# Using Abaqus to Enable Accurate Stress Predictions of a Multi-Body System with Sliding Contact

George G. Antoun, Clark Briggs

ATA Engineering, Inc.

***Abstract**: There are many mechanical systems that can be classified as a combination of flexible components and mechanisms; a few examples are gear systems, aircraft landing gear, and systems including linear bearings. Historically, dedicated multi-body dynamics (MBD) codes were used for the loads development in these systems, with subsequent detailed stress evaluations performed on the individual parts. Several problems with this approach are that it tends to greatly overpredict loads, leading to overly conservative designs; it neglects the coupling of component flexibility; and also it necessitates maintaining two different models, which may be difficult in a dynamic design environment. Dedicated MBD codes have capabilities to include modal representations of bodies that can better represent component flexibility, and these models are sometimes used for stress predictions. While this approach has the benefit of representing component flexibility and therefore producing better system-level loads, the displacement and stress predictions from the system-level models are often inaccurate due to the inherent limitations of a modal representation.*

*This paper explores the problem of a multi-body articulating mannequin carried on a flexible linear motion base stage with sliding contact to examine the benefits of a single all-FEM approach. The sliding contact in the linear stage cannot be well represented by a modal body but has been modeled in Abaqus to simulate the response of this complex dynamic system. A detailed description of the model is included, along with specific design recommendations to enable this modeling approach to work in a highly dynamic design environment.*

***Keywords**: Abaqus/CAE, Abaqus/Standard, Abaqus/Explicit, Flexible Multi-Body Dynamics, MBD, Mannequin, Sliding Contact, Connectors, Mechanism, Kinematics, Display Body, Rigid Body.*

## 1. Introduction

Many industries use multi-body dynamics (MBD) models in the design and analysis process to explore the behavior of complex mechanisms and derive component loads for systems such as suspension and gear-train assemblies in the automotive industry, solar array deployments and fairing separations in the aerospace industry, and complex gear systems for various industrial applications. Loads from these mechanism analyses are used to size actuators and motors and are often fed to detailed stress analysis models to assess structural integrity of components. If detailed

stress analysis shows undesirable results, then design changes are made, MBD models are reanalyzed, and the design/analysis cycle continues to iterate until a suitable design is found. In this design/analysis scenario, each iteration requires updates to two sets of models—the MBD loads model and the detailed stress FE model—which may become cumbersome in a highly fluid design environment.

There are many commercially available dedicated MBD software codes that allow designers and analysts to very efficiently analyze complex mechanical systems to explore design spaces and perform feasibility studies. Two popular dedicated MBD codes are MSC/Adams and RecurDyn, but there are many others, and these codes all display similar strengths and weaknesses inherent to dedicated MBD codes. The fundamental building blocks of dedicated MBD codes are rigid components, so even a large and complex MBD system model may only have a few hundred degrees of freedom (DOF), allowing very fast simulation times. These codes also tend to organize the components and associated interface points in a hierarchal structure, leading to well-organized models that are easy to interpret. Many MBD codes seed the mass properties of each component off the imported geometry, so design updates can be incorporated by simply importing and replacing the geometric bodies. And since the geometric body is a subfeature of each component (as opposed to being the owning body), updating the geometry will not disrupt the entire model. This simplifies design updates in a highly fluid design environment.

However, the advantages and simulation efficiencies of dedicated MBD codes come at a steep cost. Using rigid components in dedicated MBD codes does allow very efficient simulation times, but rigid components can be very limiting, as many designs are not well represented by rigid components. Using dedicated MBD codes with rigid components can also overpredict component loads, leading to inefficiencies in the design process. While rigid component representations are good enough to represent some components, for many components rigid representations are inappropriate. The capability to add modal representations of components has been adopted by most major dedicated MBD codes, but modal representations presume that component deformation is well represented by extrapolating the small-deformation response, potentially leading to inaccurate response and stress predictions. Modal representations are incapable of representing component nonlinearities, ruling out any accurate MBD representation of nonlinear elastic (hyperelastic seals, gaskets, etc.), plastic, or post-yielding behavior. The only structural nonlinearities that can be incorporated into dedicated MBD codes are discrete, two-point springlike elements.

