3 `davidson.cxx`

Sei $\boldsymbol{v}$ ein approximativer Eigenvektor einer Matrix $\boldsymbol{H}$ zum approximativen Eigenwert λ mit Fehler (Residuum) $\boldsymbol{r}(\lambda, \boldsymbol{v}) = \boldsymbol{H}\boldsymbol{v} - \lambda\boldsymbol{v}$. In niedrigster Ordnung kann der korrigierte Eigenvektor $\boldsymbol{v} + \Delta v$ gefunden werden durch

$$\boldsymbol{H}(\boldsymbol{v} + \Delta\boldsymbol{v}) = (\lambda + \Delta\lambda)(\boldsymbol{v} + \Delta\boldsymbol{v}) \qquad \Longrightarrow \qquad \Delta\boldsymbol{v}(\lambda, \boldsymbol{v}) \approx -(\boldsymbol{D} - \lambda\mathbf{1})^{-1}\boldsymbol{r}(\lambda, \boldsymbol{v}) \tag{1}$$

wobei wir $\boldsymbol{H}$ in der Inversen durch den diagonalen Anteil $(\boldsymbol{D})_{ij} = \delta_{ij}(\boldsymbol{H})_{ii}$ von $\boldsymbol{H}$ nähern.

Implementieren Sie den Davidson Algorithmus um die Eigenvektoren zu den niedrigsten k Eigenwerten einer hermitschen Matrix $\boldsymbol{H}$ zu finden. Dabei diagonalisieren wir $\boldsymbol{H}$ in einem kleinen Unterraum, aufgespannt von bis zu $M \geq 2k$ Basisvektoren. Ausgehend von einem normierten Vektor $\boldsymbol{v}_1$ und seinem Rayleigh Quotienten $\lambda_1 = \boldsymbol{v}_1 \cdot (\boldsymbol{H}\boldsymbol{v}_1)$ als Anfangsschätzer eines Eigenwert und -vektor Paares, führt der Davidson Algorithmus folgende drei Schritte immer wieder aus:

1. Seien $(\lambda_1, \boldsymbol{v}_1), \ldots, (\lambda_n, \boldsymbol{v}_n)$ die bisherigen Näherungen von n Eigenwerten und deren Eigenvektoren. Wir erweitern den Raum der Eigenvektoren durch ihre Korrekturvektoren $\Delta\boldsymbol{v}_i(\lambda_i, \boldsymbol{v}_i)$ und berechnen davon eine Orthonormalbasis. Dazu nehmen wir die Matrix $\boldsymbol{U}$, bestehend aus den ersten $2n$ Spalten $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_{2n}$ der Matrix $\boldsymbol{Q}$ in der QR Zerlegung der Matrix

$$\boldsymbol{W} = \Big(\boldsymbol{v}_1\Big|\cdots\Big|\boldsymbol{v}_n\Big|\Delta\boldsymbol{v}_1\Big|\cdots\Big|\Delta\boldsymbol{v}_n\Big) = \boldsymbol{Q}\boldsymbol{R} \text{ mit } \boldsymbol{Q} \text{ unitär und } \boldsymbol{R} \text{ dreieckig.}$$

2. Wir projizieren $\boldsymbol{H}$ auf den Unterraum mit $\boldsymbol{J} = \boldsymbol{U}^*\boldsymbol{H}\boldsymbol{U}$ und finden die Eigenwerte und -vektoren $(\mu_1, \boldsymbol{u}_1), \ldots, (\mu_{2n}, \boldsymbol{u}_{2n})$ von $\boldsymbol{J}$.

3. Schließlich nehmen wir die Ritz-Vektoren $\boldsymbol{v}_i = \boldsymbol{U}\boldsymbol{u}_i$ und deren Ritz-Werte μ_i als neue Schätzer der Eigenvektoren $\boldsymbol{v}_i$ und der dazugehörigen Eigenwerte λ_i. Wir reduzieren die Anzahl der Paare wieder auf jene M Paare mit den niedrigesten Eigenwerten, falls $2n > M$.

