A Krylov Subspace Accelerated Newton Algorithm

Michael H. Scott[1] and Gregory L. Fenves[2]

ABSTRACT

In the nonlinear analysis of structural systems, the Newton-Raphson (NR) method is the most common method to solve the equations of equilibrium. The NR method is easy to implement and gives an asymptotically quadratic rate of convergence. However, there are some well known shortcomings of the NR method. The Modified Newton (MN) method alleviates some of the problems of NR, but with a asymptotically linear rate of convergence requiring an excessive number of iterations to reach equilibrium.

Accelerated Newton (AN) methods strike a compromise between the NR and MN methods. Generally, the compromise is to hold the stiffness matrix constant during iteration, but to use selected information from previous iterations to accelerate convergence of the MN method. Since the mid 1980s, AN methods based on projections into Krylov subspaces have been developed by researchers in the mathematics community. Of particular interest is the method formulated by Carlson and Miller (1998) involving low rank least squares projections. Its performance in the nonlinear analysis of structures is comparable to NR, requiring only a few more iterations to converge while taking less computational time than both NR and MN.

INTRODUCTION

In the finite element analysis of nonlinear structural systems, the residual force vector, $\mathbf{R}$, is defined as the difference between the external load vector, $\mathbf{P}_f$, and the internal force vector, $\mathbf{P}_r$, which is a function of the nodal displacement vector, $\mathbf{u}$,

$$\mathbf{R}(\mathbf{u}) \equiv \mathbf{P}_f - \mathbf{P}_r(\mathbf{u}) \tag{1}$$

Structural equilibrium is attained when there is a balance between the external loads and internal forces, i.e., when the residual forces are zero,

$$\mathbf{R}(\mathbf{u}) = \mathbf{0} \tag{2}$$

In general, equilibrium cannot be found exactly and approximate root finding algorithms must be employed. Such algorithms advance the solution for the

[1]Grad. Stud. Res., Dept. of Civ. and Envir. Engrg., Univ. of California, Berkeley, CA.
[2]Prof., Dept. of Civ. and Envir. Engrg., Univ. of California, Berkeley, CA.

displacement vector from $\mathbf{u}_k$ to $\mathbf{u}_{k+1}$ by the displacement increment $\mathbf{v}_{k+1}$,

$$\mathbf{u}_{k+1} = \mathbf{u}_k + \mathbf{v}_{k+1} \tag{3}$$

Iteration begins from an initial guess for the displacement vector, $\mathbf{u}_0$, and terminates when the magnitude of the residual force vector, $\mathbf{R}_{k+1} \equiv \mathbf{R}(\mathbf{u}_{k+1})$, is sufficiently small, i.e.,

$$\|\mathbf{R}_{k+1}\| \leq \varepsilon \tag{4}$$

where ε is a specified numerical tolerance, and $\| \ \|$ indicates a vector norm, typically the Euclidean L_2 norm.

CONVENTIONAL NEWTON ALGORITHMS

Use of the term "conventional" refers to Newton algorithms which rely solely on the tangent stiffness matrix to advance the solution. Of interest are two such algorithms: Newton-Raphson and Modified Newton.

Newton-Raphson

The Newton-Raphson algorithm is derived by a truncated Taylor expansion (Stoer and Bulirsch 1993) of the residual force vector, $\mathbf{R}$, about the current displacements, $\mathbf{u}_k$,

$$\mathbf{R}(\mathbf{u}_k + \mathbf{v}_{k+1}) = \mathbf{R}(\mathbf{u}_k) + \left.\frac{\partial \mathbf{R}}{\partial \mathbf{u}}\right|_{\mathbf{u}_k} \mathbf{v}_{k+1} \tag{5}$$

The partial derivative of the residual force vector (1) gives rise to the tangent stiffness matrix, $\mathbf{K}_T$,

$$\frac{\partial \mathbf{R}}{\partial \mathbf{u}} = -\frac{\partial \mathbf{P}_r}{\partial \mathbf{u}} = -\mathbf{K}_T \tag{6}$$

and the iteration becomes

$$\mathbf{R}_{k+1} = \mathbf{R}_k - \mathbf{K}_T \mathbf{v}_{k+1} \tag{7}$$

By forcing the residual $\mathbf{R}_{k+1}$ to be zero, the displacement increment is then

$$\mathbf{v}_{k+1} = \mathbf{K}_T^{-1} \, \mathbf{R}_k \tag{8}$$

which represents the solution to a linear system of simultaneous equations.

