Time Series Analysis
Using Transprecision Computing

Ivan Fernandez Vega

NiPS Summer School

4 September 2019

UNIVERSIDAD DE MÁLAGA

About me

- 2nd-year **PhD Student at University of Malaga** (Spain)
- Advisors: Oscar Plata and Eladio Gutierrez
- Research topic: **Acceleration of time series analysis**
- Currently at **ETH Zürich** as an academic guest in SAFARI group, supervised by Prof. Onur Mutlu

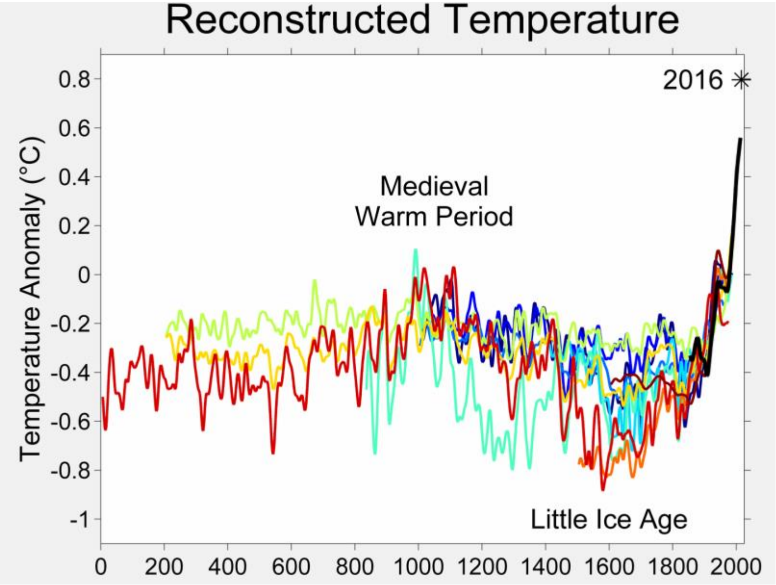
Outline

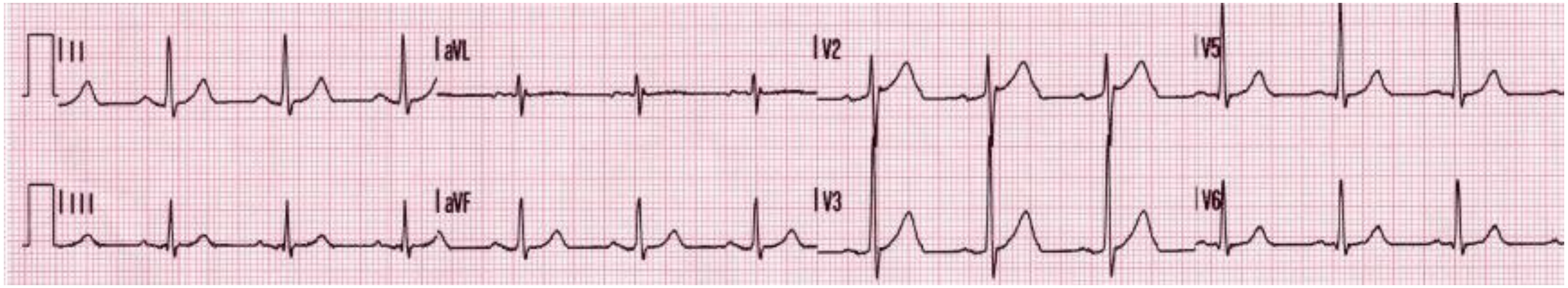
- **Introduction**
- **Background**
- **Implementation**
- **Results**
- **Conclusions and Future Work**

Introduction

Introduction

- **Time series analysis** has a huge interest in many fields
  - Climate
  - Seismology
  - Entomology
  - Bioinformatics
  - Traffic Prediction
  - Voice Recognition
  - Energy Conservation





Introduction

- **Matrix Profile (from UCR Riverside)**
  - Open source tool for **motif discovery** (anomalies, similarities, …)
  - Implemented in several languages: C++, Python, CUDA, R, MATLAB

Matrix Profile website

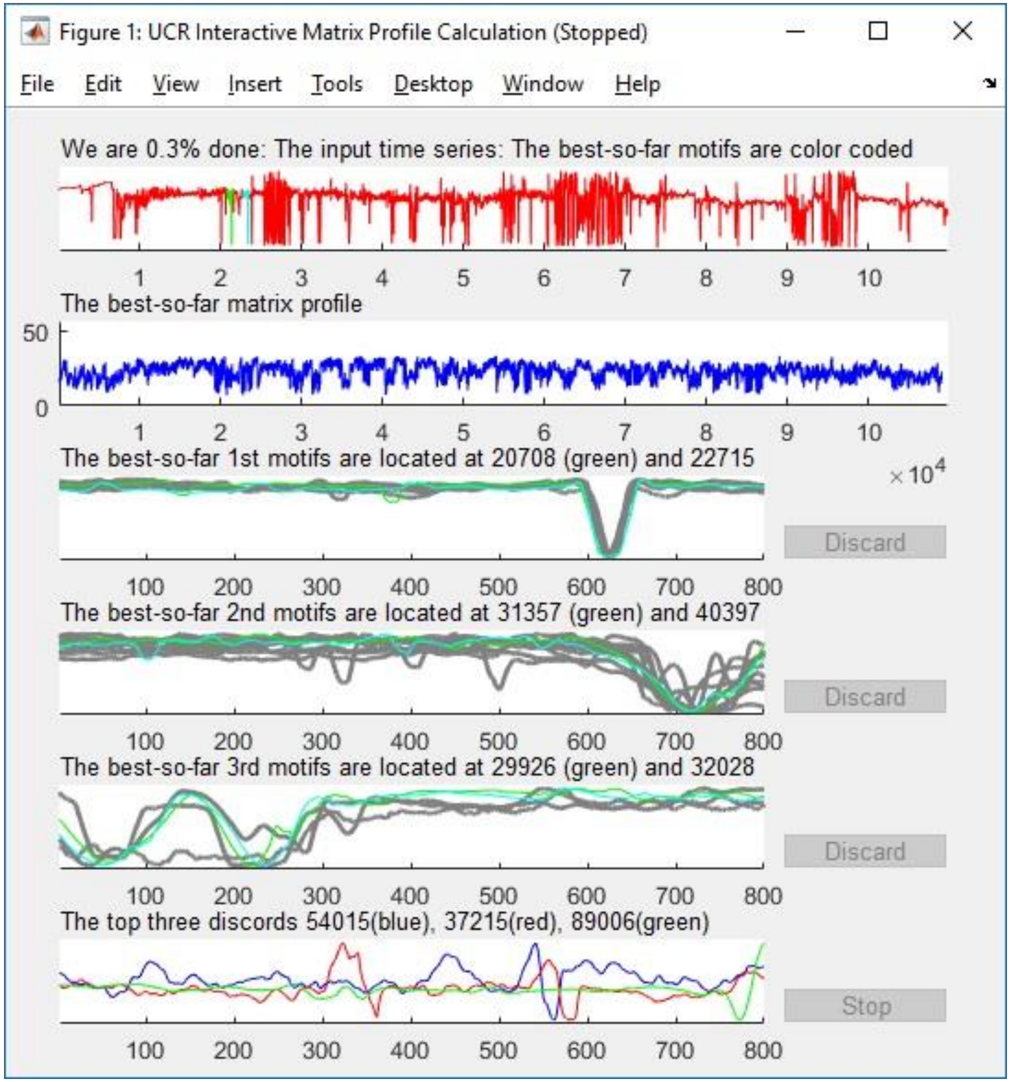


Figure 1: UCR Interactive Matrix Profile Calculation (Stopped)

We are 0.3% done: The input time series: The best-so-far motifs are color coded

The best-so-far matrix profile

The best-so-far 1st motifs are located at 20708 (green) and 22715

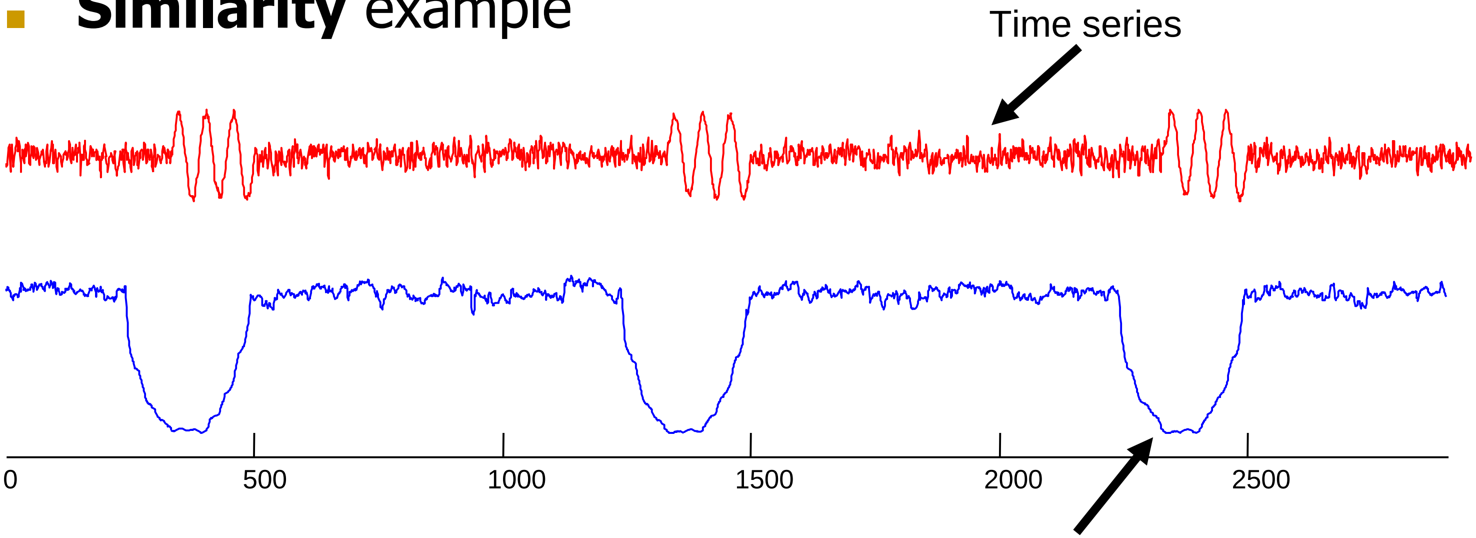
The best-so-far 2nd motifs are located at 31357 (green) and 40397

The best-so-far 3rd motifs are located at 29926 (green) and 32028

The top three discords 54015(blue), 37215(red), 89006(green)

