# Custom Linear Algebra Solutions for PyTorch MPS on Apple Silicon (M2 Pro)

## Introduction

This program is a self-entertaining game of pushing the computation performance of Apple Silicon MPS to the limit for solving naive Linear Algebra problems. Inspired by the SCDAA coursework 2024, University of Edinburgh.

A SUPER fast solution for processing Monte Carlo simulations with a high computational cost is built, which you can find in the `MC_HYPERFAST.py`.

## Main Purpose

The main purpose of this program for the author is to make the most of his computation resources at hand (which though was swiftly overtaken by its successors 🤦‍♂️🤦‍♂️🤦‍♂️). The numerical example used to tap into his computer's computational power is drawn from coursework in Stochastic Control and Dynamic Asset Allocation at the University of Edinburgh, 2024.

The initial challenge arose when the author attempted to speed up a Monte Carlo simulation by shifting from the most basic `for` loop iteration to solving a large sparse linear equation system, effectively needing to compute the inverse of a massive matrix.  Coincidentally, he had just encountered some possibly unreliable claims in an advertisement from **Apple**, stating that the **Metal Performance Shader (MPS)** was particularly adept at handling such issues. However, after expending a great deal of effort to construct the matrix, his silicon cheerfully reported a memory overflow error as its way of saying thanks.

In the end, he had to tackle this problem by delving into the fundamental mathematics himself. Thankfully, there's still a shred of amusement to be found, as this ordeal somewhat resembles character leveling in an unseen video game.

## Mathematical Foundations

### 2-D Linear Quadratic Regulator

#### Dynamic

Consider the controlled process in space, expressed as

$$dX_s = [HX_s + M\alpha_s]ds + \sigma dW_s, s \in [t, T], X_t = x.$$

#### Objective

Our aimed is to minimize

$$J^{\alpha}(t, x) := \mathbb{E}^{t,x}\left[\int_t^T (X_s^\top C X_s + \alpha_s^\top D \alpha_s)ds + X_T^\top R X_T\right].$$

#### (Optimal) Value Function

The optimal of the problem above is called value function, denoted as

$$v(t,x) := \inf_{\alpha} J^{\alpha}(t,x).$$

By solving Bellman PDE, we can obtain that

$$v(t,x) = x^{\top} S(t) x + \int_t^T \mathrm{tr}(\sigma\sigma^{\top} S(r)) dr.$$

### Riccati ODE and its Solution

The function $S(t)$ in the expression of the value function above is the solution of Riccati ODE

$$\begin{aligned} \frac{dS(r)}{dr} &= -(S(r)H + H^{\top}S(r)) + S(r)MD^{-1}M^{\top}S(r) - C, r \in [t,T], \\ S(T) &= R. \end{aligned}$$

### Optimal Control

The corresponding optimal control is

$$a(t,x) = -D^{-1}M^{\top}S(t)x.$$

## Monte Carlo Verification

### Basic Illustration

We can bilaterally verify part of the code for Monte Carlo simulation and the part for Riccati equation solver.

Substituting the optimal control $a(t,x) = -D^{-1}M^{\top}S(t)x$ back to the dynamic of $X_s$, and choose Euler as our discretizing numerical scheme to operate Monte Carlo, we'll get the formulae for iteration as follows:

- **Explicit Scheme**

$$\begin{aligned} X^N_{t_{n+1}} &= X^N_{t_n} + \tau[HX^N_{t_n} - MD^{-1}M^{\top}S(t_n)X^N_{t_n}] + \sigma(W_{t_{n+1}} - W_{t_n}), \\ n &= k, \dots, N, \\ X^N_{t_k} &= x. \end{aligned}$$

- **Implicit Scheme**

$$\begin{aligned} X^N_{t_{n+1}} &= X^N_{t_n} + \tau[HX^N_{t_{n+1}} - MD^{-1}M^{\top}S(t_{n+1})X^N_{t_{n+1}}] + \sigma(W_{t_{n+1}} - W_{t_n}), \\ n &= k, \dots, N, \\ X^N_{t_k} &= x. \end{aligned}$$

### Iteration Equation System in Matrix Form

For the two schemes above, we can rewrite them in matrices.