Despite its simplicity and asymptotically quadratic rate of convergence, the Newton-Raphson algorithm has a few shortcomings. The tangent stiffness matrix, $\mathbf{K}_T$, must be formed and factorized at every iteration, which is a costly computational procedure for large systems of equations. In addition, the Newton-Raphson algorithm is subject to residual "flip-flop" due to sudden changes in the tangent stiffness matrix.

Modified Newton

A modification of the Newton-Raphson algorithm is to keep the tangent stiffness matrix constant during iteration,

$$\mathbf{R}_{k+1} = \mathbf{R}_k - \mathbf{K}_0 \mathbf{v}_{k+1} \tag{9}$$

where $\mathbf{K}_0$ is the tangent stiffness matrix at the start of iteration. Forcing the residual $\mathbf{R}_{k+1}$ to zero, the displacement increment is

$$\mathbf{v}_{k+1} = \mathbf{K}_0^{-1} \mathbf{R}_k \tag{10}$$

An advantage of the Modified Newton algorithm is the tangent stiffness matrix need only be factorized once during a load step, but the asymptotic rate of convergence is only linear. However, the benefit of a single matrix factorization is usually outweighed by the excessive number of iterations required to reach equilibrium. Unlike the Newton-Raphson algorithm, the Modified Newton algorithm is not subject to residual "flip-flop" since the tangent stiffness matrix is held constant during iteration.

KRYLOV SUBSPACE ACCELERATOR

The Krylov subspace based algorithm developed by Carlson and Miller (1998) accelerates the convergence of the Modified Newton iteration. The acceleration brings the rate of convergence close to that of Newton-Raphson at a lower computational expense.

Formulation

For this algorithm, the tangent stiffness matrix of equation 9 is replaced by an unknown matrix, $\mathbf{A}$,

$$\mathbf{R}_{k+1} = \mathbf{R}_k - \mathbf{A}\mathbf{v}_{k+1} \tag{11}$$

Next, it is assumed the displacement increment is obtained in two parts,

$$\mathbf{v}_{k+1} = \mathbf{w}_{k+1} + \mathbf{q}_{k+1} \tag{12}$$

Substitution of equation 12 into equation 11 gives

$$\mathbf{R}_{k+1} = \mathbf{R}_k - \mathbf{A}\mathbf{w}_{k+1} - \mathbf{A}\mathbf{q}_{k+1} \tag{13}$$

The key to this accelerator is to minimize in the L_2 norm the difference between the current residual, $\mathbf{R}_k$ and the vector $\mathbf{A}\mathbf{w}_{k+1}$.

$$\min \|\mathbf{R}_k - \mathbf{A}\mathbf{w}_{k+1}\|_2 \tag{14}$$

To this end, it is assumed the vector $\mathbf{w}_{k+1}$ is a linear combination of the previous displacement increments $\mathbf{v}_1, \ldots, \mathbf{v}_k$,

$$\mathbf{w}_{k+1} = c_1 \mathbf{v}_1 + \cdots + c_k \mathbf{v}_k \tag{15}$$

The vector $\mathbf{A}\mathbf{w}_{k+1}$ is then

$$\mathbf{A}\mathbf{w}_{k+1} = c_1\mathbf{A}\mathbf{v}_1 + \cdots + c_k\mathbf{A}\mathbf{v}_k \tag{16}$$

where the $\mathbf{A}\mathbf{v}_1, \ldots, \mathbf{A}\mathbf{v}_k$ are obtained by differencing of residuals via equation 11 (drop the indices and solve for $\mathbf{A}\mathbf{v}_k$)

$$\mathbf{A}\mathbf{v}_k = \mathbf{R}_{k-1} - \mathbf{R}_k \tag{17}$$

Given the current residual $\mathbf{R}_k$ and $\mathbf{A}\mathbf{v}_1, \ldots, \mathbf{A}\mathbf{v}_k$, it is a relatively simple linear least squares problem to find the coefficients $c_1, \ldots, c_k$, and the first component, $\mathbf{w}_{k+1}$ of the displacement increment via equation 15.

