Pseudocode for Krotov's Method

Michael Goerz
(Dated: October 10, 2019)

For reference, Algorithm 1 shows the complete pseudocode of an optimization with Krotov's method, as implemented in the `krotov` package (`https://github.com/qucontrol/krotov`).

Variables are color coded. Scalars are set in blue, e.g. $\epsilon_{ln}^{(0)}$. States (Hilbert space states or vectorized density matrices) are set in purple, e.g. ϕ_k^{init}. They may be annotated with light gray superscripts to indicate the iteration-index i of the control under which state was propagated, and with light gray time arguments. These annotations serve only to connect the variables to the equations of motion: $\phi_k^{(0)}(t_n)$ and $\phi_k^{(0)}(t_{n-1})$ are the same variable ϕ_k. Operators acting on states are set in green, e.g. μ_{lkn}. These may be implemented as a sparse matrix or implicitly as a function that returns the result of applying the operator to a state. Lastly, storage arrays are set in red, e.g. Φ_0. Each element of a storage array is a state.

The Python implementation groups several of the algorithm's input parameters by introducing a list of N "objectives". The objectives are indexed by k, and each objective contains the initial state ϕ_k^{init}, the Hamiltonian or Liouvillian H_k to be used by the propagator U and for the operators μ_{lkn}, and possibly a "target" to be taken into account by the function χ. In many applications, $H_k \equiv H$ is the same in all objectives, and $\mu_{lkn} \equiv \mu_l$ if H is linear in the controls in addition.

The CPU resources required for the optimization are dominated by the time propagation (calls to the function U in lines 7, 24 37). This is under the assumption that evaluating U dominates the application of the operator μ_{lkn} to the state $\phi_k^{(i)}(t_{n-1})$ and the evaluation of the inner product of two states, lines 31, 34. This condition is fulfilled for any non-trivial Hilbert space dimension.

Loops over the index k are parallelizable, in particular in a shared-memory (multi-threaded) parallelization environment like OpenMP. In a (multi-process) method-passing environment like MPI, some care must be taken to minimize communication overhead from passing large state vectors. For some (but not all) functionals, inter-process communication can be reduced to only the scalar values constituting the sum over k in lines 31, 34.

The memory requirements of the algorithm are dominated by the storage arrays Φ_0, Φ_1, and X. Each of these must store $N(N_T+1)$ full state vectors (a full time propagation for each of the N objectives). Each state vector is typically an array of double-precision complex numbers. For a Hilbert space dimension d, a state vector thus requires $16d$ bytes of memory, or $16d^2$ bytes for a density matrix. Under certain conditions, the use of Φ_0 and Φ_1 can be avoided: both are required only when the second order update is used ($\sigma(t) \neq 0$). When the first order update is sufficient, Φ_1 may overwrite Φ_0 so that the two collapse into a single forward-storage Φ. The states stored in Φ are only used for the inhomogeneity $\partial g_b / \partial \langle \phi_k |$ in Eq. (3), and no storage Φ of forward-propagated states at all is required if $g_b \equiv 0$. Thus, in most examples, only the storage X of the backward-propagated states remains. In principle, if the time propagation U is unitary (i.e., invertible), the states stored in X could be recovered by forward-propagation of $\{\chi_k^{(i-1)}(t=0)\}$, eliminating X at the (considerable) runtime cost of an additional time propagation.

Optimization Functional and Equations of Motion

$$J[\{|\phi_k^{(i)}(t)\rangle\}, \{\epsilon_l^{(i)}(t)\}] = J_T(\{|\phi_k^{(i)}(T)\rangle\}) + \sum_l \int_0^T g_a(\epsilon_l^{(i)}(t))\,\mathrm{d}t + \int_0^T g_b(\{\phi_k^{(i)}(t)\})\,\mathrm{d}t \tag{1}$$

$$\frac{\partial}{\partial t} \left|\phi_k^{(i)}(t)\right\rangle = -\frac{\mathrm{i}}{\hbar} \hat{H}^{(i)} \left|\phi_k^{(i)}(t)\right\rangle \tag{2}$$

$$\frac{\partial}{\partial t} \left|\chi_k^{(i-1)}(t)\right\rangle = -\frac{\mathrm{i}}{\hbar} \hat{H}^{\dagger\,(i-1)} \left|\chi_k^{(i-1)}(t)\right\rangle + \left.\frac{\partial g_b}{\partial \langle \phi_k|}\right|_{(i-1)} \tag{3}$$

$$\text{with} \qquad \left|\chi_k^{(i-1)}(T)\right\rangle = -\left.\frac{\partial J_T}{\partial \langle \phi_k(T)|}\right|_{(i-1)} \tag{4}$$

