```
(defun C:ASSOC-SURF  (/ *error* n interval height width
                      origin-list profiles)
  (defun *error*  ()
    (cmd-out)
    (vla-EndUndoMark *aevl:drawing*))
  (vla-StartUndoMark *aevl:drawing*)
  (cmd-in)
  (if (= (getvar "WORLDUCS") 0)
    (vl-cmdf "_UCS" "_W"))
  (assoc-surf-data)
  (setq i 0)
  (repeat n
    (setq origin-list (cons (* interval i) origin-list)
          i           (1+ i)))
  (make-profiles
    (reverse origin-list)
    height
    width
    (strcat id "_PROFILE")
    id)
  (make-assoc-surf profiles)
  (mod-constraint)
  (cmd-out)
  (ax-SWt)
  (vla-EndUndoMark *aevl:drawing*))
```

Listing 20.12. Main Function C:ASSOC-SURF.

20.7 Creating a dynamic block from the associative surface.

I have already mentioned that the associativity of surfaces in AutoCAD is extremely frail. Any displacement can destroy it. A possibility in order to avoid this is including the surface in dynamic block. Dynamic blocks were introduced with Release 2006, but there is no programming interface for their creation. Although here, as in many other aspects we have AutoLISP's scripting possibilities. If it can be done manually, there are many possibilities that we can reproduce this behavior in our programs. The first step is to check if what we wish is possible.

Creating the block.

If I make an associative surface with `C:ASSOC-SURF` and attempt to create a block selecting both the surface, the cross-sections and all of its constraints, I will get a message indicating that all constraints have been removed. When inserting it we will have the surface, but its parameters cannot be modified.

But if we analyze the information on the removed restrictions we can see that the number of constraints that are reported as deleted equals the number of cross-sections times the number of geometric constraints. This leads us to suspect that at least the dimensional constraints were not lost. But the Parameters Manager's Parametric tab will not show any that corresponds to our dimensional constraints.

Make it dynamic.

Let’s do a double-click on the block. This displays Edit Block Definition dialog box with the double-clicked block’s name highlighted. We click OK to enter the Block Editor. As I suspected, in the Block Editor (Figure 20.4) the surface and its cross-sections with all their constraints, both geometric and dimensional appear.

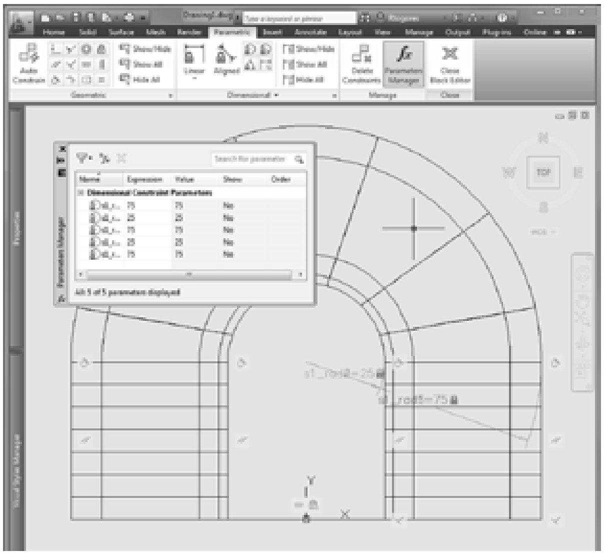
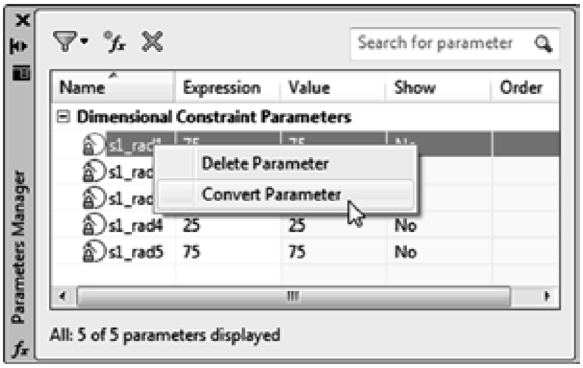


Figure 20.4. Associative surface in the Block Editor.

