Parallel Implementation of Multivariate Empirical Mode Decomposition on GPU

June 2022

Zeyu Wang， Zoltan Juhasz

University of Pannonia
Veszprem, Hungary

Content outline

1. Background

1.1 Empirical Mode Decomposition
1.2 Features of EMD and its variants
1.3 Processing pipeline of MEMD

2. Parallel implementation of MEMD

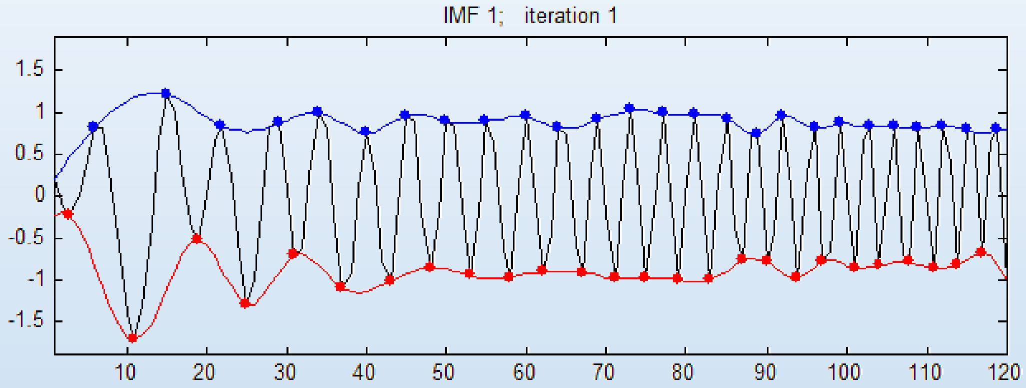
2.1 Numerical steps
2.2 Implementation details
2.3 Data layout in memory

3. Performance analysis

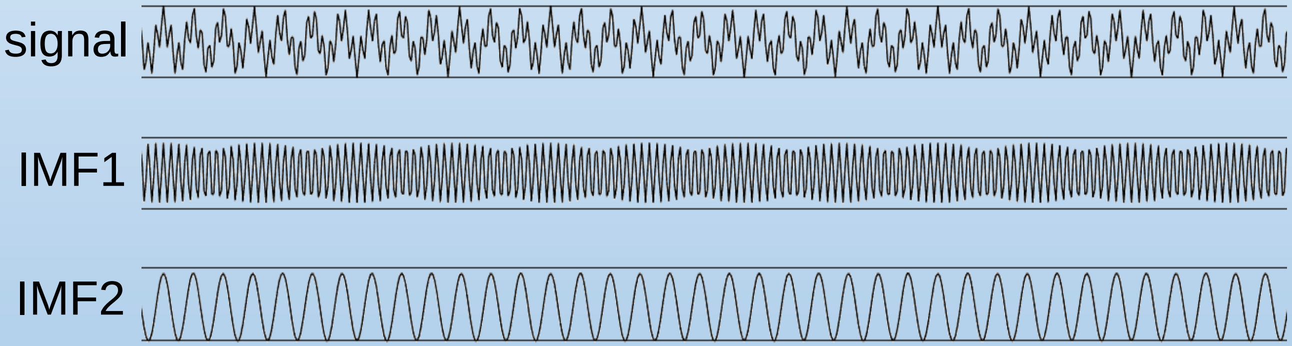
3.1 Performance overview
3.2 Kernel performance comparison