Introduction

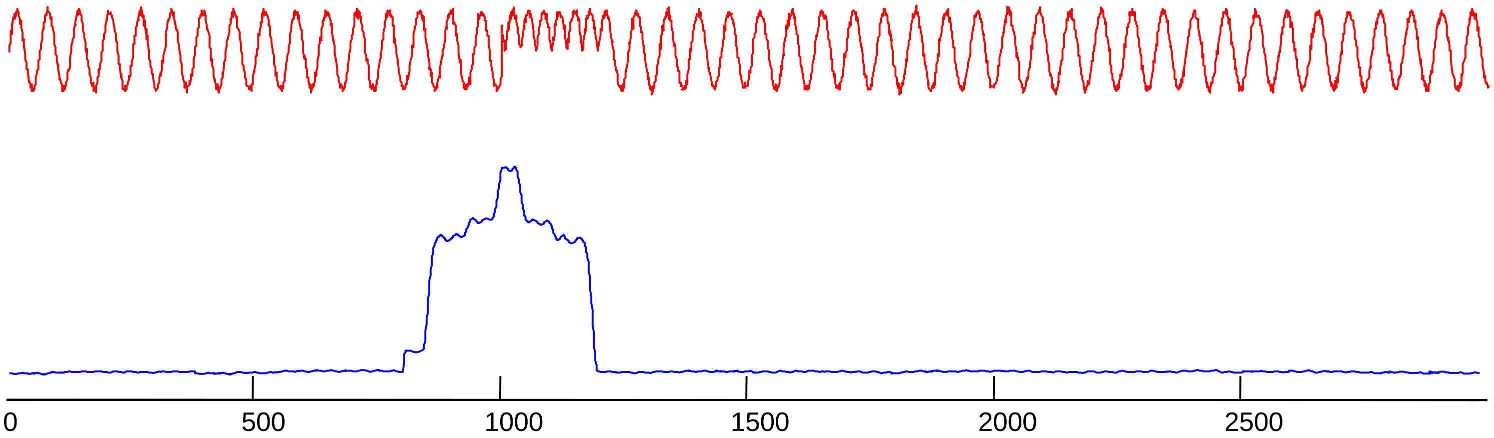
- **Similarity** example



Observation

Similarities appear as lower values of Matrix Profile

- **Anomaly** example

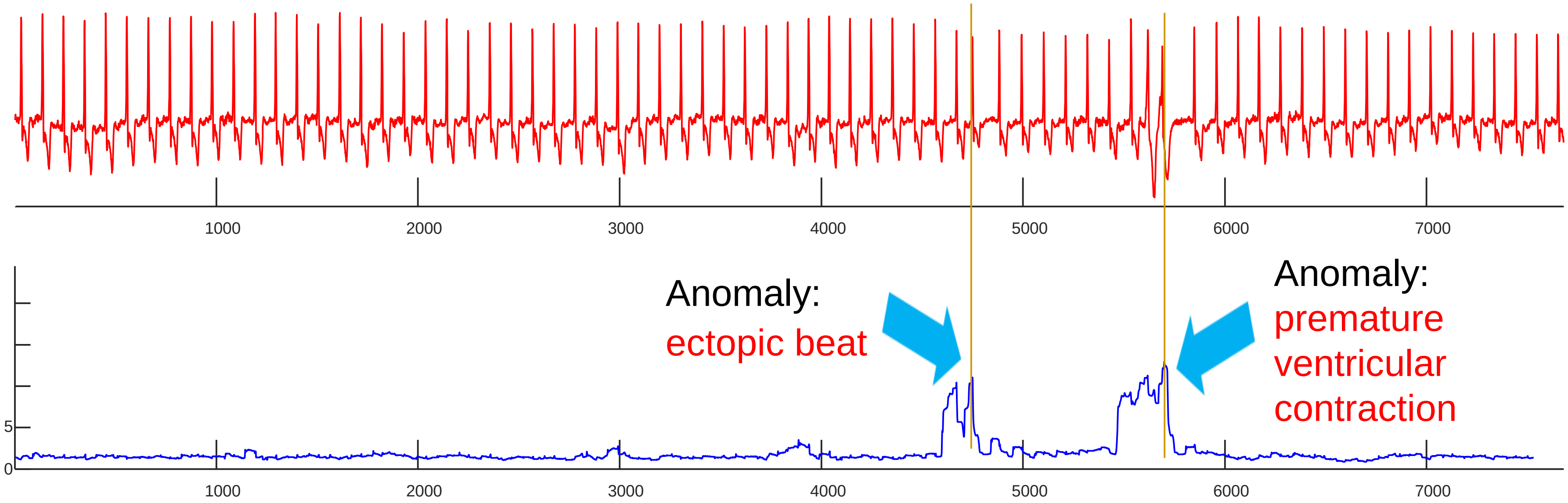


Observation

Anomalies appear as higher values of Matrix Profile

Motivation

- Real data example: **electrocardiogram**
- In this case there are **two anomalies** annotated by MIT cardiologists
- Here the subsequence length was set to 150, but we still find these anomalies if we half or double that length



Anomaly: ectopic beat

Anomaly: premature ventricular contraction

Motivation

- Typical data type used for the computation is **double** precision, while the algorithm allows for **single** or mixed **precision**

- **No previous study using** lower precision or **flex float approach**

- Analysing a time series of **131,072** elements using a window size of **1,024** elements requires:

2.4 Billion subtractions (-)
2.7 Billion multiplications (*)
2.9 Billion divisions (/)
2.8 Billion multiply-accumulations (FMA)
+ 2.8 Billion comparisons (<)

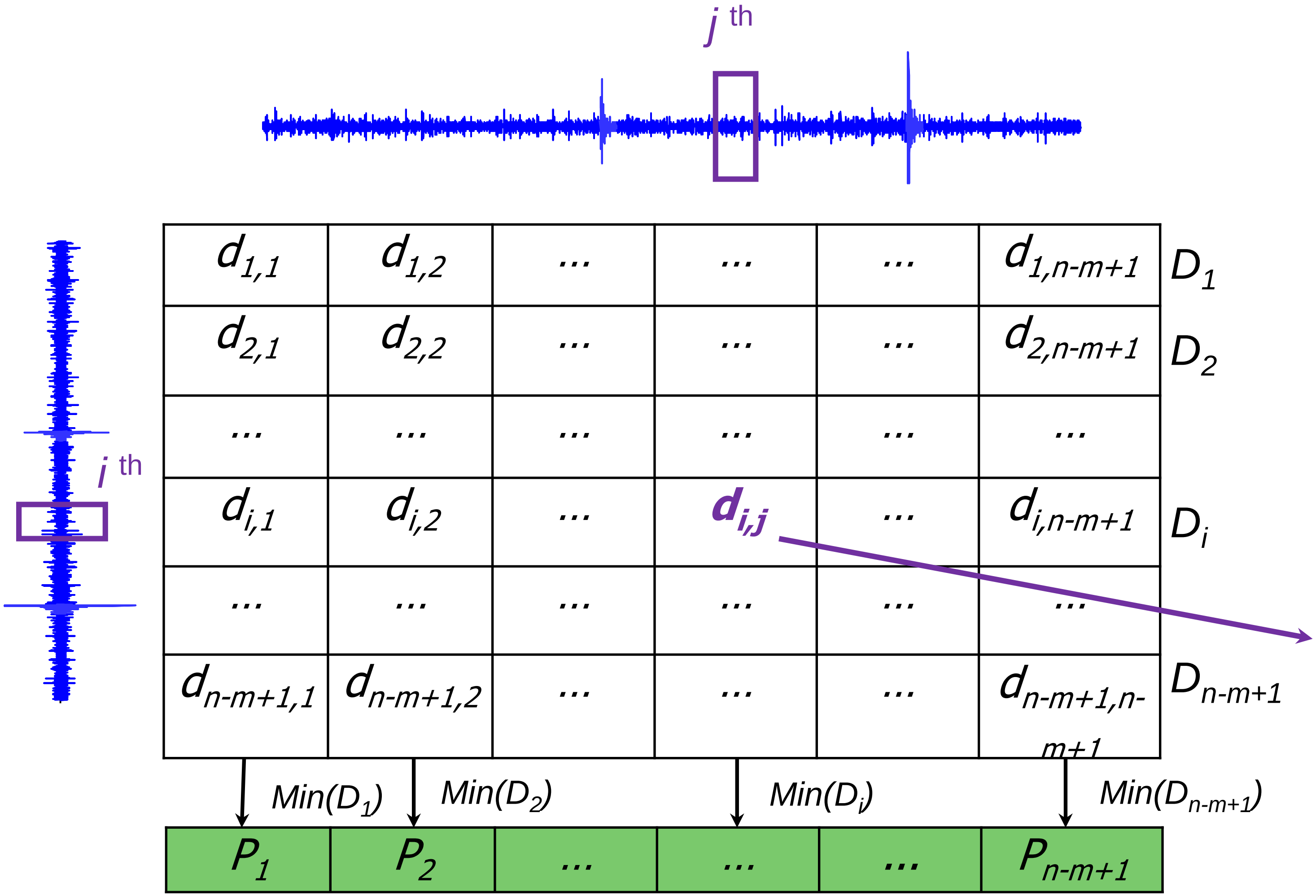
13.6 Billion operations !!!

Observation

The number of operations increases exponentially with the time series length

Background

Distance Matrix



$d_{1,1}$	$d_{1,2}$	…	…	…	$d_{1,n-m+1}$	D_1
$d_{2,1}$	$d_{2,2}$	…	…	…	$d_{2,n-m+1}$	D_2
…	…	…	…	…	…	
$d_{i,1}$	$d_{i,2}$	…	$d_{i,j}$	…	$d_{i,n-m+1}$	D_i
…	…	…	…	…	…	
$d_{n-m+1,1}$	$d_{n-m+1,2}$	…	…	…	$d_{n-m+1,n-m+1}$	D_{n-m+1}

$Min(D_1)$ $Min(D_2)$ $Min(D_i)$ $Min(D_{n-m+1})$

P_1	P_2	…	…	…	P_{n-m+1}

- Symmetric matrix
- Main diagonal = 0
- Cells close to diagonal very small

$d_{i,j}$ is the distance between the i^{th} window and the j^{th} window of the time series

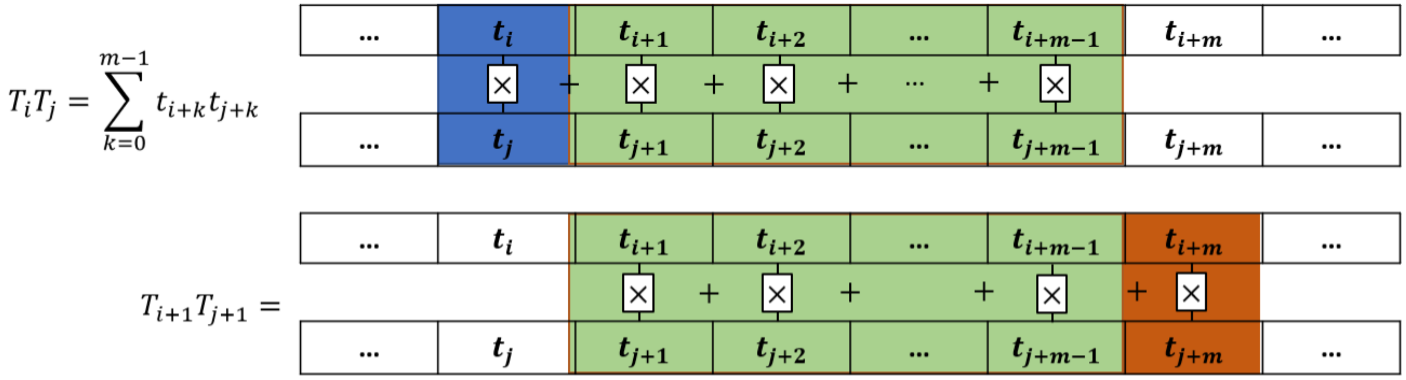
Matrix Profile: a vector of distance between each subsequence and its most similar one

Distance metric

- The similarity $d_{i,j}$ is based on Euclidean distances:

$$d_{i,j} = \sqrt{2m\left(1 - \frac{Q_{i,j} - \mu_i \mu_j}{m\sigma_i \sigma_j}\right)}$$

- The dot product ($Q_{i,j}$) can be calculated as follows:


$T_i T_j = \sum_{k=0}^{m-1} t_{i+k} t_{j+k}$
... t_i t_{i+1} t_{i+2} ... t_{i+m-1} t_{i+m} ...
⊠ + ⊠ + ⊠ + ... + ⊠
... t_j t_{j+1} t_{j+2} ... t_{j+m-1} t_{j+m} ...
$T_{i+1} T_{j+1} =$
... t_i t_{i+1} t_{i+2} ... t_{i+m-1} t_{i+m} ...
⊠ + ⊠ + + ⊠ + ⊠
... t_j t_{j+1} t_{j+2} ... t_{j+m-1} t_{j+m} ...

