

Random response analysis

Table of contents

1 What the utility computes	1
2 RMS von Mises stress	2
3 Running an analysis	2
4 The case file	2
5 Limitations	3
6 Verification	4

Random response analysis predicts how a structure answers an excitation that can only be described statistically, such as road roughness, jet noise, or a vibration qualification test. The loading is given as a power spectral density rather than a time history, and the results are spectral densities and root mean square values rather than peaks.

This matters more for Coreform IGA for Abaqus than for a native model. Coreform IGA meshes are Abaqus user elements, and Abaqus/Standard does not post-process a random response for them. For an IGA model the frequency extraction is all the solver provides, so the random response has to be computed afterwards from the modal data. The `respspec` utility does that, using the same procedure Abaqus applies to native elements.

1 What the utility computes

The excitation is a cross-spectral density matrix projected onto the modes. For base motion this follows directly from the participation factors:

$$S_{\alpha\beta}^f(f) = \sum_J g_J^2 P^J(f) \sum_i \Gamma_{\alpha}^i \Gamma_{\beta}^j \sum_I \Psi_{ij}^{IJ} (2\pi f)^{2\lambda_I}$$

Each mode then responds through its complex transfer function H_{α} , giving the cross-spectral density of the generalized coordinates, and the physical result follows from the mode shapes:

$$S_{\alpha\beta}^q(f) = H_{\alpha} S_{\alpha\beta}^f(f) H_{\beta}^* \quad S_{NN}^u(f) = \phi_{\alpha}^N S_{\alpha\beta}^q(f) \phi_{\beta}^N$$

Velocity and acceleration spectral densities carry factors of $(2\pi f)^2$ and $(2\pi f)^4$. Every RMS value is the square root of its spectral density integrated across the frequency range by the trapezoidal rule, over exactly the frequency points Abaqus would have used.

2 RMS von Mises stress

Von Mises stress is a quadratic form in the stress components, so its RMS cannot be recovered from the RMS of the individual components. The utility uses the method of Segalman et al., which is what Abaqus uses for the same variable. Writing $T_{\alpha\beta} = \psi_{\alpha}^T A \psi_{\beta}$ for the modal stress vectors ψ and the constant matrix A of the invariant:

$$\text{RMISES} = \sqrt{\sum_{\alpha} \sum_{\beta} V_{\alpha\beta} T_{\alpha\beta}}$$

where $V_{\alpha\beta}$ is the variance matrix of the generalized coordinates.

! Important

Do not form von Mises stress from RMS stress components. RMS values are unsigned, so a component that should cancel against another instead adds. On the validation models this route is wrong by up to a factor of three.

3 Running an analysis

The workflow matches the [response spectrum walkthrough](#). Extract once, then run as many cases as you like.

```
export RESPONSE_SPECTRUM_DIR=/path/to/cf/interop/abaqus/response_spectrum
export PYTHONPATH="$RESPONSE_SPECTRUM_DIR"

abaqus python -m respspec extract myjob.odb myjob_modal.npz --fields U,S
python3 -m respspec random myjob_modal.npz mycase.json myjob_rms.npz --fields
U,V,A,S
```

The results are **RU**, **RV**, **RA**, and **RS**, one RMS value per component, plus **RMISES** when the frequency step wrote stress mode shapes. Read them exactly as the peak results are read, through `load_peak_fields_npz`.

4 The case file

A random response case file describes the frequency functions, the base motions, the correlation between them, the sweep, and the modal damping.

```
{
  "psds": {
    "QUAL_TEST": {
      "type": "BASE",
      "gravity": 9.81,
```

```

    "rows": [[0.01, 0.0, 20.0], [0.04, 0.0, 200.0], [0.01, 0.0, 2000.0]]
  },
  "base_motions": [
    {"dof": 1, "load_case": 1, "kind": "ACCELERATION"},
    {"dof": 2, "load_case": 1, "kind": "ACCELERATION"}
  ],
  "correlations": [
    {"psd": "QUAL_TEST", "type": "UNCORRELATED", "weights": {"1": [1.0, 0.0]}}
  ],
  "lower_frequency_hz": 20.0,
  "upper_frequency_hz": 2000.0,
  "n_points": 20,
  "bias": 3.0,
  "frequency_scale": "LOGARITHMIC",
  "damping": 0.05
}

```

Frequency functions. Each row of a `psds` entry is a real part, an imaginary part, and a frequency in cycles per time, matching `*PSD-DEFINITION` data lines. Values are interpolated log-log in frequency and held constant outside the table. Type `BASE` needs a `gravity` value and its ordinates are in g units squared per frequency; `FORCE` is in power units; `DB` takes decibels against a `db_reference` with band numbers 1 to 15 in place of frequencies.

Base motions. Each entry is a degree of freedom from 1 to 6, a load case number, and a spectrum kind of `ACCELERATION`, `VELOCITY`, or `DISPLACEMENT`. All base motions in one load case must share a kind. Only the primary base is supported.

Correlations. A correlation ties one frequency function to one or more load cases with complex weights, written as real and imaginary pairs. Type `CORRELATED` makes every pair of degrees of freedom in the load case fully dependent, so their participation factors add before the outer product. Type `UNCORRELATED` keeps only the diagonal, so the contributions add in squares.

The sweep. The range is divided at every eigenfrequency inside it, and each interval receives `n_points` points placed by the Abaqus bias formula. A bias above 1 clusters points toward the eigenfrequencies at the interval ends, where the response varies fastest. The default of 20 points and a bias of 3 matches Abaqus. RMS values are only as accurate as this integration, so increase `n_points` if the result has not settled.

5 Limitations

Only base motion excitation is implemented. Concentrated, distributed, and connector loads, moving noise correlations, and the `UPSD` and `UCORR` user subroutines are not.

Secondary bases are not supported, matching the restriction that a random response step has only a primary base.

6 Verification

The same three steel solids used for the response spectrum work carry six random response steps each, covering one and three excitation directions, correlated and uncorrelated loading, two superposed load cases, a displacement base spectrum, and both frequency scales. Recombining the extracted modes in Python reproduces the Abaqus values for **RU**, **RV**, **RA**, **RS**, and **RMISES** to within a relative error of about 200×10^{-9} of the peak, which is the precision of the single-precision fields Abaqus writes. The generated sweep matches Abaqus point for point.

Independently of Abaqus, the single degree of freedom response to white noise is checked against its closed form, $\sigma^2 = S_0 / (8\xi\omega_0^3)$, and the agreement is shown to improve as the number of frequency points rises.