Eigenvalue problem - simple iterations

Preliminaries

Clear all symbols from previous evaluations to avoid conflicts

In[ ]:=
```
Clear["Global`*"]
```

Problem

[Taken from Trefethen, Bau: Numerical Linear Algebra, SIAM 1997, p. 315]

In[1]:=
```
n = 1000;
A = SparseArray[{{i_, i_} → 0.5 + Sqrt[i],
    {i_, j_} /; Abs[i - j] == 1 → 1.0, {i_, j_} /; Abs[i - j] == 100 → 1.0}, {n, n}];
b = ConstantArray[1.0, n];
```

The matrix is symmetric positive definite - all eigenvalues are positive

In[171]:=
```
eigenA = Sort[Eigenvalues[A]];
maxλ = Max[eigenA]; minλ = Min[eigenA];
Print["ϰ(A)  = ", Max[eigenA] / Min[eigenA]]
Print["max(λ)  = ", maxλ]
Print["min(λ)  = ", minλ]
ListPlot[{eigenA}, PlotRange → All, AxesLabel → {"n", "eigenvalue"}]
```

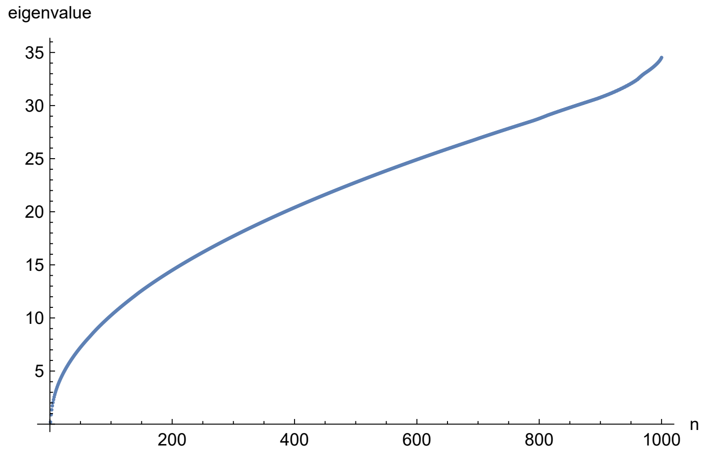
Eigenvalues: Because finding 1000 out of the 1000 eigenvalues and/or eigenvectors is likely to be faster with dense matrix methods, the sparse input matrix will be converted. If fewer eigenvalues and/or eigenvectors would be sufficient, consider restricting this number using the second argument to Eigenvalues.

κ(A) = 173.448839396

max(λ) = 34.5227980867

min(λ) = 0.199037354224

Out[176]=



Maximal eigenvalue

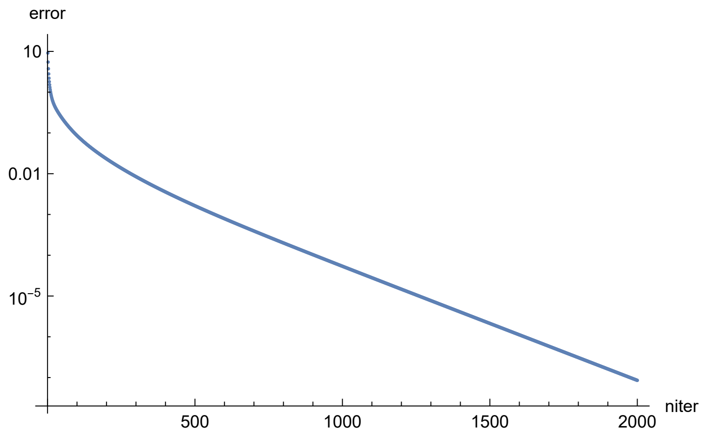
Power iteration - quite slow convergence