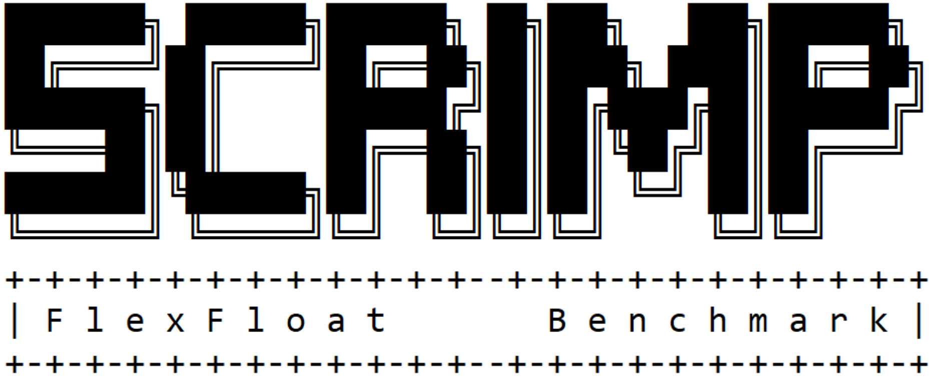
$$T_{i+1} T_{j+1} = T_i T_j - t_i t_j + t_{i+m} t_{j+m}$$

$O(1)$ time complexity

Implementation

Goal

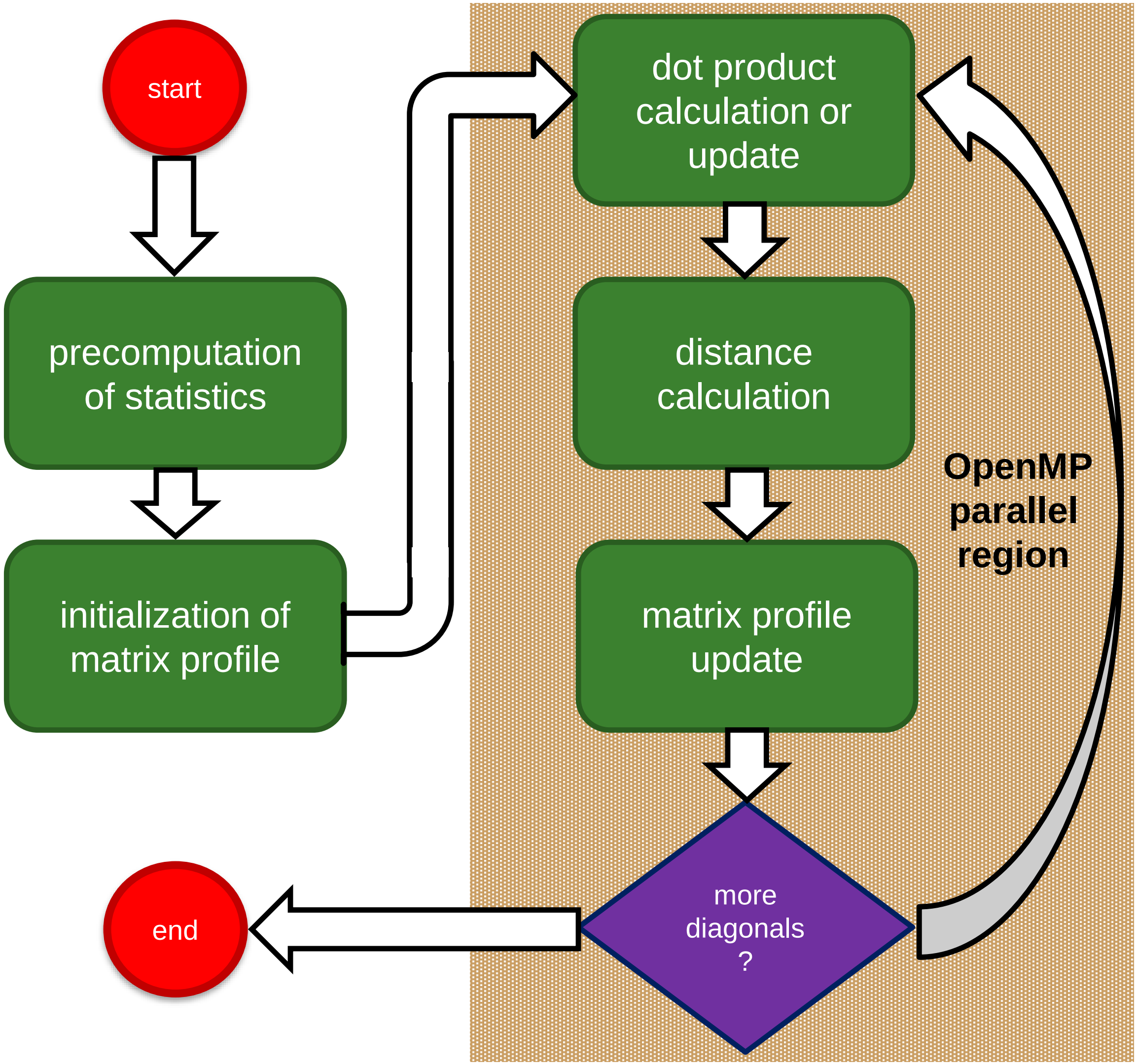
- The goal is to provide a benchmark to explore how the **accuracy** of the results of SCRIMP are affected by **changing the precision** of the floating-point operations
- This tool would be **useful for architects when designing a custom accelerator** for time series analysis
- The implementation is **open source** and based on **FlexFloat**



FlexFloat @ Github

SCRIMP FF

- SCRIMP FF computation scheme and configuration parameters


start
precomputation of statistics
initialization of matrix profile
dot product calculation or update
distance calculation
matrix profile update
more diagonals ?
end
OpenMP parallel region

The user can configure individually the precision for each block via a config file

Configurable precisions
Statistics (μ and σ)
Dot product
Distance
Matrix profile

User Interface

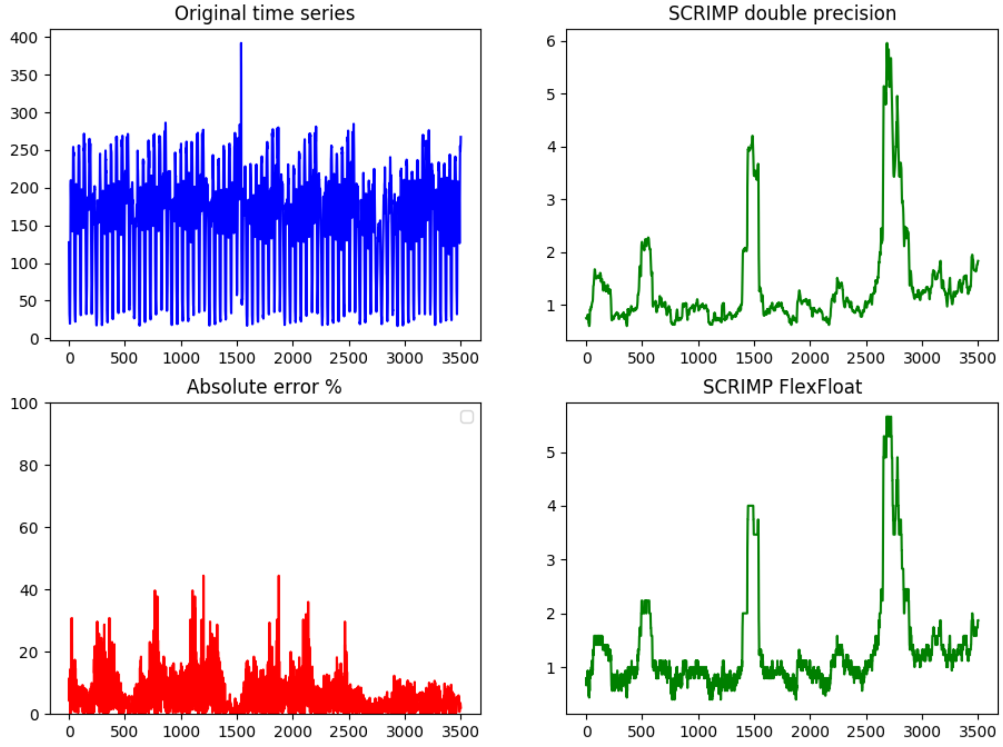
- SCRIMP FF example **input**:

$$\underbrace{\texttt{./scrimp\_ff}}_{\text{binary}}\quad \underbrace{\texttt{power\_demand.txt}}_{\text{time series file}}\quad \underbrace{\texttt{200}}_{\text{window size}}\quad \underbrace{\texttt{72}}_{\text{\# threads}}\quad \underbrace{\texttt{0.1}}_{\text{scale factor}}$$

- SCRIMP FF example **output**:

original time series →

absolute error % →



← result using original implementation (64 bits)

← result using FlexFloat implementation

FF parameters [exp, man] => distance=[7, 16]; dotprod=[7, 16]; stats=[6, 12]; profile=[5, 2]

Results

Experiments

- The benchmark has been tested using a server equipped with two Intel Xeon Gold 6154 (**72 threads**) and 384 GB of DDR4 memory

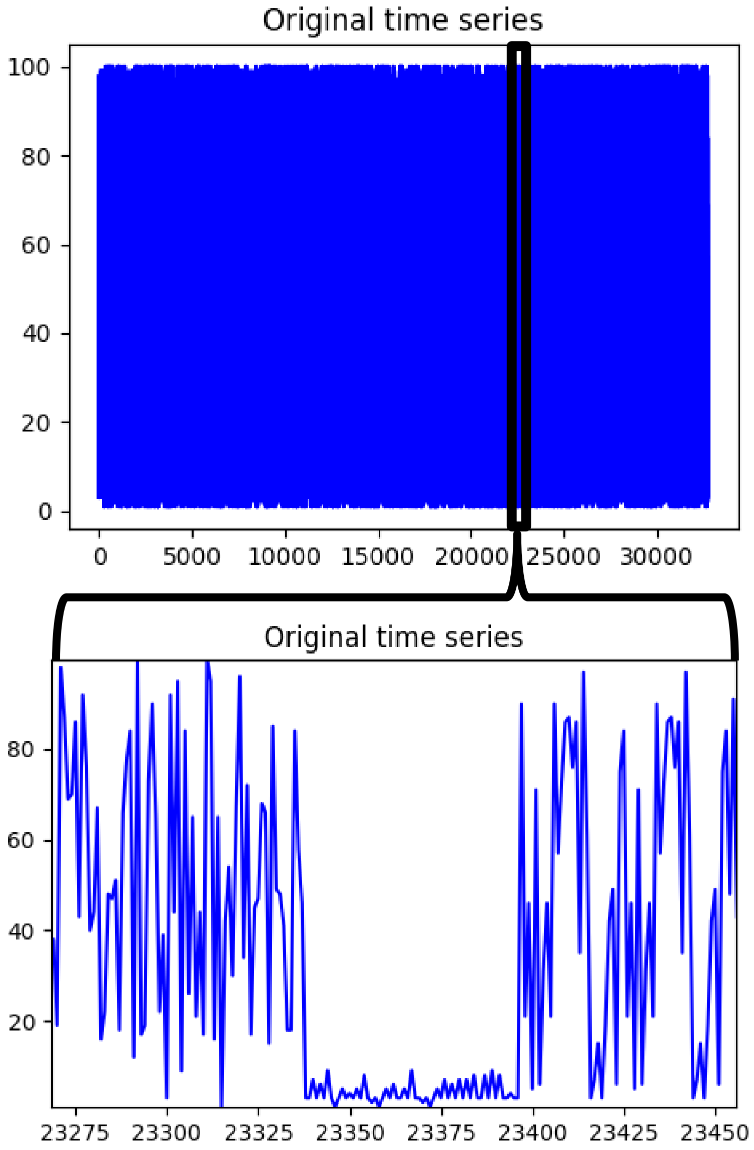
- Each FlexFloat execution is compared with the original code

- In this presentation I cover four didactical examples:
  - (1) Synthetic **random time series** with one anomaly
  - (2) Synthetic **random time series** with two (very) similar subsequences
  - (3) **Real data** time series with four anomalies
  - (4) **Real data** time series with twelve anomalies