After solving for the coefficients, there will be a least squares residual,

$$\mathbf{Q}_k = \mathbf{R}_k - \mathbf{A}\mathbf{w}_{k+1} \tag{18}$$

Equation 13 can now be written in terms of the least squares residual and the second component, $\mathbf{q}_{k+1}$, of the displacement increment,

$$\mathbf{R}_{k+1} = \mathbf{Q}_k - \mathbf{A}\mathbf{q}_{k+1} \tag{19}$$

which now takes the form of the Modified Newton iteration (9), except the matrix $\mathbf{A}$ is still unknown. At this point, a reasonable assumption to make is $\mathbf{A} \approx \mathbf{K}_0$, such that the second component of the displacement increment is

$$\mathbf{q}_{k+1} = \mathbf{K}_0^{-1}\,\mathbf{Q}_k \tag{20}$$

when the residual $\mathbf{R}_{k+1}$ is forced to zero. Both components of the displacement increment are now known (equation 12). On the first iteration ($k = 0$), the vector $\mathbf{w}_1$ is zero, and the first displacement increment is simply

$$\mathbf{v}_1 = \mathbf{K}_0^{-1}\,\mathbf{R}_0 \tag{21}$$

as is the case in the Modified Newton algorithm. A graphical representation of the first two iterations for this accelerator is shown in Figure 1.

Implementation Details

This accelerator requires the vectors $\mathbf{v}_1, \ldots, \mathbf{v}_k$ and $\mathbf{A}\mathbf{v}_1, \ldots, \mathbf{A}\mathbf{v}_k$ be stored in memory. In order to keep both the required amount of memory and the computational cost of the least squares solution low, a maximum number of vectors is specified. For this work, the maximum number of vectors is three.

When the maximum number of vectors is exceeded, the $\mathbf{v}$ and $\mathbf{A}\mathbf{v}$ vectors are cleared out, the tangent stiffness matrix is reformed, and the least squares process is restarted. There are various other restart strategies for when the maximum number of vectors is exceeded. However, these strategies are not investigated here.

