

# leigqNEWTON Manual

leigqNEWTON is a stand-alone MATLAB toolbox for computing LEFT eigenvalues and corresponding eigenvectors of quaternion matrices using a Newton-type method.

The toolbox uses only MATLAB and its built-in quaternion class (no third-party toolboxes).

## Author

**Michael Sebek**, Czech Technical University in Prague (CVUT), Faculty of Electrical Engineering (FEL),  
Department of Control Engineering, e-mail: [michael.sebek@fel.cvut.cz](mailto:michael.sebek@fel.cvut.cz)

## Citation (*current status: Submitted*)

If you use this toolbox in academic work, please cite the associated paper:

M. Sebek, "Computing Left Eigenvalues of Quaternion Matrices", submitted to Linear Algebra and its Applications (LAA), 2026.

and optionally cite the software release:

M. Sebek, leigqNEWTON (MATLAB toolbox), version 12, 2026

## BibTeX templates

```
@unpublished{Sebek_leigqNewton_LAA_2026,
author = {Michael {Sebek}}, title = {Computing Left Eigenvalues of Quaternion Matrices},
note ={Submitted to Linear Algebra and its Applications}, year = {2026} }

@software{Sebek_leigqNewton_2026,
author = {Michael {\v{S}}ebek},
title = {leigqNEWTON: Public MATLAB toolbox for left eigenvalues of quaternion matrices},
version = {12}, year = {2026} }
```

## License

MIT License.

## 0. Main goals of this public release:

- Enable readers to try out (and verify) the Newton-based method proposed in the accompanying paper on their own machines, with minimal friction.
- Provide a stand-alone MATLAB implementation that can be directly reused as a practical tool for computing left eigenvalues of quaternion matrices in the user's own research.
- Support further exploration of quaternionic spectral phenomena by offering reproducible reference examples—ranging from matrices with fewer/more than n isolated left eigenvalues to cases with one or multiple spherical families, and combinations thereof. A curated collection of such “interesting” matrices is included here and, in a broader form, in the companion LAA\_Zoo bundle.

Practical properties of the public bundle (engineering notes):

- easy download + addpath + smoke test (package verifier)
- practical solver ready both for fast computation and intensive research
- certificate-like residual checks for accepted eigenpairs
- reproducible scripts for selected paper/supplement examples
- “path hygiene”: avoids shadowing and fragile dependencies

## 1. What is this toolbox for?

leigqNEWTON solves the **left quaternion eigenvalue problem**:

Find lambda in  $\mathbb{H}$  and v in  $\mathbb{H}^n \setminus \{0\}$  such that  $Av = \lambda v$  for a given quaternion matrix A in  $\mathbb{H}^{n \times n}$ .

## 2. Requirements

- MATLAB with the built-in quaternion class available. (In many installations this comes with Aerospace Toolbox.)
- No third-party quaternion toolboxes are used or required.

```
hasQuat = exist('quaternion','class') == 8;
if ~hasQuat
    error(['This manual requires MATLAB''s built-in quaternion class. ' ...
        'Please ensure quaternion is available in your installation.']);
end
```

## 3. Installation and path hygiene

Recommended user workflow:

1. Unzip the toolbox folder somewhere convenient.
2. Add only the toolbox root to the MATLAB path (avoid genpath unless you know why).
3. Run the package verifier (smoke test).

We locate the toolbox root robustly via `which('leigqNewton')`.

```
root = fileparts(which('leigqNewton'));
fprintf('leigqNewton root: %s\n', root);

% If needed (typical for users who just unzipped the folder):
% addpath(root);
```

## 4. Smoke test (package verifier)

The public bundle ships with a verifier that checks:

- listed functions in `Contents.m` exist
- documentation pages exist for the listed functions

- basic callability checks (fast)

Run: PACKAGEVerifierNEWTON\_fromContents

```
if exist('PACKAGEVerifierNEWTON_fromContents','file') == 2
    R = PACKAGEVerifierNEWTON_fromContents;
else
    warning('PACKAGEVerifierNEWTON_fromContents not found on the path.');
end
```

## 5. First computation: a small isolated case

For a beginner, start with a small dense matrix where you typically get isolated eigenvalues.