Dedicated MBD codes generally allow contact and friction between components; however, the contact enforcement method is limited to a rudimentary penalty-stiffness method whose stiffness and damping characteristics are somewhat arbitrary. Sensitivity studies and component tests may be required to determine appropriate contact parameters to accurately resolve peak stress/force. Some dedicated MBD codes do allow contact between modally flexible components, but this flexibility is based on component modes, so stress predictions are only appropriate for small deflections. Mode-based stress predictions also cannot accurately recover stress at the contact point unless the location of contact is known a priori. For example, MSC/Adams does allow general contact using modally reduced flexible components; however, it is known and even stated in the marketing materials that this type of contact should only be used where the user is “interested in loads, but not as interested in stresses.”

The limitations and disadvantages of dedicated MBD codes described above may exacerbate the challenges of a working in a highly dynamic design environment with tight schedule constraints. For each design revision, the MBD models must be updated to generate loads that are then passed to the stress analysts to determine structural margins. All-rigid MBD models generally provide very fast simulation times, but the loads may be inaccurate because the model neglects the contribution of component flexibility. Including modally flexible components into the MBD model would likely improve the accuracy of the loads predictions at the cost of added complexity and solution time as well as the requirement to maintain a dynamic FE representation of the flexible component for modes calculation. But stress predictions from the flexible MBD component cannot be trusted, so separate detailed stress FE models will still be required to determine structural integrity. Either method requires maintaining separate MBD and detailed FE models, necessitating different models on different software platforms and tight revision control to keep both models up to date. Dividing the loads and stress analysis into different groups, as many organizations do, can thus make maintaining version control between the various models a difficult and frustrating experience.

### 1.1 Objective

The objective of this work is to demonstrate the use of Abaqus to simulate the kinematic operation of a MBD model that includes sliding contact on a flexible component while simultaneously allowing accurate stress predictions. Performing all simulations within a single Abaqus model allows the same environment to be used for the loads predictions on actuators and motors while also enabling accurate stress recovery on flexible components. This will inevitably simplify the process of keeping up to date in a very fluid design environment, since there will be only one model to maintain. And, unlike dedicated MBD codes, no special treatment is required for nonlinear material behaviors or flexible components, since these are native Abaqus capabilities.

This process will undoubtedly require longer simulation times than the corresponding dedicated MBD code since detailed FE model representation of components may be included, dramatically driving up the number of DOF. Contact is generally more expensive in an FE environment than an MBD environment, as well, since FE models can enforce contact more rigorously. However, the added simulation time of the proposed approach will be offset by a great simplification in the design/analysis process. The single Abaqus simulation will provide the necessary loads and stress analysis results, eliminating the need for a loads analysis followed by a subsequent detailed stress analysis. This may also compress the schedule required to assess each design iteration, since the loads and stress evaluations can be performed in parallel using Abaqus instead of being done in series. This single-model approach also eliminates many of the risks associated with maintaining separate MBD (loads) and FE (stress) models for the current design.

This paper demonstrates the use of Abaqus by first describing in great detail the modeling of a dancing mannequin on a base with sliding contact. Next some sample results are presented, along with a sensitivity study on the various modeling techniques that could be employed. Finally, several specific modeling recommendations are given to ensure that the Abaqus model can be quickly adapted to a rapidly changing design.

## 2. Sample model

The requirements of this demonstration model were derived through discussions with several customers who work in highly dynamic design environments, and this public-domain model was created to include many features of interest to those customers to demonstrate the capabilities of an all-Abaqus approach while exploring modeling best practices. The following sections provide specific modeling recommendations to enable the Abaqus model to be easily updated to accommodate a rapidly changing design.

An overview of this model is shown in Figure 1. The mannequin, stem, and sliding base comprise 31 rigid parts connected through a set of articulating joints using connector elements. The rail on which it slides is a detailed, flexible finite element (FE) representation, and contact is enforced between the mating faces of the rigid bearing blocks and flexible T-slots on the rail. It is assumed that the user of this model wishes to determine motor torque requirements to enforce a choreographed dance while also minimizing the weight of the rail, all while maintaining positive margins on the rail.