Denote the coefficient matrices as follows:

$$
\begin{aligned}
I &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, & H &= \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix}, & M &= \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix}, \\
D^{-1} &= \begin{bmatrix} D_{11} & D_{12} \\ D_{21} & D_{22} \end{bmatrix}^{-1}, & S(\lambda) &= \begin{bmatrix} S_{11}(\lambda) & S_{12}(\lambda) \\ S_{21}(\lambda) & S_{22}(\lambda) \end{bmatrix}, & \sigma &= \begin{bmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{21} & \sigma_{22} \end{bmatrix}.
\end{aligned}
$$

- **Explicit Scheme**

$$
\begin{aligned}
\begin{bmatrix} x^N_{1,t_{n+1}} \\ x^N_{2,t_{n+1}} \end{bmatrix} &= \left[I + \tau[H - MD^{-1}M^\top S(t_n)]\right] \begin{bmatrix} x^N_{1,t_n} \\ x^N_{2,t_n} \end{bmatrix} + \sqrt{\tau}\sigma \begin{bmatrix} z^N_{1,t_n} \\ z^N_{2,t_n} \end{bmatrix}, \\
n &= k, \dots, N, \\
\begin{bmatrix} x^N_{1,t_k} \\ x^N_{2,t_k} \end{bmatrix} &= \begin{bmatrix} x_1 \\ x_2 \end{bmatrix},
\end{aligned}
$$

where

$$
z^N_{i,t_n} \sim \mathcal{N}(0,1), \quad \forall\, i \in \{1,2\}.
$$

Denote new coefficient matrices for combination:

$$
\begin{aligned}
A_{\mathrm{E},n} &= \begin{bmatrix} A_{\mathrm{E},n,11} & A_{\mathrm{E},n,12} \\ A_{\mathrm{E},n,21} & A_{\mathrm{E},n,22} \end{bmatrix} := \left[I + \tau[H - MD^{-1}M^\top S(t_n)]\right], && n = k, \dots, N, \\
B_{\mathrm{E},n} &= \begin{bmatrix} B_{\mathrm{E},n,1} \\ B_{\mathrm{E},n,2} \end{bmatrix} := \begin{cases} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, & n = k \\ \tau \begin{bmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{21} & \sigma_{22} \end{bmatrix} \begin{bmatrix} z^N_{1,t_n} \\ z^N_{2,t_n} \end{bmatrix}, & n = k+1, \dots, N \end{cases}.
\end{aligned}
$$

Let $k = 1$, then we can construct the whole matrix system as follows:

$$
\mathbb{A}_{\mathrm{E}}^{(2N \times 2N)} \mathbb{X}^{(2N \times 1)} = \mathbb{B}_{\mathrm{E}}^{(2N \times 1)},
$$

where

$$
\mathbb{A}_{\mathrm{E}}^{(2N\times 2N)} = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\
-A_{\mathrm{E},1,11} & -A_{\mathrm{E},1,12} & 1 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\
-A_{\mathrm{E},1,21} & -A_{\mathrm{E},1,22} & 0 & 1 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\
0 & 0 & -A_{\mathrm{E},2,11} & -A_{\mathrm{E},2,12} & 1 & 0 & \dots & 0 & 0 & 0 & 0 \\
0 & 0 & -A_{\mathrm{E},2,21} & -A_{\mathrm{E},2,22} & 0 & 1 & \dots & 0 & 0 & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & 0 & 0 & 0 & \dots & -A_{\mathrm{E},N-1,11} & -A_{\mathrm{E},N-1,12} & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \dots & -A_{\mathrm{E},N-1,21} & -A_{\mathrm{E},N-1,22} & 0 & 1
\end{bmatrix},
$$

$$
\mathbb{X}^{(2N\times 1)} = \begin{bmatrix}
x_{1,t_1}^N \\ x_{2,t_1}^N \\ x_{1,t_2}^N \\ x_{2,t_2}^N \\ x_{1,t_3}^N \\ x_{2,t_3}^N \\ \vdots \\ x_{1,t_N}^N \\ x_{2,t_N}^N
\end{bmatrix}, \qquad
\mathbb{B}_{\mathrm{E}}^{(2N\times 1)} = \begin{bmatrix}
B_{\mathrm{E},1,1} \\ B_{\mathrm{E},1,2} \\ B_{\mathrm{E},2,1} \\ B_{\mathrm{E},2,2} \\ B_{\mathrm{E},3,1} \\ B_{\mathrm{E},3,2} \\ \vdots \\ B_{\mathrm{E},N,1} \\ B_{\mathrm{E},N,2}
\end{bmatrix}.
$$

