Universität Tübingen
Mathematisches Institut
Prof. Dr. Christian Lubich

Tübingen, den 05.12.2022

4. Exercise Sheet to Numerical Methods for Quantum Dynamics

Programming Exercise 6: Implement the *Hermitian Lanczos Algorithm Without Reorthogonalization* (found in the book in Algorithm 2.5). To test your implementation, you can check the following identity

$$T_m = V_m^* A V_m.$$

Use these matrices to approximate

$$e^{-i\Delta t A} v \approx V_m e^{-i\Delta t T_m} e_1. \tag{1}$$

Hint: To compute the (small) $m \times m$ matrix exponential, you can diagonalize T_m with *np.linalg.eig*.

Programming Exercise 7: Apply your implementation of (1) to the matrix

$$A = \frac{\omega \Delta t}{2} \mathbf{tridiag}(-1, 2, -1) \in \mathbb{R}^{1000 \times 1000},$$

for $\Delta t = 1$ with $\omega = 8$ and $\omega = 16$. Use $v = \frac{1}{\sqrt{N}} \mathbf{1} = \frac{1}{\sqrt{N}} (1)_{n=1}^N$. Plot the norms of the errors in a reasonable way for $m = 1, ..., 40$. Further plot the a posteriori error bound approximated by the Simpson rule, in the book described in Equation (2.22).

Hint: To compute the error, you can use the implementation of *scipy.linalg.expm*, or diagonalize the large matrix as done in the previous exercise. Once you are confident in your implementation, you can use a reference solution.

Programming Exercise 8: We attempt to compute the matrix exponential (1) with the parameters $\Delta t = 10$ and $\omega = 100$ and $v = \frac{1}{\sqrt{N}} \mathbf{1} = \frac{1}{\sqrt{N}} (1)_{n=1}^N$ and A as before, such that the error is in the order of a given tolerance $\boldsymbol{tol}$. Since the convergence of the direct method is very slow, we extend the method by time-stepping, which uses

$$e^{-i\Delta t A} v = e^{-i\tau_K A} ... e^{-i\tau_1 A} e^{-i\tau_0 A} v,$$

with a (a priori unknown) sequence of time steps $\tau_0, ..., \tau_K$. To find an appropriate sequence, use the following adaptive method. Set $m = 10$, $\tau = \Delta t$ and $t = 0$. Repeat the following steps until $t = \Delta t$:

- Compute $\widetilde{v} = V_m e^{-i\Delta t T_m} e_1$ and the error estimate e with the previous exercise.
- If $\tau \boldsymbol{tol} < e$, discard the intermediate result $\widetilde{v}$, set $\tau = \tau/2$ and jump to the above step.
- Set $v = \widetilde{v}$, $t = t + \tau$.
- If $e < 0.001 \cdot \tau \boldsymbol{tol}$, set $\tau = 2 \cdot \tau$.
- If $t + \tau > \Delta t$, set $\tau = \Delta t - \tau$.

Use the method for several values of the tolerances (e.g. $\boldsymbol{tol}_j = 2^{-j}$ for $j = 0, ..., 20$) and plot the tolerances together with the error of the method.

Programming Exercise 9: Repeat the above algorithm, but instead of adaptively choosing τ, choose m adaptively in the same way (with m initially set to 4). Plot again the tolerances and errors accordingly.

Discussed in the exercise in S08, on wednesday the 14.12.2022, from 10.15–12.