After some experimentation to determine the best modeling approach for the components and joints, creating all component and joints (connector elements) for the entire mannequin required 6–8 work-hours using Abaqus/CAE. Special care was taken to give detailed names to all reference points, coordinate systems, lumped masses, and assembly wires to make future selections intuitive. Adding logical names certainly added some time to the model generation task; however, it greatly improved model organization and simplified model debugging.

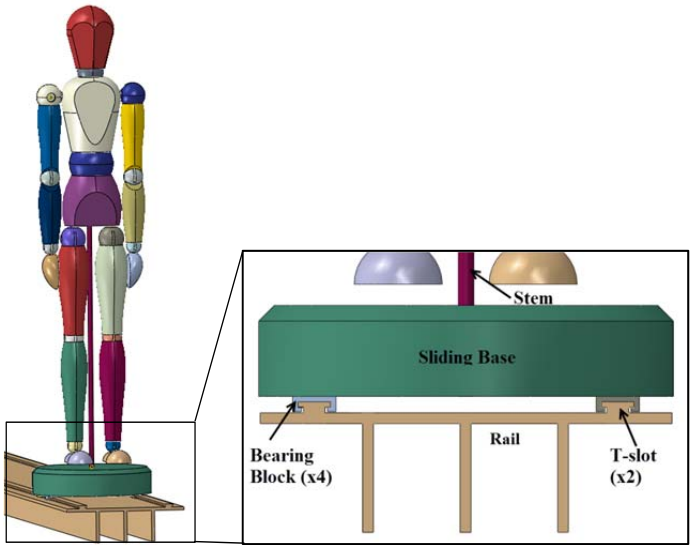


**Figure 1. Overview of mannequin, sliding base, and rail.**

## 2.1 Mannequin component modeling

The first version of this model used *DISPLAY BODYs for all rigid mannequin components to minimize the run time. (The relative cost/benefit of this approach versus a detailed FEM and *RIGID BODY is discussed later.) Since *DISPLAY BODY constraints are useful for display only, the mass and inertia of each component were represented with a lumped mass/inertia element placed on a reference point at the center of gravity (CG) of each component. The mass properties for each component were computed from the volume of the imported CAD, but they could easily be updated to represent any additional nonstructural hardware present in a design. The mass/inertia element was connected to reference points at each component's interface points using a *COUPLING, *KINEMATIC as shown by the bold red line in Figure 2. Notice in this figure that all reference points have an intuitive naming scheme: *RP_L_Upper_arm_LM* is the lumped mass location, *RP_L_Upper_arm_down* at the upper arm/shoulder interface is on the downstream side of the shoulder joint, and *RP_L_Elbow_up* is on the upstream side of the elbow joint. This process was repeated for each component of the mannequin and sliding base, for a total of 31 components.

In order to create a unique *DISPLAY BODY for each instance of a part, each instance was made independent in Abaqus/CAE and the display body was forced to follow the motion of the CG reference point.

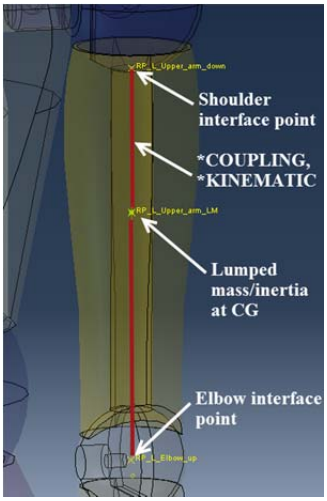


**Figure 2. Sample component model with a lumped mass/inertia at CG and *COUPLING, *KINEMATIC. For use with *DISPLAY BODY.**

## 2.2 Articulating joint modeling

To enable each mannequin joint to be articulated to follow the choreographed dance routine, connector elements were created at each joint using coincident reference points. Because all

connector elements are created between coincident reference points, it was important to give each reference point an intuitive name to make clear which reference point was connected to which component CG.

All joints in this model were represented with hinge-type connector elements defined to rotate about the local X-axis; therefore each joint also required its own coordinate system. Each coordinate system was created coincident to the connector element to simplify the organization of the model, as shown in Figure 3. This figure shows two coincident reference points (*RP_L_Elbow_up*, which is connected to the upper arm lumped mass, and *RP_L_Elbow_down*, which is connected to the elbow lumped mass), a wire between the reference points, and the coordinate system defining the hinge axis.