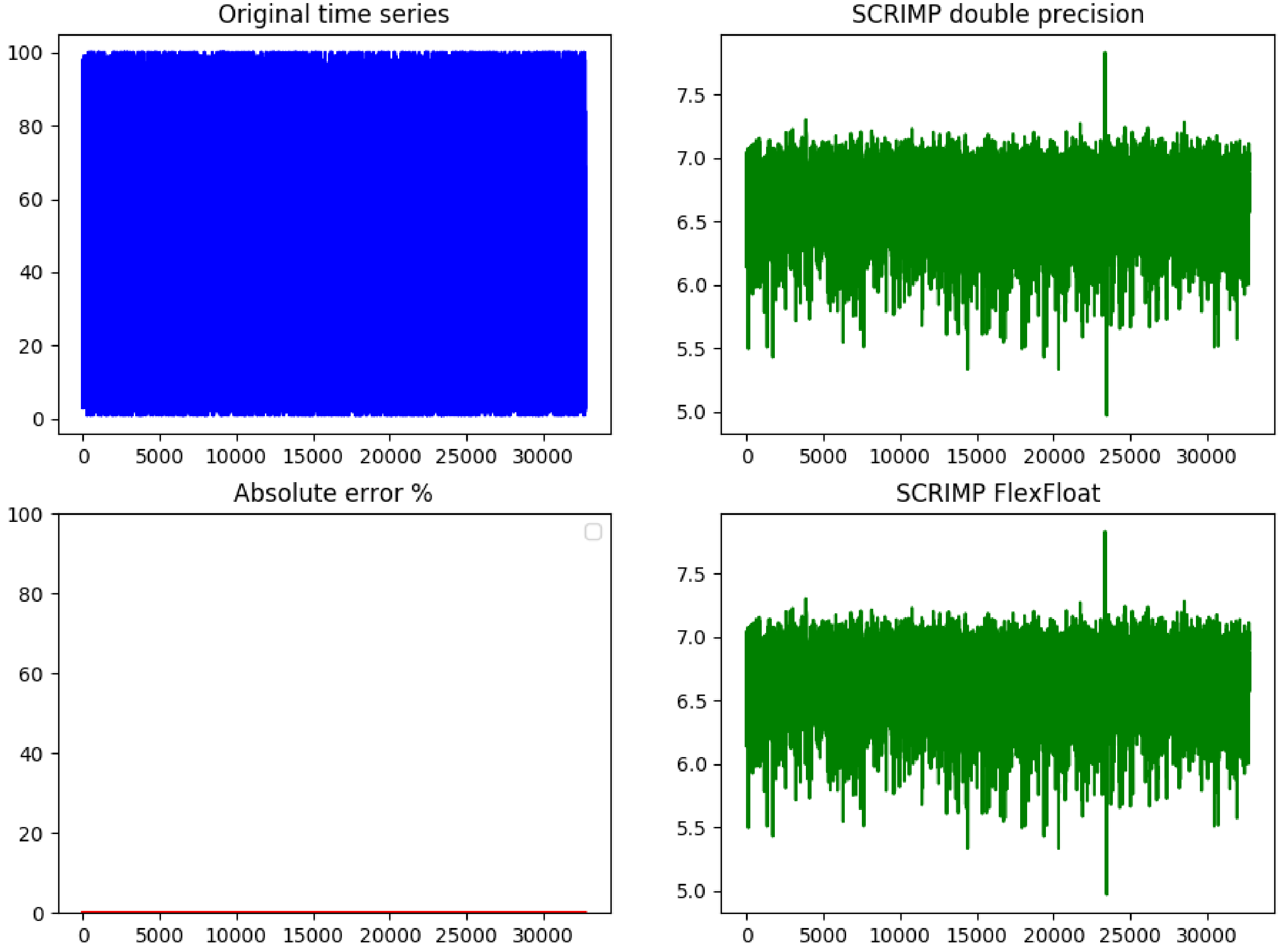
Computing a 32,768 elements time series takes approx. 4 minutes in this sever

Random Serie Anomaly

- **Case study #1**
  - Random time series
  - Values from 0 to 100
  - 32,768 elements
  - 50 window size length
  - One anomaly



Random Serie Anomaly - 64 Bits

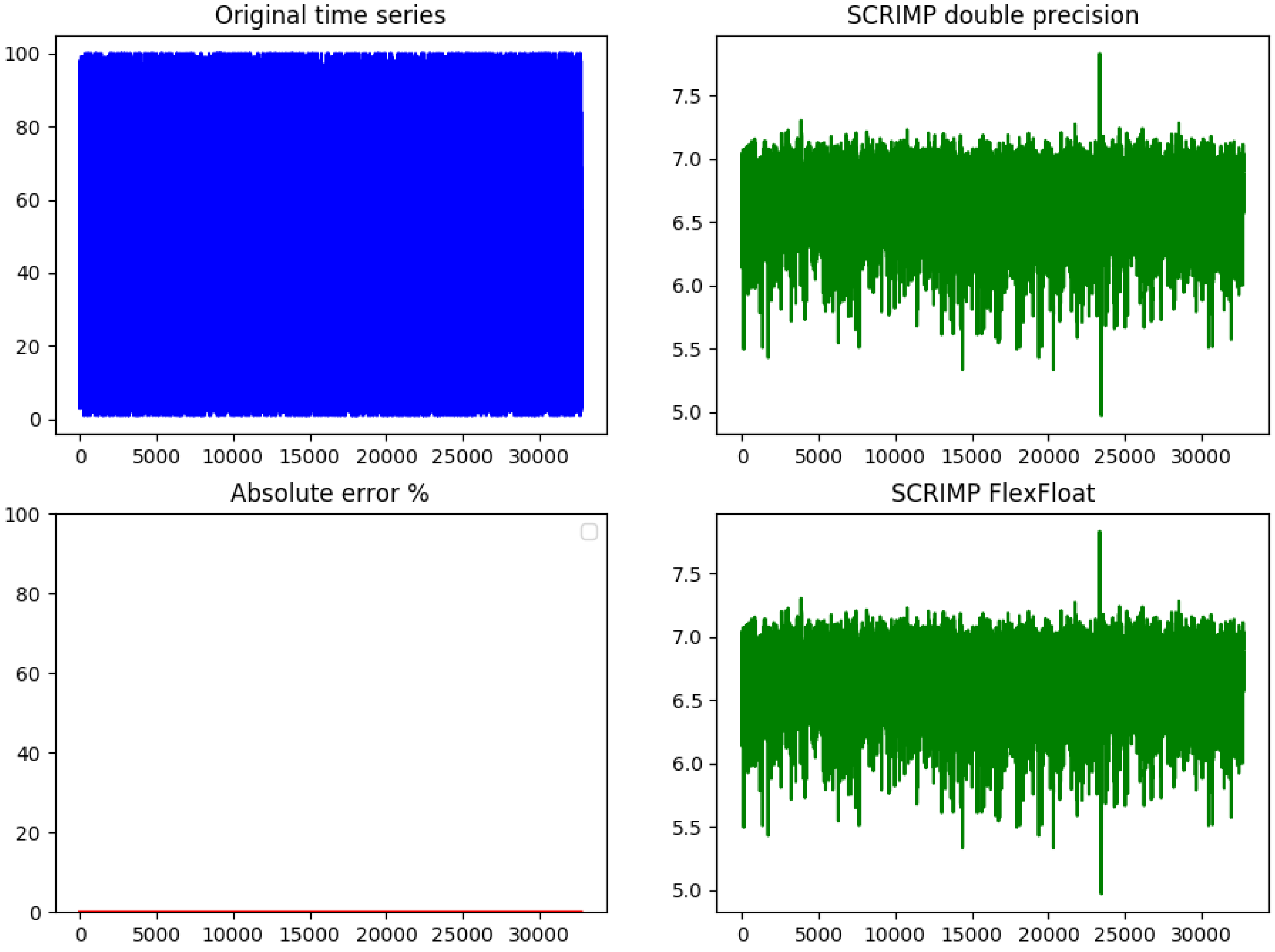


FF parameters [exp, man] => distance=[11, 52]; dotprod=[11, 52]; stats=[11, 52]; profile=[11, 52]

Observation

Using 64-bit precision and Flex Float we obtain no error, as expected

Random Serie Anomaly - 32 Bits

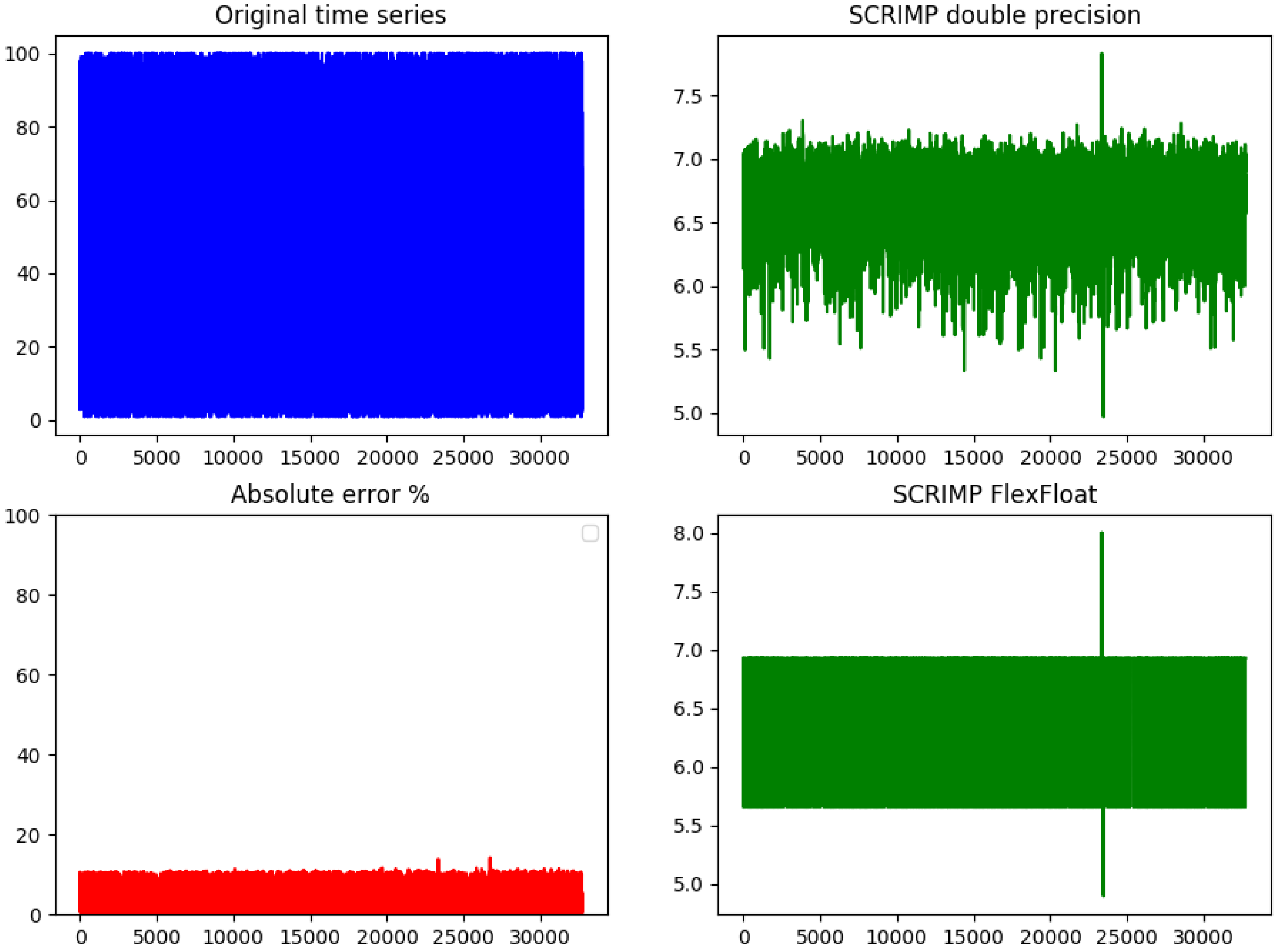


FF parameters [exp, man] => distance=[8, 23]; dotprod=[8, 23]; stats=[8, 23]; profile=[8, 23]

Observation

Using 32-bit precision and Flex Float we still obtain no error!!

