LM-06K143
January 9, 2007

A k_{eff} Search Capability in MC21

RE Morrow, TH Trumbull, TJ Donovan and TM Sutton

NOTICE

Joint International Topical Meeting on Mathematics & Computation and Supercomputing in Nuclear Applications (M&C + SNA 2007)
Monterey, California, April 15-19, 2007, on CD-ROM, American Nuclear Society, LaGrange Park, IL (2007)

A k_{eff} Search Capability in MC21

R. E. Morrow†, T. H. Trumbull, T. J. Donovan, and T. M. Sutton
Lockheed Martin Company
Schenectady, NY
†corresponding author**,** morrori@kapl.gov

ABSTRACT

The MC21 Monte Carlo code is required to permit an individual geometric component or groups of components to be tagged as 'movable' within some permissible range. Typical examples of such movable components would be control devices such as translating rods or rotating drums. Given this geometric information, a target multiplication factor (k_{eff}), and a convergence criterion, MC21 will iterate on movable component positions and return a final position that reflects a k_{eff} close to the target value. An initial version of this capability is demonstrated through modifications to MC21 that sets the geometry data structures for the movable components, calls the main Fortran-95 solver to compute k_{eff}, and converges on the final position. This approach uses an adaptive batching algorithm that continually increases the accuracy of each successive MC21 k_{eff} result as the movable geometry approaches the converged position.

Key Words: MC21, Control Device, Geometry, Adaptive Batching

1. INTRODUCTION

MC21 is the Monte Carlo radiation transport code currently under joint development at the Knolls Atomic Power Laboratory and the Bettis Atomic Power Laboratory. MC21 is intended to push the Monte Carlo method beyond its traditional role as a benchmarking tool or "tool of last resort" and into a design role. Given this mission, MC21 requires a suite of engineering and analysis capabilities that are not typically available in a Monte Carlo transport code. One example of this capability is a 'k_{eff} search' capability in which certain geometric components that represent control devices are tagged as movable within a certain range of motion, and MC21 is requested to iterate on the positions of these control devices until a target k_{eff} (nominally 1.0) is achieved.

The k_{eff} search sequence itself is rather straightforward. The first step is to specify the problem data. This data includes the set geometric entities that comprise the movable region or regions, the maximum permissible range of motion, the target eigenvalue, and the required solution accuracy. Next is the inclusion of a controller system or module that computes the modifications to the geometric data, interacts with MC21's geometric data structures to make all model changes, requests the static solution for k_{eff} from MC21, and determines whether the solution has converged. The controller must take into consideration the Monte Carlo eigenvalue uncertainties to determine how to move the control device and maximize the convergence rate. Maximization of the convergence rate should also include the reuse of the converged neutron source from search iteration i as the starting source for iteration i+1, although this was not done for this study.

The k_{eff} search sequence does not require any changes to the MC21 methods related to the static solution itself.

This paper examines two areas: 1) the use of MC21's geometric representations to permit movable geometries, and 2) the development of an adaptive batching algorithm to efficiently account for the statistical uncertainty during the iterations on device position. Finally, results are given for two models indicating the successful implementation for the case of a simple translating control device.

2. Implementation

2.1. Moving the Control Device

MC21 possesses five main geometric constructs: *surface*, *component*, *grid*, *2Dlattice*, and *ellipse* [1]. Components, which are generally equivalent to cells in MCNP [2], are volumes defined by union and/or intersection sets of surface half-spaces. A component may possess additional internal geometric detail by assigning a grid to completely fill the component volume. Furthermore, in the case of an *xy* grid, each grid cell may be filled with a repeating array of elliptical cylinders in the form of a *2Dlattice*. Figure 1, a diagram of the MC21 HEU-MET-FAST-030 model [3], shows how a simple component may be defined using only a few surfaces but which contains a large amount of internal geometric complexity.