4. Future works

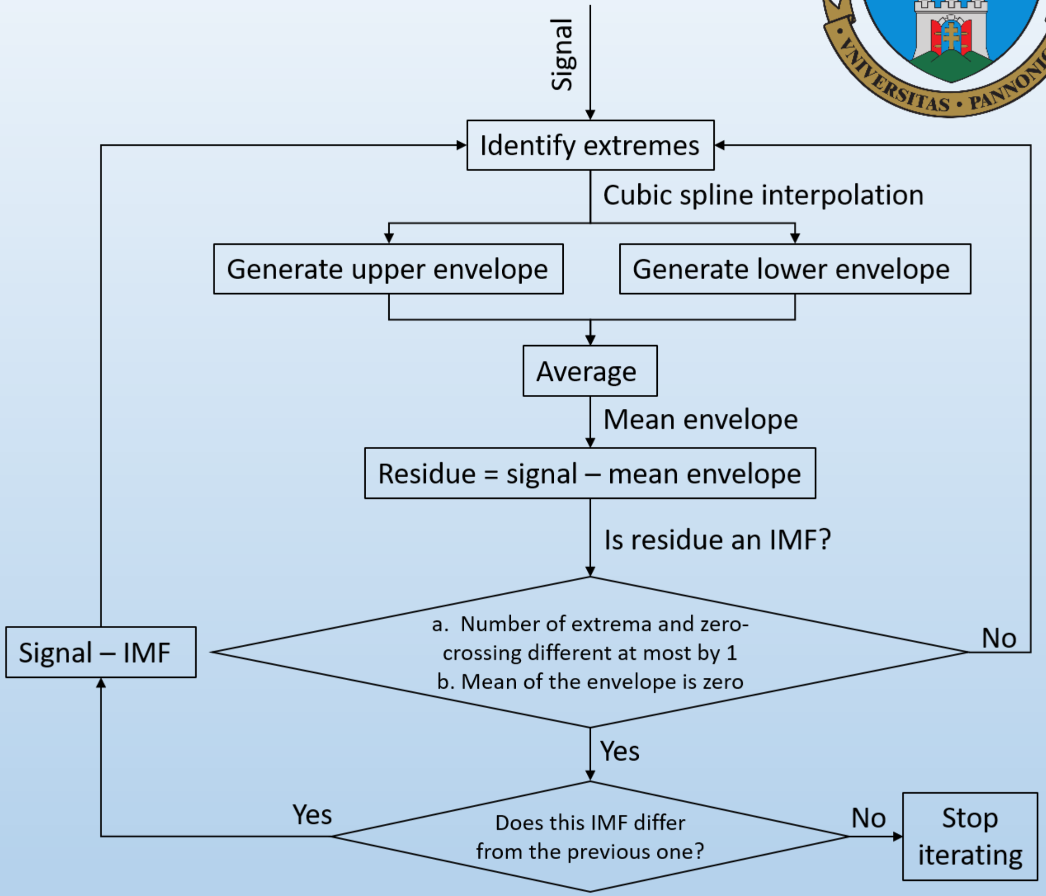
1.1 Empirical Mode Decomposition



EMD process



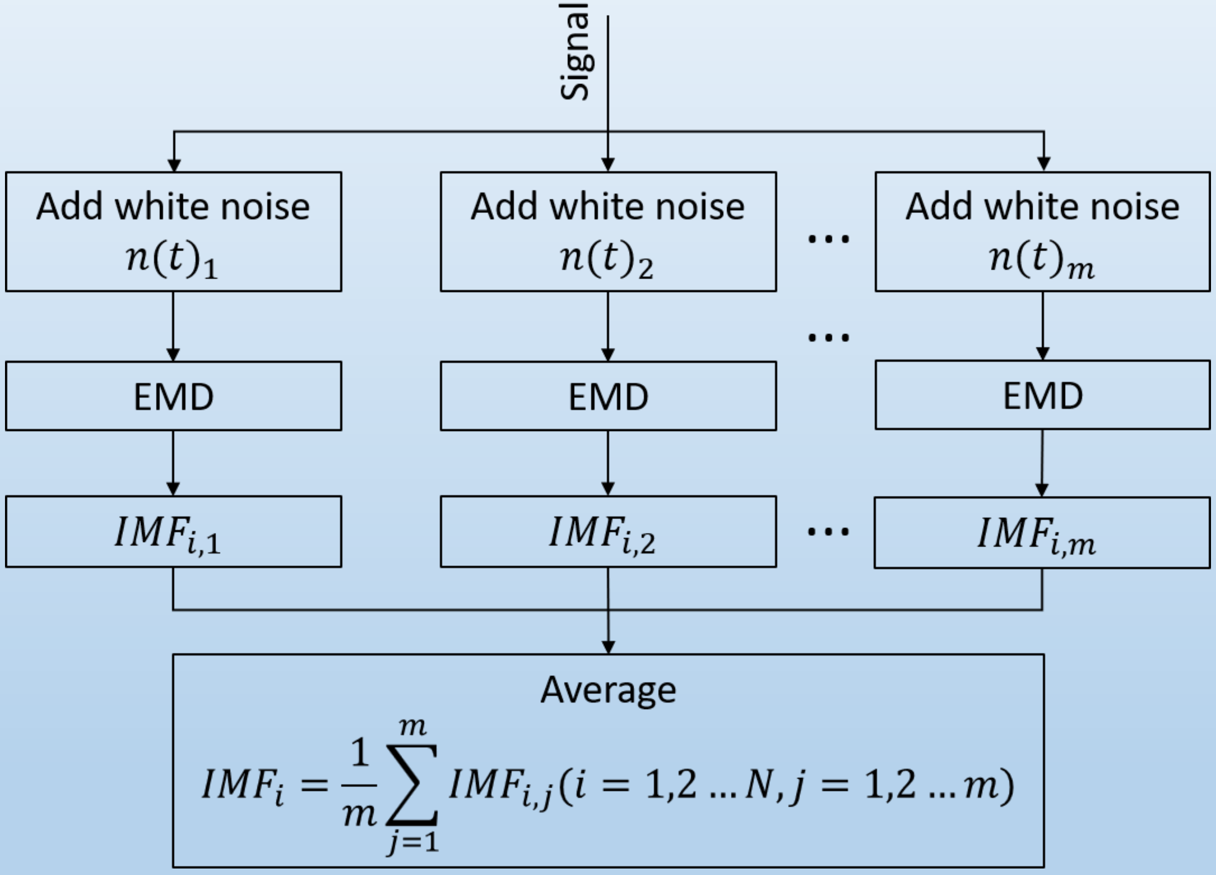
Separation of different oscillation modes



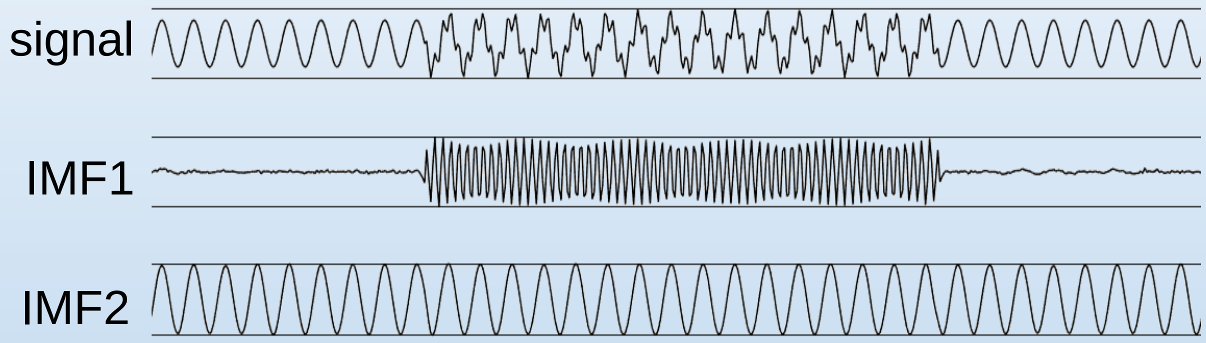
EMD process flowchart

N. E. Huang et al., "The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis," Proc. R. Soc. A Math. Phys. Eng. Sci., vol. 454, no. 1971, pp. 903–995, 1998

