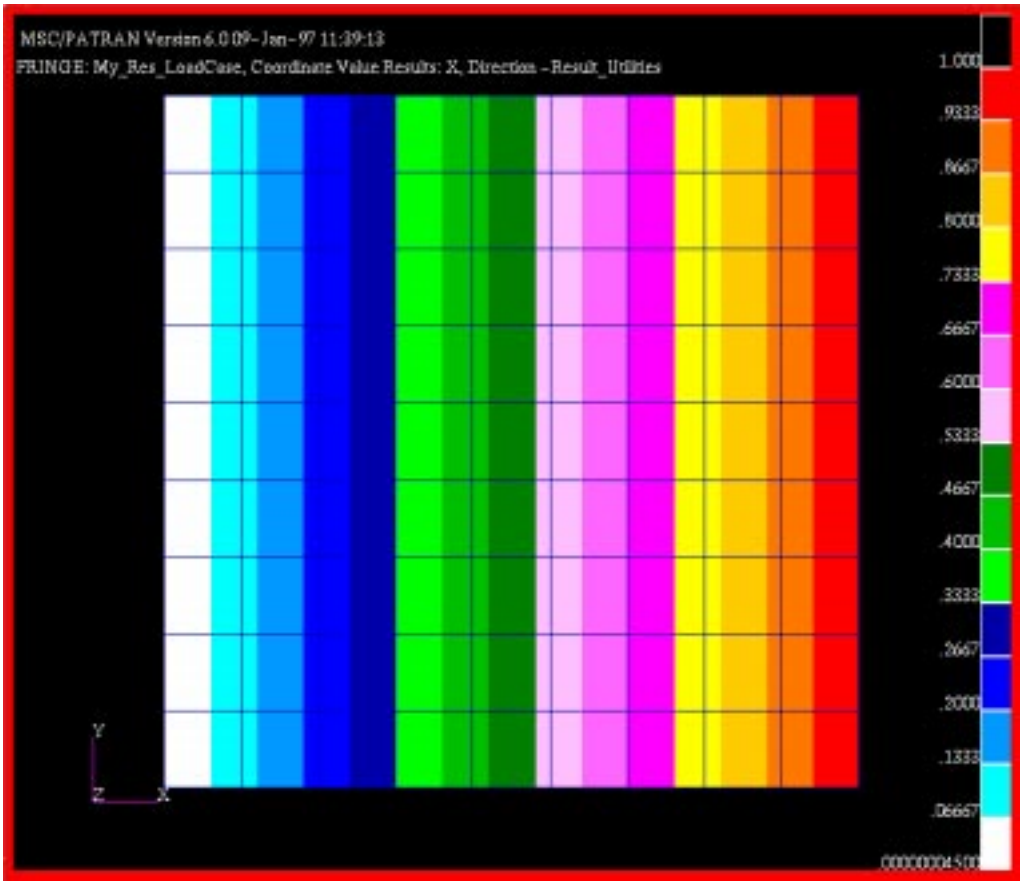
EXERCISE 19

Creating Nodal Results



Objectives:

- Write a PCL function which creates nodal results.
- Understand how MSC/PATRAN stores results.

Problem Description:

In this Exercise write a PCL function which creates nodal results based on the X, Y or Z location of the node. This will be done by using the function **res_utl_create_nodal_result()**.

Files:

All the files that used in this exercise are listed below. Each list includes the file, where it originated, and a summary of information of how it relates to the exercise.

File	Supplied/Created	Description
create_xyz_result.pcl	Created	This file should be supplied with the programs.

Suggested Exercise Steps:

- Write a PCL function **add_nodal_results()** which will be called by **create_xyz_result()**.
- Compile the function
- Verify the PCL function by making 3 sets of results.

Exercise Procedure:

1. Either use vi or jot as the text editing tool. Open the file named *create_xyz_result.pcl*. It should already exist in your directory. Complete the function called add_nodal_results().

Type **p3** at the prompt and **<return>.**

After the main menu and command window appear, type **!!input create_xyz_result.pcl** in the command line

Resolve any compile errors by editing *create_xyz_result.pcl*, and re-compiling in *p3*.

2. Open a database that has nodes, and elements.

If you do not have a database with nodes or elements create one now.

