Using parallel CPUs to solve a control problem

January 2021

This project consists of parallelizing the control problem:

$$\min_{c\in L^2(0,1)} J(c) := \|y(T) - y_{target}\|^2 + \alpha \int_0^T c^2(t)\,dt,$$

where the equation connecting y and c is of Schrödinger type:

$$i\dot{y}(t) = (A + c(t)B)\,y(t),$$

with a fixed initial state y_0.

1 Parallelization strategy

We wish to parallelize in time the solution of this problem. To do this, we use the decompposition $[0,T] = \cup_{\ell=1}^N [T_{\ell-1}, T_\ell]$. To simplify, and without losing generality, we define $T_\ell = \ell T/N$. The method that we will follow, consists of iterating three sub-steps:

1. Defining the intermediate state at each time T_ℓ.
2. Solving the control sub-problems over $[T_{\ell-1}, T_\ell]$ in parallel.
3. Construction of a new control by simply concatenation of the partial controls obtained in the previous sub-step.

It remains to clarify the definition of intermediate states $\Lambda = (\lambda_\ell)_{\ell=0,\ldots,N}$. For an arbitrary control c, we set $\Lambda^c = (\lambda_\ell^c)_{\ell=0,\ldots,N}$ with

$$\lambda_\ell^c = (1-\gamma^\ell)y^c + \gamma^\ell p^c, \tag{1}$$

with the state y^c and the adjoint p^c associated with c and $\gamma_\ell = T_\ell/T$. Now suppose in step k, we have a control c^k. we simply set

$$\lambda_\ell^k = \lambda_\ell^{c^k}.$$

2 Theoretical study

To theoretically study the previous method, we introduce a new functional:

$$J_{\|}(c,\Lambda) := \sum_{\ell=1}^N \beta_\ell J_\ell(c),$$

with

$$\beta_\ell = \frac{T}{T_{\ell+1} - T_\ell}, \quad \alpha_\ell = \frac{\alpha}{\beta_\ell}$$

and

$$J_\ell(c_\ell) = \|y_\ell(T_{\ell+1}) - \lambda_{\ell+1}\|^2 + \alpha_\ell \int_{T_\ell}^{T_{\ell+1}} c_\ell^2(t)\,dt$$

where the equation connecting y_ℓ and c is the same as before, except that it is only defined $[T_\ell, T_{\ell+1}]$:

$$i\dot{y}_\ell(t) = (A + c_\ell(t)B)\,y_\ell(t),$$

with an initial state defined by $y_\ell(T_\ell) = \lambda_\ell$.

1. For an arbitrary fixed control c, prove that the intermediate states defined by (1) are the solutions of the problem:
$$\min_\Lambda J_{\|}(c, \Lambda).$$
2. In addition, demonstrate that
$$J_{\|}(c, \Lambda^c) = J(c).$$

Answers: First, we show that

$$J_{\|}(c, \Lambda) \geq J(c)$$

for any $\Lambda = (\lambda_1, \lambda_2, \ldots, \lambda_{N-1})$ with $\lambda_\ell \in \mathbb{C}^d$.

For this, we define $\bar{y}_\ell\colon [T_\ell, T] \to \mathbb{C}^d$ to be the solution of the equation

$$\begin{cases} i\,\dot{\bar{y}}_\ell(t) = (A + c(t)B)\,\bar{y}_\ell(t) \\ \bar{y}_\ell(t = T_\ell) = \lambda_\ell \end{cases}$$

In fact, $\bar{y}_\ell$ is the extension of y_ℓ to the whole interval $[T_\ell, T]$.
It is worth saying that $\bar{y}_0 = y$ and $\bar{y}_{N-1} = y_{N-1}$ (as in Figure 1).

Now, since the distance between $\bar{y}_\ell$ and $\bar{y}_{\ell+1}$ preserves along the interval $[T_{\ell+1}, T]$, we have

