leigqNEWTON

API Reference

Core solver

leigqNEWTON

```
help leigqNEWTON
```

```
leigqNEWTON  Left eigenpairs of a quaternionic matrix via a Newton-type solver (stand-alone).

   [lambda, V, res] = leigqNEWTON(A)
   [lambda, V, res, info] = leigqNEWTON(A, Name,Value,...)
   [lambda, V, res, info, lambdaU, VU, resU] = leigqNEWTON(A, Name,Value,...)
   [...] = leigqNEWTON(A, Num, Name,Value,...)          %% convenience positional Num

 Computes quaternionic LEFT eigenpairs (lambda, V) of a square quaternion matrix A:
       A*v = lambda*v,   v ~= 0,
 where lambda is a quaternion acting on the LEFT.

 This implementation is self-contained and uses only:
   - MATLAB built-in quaternion class (Aerospace Toolbox),
   - standard MATLAB linear algebra on REAL/COMPLEX embeddings built locally.

 ------------------------------------------------------------------------
 Outputs (types and meaning)

   lambda   : K-by-1 quaternion
       The K eigenvalue "hits" accepted by the solver (K is controlled by Num).
       These may contain duplicates (the solver does not enforce uniqueness).

   V        : n-by-K quaternion
       Corresponding right eigenvectors as columns (A*V(:,k)  approx  lambda(k)*V(:,k)).

   res      : K-by-1 double
       Final residual norms reported for each accepted eigenpair.
       If ResidualNormalized=true (default), res is scale-invariant:
           res = ||A*v - lambda*v|| / den(A,lambda,v),
       where den(...) is a mild scale factor (see local_residual_den).
       If ResidualNormalized=false, res is the raw Euclidean norm.

   info     : cell array (only if requested)
       info{1} is a summary struct; info{2:end} are per-trial structs when InfoLevel='full'.
       In addition, info{1}.distinct describes how lambdaU was formed (if requested).

   lambdaU  : Ku-by-1 quaternion  (only if requested)
       A DISTINCT representative set extracted from lambda using a tolerance.
       Each group of near-identical hits contributes one representative, chosen by:
         (i) significantly better residual (if present), otherwise
         (ii) "prettiness" (prefers exact integers/zeros over small numerical tails).
       By design, lambdaU is a subset of the returned lambda (representatives are picked
       from existing hits; no additional cleaning is applied).

   VU       : n-by-Ku quaternion  (only if requested)
   resU     : Ku-by-1 double      (only if requested)
       The vectors/residuals corresponding to lambdaU.

 ------------------------------------------------------------------------
 Profiles (recommended)
 leigqNEWTON is restart-based; success depends strongly on the TRIAL budget and
 Newton iterations per trial. SolveProfile presets apply only to options you did
```

```
NOT specify explicitly:
  'fast'     : small budgets; quick scans (may return fewer eigenpairs).
  'default'  : balanced; typically succeeds for generic dense matrices.
  'reliable' : larger budgets + auto-extension; aims to return (almost) all eigenpairs.

------------------------------------------------------------------------
Options (Name,Value)  (aliases accepted; unknown options are ignored)

  Desired count:
    'Num'/'K'/'NEigs'/'NEV'         : number of eigenpairs requested (default n)
    (positional) Num               : leigqNEWTON(A, 9, ...) sets Num=9

  Trial budgets / restarts:
    'Trials'/'Restarts'/'NTrials'  : initial trial budget (profile-dependent default)
    'TrialsFactor'                : convenience Trials := TrialsFactor*n (if Trials not set)
    'MaxTrials'/'MaxRestarts'      : hard cap on total trials (also disables AutoExtend)
    'AutoExtend'                  : true/false (profile-dependent default)
    'MaxTrialsCap'/'TrialsCap'    : cap used by AutoExtend
    'ExtendBy'/'ExtendFactor'     : how much to increase budget when auto-extending

  Newton / acceptance:
    'MaxIter'/'MaxIt'              : Newton iterations per trial (profile-dependent default)
    'Tol'/'ResTol'                 : convergence tolerance (raw residual is used internally)
    'Damping'/'Alpha'/'StepSize'   : initial step size alpha in (0,1] (default 1)
    'Backtrack'/'LineSearch'       : backtracking line search (default true)
    'MinAlpha'/'AlphaMin'          : minimum alpha in backtracking (default 1/64)

  Distinct representative extraction (for lambdaU):
    'DistinctTol'/'DistinctTolAbs'/'UniqueTol' : absolute grouping tolerance (default max(1e-6, sqrt(Tol)))
    'DistinctTolRel'                          : relative grouping tolerance (default 0)
    'DistinctResFactor'                       : residual "order-of-magnitude" factor (default 10)
      Note: The distinct-grouping tolerance is intentionally more tolerant than Tol by default.
