Algorithm:

1. Embed the data set.
2. Pick a point \( x(t_0) \) somewhere in the middle of the trajectory (but closer to the beginning is good).
3. Find that point’s nearest neighbor. Call that point \( z_0(t_0) \).
4. Compute \( ||z_0(t_0) - x(t_0)|| = L_0 \).
5. Follow the “difference trajectory” — the dashed line — forwards in time, computing \( ||z_0(t_i) - x(t_i)|| = L_0(i) \) and incrementing \( i \), until \( L_0(i) > \epsilon \). Call that value \( L'_0 \) and that time \( t_1 \).
6. Find \( z_1(t_1) \), the “nearest neighbor” of \( x(t_1) \), and loop to step 4. Repeat the procedure to the end of the fiduciary trajectory \( (t = t_n) \), keeping track of the \( L_i \) and \( L'_i \).

Use this formula to compute \( \lambda_1 \), the biggest (positive) Lyapunov exponent:

\[
\lambda_1 \approx \frac{1}{N\Delta t} \sum_{i=1}^{M-1} \log_2 \frac{L'_i}{L_i}
\]
...where $M$ is the number of times you went through the loop above, and $N$ is the number of timesteps in the fiduciary trajectory. $N \Delta t = t_n - t_0$.

Schematic of the directional “nearest neighbor” computation:

Some hints:

- You’ll need to come up with some sort of metric to balance nearness (how close the point is to $x_i(t_j)$) and directionality (how close the point is to the vector from $x_i(t_j)$ to $z_i(t_j)$...the dashed line in the schematic above). That choice is up to you.

- Start with $\theta = \pi/9$ and increase if necessary.

- If your data set contains transients, that can mess up this algorithm

- Short fiduciary trajectories can also mess things up

- If your program can’t find a neighbor within $\epsilon$, it should stop and report that fact to the user (who should use that information to adjust $\epsilon$ for the next run).

References:


Please see the “Lyapunov exponents” section of the “Reading assignments for PS8-10 (Nonlinear Time Series Analysis)” handout.