Random Serie Anomaly - Reduced

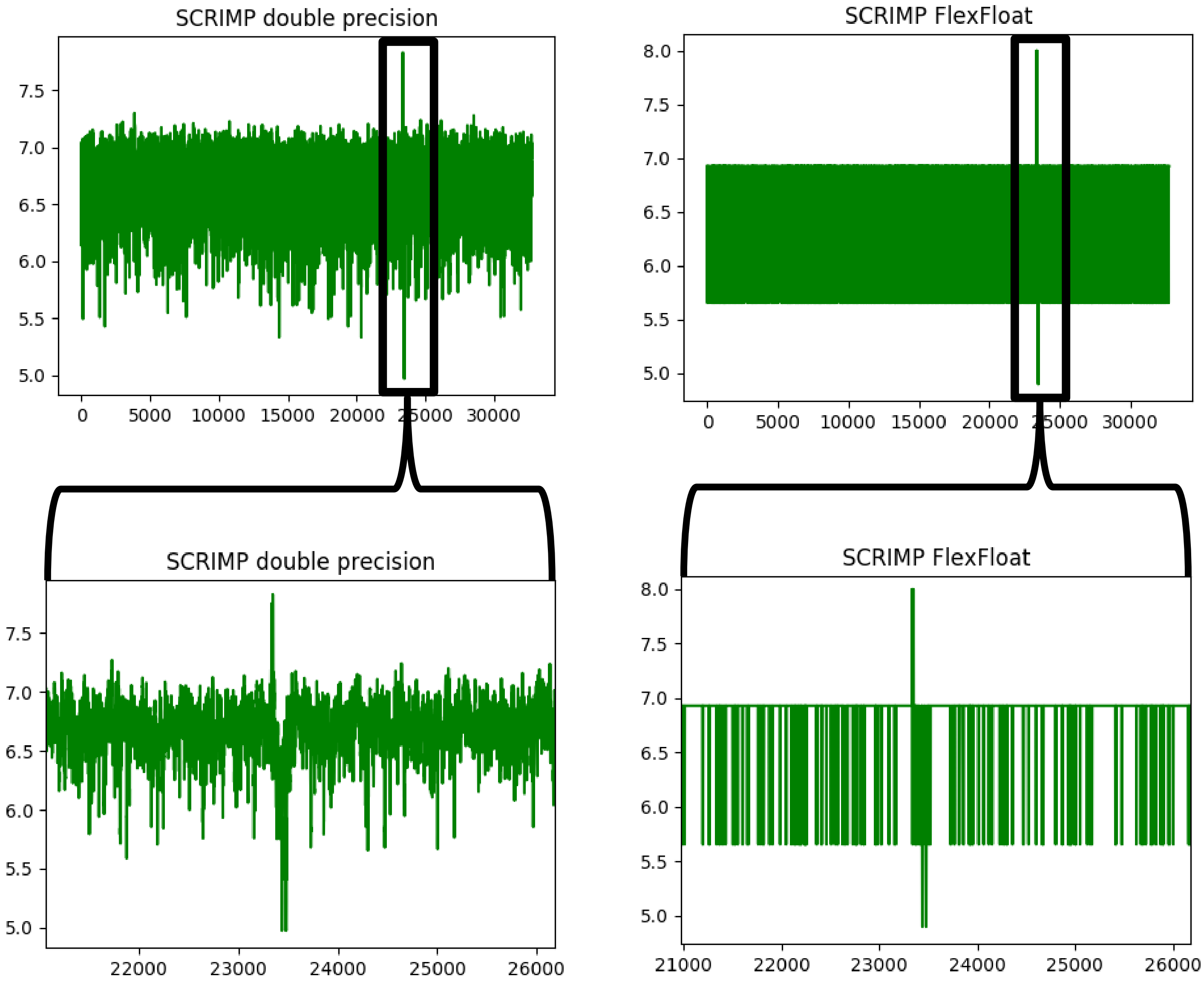


FF parameters [exp, man] => distance=[6, 15]; dotprod=[6, 10]; stats=[6, 12]; profile=[6, 1]

Observation

When we reduce significantly the precision we get just ~10% error

Random Serie Anomaly - Profile Zoom

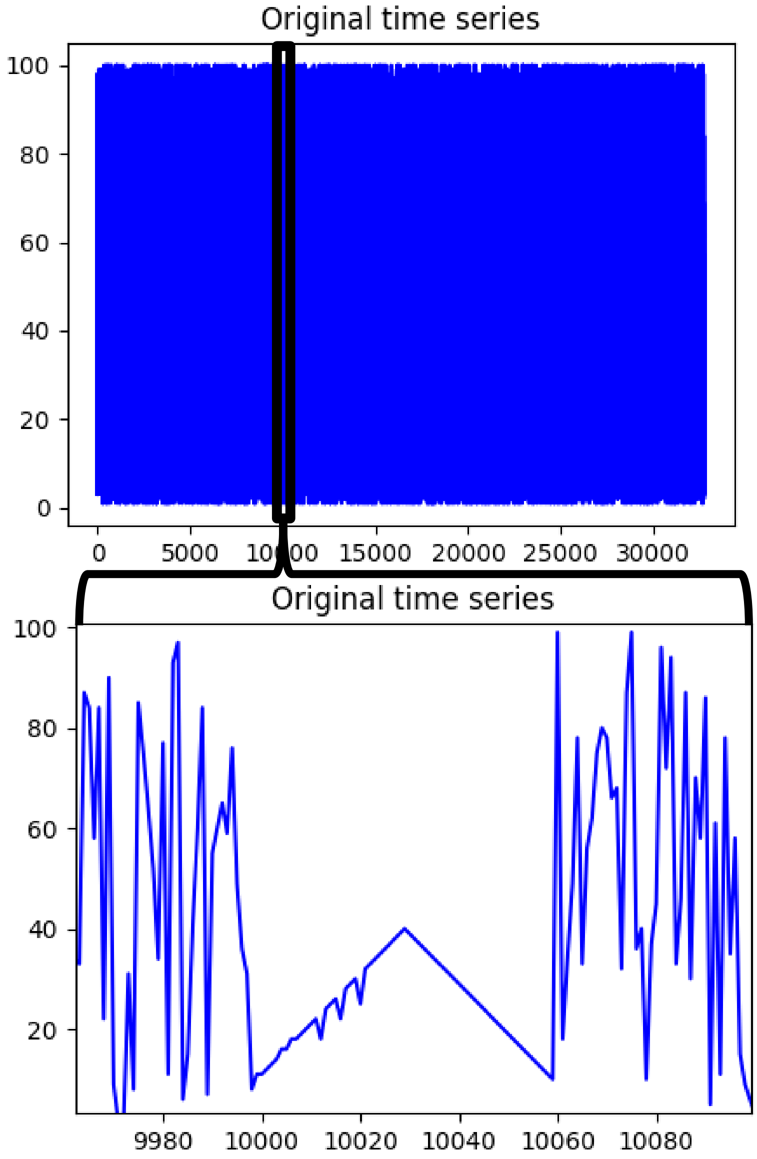


Observation

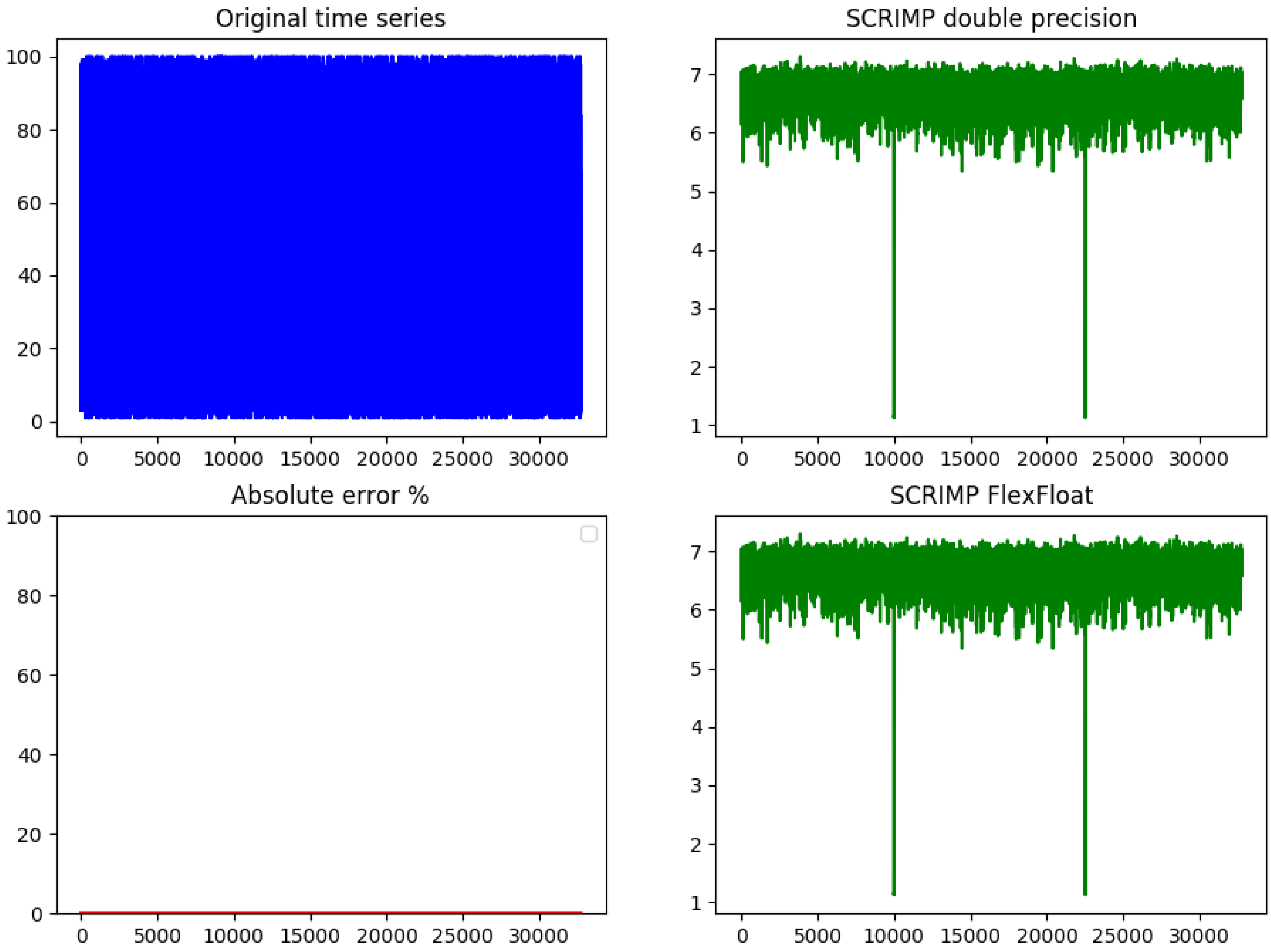
The anomaly is very easily detectable using the Flex Float approach

Random Serie Similarity

- **Case study #2**
  - Random time series
  - Values from 0 to 100
  - 32,768 elements
  - 50 window size length
  - Two (very) similar subsequences



Random Serie Similarity - 64 bits

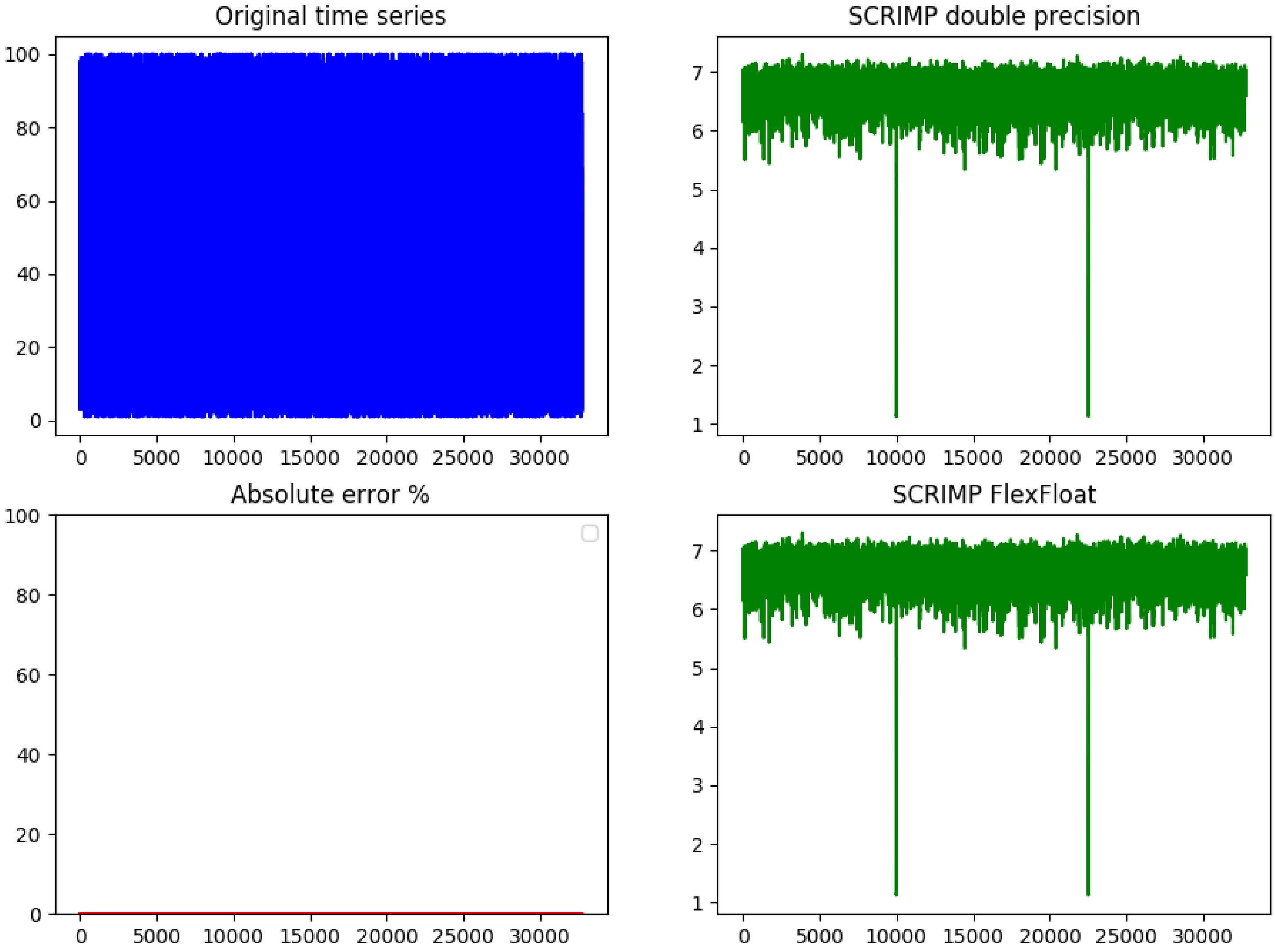


Observation

Using 64-bit precision and Flex Float we obtain no error, as expected

FF parameters [exp, man] => distance=[11, 52]; dotprod=[11, 52]; stats=[11, 52]; profile=[11, 52]

