A Moore-Penrose continuation method based on a Schur complement approach for nonlinear finite element bifurcation problems
S Léger, J Deteix, A Fortin

To cite this version:
S Léger, J Deteix, A Fortin. A Moore-Penrose continuation method based on a Schur complement approach for nonlinear finite element bifurcation problems. Computers and Structures, Elsevier, 2015, 152, pp.173 - 184. 10.1016/j.compstruc.2015.02.003 . hal-01503356

HAL Id: hal-01503356
https://hal.archives-ouvertes.fr/hal-01503356
Submitted on 7 Apr 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License
A Moore–Penrose continuation method based on a Schur complement approach for nonlinear finite element bifurcation problems

S. Léger, J. Deteix, A. Fortin

GIREF, Département de mathématiques et de statistique, Pavillon Vachon, 1045 Avenue de la médecine, Université Laval, Québec G1V 0A6, Canada

Keywords:
- Finite element method
- Moore–Penrose continuation method
- Schur complement
- Bifurcations
- Branch following
- Buckling

Continuation methods have proved to be very powerful tools when solving large finite element problems. However, implementation of these methods often require modifications to the standard finite element method. As a finite element code is already very complex, we would like to implement the continuation method as efficiently as possible. In this paper, we present a new implementation technique based on a Schur complement approach for the Moore–Penrose continuation method. This method facilitates the detection of bifurcation points and also enables branch following. Numerical examples will be presented and analyzed using the proposed approach.

1. Introduction

The finite element simulation of very large deformations of hyperelastic materials is still a challenging problem. These problems are generally driven by a loading parameter and it is often observed that for some values of this parameter, the solution varies extremely rapidly due to geometric and/or material non linearities, often leading to the break down of the solution process.

The typical solution strategy is often based on Newton like methods. The loading parameter is set to a given value, the Jacobian matrix stemming from the finite element discretization is constructed, the corresponding linear system is solved to correct the solution and this is repeated until convergence. The loading parameter is then somehow increased (“by hand”) and the process is repeated until the total load has been imposed on the structure. One crucial step in terms of computational cost is the solution of the linear systems involving the Jacobian matrix. For moderate size problems, an LU decomposition can be used but for very large problems, problem specific preconditioned iterative solvers must be designed and implemented.

In some situations, following the solution branch by increasing heuristically the loading parameter becomes extremely difficult and results in the divergence of the process. This is clearly the case in the neighborhood of limit points or bifurcation points where the Jacobian matrix becomes singular. Numerical continuation methods have proved to be a very powerful tool when dealing with these kinds of problems. They can be roughly divided into two main branches, predictor–corrector methods and piecewise linear methods (see [2]). Both types of methods share many common features and can be numerically implemented in similar ways. However, predictor–corrector methods generally perform best when high accuracy is needed and are thus frequently used in practice. Many different continuation methods exist (see for example [14,11,24,25]), but in this paper, only the Moore–Penrose (also known as Gauss–Newton) continuation method (see [2,13,12]), which is a predictor–corrector method, will be considered.

Nonlinear structural analysis leads invariably to collapse and buckling analysis [7,31]. Buckling is a mathematical instability which is characterized by a sudden failure of the structure when critical loads are reached. The value of these critical loads, which are associated with bifurcation points, are quantities of interests [29,21,30] and correspond to eigenvalues of the system. Detecting such points and following the different solution branches passed these critical loads are thus important and useful to better understand the physical properties of the problem we are solving. The analysis of structural instabilities, which includes the detection of bifurcation points, is generally based on a continuation method (see [11,6,14,18,27,1,22,28]). The implementation of the continuation method presented in this paper will therefore offer an efficient and simpler approach for these analyses.
In the Moore–Penrose method, the loading parameter becomes an additional unknown of the problem allowing a more precise control of the deformation, increasing its speed whenever possible and slowing it in some critical regions. This is done by modifying the linear system in the Newton method to take care of the additional unknown. This therefore requires a potentially important modification of an existing finite element code. In particular, new iterative solvers must be implemented to solve these increased linear systems.

As many of these problems have thousands or even millions of degrees of freedom, the continuation method should be implemented as efficiently as possible in order to reduce the already high computational cost. In order to achieve this goal, information that has already been calculated for the finite element method should be used whenever possible. This is precisely the main objective of this work. We will present a generalization of the Moore–Penrose continuation method that uses only information already available in a standard finite element code. In particular, all linear systems that have to be solved are based on the classical Jacobian matrix of the problem. Therefore, if an LU decomposition of this matrix is available, it can be used directly and if a specific iterative matrix is available, it can be used directly and if a specific iterative method was used, it can still be used without any modification to follow the solution curve, detect bifurcations and possibly follow various solution branches.

This paper thus presents a new implementation technique for the Moore–Penrose continuation method when applied in a finite element context. This new approach, which takes advantage of information already available, also facilitates the detection of bifurcation points. Section 2 briefly recalls the Moore–Penrose continuation method as well as the classical numerical implementation of this method. Section 3 is devoted to the details of the new implementation technique. Section 4 explains how the approach can be used to easily detect bifurcation points and finally, Section 5 is devoted to the validation of the proposed algorithm. In particular, a study of the classical elastic beam buckling problem will be presented and analyzed.

2. Moore–Penrose continuation method

The finite element discretization of nonlinear elasticity problems leads to nonlinear problems of the form:

\[ F(u, \lambda) = F(x) = 0 \]

where \( u \) consists of the degrees of freedom. In general, \( u \) is the displacement field, but in the case of mixed formulations, \( u \) consists of the degrees of freedom for the displacement and the pressure. In most situations, a loading parameter \( \lambda \), corresponding to either external forces or prescribed displacements or both, implicitly drives the deformation. For highly nonlinear problems, applying the desired loading will frequently cause a breakdown of the numerical method used to solve this problem. Therefore it is standard practice to gradually increase the value of the parameter until the desired loading, \( \lambda_{\text{max}} \), is attained. Typically, a Newton method is used to solve the nonlinear system. The following steps are therefore repeated until convergence is achieved:

1. Given an initial estimate \( u^0 \) and a fixed load \( \lambda \).
2. Solve the linear system:
   \[ F_u(u^k) \delta^k_u = F(u^k) \]
3. Update the solution:
   \[ u^{k+1} = u^k - \delta^k_u \]

where \( F_u \) is the Jacobian matrix of \( F \) and \( \delta^k_u \) is the correction vector. The loading parameter \( \lambda \) is then increased by a certain quantity and the process is repeated until the total load \( \lambda_{\text{max}} \) has been imposed. The increment in \( \lambda \) is performed more or less heuristically, depending on the convergence of the Newton method. This is highly inefficient and may result in very small increments or even divergence of the algorithm in the neighborhood of turning points or bifurcation points on the solution curve.

