Metaform Boundary Conditions

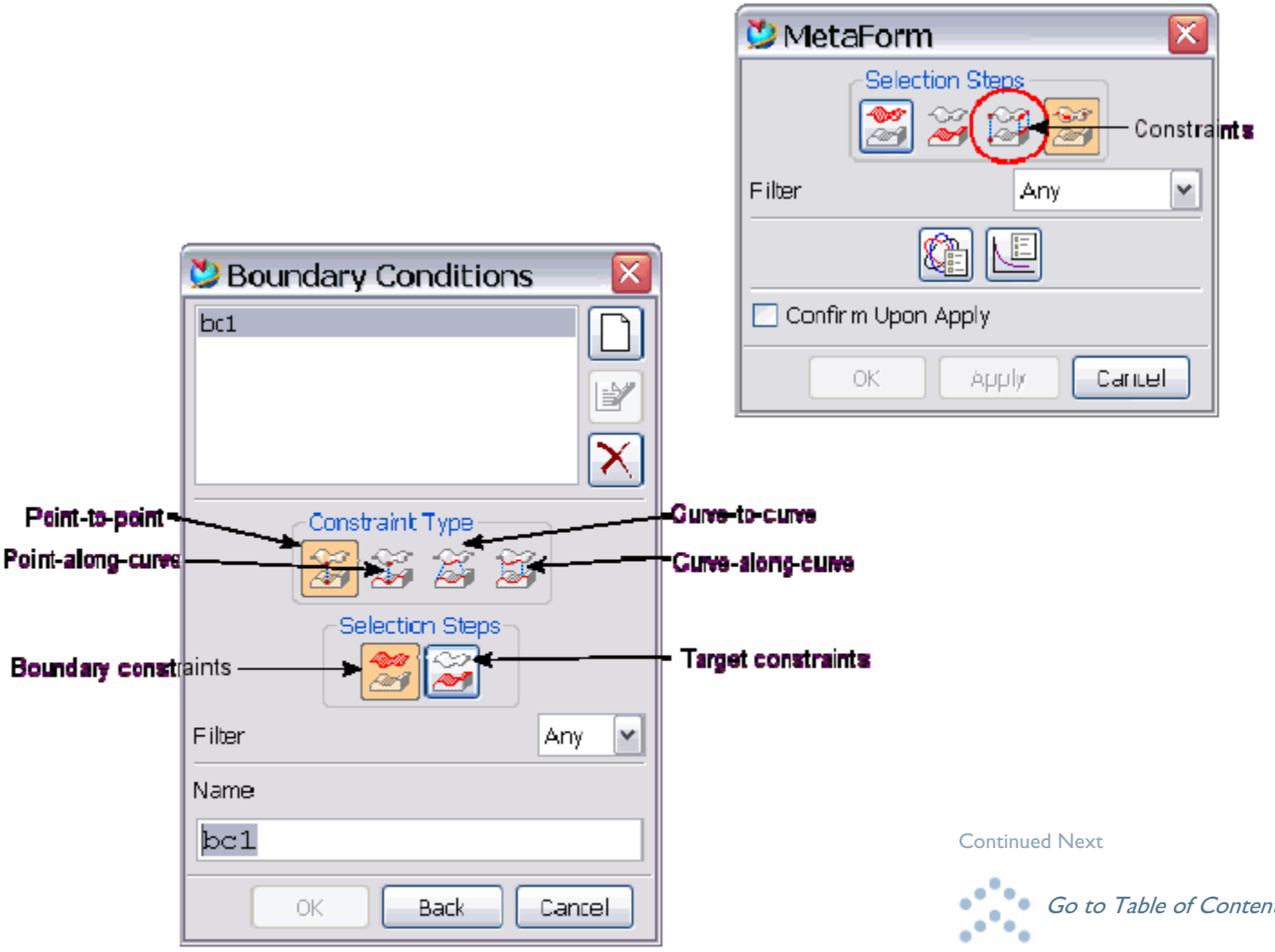
In this article, we will discuss the Boundary Condition Constraint options used in the building of a Metaform feature and conclude with some "Tips and Tricks" that may further assist you when building this feature.

