GALAXY INTERNATIONAL INTERDISCIPLINARY RESEARCH JOURNAL (GIIRJ)
ISSN (E): 2347-6915
Vol. 11, Issue 06, June (2023)

PROGRAMMING USING SEIDEL AND NEWTON METHODS TO SOLVE CURVILINEAR NODAL STRESS EQUATIONS IN PYTHON

Aktamov Shokhzod
Graduate student of Navoi State Pedagogical Institute

ANNOTATION

The Seidel and Newton methods discussed in this article provide solutions for solving curvilinear nodal stress equations in Python. The Seidel method is an iterative technique that updates nodal displacements sequentially until convergence is achieved, while the Newton method utilizes linearization and solves a linear system of equations to obtain the update in nodal displacements.

These methods are widely used in engineering fields such as structural analysis and finite element analysis to analyze the stress distribution within complex structures. By implementing the Seidel or Newton method in Python, engineers and analysts can efficiently solve the nonlinear equations and obtain converged nodal displacements.

It is important to note that the accuracy and convergence of these methods depend on the nature of the stress equations, the initial guess for nodal displacements, and the chosen convergence criteria. Sensitivity to the initial guess and appropriate convergence criteria should be considered to ensure reliable results.

Additionally, validation and verification of the computational model using analytical solutions, experimental data, or benchmark examples are essential steps to ensure the accuracy and reliability of the implemented methods and the numerical model itself.

Overall, the Seidel and Newton methods provide powerful tools for solving curvilinear nodal stress equations and can aid in the accurate analysis of structural systems. By understanding the concepts and considerations of these methods, engineers and analysts can make informed decisions and obtain valuable insights into the behavior and performance of the structures under study.

Keywords: curvilinear nodal stress equations, Seidel method, Newton method, nonlinear equations, iterative technique, nodal displacements, equilibrium conditions, boundary conditions, Python programming, structural analysis, finite element analysis, iterative approaches, linearization, linear system, convergence.

INTRODUCTION

Curvilinear nodal stress equations play a crucial role in various engineering fields, such as structural analysis and finite element analysis. These equations describe the stress distribution within a structure, considering its geometric shape and material properties. In this article, we will explore two methods, namely Seidel and Newton methods, for solving curvilinear nodal stress equations. We will implement these methods in Python and provide a step-by-step guide to help you understand the process.

MATERIALS AND METHODS

To solve curvilinear nodal stress equations using the Seidel and Newton methods, we will require the following materials and methods:

Materials:

1. Python programming language: We will utilize Python for implementing the algorithms and solving the equations.
2. Numerical libraries: We will use numerical libraries such as NumPy and SciPy for matrix operations and solving linear systems of equations.

Methods:

1. Define the stress equations: Start by defining the curvilinear nodal stress equations that represent the relationship between nodal displacements and stresses at each node. This equation depends on the specific problem you are trying to solve.
2. Implement the Seidel method:
- Define a function `compute_new_displacement(nodal_displacements, node_index)` that computes the new displacement for a given node based on the current nodal displacements.
- Implement the `solve_seidel(nodal_displacements, iterations, tolerance)` function that iteratively updates the nodal displacements until convergence is achieved. It uses the `compute_new_displacement` function for each node. The iterations and tolerance parameters determine the maximum number of iterations and the convergence threshold, respectively.
3. Implement the Newton method:
- Define a function ` compute_new_displacement_and_residual ( nodal_displacements, node_index)` that computes the new displacement, residual, and the Jacobian matrix for a given node based on the current nodal displacements.
- Implement the `solve_newton(nodal_displacements, iterations, tolerance)` function that iteratively updates the nodal displacements until convergence is achieved. It uses the `compute_new_displacement_and_residual` function for each node. Additionally, it solves a linear system of equations using the `solve_linear_system` function to obtain the update in nodal displacements. The iterations and tolerance parameters determine the maximum number of iterations and the convergence threshold, respectively.
4. Solve the stress equations:
- Define the initial guess for nodal displacements.
- Call either the `solve_seidel` or `solve_newton` function, passing the initial nodal displacements, desired number of iterations, and tolerance.
- Obtain the converged nodal displacements as the output.
5. Post - processing and analysis:
- Once the nodal displacements are obtained, you can analyze the results further based on your specific requirements. This may involve calculating stresses, strains, or other quantities of interest derived from the nodal displacements.