A more efficient approach is to use continuation methods where the loading parameter \( \lambda \) is explicitly introduced in the system of nonlinear equations which is now expressed as:

\[ F(u, \lambda) = F(x) = 0 \]

with \( F \) a smooth function of \( \mathbb{R}^{n+1} \) into \( \mathbb{R}^n \). The major difference is that the vector of unknowns, \( x \), now consists of the displacement plus the loading parameter. Starting from a point \( x(0) \in \mathbb{R}^{n+1} \) satisfying \( F(x(0)) = 0 \), and given a vector \( v(0) \) tangent to the solution curve at this point, the goal is to follow the curve up to the point where \( \lambda = \lambda_{\text{max}} \). It is then natural to try to follow the solution curve by first making a prediction step of length \( h_i \) in the tangential direction:

\[ x^i = x(0) + h_i v(0) \]

As \( x^i \) is not likely to be a solution of Eq. (1), the Newton method can be used to obtain the next point \( x^{i+1} \in \mathbb{R}^{n+1} \) on the solution curve. By linearizing Eq. (1) around the prediction point \( x^i = [u^i \; \lambda_i]^T \), we can solve:

\[ F_u(x^i) \delta^i_x + F_\lambda(x^i) \delta^i_\lambda = F(x^i) \]

which leads to a rectangular linear system of the form:

\[ A(x^i) \delta_x = F(x^i) \]

with \( A(x^i) = [F_u(x^i) \; F_\lambda(x^i)] \), a rectangular matrix of dimension \( n \times (n+1) \) and \( \delta_x = [\delta^i_x \; \delta^i_\lambda]^T \) a correction vector of dimension \( (n+1) \times 1 \).

Assuming that the rows of \( A(x^i) \) are linearly independent (i.e. \( A(x^i) \) has full rank), which is generally the case since \( F_u(x^i) \) is the Jacobian matrix of the standard finite element method and is therefore invertible, solution of system (3) is given by:

\[ \delta_x = A^+(x^i) F(x^i) \]

where \( A^+ \) is the Moore–Penrose pseudoinverse of matrix \( A \) defined by \( A^+ = A^T (AA^T)^{-1} \). This therefore leads to the following generalization of the Newton method, which is also called the Moore–Penrose continuation method:

- For \( k = 0, 1, 2, \ldots, k_{\text{max}} \)
  1. Calculate the Moore–Penrose correction:
     \[ \delta^k_x = A^+(x^k) F(x^k) \]
  2. Update the solution vector:
     \[ x^{k+1} = x^k - \delta^k_x \]
  3. If \( \|F(x^k)\| \leq \varepsilon_F \) and \( \|x^{k+1} - x^k\| \leq \varepsilon_x \), convergence attained:
     \[ x^{(k+1)} = x^{k+1} \]

We note that \( k_{\text{max}} \) represents the maximum number of iterations allowed while \( \varepsilon_F \) and \( \varepsilon_x \) are the desired tolerances on \( F \) and \( x \) respectively.

To obtain the correction \( \delta^k_x \) in Eq. (4), the Moore–Penrose pseudoinverse \( A^+ \) of the matrix \( A(x^k) = [F_u(x^k) \; F_\lambda(x^k)] \) is needed. The
explicit computation of this matrix is extremely costly in realistic situations where the matrix $A$ is large. It is however easily seen that $\delta^k_z$ is also a solution of the following system:

$$
\begin{align*}
A(X^k)\delta^k_x &= F(X^k) \\
L^T\delta^k_x &= 0
\end{align*}
$$

(5)

where $L^T$ is any vector in the kernel of the rectangular matrix $A(X^k)$. This can easily be verified by replacing $\delta^k_z$ (i.e. Eq. (4)) in the system.

The major difficulty that arises with this approach however is the construction of vector $L^T$ as it must be in the kernel of $A(X^k)$. In an iterative process, a natural way to overcome this difficulty is to approximate vector $L^T$ by a vector $V^k$ in the kernel of $A(X^{k-1})$ instead of $A(X^k)$. With this approach, an initial approximation $V^0$ is needed and will be chosen as the tangent vector $v^0$. At each iteration $k$, the following system will therefore be solved:

$$
\begin{align*}
A(X^{k-1})V^k &= 0, \\
V^k &= v^0
\end{align*}
$$

(7)

As proposed in [13], we can replace the preceding algorithm by the following one which splits the solution of system (6) (including the construction of the vector $V^k$) into two linear systems sharing the same matrix. We note that in the following, $I^k$ represents a correction on the tangent vector.$$

- X^k = x^{(i)} + \alpha v^0.
- V^k = u^0.
- I^k = \alpha.

1. Solve the linear system:

$$
\begin{align*}
A(X^k)\delta^k_x &= F(X^k) \\
V^k \delta^k_x &= 0
\end{align*}
$$

(8)

2. Solve the linear system:

$$
\begin{align*}
A(X^k)I^k &= A(X^k)V^k \\
V^k I^k &= 0
\end{align*}
$$

(9)

3. Update the solution vector:

$$
X^{k+1} = X^k - \delta^k_x
$$

4. Update the tangent vector:

$$
Z = V^k - I^k
$$

5. Normalization of the tangent vector:

$$
V^{k+1} = \frac{Z}{\|Z\|}
$$

6. If $\|F(X^k)\| < \varepsilon_F$ and $\|X^{k+1} - X^k\| < \varepsilon$, convergence attained:

$$
X^{k+1} = x^{(i)}, \\
v^{(i+1)} = V^{k+1}
$$

From there, we follow the direction perpendicular to $V^0$ to reach $X^i$. Defining $V^1$ as a vector tangential to $A(X^0)$, we then follow the direction orthogonal to $V^1$, and so forth.