All hinges defined in this model are simple hinges: all translations and the rotation about the location Y and Z axes are rigid, and no constitutive relationship is defined about the location X direction. The motion about X of each hinge was defined with a *CONNECTOR MOTION boundary condition specifying the rotational time history to enforce the desired motion.

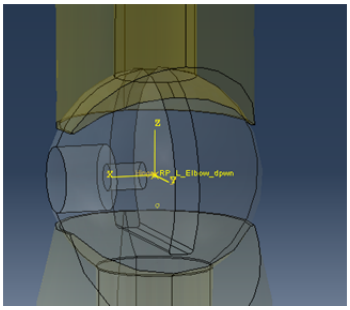

**Figure 3. Sample joint modeling showing coincident reference points, assembly wire, and coordinate system defining connector orientation.**

## 2.3 Bearing block and rail modeling

Four bearing blocks connect the rigid sliding base to the flexible rail through contact along two T-slot connections. A detailed mesh with structural elements was required for each bearing block to enable contact between the mating faces. Each bearing block was meshed with solid elements and made rigid using four *RIGID BODY constraints, and the reference point of each bearing block was connected to the sliding base CG reference point using a *COUPLING, *KINEMATIC.

The rail is the only flexible member in the model. The T-slots were modeled with solid C3D8R elements and the rest of the rail has been midsurfaced and meshed with S4R elements. Representations of these bodies, with and without the mesh, are shown in Figure 4. The rail uses aluminum material properties.

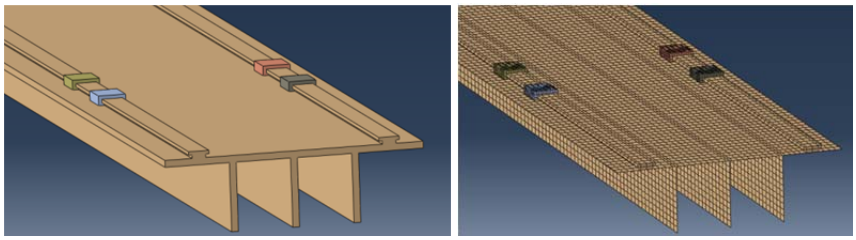

**Figure 4. Bearing blocks and rail models with and without mesh.**

Contact between the bearing blocks and rail T-slots was enforced using the general contact algorithm, but face pairs were manually created to reduce the size of the contact domain. Hard contact was defined and exponential contact damping was specified to mitigate contact chattering. Since the bearing block/T-slot interface is representative of a bearing system, no friction was applied.

The 240" long rail is clamped to ground at both ends.

## 3. Results

A series of simulations was performed to compare run times and results. Each simulation had the mannequin follow the same 15 second choreographed dance by enforcing the joint motions using a set of *CONNECTOR MOTION boundary conditions. While the mannequin was performing its dance, the sliding base was enforced to move 150" along the rail over the 15 seconds at a constant velocity. The leaning of the mannequin was enforced with a hinge connector and enforced connector motion between the sliding base and the stem.

Frames showing the first 7 seconds of the routine can be seen in Figure 5 (the sliding base and bearing blocks are removed for clarity). Element lines are also removed. The final 8 seconds of the simulation has the sliding base continue to traverse the rail while the mannequin goes through another cycle of its dance routine. These results clearly show the progression of stress, driven by the bearing block/T-slot contact, on the flexible rail as the mannequin slides along the T-slots. Notice in particular that the local stress hotspots at the bearing block/rail interface is captured by this simulation, a feature that is not possible with a dedicated MBD approach with modally flexible components. While this model does not excessively stress the rail, there are no limitations in the material behavior so nonlinear material effects could easily have been included if they were relevant.