      You still see all raw hits in lambda; lambdaU only groups near-duplicates into representatives.

  Reproducibility / logging:
    'Seed'/'RNG'                   : RNG seed (default [])
    'Verbose'/'Disp'               : 0/1 (default 0)
    'InfoLevel'/'InfoMode'         : 'summary'|'full' OR numeric 0/1/2 (2='full')

  Robustness / post-processing:
    'VerifyZeroNull'/'ZeroNullVerify' : verify Aq*x approx 0 in the zero-eigenvalue pre-pass (default true)
    'ZeroNullTol'                     : tolerance for the zero-eig pre-pass (raw units). [] -> auto
    'UseNullFallbackLA'/'FallbackLA'  : if verification fails, recompute nullspace using real 4n embedding (defaul
    'ResidualNormalized'              : if true, res output is scale-invariant (default true)
    'RefineV'                         : recompute eigenvectors for final lambdas via real least-squares/SVD (defau

  Initialization / triangular handling:
    'Lambda0'/'Lam0'/'InitLambda'  : initial lambda guess (scalar quaternion)
    'V0'/'X0'/'InitVec'            : initial eigenvector guess (n-by-1 quaternion)
    'TriangularInit'/'TriInit'     : if true, deterministic diagonal/basis seeding for (detected) triangular A (de
    'TriTol'/'TriangularTol'       : triangular detection tolerance (default 0)
    'IsTriangular'/'TriangularFlag'/'ForceTriangular' : user override for triangular flag (true/false)
    'TriangularShortcut'/'TriShortcut' : 'diag' (default) shortcut for diagonal A; 'off' forces Newton

Notes on triangular matrices:
  If TriangularInit is true and A is treated as triangular, the solver uses
  deterministic diagonal/basis seeding for the first n trials. This often leads
  to very small iteration counts (sometimes 0), which is expected.
  To benchmark "generic" random initialization on triangular A, set:
      'TriangularInit', false.

------------------------------------------------------------------------
Useful one-liners (copy/paste)
  % Default (balanced):
```

```
  [lam,V,res] = leigqNEWTON(A);

  % Also return distinct representatives (lambdaU):
  [lam,V,res,info,lamU,VU,resU] = leigqNEWTON(A,'SolveProfile','reliable');
  % Add a larger distinct-grouping tolerance explicitly:
  [lam,V,res,info,lamU,VU,resU] = leigqNEWTON(A,'SolveProfile','reliable','DistinctTolAbs', 1e-5);  % or 3e-5, 1e-
  % If you want it scale-aware (relative as well)
  [lam,V,res,info,lamU,VU,resU] = leigqNEWTON(A, 'SolveProfile','reliable','DistinctTolAbs', 1e-5,'DistinctTolRel

  % Rule of thumb: Start with DistinctTolAbs = 10*sqrt(Tol) (or 30*sqrt(Tol) for very defective/triangular-ish cas
    This affects only deduplication into lamU. You still see all raw hits in lam, so being more tolerant here is u

  % Request exactly 9 eigenpairs (two equivalent forms):
  [lam,V,res] = leigqNEWTON(A, 9, 'SolveProfile','reliable');
  [lam,V,res] = leigqNEWTON(A,'Num',9,'SolveProfile','reliable');

  % Reproducible run:
  [lam,V,res] = leigqNEWTON(A,'Seed',1,'SolveProfile','default');

--------------------------------------------------------------------------
Legacy name mapping
  leigqVANILA  ->  leigqNEWTON

See also: quaternion, parts, null, rank, leigqNEWTON_refine_polish, leigqNEWTON_cert_resMin
```

Certification

```
    leigqNEWTON_cert_resMin
    leigqNEWTON_cert_resPair
```

```
help leigqNEWTON_cert_resMin
```

```
leigqNEWTON_cert_resMin  Minimum-residual certificate for a quaternion LEFT eigenvalue candidate.