In[10]:=

```
niter = 2000;
error = ConstantArray[0.0, niter];
x = b / Norm[b];
Do[
  w = A.x;
  λ = x.w;
  error⟦i⟧ = Abs[maxλ - λ];
  x = w / Norm[w],
  {i, 1, niter}
 ];
Print["Exact maximum   λ: ", maxλ]
Print["Power iteration λ: ", λ, ", error after ", niter, " iterations: ", error⟦niter⟧]
ListLogPlot[{error}, PlotRange → All, AxesLabel → {"niter", "error"}]
```

```
Exact maximum   λ: 34.5227980867
Power iteration λ: 34.5227980014, error after 2000 iterations: 8.53360049291×10^-8
```

Out[16]=



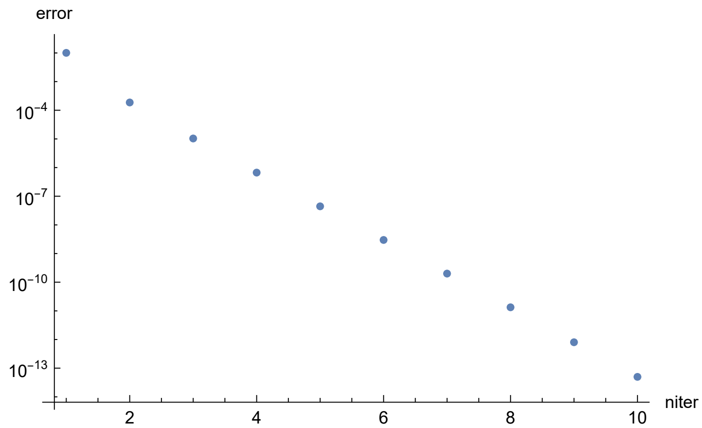
Inverse iteration - reasonable convergence if we set the shift μ close to actual eigenvalue

In[17]:=

```
niter = 10;
error = ConstantArray[0.0, niter];
x = b / Norm[b];
μ = 34.5;
Do[
  w = LinearSolve[A - μ IdentityMatrix[n], x];
  x = w / Norm[w];
  λ = x.(A.x);
  error[[i]] = Abs[maxλ - λ],
  {i, 1, niter}
 ];
Print["Exact maximal λ: ", maxλ]
Print["Inverse iteration λ: ", λ,
 ", error after ", niter, " iterations: ", error[[niter]]]
ListLogPlot[{error}, PlotRange → All, AxesLabel → {"niter", "error"}]
```

```
Exact maximal λ: 34.5227980867
Inverse iteration λ: 34.5227980867, error after 10 iterations: 4.97379915032×10^-14
```

Out[24]=



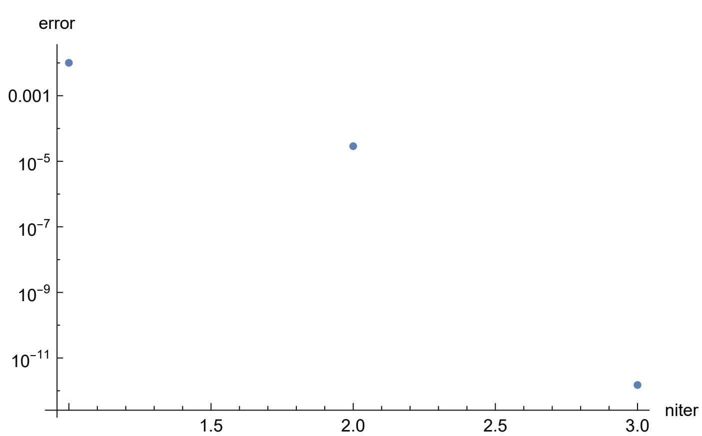
Rayleigh quotient iteration - fast convergence if we set initial guess of λ close to actual eigenvalue

In[53]:=

```
niter = 3;
error = ConstantArray[0.0, niter];
x = b / Norm[b];
(* x=xInvIter; *)
λ = 34.5;
Do[
  w = LinearSolve[A - λ IdentityMatrix[n], x];
  x = w / Norm[w];
  λ = x.(A.x);
  error[[i]] = Abs[maxλ - λ],
  {i, 1, niter}
 ];
Print["Exact maximal λ: ", maxλ, ", Rayleigh quotient iteration λ: ", λ]
ListLogPlot[{error}, PlotRange → All, AxesLabel → {"niter", "error"}]
```

```
Exact maximal λ: 34.5227980867, Rayleigh quotient iteration λ: 34.5227980867
```