3. Test the function.

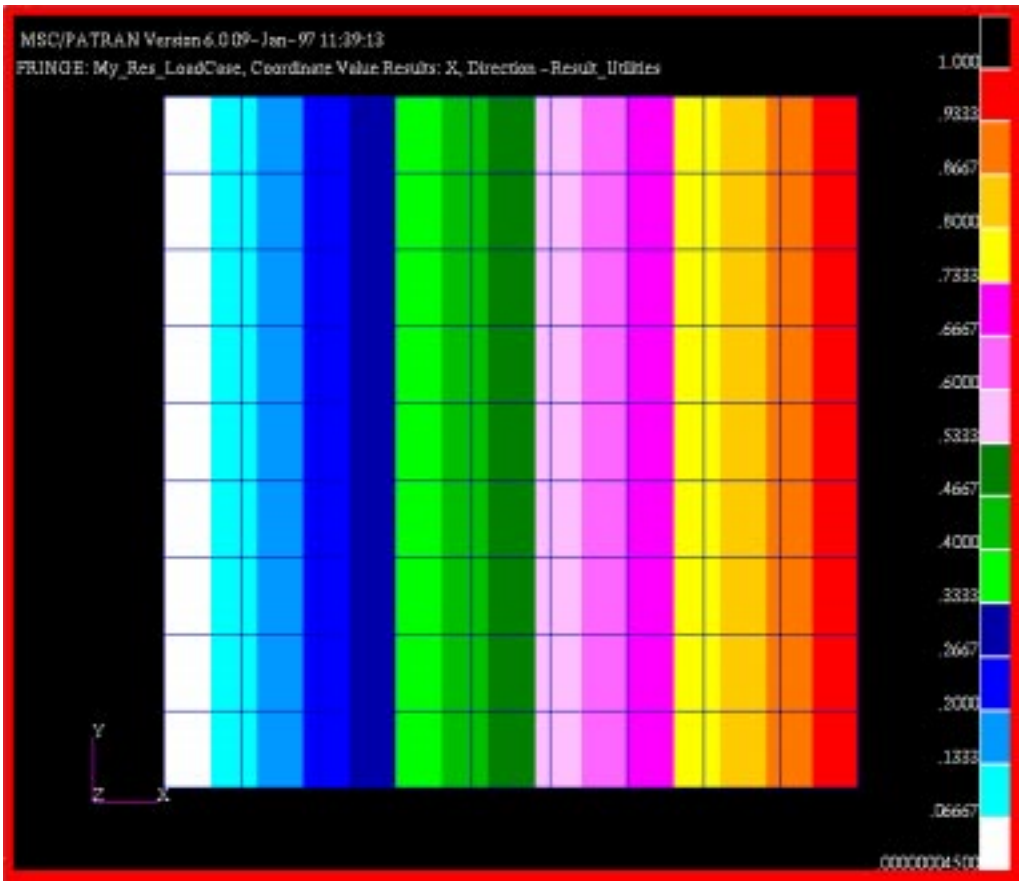
```
!!input create_xyz_result.pcl
create_xyz_result(1)
```

The “1” in the parentheses denotes that you are asking for the position of the nodes in the x location. By typing in the same command, however replacing the 1 with a 2 will give the y locations of the nodes.

There should be two results files to choose from, since the PCL file added a separate result file for the y locations.

