Local refinement toward a feature

Table of contents

1 Description 1
2 How local refinement works 1
3 Naming the feature in Abaqus/CAE 2
4 Regenerating the model file 2
5 Running refinecf 3
5.1 Other predicates 3
6 Meshing and solving 4
7 What to expect 4

> ⚠ Warning
>
> Local refinement is an **experimental** capability. The workflow below drives the meshing pipeline from the command line; tighter integration with the Abaqus/CAE job flow is planned. Run the meshing step **serially** — local refinement does not yet support MPI-parallel meshing.

1 Description

In this problem we will locally refine an immersed Coreform IGA mesh toward a small geometric feature. We continue directly from the thin plate with a hole example — complete that example first, since we reuse its model, mesh, and job.

The plate's stress concentration lives at the edge of the hole, and that is where mesh resolution pays off: refining the entire background mesh sharpens the answer but multiplies the element count everywhere, including the far field where the stress is essentially uniform.

Local refinement instead refines the background mesh **only near the hole**: you name the hole's surface in Abaqus/CAE, and the `refinecf` utility marks every background cell that surface touches for hierarchical refinement, level by level, down to the resolution you request.

2 How local refinement works

The pieces fit together as follows:

1. In Abaqus/CAE you create a **named assembly set** containing the feature's faces.
2. When the Coreform IGA job writes its `<job>.cf` file, the set is exported into the file's geometry description.

3. You run `refinecf` on the `.cf` file which will discover the background cells the named surfaces touch, so each refinement level tightens around the feature, and writes the resulting refinement instructions back into the `.cf` file.
4. The Coreform IGA meshing step reads those instructions and builds a hierarchical spline basis.

Everything downstream (trimming, the Abaqus solve, visualization) then works exactly as it does for a uniform mesh.

3 Naming the feature in Abaqus/CAE

Open the model database from the thin plate with a hole example. The only adaptivity-specific modeling step happens in the **Assembly module**: create a set containing the hole's inner cylindrical face.

1. In the model tree, expand **Assembly** and double-click **Sets**.
2. In the **Create Set** dialog, enter the name `hole` and click **Continue**.
3. In the viewport, select the hole's inner cylindrical face, and click **Done**.

The equivalent scripting-interface command selects the hole face by a bounding box that contains the hole but stops short of the plate's outer faces:

```
inst = mdb.models["PWH-Kt"].rootAssembly.instances["Plate-1"]
hole_faces = inst.faces.getByBoundingBox(
    xMin=-1.1, xMax=1.1,   # the hole has radius 1 at the origin;
     yMin=-1.1, yMax=1.1,    # the plate's outer faces extend to x = +/-5, y
= +/-2.5
    zMin=-1.0, zMax=1.0,
)
mdb.models["PWH-Kt"].rootAssembly.Set(faces=hole_faces, name="hole")
```

> **i** Note
>
> Create the set on the **assembly**, not on the part. An assembly set is exported with its bare name (`hole`); a part-level set is prefixed with the instance name, and `refinecf --surface-set=hole` will not find it.

4 Regenerating the model file

The `<job>.cf` file written by the Coreform IGA job carries the geometry, the background-mesh definition, and the exported sets, and is the input to every step below. Because we just added the `hole` set, the `Job-PWH-Kt.cf` file from the original example run does not contain it — resubmit `Job-PWH-Kt` from the **Coreform IGA for Abaqus Job Manager** so the file

is regenerated with the set included. The refreshed `Job-PWH-Kt.cf` is written to the Abaqus working directory.

5 Running refinecf

`refinecf` is installed alongside the other Coreform command-line tools. Its inputs are the `.cf` file, the set name, and the number of refinement levels:

```
refinecf Job-PWH-Kt.cf \
    --surface-set=hole \
    --max-level=2 \
    --surface-offset=0.1 \
    --balance \
    --apply
```

- `--surface-set=hole` selects the predicate: refine cells that the named CAD surfaces touch.
- `--max-level=2` refines two levels deep. Each level halves the element size, so the in-plane element size at the hole edge goes from `0.3` to `0.075`.
- `--surface-offset=0.1` widens the catch: cells within that distance of the surface refine too, not only the cells it passes through. About a third of a background cell works well — it keeps the refined band snug against the feature while avoiding lone coarse cells grazing the surface. Pass `0.0` for the exact-touch band, or a per-level list such as `--surface-offset=0.6,0.3` to cast a wider net at the coarse level and taper toward the surface at the deepest level.
- `--balance` enforces 2:1 admissibility: after the predicate-driven refinement, additional cells are refined so that no two face-adjacent elements differ by more than one level, keeping the element-size transition gradual.
- `--apply` writes the refinement instructions directly into the `.cf` file. Without it, `refinecf` only writes a JSON description (`--output`, default `refinement.json`) and leaves the model file untouched.

The log reports how many elements were selected at each level, per part.

5.1 Other predicates

If your feature has no convenient named surface, `refinecf` also accepts purely geometric predicates:

- `--center=x,y,z --radius=R` — refine inside a sphere (add `--inner-radius` for a spherical shell),
- `--bbox=xmin,ymin,zmin,xmax,ymax,zmax` — refine inside an axis-aligned box.