Random Serie Similarity - 32 Bits


Original time series

SCRIMP double precision

Absolute error %

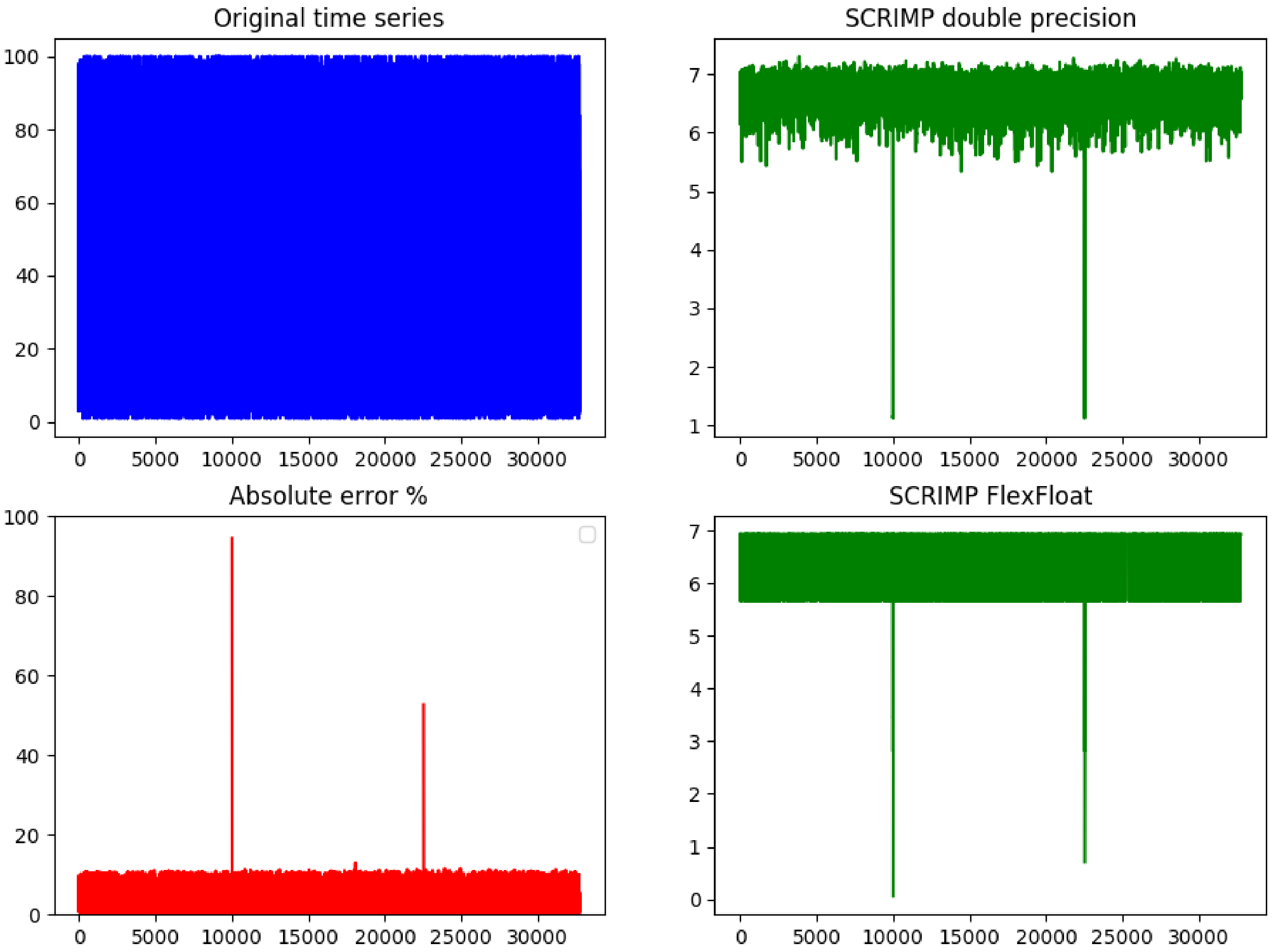
SCRIMP FlexFloat

FF parameters [exp, man] => distance=[8, 23]; dotprod=[8, 23]; stats=[8, 23]; profile=[8, 23]

Observation

Using 32-bit precision and Flex Float we still obtain no error!!

Random Serie Similarity - Reduced

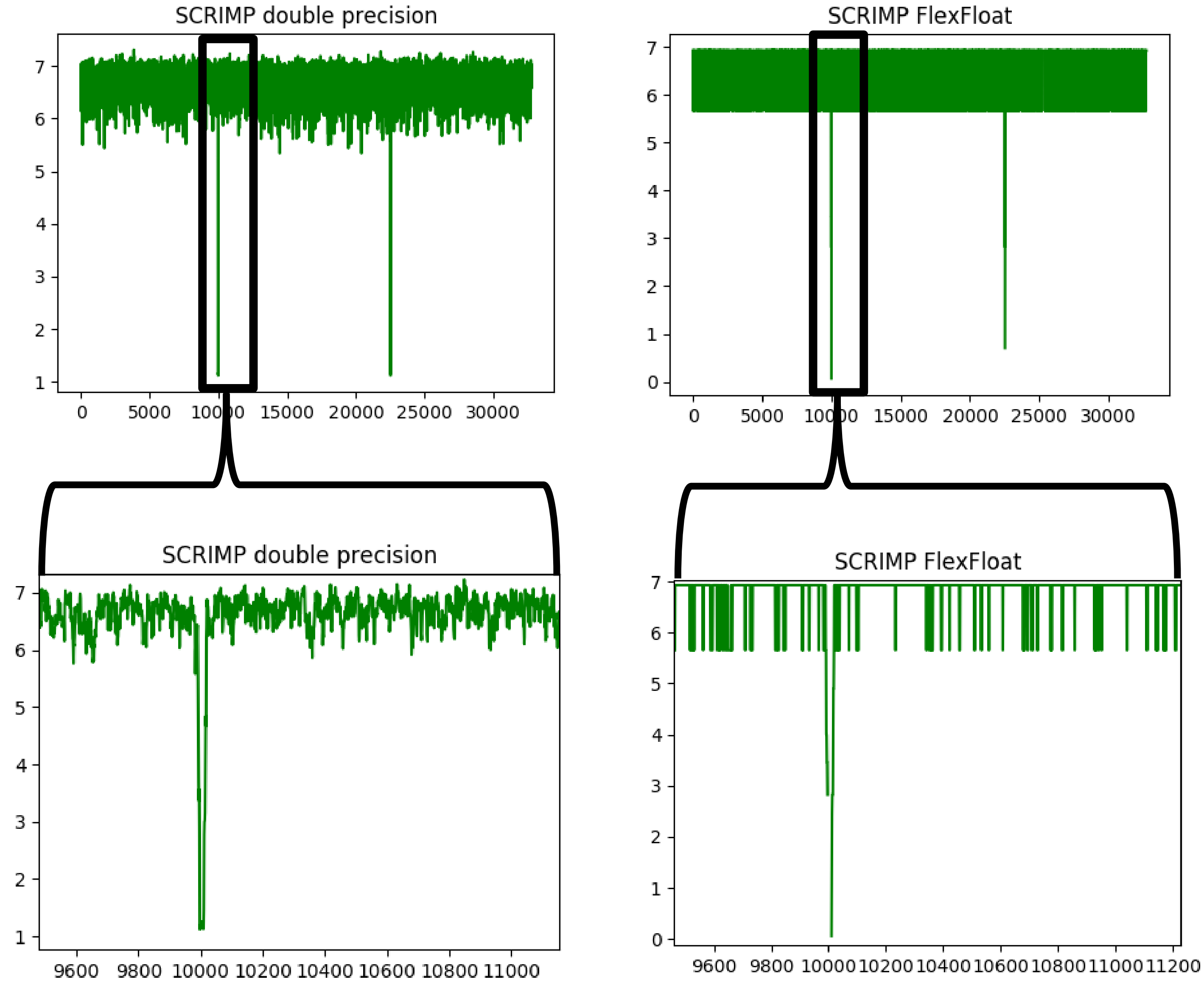


Observation

We obtain error in the lower values, however they are still detectable

FF parameters [exp, man] => distance=[6, 17]; dotprod=[6, 15]; stats=[6, 10]; profile=[5, 1]