Wenden Sie den Davidson Algorithmus an um den Grundzustand $|\psi_0\rangle$ und den ersten angeregten Zustand $|\psi_1\rangle$ des anharmonischen Oszillators $-\nabla^2/2 + x^4/24$ zu finden. Mit endlichen Differenzen in der diskreten Ortsbasis mit Punktgitterabstand Δx ist der Hamilton Operator gegeben durch:

$$\frac{-1}{2\Delta x^2}\begin{pmatrix} -2 & 1 & & & & 1 \\ 1 & -2 & 1 & & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -2 & 1 \\ 1 & & & & 1 & -2 \end{pmatrix} + \frac{\Delta x^4}{24}\begin{pmatrix} (-\frac{N}{2})^4 & & & & \\ & (-\frac{N}{2}+1)^4 & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & (\frac{N}{2}-1)^4 \end{pmatrix}$$

Finden Sie eine geeignete Länge $a = N\Delta x$ und stellen Sie beide Zustände in den einzelnen Schritte der Davidson Iteration bis zur Konvergenz dar. Zum Vergleich können Sie einen numerisch exakten Algorithmus verwenden.

Deliverables für `NMS2.zip`: `README.md, davidson.cxx`

function DAVIDSON($\boldsymbol{H}, \boldsymbol{v}_1, M$, iterations)
  $(\boldsymbol{D})_{ij} = \delta_{ij}(\boldsymbol{H})_{ii}$
  $\lambda_1 \leftarrow \boldsymbol{v}_1 \cdot (\boldsymbol{H}\boldsymbol{v}_1)$ ▷ Rayleigh coefficient as initial eigenvalue
  $\boldsymbol{V} \leftarrow \left(\boldsymbol{v}_1\right)$ ▷ $\boldsymbol{V}$ has single column $\boldsymbol{v}_1$
  iteration $\leftarrow 0$
  while iteration $<$ iterations **do**
    $(\lambda_i, \ldots, \lambda_n), \boldsymbol{V} = \left(\boldsymbol{v}_1 \middle| \cdots \middle| \boldsymbol{v}_n\right)$ ▷ current eigenvalues and vectors
    for all $i \in \{1, \ldots, n\}$ **do**
      $\Delta\boldsymbol{v}_i \leftarrow -(\boldsymbol{D} - \lambda_i \mathbf{1})^{-1}(\boldsymbol{H}\boldsymbol{v}_i - \lambda_i \boldsymbol{v}_i)$ ▷ relevant direction correcting $\boldsymbol{v}_i$
    end for
    $\boldsymbol{W} \leftarrow \left(\boldsymbol{v}_1 \middle| \cdots \middle| \boldsymbol{v}_n \middle| \Delta\boldsymbol{v}_1 \middle| \cdots \middle| \Delta\boldsymbol{v}_n\right)$ ▷ expand trial space
    $(\boldsymbol{Q}, \boldsymbol{R}) \leftarrow$ Householder-QR$(\boldsymbol{W})$ ▷ orthogonalize trial space
    $\boldsymbol{U} \leftarrow \boldsymbol{Q}\, \mathbf{1}^{N \times 2n}$ ▷ basis of column span of $\boldsymbol{W}$
    $\boldsymbol{J} \leftarrow \boldsymbol{U}^* \boldsymbol{H} \boldsymbol{U}$ ▷ project $\boldsymbol{H}$ onto basis
    $(\mu_1, \boldsymbol{u}_1), \ldots, (\mu_{2n}, \boldsymbol{u}_{2n}) \leftarrow$ Eigensystem$(\boldsymbol{J})$ ▷ ascending eigenvalues
    $m \leftarrow \min(2n, M)$ ▷ reduce trial space
    $(\lambda_1, \ldots, \lambda_m) \leftarrow (\mu_1, \ldots, \mu_m)$ ▷ take Ritz values
    $\boldsymbol{V} \leftarrow \left(\boldsymbol{U}\boldsymbol{u}_1 \middle| \cdots \middle| \boldsymbol{U}\boldsymbol{u}_m\right)$ ▷ & Ritz vectors
    iteration $\leftarrow$ iteration $+ 1$
  end while
  return $(\lambda_1, \ldots, \lambda_M), \boldsymbol{V}$
end function

```
size_t n = V.cols()
MatrixXd W(N,2*n);      // (N,2n) expanded trial space
for (size_t i = 0; i < n; ++i) {
  W.col(i) = V.col(i);
  W.col(n+i) = ...
}
HouseholderQR<MatrixXd> W_QR(W); // Householder solver for W
// (N,N) unitary whose first 2n columns form the base:
MatrixXd Q = W_QR.householderQ();
...
// (N,m) identity matrix
... MatrixXD::Identity(N,m);
...
MatrixXd J = ...;       // (2n,2n) projection of H
SelfAdjointEigenSolver<MatrixXd> J_eigensystem(J);
// Ritz values
... J_eigensystem.eigenvalues();
// .col(i) contains u_i
... J_eigensystem.eigenvectors();
```

Deliverables für `NMS2.zip`: `README.md, davidson.cxx`