Both accept per-level value lists (level 0 first) to shape the refinement gradation, exactly like `--surface-offset`.

6 Meshing and solving

Run the Coreform IGA meshing step on the refined model file:

```
coreform_iga_mesh Job-PWH-Kt.cf
```

The refinement instructions stored in the file are picked up automatically and the remaining pipeline (interop, the Abaqus solve, and post-processing) proceeds exactly as in the uniform-mesh workflow.

> ! Important
>
> Submitting the job from Abaqus/CAE regenerates the `.cf` file **without** the refinement instructions. Any time the file is regenerated, re-run `refinecf --apply` before meshing again.

7 What to expect

The refined mesh concentrates its added degrees of freedom in a band around the hole: the level-2 elements resolve the stress gradient at the hole edge while the plate's far field stays at the coarse background resolution.

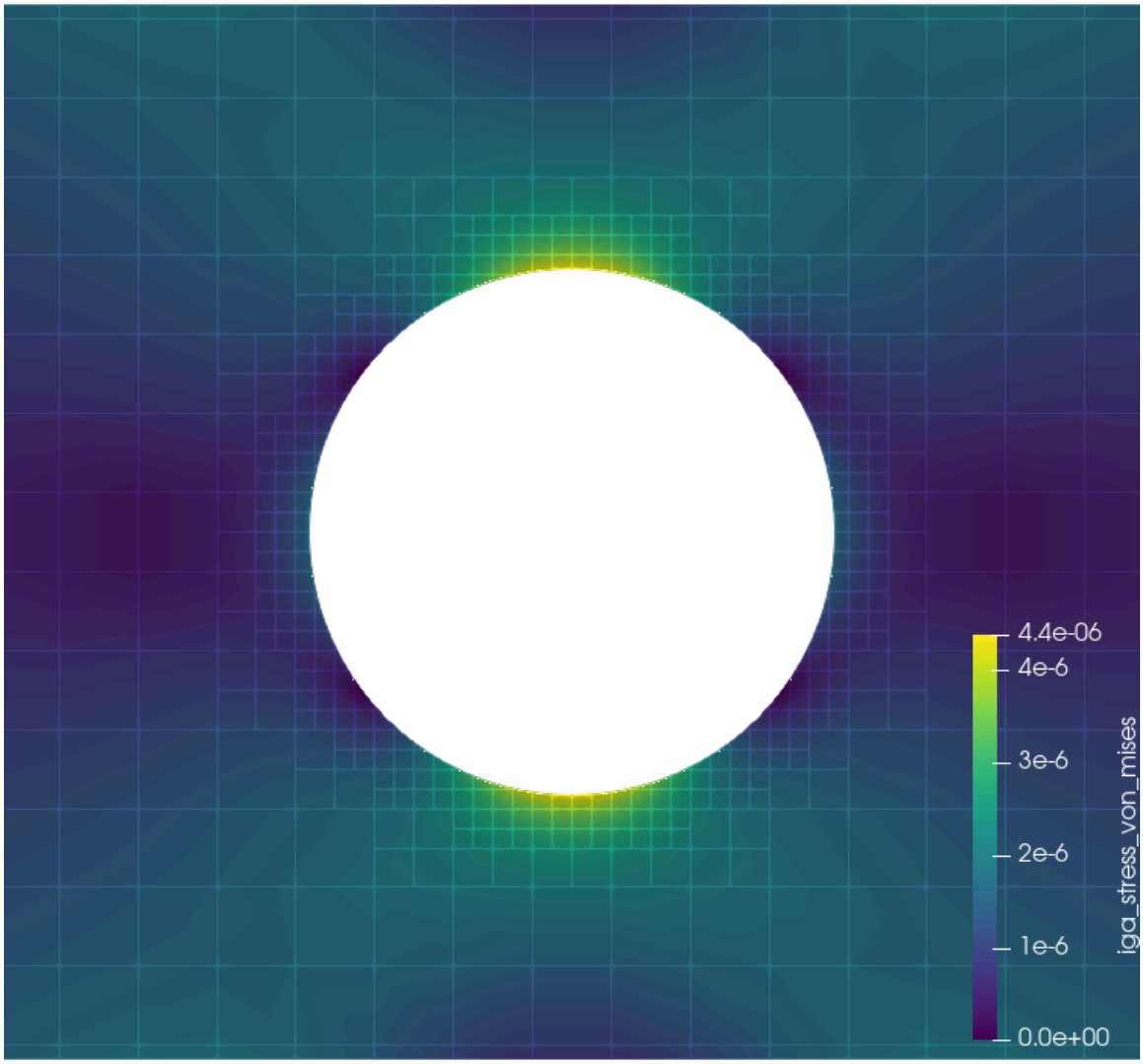

Figure 1: Von Mises stress near the hole on the locally refined mesh. The element boundaries show the two balanced refinement levels hugging the hole, with the stress concentration at the top and bottom of the hole resolved by the finest elements.

Compare the maximum principal stress along the line probe from the original example against the uniform-mesh run — the refined mesh sharpens the peak at the hole edge for a small fraction of the element count a uniformly-fine mesh would need.