The Metaform feature in NX is used for forming and flattening of bodies from one shape to another. The feature was originally created for use with complex sheet metal forming, but has been used successfully on parts that were not intended to be manufactured with sheet metal. (See *Thinking Outside the Box Using Metaform*, *GTAC Quarterly,* 2nd Quarter, 2005.)

The usual workflow for a Metaform operation is to start with a fully formed part and apply a Metaform operation in order to calculate the flat blank shape. However, Metaform is more than just a flattener and can also be used to form a part from one formed state to another, or even to take a flat blank and form it to another surface. Metaform accomplishes this by creating a triangular mesh across the original part and then deforming that mesh to match the target surface. The mesh is made up of ***nodes*** which are discrete points along each edge and through each surface with ***elements*** extending between each of the nodes. Each node in the mesh is pushed around through a series of mapping algorithms until each element and node of that mesh resides on the target surface, and the internal strains in the material are minimized. The ***boundary condition***, or ***constraint***, system in Metaform places limits on the movement of certain nodes. For example, if you select a curve-to-curve boundary condition, Metaform will take every node that is assigned to a point along the curve constraint on the region and place that node at a position along the curve that was selected as the target constraint. Those nodes will not be allowed to travel off of the target constraint curve. This document outlines the different constraint options, how they work, and what you can expect for results. (The terms ***boundary condition*** and ***constraint*** are used interchangeably in this document.)

Metaform Constraints Overview

The Metaform system has four different types of constraints. The constraints are part of the selection steps that a user works through when using the Metaform feature. The selection step and the constraint methods are shown in the dialog pictures below.



Continued Next

Go to Table of Contents

>> NX

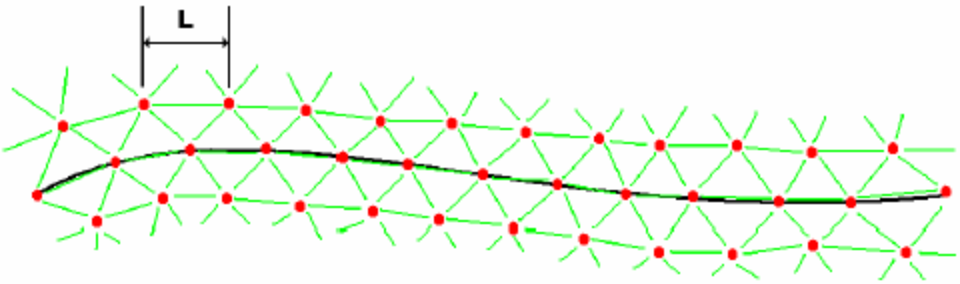
Continued from previous

In NX 3 or later, you can leave the Target array empty and the system will duplicate the items in the Boundary array for use as items in the target array.

Using Points and Curves as Constraints

In Metaform, each point used as a constraint is matched to the closest node in the Metaform mesh. This mesh node is then matched to the corresponding point on the target surface during the mapping phase of the analysis. The elements attached to the point are then moved around as the mapping algorithms are applied, but the point will remain as static as possible on the target. Each curve used in a constraint is sampled and the points along the curve matched to the nearest node in the mesh. The number of node points applied to sample a curve is directly related to the Length Tolerance given to Metaform. The length tolerance specifies the maximum distance between nodes in the mesh, so this value also indirectly specifies the number of nodes used in the sampling of the curve as shown in the picture below where the maximum distance between nodes is less than or equal to **"L".**

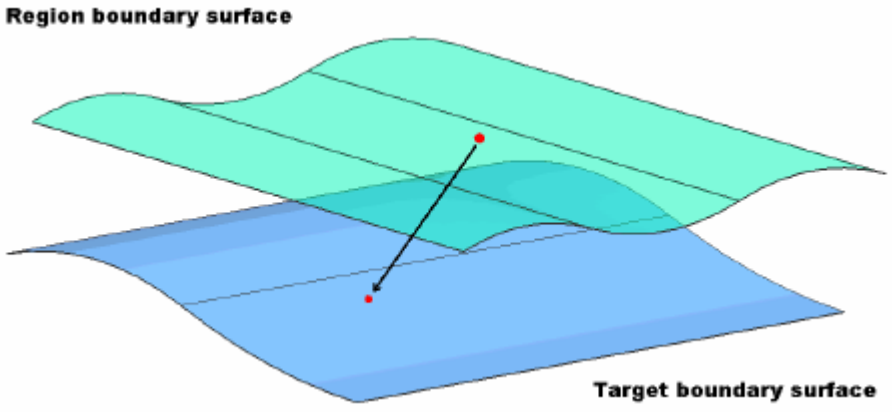
A sample mesh with the points being mesh "nodes" and the lines being mesh "elements".