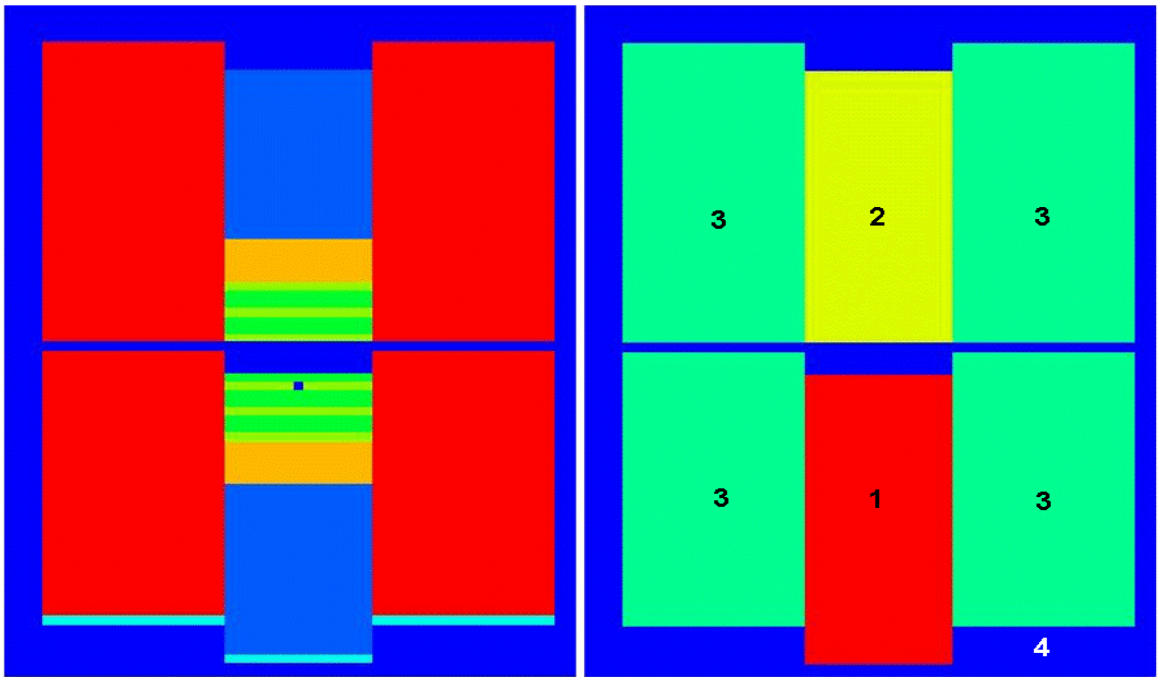

Figure 1. HEU-MET-FAST-030 model *yz* Cross Section View at *x*=0. Right view is a component plot and left view is a material plot.

The right hand view of Fig. 1 shows the components of the model. There are four components in all. Three of them—component 1 (red), component 2 (yellow), and component 3 (green)—reside at the same hierarchical level within the bounding component—component 4 (blue). The left hand view of Figure 2 shows the material detail of the benchmark problem. This detail is added to the three primary components through the use of the cylindrical grid construct.

In the actual criticality experiment associated with HEU-MET-FAST-030, the control device is the collection of regions contained by component 1. This control device is moved upward from beneath components 2 and 3 into the critical position shown in the figure. Component 1 itself is very simple, being defined as the intersection of a top plane, a bottom plane, and a right circular cylinder. Because the internal detail is associated with the component through a cylindrical grid, if component 1 is 'moved' by translating the three surfaces that define it, the internal detail moves with it with a very simple translation of the grid's origin. Furthermore, because component 4 (blue) hierarchically contains component 3, moving component 3 does not require any shifting or redefining of the bounding component. This assumes that the motion of the control device is constrained within a physically acceptable range. The HEU-MET-FAST-030 model illustrates how MC21's component hierarchy and internal grid detail will be used to full advantage as part of the k_{eff} search capability.

2.2. Iteration Controller

For the purposes of this study, MC21 was modified to control the k_{eff} search iterations, which includes manipulating the MC21 geometric data and interfacing with the MC21 static k_{eff} solver. The activity diagram governing the k_{eff} search sequence follows the simple description of §1 and shown in detail in Figure 3. Multiple eigenvalue calculations of MC21 are performed towards the goal of converging on a control device position based on a target k_{eff}. The control device movement is based on the sensitivity, or worth, of the control device calculated at each new step. For the initial study, motion of the control device is limited to a $\pm z$ direction only. The main responsibilities of the modifications to MC21 include: determining each new control device position, modifying the MC21 geometric data structures to reflect a change in geometry, calling the next k_{eff} calculation, and determining whether the search has completed.

2.3. Adaptive Batching Algorithm

The Adaptive Batching Algorithm (ABA) concept is intended to reduce the time spent in converging on a control device position that corresponds to a desired reactivity condition. The premise is that it is unnecessary to converge the eigenvalue to high precision if the current control device position is far from the final position. In practice, the number of active histories run for a given control device position need only be adequate to either 1) enable the determination of the next move, or 2) confirm that the device is at the critical position with the desired precision.

Two initial calculations are required to obtain a trial differential device worth (DW) and begin the series. Alternatively, the user could supply an initial guess at a differential device worth. The former case is considered here. Let the first calculation have active histories, n_a, the device positioned at z_{old}, and resulting eigenvalue, k_{old}. For the second calculation the device is moved to a new position, preferably one that guarantees the target position has been bracketed. Let the

new position be z_{new} and the resulting eigenvalue be k_{new} with the same number of active histories run. The differential device worth is then defined as

$$DW_{new} = \frac{d\rho}{dz} \approx \frac{k_{new} - k_{old}}{z_{new} - z_{old}} \quad , \tag{1}$$

where k_{new} and k_{old} are assumed to be close to unity. The extrapolated position, z', to achieve the target eigenvalue, k_{tar}, can now be written as

$$z' = z_{new} + \frac{\varepsilon(k_{tar} - k_{new})}{DW_{new}} \quad , \tag{2}$$

where an optional relaxation parameter, ε, has been added to moderate the change in device position. The magnitude of the change in device position is then

$$\delta = \left| z' - z_{new} \right| \quad . \tag{3}$$

The decision to move the device is made by propagating the eigenvalue uncertainties on the calculated differential device worth and extrapolated position. Assuming that k_{new} and k_{old} are uncorrelated, then the uncertainty on the extrapolated position, z', is

$$\sigma_{z'}^2 = \left(\frac{\varepsilon\left(z_{new} - z_{old}\right)\left(k_{old} - k_{tar}\right)}{\left(k_{new} - k_{old}\right)^2} \right)^2 \sigma_{k_{new}}^2 + \left(\frac{\varepsilon\left(z_{new} - z_{old}\right)\left(k_{tar} - k_{new}\right)}{\left(k_{new} - k_{old}\right)^2} \right)^2 \sigma_{k_{old}}^2 . \tag{4}$$

If the resulting uncertainty on the extrapolated position is low relative to the distance to be moved, then the control device is moved to the extrapolated position and a new eigenvalue calculation is started. If the uncertainty on the extrapolated position is large relative to the distance to be moved, then the calculated move is considered not statistically significant. The number of active batches is increased and a new eigenvalue calculation is started. The process ends when the eigenvalue has converged to the desired precision about the target.