Out[59]=



Minimal eigenvalue

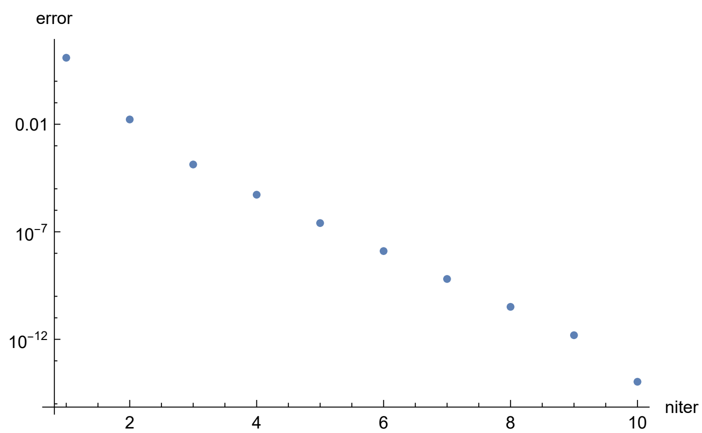
Inverse iteration - again quite fast convergence

In[213]:=

```
niter = 10;
error = ConstantArray[0.0, niter];
x = b / Norm[b];
μ = 0.0;
Do[
  w = LinearSolve[A - μ IdentityMatrix[n], x];
  x = w / Norm[w];
  λ = x.(A.x);
  error⟦i⟧ = Abs[minλ - λ],
  {i, 1, niter}
 ];
Print["Exact minimal λ: ", minλ, ", inverse iteration λ: ", λ]
ListLogPlot[{error}, PlotRange → All, AxesLabel → {"niter", "error"}]
```

```
Exact minimal λ: 0.199037354224, inverse iteration λ: 0.199037354224
```

Out[219]=



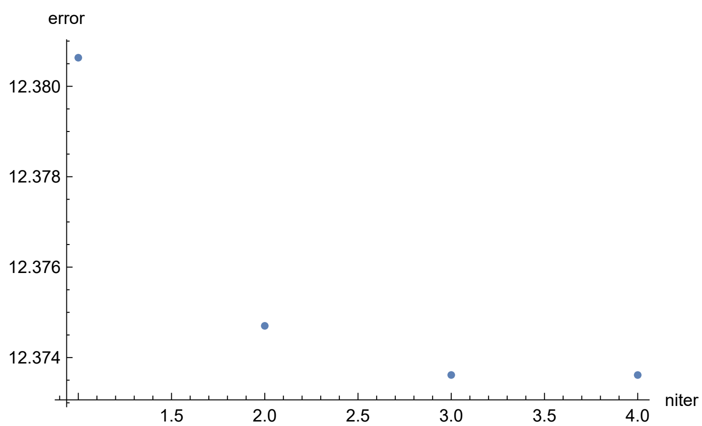
Rayleigh quotient iteration - fast convergence but to an incorrect eigenvalue 12.57265157916

In[325]:=

```
niter = 4;
error = ConstantArray[0.0, niter];
x = b / Norm[b]; (* not a good initial guess for the minimal eigenvalue *)
λ = 0.0;
Do[
  w = LinearSolve[A - λ IdentityMatrix[n], x];
  x = w / Norm[w];
  λ = x.(A.x);
  error[[i]] = Abs[Min[eigenA] - λ],
  {i, 1, niter}
 ];
Print["Exact minimal λ: ", minλ, ", inverse iteration λ: ", λ]
ListLogPlot[{error}, PlotRange → All, AxesLabel → {"niter", "error"}]
```

```
Exact minimal λ: 0.199037354224, inverse iteration λ: 12.5726515792
```

Out[331]=



Rayleigh quotient iteration - fast convergence if we get close enough using inverse iteration first