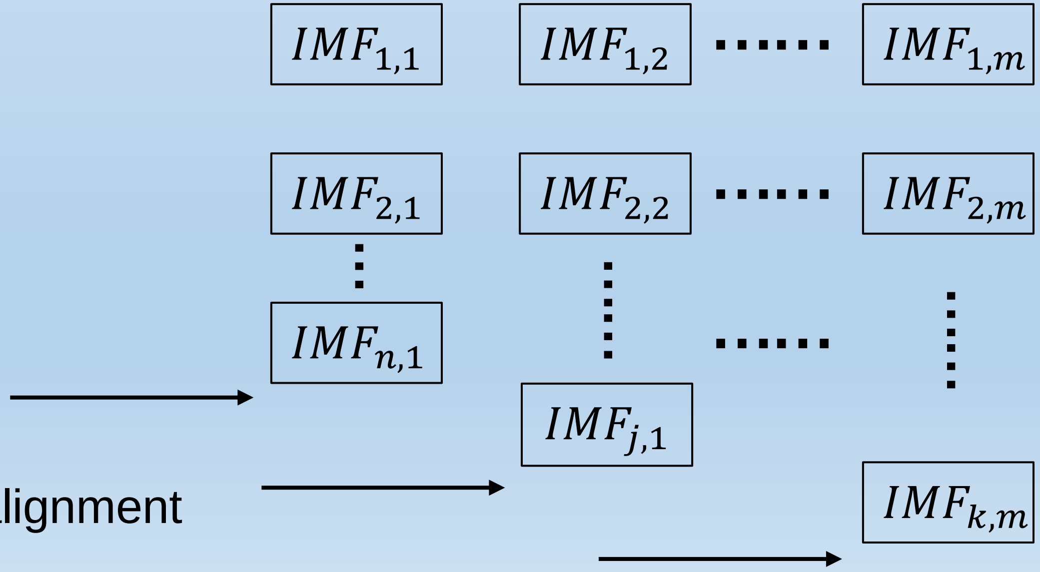
1.2 Features of EMD and its variants



EEMD process

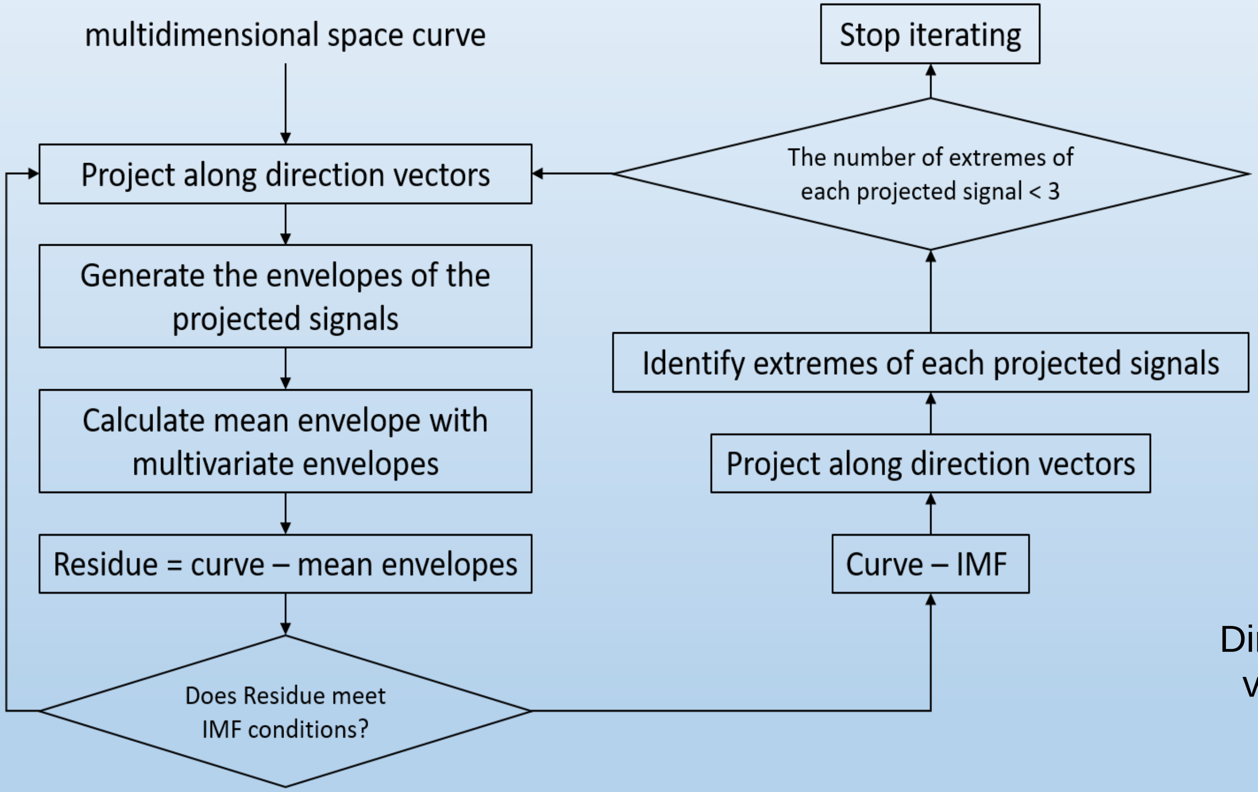


Handle of intermittent noise

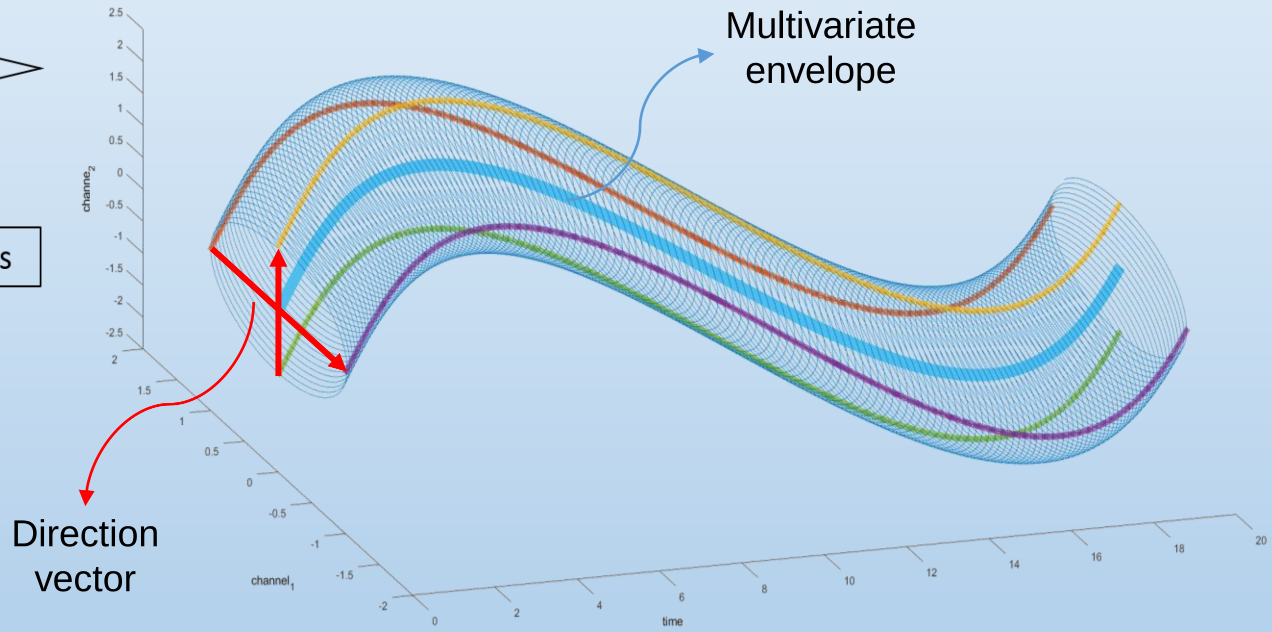


