The Diffusion Problem: A To Do List

In the following, we briefly list some points that have not been addressed in the current implementation or documentation, and discuss possible future extensions of the theory and the computer program.

(a) Extensions of the theory.

From a mathematical point of view, the following points are direct extensions of the theory that would deserve further attention:

(i) The degeneracy conjecture. In the script on simple diffusion, Section (b), we noted that the symmetry properties of the eigenstates of the diffusion problem and the degeneracy of the spectrum are apparently tightly linked, but we could not provide a proof for these observations, nor give a heuristic interpretation of this behavior. Clearly, resolving this deficiency would be desirable.

(ii) Composite numbers p of sites. Much of the theory presented here is based on the observation that for prime numbers p, all conventional rules of arithmetics hold for the finite and discrete set of numbers $0, 1, 2, \ldots, p-1$, if addition and multiplication are performed modulo p. The techniques developed here will generally fail if the number of sites is composite, and a rather different classification scheme and decoupling method probably will have to be invoked.

As an intermediate case, many of the properties of primitives and eigenstates derived here supposedly will continue to be valid under the weaker condition that the atomic number k and the number of sites p are only relatively prime, i.e., k and p possess no common divisors, or $\gcd(k, p) = 1$. Under these circumstances, the rotational equivalence classes introduced in Section (b), script on diagonalization, will continue to contain p members that are represented by a unique primitive ζ with vanishing pattern sum r.

As an example of a property that does not extend to the case of composite p, even if k and p are relatively prime, consides the "power" operation on primitives [Section (a), script on simple diffusion]. The power $\mathcal{Z}^j$ of $\mathcal{Z}$ then by definition contains atoms at the sites $(jv)_{\mathrm{mod}\,p}$ if $\mathcal{Z}$ occupies the sites v. For prime p, the operation creates a permutation within the set of primitives, while for composite p, it will not conserve the number of atoms. Consider e.g. the case k=3, p=8. Then $\mathcal{Z} = \circ\bullet\circ\circ\circ\bullet\circ\bullet$ (positions 6, 2, 0) is a primitive, but $\mathcal{Z}^2$ would have atoms at the locations $0 = 2\cdot 0$, and $4 = 2\cdot 2 = (6\cdot 6)_{\mathrm{mod}\,8}$! Thus, $\mathcal{Z}^2$ is not defined within the set of 7 primitives with k=3, p=8.

(b) Implementation issues

This section identifies extensions of the DIFF program capabilities.

(i) Limits on pattern lengths. In the current implementation of the program, the atomic arrangements are stored in binary form in variables of type LONG (32 bit). This limits the maximum number of sites to $p \leq 31$. If bigger systems are to be considered, the size of the binary code storage variable has to be extended (e.g., to the INT64 type for $p \leq 63$).

(ii) Restriction to pure initial states. Currently, the program exclusively uses distinct atomic patterns (supplied by the user as a string of atoms and holes) as input data for the initial atomic configuration at time t=0. This means that the system is initially in a "pure state" as the initial average of one specific configuration is unity, and the average of all other patterns vanishes. More general "mixed" initial states are currently not supported, but could be incorporated with additional input facilities and a slightly modified temporal evolution algorithm.

(c) Numerical complexity

Naturally, the problem solving capacity of the DIFF program is limited by the resources of memory usage and execution

time. Their dependence on the size of the problem is assessed below:

(i) Memory consumption. The memory requirements of the program are largely controlled by the size of the unitary transform matrices $\mathcal{U}(q)$ in the various momentum subspaces [see Section (g), diagonalization script]. Because $\mathcal{U}(p-q)^* = \mathcal{U}(q)$, only $(p+1)/2$ of these matrices need to be evaluated and stored. Their dimension is $N_{prim} = \frac{1}{p}\binom{p}{k}$; despite being generally complex (with the exception of $\mathcal{U}(0)$), reflection symmetry allows to compress them into a format that only consumes one real number per entry [see Section (d), script on reflection properties]. In double accuracy, each number occupies 8 bytes, so that storage of the transformation matrices requires:

$$4\,(p+1)\,N_{prim}^2 \qquad (1)$$

bytes of memory. For a large example problem with $p=19$ and $k=7$, one has a set of $N_{prim} = 2652$ primitives, and the transformation matrix will occupy ~ 530 MBytes.

(ii) Execution time. Diagonalization of the decay coefficient matrices $\mathcal{R}(q)$ is also the dominant factor that controls the execution time of the program. Both the Householder tridiagonalization step, as well as QR diagonalization with calculation of eigenvectors grow in complexity with the cube of the dimension N_{prim}, and one can estimate that the diagonalization of a hermitian matrix with double accuracy requires about $25\,N_{prim}^3$ floating point operations, or in total

$$12.5\,(p+1)\,N_{prim}^3 \qquad (2)$$

operations. For the aforementioned problem, this amounts to some 4.7×10^{12} operations, or 13 hours of CPU time on a PC with 100 Mflops computing power.

One infers that both memory usage and execution time grow rapidly with increasing size of the problem.

A possible solution to this practical problem is to abandon the strategy to find a numerically exact solution of the master equation, and instead to concentrate on the long-lived decaying modes of the system that are characterized by eigenvalues of small modulus. While this ansatz clearly violates the conservation of probability and delivers solutions that are applicable only in the long-term limit, it has the great advantage of strongly reduced numerical complexity and memory consumption.

Let us outline a possible scheme. In the current implementation of DIFF, diagonalization involves a two-step algorithm, Householder tridiagonalization and subsequent iterative diagonalization by the QR method. Here, we chose the Householder formalism because of its superior numerical stability. However, alternative techniques are available. Here, the method of Lanczos (Stoer/Bulirsch, Chapter 6.5.3) is of particular interest, because it can take advantage of the fact that with large N_{prim}, the decay coefficient matrices $R(q)$ become rather sparse [see Section (c), script on diagonalization]. Furthermore, the Lanczos algorithm delivers a sequence of tri-diagonal matrices whose eigenvalues quickly converge against the extremal eigenvalues of the complete problem. It therefore offers inexpensive partial tridiagonalization. In the subsequent QR step, approximate eigenvalues and eigenvectors are generated which are used for a partial spectral decomposition of $R(q)$:

$$R(q) \sim \sum_{\nu=1}^{n} \tilde{\lambda}_\nu \, \vec{u}_\nu \otimes \vec{u}_\nu^{+} \qquad (3)$$

where n is the number of approximate eigenvalues $\tilde{\lambda}_\nu$ considered in the scheme, and $\vec{u}_\nu$ are the corresponding approximations to the eigenvectors of $R(q)$ determined in the reduced scheme.