In[351]:=

```
niterInverse = 2;
niterRayleigh = 3;
error = ConstantArray[0.0, niterInverse + niterRayleigh];
(* inverse iteration *)
x = b / Norm[b];
μ = 0.0;
Do[
  w = LinearSolve[A - μ IdentityMatrix[n], x];
  x = w / Norm[w];
  λ = x.(A.x);
  error[[i]] = Abs[Min[eigenA] - λ],
  {i, 1, niterInverse}
 ];
(* Rayleigh quotient iteration *)
Do[
  w = LinearSolve[A - λ IdentityMatrix[n], x];
  x = w / Norm[w];
  λ = x.(A.x);
  error[[niterInverse + i]] = Abs[Min[eigenA] - λ],
  {i, 1, niterRayleigh}
 ];
Print["Exact minimal λ: ", minλ, ", inverse iteration λ: ", λ]
ListLogPlot[{error}, PlotRange → All, AxesLabel → {"niter", "error"}]
```

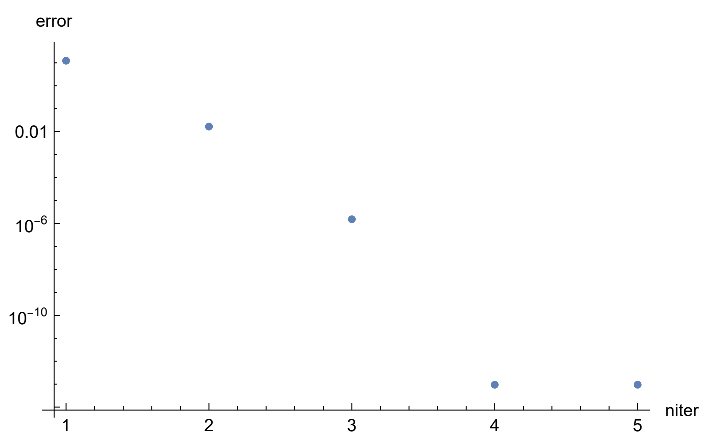
LinearSolve: Result for LinearSolve of badly conditioned matrix

{{1.30096264578, 1., 0., 0., 0., 0., 0., 0., 0., 0., ≪990≫}, {1., 1.71517620815, 1., 0., 0., 0., 0., 0., 0., 0., ≪990≫}, {0., 1., 2.03301345334, 1., 0., 0., 0., 0., 0., 0., ≪990≫}, ≪6≫, {0., 0., 0., 0., 0., 0., 0., 0., 1., 3.46324030594, ≪990≫}, ≪990≫}

may contain significant numerical errors.

```
Exact minimal λ: 0.199037354224, inverse iteration λ: 0.199037354224
```

Out[359]=



Other eigenvalues

In[360]:=

```
eigenA[[380 ;; 400]]
```

Out[360]=

```
{19.8901986235, 19.9159708767, 19.9417088787, 19.9674128469, 19.9930828924, 20.0187191157,
 20.0443216827, 20.0698907391, 20.0954263926, 20.1209287923, 20.1463980501,
 20.1718343205, 20.1972377029, 20.2226083538, 20.2479463708, 20.2732519095,
 20.2985250648, 20.3237659907, 20.3489747813, 20.3741515875, 20.3992965044}
```

Inverse iteration - finds an eigenvalue closest to the initial quess

In[365]:=

```
x = b / Norm[b];
μ = 20.0; (* initial guess *)
Do[
  w = LinearSolve[A - μ IdentityMatrix[n], x];
  x = w / Norm[w];
  λ = x.(A.x);
  Print[λ],
  {i, 1, 10}
 ];
```

```
19.9990817898
19.993525855
19.9931463251
19.9930917153
19.9930841053
19.9930830584
19.9930829151
19.9930828955
19.9930828928
19.9930828924
```

Rayleigh quotient iteration

In[368]:=

```
λ = 20.0;
x = b / Norm[b];
Do[
  w = LinearSolve[A - λ IdentityMatrix[n], x];
  x = w / Norm[w];
  λ = x.(A.x);
  Print[λ],
  {i, 1, 4}
 ];
```

```
19.9990817898

19.9933821313

19.993082933

19.9930828924
```