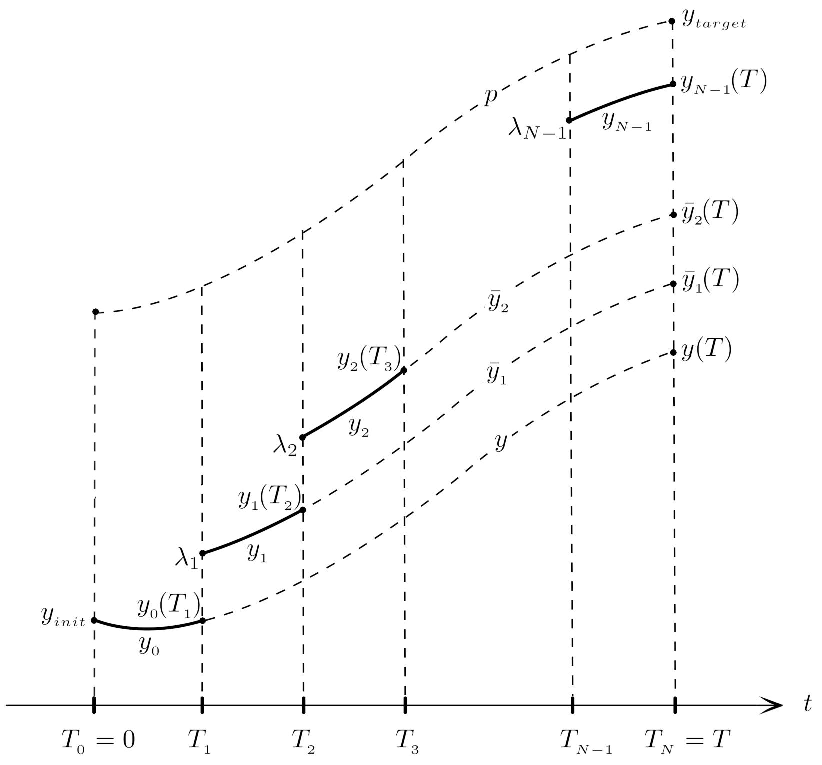

Figure 1

$$\|y_\ell(T_{\ell+1}) - \lambda_{\ell+1}\| = \|\bar{y}_\ell(T_{\ell+1}) - \bar{y}_{\ell+1}(T_{\ell+1})\| = \|\bar{y}_\ell(T) - \bar{y}_{\ell+1}(T)\| \tag{2}$$

The following calculation explains why $\|\bar{y}_\ell(t) - \bar{y}_{\ell+1}(t)\|$ remains invariant along $[T_{\ell+1}, T]$,

$$\begin{aligned} \frac{d}{dt}\|\bar{y}_\ell(t) - \bar{y}_{\ell+1}(t)\|^2 &= \frac{d}{dt}\left\langle \bar{y}_\ell(t) - \bar{y}_{\ell+1}(t)\,,\, \bar{y}_\ell(t) - \bar{y}_{\ell+1}(t)\right\rangle \\ &= \left\langle \dot{\bar{y}}_\ell(t) - \dot{\bar{y}}_{\ell+1}(t)\,,\, \bar{y}_\ell(t) - \bar{y}_{\ell+1}(t)\right\rangle + \left\langle \bar{y}_\ell(t) - \bar{y}_{\ell+1}(t)\,,\, \dot{\bar{y}}_\ell(t) - \dot{\bar{y}}_{\ell+1}(t)\right\rangle \end{aligned}$$

$$
\begin{aligned}
&= \big\langle -i\,(A+c(t)B)(\bar y_\ell(t)-\bar y_{\ell+1}(t))\,,\, \bar y_\ell(t)-\bar y_{\ell+1}(t)\big\rangle + \big\langle \bar y_\ell(t)-\bar y_{\ell+1}(t)\,,\, -i\,(A+c(t)B)(\bar y_\ell(t)-\bar y_{\ell+1}(t))\big\rangle \\
&= \big\langle -i\,(A+c(t)B)(\bar y_\ell(t)-\bar y_{\ell+1}(t))\,,\, \bar y_\ell(t)-\bar y_{\ell+1}(t)\big\rangle + \big\langle i\,(A+c(t)B)(\bar y_\ell(t)-\bar y_{\ell+1}(t))\,,\, \bar y_\ell(t)-\bar y_{\ell+1}(t)\big\rangle \\
&= 0
\end{aligned}
$$

Back to the problem, by the equation (2) we have

$$
\begin{aligned}
J_{\|}(c,\Lambda) = \sum_{\ell=0}^{N-1}\beta_\ell J_\ell(c_\ell) &= \sum_{\ell=0}^{N-1}\beta_\ell\|y_\ell(T_{\ell+1})-\lambda_{\ell+1}\|^2+\alpha\int_0^T c(t)^2\,dt \\
&= \sum_{\ell=0}^{N-1}\beta_\ell\|\bar y_\ell(T)-\bar y_{\ell+1}(T)\|^2+\alpha\int_0^T c(t)^2\,dt
\end{aligned}
$$

with the convention $\lambda_N := \bar y_N(T) := y_{target}$. Now, since $\|\cdot\|^2$ is a convex function on $\mathbb{C}^d$ and $\sum_{\ell=0}^{N-1}\frac{1}{\beta_\ell}=1$, one can continue the above equality and deduces that