IMFs alignment

1.3 Processing pipeline of MEMD



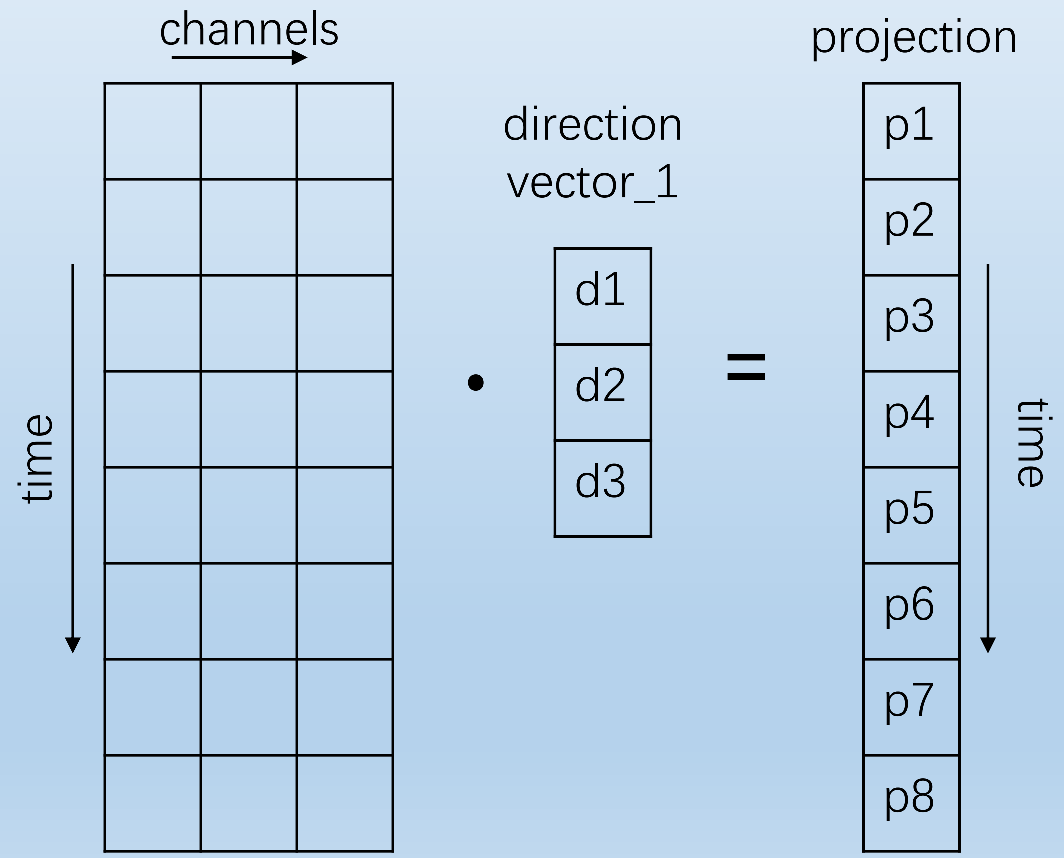
MEMD process flowchart



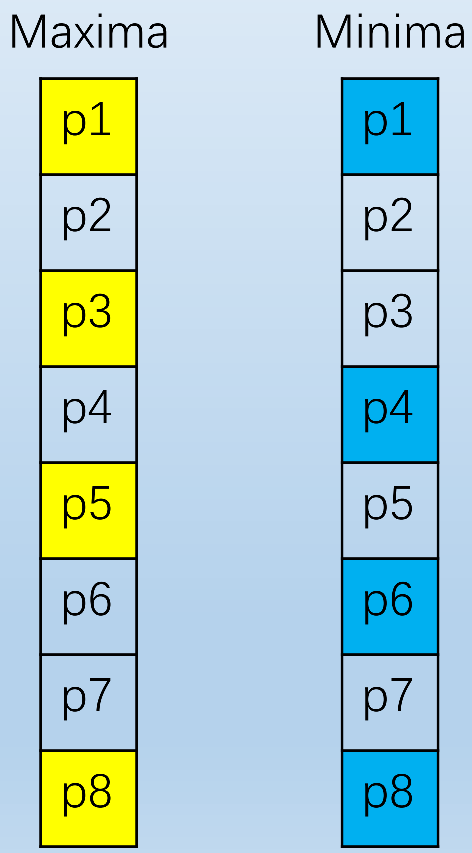
Envelopes of projected signals and the multivariate envelope