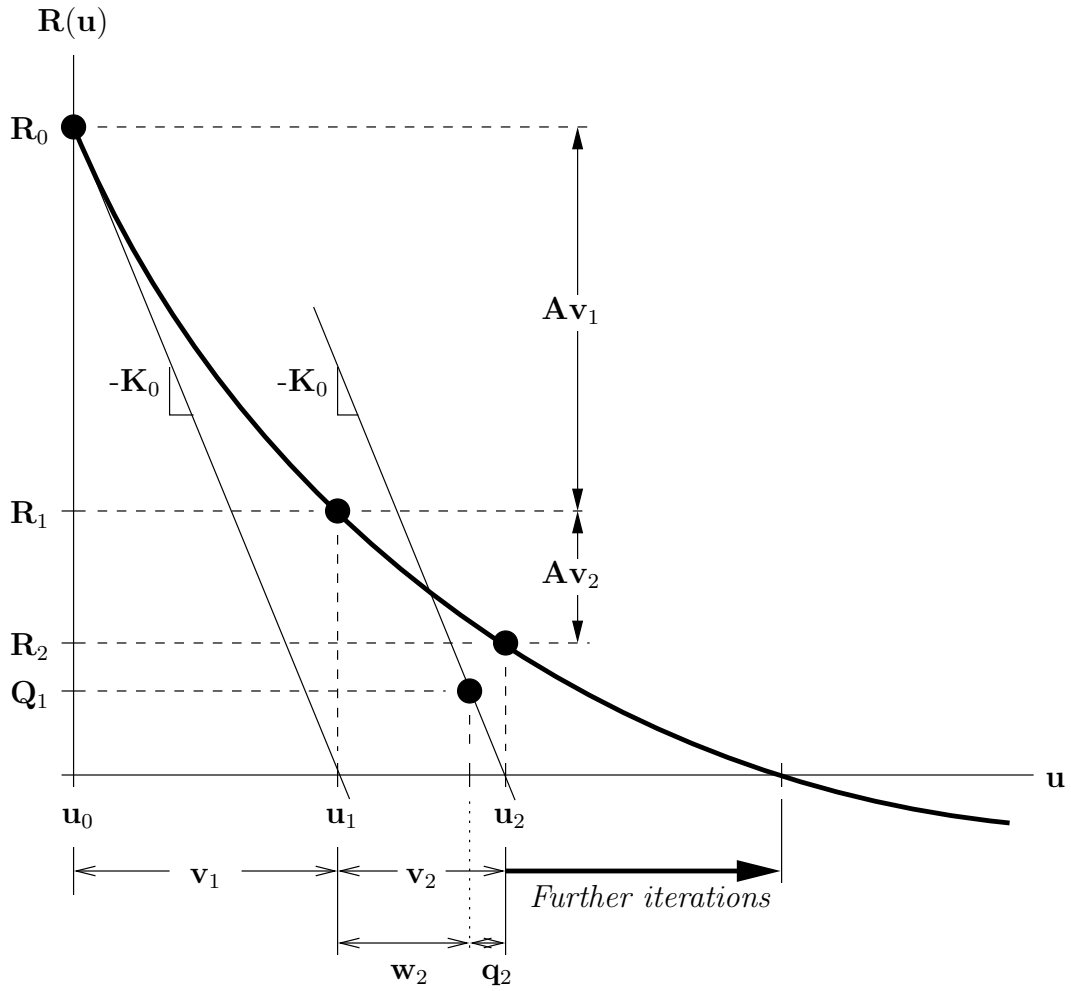

FIG. 1. Graphical representation of the first two Krylov Newton iterations

SOFTWARE IMPLEMENTATION

Within the framework of OpenSees (McKenna et al. 2000), a new class is added for finding equilibrium by an accelerated Newton algorithm. This class is called *AcceleratedNewton*, and is composed of an instance of the *Accelerator* class, as shown in Figure 2. This design allows accelerators to be interchanged and allows new accelerators, such as the secant based accelerator of Crisfield (1991), to be seamlessly added to the framework.

The virtual method *solveCurrentStep()* of the *EquiSolnAlgo* class is to encapsulate the equilibrium solution algorithm for a single load step in OpenSees. Pseudo-code for this method as it is implemented by the *AcceleratedNewton* class is shown in Figure 3. To form the residual vector and tangent stiffness matrix, the methods *formUnbalance()* and *formTangent()* are invoked upon an instance of the *IncrementalIntegrator* class of OpenSees via the pointer `theIntegrator`. A pointer to an instance of the *LinearSOE* class is contained in `theSOE`. The method

solve() provides the solution to the current system of linear equations. Further details of the *EquiSolnAlgo*, *IncrementalIntegrator*, and *LinearSOE* classes of OpenSees can be found in the work of McKenna (1997).

The variable `theAccelerator` is a pointer to an instance of the *Accelerator* class. Two methods are invoked upon an *Accelerator* object during an equilibrium iteration:

- `accelerate()` – Performs the necessary matrix-vector operations, e.g., least squares, to accelerate the Modified Newton iteration.
- `updateTangent()` – Updates the tangent stiffness matrix, if necessary.

Note that when `theAccelerator` is null, the pseudo-code in Figure 3 reduces to the Modified Newton algorithm.