$$
\begin{aligned}
J_{\|}(c,\Lambda) &= \sum_{\ell=0}^{N-1}\beta_\ell\|\bar y_\ell(T)-\bar y_{\ell+1}(T)\|^2+\alpha\int_0^T c(t)^2\,dt \\
&= \sum_{\ell=0}^{N-1}\frac{1}{\beta_\ell}\big\|\beta_\ell\big(\bar y_\ell(T)-\bar y_{\ell+1}(T)\big)\big\|^2+\alpha\int_0^T c(t)^2\,dt \\
&\geq \Big\|\sum_{\ell=0}^{N-1}\frac{1}{\beta_\ell}\,\beta_\ell\big(\bar y_\ell(T)-\bar y_{\ell+1}(T)\big)\Big\|^2+\alpha\int_0^T c(t)^2\,dt \\
&= \Big\|\sum_{\ell=0}^{N-1}\bar y_\ell(T)-\bar y_{\ell+1}(T)\Big\|^2+\alpha\int_0^T c(t)^2\,dt \\
&= \|\bar y_0(T)-\bar y_N(T)\|^2+\alpha\int_0^T c(t)^2\,dt \\
&= \|y(T)-y_{target}\|^2+\alpha\int_0^T c(t)^2\,dt \\
&= J(c)
\end{aligned}
$$

Now, we show that $J_{\|}(c,\Lambda)$ attains $J(c)$ for

$$
\lambda_\ell := (1-\gamma^\ell)\,y(T_\ell)+\gamma^\ell p(T_\ell) \tag{3}
$$

where $\gamma^\ell := \frac{T_\ell}{T}$ and $\ell = 0,1,\ldots,N-1$.

For this, we use the fact that $\|y_\ell(t)-(1-\gamma^{\ell+1})\,y(t)-\gamma^{\ell+1}p(t)\|$ is constant along the interval $[T_\ell,T_{\ell+1}]$ (implied with the same argument as above). Therefore one can write

$$
\begin{aligned}
\|y_\ell(T_{\ell+1})-\lambda_{\ell+1}\|^2 &= \|y_\ell(T_{\ell+1})-(1-\gamma^{\ell+1})\,y(T_{\ell+1})-\gamma^{\ell+1}p(T_{\ell+1})\|^2 \\
&= \|y_\ell(T_\ell)-(1-\gamma^{\ell+1})\,y(T_\ell)-\gamma^{\ell+1}p(T_\ell)\|^2 \\
&= \|\lambda_\ell-(1-\gamma^{\ell+1})\,y(T_\ell)-\gamma^{\ell+1}p(T_\ell)\|^2 \\
&= \|(1-\gamma^\ell)\,y(T_\ell)+\gamma^\ell p(T_\ell)-(1-\gamma^{\ell+1})\,y(T_\ell)-\gamma^{\ell+1}p(T_\ell)\|^2 \\
&= (\gamma^\ell-\gamma^{\ell+1})^2\,\|y(T_\ell)-p(T_\ell)\|^2 \\
&= (\gamma^\ell-\gamma^{\ell+1})^2\,\|y(T)-p(T)\|^2
\end{aligned} \tag{4}
$$

The last equality is due to the fact that the distance between $y(t)$ and $p(t)$ remains invariant along the whole interval $[0, T]$. Hence, by (4), for Λ given by (3), we have

$$\begin{aligned} J_{\|}(c, \Lambda) = \sum_{\ell=0}^{N-1} \beta_\ell J_\ell(c_\ell) &= \sum_{\ell=0}^{N-1} \beta_\ell \|y_\ell(T_{\ell+1}) - \lambda_{\ell+1}\|^2 + \alpha \int_0^T c(t)^2\, dt \\ &= \sum_{\ell=0}^{N-1} \beta_\ell (\gamma^\ell - \gamma^{\ell+1})^2 \, \|y(T) - p(T)\|^2 + \alpha \int_0^T c(t)^2\, dt \\ &= \sum_{\ell=0}^{N-1} \frac{1}{\beta_\ell} \, \|y(T) - p(T)\|^2 + \alpha \int_0^T c(t)^2\, dt \\ &= \|y(T) - p(T)\|^2 + \alpha \int_0^T c(t)^2\, dt \\ &= J(c) \end{aligned}$$

in which, as we know $p(T) = y_{target}$.

3 Practical Study

We now test the algorithm!

