDARD INPUT FILE for program IMPFAC

-------------------------------------------------------------------C
APPENDIX E

Input files for program COHPAT

To run program COHPAT two input files are needed of which the input fields must be provided by the user. Example listings of these two input files are given in this appendix. One of them is the so called standard input file of program COHPAT that must be named COHPAT.INP. It handles the input and output streams of program COHPAT. An example listing of this input file is given in appendix E.1. The other one is the so called calculation control file of which file CONTRL.INP listed in appendix E.2 is an example. The name for the calculation control file can be chosen by the user.

E.1 Standard input file COHPAT.INP : example listing

C--------- C--------- C--------- C--------- C--------- C---------
C
C
C STANDARD INPUT FILE for program COHPAT C
C
C
C purpose : - definition of calculation control file
C - definition of bifurcation loads, postbuckling
C coefficients and imperfection form factors file
C or restart file
C - definition of result output file
C - definition of history output file
C

IHIST = flag to specify the destination of the
      calculation process information :
      not 6 or 7  --> no calculation process
      = 6         --> to screen
      = 7         --> to history output file

calculation control = name of the file that directs the path-
file following procedure

bifurcation loads, = name of a path following input file cre-
postbuckling coeffi-
   cients & imperfection form factors file
   This file is used as input file if flag
   ISTART in the calculation control file
   is not set equal to 1 or 2

restart file = name of a result output file of program
               COHPAT
               (Such a file is used as input file if
                flag ISTART in the calculation control
                file is set equal to 1 or 2)

result output file = name of file to which the read in data
                     and the computed solution points must be
                     written

history output file = name of file to which information about
                     the calculation process is written
                     (only needed if IHIST=7)

Notice : Input fields for file names start at column 1
         and end at column 70

75
E.2 Calculation control file: example listing CONTRL.INP

```
C
C $  7
C $ calculation control file
C $ CONTROL.INP
C $ bifurcation loads, postbuckling coefficients & imperfect form factors file
C $ or restart file
C $  KNOT.CFF
C $
C $ result output file
C $ KNOT.PAT
C $
C $ history output file
C $ (only needed if INIST=7)
C $ SCREEN.OUT
C
C END of STANDARD INPUT FILE for program COHFAT
C
C
```

```
C
C $ 0
C
```

76
### Appendix E

<table>
<thead>
<tr>
<th>NUMSTR</th>
<th>STLOAD</th>
<th>STPATH</th>
<th>STSTEP</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.0000</td>
<td>0.00</td>
<td>0.0100</td>
</tr>
</tbody>
</table>

- **NUMMAX** = maximum number of points of the path that are allowed to be computed
- **ETAMAX** = maximum value that is allowed for the path parameter
- **FACLIM** = factor that determines the maximum increase in path parameter that is allowed after finding the first limit point as follows: \( \text{FACLIM} = 0 \) \( \text{max_increase_path_parameter_from_first_limit_point} = \frac{\text{FACLIM}}{\text{path_parameter_first_limit_point}} \)
  \( \text{path_parameter_starting_value} = \text{STPATH} \)

<table>
<thead>
<tr>
<th>NUMAX</th>
<th>ETAMAX</th>
<th>FACLIM</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>300.00</td>
<td>1.00</td>
</tr>
</tbody>
</table>

- **ITARGT** = target iteration count. Used in determining the next step in path parameter when not calculating bifurcation or limit points as follows: `next_step = [(ITARGT/IK)*DAMPIN + previous_step]` in which **IK** is the Newton-iteration count for the last obtained converged solution.
- **DAMPIN** = damping factor (usually set to 0.5), (See **ITARGT**)
- **STPMIN** = minimum allowed stepsize in path parameter (>0)
- **STMAX** = maximum allowed stepsize in path parameter (>0)

<table>
<thead>
<tr>
<th>ITARGT</th>
<th>DAMPIN</th>
<th>STPMIN</th>
<th>STMAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>0.5</td>
<td>0.0000001</td>
<td>0.1000</td>
</tr>
</tbody>
</table>

- **MAXCUT** = maximum number of times that the stepsize in path parameter is allowed to be reduced per step along the path in order to obtain a converging Newton-iteration
- **CUTFAC** = factor with which the stepsize is multiplied if the Newton-iteration did not converge in order to reduce the stepsize \( 0 < \text{CUTFAC} < 1 \); usually set equal to 0.5
- **OVERSH** = overshoot factor with which the stepsize in path parameter is multiplied in case it was by accident chosen such that the corresponding solution point is close to a bifurcation point in which case the matrix in the Newton-iteration process became nearly singular (set **OVERSH** > 1 to move the to be computed solution point passed the bifurcation point)

<table>
<thead>
<tr>
<th>MAXCUT</th>
<th>CUTFAC</th>
<th>OVERSH</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>0.50</td>
<td>1.10</td>
</tr>
</tbody>
</table>
appendix E

IBIMAX = maximum number of regula falsi iteration steps that are allowed for finding a bifurcation or limit point per bifurcation or limit point that must be computed.

