Solver and options

Practical usage patterns and the most common options.

Requirements

These examples use MATLAB built-in `quaternion` class (Aerospace Toolbox).

```
hasQuat = true;
try
    quaternion(0,0,0,0);
catch
    hasQuat = false;
end

if ~hasQuat
    disp('doc_SolverAndOptions: MATLAB class "quaternion" not available. Examples
are skipped.');
end
```

0) Setup

If you already have A in the workspace, this section does nothing.

```
if hasQuat
    if ~exist('A','var') || isempty(A)
        rng(1);
        n = 5;
        A = quaternion(randn(n),randn(n),randn(n),randn(n));
    end
end
```

1) Typical calls

```
if hasQuat
    fprintf("\nTypical calls");
    % Default profile (balanced):
    fprintf("\nDefault profile (balanced):");
    [lam,~,res] = leigqNEWTON(A);
    %
    fprintf("\n=== leigqNEWTON (default) ===\n");
    fprintf("hits: %d\n", numel(lam));
    fprintf("res:  median = %.3e, max = %.3e\n", median(res), max(res));
    fprintf("lam:\n");
    %disp(lam(1:min(end,5)));   % show first few only
    disp(lam);

    % Reliable profile (often finds more hits on dense A) + reproducible run:
    fprintf("\nReliable profile:");
```

```
    [lam,~,res,info] =
leigqNEWTON(A,'SolveProfile','reliable','Seed',1,'InfoLevel','summary');
    %
    fprintf("\n=== leigqNEWTON (reliable, Seed=1) ===\n");
    fprintf("hits: %d\n", numel(lam));
    fprintf("res:  median = %.3e, max = %.3e\n", median(res), max(res));
    fprintf("lam:\n");
    disp(lam);
    fprintf("info:\n");
    disp(info)                 % summary struct (already small)

    % Also return a distinct representative set (lamU):
    fprintf("\nAlso return a distinct representative set:");
    [lam,V,res,info,lamU,VU,resU] =
leigqNEWTON(A,'SolveProfile','reliable','Seed',1);
    %
    fprintf("\n=== leigqNEWTON (reliable, distinct set) ===\n");
    fprintf("ALL hits: %d | res(med,max) = (%.3e, %.3e)\n", ...
        numel(lam), median(res), max(res));
    fprintf("lam:\n");
    disp(lam);
    fprintf("DISTINCT:  %d | resU(med,max) = (%.3e, %.3e)\n", ...
        numel(lamU), median(resU), max(resU));
    fprintf("lamU:\n");
    disp(lamU);
end
```

```
Typical calls
Default profile (balanced):
=== leigqNEWTON (default) ===
hits: 2
res:  median = 7.160e-14, max = 1.432e-13
lam:
     0.5 - 0.5i - 0.5j - 0.5k
     0.5 + 0.5i + 0.5j - 0.5k
Reliable profile:
=== leigqNEWTON (reliable, Seed=1) ===
hits: 2
res:  median = 2.073e-12, max = 4.147e-12
lam:
     0.5 + 0.5i + 0.5j - 0.5k
     0.5 - 0.5i - 0.5j - 0.5k
info:
    {1×1 struct}
    {1×1 struct}
    {1×1 struct}
Also return a distinct representative set:
=== leigqNEWTON (reliable, distinct set) ===
ALL hits: 2 | res(med,max) = (2.073e-12, 4.147e-12)
lam:
     0.5 + 0.5i + 0.5j - 0.5k
     0.5 - 0.5i - 0.5j - 0.5k
DISTINCT:  2 | resU(med,max) = (2.073e-12, 4.147e-12)
lamU:
     0.5 + 0.5i + 0.5j - 0.5k
     0.5 - 0.5i - 0.5j - 0.5k
```

2) Reading info

If requested, `info` is a cell array: * info{1} is the summary struct * info{2:end} are per-trial structs when InfoLevel='full'

```
if hasQuat
    fprintf("\nReading info: info{1} includes\n");
    S = info{1};
    fprintf("Runs: %d accepted / %d converged (targetK=%d)\n", ...
        S.nAccepted, S.nConverged, S.targetK);
    fprintf("Trials: total=%d, iters total=%d\n", S.trialsTotal, S.itersTotal);
    fprintf("Distinct hits: %d\n", S.summary.nDistinct);
end
```

```
Reading info: info{1} includes
Runs: 2 accepted / 2 converged (targetK=2)
Trials: total=2, iters total=13
Distinct hits: 2
```