Then, why is it that its parameters are not available in the drawing? If I open the Block Editor’s Parameters Manager palette I find that those dimensional constraints appear as Dimensional Constraint Parameters. But now we have a column titled Show that did not appear before. And all of this column’s values are set to "No".

If I select one of the restrictions and right-click, a context menu appears (Figure 20.5) displaying the options Delete Parameter and Convert Parameter. If we choose this second option the entry in the Show column changes from No to Yes.



Search for parameter

Name	Expression	Value	Show	Order
Dimensional Constraint Parameters				
s1_rad1	75	75	No	
s1_rad				
s1_rad				
s1_rad4	25	25	No	
s1_rad5	75	75	No	

Delete Parameter
Convert Parameter

All: 5 of 5 parameters displayed

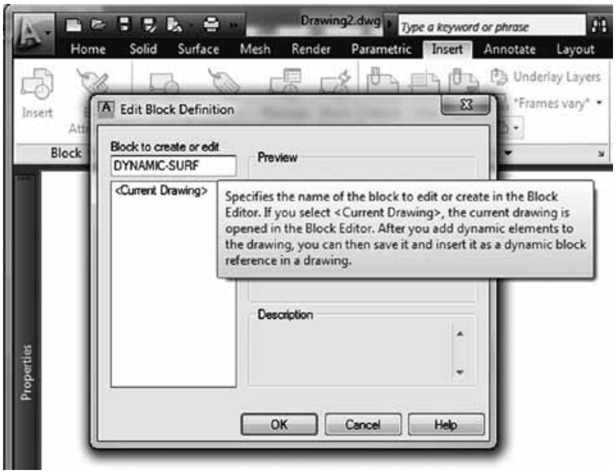
Figure 20.5. Options for deleting or converting Parameters.

If I do this with all the constraints and save the block, all these parameters will now be available in the block's Properties palette. Having the surface as a block it can be moved without fear of losing its associativity and its parameters can be modified any way I please.

Automating the conversion of Dimensional Constraints into Parameters.

Knowing what to do, we must now find the commands and options that nay be used for executing this conversion in our program. As was shown in Chapter 10, Visual LISP programs can be executed within the Block Editor. This holds for **C:ASSOC-SURF**. It is in the Block Editor where we can find the commands allowing us to create or covert the *Dimensional Constraints* into *parameters*. The Block Editor's specific commands include _BCPARAMETER which is equivalent to _DIMCONSTRAINT. These commands are practically the same. The options it offers (see Table 20.14) can create in the Block Editor constraint parameters for every situation where it was possible to create dimensional constraints.

For us the most interesting option is the one that converts dimensional constraints into constraint parameters. This option can be included in our **C:ASSOC-SURF** program supposing we can detect if it is running within the Block Editor. This way we would have a more universal program, creating a normal associative surface if we work in Model Space or creating a dynamic block by working in the Block Editor.



Block to create or edit
DYNAMIC-SURF
<Current Drawing>

Specifies the name of the block to edit or create in the Block Editor. If you select <Current Drawing>, the current drawing is opened in the Block Editor. After you add dynamic elements to the drawing, you can then save it and insert it as a dynamic block reference in a drawing.

Figure 20.6. Dialog to create a new empty block with the Block Editor.

The criterion we can use to check if the program is running in the Block Editor is the value of the BLOCKEDITOR system variable which in that case would be set to **1**. We can then add an auxiliary function that will only operate if in the block editor, automatically converting the dimensional constraints into block dimensional parameters. This would be done by creating a selection set of dimensional constraints following the same criteria used in Listing 20.11 that will be traversed applying to each dimensional constraint the _Convert option of the _BCPARAMETER command.