The node points along the curve are matched to points along the target curve in a curve-to-curve or curve-along-curve constraint.

Point-to-Point Constraints

A point-to-point constraint is the basic constraint in Metaform. This constraint simply holds a point on the region surface to a point on the target surface. For a point-to-point type constraint the region constraints and target constraints must be a matched set. For example, if there were six points selected to be in the region constraints array, then there must be six items selected for the target constraints array. One point-to-point constraint is the minimum constraint solution for Metaform to perform its calculations. In this case, Metaform would match the point on the region surface to the point on the target surface without regard to orientation. All of the deformation would flow freely around that point, but the node at that point would be held to the point on the target surface.



Continued Next

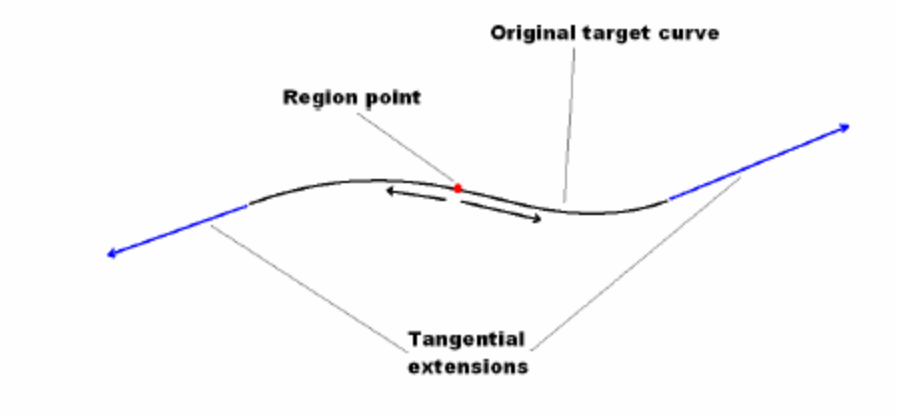
Go to Table of Contents

Continued from previous

A single point-to-point constraint is usually used when you desire a free flow of the material throughout the entire part. The only drawback to using this method is that the nodes are free to rotate around that base node which sometimes requires a lower linear tolerance. It is also worth noting that in order to avoid a singularity in the model when a single point-to-point boundary condition is used; the node associated with the boundary condition point is not allowed to rotate itself. This does not limit the other nodes from rotating around it with respect to position, but the node itself is not allowed to rotate. For a point-to-point constraint, Metaform only allows one point-point match per constraint. However the user interface does not limit you to selecting only one point. If you require multiple point-to-point constraints, be sure to create multiple individual constraints instead of selecting an array of points for the region and target constraints. There is the ability to create a point within the Metaform command by switching the filter to ***Point*** when creating constraints. Once the filter has been switched to ***point*** the point selection filters become active and you are able to create points using the normal point constructor methods in NX. This means that you are not required to exit the Metaform feature to create points for point-to-point or point-along-curve constraints.

Point-Along-Curve Constraints

The point-along-curve method allows one point on the region surface to slide along a curve or set of curves on the target surface. If a set of curves is supplied as the target constraint, Metaform will process the curves in the set in order trying to join them so the curves must be joinable. If any curves in the set are unable to be joined, Metaform will try to correct for this bad input and continue the operation by ignoring the un-joinable curves. For example, if you pick eight curves and the first six join properly but the seventh cannot be joined to the first six, the seventh curve will be ignored and Metaform will try to join the eighth curve in the set to the first six that have already been joined. If the eighth curve is also unable to be joined to the first six, it will also be ignored. Using this method, Metaform tries to recover from bad user input where the curve set is not joinable. If needed, Metaform will also extend the final joined curve in a tangential method to allow the point to move freely along the curve vector.