Understanding Curvilinear Nodal Stress Equations: Curvilinear nodal stress equations are typically represented as a set of nonlinear algebraic equations. These equations consider the

relationship between the nodal displacements and the stresses at each node. The goal is to find the nodal displacements that satisfy the equilibrium conditions and boundary conditions.

Seidel Method:

The Seidel method is an iterative technique used to solve nonlinear equations. It starts with an initial guess for the nodal displacements and iteratively updates the values until convergence is achieved. Here's an implementation of the Seidel method for solving curvilinear nodal stress equations:

```
def solve_seidel(nodal_displacements, iterations=100, tolerance=1e-6):
    num_nodes = len(nodal_displacements)
    for _ in range(iterations):
        max_error = 0.0
        for i in range(num_nodes):
            old_displacement = nodal_displacements[i]
            # Compute new displacement using the stress equations
            nodal_displacements[i] = compute_new_displacement(nodal_displacements, i)
            error = abs(nodal_displacements[i] - old_displacement)
            if error > max_error:
                max_error = error
        if max_error < tolerance:
            break
    return nodal_displacements
```

Newton Method:

The Newton method is another popular approach for solving nonlinear equations. It utilizes the concept of linearization to approximate the solution iteratively. Here's an implementation of the Newton method for solving curvilinear nodal stress equations:

```
def solve_newton(nodal_displacements, iterations=100, tolerance=1e-6):
    num_nodes = len(nodal_displacements)
    for _ in range(iterations):
        residuals = []
        jacobian = []
        for i in range(num_nodes):
            old_displacement = nodal_displacements[i]
            # Compute new displacement and residual using the stress equations
            nodal_displacements[i], residual, jacobian_row = compute_new_displacement_and_residual(nodal_displacements, i)
            residuals.append(residual)
            jacobian.append(jacobian_row)
        max_residual = max(abs(residuals))
        if max_residual < tolerance:
            break
        # Solve for the update in nodal displacements
```

```
        delta_displacements = solve_linear_system(jacobian, residuals)
        nodal_displacements -= delta_displacements
    return nodal_displacements
```

RESULTS

After implementing the Seidel and Newton methods to solve the curvilinear nodal stress equations, the following results can be obtained:

1. Converged nodal displacements: Both the Seidel and Newton methods aim to find the nodal displacements that satisfy the equilibrium conditions and boundary conditions of the system. The final output of the methods will be the converged nodal displacements, which represent the solution to the stress equations.
2. Convergence information: The methods iterate until a convergence criterion is met. This criterion is typically based on the difference between the nodal displacements in consecutive iterations or the residuals of the equations. The number of iterations required for convergence and the achieved tolerance level can provide insights into the behavior of the system and the efficiency of the solution method.
3. Post - processing analysis: With the obtained nodal displacements, you can perform further analysis on the system. This may involve calculating stress distributions, strains, deformations, or any other quantities of interest derived from the nodal displacements. These results can provide valuable insights into the behavior and performance of the structure under the applied loads and boundary conditions.

It is important to note that the specific results obtained will depend on the problem at hand, including the nature of the stress equations, the complexity of the system, and the accuracy of the input parameters. It is advisable to validate the results by comparing them with known analytical solutions or experimental data, if available, to ensure the accuracy and reliability of the computational approach.

By following the outlined methods and analyzing the results obtained, you can gain a better understanding of the stress distribution within the structure and make informed decisions for design, optimization, or further analysis purposes.