Two approaches to increase the number of active batches were explored. The first technique bases the increase on the reduction in the uncertainty necessary to achieve a statistically significant move. The required uncertainty, σ_{new}, is set equal to the magnitude of the calculated move, δ. Then, assuming that the uncertainty is reduced as the square of the number of active batches, the number of active batches is increased by

$$n_a = n_{a,old} \left(\frac{\sigma_{old}}{\sigma_{new}} \right)^2 \quad , \tag{5}$$

where σ_{old} is the eigenvalue uncertainty after $n_{a,old}$ batches. Alternatively, the number of active batches can simply be increased by a constant factor when a move is calculated to be not statistically significant.

Pseudo-code of the ABA is shown in Fig. 2 along with a block-diagram in Fig. 3. User input parameters include a target eigenvalue, k_{tar}, with a desired precision, ∂_{tar}, and an estimate of the number of active histories n_a for the initial run. Control device parameters that represent the relative maximum z_{max} and minimum z_{min} device movement along a unit translational vector $\mathbf{v}_t$ are used to calculate the initial device worth DW_0 at step {6}. The initial device worth relies on calculating eigenvalue solutions at the maximum swing, k_{max}, and the minimum swing, k_{min},

performed at step {5}. With this device worth estimate, an initial movement distance, z', is determined in step {7}.

1. Initialize: k_{tar}, ∂_{tar}, n_a, z_{max}, z_{min}, $\mathbf{v}_t$.
2. Find control device components, c_i, $i = 1,...,n_c$.
3. Find number of unique surfaces, s_i, $i = 1,...,n_s$.
4. Solve: k_0, ∂_0 .
5. Solve: k_{max}, k_{min} .
6. Calculate: DW_0
7. Calculate: z', δ, $\sigma_{z'}$
8. update variables: $k_{old} \leftarrow k_0$, $\partial_{old} \leftarrow \partial_0$, $z_{old} \leftarrow z_{new}$, $z_{new} \leftarrow z'$
9. *converged* $\leftarrow$ *false*
10. While $(i < i_{max})$
    a. if $(k_{old} + \partial_{old}) \le (k_{tar} + \partial_{tar})$ and $(k_{old} - \partial_{old}) \ge (k_{tar} - \partial_{tar})$, *converged* $\leftarrow$ *true*
    b. if $(converged)$, exit
    c. if $(\delta \le \sigma_{z'})$, $n_a = C \cdot n_a$
    d. for $j = 1, n_c$
        i. *moveGrids*(c_j, z_{new})
    e. for $m = 1, n_s$
        i. *moveSurfaces*(s_m, z_{new})
    f. Solve: k_{new}, ∂_{new}
    g. Calculate: DW_{new}
    h. Calculate: z', δ, $\sigma_{z'}$
    i. update variables: $k_{old} \leftarrow k_{new}$, $\partial_{old} \leftarrow \partial_{new}$, $z_{old} \leftarrow z_{new}$, $z_{new} \leftarrow z'$
    j. $i = i + 1$

Figure 2. Pseudo-code for the Adaptive Batching Algorithm.

Step {10} starts the convergence loop of the Adaptive Batch Algorithm. Convergence occurs when the calculated eigenvalue ± its uncertainty is completely contained within the target eigenvalue ± its tolerance. If the magnitude of the control device move is less than its uncertainty then the move is considered not statistically significant and the number of active batches is increased by a factor, *C*. The factor *C* can be calculated based on Eq. (5), or set to a constant factor, e.g., a factor of four increase in n_a results in a reduction of the uncertainty on the eigenvalue of approximately two.

The hierarchical geometry structure of MC21 requires both the Grids and Surfaces to be moved. Grids are associated with components and surfaces are associated with the components that make up the control device. The algorithm was constrained to translational movement only.