- **Implicit Scheme**

$$
\begin{aligned}
\left[I - \tau[H - MD^{-1}M^\top S(t_n)]\right] \begin{bmatrix} x_{1,t_{n+1}}^N \\ x_{2,t_{n+1}}^N \end{bmatrix} &= \begin{bmatrix} x_{1,t_n}^N \\ x_{2,t_n}^N \end{bmatrix} + \sqrt{\tau}\sigma \begin{bmatrix} z_{1,t_n}^N \\ z_{2,t_n}^N \end{bmatrix}, \\
n &= k, \dots, N, \\
\begin{bmatrix} x_{1,t_k}^N \\ x_{2,t_k}^N \end{bmatrix} &= \begin{bmatrix} x_1 \\ x_2 \end{bmatrix},
\end{aligned}
$$

where

$$
z_{i,t_n}^N \sim \mathcal{N}(0,1), \quad \forall\, i \in \{1,2\}.
$$

Denote new coefficient matrices for combination:

$$
A_{\mathrm{I},n} = \begin{bmatrix} A_{\mathrm{I},n,11} & A_{\mathrm{I},n,12} \\ A_{\mathrm{I},n,21} & A_{\mathrm{I},n,22} \end{bmatrix} := \left[I - \tau[H - MD^{-1}M^\top S(t_n)]\right], \qquad n = k, \dots, N,
$$

$$
B_{\mathrm{I},n} = \begin{bmatrix} B_{\mathrm{I},n,1} \\ B_{\mathrm{I},n,2} \end{bmatrix} := \begin{cases} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, & n = k \\ \tau \begin{bmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{21} & \sigma_{22} \end{bmatrix} \begin{bmatrix} z_{1,t_n}^N \\ z_{2,t_n}^N \end{bmatrix}, & n = k+1, \dots, N \end{cases}.
$$

Let $k = 1$, then we can construct the whole matrix system as follows:

$$\mathbb{A}_{\mathrm{I}}^{(2N\times 2N)}\mathbb{X}_{\mathrm{I}}^{(2N\times 1)}=\mathbb{B}_{\mathrm{I}}^{(2N\times 1)},$$

where

$$\mathbb{A}_{\mathrm{I}}^{(2N\times 2N)}=\begin{bmatrix}1&0&0&0&0&0&\dots&0&0&0&0\\0&1&0&0&0&0&\dots&0&0&0&0\\-1&0&A_{\mathrm{I},2,11}&A_{\mathrm{I},2,12}&0&0&\dots&0&0&0&0\\0&-1&A_{\mathrm{I},2,21}&A_{\mathrm{I},2,22}&0&0&\dots&0&0&0&0\\0&0&-1&0&A_{\mathrm{I},3,11}&A_{\mathrm{I},3,12}&\dots&0&0&0&0\\0&0&0&-1&A_{\mathrm{I},3,21}&A_{\mathrm{I},3,22}&\dots&0&0&0&0\\\vdots&\vdots&\vdots&\vdots&\vdots&\vdots&\vdots&\vdots&\vdots&\vdots&\vdots\\0&0&0&0&0&0&\dots&-1&0&A_{\mathrm{I},N,11}&A_{\mathrm{I},N,12}\\0&0&0&0&0&0&\dots&0&-1&A_{\mathrm{I},N,21}&A_{\mathrm{I},N,22}\end{bmatrix},$$

$$\mathbb{X}^{(2N\times 1)}=\begin{bmatrix}x_{1,t_1}^N\\x_{2,t_1}^N\\x_{1,t_2}^N\\x_{2,t_2}^N\\x_{1,t_3}^N\\x_{2,t_3}^N\\\vdots\\x_{1,t_N}^N\\x_{2,t_N}^N\end{bmatrix},\qquad\mathbb{B}_{\mathrm{I}}^{(2N\times 1)}=\begin{bmatrix}B_{\mathrm{I},1,1}\\B_{\mathrm{I},1,2}\\B_{\mathrm{I},2,1}\\B_{\mathrm{I},2,2}\\B_{\mathrm{I},3,1}\\B_{\mathrm{I},3,2}\\\vdots\\B_{\mathrm{I},N,1}\\B_{\mathrm{I},N,2}\end{bmatrix}.$$

# Block Matrix Inversion

## Basic Idea

For a full-rank matrix, if partitioned into four blocks, then it can be inverted blockwise, which will finally generate a recursion chain.