  resMin = leigqNEWTON_cert_resMin(A, lambda)
  [resMin, xMin] = leigqNEWTON_cert_resMin(A, lambda)
  [resMin, xMin, rMin, info] = leigqNEWTON_cert_resMin(A, lambda, Name,Value,...)

Defines the certificate-like quantity
  resMin(A,lambda) := min_{||x||_2=1} ||A*x - lambda*x||_2,
computed via a real/complex embedding and a smallest-singular-value routine.

Inputs
  A      : n-by-n quaternion (or numeric; interpreted as real quaternion)
  lambda : scalar quaternion or K-by-1 quaternion array

Name,Value options (selected)
  'Method'             : 'svd'|'eigs' (default 'svd')  % internal method for sigma_min
  'MaxIter'            : iterations for iterative methods (if used)
  'Tol'                : tolerance for iterative methods (if used)
  'ResidualNormalized' : if true, return scale-invariant resMin (default true)
  'UseLambda'          : 'lambda'|'lambdac' (default 'lambda')  % choose which lambda vector to use
  'LambdaC'            : cleaned lambdas (required if UseLambda='lambdac')

Outputs
  resMin : K-by-1 double   (certificate value for each lambda)
  xMin   : n-by-K quaternion (minimizers; columns; returned if requested)
  rMin   : cell or struct  (auxiliary residual vectors/metrics; for diagnostics)
  info   : struct          (method statistics; timings; etc.)

Useful one-liners
```

```
  r = leigqNEWTON_cert_resMin(A, lam(1));
  [r,x] = leigqNEWTON_cert_resMin(A, lam, 'ResidualNormalized',true);

 See also: leigqNEWTON_cert_resPair, leigqNEWTON_refine_lambda, leigqNEWTON
```

```
help leigqNEWTON_cert_resPair
```

```
 leigqNEWTON_cert_resPair  Residual certificate for a quaternion LEFT eigenpair (lambda,v).

   resPair = leigqNEWTON_cert_resPair(A, lambda, v)
   [resPair, rVec] = leigqNEWTON_cert_resPair(A, lambda, v)
   [resPair, rVec, info] = leigqNEWTON_cert_resPair(A, lambda, v, Name,Value,...)

 Computes ||A*v - lambda*v||_2 for one or many eigenpairs. If v is provided as
 n-by-K (columns), lambda may be scalar or K-by-1.

 Inputs
   A      : n-by-n quaternion (or numeric; interpreted as real quaternion)
   lambda : scalar quaternion or K-by-1 quaternion array
   v      : n-by-1 or n-by-K quaternion array (eigenvectors in columns)

 Name,Value options (selected)
   'NormalizeV'         : true/false (default true). If true, normalize each column of v.
   'ResidualNormalized' : true/false (default false). If true, return a scale-invariant residual.

 Outputs
   resPair : K-by-1 double     (residual norm(s))
   rVec    : cell array        (optional residual vectors; for diagnostics)
   info    : struct            (diagnostics; normalization; denominators if used)

 Useful one-liners
   r = leigqNEWTON_cert_resPair(A, lam(1), V(:,1));
   r = leigqNEWTON_cert_resPair(A, lam, V, 'NormalizeV',true);

 See also: leigqNEWTON_cert_resMin, leigqNEWTON_refine_polish, leigqNEWTON
```

Refinement

```
    leigqNEWTON_init_vec
    leigqNEWTON_refine_polish
    leigqNEWTON_refine_lambda
    leigqNEWTON_refine_auto
    leigqNEWTON_refine_batch
```

```
help leigqNEWTON_refine_batch
```

```
 leigqNEWTON_refine_batch  Refine + certify a list of eigenvalue candidates.