Random Serie Similarity - Profile Zoom



SCRIMP double precision

SCRIMP FlexFloat

SCRIMP double precision

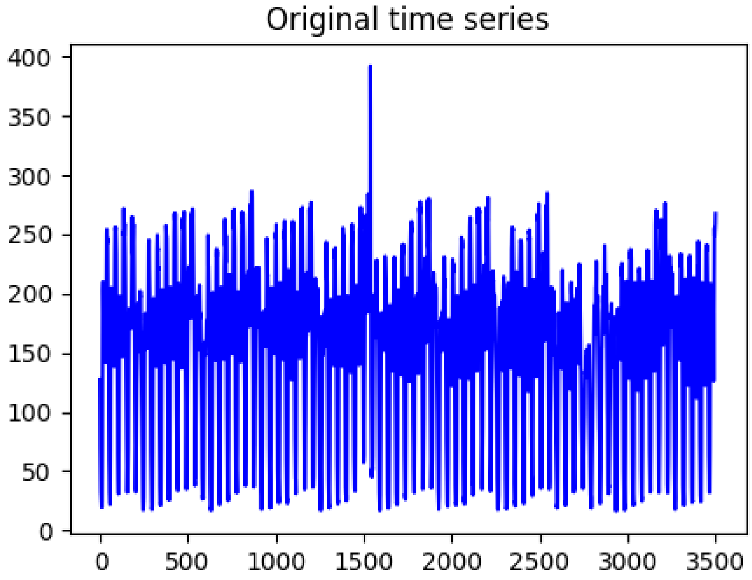
SCRIMP FlexFloat

Observation

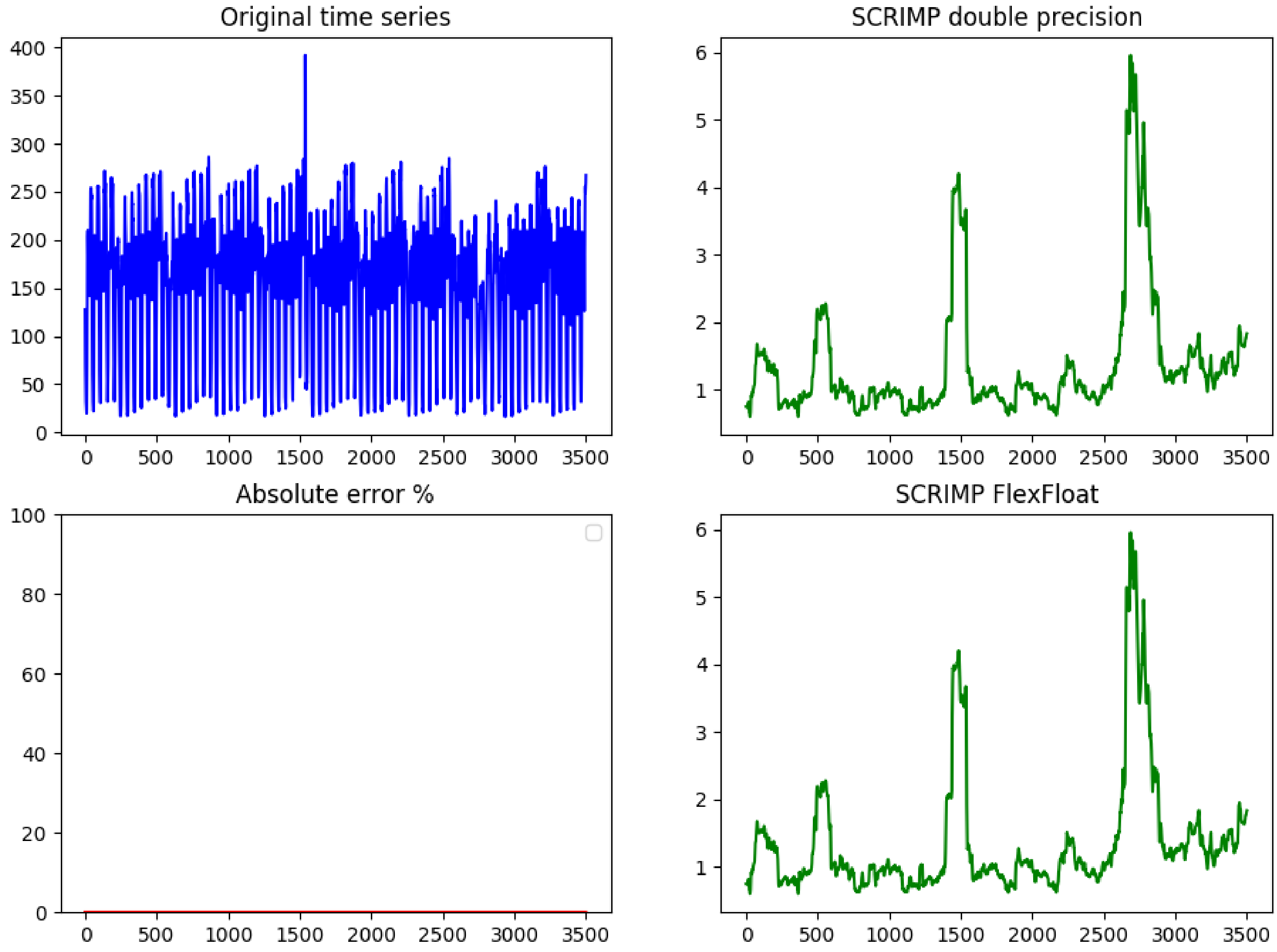
The similarities are still detectable using Flex Float

Taxi Demand Data

- **Case study #3**
  - Taxi demand data
  - 3,600 elements
  - 50 window size length
  - Four anomalies



Taxi Demand Data - 64 Bits

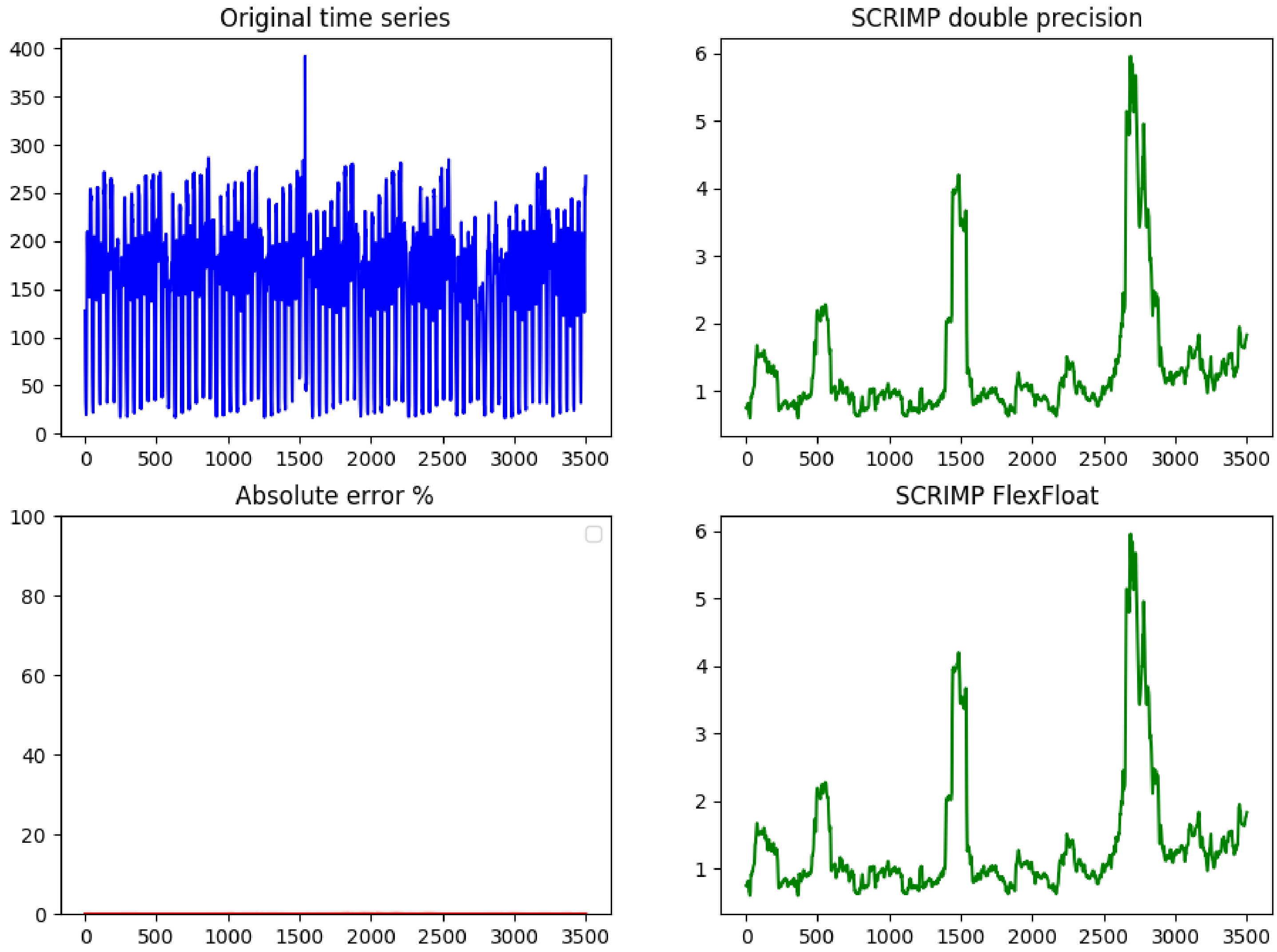


FF parameters [exp, man] => distance=[11, 52]; dotprod=[11, 52]; stats=[11, 52]; profile=[11, 52]

Observation

Using 64-bit precision and Flex Float we obtain no error, as expected

Taxi Demand Data - 32 Bits

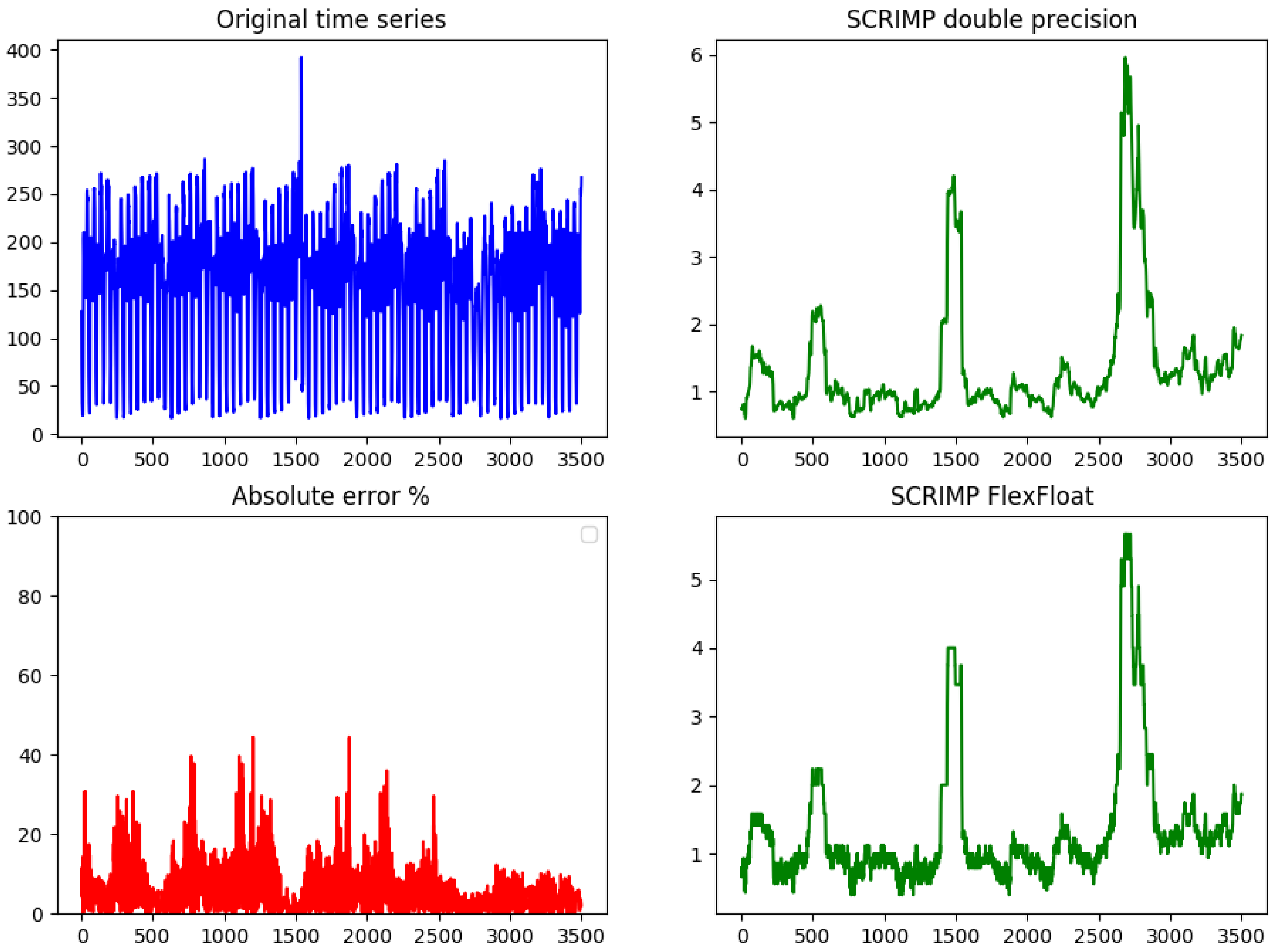


FF parameters [exp, man] => distance=[8, 23]; dotprod=[8, 23]; stats=[8, 23]; profile=[8, 23]

Observation

Using 32-bit precision and Flex Float we still obtain no error!!