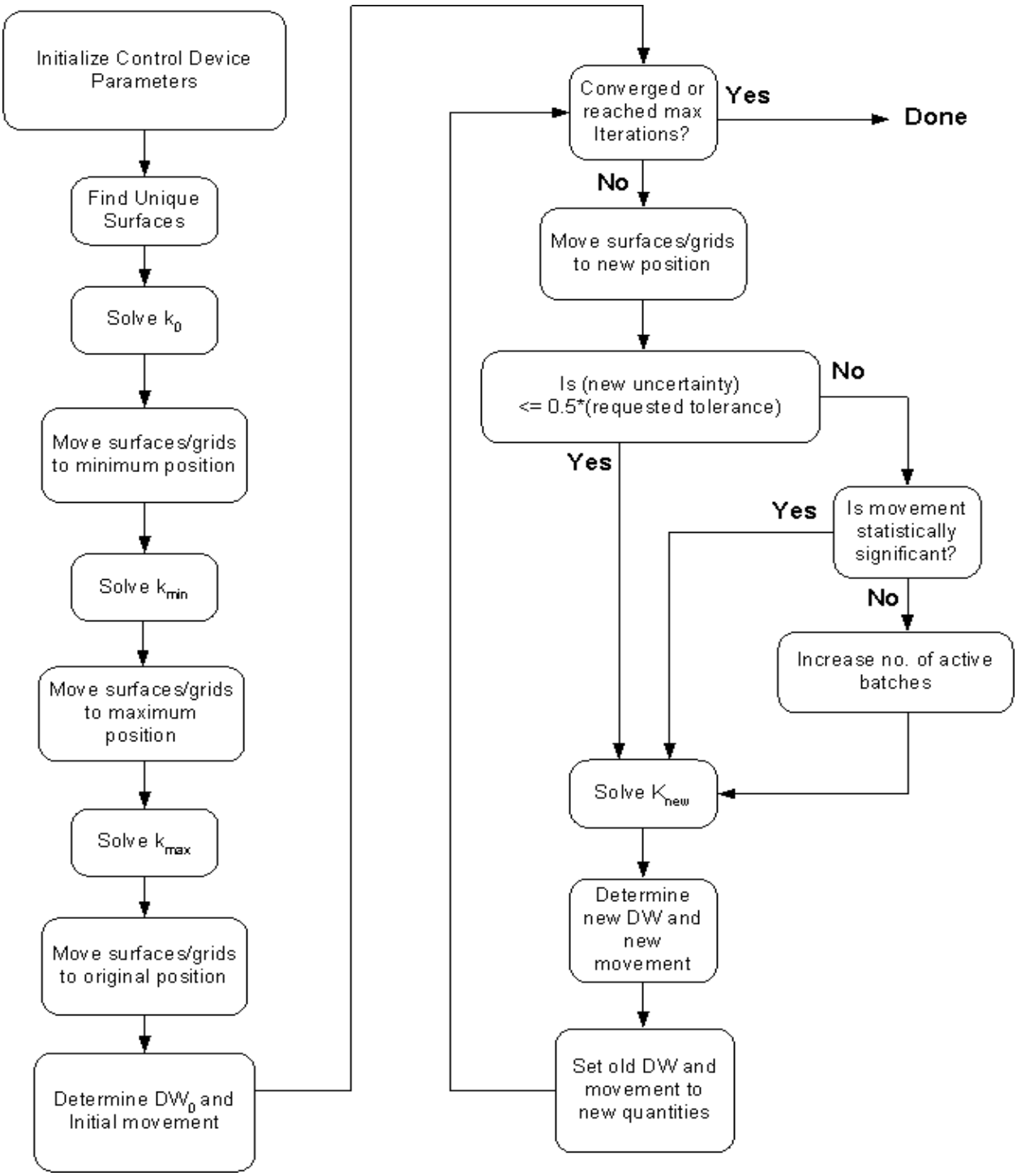

Figure 3. Block-diagram of the Adaptive Batching Algorithm.

3. RESULTS

Two models were examined for the purposes of this study. The MC21 HEU-MET-FAST-030 model detailed in §2.1 and a more realistic model based on GE-9 [1]. The HEU-MET-FAST-030 model was used to validate the ABA, although the integral device worth curve of Fig. 4 is not representative of standard reactor designs. It is interesting to note that the device worth curve is well approximated by a line over a large portion of device travel. This suggests that the linear extrapolations used in the device positioning algorithm will perform well for this model.

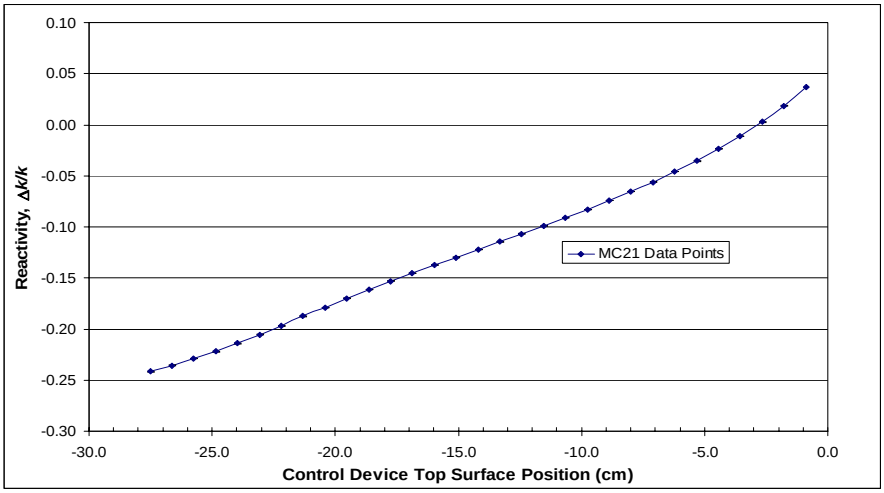

Figure 4. Reactivity versus Device Position for the HEU-MET-FAST-030 Model The calculated reactivities have 95% confidence intervals of ± 0.0003 Δ*k*/*k*.