   [lambdaRef, vRef] = leigqNEWTON_refine_batch(A, lambda0)
   [lambdaRef, vRef, cert] = leigqNEWTON_refine_batch(A, lambda0, Name,Value,...)

 Batch wrapper that turns coarse candidates lambda0 into refined/certified
 eigenvalues (and optionally polished eigenvectors). This is the recommended
 “production” post-processing step after leigqNEWTON.

 Inputs
   A       : n-by-n quaternion
   lambda0 : K-by-1 quaternion candidates (raw or cleaned)

 Name,Value options (selected)
   'Mode'         : 'auto'|'rand+fminsearch'|'fminsearch'|'none' (default 'auto')
```

```
  'TargetResMin' : target certificate value (default 1e-13)
  'DoPolish'     : true/false (default true)

Outputs
  lambdaRef : K-by-1 quaternion (refined)
  vRef      : n-by-K quaternion (associated vectors; columns)
  cert      : struct with fields such as:
              resMin, resPair, lambdaClean, acceptedMask, timings, ...

Useful one-liners
  [lamR,vR,cert] = leigqNEWTON_refine_batch(A, lam);
  [lamR,vR,cert] = leigqNEWTON_refine_batch(A, lam, 'DoPolish',true,'TargetResMin',1e-14);

See also: leigqNEWTON_refine_auto, leigqNEWTON_refine_lambda, leigqNEWTON_cert_resPair
```

Sphere module

```
leigqNEWTON_sphere_sample
leigqNEWTON_sphere_refine
```

```
help leigqNEWTON_sphere_refine
```

```
leigqNEWTON_sphere_refine  Refine/validate sphere candidates and decide sphere-vs-artifact (advanced engine).

  This routine is the engine behind LEIGQNEWTON_SPHERE_VALIDATE. It is intended
  for sphere-hunting experiments: it refines sampled eigenvalue candidates, computes
  residual certificates, and (optionally) verifies a detected sphere by grid tests
  and refitting. Use LEIGQNEWTON_SPHERE_VALIDATE for a user-facing entry point.

Canonical form (case list + index)
  [A, lamAll, resAll, lamSamples, resSamples, sph, conf, out] = ...
      leigqNEWTON_sphere_refine(cases, k, Name,Value,...)

Convenience overloads
  leigqNEWTON_sphere_refine(caseStruct, Name,Value,...)
  leigqNEWTON_sphere_refine(A, lamAll, lamSamples, Name,Value,...)

Inputs
  cases / caseStruct  Case container (struct, struct array, or cell array). Each case
                      should provide A and lambda candidates.
  k                  Case index (when cases is a list). Use [] for a single case struct.

Positional form inputs
  A          n-by-n quaternion matrix.
  lamAll     M-by-1 quaternion, candidate list.
  lamSamples K-by-1 quaternion, DISTINCT samples (may be []).

Name,Value options (selected)
  'RefineArgs'        Cell array of Name,Value passed to LEIGQNEWTON_REFINE_AUTO.
  'TargetRes'         Target residual for polishing (default 1e-14).
  'VerifySphere'      Enable sphere validation tests (default true).
  'SphereTolRel'      Relative tolerance for sphere validation (default 5e-3).
  'GridTheta','GridPhi'  Sphere grid resolution (defaults 8 and 16).
  'RefineGrid'        Refine grid points (default true).
  'RefitSphere'       Refit sphere after refinement (default true).
  'Verbose'           0/1/2 (default 1).

Outputs
  A, lamAll, lamSamples  Refined lists.
  resAll, resSamples     Certificate-like residual values (via LEIGQNEWTON_CERT_RESMIN).
  sph                    Sphere model struct (empty if none/invalid).
```

```
  conf                  Confidence indicator/struct.
  out                   Detailed diagnostics.

Examples
  % Validate a detected sphere (positional form):
  [A, lamAll2, resAll, lamS2, resS, sph2] = leigqNEWTON_sphere_refine(A, lamAll, lamSamples);

See also leigqNEWTON_sphere_validate, leigqNEWTON_sphere_sample,
         leigqNEWTON_refine_auto, leigqNEWTON_cert_resMin
```