1. Implement the algorithm: we will simulate the parallelization, in the sense that we put in a "for" loop the second sub-step of the algorithm (relating to the sub-intervals). To carry out the second sub-step, one will use either a method of gradient, either a monotonic algorithm, or both (as desired).
2. Use the `cputime()` command to measure the elapsed time by parallelization. By *full efficiency* we mean that the elapsed time is exactly divided by the number of processors involved in parallelization.
3. Why can we say that the first sub-step does not fall within the scope of the parallelization, that is, it requires a solution over the entire interval $[0, T]$? Why does it necessarily prevent the achievement of full-efficiency?
4. To solve this problem, we can proceed as follows: on each sub-interval, the second sub-step makes it possible to calculate in parallel for each sub-interval $[T_{\ell-1}, T_\ell]$ the operator P_ℓ associated with the function $y_\ell(T_\ell) \mapsto y_\ell(T_{\ell+1})$. Explain why this calculation allows to get closer to full-efficiency.
5. Use the `cputime()` command to measure the elapsed time by this new idea of parallelization.

Answers: In practice, if we have $N + 1$ CPUs, for numerically solving the minimization problem

$$\min_c \|y(T) - y_{target}\|^2 + \alpha \int_0^T c(t)^2\, dt$$

it suffices to partition the whole interval into N smaller intervals, and let each CPU minimize the same problem

$$\min_{c_\ell} \|y_\ell(T_{\ell+1}) - \lambda_{\ell+1}\|^2 + \alpha_\ell \int_0^T c_\ell(t)^2\, dt \tag{5}$$

in parallel to the other CPUs. Yet, we need another CPU alongside these N CPUs, which must process the global operations that can not be done by other CPUs individually. More precisely, we can proceed as follow

1. *Passing a new control c to the main CPU,*

2. *The main CPU solves the equations*

$$\begin{cases} i\,\dot{y}(t) = (A + c(t)B)\,y(t) \\ y(t=0) = y_{init} \end{cases} \qquad \begin{cases} i\,\dot{p}(t) = (A + c(t)B)\,p(t) \\ p(t=T) = y_{target} \end{cases}$$

   by means of a discretization, say, $y_n = e^{i\Delta t A} e^{i\Delta t\, c_{n-1} B} y_{n-1}$ *and* $p_n = e^{-i\Delta t\, c_n B} e^{-i\Delta t A} p_{n+1}$*, and then calculates each intermediate state* λ_ℓ *provided by* (3)*, and passes these vectors to the parallel CPUs.*

3. *Each parallel CPU finds a better control* c'_ℓ *for the problem* (5) *using either* ***gradient descent*** *or* ***monotonic*** *method, and returns it to the main CPU again.*

In the third step, the main CPU concatenates these local controls to obtain a global one over the whole interval $[0, T]$ and passes it to the first step. This way, we let the algorithm iterate for several times. Now if we let c^k

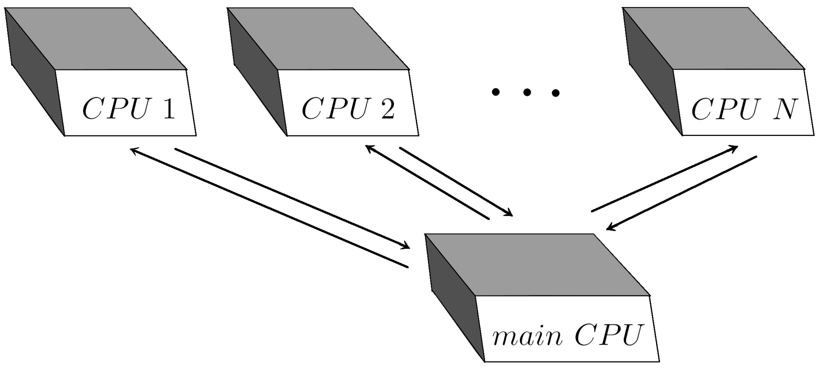

Figure 2: N CPUs working in parallel and interacting with a main CPU for solving a control problem.

to be the control obtained after k iterations of the algorithm, and Λ^k, the related intermediate states, then in the third step, each CPU ℓ gives an updated local control c_ℓ^{k+1} over $[T_{\ell-1}, T_\ell]$ which satisfies

$$J_\ell(c_\ell^{k+1}) \le J_\ell(c_\ell^k)$$

Multiplying each side of the above inequality by β_ℓ and summing, we get

$$J(c^{k+1}) = \min_{\Lambda} J_{||}(c^{k+1}, \Lambda) \le J_{||}(c^{k+1}, \Lambda^k) = \sum_{\ell=0}^{N-1} \beta_\ell J_\ell(c_\ell^{k+1}) \le \sum_{\ell=0}^{N-1} \beta_\ell J_\ell(c_\ell^k) = J_{||}(c^k, \Lambda^k) = J(c^k)$$

So, we can make sure that after each iteration the cost function J becomes smaller.