2.1 Numerical steps of MEMD

1. Use the data matrix to dot product the direct vector:

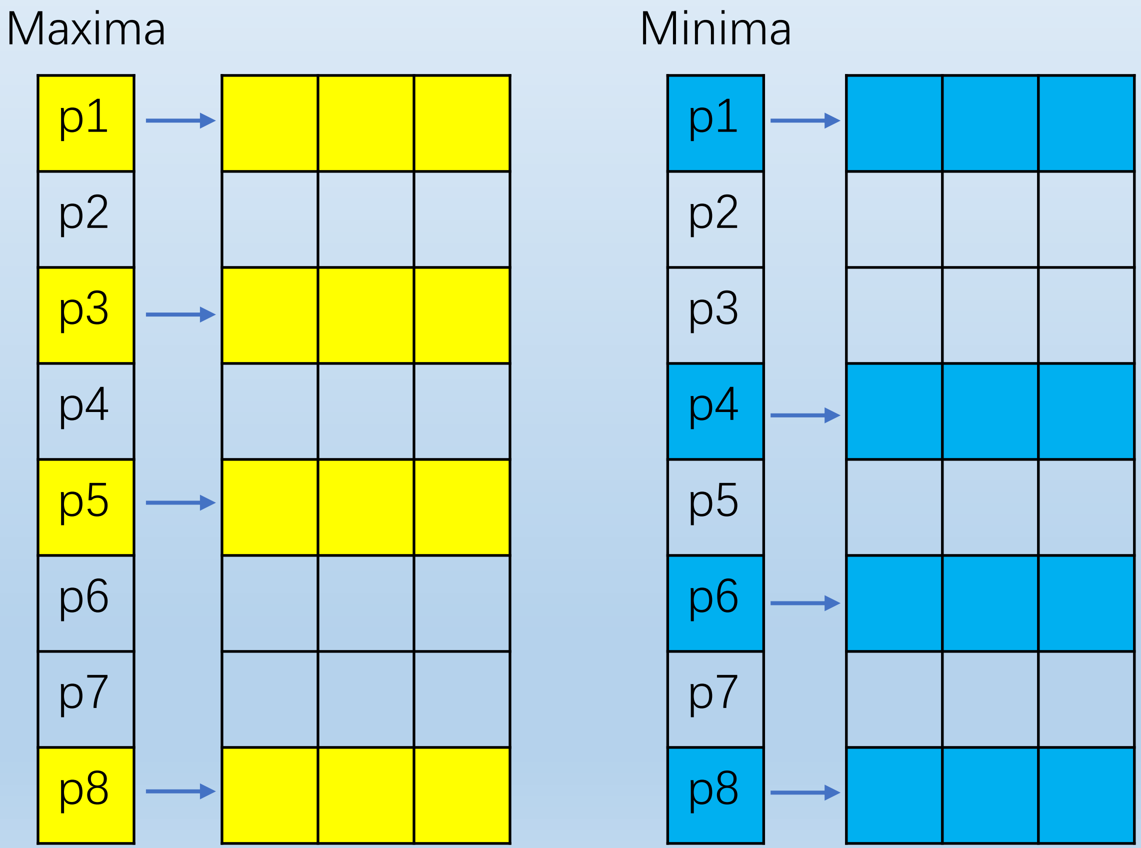


2. Detect the extreme points on the projection signal

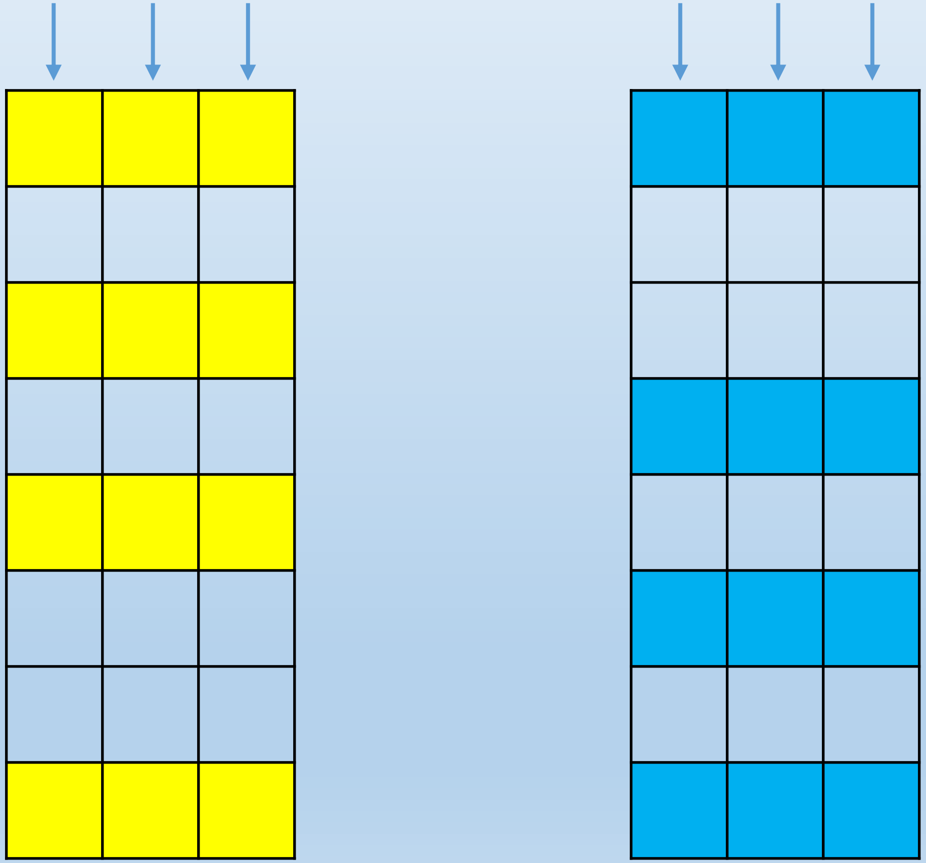


2.1 Numerical steps of MEMD

3. Find the corresponding multivariate extrema



4. Interpolate on the dimensions of multivariate extrema



(Here we need to perform six interpolation operations)