Basic QR algorithm

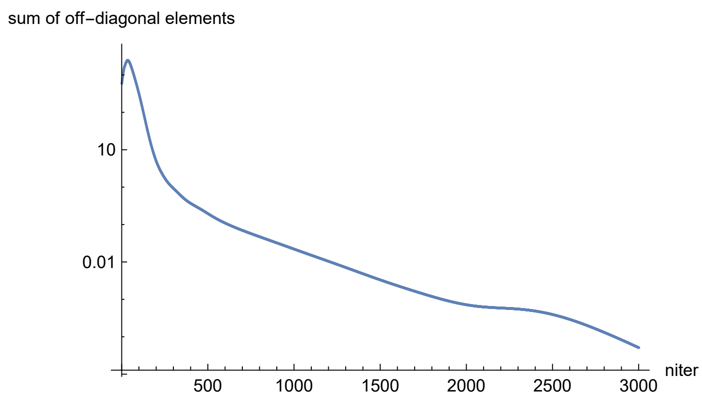
In[371]:=

```
n = 300;
A = SparseArray[{{i_, i_} → 0.5 + Sqrt[i], {i_, j_} /; Abs[i - j] == 1 → 1.0,
    {i_, j_} /; Abs[i - j] == 10 → 1.0}, {n, n}];
(* A=SparseArray[{{i_,i_}→0.5+Sqrt[i],{i_,j_}/;Abs[i-j]==1→1.0},{n,n}]; *)
niter = 3000;
```

In[374]:=

```
AA = A;
SumOffDiagQR = ConstantArray[0.0, niter + 1];
SumOfLastRowQR = ConstantArray[0.0, niter + 1];
SumOffDiagQR[[1]] = Total[Abs[LowerTriangularize[AA, -1]]^2, 2];
SumOfLastRowQR[[1]] = Total[Abs[AA[[n, 1 ;; n - 1]]]^2];
Do[
  {Q, R} = QRDecomposition[AA];
  AA = R.ConjugateTranspose[Q];
  SumOffDiagQR[[i + 1]] = Total[Abs[LowerTriangularize[AA, -1]]^2, 2];
  SumOfLastRowQR[[i + 1]] = Total[Abs[AA[[n, 1 ;; n - 1]]]^2],
  {i, 1, niter}
 ];
ListLogPlot[SumOffDiagQR, AxesLabel → {"niter", "sum of off-diagonal elements"}]
Norm[Sort[Diagonal[AA]] - Sort[Eigenvalues[A]]]
```

Out[380]=



Eigenvalues: Because finding 300 out of the 300 eigenvalues and/or eigenvectors is likely to be faster with dense matrix methods, the sparse input matrix will be converted. If fewer eigenvalues and/or eigenvectors would be sufficient, consider restricting this number using the second argument to Eigenvalues.

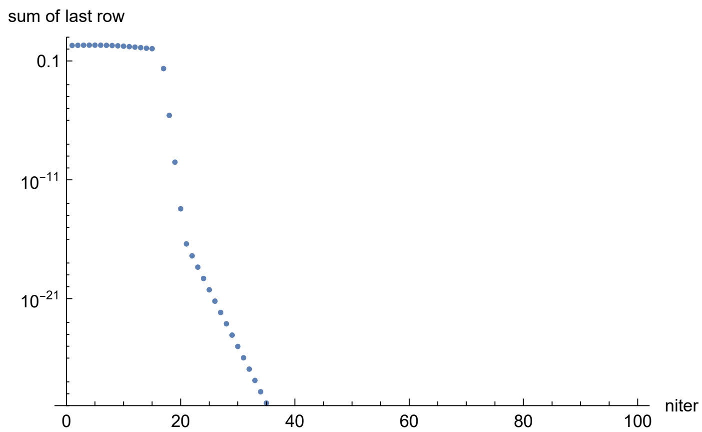
Out[381]=

```
0.00158362657794
```

In[382]:=

```
ListLogPlot[SumOfLastRowQR[[1 ;; 100]],
 AxesLabel → {"niter", "sum of last row"}, PlotRange → {10^-30, 10}]
```