The GE-9 cluster of Ref. 1 was modified to be reflecting in the x and y directions and possess escape boundaries conditions on the top and bottom. The axial height of the cluster was set to 7 feet, or 213.36 cm. This resulted in a more typical sigmoidal integral device worth curve (Fig. 5).

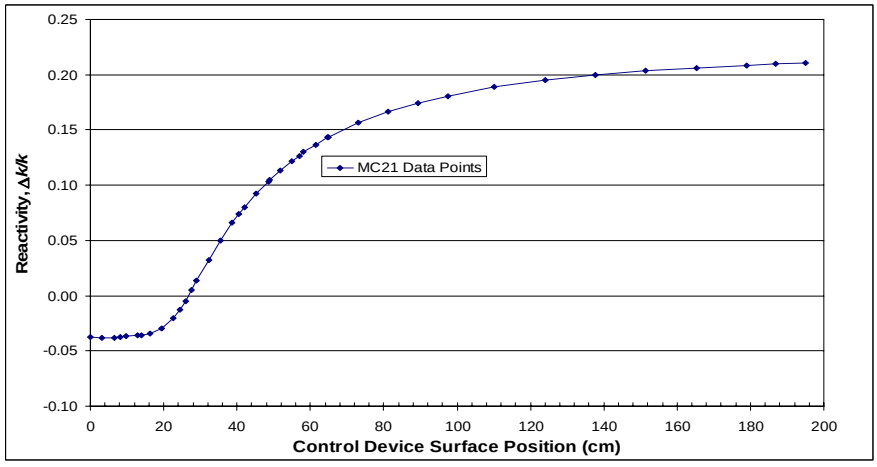

Figure 5. Reactivity versus Device Position for the GE-9 Model The calculated reactivities have 95% confidence intervals of ± 0.0004 Δ*k*/*k*.

3.1. HEU-MET-FAST-030 Model Results

The MC21 ABA was used to determine the control device sensitivity and the total number of histories required to reach a target eigenvalue solution of $k_{\text{eff}} = 1.0000 \pm 0.0005$ (95% CI). The control device consists of only component 1. Device sensitivity was determined by moving component 1 in the positive *z* direction from -27.52 to 0 cm, as shown in Fig. 6.

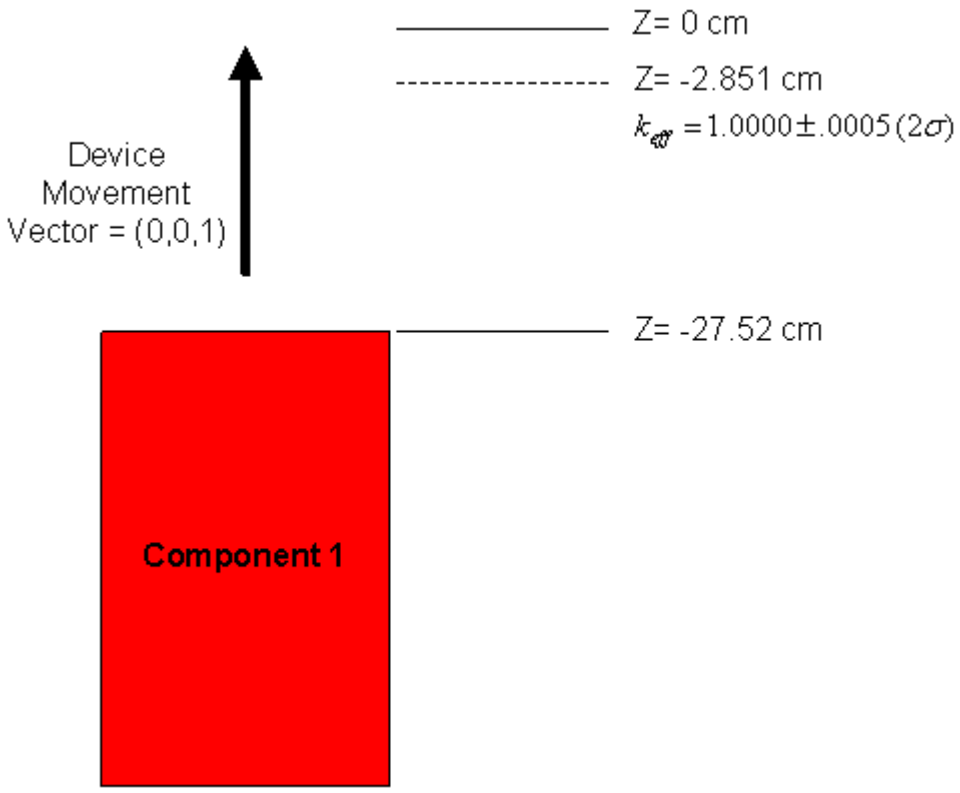

Figure 6. HEU-MET-FAST-030 Control Device Movement

Figure 7 shows the critical control device search history comparing the ABA method to a "baseline" run using MC21. The baseline run consists of a series of highly precise MC21 runs using a fixed number of active batches, discarded batches, and histories per batch. The low uncertainty in the eigenvalue allowed the baseline model to converge in three iterations. The three iterations required a total of 172.8 million histories. Relative to the baseline calculation, the ABA case required double the number of iterations to achieve the target eigenvalue. However, since the number of active histories was small to start and incrementally increased as necessary, the total number of histories required was ~7 times less than the baseline case. The ABA case resulted in a critical device position of *z* = -2.85 cm compared to a baseline position of *z* = -2.84 cm.