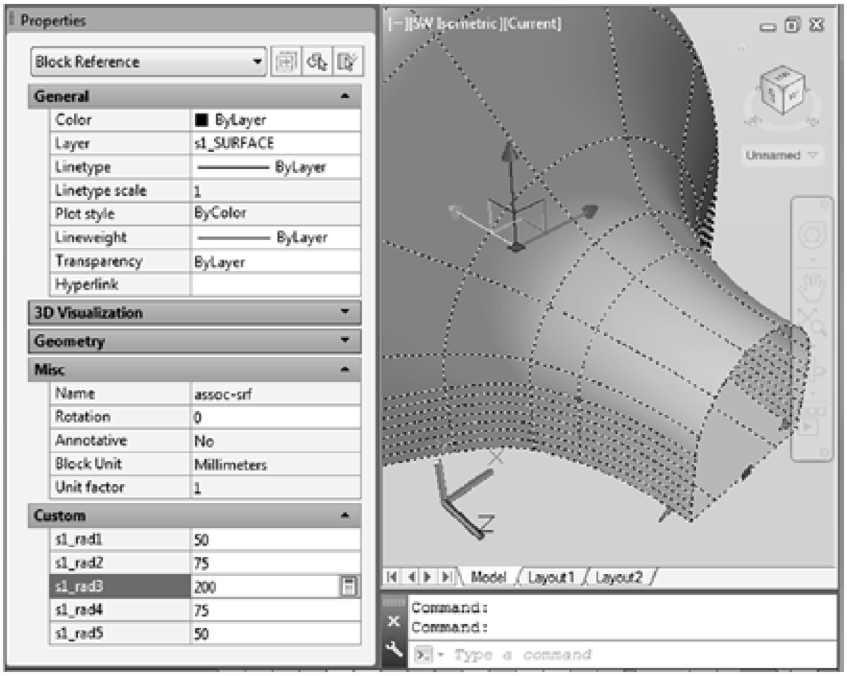
Table 20.14. _BCPARAMETER command options.

Option:	Description:
_Linear	Creates a horizontal or vertical parameter constraint based on the extension line origins and the location of the dimension line
_Horizontal	Constrains the X distance of a segment or between two points of different objects. Applicable to polyline segments and lines.
_Vertical	Constrains the Y distance of a segment or between two points of different objects. Applicable to polyline segments and lines.
_Aligned	Constrains dimensions depending on the options: _Object: length of a linear segment. _Point & Line: distance between a point and the nearest point of a line. _2Lines: distance between two selected lines that become parallel.
_ANgular	Constrains the angle between two line segments. Applied to two line segments, 3 points or an arc.
_Radial	Constrains the radius of a circle, arc or polyline arc segment.
_Diameter	Constrains the diameter of a circle, arc or polyline arc segment.
_Convert	Converts dimensional constraints on constraint parameters.

```
(defun cmd-conv-param  (id / dim-constraints i)
  (if (= (getvar "BLOCKEDITOR") 1)
    (progn (setq dim-constraints (ssget
                                  "X"
                                  (list
                                    '(0 . "DIMENSION")
                                    '(8 . "*ADSK_CONSTRAINTS")
                                    (cons 1
                                          (strcat id "_rad*"))))
                 i               0)
           (repeat (sslength dim-constraints)
             (vl-cmdf "_bcparameter"
                      "_Convert"
                      (ssname dim-constraints i)
                      "")
             (setq i (1+ i))))))
```

Listing 20.13. Converting dimensional constraints into parameters when working in the Block Editor.

Including a call to this function in **`C:ASSOC-SURF`** we will have a program capable of creating a normal associative surface in Model Space or a dynamic block if executed within the Block Editor.