To obtain the next point on the solution curve, two systems need to be solved at each iteration. In a finite element context, especially for problems with a high number of degrees of freedom, constructing and solving these two systems can be costly and should be avoided. In the next section, we will show that the explicit construction of these systems, which in matrix form are given by:

$$
\begin{align*}
F_u(X^k) & F_z(X^k) \begin{pmatrix} \delta^k_x \\ \delta^k_z \end{pmatrix} = \begin{pmatrix} 0 \end{pmatrix} \\
V^k_u & V^k_z = A(X^k)V^k
\end{align*}
$$

(10)

and

$$
\begin{align*}
F_u(X^k) & F_z(X^k) \begin{pmatrix} T^k_u \\ T^k_z \end{pmatrix} = A(X^k)V^k
\end{align*}
$$

(11)

is not necessary. We note that in systems (10) and (11), vector $V^k$ was decomposed into its components in the $u$ and $z$ directions (i.e. $V^k = [V^k_u \ V^k_z]^T$). The correction vectors were also decomposed in the same way (i.e. $\delta^k = [\delta^k_u \ \delta^k_z]^T$ and $T^k = [T^k_u \ T^k_z]^T$).

3. Numerical implementation based on a Schur complement approach

3.1. Schur complement

The Schur complement of the block $M_{11}$ of a matrix

$$
M = \begin{pmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{pmatrix}
$$

dimension $(p + q) \times (p + q)$ is defined by

$$
M_{22} - M_{21}M_{11}^{-1}M_{12}
$$

and arises naturally when solving a linear system of equations of the form:

$$
\begin{align*}
M_{11}x + M_{12}y &= a \\
M_{21}x + M_{22}y &= b
\end{align*}
$$

(12)

where $x$ and $a$ are column vectors of dimension $p$ and $y$ and $b$ are column vectors of dimension $q$. As for the blocks $M_{11}, M_{12}, M_{21},$ and $M_{22}$, they are matrices of dimension $p \times p, p \times q, q \times p,$ and $q \times q$, respectively. Assuming $M_{11}$ is nonsingular the Schur complement is invertible (see [23]) and we get

$$
[M_{22} - M_{21}M_{11}^{-1}M_{12}]y = b - M_{21}M_{11}^{-1}a \\
x = M_{11}^{-1}a - M_{11}^{-1}M_{12}y
$$

(13)

The Schur complement approach presented here is a mathematical tool used in various circumstances in structural analysis or in fluid dynamics (for instance, the static condensation as proposed

![Fig. 1. Correction algorithm for the Moore–Penrose continuation method.](image-url)
by Bathe [7] uses the Schur complement approach on slave nodes. Since the Schur complement is a matrix of dimension \( q \times q \), the Schur complement approach thus reduces the problem of inverting a \((p + q) \times (p + q)\) matrix (matrix \( M \)) to that of inverting two matrices, a \( p \times p \) matrix (matrix \( M_{11} \)) and a \( q \times q \) matrix (the Schur complement).

In the case where \( q = 1 \), this approach is very advantageous since the Schur complement, which we will denote by \( \beta \), is a scalar. The solution of system (12) is therefore easily obtained.

1. Solve \( M_{11} \Delta = a \).
2. Solve \( M_{11} W = M_{12} \).
3. Calculate the Schur complement \( \beta = M_{22} - M_{21} W \).
4. Calculate \( y = \frac{1}{2} (b - M_{21} \Delta) \).
5. Calculate \( x = \Delta - W y \).

The solution of system (12) is therefore obtained by solving two systems, both of which involve the same matrix \( M_{11} \).

For the Moore–Penrose algorithm, we have two systems of the form (12) to solve (see systems (10) and (11)), where the matrix \( M \) is the same for both systems and is given by:

\[
\begin{pmatrix} F_u(X^k) & F_f(X^k) \\ V_u^T & V_f^T \end{pmatrix}
\]

\( V_k \) is a block of dimension \( 1 \times 1 \) and matrix \( M_{11} \) is given by \( F_u(X^k) \) which is nothing but the classical Jacobian matrix in a classical Newton method. Using a Schur complement approach to solve these two systems would therefore necessitate solving 4 linear systems involving \( F_u(X^k) \). Fortunately, the specific form of systems (10) and (11) allows some simplifications as we will see in the next section.

3.2. Schur complement approach applied to the Moore–Penrose method

Applying the Schur complement approach to system (10) leads to the following steps:

1. Solve \( F_u(X^k)A_{u,k} = F(X^k) \).
2. Solve \( F_u(X^k)W_{u,k} = F_f(X^k) \).
3. \( \beta^k = V_k - V_u \cdot W_{u,k} \).
4. \( \delta^k_u = -\beta^k V_u \cdot A_{u,k} \).
5. \( \delta^k_u = A_{u,k} - W_{u,k} \delta^k_u \).

at each iteration \( k \).

Concerning system (11), we note that it is not necessary to redo all the above steps. Indeed, the computation of \( W_{u,k} \) and \( \beta^k \) is exactly the same for both systems. Moreover, step 1 gives:

\[
\begin{align*}
\mathbf{A}_{u,k} &= F_u(X^k)^{-1} A'(X^k) V = F_u(X^k)^{-1} F_u(X^k) V_u + F_f(X^k) V_f \\
&= V_u^T + W_u V_f
\end{align*}
\]

and from step 4 and 5 we get

\[
\begin{align*}
T_u^k &= -\frac{1}{\beta^k} [V_u^T \cdot V_u + V_u \cdot W_u V_f] \\
T_f^k &= V_f + W_u \cdot [V_f - T_u^k]
\end{align*}
\]

The following algorithm summarizes the implementation technique of the Moore–Penrose continuation method when using a Schur complement approach.