Algorithm 1 KROTOV'S METHOD FOR QUANTUM OPTIMAL CONTROL

Input:

1. list of guess control values $\{\epsilon_{ln}^{(0)}\}$ where $\epsilon_{ln}^{(0)}$ is the value of the l'th control field on the n'th interval of the propagation time grid ($t_0 = 0, \dots, t_{N_T} = T$), i.e., $\epsilon_{ln}^{(0)} \equiv \epsilon_l^{(0)}(\tilde{t}_n)$ with $\tilde{t}_n \equiv t_n + ((t_{n+1} - t_n)/2)$
2. list of update-shape values $\{S_{ln}\}$ with each $S_{ln} \in [0, 1]$
3. list of update step size values $\{\lambda_{a,l}\}$
4. list of N initial states $\{\phi_k^{\text{init}}\}$ at $t = t_0 = 0$
5. propagator function U that in "forward mode" receives a state $\phi_k(t_n)$ and a list of control values $\{\epsilon_{ln}\}$ and returns $\phi_k(t_{n+1})$ by solving the differential equation (2), respectively in "backward mode" (indicated as $U^\dagger$) receives a state $\chi_k(t_n)$ and returns $\chi_k(t_{n-1})$ by solving the differential equation (3)
6. list of operators $\mu_{lkn} = \frac{\partial H_k}{\partial \epsilon_{ln}}$, where H_k is the right-hand-side of the equation of motion of $\phi_k(t)$, up to a factor of $(-\mathrm{i}/\hbar)$, cf. Eq. (2);
7. function χ that receives a list of states $\{\phi_k(T)\}$ and returns a list of states $\{\chi_k(T)\}$ according to Eq. (4);
8. optionally, if a second order construction of the pulse update is necessary: function $\sigma(t)$.

Output: optimized control values $\{\epsilon_{ln}^{(\text{opt})}\}$, such that $J[\{\epsilon_{ln}^{(\text{opt})}\}] \leq J[\{\epsilon_{ln}^{(0)}\}]$, with J defined in Eq. (1).

procedure KROTOVOPTIMIZATION($\{\epsilon_{ln}^{(0)}\}$, $\{S_{ln}\}$, $\{\lambda_{a,l}\}$, $\{\phi_k^{\text{init}}\}$, U, $\{\mu_{lkn}\}$, χ, σ)

  $i \leftarrow 0$ ▷ iteration number

  allocate forward storage array $\Phi_0[1 \dots N, 0 \dots N_T]$

  for $k \leftarrow 1, \dots, N$ **do** ▷ initial forward-propagation

    $\Phi_0[k, 0] \leftarrow \phi_k^{(0)}(t_0) \leftarrow \phi_k^{\text{init}}$

    for $n \leftarrow 1, \dots, N_T$ **do**

      $\Phi_0[k, n] \leftarrow \phi_k^{(0)}(t_n) \leftarrow U(\phi_k^{(0)}(t_{n-1}), \{\epsilon_{ln}^{(0)}\})$ ▷ propagate and store

    end for

  end for

  while not converged **do** ▷ optimization loop

    $i \leftarrow i + 1$

    $\Phi_1, \{\epsilon_{ln}^{(i)}\} \leftarrow$ KROTOVITERATION($\Phi_0, \{\epsilon_{ln}^{(i-1)}\}, \dots$)

    $\Phi_0 \leftarrow \Phi_1$

  end while

  $\forall l, \forall n : \epsilon_{ln}^{(\text{opt})} \leftarrow \epsilon_{ln}^{(i)}$ ▷ final optimized controls

end procedure

procedure KROTOVITERATION(Φ_0, $\{\epsilon_{ln}^{(i-1)}\}$, $\{S_{ln}\}$, $\{\lambda_{a,l}\}$, $\{\phi_k^{\text{init}}\}$, U, $\{\mu_{lkn}\}$, χ, σ)