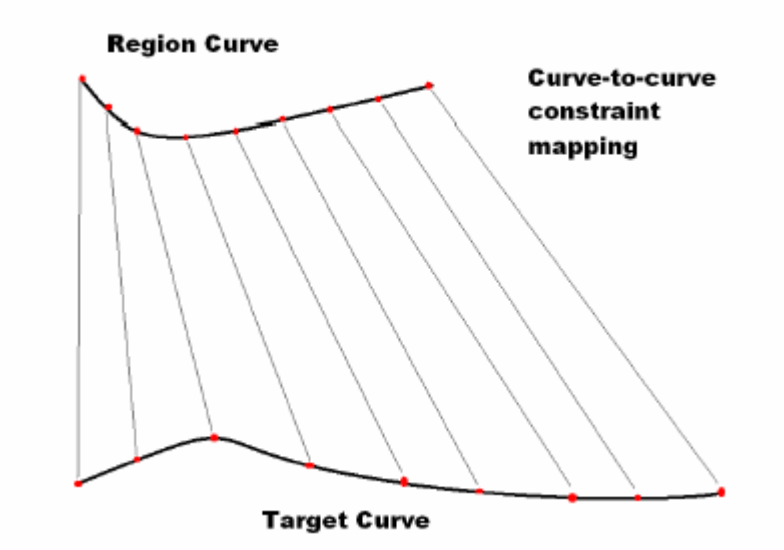
Curve-to-Curve Constraints

Curve-to-curve constraints will take a curve on the region and match the nodes from that curve to a curve on the target.

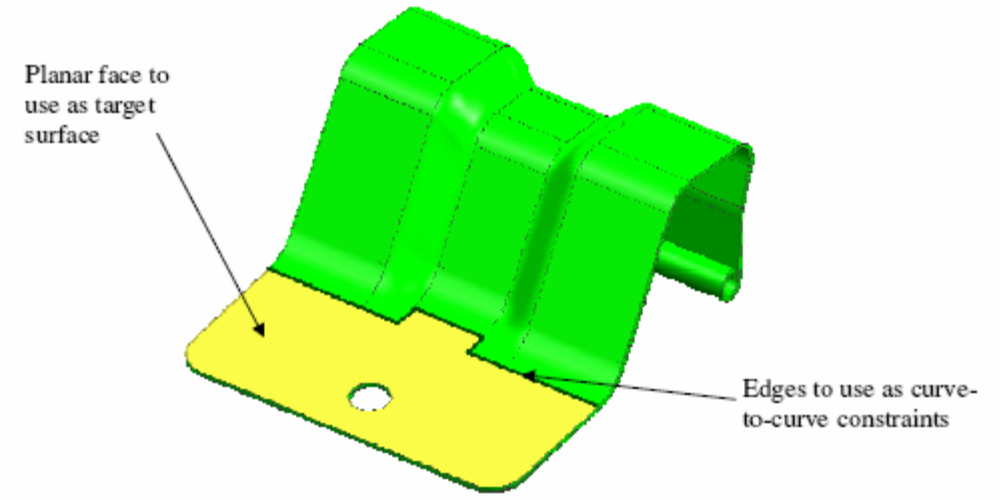
Continued Next

Go to Table of Contents

Continued from previous



Just as with the point-to-point constraint, the curve-to-curve constraints must be a matched set. If you select three curves for your region constraint, then you must select three curves for your target constraint. Once again, if you select more than one curve in a constraint set, Metaform will attempt to join the curves into a single curve as input, so all curves in the curve set must be joinable. Just like with the point-along-curve constraint, if any of the curves are not joinable, as before Metaform will attempt to work around the bad input by ignoring the extra curves. Metaform will attempt to match the region curve to the target curve by first taking the end points of the region curve and matching them to the target curve. Metaform then tries to distribute the remaining nodes from the region curve onto the target curve in the best possible fashion. This may result in a compressing or stretching of the nodes from the region curve to fit the target curve. If this results in a large amount of compression or stretching and this is not desirable, then you may wish to consider using a curve along curve constraint instead. As with a point-to-point constraint, if you desire multiple discrete curve-to-curve constraints you must create individual constraints to separate the discrete sets. Curve-to-curve constraints are especially handy for flattening models which already include planar surfaces. When you wish to flatten such a model, you can select the surfaces you want to flatten as your region and select the planar surface on the model as your target. Since the planar surface is already on the target surface, you can then select the edges of that planar face as curve-to-curve constraints for the region and target. This will hold the edges you have selected steady and move the rest of the material around the planar surface. Planar face to use as target surface Edges to use as curve-to-curve constraints.