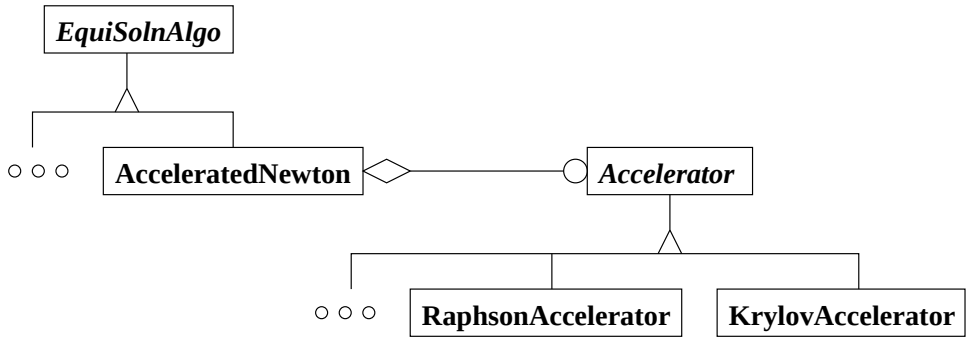

FIG. 2. Accelerator class hierarchy

EXAMPLE STRUCTURAL APPLICATIONS

Two examples are presented: the first a metal strip under tension, the second a moment-resisting frame under cyclic loading. The first example shows the computational benefits of the Krylov Newton algorithm, while the second demonstrates a case where Krylov Newton finds a solution that Newton-Raphson and Modified Newton cannot.

Tension Strip

This example is of a metal strip of unit area in a state of tension (Figure 4). Bilinear isoparametric quad elements (Zienkiewicz and Taylor 2000) are used with a plane strain J_2 plasticity material model (Simo and Hughes 1998). The mesh is repeatedly refined in order to increase the number of equations, and thereby increase the time spent solving the linear system of equations at each iteration.

The ratio of the number of elements along the strip length, N_x, to those across the width, N_y, remains constant at 20:1 and the values of N_y range over $\{5, 10, 15, 20, 25\}$, such that both the number of equations and profile of the system of equations increase with increasing N_y. For these values of N_y, the corresponding number of equations is $\{1200, 4400, 9600, 16800, 26000\}$.

```
AcceleratedNewton::solveCurrentStep(void)
{
   // Form initial residual, R_0
   theIntegrator->formUnbalance();

   // Form initial tangent, K_0
   theIntegrator->formTangent();

   do {
      // Solve for the Modified Newton increment
      theSOE->solve();
      vAccel = theSOE->getX();

      // Accelerate the increment
      if (theAccelerator != 0)
         theAccelerator->accelerate(vAccel);

      // Update the system with the accelerated increment, v_{k+1}
      theIntegrator->update(vAccel);

      // Form new residual, R_{k+1}
      theIntegrator->formUnbalance();

      // Allow the accelerator to update the tangent
      if (theAccelerator != 0)
         theAccelerator->updateTangent();

   } while (theTest->test() == FAIL);
}
```

FIG. 3. Pseudo-code for the AcceleratedNewton class

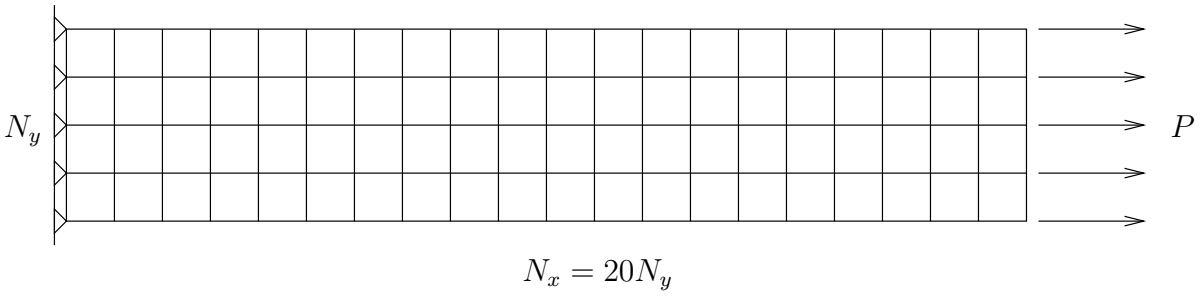

FIG. 4. Tension strip mesh

The material properties are $E = 30000$, $\nu = 0.3$, $\sigma_y = 60$, and 2% strain hardening. The maximum resultant loading on the free end is $P = 180$, applied in ten increments. The convergence criteria is a relative norm of the residual vector with respect to the initial residual for each load step:

$$\frac{\|\mathbf{R}_k\|_2}{\|\mathbf{R}_0\|_2} \leq 10^{-7} \tag{22}$$

Results for the case of $N_y = 25$ are shown in Table 1. More iterations are required for the Krylov Newton algorithm than Newton-Raphson, however less CPU time (reported in seconds) is needed for the analysis. An excessive number of iterations and CPU time are required for the Modified Newton algorithm. Parenthesized values in Table 1 are times normalized with respect to those for Newton-Raphson.