Apparently, the second step, does not lie in the scope of parallel computing, because it is implemented in the main CPU, and therefore it may be a matter of time-consuming and prevents approaching the full-efficiency. In order to overcome this problem, we should also let the parallel CPUs to intervene and do some part of the operation. More concretely, let's consider the example of 5 parallel CPUs and a main one, solving a control problem over the interval $[0, 10]$ such that each parallel CPU solving a local problem over an interval of length 2, discretized with 41 points. We know that in the k^{th} iteration, each parallel CPU $\ell = 1, 2, 3, 4, 5$ computes the new control c^{k+1} on the interval $[T_{\ell-1}, T_\ell]$ and passes it to the main CPU. Now that the main CPU has the whole c^{k+1}, it starts to compute the $y^{k+1}(T_1), \ldots, y^{k+1}(T_5)$ as well as $p^{k+1}(T_4), \ldots, p^{k+1}(T_0)$. In fact, it computes

$$\begin{aligned}
y^{k+1}(T_1) &= \left(e^{i\Delta t A} e^{i\Delta t\, c_{40}^{k+1} B}\right)\left(e^{i\Delta t A} e^{i\Delta t\, c_{39}^{k+1} B}\right)\cdots\left(e^{i\Delta t A} e^{i\Delta t\, c_{1}^{k+1} B}\right) y_{init} \\
y^{k+1}(T_2) &= \left(e^{i\Delta t A} e^{i\Delta t\, c_{80}^{k+1} B}\right)\left(e^{i\Delta t A} e^{i\Delta t\, c_{79}^{k+1} B}\right)\cdots\left(e^{i\Delta t A} e^{i\Delta t\, c_{41}^{k+1} B}\right) y^{k+1}(T_1) \\
&\vdots \\
y^{k+1}(T_5) &= \left(e^{i\Delta t A} e^{i\Delta t\, c_{200}^{k+1} B}\right)\left(e^{i\Delta t A} e^{i\Delta t\, c_{199}^{k+1} B}\right)\cdots\left(e^{i\Delta t A} e^{i\Delta t\, c_{161}^{k+1} B}\right) y^{k+1}(T_4)
\end{aligned}$$

As the above formulas suggest, we have

$$y^{k+1}(T_\ell) = M_\ell\, y^{k+1}(T_{\ell-1}) \qquad \ell = 1, \ldots, 5$$

where M_ℓ's are square matrices with the same dimension as A and B.

For the $p^{k+1}(T_\ell)$'s, we have the same calculation

$$\begin{aligned}
p^{k+1}(T_4) &= \left(e^{-i\Delta t\, c_{161}^{k+1} B} e^{-i\Delta t A}\right)\left(e^{-i\Delta t\, c_{162}^{k+1} B} e^{-i\Delta t A}\right)\cdots\left(e^{-i\Delta t\, c_{200}^{k+1} B} e^{-i\Delta t A}\right) y_{target} \\
p^{k+1}(T_3) &= \left(e^{-i\Delta t\, c_{121}^{k+1} B} e^{-i\Delta t A}\right)\left(e^{-i\Delta t\, c_{122}^{k+1} B} e^{-i\Delta t A}\right)\cdots\left(e^{-i\Delta t\, c_{160}^{k+1} B} e^{-i\Delta t A}\right) p^{k+1}(T_4) \\
&\vdots \\
p^{k+1}(T_0) &= \left(e^{-i\Delta t\, c_{1}^{k+1} B} e^{-i\Delta t A}\right)\left(e^{-i\Delta t\, c_{2}^{k+1} B} e^{-i\Delta t A}\right)\cdots\left(e^{-i\Delta t\, c_{40}^{k+1} B} e^{-i\Delta t A}\right) p^{k+1}(T_1)
\end{aligned}$$

therefore the same equations

$$p^{k+1}(T_\ell) = M'_\ell\, p^{k+1}(T_{\ell+1}) \qquad \ell = 0, \ldots, 4$$

But wait a minute! These 10 matrices can also be computed by the parallel CPUs, because any of these matrices only needs the partial information of c^{k+1} which is obtained during the parallel process. So, if the parallel CPUs also compute these matrices and return them to the main CPU, then the main CPU can compute the Λ^{k+1}'s faster and in a more efficient way.