- \( X^0 = x^{(0)} + h \nu^{(0)} \).
- \( V^0 = \nu^{(0)} \).
- For \( k = 0, 1, 2, \ldots, k_{\text{max}} \).
  1. Solve to obtain the standard Newton correction (for a fixed value of \( \lambda \)):
     - \( F_u(X^k)A_{u,k} = F(X^k) \)
     - \( F_u(X^k)W_{u,k} = F_f(X^k) \)
     - \( \beta^k = V_k - V_u \cdot W_{u,k} \)
     - \( \delta^k_u = -\beta^k V_u \cdot A_{u,k} \)
     - \( \delta^k_u = A_{u,k} - W_{u,k} \delta^k_u \)
  2. Calculate the correction for the \( x \) component of the tangent vector:
     - \( T_u^k = -\frac{1}{\beta^k} [V_u \cdot V_u + V_u \cdot W_u V_f] \)
  3. Update the solution vector:
     - \( X^{k+1} = X^k - \delta^k \)
  4. Update the tangent vector:
     - \( Z = V^k - T^k \)
  5. Normalization of the tangent vector:
     - \( V^{k+1} = \frac{Z}{\|Z\|} \)
  6. If \( \|F(X^k)\| \leq \varepsilon_F \) and \( \|X^{k+1} - X^k\| \leq \varepsilon_u \), convergence attained:
     - \( X^{(i+1)} = X^{k+1}, \quad V^{(i+1)} = V^{k+1} \)

3.2.1. Important remarks

- Both systems needed for the Moore–Penrose continuation method can thus be solved easily using the Schur complement approach. Furthermore, one interesting thing to notice is that \( \delta^k_u \), the Moore–Penrose correction for the \( N \) degrees of freedom at iteration \( k \), is calculated using the formula \( \delta^k_u = A_{u,k} - W_{u,k} \delta^k_u \), where \( A_{u,k} \) is the correction obtained from the standard Newton method. The Moore–Penrose corrections can thus simply be obtained by modifying the corrections obtained without continuation.
- This also means that the Moore–Penrose continuation algorithm can be incorporated numerically in the finite element code by simply adding a postcondition to the Newton method solver. This postcondition is added to modify the correction obtained by the Newton method in order to obtain the desired Moore–Penrose correction. This postcondition can be activated
when necessary and does not require the construction of matrices that are not already constructed for the finite element method.

- When using a direct solver to obtain the standard Newton correction \( \Delta u^k \), solving the additional system (to obtain \( W^k v^k \)) is not very costly since the \( LU \) decomposition of the Jacobian matrix is already known.

- When an iterative method is used (e.g. in the case of very large finite element 3D problems) to obtain the correction \( \Delta u^k \), this approach is also advantageous as the same iterative method can be used to solve the additional system and obtain \( W^k v^k \). Iterative methods are very sensitive, and there is no guarantee that the iterative method developed to obtain \( \Delta u^k \) would enable us to solve the augmented systems (10) and (11).

- As the systems needed for most predictor–corrector continuation methods have the same form as the ones needed for the Moore–Penrose continuation method, we note that a similar approach could be used for other continuation methods as well.

3.3. Numerical algorithm

The following algorithm describes how the Moore–Penrose continuation method (based on a Schur complement approach) can be incorporated numerically in a finite element code. We will assume that the load is applied in terms of a parameter which varies from 0 to 1. At \( \lambda = 1 \), the desired load will have been applied completely and the solution obtained will be the desired one. If the algorithm converges for \( 0 < \lambda < 1 \), the solution will be considered as an intermediate solution and the algorithm will continue until convergence is attained for \( \lambda = 1 \) (if possible).

The Moore–Penrose continuation is added in the nonlinear loop that solves the system given by \( F(u, \lambda) = 0 \) and can be described by Algorithm 1. We note that the algorithm assumes that the variational formulation is solved using the Newton method and that \( X^k = [u^k \; \lambda^k]^\top \).

Algorithm 1. Solve \( F(u, \lambda) = 0 \)

Require: \( x^{(0)} = [u^{(0)} \; \lambda^{(0)}]^\top \), \( e, Tol_u, Tol_\lambda \), for \( k = 0, 1, \ldots, k_{\text{max}} \) do

Solve \( F'\lambda(X^k) d\lambda_{\lambda} = F(X^k) \) (standard correction for the Newton method for fixed \( \lambda \))

if \( k = 0 \) then

\( V^k_u = A_{\lambda u} + e \lambda \); \( V^k_{\lambda} = e \lambda \) (initial approximation of the tangent)

\( V^k = V^k_u ||V^k\| \)

\( u^k = u^{(0)} + h V^k_u \); \( \lambda^k = \lambda^{(0)} + h V^k_\lambda \) (construction of \( X^0 \))

end if

\( F'\lambda(X^k) = (F(u^k, \lambda^k) + e \lambda - F(u^k, \lambda^{(0)}))/e \lambda \)

Solve \( F'\lambda(X^k) W^k_u = F'(X^k) \) (Solution of system (10))

\( \beta^k = V^k_u - V^k_{\lambda} W^k_u \)

\( \delta^k_u = -\frac{1}{\beta^k} V^k_u - A_{\lambda u} \)

\( \delta^k_{\lambda} = A_{\lambda u} - W^k_u \beta^k \)

\( T^k_u = -\frac{1}{\beta^k} V^k_{\lambda} V^k_u + V^k_{\lambda} W^k_u V^k_{\lambda} \) (Solution of system (11))

\( T^k_u = V^k_u + W^k_u (V^k_{\lambda} - T^k_\lambda) \)

end for

In Algorithm 1, a first order forward finite difference scheme (in regards to \( \lambda \)) is used to approximate the initial tangent \( V^0 \). The step size used for this finite difference scheme is denoted by \( e \), while \( Tol_u \) and \( Tol_\lambda \) are the desired tolerances for \( u \) and \( \lambda \) respectively. We also note that the derivative of \( F \) with respect to \( \lambda \) is also calculated (approximated) with a first order forward finite difference scheme.

Using this approach, we therefore only exit the nonlinear solver loop once the desired load has been attained at \( \lambda = 1 \). In an updated Lagrangian context (see [7,20]), since the geometry is only updated once the convergence is attained, this means that the desired load should not be imposed for the complete simulation, but only for one updated Lagrangian step. Obviously, if the imposed load is too high, the algorithm will not be able to converge to \( \lambda = 1 \) without remeshing. In this situation, a mechanism is therefore needed to exit the loop and update the mesh at the largest load attained. We note that this mechanism, which enables a user to exit the loop with either the last converged solution or the second last converged solution (if we want a state of the solution that enables convergence for at least one more step), was added to Algorithm 1.