TABLE 1. Tension strip results for $N_y = 25$

Algorithm	Iterations	Total CPU Time	Solve CPU Time
Newton-Raphson	52	109.83 (1.0)	51.06 (1.0)
Krylov Newton	91	96.26 (0.88)	39.28 (0.77)
Modified Newton	4247	2468.15 (22.5)	646.44 (12.7)

Results for all values of N_y for the Newton-Raphson and Krylov Newton algorithms are shown in Figure 5. The general trend of Krylov Newton taking more iterations at a lower computational expense than Newton-Raphson is apparent in this Figure.

Steel Frame

The structural model for this example is a five story, one bay steel frame, as shown in Figure 6 with $H = 4$ and $W = 8$. Each structural member is discretized into five displacement based beam-column elements with a linear curvature approximation along the element length (Cook et al. 1989). This discretization gives a relatively small system of 210 equations of equilibrium. The constitutive model for each element is a bilinear moment-curvature relationship with linear kinematic hardening. The material properties are $E = 30000$, $I = 984$, $M_y = 3400$, and 5% strain hardening in the moment-curvature relationship.

A triangular reference load pattern is applied to the frame with peak intensity, P. The peak intensity varies with time as a sine wave,

$$P(t) = P_{max} \sin(t) \tag{23}$$

where $P_{max} = 240$. The analysis is conducted at pseudo-time increments of $\Delta t = \pi/6$ over twelve load steps, such that the fourth is the first unloading step. The same convergence criteria (22) as the first example is used for this example.