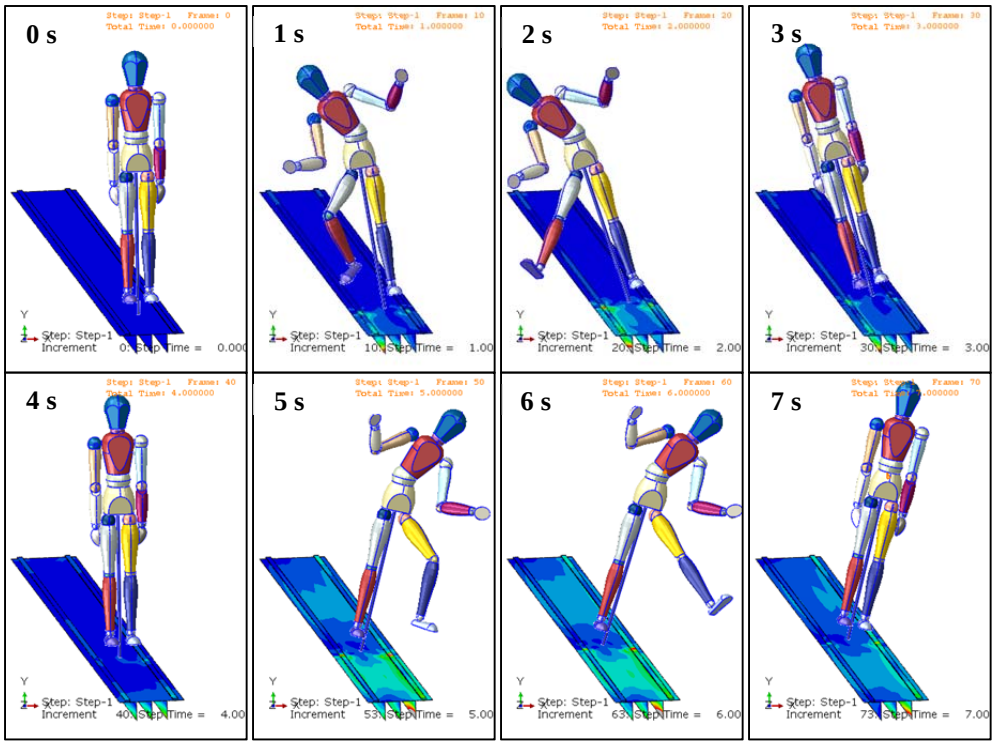


**Figure 5. Time slices showing displacement and rail stress through the choreographed motion (base and bearing blocks removed for clarity).**

For comparison, another version of this model was also simulated with a rigid rail representation, eliminating the only flexible component. This version of the model is comparable to what would traditionally be thought of as a pure MBD problem that would be solved in a dedicated MBD code. The mannequin used the same choreographed motion; however, the rail was completely eliminated and the sliding base was restrained in the vertical and lateral directions and in all rotations. The sliding base was still enforced to traverse 150" in 15 seconds at a constant velocity. With a rigid rail, the bearing block/rail contact was not necessary since the four contact points formed a statically indeterminate set of connections. The best that could be obtained from the bearing block/rigid rail connection is the net reaction forces and moments, so it seemed appropriate to eliminate the rail entirely and recover the reaction forces and moments.

### 3.1 Solver selection, run time, and results

This model does not include any modeling features that rule out either the /Standard or /Explicit solver, so a decision was required. There are a variety of considerations when deciding to use the /Standard or /Explicit solver, such as the speed of the loading, the structure's dominant response frequency, the complexity of the contact interfaces, and degree of material nonlinearity. The

addition of flexible components into a rigid mechanism model, however, has no repercussions for which solver should be selected, so for this case the best solver was selected using the logic discussed in the Abaqus User's Manual: the loading of this structure is relatively low frequency and the response of the flexible rail is expected to be less than 100 Hz, so a /Standard simulation was used. The extremely small stable time increment that would be required for /Explicit with a stress-model fidelity mesh of the rail made this a less palatable option, even considering that contact is more expensive in a /Standard simulation.

For the no-rail configuration, there are no flexible elements so the time-step size of an /Explicit simulation could be manually specified, making this an attractive option. For a set number of iterations, /Explicit simulations tend to be faster than /Standard simulations. Because it was very easy to switch between the two solvers, the no-rail configuration was simulated with both solvers.

Table 1 shows the run time and number of iterations for thee simulations: the flexible rail case with /Standard and the no-rail case with both /Standard and /Explicit. All simulations were performed on a high-end desktop computer with ten CPUs using a maximum time-step size of 0.02 seconds. A sensitivity study on the output time step revealed that further reductions in the time-step size did not change the output response. Geometric nonlinearity was required because the joints go through large rotations. All simulations were performed using Abaqus 6.12-2.