Nonconvergence before \( \lambda = 1 \) could also mean that the step size \( h \) is too large. At each converged intermediate solution, the algorithm always tries to find a next point on the solution curve. To find this next point, the first step always consists in completing a prediction step of step size \( h \) in the tangent direction. If \( h \) is too large, the prediction point will be too far from the solution curve and convergence problems will be encountered. A mechanism therefore also has to be added to the algorithm in order to modify the value of \( h \) if needed. This mechanism must verify, at each intermediate solution, if the value of \( h \) should be increased or reduced to obtain the next point. In general, the number of iterations needed for convergence is the criterion used to determine if the value of \( h \) should be modified or not, and if so, in which way. This mechanism should also enable a restart from the last converged solution (with a reduced value of \( h \)) when the algorithm does not converge (and this before exiting the loop completely). We note that these two mechanisms were implemented numerically in our finite element code, but were not mentioned in Algorithm 1 in order to simplify the reading.
4. Detection of bifurcations

The approach proposed in Section 3 for the implementation of the Moore–Penrose continuation method facilitates the detection of bifurcation points. In fact, as we will see, most of the informations needed for the detection of bifurcation points are already known when using a Schur complement approach to solve the nonlinear system of equations, which means that important information about the problem can be obtained at a low computational cost. In this work, we will concentrate on the detection of simple bifurcation points only.

Let $x \in \mathbb{R}^{N+1}$ be a simple bifurcation point of equation $F(x) = 0$ (see [18] for a formal definition). We thus have that $F(x) = 0$ and that two distinct continuous solutions passes through $x$ (as illustrated in Fig. 2). To detect a simple bifurcation point, we can use the following result.

**Lemma 4.1** (Georg [15]). Let $s \mapsto x(s)$ be a parametrization of a solution branch such that $\|x(s)\|_2 = 1$. The bifurcation point can thus be expressed as $x = x(s)$. By choosing a tangent vector $v$ such that $v^T x(s) > 0$, the determinant of the matrix $J(x)$, which is defined by $J(x) = \begin{pmatrix} A \\ v^T \end{pmatrix}$, changes sign at $s = \bar{s}$. We note that an approximation of the tangent vector $v$ suffices to detect the change of sign.

Obviously, calculating the determinant of this matrix is not desirable. The following result gives a more practical solution.

**Theorem 4.2** (Georg [15]). Let $B \neq \text{Im}(J(x))$ and $C \neq \text{Im}(J(x)^T)$. By defining $\tau(x)$ via the system:

$$
\begin{pmatrix} J(x) & B \\ C^T & 0 \end{pmatrix} \begin{pmatrix} Y(x) \\ \tau(x) \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix},
$$

then $\tau(x(s))$ changes sign at $s = \bar{s}$.

**Proof.** From Cramer’s rule, we have that

$$
\tau(x(s)) = \frac{\det J(x(s))}{\det \begin{pmatrix} J(x(s)) & B \\ C^T & 0 \end{pmatrix}}
$$

and from Lemma 4.1, the determinant of the numerator changes sign at the bifurcation point. It can then be shown that the determinant of the denominator does not vanish and therefore does not change sign at $s(\bar{x})$.

The detection of bifurcations can therefore be made by solving an additional system (i.e. system (14)), which is simply an augmented system of the system used for the Moore–Penrose continuation. Clearly, for $s \neq \bar{s}$, the solution of this system is

$$
J(x)G = B \quad \tau = -(C \cdot G)^{-1} \quad Y = -\tau G
$$

As for the vectors $B$ and $C$, we note that these can be chosen randomly due to the high probability that they will satisfy the conditions stated above.

In a finite element context (with continuation), the system to solve can be written as:

$$
\begin{pmatrix} F'_u & F'_s & B_u \\ V'_u & V'_s & B_s \\ C'_u & C'_s & 0 \end{pmatrix} \begin{pmatrix} Y_u \\ Y_s \\ \tau \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}
$$

where the vectors $B, C$ and $Y$ have been decomposed into their components in the $u$ and $s$ directions respectively (i.e. $B = [B_u B_s]$, $C = [C_u C_s]$ and $Y = [Y_u Y_s]$).

Using the Schur complement approach to find $G = [G_u G_s]^{-1}$ in (16), we get

$$
G_u = \frac{1}{P_u} (B_u - V_u \cdot P_u) \quad G_s = P_s - G_u W_u
$$

with $W_u = F^{-1}_u$ and $P_u = F^{-1}_u B_u$. Finally, by replacing expressions (18) in (16), we obtain:

$$
\tau = -(C_u G_u + C_s \cdot G_s)^{-1}
$$

At a given iteration $k$, the solution of system (17) is thus given by:

- $\beta^k = V^k_u - V^k_s \cdot W^k_u$ (Schur complement)
- $G^k_u = \frac{1}{P^k_u} (B_u - V^k_u \cdot P^k_u)$
- $C^k_u = P^k_u - G^k_u W^k_u$
- $\tau^k = -(C^k_u G^k_u)^{-1}$
- $Y^k_u = -\tau^k G^k_u$
- $Y^k_s = -\tau^k G^k_s$

where $P^k_u = F^{-1}_u (X^k)^{-1} B_u$ and $W^k_u = F^{-1}_u (X^k)^{-1} F'_s (X^k)$.

We note that $\beta^k$ and $W^k_u$ are already known and do not need to be recalculated. To obtain the solution, we therefore only need to solve one more linear system with the matrix $F'_s (X^k)$ (to calculate $P^k_u$) as well as compute a few scalar products. Furthermore, if the goal is only to detect bifurcation points, vector $Y$ (i.e. $Y_u$ and $Y_s$) does not need to be calculated (because the bifurcation can be detected with the sign of $\tau$ only). We also note that $\tau$ does not need to be calculated at each iteration $k$, only at a converged intermediate solution. At each converged intermediate solution, the value of $\tau$ is thus calculated and is compared with the value of $\tau$ at the previous converged intermediate solution to detect a possible change of sign.