  $\forall k : \phi_k^{(i-1)}(T) \leftarrow \Phi_0[k, N_T]$

  $\{\chi_k^{(i-1)}(T)\} \leftarrow \chi(\{\phi_k^{(i-1)}(T)\})$ ▷ backward boundary condition

  allocate backward storage array $X[1 \dots N, 0 \dots N_T]$.

  for $k \leftarrow 1, \dots, N$ **do**

    $X[k, N_T] \leftarrow \chi_k^{(i-1)}(T)$

    for $n \leftarrow N_T, \dots, 1$ **do** ▷ backward-propagate and store

      $X[k, n-1] \leftarrow \chi_k^{(i-1)}(t_{n-1}) \leftarrow U^\dagger(\chi_k^{(i-1)}(t_n), \{\epsilon_{ln}^{(i-1)}\}, \Phi_0)$

    end for

  end for

  allocate forward storage array $\Phi_1[1 \dots N, 0 \dots N_T]$

  $\forall k : \Phi_1[k, 0] \leftarrow \phi_k^{(i)}(t_0) \leftarrow \phi_k^{\text{init}}$

  for $n \leftarrow 1, \dots, N_T$ **do** ▷ sequential update loop

    $\forall k : \chi_k^{(i-1)}(t_{n-1}) \leftarrow X[k, n-1]$

    $\forall l : \Delta\epsilon_{ln} \leftarrow \frac{S_{l,n-1}}{\lambda_{a,l}} \operatorname{Im} \sum_k \left\langle \chi_k^{(i-1)}(t_{n-1}) \middle| \mu_{lkn} \middle| \phi_k^{(i)}(t_{n-1}) \right\rangle$ ▷ first order

    if $\sigma(t) \neq 0$ **then** ▷ second order

      $\forall k : \Delta\phi_k^{(i)}(t_{n-1}) \leftarrow \phi_k^{(i)}(t_{n-1}) - \Phi_0[k, n-1]$

      $\forall l : \Delta\epsilon_{ln} \leftarrow \Delta\epsilon_{ln} + \frac{S_{l,n-1}}{\lambda_{a,l}} \operatorname{Im} \sum_k \frac{1}{2}\sigma(\tilde{t}_n) \left\langle \Delta\phi_k^{(i)}(t_{n-1}) \middle| \mu_{lkn} \middle| \phi_k^{(i)}(t_{n-1}) \right\rangle$

    end if

    $\forall l : \epsilon_{ln}^{(i)} \leftarrow \epsilon_{ln}^{(i-1)} + \Delta\epsilon_{ln}$ ▷ apply update

    $\forall k : \Phi_1[k, n] \leftarrow \phi_k^{(i)}(t_n) \leftarrow U(\phi_k^{(i)}(t_{n-1}), \{\epsilon_{ln}^{(i)}\})$ ▷ propagate and store

  end for

  if $\sigma(t) \neq 0$ **then**

    Update internal parameters of $\sigma(t)$ if necessary

  end if

end procedure

Notes:

- The index k numbers the independent states to be propagated, respectively the independent "objectives" (see text for details), l numbers the independent control fields, and n numbers the intervals on the time grid.
- The optimization loop may be stopped if the optimization functional or the change of functional falls below a pre-defined threshold, a maximum number of iterations is reached, or any other criterion.
- The braket notation in lines 31 indicates the (Hilbert-Schmidt) inner product of the state $\chi_k^{(i-1)}(t_n - 1)$ and the state resulting from applying μ_{lkn} to $\phi_k^{(i)}(t_{n-1})$. In Hilbert space, this is the standard braket. In Liouville space, it is $\operatorname{tr}\left(\chi_k{}^{\dagger}\ \mu_{lkn}[\phi_k]\right)$ with density matrices χ_k, ϕ_k and a super-operator μ_{lkn}.
- For numerical stability, the states $\chi_k^{(i-1)}(T)$ in line 19 may be normalized. This norm then has to taken into account in the pulse update, line 31.
- In line 24, the storage array Φ_0 is passed to $U^{\dagger}$ only to account for the inhomogeneity due to a possible state-dependent constraint, $\partial g_b / \partial \langle \phi_k |$ in Eq. (3). If $g_b \equiv 0$, the parameter can be omitted.