emcee.autocorr.function_1d(chain_1d) goes quickly from 1 to about 0 but emcee.autocorr.integrated_time(sampler.get_chain() keeps growing. Even the quotient I get when dividing by the number of iteration steps is growing. Why is that?

When using `emcee.autocorr.function_1d(chain_1d)`, you are examining the *autocorrelation function* (ACF) for a one-dimensional chain. The ACF typically starts at 1 (by definition, full correlation at lag 0) and usually decays quickly toward 0, especially if your chain mixes well. This visual can give you the impression that correlations disappear rapidly.

In contrast, `emcee.autocorr.integrated_time(sampler.get_chain())` estimates the *integrated autocorrelation time* (IAT). The IAT is the sum (technically, one plus twice the sum) of the ACF over lag, up to where the ACF drops to zero (or is no longer significantly positive). This value determines the effective number of independent samples in your chain.

Why does your integrated time keep growing?

- **Noisy tail:** If your chain isn't long enough or is poorly mixed, the autocorrelation function can be very noisy for larger lags. While the *initial* lags decay quickly, random fluctuations in the tail (where ACF should be ≈0) can sum to a sizable value, especially as the IAT calculation considers many lags by default. This "noisy tail" artificially inflates the IAT.
- **Short chain:** If your chain is too short compared to the *real* autocorrelation length, `integrated_time` will be underestimated at first. As the chain grows, the sum over noise at larger lags can swamp your estimate, causing it to keep increasing with chain length—even if `function_1d` *looks* like it's dropping off quickly.
- **Default windowing:** emcee's IAT calculation uses a windowing procedure (e.g., 5 times a running IAT estimate) to decide where to cut off the sum. If this cutoff drifts with more samples, your estimated IAT can grow.
- **Chain not settled:** If burn-in isn't sufficient or the sampler is stuck in part of the parameter space, early `function_1d` may *look* OK, but `integrated_time` will continue to grow as the chain explores new regions.

Dividing by the number of steps

If the ratio `integrated_time / num_samples` grows, it means the estimated IAT grows *faster* than you add samples. This is a classic symptom that your chain is either underconverged, too short relative to the true autocorrelation, or the tail of the autocorrelation is too noisy.

Solutions and Checks

- **Plot longer chains:** Run much longer chains if feasible.
- **Bootstrapping:** Estimate the IAT over different segments to assess stability.
- **Plot tail of ACF:** Carefully examine the tail of your autocorrelation function after lag 20, 30, etc., to see if it's really zero or just fluctuating.
- **Try capping window:** Use a fixed, smaller window in the IAT calculation to avoid summing over noisy lags.
- **Check mixing:** Assess trace plots and compare walkers to ensure sampler is exploring parameter space correctly.

> **Summary:** Rapid initial decay of the ACF doesn’t guarantee a short integrated time unless the ACF is zero (not noisy) for all further lags. The IAT grows if tail noise accumulates, which is typical for insufficiently long or poorly mixed chains. This is an intrinsic limitation of the estimator in the presence of noise.

: emcee documentation on autocorrelation time
: Online discussions and emcee GitHub issues on IAT estimation