Properties

Block Reference

General	
Color	■ ByLayer
Layer	s1_SURFACE
Linetype	——— ByLayer
Linetype scale	1
Plot style	ByColor
Lineweight	——— ByLayer
Transparency	ByLayer
Hyperlink	
3D Visualization	
Geometry	
Misc	
Name	assoc-srf
Rotation	0
Annotative	No
Block Unit	Millimeters
Unit factor	1
Custom	
s1_rad1	50
s1_rad2	75
s1_rad3	200
s1_rad4	75
s1_rad5	50

```
Command:
Command:
Type a command
```

Figure 20.7. Properties palette showing the dimensional constraints.

```
(defun C:ASSOC-SURF  (/ *error* n interval height width
                      origin-list profiles)
  (defun *error*  ()
    (cmd-out)
    (vla-EndUndoMark *aevl:drawing*))
  (vla-StartUndoMark *aevl:drawing*)
  (cmd-in)
  (if (= (getvar "WORLDUCS") 0)
    (vl-cmdf "_UCS" "_W"))
  (assoc-surf-data)
  (setq i 0)
  (repeat n
    (setq origin-list (cons (* interval i) origin-list)
          i           (1+ i)))
  (make-profiles
    (reverse origin-list)
    height
    width
    (strcat id "_PROFILE")
    id)
  (make-assoc-surf profiles)
  (mod-constraint)
  (cmd-conv-param id)
  (cmd-out)
  (ax-SWt)
  (vla-EndUndoMark *aevl:drawing*))
```

Listing 20.14. C:ASSOC-SURF with constraints conversion into parameters.

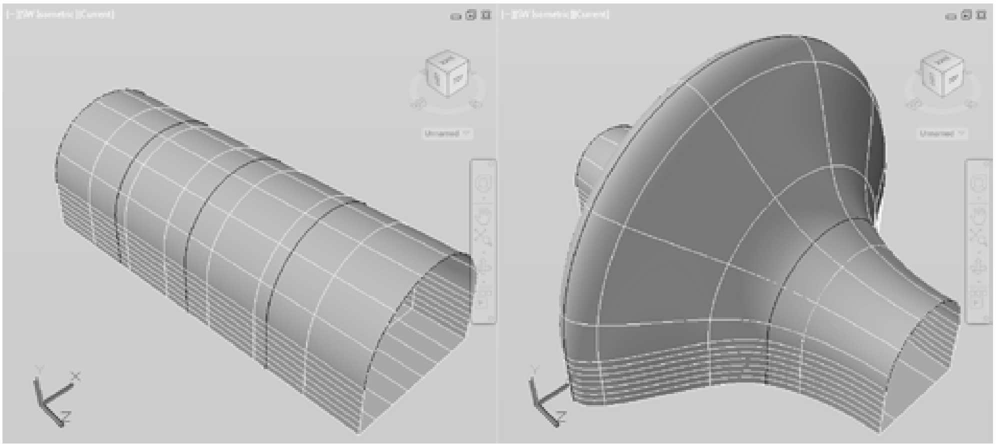


Figure 20.8. Associative surface with initial (left) and modified parameter values (right).

20.8 Summary.

We have considered in this chapter the programming options for Procedure and NURBS surfaces. There are no ActiveX methods for creating them and the data associated to most of their DXF group codes is encrypted, so the only way to create them is the through the `command/vl-cmdf` interface.

- In the case of Procedural surfaces which are defined from linear cross-sections it is possible to retain the association between the surface and them. The cross sections can be assigned geometric and dimensional constraints so that they can be modified by parameters. Parameter modification can be implemented in a program. This way we can have surfaces whose shape changes by manually or programmatically modifying these parameters. A very practical way for using these surfaces is the creation of dynamic blocks with the aid of the Block Editor.
- The most widely used surfaces in industrial design are the NURBS surfaces. This is due to their great advantages, which include rich interactive design capabilities and accurate representation of closed shapes such as conics and quadrics. In fact, many CAD/CAM, virtual reality, visualization and animation applications use models built with NURBS surfaces.
- The **`SPLINE`** entities explained in Chapter 14 are especially suitable for creating NURBS surfaces. These surfaces cannot be modified through Visual LISP programming. They can only be modified interactively using their control vertices and gizmos.