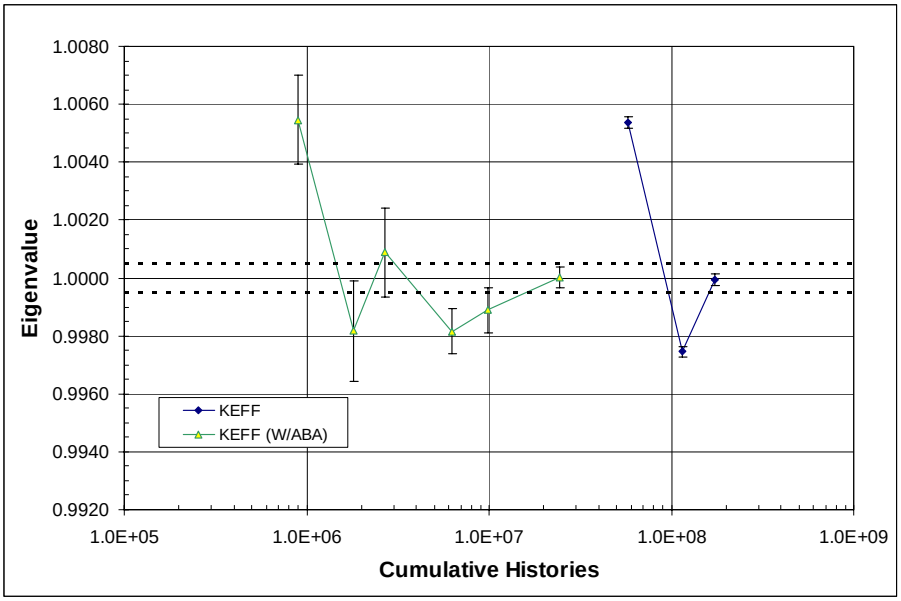

Figure 7. HEU-MET-FAST-030 Control Device Search History

3.2. GE-9 Model Detail and Results

The benchmark problem of Ref. 1 was modified and used as a test problem for the MC21 eigenvalue search capability. Figure 8 shows an axial and radial view of this model.

The model is a radially reflected conventional BWR 8x8 fuel assembly design with a single large central water rod and eight urania-gadolinia poison rods. These eight rods (including their claddings) are the control device for the model. The height of the pins and the bounding cell has been set to 7 feet (213.36 cm). All pins exist hierarchically within a water-filled bounding component (shown in blue) which extends axially from z=-300 cm to z=500 cm, of which an axially truncated view is shown in the figure. The top and bottom of the bounding component are escape boundaries. The eight control rods can be arbitrarily moved in the axially direction. Any portion of the rods that is moved beyond the bounding component is effectively removed from the problem.

The figure shows a *xy* plot at z = 20 cm. At this axially position the eight rods are not visible since they are positioned at z= 26.95cm, which is approximately the critical position. The *xz* slice plot at y = 3.5cm shows two of the rods withdrawn. Because of the hierarchical relationship of the rods and the bounding component, as the rods are moved upward the space left behind is implicitly filled with water.

