

Response spectrum case files

Table of contents

1 Spectra	2
2 Excitations	2
3 Summation rules	3
4 Modal damping	4
5 Writing a case file from Python	4

A case file is a JSON document that describes one response spectrum analysis. Random response cases use a different file, documented on the [random response page](#). It carries the same information as the `*SPECTRUM`, `*RESPONSE SPECTRUM`, and `*MODAL DAMPING` keyword blocks, so a case file and an Abaqus step can be written from a single definition.

A complete case file has four top-level keys, all of which are required:

```
{
  "spectra": {
    "DESIGN": {
      "type": "ACCELERATION",
      "gravity": null,
      "rows": [
        [0.8, 0.5, 0.02], [22.0, 50.0, 0.02], [5.0, 5000.0, 0.02],
        [0.6, 0.5, 0.05], [14.0, 50.0, 0.05], [4.0, 5000.0, 0.05]
      ]
    }
  },
  "excitations": [
    {"spectrum": "DESIGN", "direction": [1, 0, 0], "scale": 1.0, "duration": null},
    {"spectrum": "DESIGN", "direction": [0, 1, 0], "scale": 0.7, "duration": null},
    {"spectrum": "DESIGN", "direction": [0, 0, 1], "scale": 0.4, "duration": null}
  ],
  "modal_summation": "CQC",
  "directional_summation": "SRSS",
  "damping": {"mode_ranges": [[1, 10, 0.05], [11, 20, 0.08]]}
}
```

1 Spectra

`spectra` maps a name to one spectrum definition. The name is how an excitation refers to the spectrum, so it plays the same role as the `NAME` parameter of `*SPECTRUM`.

Field	Required	Description
<code>type</code>	no	<code>ACCELERATION</code> (default), <code>VELOCITY</code> , <code>DISPLACEMENT</code> , or <code>G</code>
<code>gravity</code>	only for <code>G</code>	Acceleration of gravity, in the model's unit system
<code>rows</code>	yes	Table of magnitude, frequency in cycles per time, and damping ratio

Each row is a triple in the same order as a `*SPECTRUM` data line: magnitude first, then frequency, then the associated fraction of critical damping. Rows at the same damping ratio form one curve. Give as many curves as you have damping ratios; they are sorted for you, but a repeated damping ratio is an error.

Within a curve, frequencies must be positive and strictly increasing, and magnitudes must be positive. Both restrictions follow from the interpolation: the magnitude is interpolated linearly on a log-log scale in frequency and held constant beyond the ends of the table. Between two damping curves, the interpolation weight is linear in the damping ratio and the magnitude is interpolated logarithmically. Damping outside the tabulated range is clamped to the nearest curve.

A `G` spectrum is converted to an acceleration spectrum by multiplying by `gravity`. All three physical types are interchangeable: whichever type you supply, the utility converts it to the one a given output needs using $S^D = S^V/\omega = S^A/\omega^2$, the undamped relations Abaqus uses for the same purpose.

2 Excitations

`excitations` is a list of one to three entries, each corresponding to one data line of `*RESPONSE SPECTRUM`.

Field	Required	Description
<code>spectrum</code>	yes	Name of an entry in <code>spectra</code>
<code>direction</code>	yes	Direction cosines as a unit vector, such as <code>[1, 0, 0]</code>

Field	Required	Description
<code>scale</code>	no	Factor multiplying the spectrum magnitudes; defaults to <code>1.0</code>
<code>duration</code>	only for <code>DSC</code>	Duration of the strong motion, in the model's time unit

Directions must be unit vectors and mutually orthogonal, exactly as Abaqus requires of the second and third spectrum lines. The same spectrum may be named by more than one excitation.

3 Summation rules

`modal_summation` selects how the peak responses of the individual modes are combined.

Value	Rule
<code>ABS</code>	Sum of absolute modal peaks; the most conservative estimate
<code>SRSS</code>	Square root of the sum of the squares
<code>NRL</code>	Naval Research Laboratory: largest modal peak plus the square root of the sum of the squares of the rest
<code>TENP</code>	Ten percent method; adds a cross term for modes within 10% of each other
<code>CQC</code>	Complete quadratic combination, using the Der Kiureghian cross-correlation coefficients
<code>GRP</code>	Grouping method; sums within frequency groups, then combines groups by square root of the sum of the squares
<code>DSC</code>	Double sum combination, which needs a <code>duration</code> on every excitation

`directional_summation` selects how the excitation directions are combined.

Value	Rule
ALGEBRAIC	Signed sum of the modal amplitudes, applied before the modal summation
SRSS	Square root of the sum of the squares of the per-direction peaks
R40	Forty percent rule from ASCE 4-98
R30	Thirty percent rule from ASCE 4-98

Use `ALGEBRAIC` when the spectra are components of one base motion acting in an arbitrary direction, and one of the other rules when the excitations are statistically independent. The ordering matters: `ALGEBRAIC` merges the directions first and then combines modes, while the other three combine modes within each direction first.

4 Modal damping

`damping` accepts three spellings, matching the ways `*MODAL DAMPING` can specify a fraction of critical damping.

A single number applies one damping ratio to every mode:

```
{"damping": 0.05}
```

`mode_ranges` gives ratios by mode number, as `[first, last, ratio]` triples that are one-based and inclusive. Modes not covered by any range get zero damping, as they do in Abaqus:

```
{"damping": {"mode_ranges": [[1, 10, 0.05], [11, 20, 0.08]]}}
```

`frequency_table` gives ratios by frequency, as `[frequency, ratio]` pairs. Values are interpolated linearly in frequency between the entries and held constant outside the tabulated range:

```
{"damping": {"frequency_table": [[30.0, 0.02], [3000.0, 0.08]]}}
```

Damping enters the result twice. It selects where each mode samples the spectrum, and the `CQC` and `DSC` rules use it to compute the correlation between modes.

5 Writing a case file from Python

Case files are ordinary JSON, so you can write one with any tool. To build one from the same objects the utility uses internally, use `dump_case`:

```
from respspec import (
    DirectionalSummation, Excitation, ModalDamping,
    ModalSummation, ResponseSpectrumCase, Spectrum, SpectrumType,
)
from respspec.case_file import dump_case

spectrum = Spectrum.from_rows(
    "DESIGN",
    [(0.8, 0.5, 0.05), (22.0, 50.0, 0.05), (5.0, 5000.0, 0.05)],
    SpectrumType.ACCELERATION,
)

case = ResponseSpectrumCase(
    excitations=(
        Excitation(spectrum, (1.0, 0.0, 0.0)),
        Excitation(spectrum, (0.0, 1.0, 0.0), scale=0.7),
    ),
    modal_summation=ModalSummation.CQC,
    directional_summation=DirectionalSummation.SRSS,
    damping=ModalDamping(uniform=0.05),
)

dump_case(case, "mycase.json")
```

The same objects render as Abaqus keyword blocks through `respspec.abaqus_keywords`, which is how the validation models keep their input decks and their Python cases in agreement.