Refinement and Certificates

This page focuses on post-processing: Certificates to assess quality and refinement to improve candidates

0) Setup: requirements and a reproducible demo matrix

```
hasQuat = true;
try
    quaternion(0,0,0,0);
catch
    hasQuat = false;
end
if ~hasQuat
    disp('This toolbox requires MATLAB''s built-in quaternion class
(quaternion(w,x,y,z)).');
    disp('Examples in this page are skipped.');
    return;
end

% Ensure toolbox root is on the path (add only the root, no genpath).
if exist('leigqNEWTON','file') ~= 2
    thisFile = mfilename('fullpath');
    if ~isempty(thisFile)
        rootGuess = fileparts(fileparts(fileparts(thisFile))); % .../docs/source ->
toolbox root
        if exist(fullfile(rootGuess,'leigqNEWTON.m'),'file')
            addpath(rootGuess);
            rehash toolboxcache
        end
    end
end
if exist('leigqNEWTON','file') ~= 2
    error('leigqNEWTON not found on the MATLAB path. Add the toolbox root folder.');
end

% A reproducible demo matrix
rng(1);
n = 5;
A = quaternion(randn(n),randn(n),randn(n),randn(n));
```

1) Eigenvalue-only certificate: resMin(lambda)

The certificate `resMin` is a **left eigenvalue residual**: it measures how close `lambda` is to the left spectrum (smaller is better).

```
[~, V, res, info, lambdaU, VU, resU] =
leigqNEWTON(A,'SolveProfile','default','Seed',1);
```

```
lambda = lambdaU;
resMin = leigqNEWTON_cert_resMin(A, lambda);

disp('First few distinct eigenvalue candidates and their resMin certificates:');
```

```
First few distinct eigenvalue candidates and their resMin certificates:
```

```
kShow = min(5, numel(lambda));
disp(table(lambda(1:kShow), resMin(1:kShow), 'VariableNames', {'lambda','resMin'}));
```

lambda	resMin
1×1 quaternion	2.7112e-13
1×1 quaternion	3.2585e-16
1×1 quaternion	1.3791e-17

2) Eigenpair certificate: resPair(lambda,v)

The eigenpair certificate requires an eigenvector v. For robustness, we build an initial `v0` using `leigqNEWTON_init_vec` and then certify the pair.

```
lam0 = lambda(1);
v0   = leigqNEWTON_init_vec(A, lam0);

[resPair0, resPairRaw0, vUnit0] = leigqNEWTON_cert_resPair(A, lam0, v0);
fprintf('Initial resPair for first candidate: %.3e\n', resPair0);
```

```
Initial resPair for first candidate: 2.719e-13
```

3) Polish refinement of the eigenpair

Use Newton polishing to improve `(lam0,v0)` → `(lam1,v1)`.

```
[lam1, v1, res1, info1] = leigqNEWTON_refine_polish(A, lam0, v0, ...
    'TolRes', 1e-14, 'TolStep', 1e-14, 'MaxIter', 20, 'Verbose', 0);

% Re-certify after polishing
resMin1 = leigqNEWTON_cert_resMin(A, lam1);
resPair1 = leigqNEWTON_cert_resPair(A, lam1, v1);

disp('Certificates before/after polishing:');
```

```
Certificates before/after polishing:
```

```
disp(table([resMin(1); resMin1], [resPair0; resPair1], ...
    'VariableNames', {'resMin','resPair'}, 'RowNames', {'before','after'}));
```

	resMin	resPair
before	2.7112e-13	2.7188e-13
after	2.3529e-13	2.6994e-13

See also

leigqNEWTON_cert_resMin, leigqNEWTON_cert_resPair leigqNEWTON_refine_polish, leigqNEWTON_refine_batch