Out[382]=



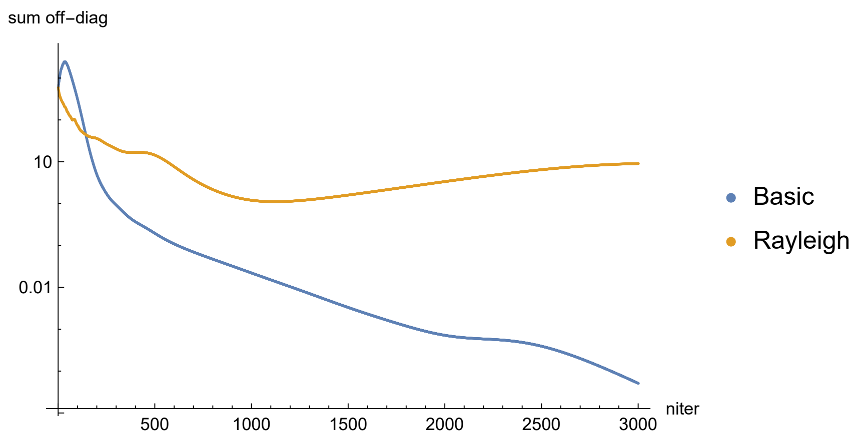
QR algorithm with shifts

Rayleigh quotient shift

In[383]:=

```
AA = A;
SumOffDiagRay = ConstantArray[0.0, niter + 1];
SumOfLastRowRay = ConstantArray[0.0, niter + 1];
SumOffDiagRay[[1]] = Total[Abs[LowerTriangularize[AA, -1]]^2, 2];
SumOfLastRowRay[[1]] = Total[Abs[AA[[n, 1 ;; n - 1]]]^2];
μ = AA[[n, n]];
Do[
  {Q, R} = QRDecomposition[AA - μ IdentityMatrix[n]];
  AA = R.ConjugateTranspose[Q] + μ IdentityMatrix[n];
  μ = AA[[n, n]];
  SumOffDiagRay[[i + 1]] = Total[Abs[LowerTriangularize[AA, -1]]^2, 2];
  SumOfLastRowRay[[i + 1]] = Total[Abs[AA[[n, 1 ;; n - 1]]]^2],
  {i, 1, niter}
 ];
ListLogPlot[{SumOffDiagQR, SumOffDiagRay},
 AxesLabel → {"niter", "sum off-diag"}, PlotLegends → {"Basic", "Rayleigh"}]
Norm[Sort[Diagonal[AA]] - Sort[Eigenvalues[A]]]
```

Out[390]=



Eigenvalues: Because finding 300 out of the 300 eigenvalues and/or eigenvectors is likely to be faster with dense matrix methods, the sparse input matrix will be converted. If fewer eigenvalues and/or eigenvectors would be sufficient, consider restricting this number using the second argument to Eigenvalues.

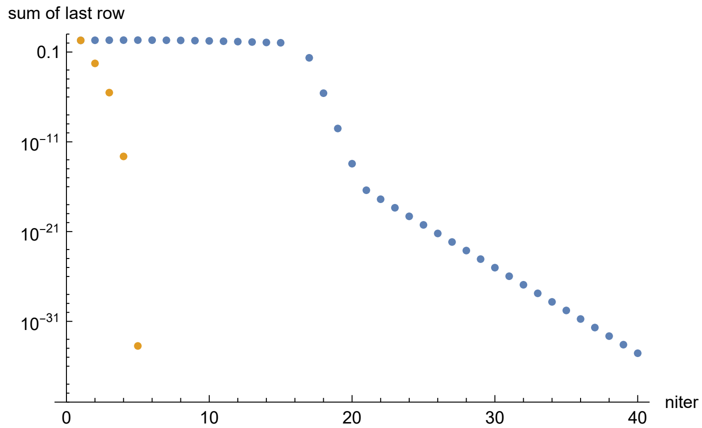
Out[391]=

1.03150363197

In[392]:=

```
ListLogPlot[{SumOfLastRowQR[[1 ;; 40]], SumOfLastRowRay[[1 ;; 20]]},
 AxesLabel → {"niter", "sum of last row"}, PlotRange → {10^-40, 10}]
```