**Table 1. Run-time comparison for different configurations and solvers.**

| | | # Iterations | | | |
|---|---|---|---|---|---|
| Configuration | Solver | Severe Discont. | Equilib. | Total | Wallclock, min |
| Flexible Rail | /Standard | 4203 | 663 | 4866 | 144 |
| No Rail | /Standard | n/a | 1499 | 1499 | 5 |
| No Rail | /Explicit | n/a | n/a | 750 | 3 |

The two runs without the rail only required 3–5 minutes of run time versus almost 2.5 hours for the case with the flexible rail. This is not surprising since the flexible rail configuration contains several orders of magnitude more DOF and also includes contact. Notice that most of the iterations of the flexible rail configuration are severe discontinuity iterations that resolve the contact status.

Despite the high cost of the flexible-rail simulation, it is well worth the additional simulation time because of the increased accuracy in actuator loads and insight into the rail stress. The actuator loads that are extracted from each joint include the coupling of a flexible rail, which can have a dramatic effect. Figure 6 shows a sample comparison of the sliding base/stem moment requirement from the various simulations. Data point symbols on this figure are only shown at a frequency of 5 for clarity, so for every data point shown there are four points not shown. The two rigid rail simulations show excellent agreement, as expected.

The flexible rail simulation shows significant coupling of the rail flexibility with the mannequin motion, evidenced by the large amplitude oscillations beginning at about 5 seconds. Notice in particular that the frequency and amplitude of the oscillations tend to increase as the mannequin approaches the center of the unsupported rail. Neglecting the rail flexibility would not only lead to

an undersized actuator but would also neglect the oscillating loads on the motor and bearings, which will affect life predictions. Of course, the degree of coupling will vary across the joints and will be very dependent on the model and design; however, this modeling and simulation approach is sufficient to capture those effects, enabling more accurate assessment of the motor and actuator requirements.

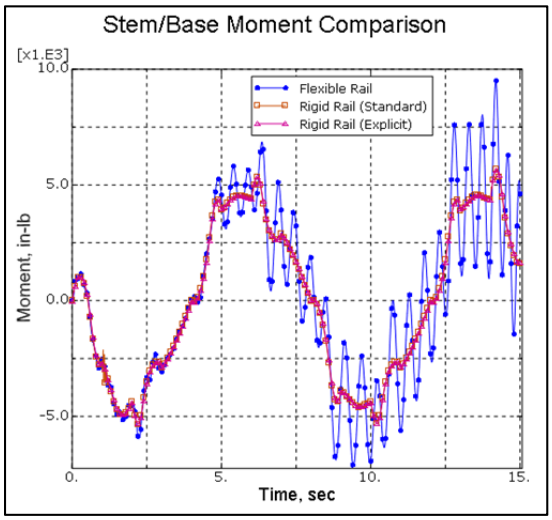


**Figure 6. Comparison of sliding base/stem moment requirement for flexible rail versus rigid rail (data points symbols shown with frequency = 5 for clarity).**

The results of the flexible-rail simulation can also be used to assess the structural integrity of the flexible rail directly. If the no-rail simulation is used to determine the joint and rail loads, additional work-hours would be required to determine how to envelope the loads and apply them to a detailed stress model of the rail to assess structural integrity, and this can be a very time-consuming task if the model is complex. By contrast, the only work-time required to assess the structural integrity of the rail in the flexible-rail simulation is the time to review the results of the dynamic simulation, which are more accurate since the all-Abaqus simulation includes the rigid body/flexible body coupling. And since human-time is more expensive and in shorter supply than computer time, using the more computationally expensive flexible rail simulation will generally enable faster design evaluations despite the longer simulation time.

With the entire system modeled in a single file and a single platform, it will also be much easier to maintain version control; maintaining a separate loads model of the mechanism and a detailed FE model of the rail as two separate models, while also using different software packages, can be challenging in a highly dynamic design environment. Using a single system model of the mannequin and flexible rail will make it much easier keep all aspects of this model current to the design.