It is important to comment a little further on the computation of the critical value $\tau$. The expression (15) (or simply the determinant of the Jacobian matrix) can be used for small matrices like those encountered in the study of small dynamical systems to detect bifurcation points. For large systems as those obtained in finite element simulations (when an $LU$ decomposition is used), the computation of these determinants would end up with machine overflows. This is why we use the expression given in Eq. (16). However, while following the solution curve, vector $C$ can become orthogonal to vector $B$ thus producing a singularity in the computation of $\tau$. This is also an indication that the matrix $M(x(s))$, defined by

$$
M(x(s)) = \begin{pmatrix} J(x(s)) & B \\ C^T & 0 \end{pmatrix}
$$

is singular for some value of $s$. The algorithm, as it is now, may detect a change of sign even though $\tau$ is not defined there. To avoid
this situation, a supplementary precaution is added to the algorithm when a change of sign is detected. We simply make another choice for vectors $B$ and $C$ in system (14) in order to verify if these new vectors also generate a change in the sign of $\tau$. To avoid unnecessary computations, vector $B$ is left unchanged in our algorithm for the second verification test, but vector $C$ is chosen as $f^{-1}(x(s_0))B$ where $s_0$ represents the last converged point before the detected point. Making this choice only requires to store in memory the last computed value of vector $G$ and to compute the scalar product of this vector with vector $G$ at the current converged point to see if the sign is positive. If so, it implies that $\tau$ does not change sign with the use of these new vectors and that consequently the detected point is not a real bifurcation point.

As for vector $Y$, it can be used to follow a specific bifurcation branch. In fact, in the case of a simple bifurcation point where only two distinct solution branches intersect, vector $Y$ can be used as a prediction direction (instead of the usual tangent vector) to find a point on the new branch. More specifically, by denoting $\tilde{x}$ an approximation of $x$ (obtained by looking at the sign of $\tau$) and $Y(\tilde{x})$ the first part of the solution of the augmented system (14) used to compute $\tau(\tilde{x})$, one can take $Y(\tilde{x})$ as a prediction direction and complete one continuation step starting from the point $(\tilde{x}, Y(\tilde{x}))$ to obtain a first point on the new branch. Once this first continuation step has been completed, the standard continuation method can be used to follow the new branch.

5. Validation of the approach

5.1. Elastic beam problem

To verify that the bifurcation detection algorithm works well, we can first consider the classical elastic beam buckling problem. As the bifurcation modes for this problem are well known, it is

![Fig. 3.](Image)

**Fig. 3.** Evolution of the deformation of the beam (2D) in terms of the applied load.

![Fig. 4.](Image)

**Fig. 4.** Elastic beam (2D): detected bifurcation points.

![Table 1](Image)

**Table 1**

| $v_1$ | $v_2$ | $v_1$ | $v_2$ | $v_1$ | $v_2$ |
|---|---|---|---|---|---|
| $0.2732 \times 10^{-6}$ | $-0.1409 \times 10^{-6}$ | $-0.2733 \times 10^{-6}$ | $-0.1415 \times 10^{-6}$ | $-0.5539 \times 10^{-6}$ | $-0.5531 \times 10^{-6}$ |
| $-0.5539 \times 10^{-6}$ | $-0.5540 \times 10^{-6}$ | $-0.5539 \times 10^{-6}$ | $-0.5541 \times 10^{-6}$ | $-0.5972 \times 10^{-6}$ | $-0.5975 \times 10^{-6}$ |
| $-0.5981 \times 10^{-6}$ | $-0.5981 \times 10^{-6}$ | $-0.5981 \times 10^{-6}$ | $-0.5982 \times 10^{-6}$ | $-0.6409 \times 10^{-6}$ | $-0.6419 \times 10^{-6}$ |
| $-0.6428 \times 10^{-6}$ | $-0.6425 \times 10^{-6}$ | $-0.6426 \times 10^{-6}$ | $-0.6425 \times 10^{-6}$ | $-0.6426 \times 10^{-6}$ | $-0.6425 \times 10^{-6}$ |
| $-0.6427 \times 10^{-6}$ | $-0.6427 \times 10^{-6}$ | $-0.6433 \times 10^{-6}$ | $-0.6432 \times 10^{-6}$ | $-0.6437 \times 10^{-6}$ | $-0.6437 \times 10^{-6}$ |
| $-0.6440 \times 10^{-6}$ | $-0.6442 \times 10^{-6}$ | $-0.6440 \times 10^{-6}$ | $-0.6442 \times 10^{-6}$ | $-0.6472 \times 10^{-6}$ | $-0.6471 \times 10^{-6}$ |

The values are shown in bold in order to emphasize the change in sign that occurs at this point during the simulation.
Let us consider a beam of dimension \( \frac{1}{C^2} \), meshed as in Fig. 3(a), on which an imposed vertical displacement \( v \) will be applied on its top edge. We note that the top and bottom edges are free to slide in the horizontal direction, with the exception of the middle point on the bottom edge which is fixed in order to prevent rigid body movement. A Mooney–Rivlin model with parameters \( k = 100; c_1 = 1.5 \) and \( c_2 = 0.5 \) will be used to describe the elastic material and \( \frac{P_2 - P_1}{C_0} \) elements (quadratic discretization for the displacement and linear for the pressure also known as the Taylor–Hood element [8]) will be used for the computation. The unknown vector \( u \) in this case thus consists of the degrees of freedom for the displacement as well as for the pressure. We also refer to Chamberland et al. [10] for a detailed presentation of the discretization and a justification of this choice based on accuracy and efficiency.

5.1.1. Antisymmetric modes

Many bifurcation modes exist for the elastic beam problem (Hill and Hutchinson [16], Bardet [5], Triantafyllidis et al. [26]). On the main solution branch, the beam should remain straight and continue to widen with increasing load. This is precisely what is observed in Fig. 3. Using the procedure described in Section 4, we have been able to detect different bifurcations and this is illustrated in Fig. 4.