Out[392]=



Wilkinson shift

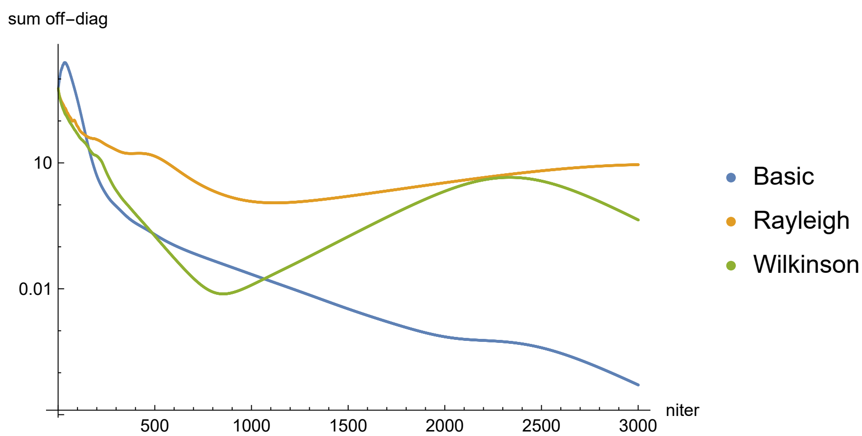
In[393]:=

```
AA = A;
SumOffDiagW = ConstantArray[0.0, niter + 1];
SumOfLastRowW = ConstantArray[0.0, niter + 1];
SumOffDiagW[[1]] = Total[Abs[LowerTriangularize[AA, -1]]^2, 2];
SumOfLastRowW[[1]] = Total[Abs[AA[[n, 1 ;; n - 1]]]^2];
Do[
  δ = (AA[[n - 1, n - 1]] - AA[[n, n]]) / 2;
  μ = AA[[n, n]] - Sign[δ] * AA[[n, n - 1]]^2 / (Abs[δ] + Sqrt[δ^2 + AA[[n, n - 1]]^2]);
  {Q, R} = QRDecomposition[AA - μ IdentityMatrix[n]];
  AA = R.ConjugateTranspose[Q] + μ IdentityMatrix[n];
  SumOffDiagW[[i + 1]] = Total[Abs[LowerTriangularize[AA, -1]]^2, 2];
  SumOfLastRowW[[i + 1]] = Total[Abs[AA[[n, 1 ;; n - 1]]]^2],
  {i, 1, niter}
 ];
ListLogPlot[{SumOffDiagQR, SumOffDiagRay, SumOffDiagW},
 AxesLabel → {"niter", "sum off-diag"}, PlotLegends → {"Basic", "Rayleigh", "Wilkinson"}]
Norm[Sort[Diagonal[AA]] - Sort[Eigenvalues[A]]]
```

General: $(-3.05959720127 \times 10^{-178})^2$ is too small to represent as a normalized machine number; precision may be lost.

General: $(-3.05959720127 \times 10^{-178})^2$ is too small to represent as a normalized machine number; precision may be lost.

Out[399]=



Eigenvalues: Because finding 300 out of the 300 eigenvalues and/or eigenvectors is likely to be faster with dense matrix methods, the sparse input matrix will be converted. If fewer eigenvalues and/or eigenvectors would be sufficient, consider restricting this number using the second argument to Eigenvalues.

Out[400]=

0.129966200058

In[401]:=

```
ListLogPlot[{SumOfLastRowQR⟦1 ;; 40⟧, SumOfLastRowRay⟦1 ;; 20⟧, SumOfLastRowW⟦1 ;; 20⟧},
 AxesLabel → {"niter", "sum of last row"}, PlotRange → {10^-40, 10}]
```

Out[401]=