### 3.2 Alternative component modeling using *RIGID BODY

The component modeling scheme described in section 2.1 represents the most efficient way to model the components in terms of run time: each component is modeled with a lumped mass and a *COUPLING, *KINEMATIC attaches the lumped mass to the interface reference points, and the CAD is attached to the lumped mass for visualization purposes only using *DISPLAY BODY constraints. But while this modeling scheme enables the fastest run times, it also requires each lumped mass and CG to be manually created by the analyst. For models with large numbers of bodies, this can require significant time to set up. The benefit of the lumped mass approach for rigid bodies, however, is that it is very easy to update the mass properties and CG location with absolutely no disruption to the model.

An alternative component modeling scheme is to include a FEM representation of each mannequin component and then rigidize the component using *RIGID BODY statements. The TIE NSET field can be used to connect the interface reference points to each rigid body. This modeling scheme is simpler to set up because the mass properties of each component do not need to be explicitly defined by the analysts; rather, they are determined by the software from the meshed volume and material density. This approach would also be preferable if the analyst plans to eventually convert a rigid body into a flexible body. The *RIGID BODY component representation also enables contact between the components if that is of interest. The tradeoff is that run time is increased.

To examine the difference in run time, each mannequin component was meshed and rigidized using *RIGID BODY constraints. Very coarse meshes of linear elements were used since the mesh on each component was only intended to fill the volume and represent the mass properties. A total of 5,422 nodes and 18,213 elements were used for the mannequin representation.

**Figure 7. Components meshed coarsely and rigidized with *RIGID BODY.**

The same three simulations were analyzed using the same desktop computer, run parameters, and output times as discussed previously. A few spot checks confirmed that using the *RIGID BODY modeling scheme produced the same results as the *DISPLAY BODY method. The total run time and number of iterations of each run are shown below.

**Table 2. Run-time comparison for different configurations and solvers using *RIGID BODY components.**

| | | # Iterations | | | |
|---|---|---|---|---|---|
| Configuration | Solver | Severe Discont. | Equilib. | Total | Wallclock, min |
| Flexible Rail | /Standard | 4083 | 674 | 4757 | 170 |
| No Rail | /Standard | n/a | 1694 | 1694 | 22 |
| No Rail | /Explicit | n/a | n/a | 750 | 9 |

Using *RIGID BODY representations of each component dramatically drives up the run time for each configuration even though the number of iterations does not change substantially. Despite the increased run time, it is up to the analyst to balance the extra run time of the *RIGID BODY modeling method against the extra setup time of the lumped mass and *COUPLING, *KINEMATIC method described in section 2.1. A mix of lumped masses/*DISPLAY BODY component modeling and detailed meshes/*RIGID BODY may be appropriate if some parts of a design are more mature than others.

## 4. Modeling recommendations for dynamic design environment

The preceding section shows that solving this class of problems with Abaqus is possible and details the types of outputs that can be obtained. From a single simulation, accurate actuator loads that account for coupling with any flexible or nonlinear components can be recovered to enable motor sizing while at the same time recovering time-accurate stress results on the flexible components for structural margin calculations.

In order for this modeling approach to be a feasible solution for the highly dynamic design environment characteristic of many industries, the model must be able to be quickly updated with minimal rework. As the most time-consuming part of building this and many other complex models is creating the interactions and constraints that tie the various instances together, careful planning using some of the modeling best practices described below can facilitate rapid model updates while minimizing the amount of rework.

### 4.1 Use intuitive names to all modeling features

Managing large, complex files with many instances, constraints, and joints can be difficult in any software, and Abaqus/CAE is no different. As an example, the flexible-rail model described in Section 2 contains the following:

- Thirty-one mannequin components (instances), each requiring a lumped mass, *COUPLING *KINEMATIC, and *DISPLAY BODY

- Twenty-five connectors representing the joints, each requiring its own coordinate system and wire
- Four *RIGID BODY statements for the bearing blocks
- Eighty-five reference points to represent the various connections and lumped masses

To efficiently manage this model, it was necessary to give each modeling feature an intuitive name so that modeling features could be quickly found and edited as required by design changes. This is especially necessary because Abaqus/CAE does not allow selection of modeling features such as constraints or lumped masses from the graphics window. The only way to select and edit a *COUPLING constraint, for example, is to select that item from the list in the model tree or the Constraint Manager form. Although giving detailed names will add some time to the initial model development, that investment in model organization will pay dividends when model updates are required.