EPS = singularity threshold for matrices. As soon as all the elements of the first column of a still to be solved submatrix in the LU decomposition become in absolute value smaller than EPS, then the matrix on which the LU-decomposition is executed is considered to be singular.

TOLETA = offset from zero for the path parameter change when calculating bifurcation and limit points. As soon as the absolute correction in the path parameter becomes smaller than TOLETA when calculating a bifurcation point this point is considered to be found (TOLETA > 0).

TOLDK = offset from zero for the determinant of matrix \( K \) when computing limit points. As soon as the absolute value of the determinant of matrix \( K \) is smaller than TOLDK it is considered to be zero. When calculating a limit point the solution \( I \) is accepted as limit point if the determinant of matrix \( K \) at this solution is in absolute value smaller than TOLDK and the absolute change in path parameter is smaller than TOLETA (TOLDK > 0).

<table>
<thead>
<tr>
<th>IBIMAX</th>
<th>EPS</th>
<th>TOLETA</th>
<th>TOLDK</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>1E-07</td>
<td>1E-7</td>
<td>1E-7</td>
</tr>
</tbody>
</table>

EPSEIG = tolerance in the eigenvalue calculated with the inverse power iteration. This iteration is used when computing tangents to secondary branches at bifurcation points. A converged eigenvalue and eigenvector are found with the inverse power iteration as soon as the error in the eigenvalue becomes smaller than EPSEIG.

ITRMAX = maximum number of inverse power iteration steps that are allowed for finding a converged eigenvalue when computing the tangent to a secondary branch at a bifurcation point.

<table>
<thead>
<tr>
<th>EPSEIG</th>
<th>ITRMAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>1E-6</td>
<td>100</td>
</tr>
</tbody>
</table>

NEWTON ITERATION PARAMETERS:

IFULLN = type of Newton iteration:
- 0 -> STANDARD Newton-iteration process in which the matrix is assembled and factorized at every new Newton-iterate
- not 0 -> MODIFIED Newton-iteration process in which the matrix is assembled and factorized at the prediction, but not at all the Newton-iterates. The matrix is further only assembled and factorized if divergence is detected after initial convergence, in which case this assembling and factorizing takes place at the last converging Newton-iterate.

INTMAX = maximum number of Newton-iteration steps that are allowed to be taken to obtain a converged solution from its prediction.

MAXFAC = maximum number of times that the matrix of the Newton-iteration equations is allowed to be assembled and factorized in all the corrector calculations in one run with program COHAPI. Together

<table>
<thead>
<tr>
<th>IFULLN</th>
<th>INTMAX</th>
<th>MAXFAC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>40</td>
<td>300</td>
</tr>
</tbody>
</table>
appendix E

TOLRES = absolute tolerance from 0 in the norm of the
residual equations vector f that is accepted
for a converged Newton-iterate (f is right
hand side vector of the system of Newton-
iteration equations and should converge to
the zero vector)
TOLINK = relative tolerance from 0 in the change of the
norm of a Newton-iterate with respect to the
norm of the Newton-iterate itself that is ac-
cepted for a converged Newton-iterate
(becomes the absolute tolerance from 0 if the
Newton-iterate is identically zero)

\begin{align*}
\text{TOLRES} & = 1 \times 10^{-6} \\
\text{TOLINK} & = 1 \times 10^{-6}
\end{align*}

C*************************************************************************
C
C
C
C IMPERFECTION AMPLITUDES : (not read in if ISTART=1 or 2)
C*************************************************************************
C
C IFACCH = flag for choosing which imperfection form
factors should be taken into account :
\begin{align*}
0 & \rightarrow \text{none} \\
1 & \rightarrow \text{alpha}_L \text{ and } \beta_{\text{a}_L} \\
\text{not } 0 \text{ or } 1 & \rightarrow \text{alpha}_L \text{ and } \beta_{\text{a}_L}
\end{align*}

L
\begin{align*}
\text{where : } \text{alpha}_L & = \text{alpha}_L \\
\beta_{\text{a}_L} & = \beta_{\text{a}_L}
\end{align*}

L

\begin{align*}
\text{IFACCH} & = 2
\end{align*}

C*************************************************************************
C
C IMPERFECTION AMPLITUDE
C*************************************************************************
C
\begin{align*}
1 & \rightarrow 0.1 \\
2 & \rightarrow 0.2
\end{align*}

C*************************************************************************
C
C END of CALCULATION CONTROL FILE of program CODFAT