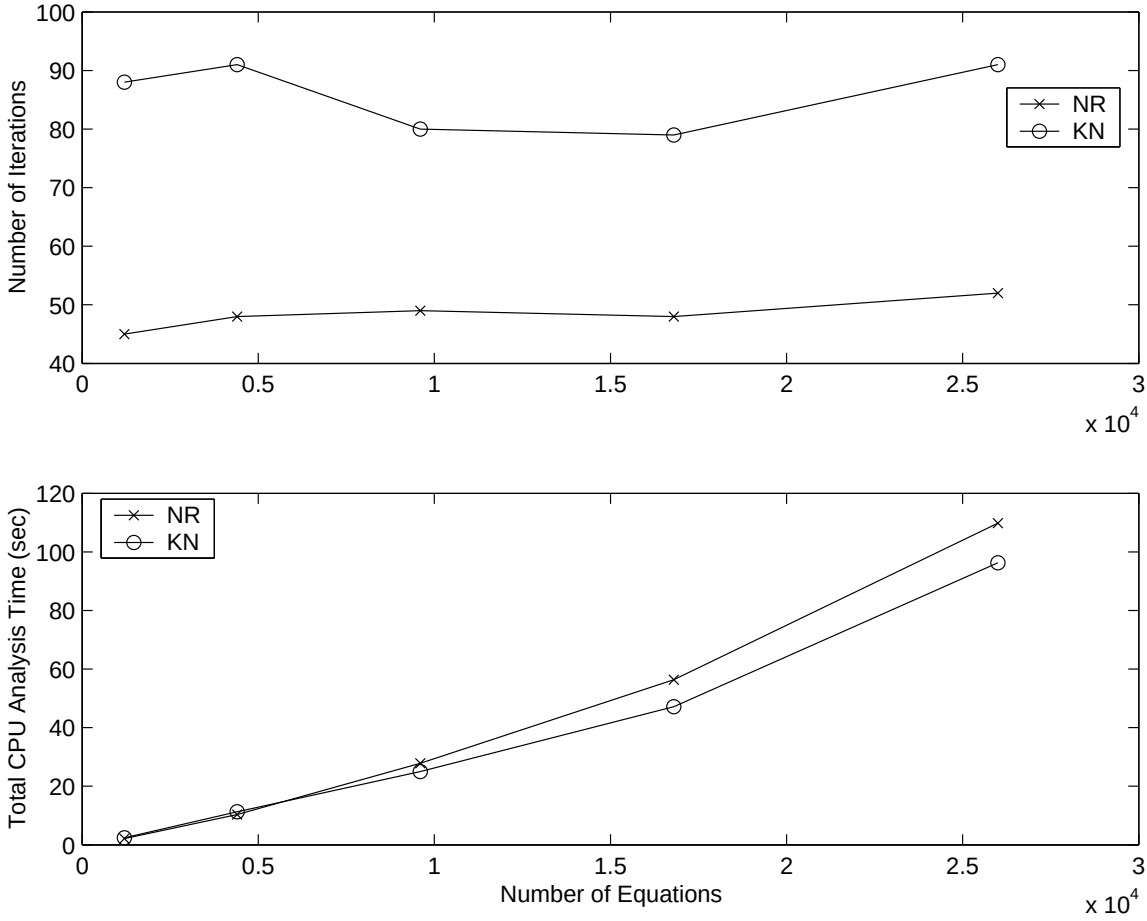

FIG. 5. Tension strip results for Newton-Raphson and Krylov Newton

The number of iterations required to reach equilibrium at each load step for the three algorithms is shown in Table 2. The Newton-Raphson algorithm fails on the third load step due to residual flip-flop, while the Modified Newton algorithm fails upon unloading, on the seventh load step. The Krylov Newton algorithm converges for all load steps, thereby exhibiting superior convergence properties for this particular problem.

REFERENCES

Carlson, N. and Miller, K. (1998). "Design and Application of a 1D GWMFE Code." *SIAM Journal of Scientific Computing*, 19, 766–798.

Cook, R. D., Malkus, D. S., and Plesha, M. E. (1989). *Concepts and Applications of Finite Element Analysis*. John Wiley & Sons.

Crisfield, M. A. (1991). *Non-linear Finite Element Analysis of Solids and Structures*, Vol. 1. John Wiley & Sons.

McKenna, F. (1997). "Object-oriented finite element programming: Frameworks

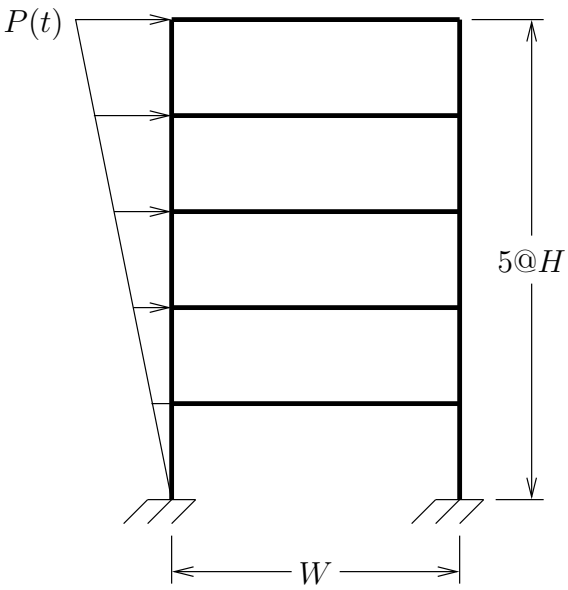

FIG. 6. Moment resisting frame model

TABLE 2. Iterations to convergence for moment resisting frame

Load Step	Newton-Raphson	Modified Newton	Krylov Newton
1	1	1	1
2	4	346	25
3	fail	365	29
4	-	338	57
5	-	296	5
6	-	278	5
7	-	fail	5
8	-	-	1
9	-	-	1
10	-	-	1
11	-	-	4
12	-	-	37

for analysis, algorithms, and parallel computing," PhD thesis, University of California.

McKenna, F., Fenves, G. L., Jeremic, B., and Scott, M. H. (2000). "Open System for Earthquake Engineering Simulation. http://opensees.berkeley.edu.

Simo, J. C. and Hughes, T. J. R. (1998). *Computational Inelasticity*. Springer-Verlag.

Stoer, J. and Bulirsch, R. (1993). *Introduction to Numerical Analysis*. Springer-Verlag, second edition.

Zienkiewicz, O. C. and Taylor, R. L. (2000). *The Finite Element Method: Volume 1, The Basis*. Butterworth-Heinman, fifth edition.