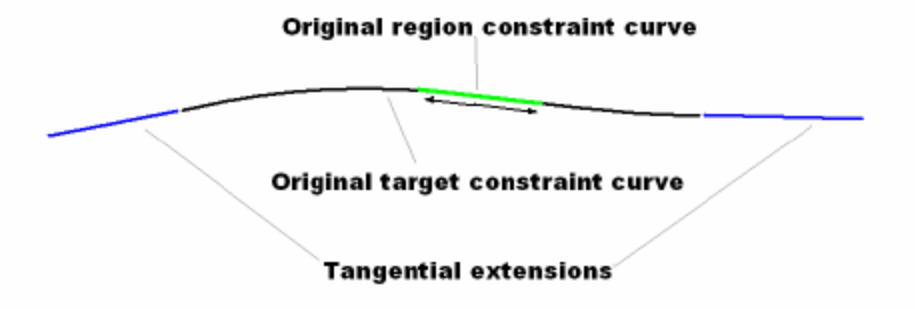
Continued Next

Go to Table of Contents

Continued from previous

Curve-Along-Curve Constraints

With the curve-along-curve method of constraint, a curve on the region is matched to a curve on the target and the region curve is allowed to slide along the target curve. This also has the effect of allowing the nodes along the region curve to freely expand or contract according to the flow of the material. When necessary, the target curve will be tangentially extended much like in the point along-curve method. The nodes from the region curve can slide off of the tangent curve in a tangential direction. Using the curve-along-curve constraint method will give you results similar to those of a curve-to-curve but the curve mapped to the target will be allowed to stretch or compress more freely according to the distortion of the material. With this type of constraint the end nodes of the region curve are not locked to the ends of the target curve, but rather are allowed to float freely along the target curve with the rest of the nodes on the region curve.



This constraint is specifically useful when you wish to constrain the flow of the material or deformation to one specific direction.

Summary

The Metaform feature in NX gives you various ways to control the flow of the given material during the forming process. Depending on your needs and the nature of the model you are operating upon, you may wish to try any of the different constraints. This document has outlined the different constraint types, described how they relate to the underlying Metaform mesh, and given suggestions for how each constraint type may be used and how they apply to certain situations that you may encounter during your use of the Metaform feature.

Bonus Metaform *Tips and Tricks*

Speed vs. Accuracy - As with most finite element tools, there is a trade-off between the speed with which the tool finds a solution and the accuracy of the result when using Metaform. The more mesh points you apply to your model, the higher degree of accuracy your solution will have, but the longer the system will take to find that solution.

> ***More mesh points = Higher accuracy & Lower performance***
> ***Less mesh points = Lower accuracy & Higher performance***

Metaform likes clean models - The cleaner your model, the better your chances of getting a good result. Look to clean up sliver faces and small gaps between the surfaces in your model. Metaform has a hard time bridging gaps in the model, and each sliver face has to be individually meshed which takes processing time.

Linear tolerance effects - The linear tolerance in Metaform is the one parameter that will have the most affect on the speed vs. accuracy equation. In general, a smaller tolerance will create a more accurate but slower solution, and a larger tolerance will do the inverse. Note however, that there is a threshold where setting the value too far in either direction for a given model size will bring diminishing returns.

Estimating the linear tolerance - As a good rule of thumb, we suggest starting with a size of 5-10 grid points across your part as a "*first guess*" for a solution. To do this, use the NX distance measuring tool (***Analysis => Distance***) to measure the length of your part at its longest distance.

Then take that measurement, divide it by a number between 5 and 10, and put that number in for the linear tolerance in Metaform. Use this as a "*first guess*" and adjust the tolerance according to the results you get.