As an additional precaution, we present in Table 1 the six smallest eigenvalues for the different matrices used during the computation in the region of \( v = 0.0329 \) (the second point in Fig. 4). We see clearly that the Jacobian matrix \( F_u \) has an eigenvalue that passes through zero (which confirms the presence of a bifurcation point). As expected, the same thing is also true for the Moore–Penrose matrix \( J(x) \). The matrix \( M \) is in turn invertible.

Now, by using the approach described in Section 4 to change branches, we are able to bifurcate towards the different bifurcation modes (which are all antisymmetric). Fig. 5 illustrates the different solutions obtained while Fig. 6 shows the bifurcation diagram for these same solutions. We note that the curve that passes through the detected bifurcation points (indicated by the symbol (c) in the figure) represents the fundamental branch.
5.1.2. Symmetric modes

Symmetric bifurcation modes also exist for the elastic beam problem. As described in Brun et al. [9], these bifurcations become possible after the accumulation point of the antisymmetric modes (which defines a surface instability). However, it is important to mention that these modes are generally difficult to obtain numerically with asymmetrical meshes. We can however hope to obtain these modes by using a mesh completely symmetric from the start. We thus used a regular symmetric mesh (not illustrated) composed of 4320 quadrilaterals \( 20 \times 216 \) and 4557 nodes and we have been able to detect the different antisymmetric modes (at approximately the same values of \( \nu \) than for the asymmetric mesh of Fig. 3(a)). We have also obtained the surface instability at an approximate load of \( \nu = 0.451 \) (which is characterized by a very wavy deformation mode when we approach it, as is illustrated in Fig. 7) as well as the symmetric modes afterwards. However, as the symmetric bifurcation modes are often very close to one another, they become very difficult to detect (because if we move too quickly along the solution curve, the algorithm will not be able to detect the two consecutive changes in the sign of \( s \)). To detect these modes, it is thus preferable to first look at the appearance of the vector \( Y \) at the different intermediate solutions in order to see the different mode changes appear. Afterwards, the different bifurcations can be detected more precisely by moving slowly on the solution curve in the regions where the different mode changes occur.

Fig. 8 illustrates some of the symmetric modes for the elastic beam problem and indicates the approximative load at which these different bifurcations occur. As we can see, the symmetric bifurcation modes decrease with increasing load, which is different from the antisymmetric modes where the opposite is true. For the antisymmetric modes, the greater the load, the wavier the bifurcation modes are. For the symmetric modes, the wavier modes occur at smaller loads. We note that this phenomenon was also observed in the numerical results of [9]. We also note that Fig. 8 illustrates the appearance of vector \( Y \) at the different bifurcation points and not the bifurcated solutions (i.e. solutions that we would obtain if we were to follow the different branches). However, to show that we can also follow the different symmetric branches, we have used the approach described in Section 4 to bifurcate towards the first symmetric mode which takes place at a load of \( \nu = 0.747 \). Fig. 9 shows the solution obtained.

5.1.3. Computational time

To show the efficiency of the proposed algorithm, we have also solved the buckling beam problem with the classical approach described in Section 2. Table 2 compares the computational times when solved using the asymmetrical mesh of Fig. 3(a). For this test, a total load of \( \nu = 0.2 \) was imposed (while staying on the main solution branch) and the same level of accuracy was attained with both algorithms. We also note that the detection of bifurcation points algorithm was not activated during the simulation as it had never been implemented in the classical algorithm (due to the computational cost). However, the cost associated with the detection of bifurcation points when using the new algorithm is very low (approximately 0.1 of a second).

5.1.4. Three-dimensional case

We can also validate the approach in the tridimensional case. Let us consider the beam illustrated in Fig. 10(a), which is a 3D generalization of the previous example, and solve using \( Q_2 - Q_1 \) elements. We note that the regular mesh is composed of 1250 hexahedra \( (5 \times 5 \times 50) \) and 6696 nodes.

In the same way as for the 2D case, we illustrate the evolution of the deformation of the beam (on the fundamental branch) in Fig. 10. As for the bifurcations detected by the algorithm, we have illustrated them in Fig. 11. As we can see, the same bifurcations are detected in the 3D case, which is normal since we are solving the same problem but with an extra dimension. We were also able to bifurcate towards the different bifurcation modes, as illustrated in Fig. 12.

![Convergence towards the first symmetric mode: evolution of the deformation. The scale illustrates the \( L^2 \) norm of the total displacement.](image)

**Table 2**

| Comparison of the computational times (classical algorithm vs new algorithm). |
|---------------------------------|---------------------|
| Classical algorithm | New algorithm |
| 52.1 s | 19.3 s |

- The test described in Section 4 to bifurcate towards the first symmetric mode which takes place at a load of \( \nu = 0.747 \). Fig. 9 shows the solution obtained.
5.2. Finite strain incompressible elasticity problem

To test the robustness of the proposed approach, let us now consider the finite strain incompressible elasticity problem recently presented in [3,4] where a square of initial configuration $\Omega^0 = [-1, 1] \times [-1, 1]$, with coordinates $x$ and $y$ and meshed as in Fig. 14(a), is considered. For this problem, the top edge of the configuration will be traction free, while all the other edges will be fixed. The body will be subjected to a vertical uniform body force $F$ and a Mooney–Rivlin model with parameters $k = 40,000,000$, $c_1 = 20$ and $c_2 = 0$ will be used to describe the incompressible elastic material. We note that these parameters correspond to the parameters used in [4] with the value of $k$ chosen to represent a purely incompressible elastic material. $P_2 - P_1$ elements will also be used for the computation.

Without continuation, the standard Newton approach converges to the trivial solution of this problem, which is given by $u = 0$ and $p = -F(1 - y)$, until convergence problems are encountered at a load of approximately $F = 265$. This load corresponds to the critical value detected in [4]. Fig. 13 illustrates the deformation associated with this load. As we can see, the solution seems to have diverged from the fundamental branch (which corresponds to the trivial solution) at this breaking point and seems to indicate the presence of a bifurcation point. Our proposed Moore–Penrose continuation algorithm can then be activated to see if a bifurcation point was encountered and if better results can be attained.

By activating our proposed algorithm, the solution initially converges on the fundamental branch and a bifurcation point is detected at an approximate value of $F = 264$ (which explains the convergence problems of the standard Newton approach). By following the bifurcation branch associated with this point, much larger deformations than with the standard Newton approach can be attained, as can be seen in Fig. 14 where the evolution of the deformation of the configuration over time is shown.