Reproducibility and timing: Many randomized initializations depend on RNG state. For documentation runs, we recommend restoring MATLAB's **startup RNG state**: `rng('default')`

This is for reproducible demo behavior (and often reproducible runtime), not a mathematical requirement.

```
rng('default');

n = 3;
A = quaternion(randn(n), randn(n), randn(n), randn(n));
fprintf("\nMatrix:\n");
display(A);

fprintf("\nSolver:");
[lambda, V, res] = leigqNEWTON(A);

fprintf(' Computed %d eigenpairs.\n', numel(lam));
fprintf("Left eigenvalues:\n");
display(lambda);
fprintf("Eigenvectors:\n");
display(V);
fprintf('Residual summary (res): median = %.3e, max = %.3e\n', median(res),
max(res));
```

## 6. Profiles and common options

The solver supports Name,Value options. Typical user-facing pattern:

- `SolveProfile` ('standard' or 'reliable')
- Seed for reproducible behavior
- `InfoLevel` for reporting ('none' / 'summary' / 'full')

**Example:** reproducible run + summary info

```
[lam, V, res, info] = leigqNEWTON(A, ...
    'SolveProfile','reliable', ...
```

```

'Seed', 1, ...
'InfoLevel','summary');

fprintf('\nReliable run: %d eigenpairs. \n res median=%e, max=%e\n',
numel(lam), median(res), max(res));

% info is typically a cell array; print the first summary struct
if iscell(info) && ~isempty(info)
    S = info{1};
    disp('info{1} (summary struct):');
    disp(S);
end

```

## 7. Distinct representative set (lamU)

Some problems can return repeated/near-duplicate hits. The solver can also return a distinct representative set `lambdaU` with `VU` and `resU`.

```

fprintf("\nDistinct representative set:\n");
[lambda, V, res, info, lambdaU, VU, resU] = leigqNEWTON(A, ...
    'SolveProfile','reliable', ...
    'Seed', 1);

fprintf('All hits: %d, representative distinct: %d\n', numel(lam), numel(lambdaU));
fprintf("Distinct eigenvalues:\n");
display(lambdaU)
fprintf('resU summary: median=%e, max=%e\n', median(resU), max(resU));

```

## 8. Certificates (recommended for reporting)

The toolbox provides certificate-like residuals:

- `leigqNEWTON_cert_resMin(A, lambda)` : minimized residual over v
- `leigqNEWTON_cert_resPair(A, lambda, v)` : residual of a pair (`lambda, v`)

Printing tip: If you want to print values and keep the editor happy, use `disp(...)` or `fprintf(...)`.

(Avoid writing a bare expression without semicolon if the Code Analyzer complains.)

```

fprintf('\nPost hoc certificates computation:\n')
lambda1 = lamU(1);
v1      = VU(:,1);
display(lambda1);
display(v1);

rMin1  = leigqNEWTON_cert_resMin(A, lambda1);
rPair1 = leigqNEWTON_cert_resPair(A, lambda1, v1);

fprintf('Certificates for this eigenpair:\n rMin1=%e, rPair1=%e\n',
median(res), max(res));

```

## 9. Nicely printing residual collections

Preferred: `fprintf('median=%e, max=%e\n', median(x), max(x));`

```
fprintf("\nNicely printing residual collections:\n");
fprintf('res (all hits) : median=%e, max=%e\n', median(res), max(res));
fprintf('resU (distinct) : median=%e, max=%e\n', median(resU), max(resU));
```

## 10. Running included paper examples (optional)

The public bundle typically includes scripts in `root/examples`.

Robust calling pattern:

```
ex = fullfile(root,'examples','ExNEWTON_1_HuangSo.m');
if exist(ex,'file'), run(ex); else, error('Example not found: %s', ex);
end
```

For documentation builds we keep them OFF by default to avoid long runs.

```
RUN_PAPER_EXAMPLES = false;

if RUN_PAPER_EXAMPLES
    ex1 = fullfile(root,'examples','ExNEWTON_1_HuangSo.m');
    ex2 = fullfile(root,'examples','ExNEWTON_2_MVPS.m');

    if exist(ex1,'file'), run(ex1); else, error('Example not found: %s', ex1); end
    if exist(ex2,'file'), run(ex2); else, error('Example not found: %s', ex2); end
end
```

## 11. Sphere hunting (advanced topic)

Spherical families of left eigenvalues are documented separately to keep “Getting Started” and this Manual fast and focused.

See: `doc_SphereHunting`

Note: sphere-hunting runs can take minutes depending on settings.

## 12. Troubleshooting

- "Built-in quaternion A\*B not available in this MATLAB configuration." This indicates a limitation in your MATLAB installation. The toolbox provides wrapper operations (`qmtimesNEWTON`, `qmrddivideNEWTON`, `qmldivideNEWTON`) that avoid relying on unavailable quaternion matrix kernels.
- Warnings about "Matrix is close to singular or badly scaled." These may occur during polishing/refinement for hard cases. For documentation, prefer small sizes unless you are stress testing.

- Timing variability across repeated runs in the same MATLAB session: For reproducible demo timing, use `rng('default')` at the beginning of demos.