It is also helpful to keep in mind that the assigned names will be sorted alphanumerically, so a naming convention should be used to help group similar features. For example, the naming convention shown in Figure 8 has been chosen to group the *COUPLING and *DISPLAY BODY of each mannequin component together, improving organization and speeding up future edits.

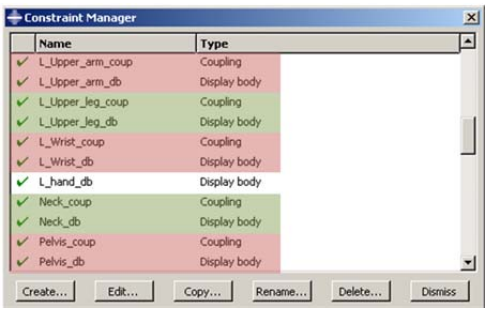


**Figure 8. Constraint manager form showing naming convention that groups similar features.**

## 4.2 Use sets to minimize graphical selection

Making seemingly minor model changes can cause significant disruption to a complex mechanism model such as the mannequin if some simple guidelines are not followed. Most mechanism models include numerous joints that are represented in Abaqus using connectors and assembly wires, so it is critical that these joints and the supporting modeling features are not invalidated by design changes. The best example of this is the assembly wires that are used to create each connector. Each wire is assigned to a connector section that defines its behavior, a boundary condition that defines the connector motion time history, and a history output request. If the wire is selected from the graphics window for each of these assignments and the wire is then changed, these three features become invalid and each would need to be updated.

A better approach is to use named geometry sets as much as possible. For this model, each wire was placed in an appropriately named set and then the connector behavior, connector motion, and output history request all referred to that set name. With this approach, if a wire is modified or must be deleted and recreated, only the geometry set would become invalid and need to be updated to include the new wire. Since the other modeling features (connector behavior, connector motion, and history output) all refer to the set name, once the set's contents are updated that change will propagate to all other modeling features referencing the set name. The same guideline could be established for surfaces that are referenced by *COUPLING, *TIE, or *CONTACT definitions, as well as surfaces, edges, or vertices that are assigned boundary conditions or loads.

### 4.3 Avoid using geometric vertices for reference point or coordinate system creation

When working in a highly dynamic design environment, it is likely that the initial version of CAD used to build a model will change as the design matures, requiring suppressing or deleting instances and replacing them with the most recent design. This type of change will no doubt require some rework as *DISPLAY BODY or *RIGID BODY statements are updated; however, the amount of rework can be minimized if reference points and coordinate systems are kept independent of the CAD geometry. If reference points or coordinate systems are created by directly selecting geometric vertices, these features will become invalid when an instance is replaced, requiring significant rework to correct the model.

A better approach for creating reference points and coordinate systems is to create them in such a way that they are independent of CAD-based vertices. For coordinate systems, this requires creating datum points by coordinates for the origin, X axis, and Y axis of each coordinate system. When the CAD is updated, the datum points and coordinate systems will not be affected, and the orientation of the coordinate system can be quickly changed by simply modifying the coordinates of the datum points.

Similarly, reference points should also be kept independent of the geometric vertices by either manually typing the reference point coordinates or by placing the reference point on a datum point that has been created by typing in coordinates. Ensuring that reference points are independent of the geometry is especially important since assembly wires and connectors require reference points, and recreating or modifying multiple assembly wires can be time consuming.

## 5. Conclusions

This investigation has demonstrated that Abaqus can be used to simulate a class of flexible MBD problems that include sliding contact. With this all-FEM approach, accurate stress predictions and actuator sizing that includes the effects of flexible-body coupling come from a single model, which is especially advantageous in a fluid design environment. Specific recommendations were also provided to ensure that this model can be quickly updated to new designs with minimum rework.

## 6. Acknowledgements

The authors would like to thank Dan Champion (ATA), who helped develop this model, and Laura Hoffman (ATA), who also developed this model and whose choreographic skills produced the mannequin's dance. Great thanks also go to user "Tony" on grabcad.com, who allowed the use of the mannequin CAD.