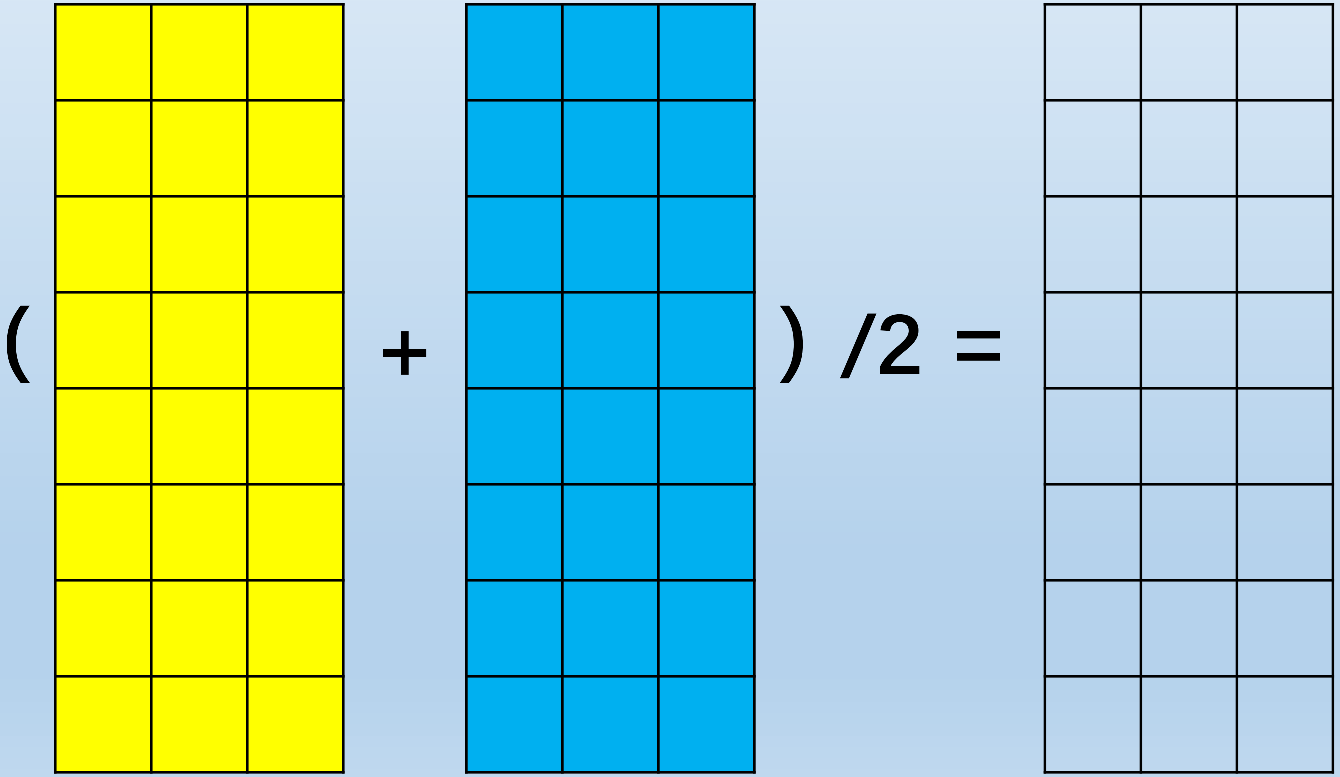
2.1 Numerical steps of MEMD

5. Calculate multivariate mean envelope

Upper envelope

Lower envelop

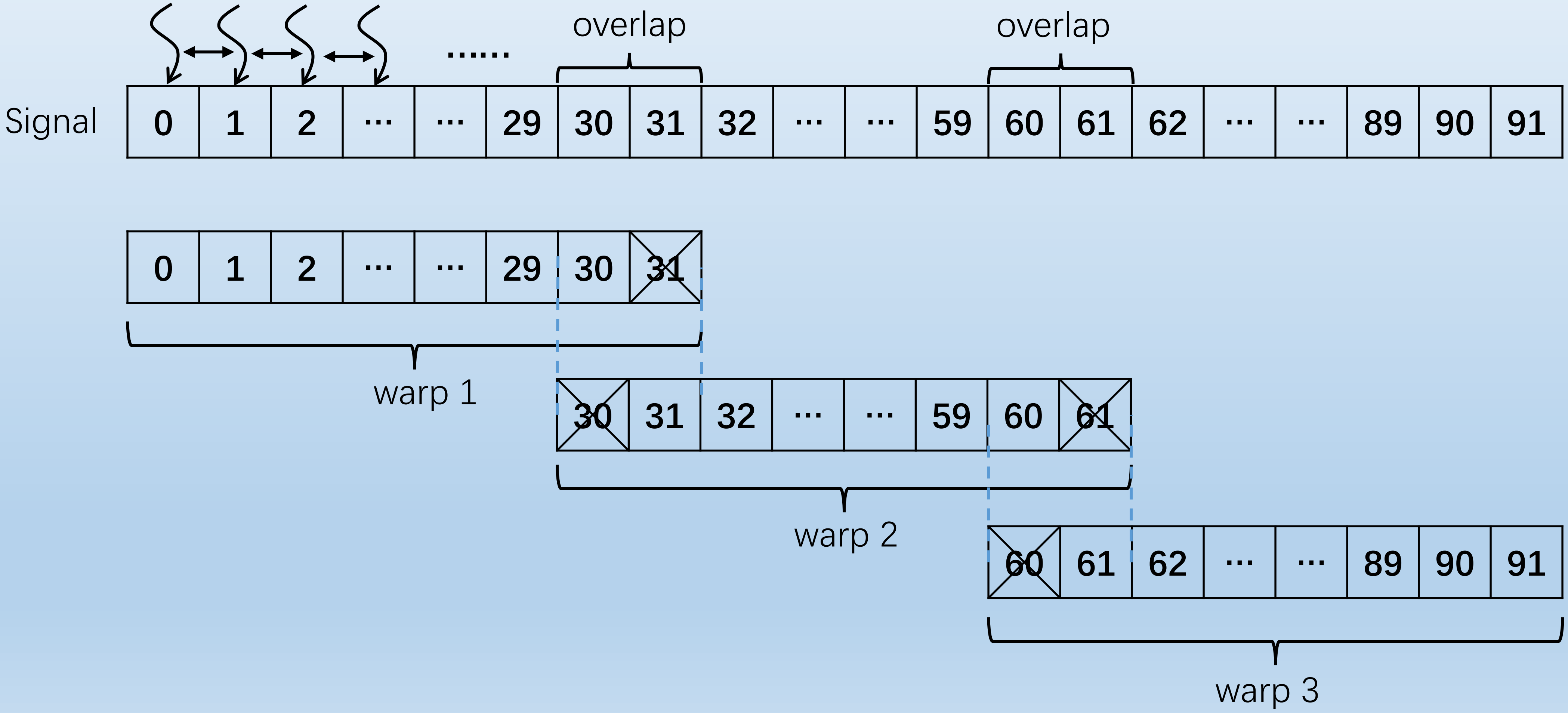
Mean envelope from direction vector_1



6. Calculates the mean envelope from the mean envelopes of all direction vectors.

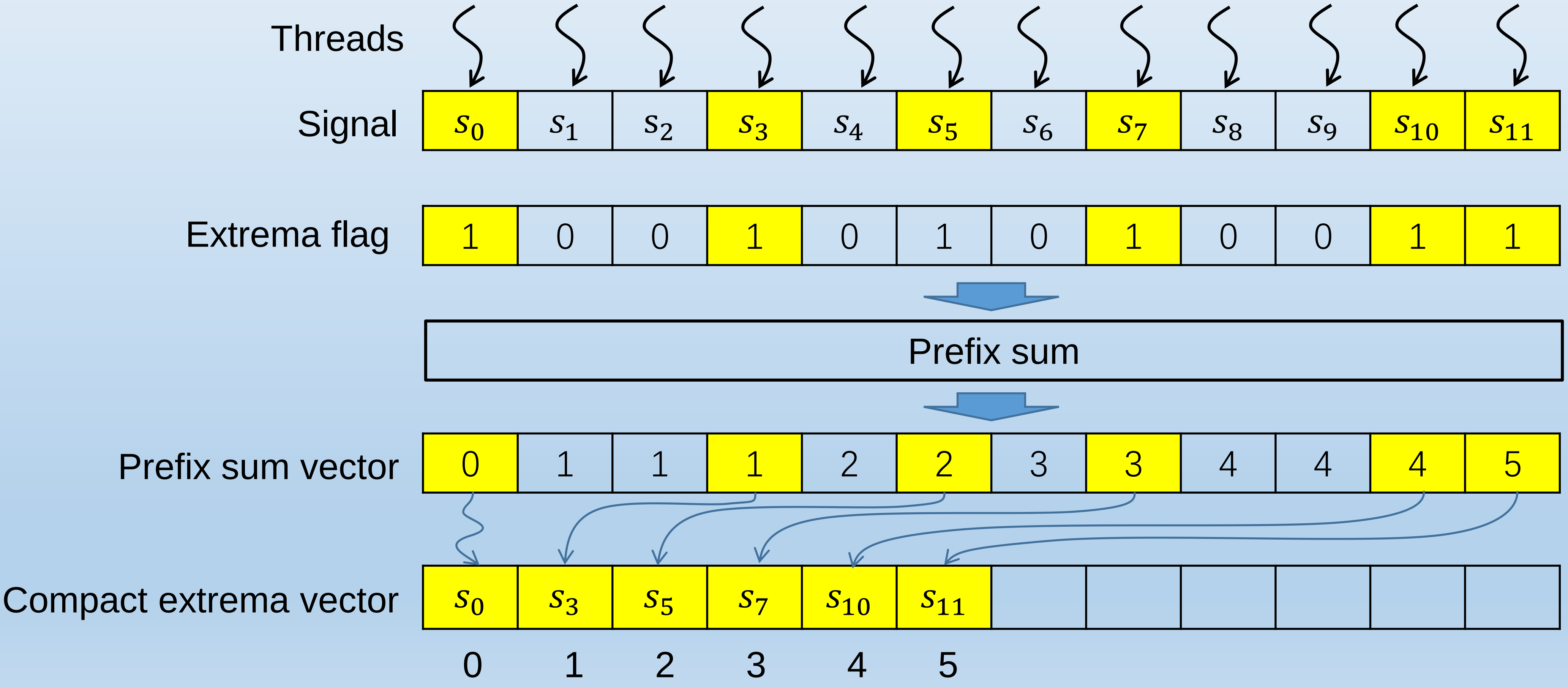
2.2 Implementation details

Use CUDA shuffle operation to detect extrema



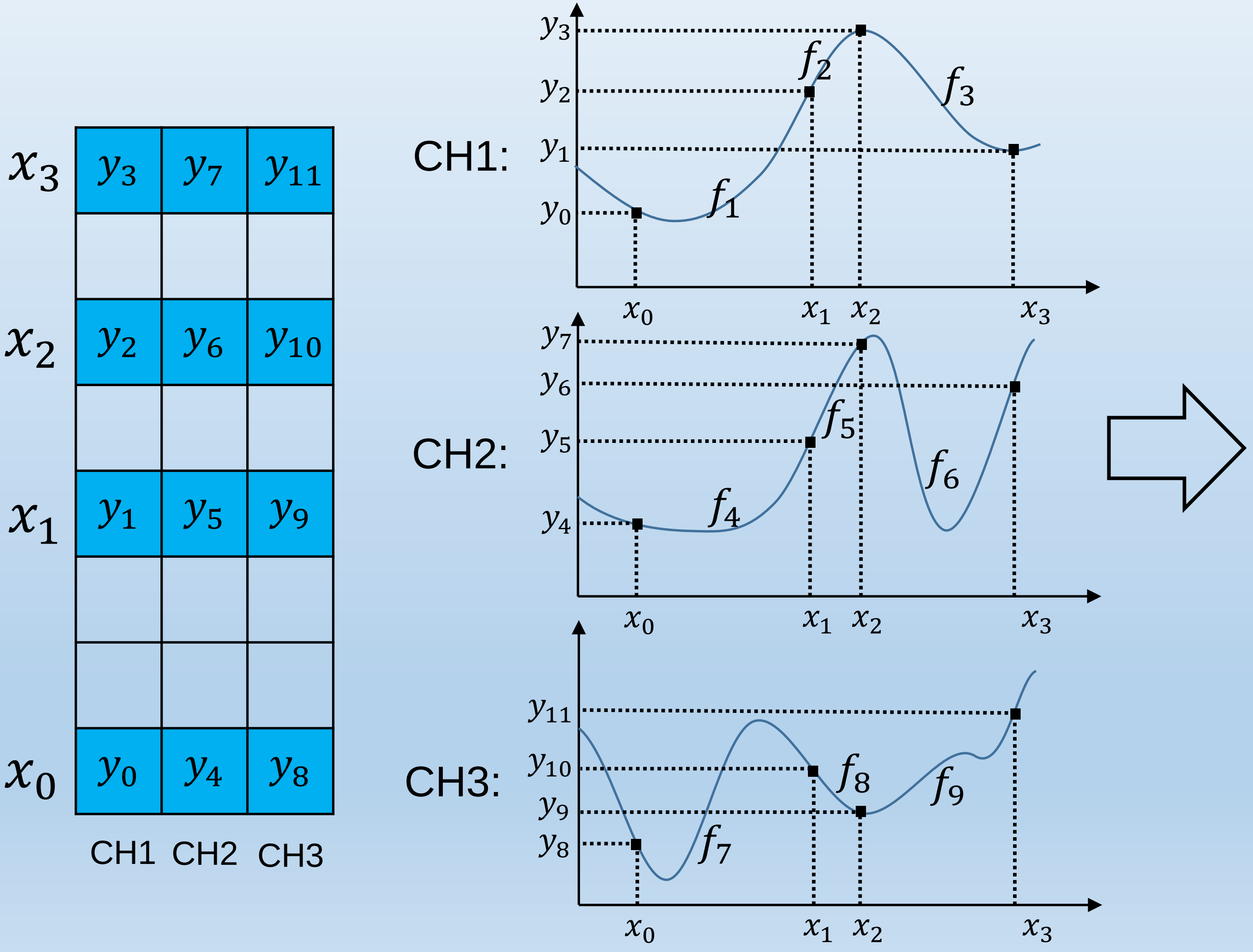
2.2 Implementation details

Use prefix sum to get compact extrema vector



2.2 Implementation details

Multi tridiagonal systems solver in interpolation