3) Reproducibility controls

The solver is stochastic due to random restarts. Use 'Seed' to reproduce results.

```
if hasQuat
    fprintf("\nReproducibility controls:");
    fprintf("\n=== leigqNEWTON (seed) ===\n");
    [lam1,~,res1] = leigqNEWTON(A,'Seed',1);
    [lam2,~,res2] = leigqNEWTON(A,'Seed',1);
    fprintf('Same seed -> same hit count: %d vs %d\n', numel(lam1), numel(lam2));
    fprintf('Residual medians: %.2e vs %.2e\n', median(res1), median(res2));
end
```

```
Reproducibility controls:
=== leigqNEWTON (seed) ===
Same seed -> same hit count: 2 vs 2
Residual medians: 2.07e-12 vs 2.07e-12
```

4) Controlling the requested number of hits

You can request a desired number K via: leigqNEWTON(A, K, ...) or leigqNEWTON(A, 'Num', K, ...)

```
if hasQuat
    fprintf("\nControlling the requested number of hits:");
    K = 9;
    fprintf("\n=== leigqNEWTON (set number of eigenvalues) ===\n");
    [lamK,~,resK] = leigqNEWTON(A, K, 'SolveProfile','reliable','Seed',1);
    fprintf('Requested K=%d, returned hits=%d, median(res)=%.2e\n', K, numel(lamK),
median(resK));
end
```

```
Controlling the requested number of hits:
```

```
=== leigqNEWTON (set number of eigenvalues) ===
Requested K=9, returned hits=2, median(res)=2.07e-12
```

5) Distinct representatives

The raw list `lam` may contain duplicates. If you request outputs 5–7, you also get `lamU` which groups near-identical hits (tolerance-based).

```
if hasQuat
    fprintf("\nDistinct representatives:");
    fprintf("\n=== leigqNEWTON (reliable, distinct set) ===\n");
    [lam,~,res,~,lamU,~,resU] = leigqNEWTON(A,'SolveProfile','reliable','Seed',1);
    fprintf('Raw hits: %d, distinct reps: %d\n', numel(lam), numel(lamU));
    fprintf('median(res)=%.2e, median(resU)=%.2e\n', median(res), median(resU));

    % If you want stronger/looser grouping, set DistinctTolAbs explicitly:
    % [~,~,~,~,lamU2] =
leigqNEWTON(A,'SolveProfile','reliable','Seed',1,'DistinctTolAbs',1e-5);
end
```

```
Distinct representatives:
=== leigqNEWTON (reliable, distinct set) ===
Raw hits: 2, distinct reps: 2
median(res)=2.07e-12, median(resU)=2.07e-12
```

6) Triangular matrices: initialization behavior

If A is triangular, `TriangularInit` seeds with diagonal/basis vectors. Disable it if you want purely random initialization (e.g., for benchmarking).

```
if hasQuat
    fprintf("\nTriangular matrices:");
    fprintf("\n=== leigqNEWTON (not triangular init) ===\n");
    [lamT,~,resT] = leigqNEWTON(A,'TriangularInit',false,'Seed',1);
    fprintf('TriangularInit=false -> hits=%d, median(res)=%.2e\n', numel(lamT),
median(resT));

    % Diagonal shortcut (default) avoids Newton on diagonal matrices.
    % To force Newton even for diagonal A, set:
    % leigqNEWTON(A,'TriangularShortcut','off',...)
end
```

```
Triangular matrices:
=== leigqNEWTON (not triangular init) ===
TriangularInit=false -> hits=2, median(res)=2.07e-12
```

7) Performance knobs: speed vs robustness

Two common switches that reduce post-processing costs:

- `RefineV = false` (skip final vector recomputation)
- `ResidualNormalized = false` (report raw `|A*v - lam*v|`)

```
if hasQuat
    fprintf("\nPerformance knobs (speed vs robustness):");
    fprintf("\n=== leigqNEWTON (fast, no refine) ===\n");
    [lamF,~,resF] =
leigqNEWTON(A,'Seed',1,'RefineV',false,'ResidualNormalized',false);
    fprintf('Fast knobs -> hits=%d, median(raw-res)=%.2e\n', numel(lamF),
median(resF));
end
```

```
Performance knobs (speed vs robustness):
=== leigqNEWTON (fast, no refine) ===
Fast knobs -> hits=2, median(raw-res)=1.89e-11
```

8) Where to see the full option list

Type in MATLAB: help leigqNEWTON doc leigqNEWTON

See also

doc_leigqNEWTON, checkNEWTON, leigqNEWTON_refine_batch, leigqNEWTON_cert_resMin