DISCUSSION

The use of the Seidel and Newton methods for solving curvilinear nodal stress equations provides engineers and analysts with powerful tools to analyze the stress distribution within complex structures. Let's discuss some key points regarding the methods and their implications:

1. Iterative nature: Both the Seidel and Newton methods are iterative techniques that rely on updating the nodal displacements until a convergence criterion is met. This iterative approach allows for the solution of nonlinear equations that may not have analytical solutions. However, it is crucial to set appropriate convergence criteria and monitor the number of iterations required to achieve convergence. In some cases, the methods may converge slowly or exhibit convergence issues if the initial guess is far from the true solution.
2. Seidel method: The Seidel method is a simple and easy-to-implement iterative technique. It sequentially updates the nodal displacements based on the current values, converging towards the solution. However, the Seidel method may require a larger number of iterations to reach

convergence compared to the Newton method. It is suitable for problems where speed is not a critical factor or when the equations exhibit strong convergence properties.

3. Newton method: The Newton method provides a more efficient approach by utilizing linearization and solving a linear system of equations for the update in nodal displacements. It typically converges faster than the Seidel method, especially for problems with complex nonlinearities. However, the Newton method requires the computation of the Jacobian matrix, which can be computationally expensive for large-scale problems.

4. Sensitivity to initial guess: Both methods can be sensitive to the initial guess for nodal displacements. Choosing a good initial guess that is close to the true solution can significantly improve convergence and accuracy. It is often beneficial to use insights from engineering intuition, previous analyses, or simplified models to estimate a reasonable initial guess.

5. Convergence and accuracy trade-off: The convergence criteria and tolerance level need to be carefully selected to balance convergence speed and accuracy. Setting a tight tolerance may increase computational time but yield more accurate results, while a looser tolerance may speed up the computation but sacrifice accuracy. It is important to perform convergence studies and verify the results against known solutions or experimental data to ensure the reliability of the obtained results.

6. Model validation and verification: It is essential to validate the computational model by comparing the results with analytical solutions, experimental data, or benchmark examples whenever possible. This step helps verify the accuracy and reliability of the implemented methods and the numerical model itself. If discrepancies are found, further investigation and refinement of the model may be required.

In conclusion, the Seidel and Newton methods offer effective approaches to solve curvilinear nodal stress equations, enabling accurate analysis of structural systems. Each method has its advantages and considerations, and the choice between them depends on the specific problem, computational resources, and required accuracy. By understanding the underlying principles, limitations, and implications of these methods, engineers and analysts can make informed decisions and obtain reliable results for their structural analysis tasks.

CONCLUSION

In this article, we explored the Seidel and Newton methods for solving curvilinear nodal stress equations. Both methods provide iterative approaches to approximate the solution. The Seidel method updates the nodal displacements sequentially, while the Newton method utilizes linearization and solves for the update in nodal displacements using a linear system. By implementing these methods in Python, you can efficiently solve complex curvilinear nodal stress equations, enabling accurate analysis of structural systems.

Remember to adapt the code to your specific problem by implementing the `compute_new_displacement`, `compute_new_displacement_and_residual`, and `solve_linear_system` functions according to the equations relevant to your application.

Keep in mind that the convergence of these methods heavily depends on the nature of the stress equations and the initial guess for the nodal displacements. Adjust the number of iterations and tolerance values accordingly to achieve accurate results.

References

1. Ratschek H. Teilbarkeitskriterien der Intervallarithmetik // Journal fËur die reine und angewandte Mathematik. – 1972. – Bd. 252. – S. 128–138.
2. Rump S. M. Introduction to ACRITH — accurate scientific algorithms // Computerarithmetic, Scientific Computation and Programming Languages / Ed.:
3. S.M. Markov (Ed.), Scientific Computation and «Mathematical» Modelling, DATECS Publishing, Sofia, 1993.
4. «Elektr tarmoqlari va tizimlari» fanidan kurs loyihasini bajarish bo‘yicha uslubiy qo‘llanma./Sharipov U.B., Xamidov Sh.V., Xaydarov S.J. – T.: ToshDTU, 2003.
5. G‘oyibov T.Sh. Elektr tarmoqlari va tizimlari/O‘quv qo‘llanma. – T.: «Voris-nashriyot». 2010. 160 b.