$$\begin{bmatrix} b_1 & c_1 & & & 0 \\ a_2 & b_2 & c_2 & & \\ & a_3 & b_3 & \ddots & \\ & & \ddots & \ddots & c_{n-1} \\ 0 & & & a_n & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} =$$

Coefficient matrix

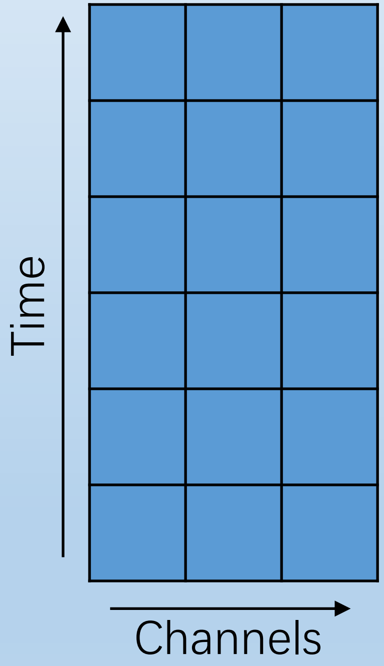
$$\begin{bmatrix} d_{11} & d_{21} & d_{31} \\ d_{12} & d_{22} & d_{32} \\ d_{13} & d_{23} & d_{33} \\ \vdots & \vdots & \vdots \\ d_{1n} & d_{2n} & d_{3n} \end{bmatrix}$$

Right hand matrix

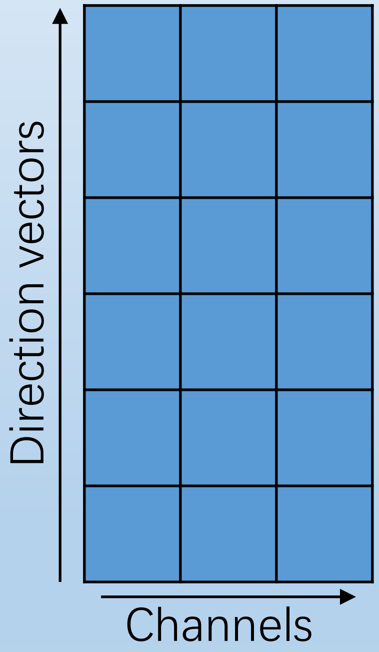
CH1 CH2 CH3

2.3 Data layout in memory

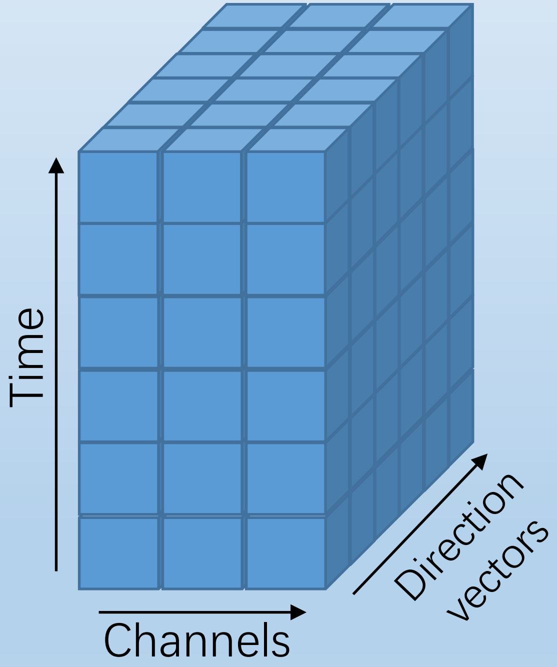
To store raw signal and multivariate IMF