We also note that by staying on the fundamental branch (instead of following the bifurcation branch associated with the bifurcation point at $F = 264$), we were able to detect another

![Fig. 10. Evolution of the deformation of the beam (3D) in terms of the applied load.](image)

![Fig. 11. Elastic beam (3D): bifurcations detected.](image)

![Fig. 12. First antisymmetric bifurcation modes (3D).](image)

![Fig. 13. Finite strain incompressible elasticity problem: final deformed configuration (standard Newton approach).](image)
a postcondition to the Newton method solver. This postcondition modifies the Newton correction accordingly in order to obtain the desired Moore–Penrose correction. This postcondition can be activated when needed without modifying the implementation of the standard finite element method. Detection of bifurcation points and branch following are also simplified with this approach as most of the informations needed are already available. The numerical examples presented illustrated the good performance of this method.

**Acknowledgement**

The authors wish to acknowledge the financial support of the Natural Sciences and Engineering Research Council of Canada (NSERC).

**References**

[1] Abbot JP. An efficient algorithm for the determination of certain bifurcation points. J Comp Appl Math 1978;4:19–27.
[2] Allgower EL, Georg K. An introduction to numerical continuation methods. Springer series in computational mathematics, vol. 13. New York: Springer-Verlag; 1990.
[3] Auricchio F, Beirão da Veiga L, Lovadina C, Reali A. A stability study of some mixed finite elements for large deformation elasticity problems. Comp Meth Appl Mech Eng 2005;194(9–11):1075–92.
[4] Auricchio F, Beirão da Veiga L, Lovadina C, Reali A, Taylor R, Wriggers P. Approximation of incompressible large deformation elastic problems: some unresolved issues. Comput Mech 2013;52(5):1153–67.
[5] Bardet JP. Finite element analysis of plane strain bifurcation within compressible solids. Comp Struct 1990;36(6):993–1007.
[6] Bathe KJ. Dvorkin EN. On the automatic solution of nonlinear finite element equations. Comp Struct 1983;17:871–9.
[7] Bathe KJ. Finite element procedures. NJ: Prentice-Hall; 1996.
[8] Boffi D, Brezzi F, Fortin M. Mixed finite elements and applications. Springer series in computational mathematics, vol. 44. Berlin, Heidelberg: Springer; 2013.
[9] Brun M, Bigoni D, Capuani D. A boundary element technique for incremental, non-linear elasticity. Part ii: bifurcation and shear bands. Comp Meth Appl Mech Eng 2003;192:2481–99.
[10] Chamberlain E, Fortin A, Fortin M. Comparison of the performance of some finite element discretizations for large deformation elasticity problems. Comp Struct 2010;88(11–12):664–73.
[11] Crisfield MA. A fast incremental/iterative solution procedure that handles snap through. Comp Struct 1981;13:55–62.
[12] Deulphard P, Heindl G. Affine invariant convergence theorems for Newton’s method and extensions to related methods. SIAM J Numer Anal 1979;16(1):1–10.
[13] Dhooge A, Govaerts W, Kuznetsov YA. MATCONT: a MATLAB package for numerical bifurcation analysis of ODEs. ACM Trans Math Softw 2003;29(2).
[14] Riks E. The application of Newton’s method to the problem of elastic stability. J Appl Mech 1972;39:1060–6.
[15] Georg K. Matrix-free numerical continuation and bifurcation. Numer Funct Anal Optimiz 2001;22:303–20.
[16] Hill R, Hutchinson JW. Bifurcation phenomena in the plane tension test. J Mech Phys Solids 1975;23:239–64.
[17] Ikeda K, Yamakawa Y, Tsutsumi S. Simulation and interpretation of diffuse shear bands. Int J Numer Meth Eng 2005;63(4):439–64.
[18] Keller HB. Lectures on numerical methods in bifurcation problems. New Delhi: Narosa Publishing House; 1987.
[19] Le Tallec P, Vidrascu M. Une méthode numérique pour les problèmes d’équilibre de corps hyperélastiques compressibles en grandes déformations. Numer Math 1984;43:199–224.
[20] Léger S, Fortin A, Tibirsa C, Fortin M. An actualized Lagrangian formulation with automatic remeshing for large deformation problems. Int J Numer Meth Eng 2014;100(13):1006–30.
[21] Neto LB, Machado RD, Hecke MB. Finite element analysis of buried pipelines subjected to buckling. Int J Model Simul Petrol Indus 2009;3(1):23–33.
[22] Rheinboldt WC. Numerical analysis of continuation methods for nonlinear structural problems. Comp Struct 1981;13:103–13.
[23] Saad Y. Iterative methods for sparse linear systems. 2nd ed. Philadelphia: SIAM; 2003.
[24] Schweizerhof KH, Wriggers P. Consistent linearization for path following methods in nonlinear fe analysis. Comp Meth Appl Mech Eng 1986;59:261–79.
[25] Simo JC, Wriggers P, Schweizerhof KH, Taylor RL. Post-buckling analysis involving inelasticity and unilateral constraints. Int J Numer Meth Eng 1986;23:779–800.
[26] Triantafyllidis N, Scherzinger WM, Huang H-J. Post-bifurcation equilibria in the plane-strain test of a hyperelastic rectangular block. Int J Solids Struct 2007;44(11–12):3700–19.
[27] Wagner W, Wriggers P. A simple method for the calculation of post-critical branches. Eng Comp 1988;5:103–9.
[28] Wriggers P, Simo JC. A general procedure for the direct computation of turning and bifurcation points. Int J Numer Meth Eng 1990;30:155–76.
[29] Zhou Z, Murray DW. Analysis of postbuckling behavior of line pipe subjected to combined loads. Int J Solids Struct 1995;32(20):3015–36.

[30] Zhou Z, Nishida A, Kuwamura H. Applicability of finite element method to collapse analysis of steel connection under compression. Prog Nucl Sci Technol 2011;2:481–5.
[31] Zienkiewicz OC, Taylor RL. The finite element method: basic formulations and linear problems, 4 ed., vol. 1. London: McGraw-Hill; 1989.