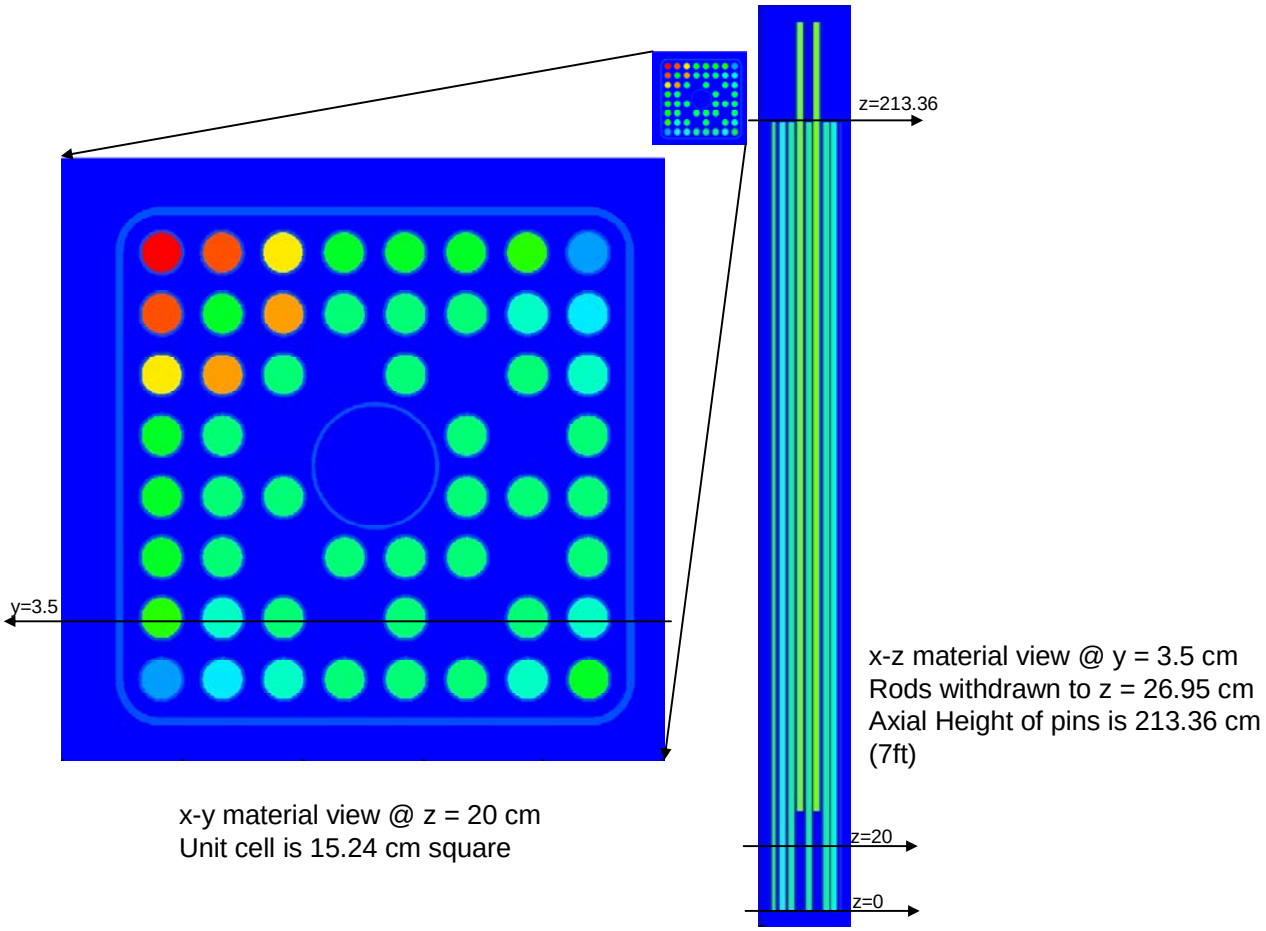

Figure 8. Modified GE-9 BWR Assembly Device Search Benchmark

The MC21 ABA was used to determine the control device sensitivity and the total number of histories required to reach a target eigenvalue solution of $k_{\text{eff}} = 1.0000 \pm 0.0005$ (95% CI). Each control device consists of a rod with surrounding cladding. The eight rod/cladding combinations make a total of 16 components. These 16 components all move together as one unit. Device sensitivity was determined by moving the components in the positive *z* direction from 0 cm to 213.36 cm, as shown in Fig. 9.

Figure 10 shows the critical control device search history comparing the ABA method to a "baseline" run using MC21. The highly precise baseline calculations required 3 iterations to converge for a total of 117 million histories. MC21 with the ABA method, resulted in the critical device position of *z* = 26.65 cm compared to a baseline prediction of *z* = 26.61 cm, which is within the uncertainty of the device worth sensitivity. Relative to the baseline calculation, the ABA case required eight iterations to achieve the target eigenvalue for a total of 33.7 million histories, or ~3.5 times fewer histories.