To store direction vectors

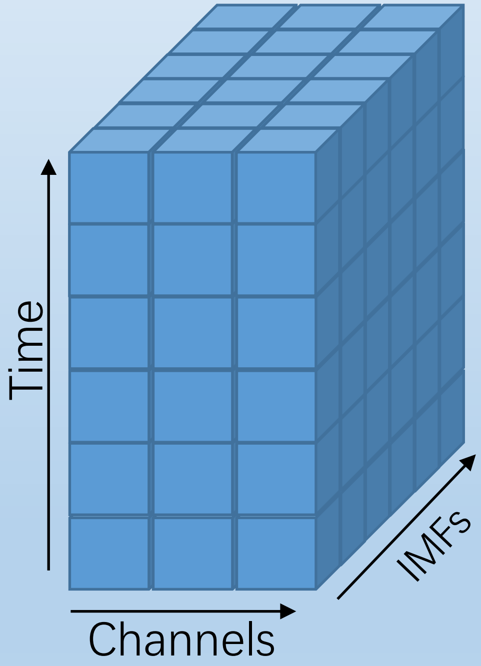


To store projected signals



X2

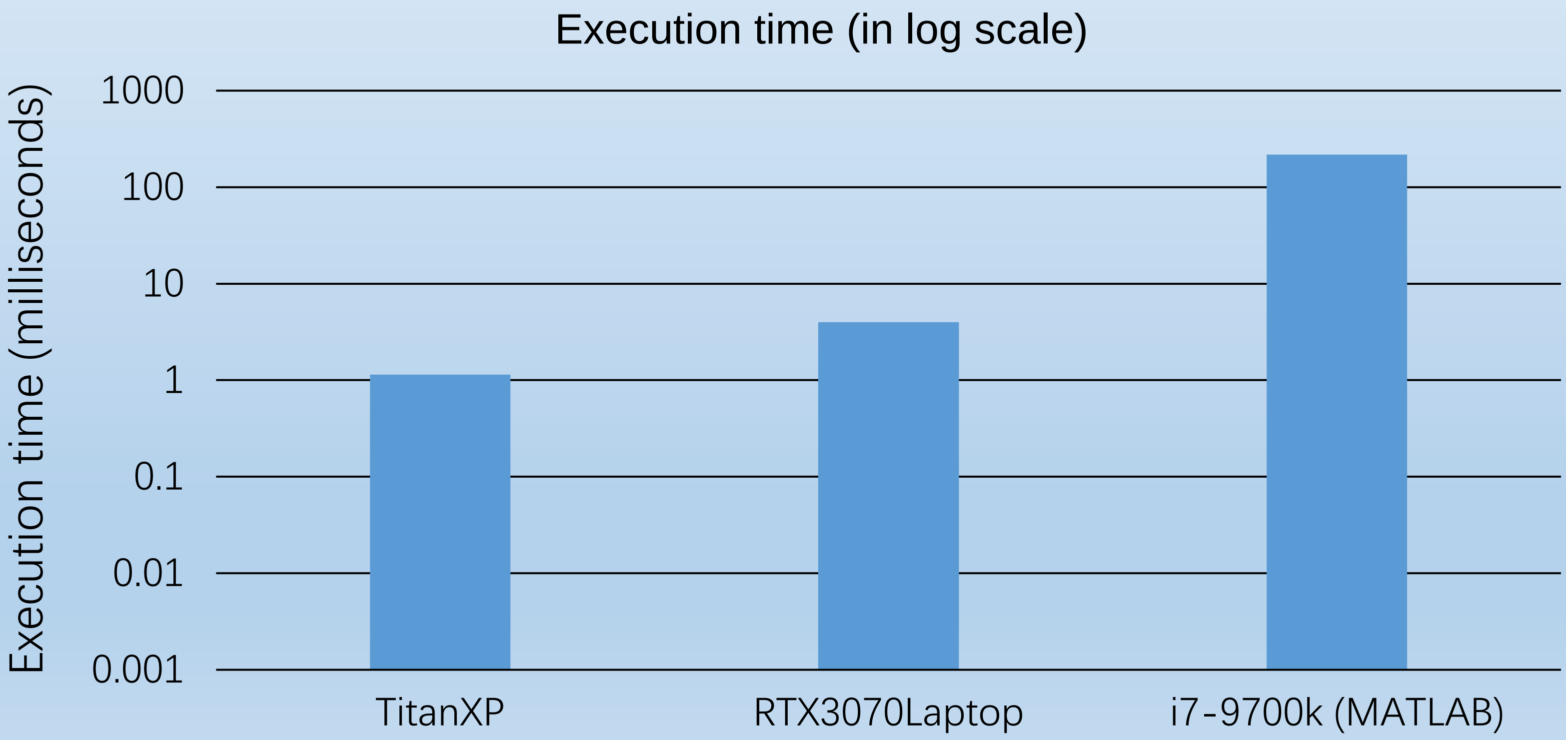
To store IMFs result



3.1 Performance overview

Dataset: EEGLAB sample dataset
Number of channels: 4
Number of direction vectors: 64
Length of signal: 30504

Number of IMFs: 8
Number of iteration: 10



Compared to CPU,
Titan achieved **190X**
RTX3070 achieved **55X**
speedup

3.2 Kernel performance compared to literature

Number of channels: 16
Number of direction vectors: 64
Length of signal: 1001

Number of IMFs: 8
Number of iteration: 10

		Execution time (µs)		
Kernels	Number of calls	Literature [1]	Our version	Speedup
HammersleySeqGen	1	11	3.42	**3x**
DirectionVecGen	1	148	64.64	**2x**
Projections	7	802	10.21	**79x**
PeaksDetection	70	1232	6.61	**187x**
BoundaryCondSet	140	3961	84.73	**47x**
EnvelopeMean	140	152	28.34	**5x**

[1] Mujahid, T., Rahman, A. U., & Khan, M. M. (2017). GPU-Accelerated Multivariate Empirical Mode Decomposition for Massive Neural Data Processing. *IEEE Access*, *5*, 8691–8701.

4 Future works

1. There are still limitations in our performance tests, and in the future, we will test more datasets including 128-channel EEG signals under different execution parameters.
2. The effects of some detailed parameter settings on the decomposition results still need to be further studied, such as the settings of extrema and tridiagonal matrix boundary conditions, and the setting of sifting stop criterion.
3. Some numerical validations are currently ongoing, and the stream mechanism will be introduced into MEDM computations to further improve the parallelization and performance.