In general, matrix inversion by partition can be expressed as:

$$\begin{bmatrix}\Delta&\Lambda\\\Xi&\Sigma\end{bmatrix}^{-1}=\begin{bmatrix}\Delta^{-1}+\Delta^{-1}\Lambda(\Sigma-\Xi\Delta^{-1}\Lambda)^{-1}\Xi\Delta^{-1}&\Delta^{-1}\Lambda(\Sigma-\Xi\Delta^{-1}\Lambda)^{-1}\\-(\Sigma-\Xi\Delta^{-1}\Lambda)^{-1}\Xi\Delta^{-1}&(\Sigma-\Xi\Delta^{-1}\Lambda)^{-1}\end{bmatrix},$$

which means we can first compute $\Delta^{-1}$, then solve the sub-inversion $(\Sigma-\Xi\Delta^{-1}\Lambda)^{-1}$, where we can partition again and again. These steps will form a whole recursion.

## The Second Level for Partitioning the Computation Workload

An ideal way to partition the workload at a higher level is to consider the characteristics of the problem itself, and for each time take the upper left square part with the row number to the multiple of the dimension of $X_t$. Take the $\mathbb{A}_{\mathrm{I}}^{(2N\times 2N)}$ for instance. We can partition it as below,

$$\left[\begin{array}{cccc|cccccc} 1 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ -1 & 0 & A_{\mathrm{I},2,11} & A_{\mathrm{I},2,12} & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & -1 & A_{\mathrm{I},2,21} & A_{\mathrm{I},2,22} & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & -1 & 0 & A_{\mathrm{I},3,11} & A_{\mathrm{I},3,12} & \dots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & A_{\mathrm{I},3,21} & A_{\mathrm{I},3,22} & \dots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & -1 & 0 & A_{\mathrm{I},N,11} & A_{\mathrm{I},N,12} \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & -1 & A_{\mathrm{I},N,21} & A_{\mathrm{I},N,22} \end{array}\right].$$

Next, we solve the first upper-left square problem by solving its inversion by block inversion. Then, we obtain the first $(2 \times 1)$ section of $\mathbb{X}^{(2N\times 1)}$ by

$$\begin{bmatrix} x_{1,t_1}^N \\ x_{2,t_1}^N \\ x_{1,t_2}^N \\ x_{2,t_2}^N \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & A_{\mathrm{I},2,11} & A_{\mathrm{I},2,12} \\ 0 & -1 & A_{\mathrm{I},2,21} & A_{\mathrm{I},2,22} \end{bmatrix}^{-1} \begin{bmatrix} B_{\mathrm{I},1,1} \\ B_{\mathrm{I},1,2} \\ B_{\mathrm{I},2,1} \\ B_{\mathrm{I},2,2} \end{bmatrix}.$$

When dealing with the next block, to keep the correctness, we should change the first terms of the next section of $\mathbb{B}_{\mathrm{I}}^{(2N\times 1)}$ by **massaging a little bit** with the tail of the last solution piece of $\mathbb{X}^{(2N\times 1)}$.

It is probably noticed that the matrix problem with $2$ **time steps** as described essentially duplicates the behavior of the most elementary iteration, which tackles just $1$ **time step** at a time. However, the advantage of using matrix representation is that it allows us to effortlessly scale the number of time steps we wish to solve simultaneously to virtually any extent. This flexibility enables us to really push our computer to its limits, effectively making it sweat by working at its maximum capacity 😈.

## Additional Observation

By expanding the second dimension of $\mathbb{B}$ and thus $\mathbb{X}$, for example,

$$\mathbb{X}^{(2N\times\texttt{sample\_size})} = \begin{bmatrix}\begin{bmatrix} \\ X^{N,1} \\ \\ \end{bmatrix} & \begin{bmatrix} \\ X^{N,2} \\ \\ \end{bmatrix} & \cdots & \begin{bmatrix} \\ X^{N,\texttt{sample\_size}} \\ \\ \end{bmatrix}\end{bmatrix},$$

$$\mathbb{B}_{\mathrm{I}}^{(2N\times\texttt{sample\_size})} = \begin{bmatrix}\begin{bmatrix} \\ B_{\mathrm{I}}^{N,1} \\ \\ \end{bmatrix} & \begin{bmatrix} \\ B_{\mathrm{I}}^{N,2} \\ \\ \end{bmatrix} & \cdots & \begin{bmatrix} \\ B_{\mathrm{I}}^{N,\texttt{sample\_size}} \\ \\ \end{bmatrix}\end{bmatrix},$$

we can now scale the sampling batch size as well! 🥳