Metaform as a flattener - The Metaform system needs a sheet body to which it will form the selected geometry. When using Metaform as a flattener, this means that a planar sheet body or face must exist in the part so that the Metaform engine can flatten to that plane. If there is a planar face on the model, that face can be selected as the target for the Metaform operation. Otherwise, the user must create a planar sheet body for the engine to flatten the part to.

Continued Next

Go to Table of Contents

>> NX

Continued from previous

Constraints, Part 1 – Usually, if you choose to use a planar face that already exists on the body as your target, you will want the distortion to flow out from that face to the rest of the part. This would be the equivalent of using that face as the start face for a flat pattern. To do this, it is suggested that you use any edges of that face as curve to-curve constraints for the solver. In this way, the distortion will flow out from that planar face, and the planar face itself will remain undistorted.

Constraints, Part 2 - If you wish to use an existing planar face, but you want distortion to be applied to that face, then it is suggested that you use a point-to point constraint and select a point on one of the edges of the face as the constraint.

Constraints, Part 3 - If you are not using Metaform to form geometry to a preexisting face, then it is required that you at least make a point-to-point constraint for the solver to use. To do this, you will need a point entity on the target body and a point entity on the region body with which to apply the constraint.

Associative Output - Since mesh analysis can be a time consuming operation, it is suggested that you give some forethought as to whether you want your Metaform output to be associative or not before you perform the operation. If you are applying the Metaform feature in the middle of your modeling operations as a "*first pass*" to get an idea of what the final solution will look like, we suggest turning off the "*associative output*" on the Analysis Options dialog. On the other hand, if you are using Metaform at the end of your modeling process and wish for the output to be regenerated if the model is updated or changed, then it is suggested that you leave this option checked with the understanding that any model updates will trigger a re-computation of the Metaform output.

Use Symmetry – If a part is symmetrical across an axis, or a plane, then you can consider using that symmetry and only performing the Metaform analysis on a fraction of the part, and then just mirroring the results to get your complete flat pattern. In this way, you can cut your analysis time into fractions. Note that this should only be done on parts where there is not much metal flow across the line of symmetry. If the line of symmetry divides a punch or other highly formed area, separating the part into pieces across that line will probably result in results that are less accurate than using Metaform on the entire part.

Multiple Metaform operations - If a part has several flanged or punched areas that are independent from the rest of the part, it is possible that these areas can be unformed individually using multiple Metaform operations. Once each area has been flattened, just join the curves for your final blank shape.

Sheet Bodies from Curves - If you need to use the curves that you get back from Metaform to create a sheet body, consider extracting the edge curves from the original geometry, joining them, and using those curves as your Metaform input. Then, once you get your Metaform output, project those curves onto the surface you picked for your target. You should then have completely joined curves on a plane that are suitable for creating a sheet body.

Metaform Notes:

1) Metaform requires an Advanced Sheet Metal license.

2) There are no facilities for a ***tipping angle*** in Metaform. Metaform uses a "*path of least resistance*" when laying the region mesh down to the target which usually means that the part mapping is applied in such a way as to maximize the parallel area between the region and target surfaces.

3) The ***Associative Output*** toggle is not currently activated. All output is associative to its base geometry. This means that any Metaform features that are not suppressed will be regenerated during the next update cycle if the underlying geometry changes.

Common Metaform Error Messages

SMD PK module received an invalid object reference - This error message relates to your boundary and target constraint sets. There must be at least as many objects in the target constraint set as there are in the boundary set. Otherwise, Metaform doesn't have enough target constraints to match the boundary constraints to. For example, if you've picked 3 curves for the boundary constraints in a curve-to-curve constraint set, then there must be at least 3 curves picked for the target constraint set.

Metaform received an invalid integration weight - This error message results when Metaform does not have a small enough granularity in the tolerance settings to find a solution to the mapping, or when the granularity is too small and results in a collapsed mesh. When this error occurs, you will usually see that the Metaform mesh has gotten grossly distorted during the calculation process. Adjusting your linear tolerance will usually solve this problem by giving the Metaform engine the granularity it needs to find a solution.

Ashley Eckhoff and Walter Schnurr

Go to Table of Contents