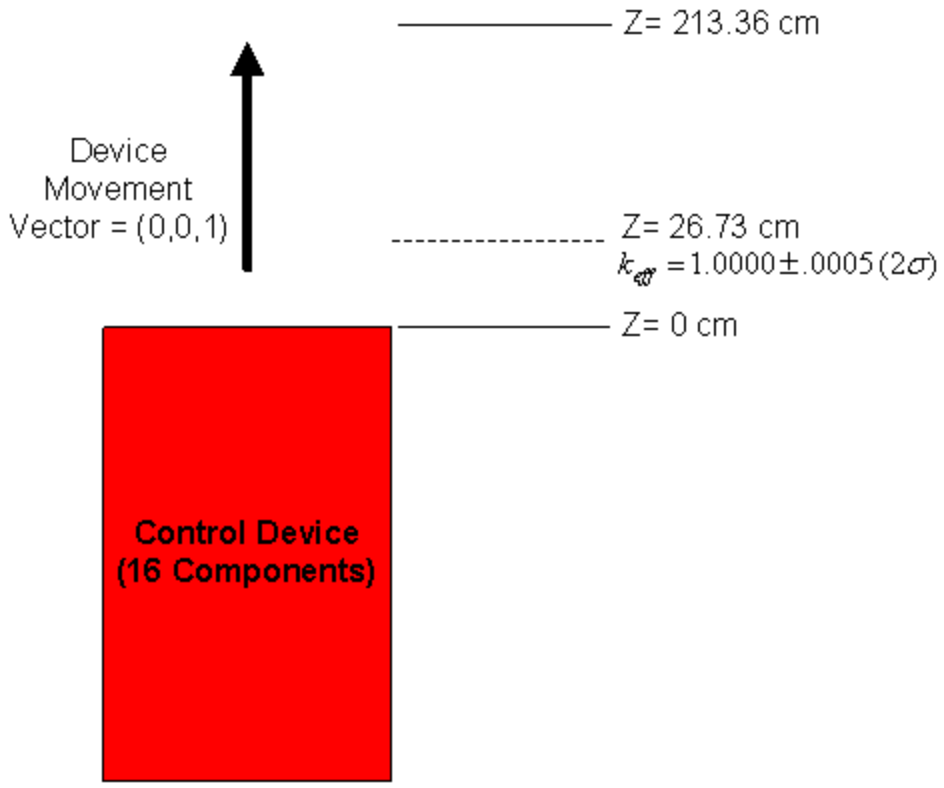

Figure 9. GE-9 Control Device Movement

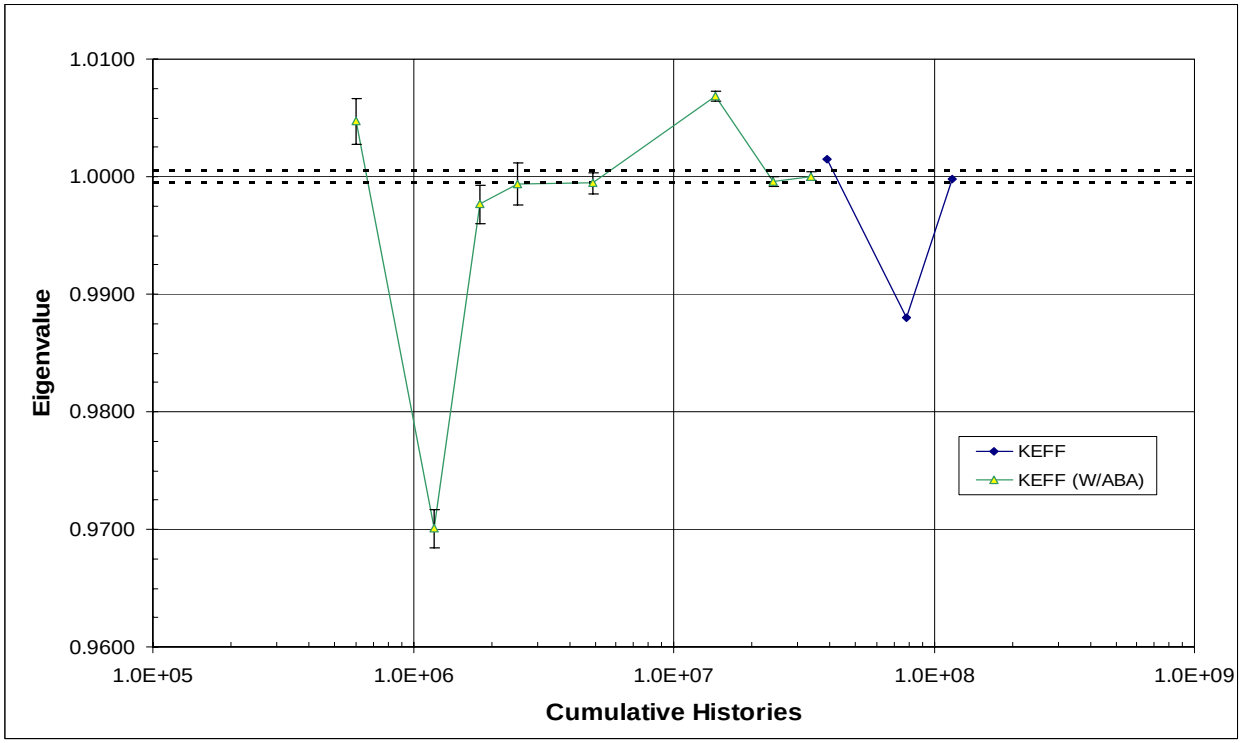

Figure 10. GE-9 Control Device Search History

4. CONCLUSIONS

A proof of concept k_{eff} search capability has been developed using MC21. This capability takes advantage of the geometric hierarchy and grid geometry constructs of MC21 to minimize the complexity potentially associated with moving large sections of geometry. The capability was demonstrated for a single large component in the HEU-MET-FAST-30 model as well as a more traditional control-device based model.

The ABA has demonstrated that significant reductions in the number of active histories (and thus runtime) can be achieved for eigenvalue search problems. By starting with relatively low-precision eigenvalue solutions and incrementally increasing the number of histories as the device converges to the desired position, the cumulative number of histories required for convergence was reduced by a factor of approximately seven for the HEU-MET-FAST-30 model and approximately 3.5 for the GE-9 model tested. In addition to reducing the total number of histories required for convergence, this method also has the advantage of appearing to the user as a single job type request because it folds an entire k_{eff} search sequence into a single job. This will eliminate engineering time associated with executing individual jobs, estimating control device moves by hand, and iterating manually.

Further testing of the ABA method is warranted to evaluate and refine the method to increase histories and calculate differential device worth. For instance, in these calculations a straight factor of 4 was used to increase the number of histories when a control device move was calculated to be statistically insignificant. More development is required before a more adaptive approach for increasing batch sizes, such as suggested by Eq. 5, can be implemented. Basing the increase in histories on the Eq. (5) caused the number of histories to grow very rapidly in some cases and produced undesirable results. Also, in this study, the relaxation parameter, ε, shown in Eq. (2) was set to 1.0. The effect of varying this parameter on convergence has not been tested to date.

REFERENCES

1. T. J. Donovan and L. Tyburski, “Geometric Representations in the Developmental Monte Carlo Transport Code MC21,” *Physor-2006 Topical Meeting*, Vancouver, BC, Sept. 10-14 (2006).
2. X-5 Monte Carlo Team, “MCNP – A General Monte Carlo N-Particle Transport Code, Version 5”, LA-UR-03-1987 (2003).
3. NEA/NSC/DOC(95)03, *International Handbook of Evaluated Criticality Safety Benchmark Experiments, NEA Nuclear Science Committee* (2003).