4. Click on the Results radio button and click apply on the form.

5. The following results should appear for a simple patch.



```
Function add_nodal_results(num_nodes, nodeids, values,@
                           subcase_title, pri_title, sec_title)
   INTEGER nodeids(), num_nodes
   REAL values()
   STRING subcase_title[], pri_title[], sec_title[]
   STATIC INTEGER lcid = -1
   INTEGER scid, sect_id
   INTEGER pri_res_id, status
   INTEGER sec_res_id
   INTEGER layer_id = 1
   INTEGER res_ids(5)

/*
* Create a load case
*/
  If ( lcid == -1 ) THEN

   status = db_create_load_case(  @
      /* load case name */ ******** 1 *******, @
      /* load case type */ ******** 2 *******, @
      /* load case desc */ ******** 3 *******, @
      /* num_loads_assc */ 0, @
      /* load_ids       */ [1], @
      /* dynamic case   */ "", @
      /* evaluation pt  */ 0., @
      /* load priority  */ [0], @
      /* load case id   */ lcid )

   IF ( status != 0 ) THEN
      write("error creating load case")
      RETURN Status
   END IF
 END IF
/*
* Create a sub case in that load case and get the result case id
*/
```

```
status = db_create_sub_case( @
/* Load Case Name */ ******* 4 *******, @
/* Load Case Type */ ******* 5 *******, @
/* Load Case Desc */ ******* 6 *******, @
/* num_loads_assc */ ******* 7 *******)

   IF ( status != 0 ) THEN
      write("error creating sub case")
      RETURN Status
   END IF

/*
* Create a section position
*/

   status = dbt_create_sect_pos ( 1, "", 0, [0., 0., 0., 0.],
                        sect_id)
   IF ( status != 0 ) THEN
      write("error creating sub case")
      RETURN Status
   END IF

/*
* Every section must have one layer
*/

   status = dbt_create_layers( @
      /* num  */   1, @
      /* comp */   0, @
      /* sectn_id */ ******** 8 ********, @
      /* layer_id */ ******** 9 ******** )

   IF ( status != 0 ) THEN
      write("error creating layers")
      RETURN Status
   END IF

/*
```

```
* Use res_utl_create_nodal_results to create your results
*/

    res_ids(1) = ******* 10 *******
    res_ids(2) = ******* 11 *******
    res_ids(3) =  pri_res_id
    res_ids(4) =  sec_res_id
    res_ids(5) = ******* 12 *******

    status = res_utl_create_nodal_result(@
       /* res_ids */ ******* 13 *******, @
       /* pri_title */ pri_title, @
       /* sec_title */ sec_title, @
       /* num_nodes */ ******* 14 *******, @
       /* node_ids  */ ******* 15 *******, @
       /* cid_list  */ "", @
       /* datatyep  */ 1, @
       /* results   */ ******* 16 *******)

    IF ( status != 0 ) THEN
       write ("error createing results")
       RETURN status
    END IF

    ui_writef("'Result Ids are ', 5I3", res_ids)

RETURN 0

END FUNCTION /* add_nodal_results */

FUNCTION create_xyz_result(axis)

   INTEGER axis
   INTEGER num_nodes, node_ids(virtual), i
   INTEGER status
   INTEGER ref(virtual), anal(virtual)
   REAL    node_values(virtual), xyzs(virtual)
   STRING  pri_title[3]

   status = db_count_nodes(num_nodes)
```

```
IF ( status != 0 ) THEN
   msg_to_form( status, 3, 0, 0, 0., “”)
   RETURN status
END IF

sys_allocate_array(node_ids, 1, num_nodes)
sys_allocate_array(node_values, 1, num_nodes)
sys_allocate_array(ref, 1, num_nodes)
sys_allocate_array(anal, 1, num_nodes)
sys_allocate_array(xyzs, 1, num_nodes, 1, 3)

status = db_get_node_ids(num_nodes, node_ids)
IF ( status != 0 ) THEN
   msg_to_form( status, 3, 0, 0, 0., “”)
   RETURN status
END IF

status = db_get_nodes(num_nodes, node_ids, ref, anal, xyzs)
IF ( status != 0 ) THEN
   msg_to_form( status, 3, 0, 0, 0., “”)
   RETURN status
END IF

FOR ( i = 1 to num_nodes )
   node_values(i) = xyzs(i, axis)
END FOR

IF ( axis == 1 ) THEN pri_title = “X”
IF ( axis == 2 ) THEN pri_title = “Y”
IF ( axis == 3 ) THEN pri_title = “Z”


add_nodal_results(num_nodes, node_ids, node_values,@
          “Coordinate Value Results”, pri_title, “Direction”)

sys_free_array(node_ids)
sys_free_array(node_values)
```

```
   RETURN 0

END FUNCTION /* create_xyz_result */

END FUNCTION /* add_nodal_results */




FUNCTION create_xyz_result(axis)

   INTEGER axis
   INTEGER num_nodes, node_ids(virtual), i
   INTEGER status
   INTEGER ref(virtual), anal(virtual)
   REAL    node_values(virtual), xyzs(virtual)
   STRING  pri_title[3]

   status = db_count_nodes(num_nodes)
   IF ( status != 0 ) THEN
      msg_to_form( status, 3, 0, 0, 0., "")
      RETURN status
   END IF

   sys_allocate_array(node_ids, 1, num_nodes)
   sys_allocate_array(node_values, 1, num_nodes)
   sys_allocate_array(ref, 1, num_nodes)
   sys_allocate_array(anal, 1, num_nodes)
   sys_allocate_array(xyzs, 1, num_nodes, 1, 3)

   status = db_get_node_ids(num_nodes, node_ids)
   IF ( status != 0 ) THEN
      msg_to_form( status, 3, 0, 0, 0., "")
      RETURN status
   END IF

   status = db_get_nodes(num_nodes, node_ids, ref, anal, xyzs)
   IF ( status != 0 ) THEN
      msg_to_form( status, 3, 0, 0, 0., "")
      RETURN status
```

```
   END IF

   FOR ( i = 1 to num_nodes )
      node_values(i) = xyzs(i, axis)
   END FOR

   IF ( axis == 1 ) THEN pri_title = "X"
   IF ( axis == 2 ) THEN pri_title = "Y"
   IF ( axis == 3 ) THEN pri_title = "Z"


   add_nodal_results(num_nodes, node_ids, node_values,@
               "Coordinate Value Results", pri_title, "Direction")

   sys_free_array(node_ids)
   sys_free_array(node_values)

   RETURN 0

END FUNCTION /* create_xyz_result */
```

PCL Solutions:

1 “My_Res_Loadcase”
2 1,
3 “User defined results, Made for PAT304 class”,
4 lcid,
5 subcase_title
6 scid
7 pri_res_id
8 sect_id
9 layer_id
10 lcid
11 scid
12 layer_id
13 res_ids
14 num_nodes
15 nodeids
16 values