Taxi Demand Data - Reduced

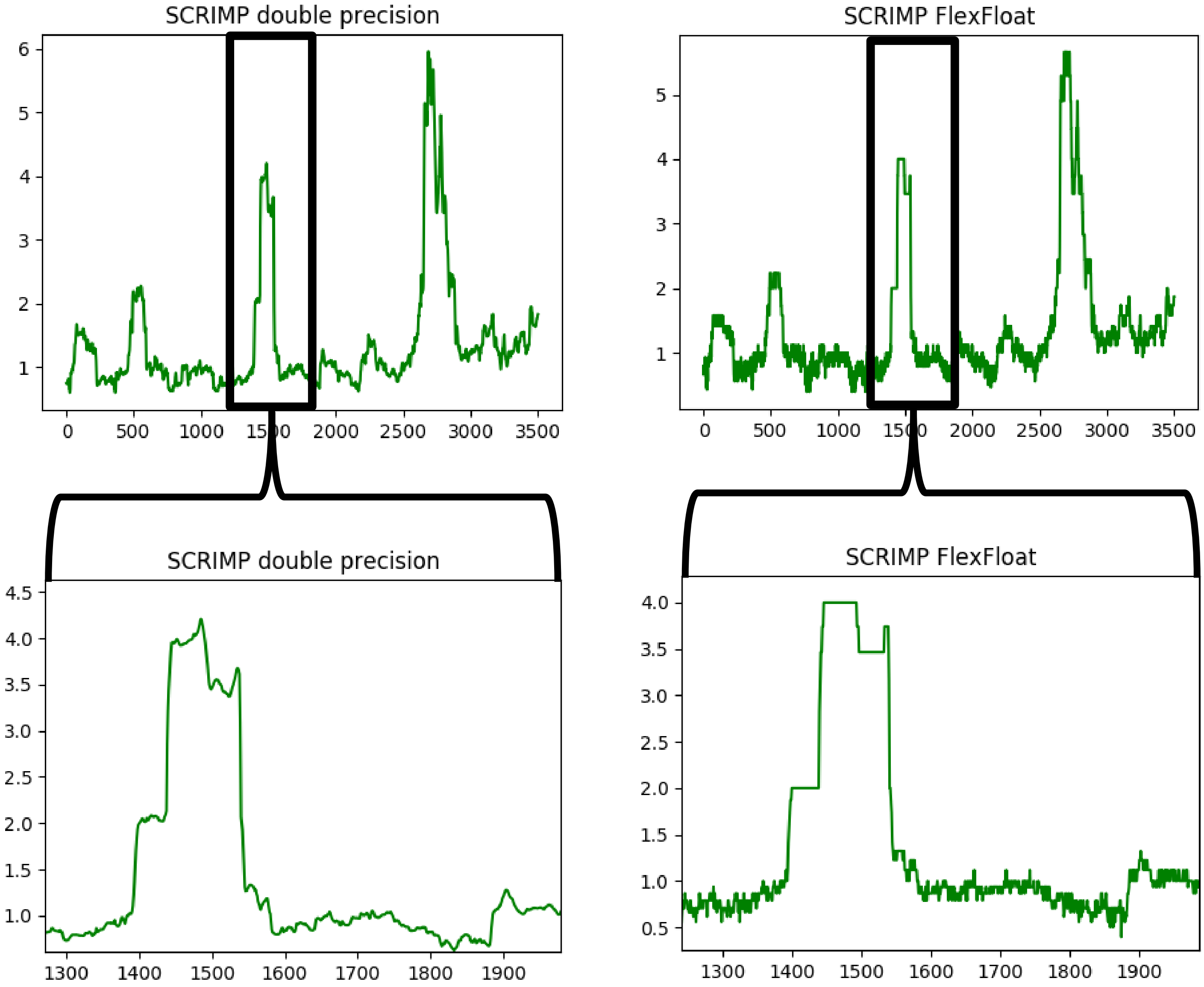


FF parameters [exp, man] => distance=[7, 16]; dotprod=[7, 16]; stats=[6, 12]; profile=[5, 2]

Observation

We obtain error in lower values, but anomalies are still detectable

Taxi Demand Data - Profile Zoom

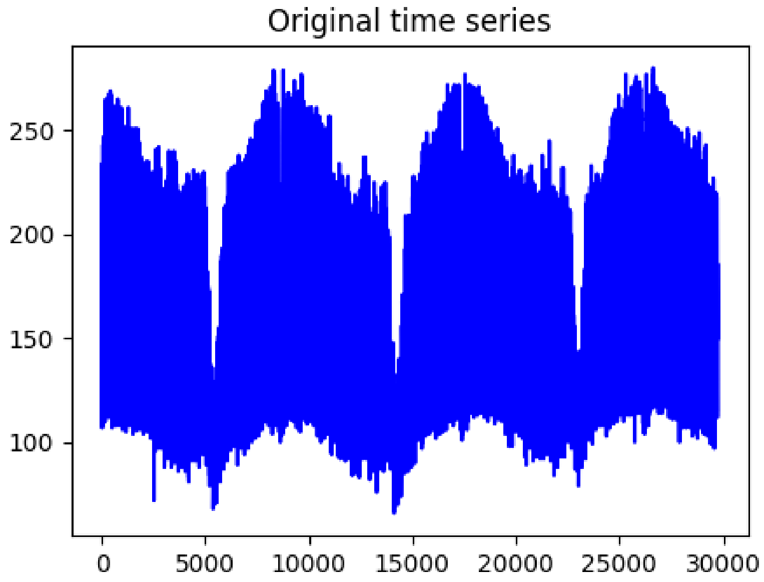


Observation

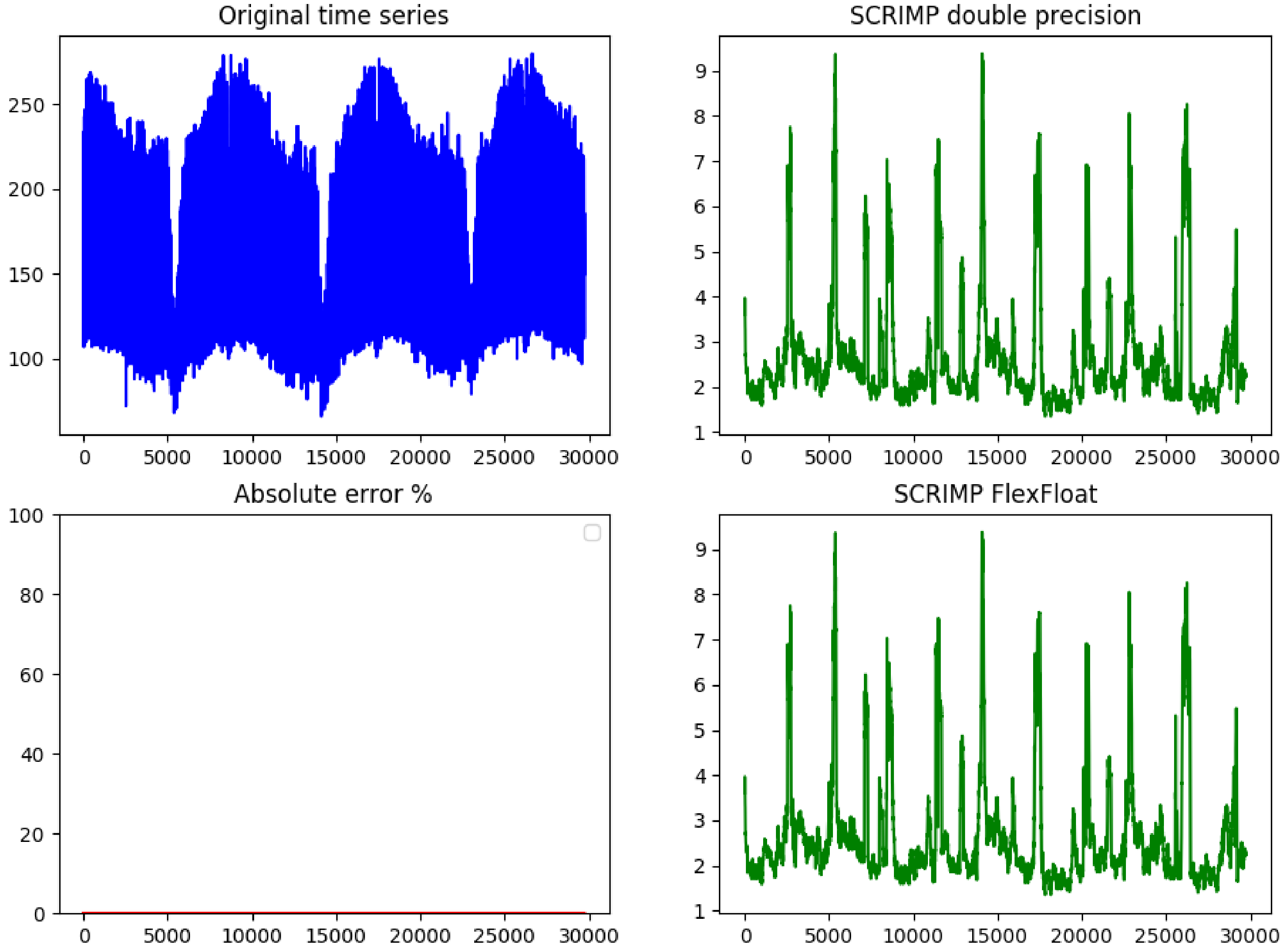
The anomalies are still detectable using Flex Float

Power Demand Data

- **Case study #4**
  - Electric power demand data
  - 30,000 elements
  - 50 window size length
  - Twelve anomalies



Power Demand Data - 64 Bits

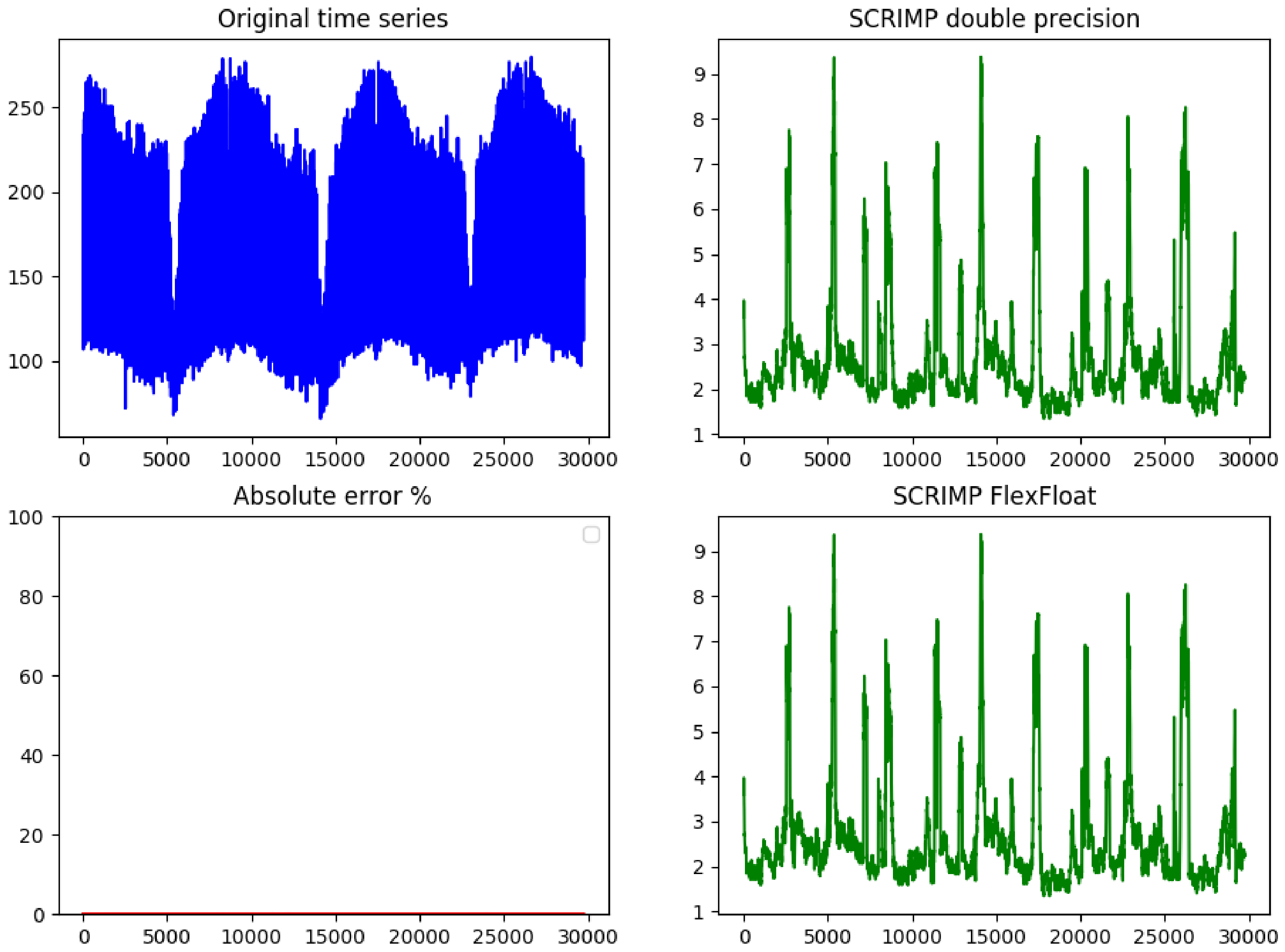


FF parameters [exp, man] => distance=[11, 52]; dotprod=[11, 52]; stats=[11, 52]; profile=[11, 52]

Observation

Using 64-bit precision and Flex Float we obtain no error, as expected

Power Demand Data - 32 Bits

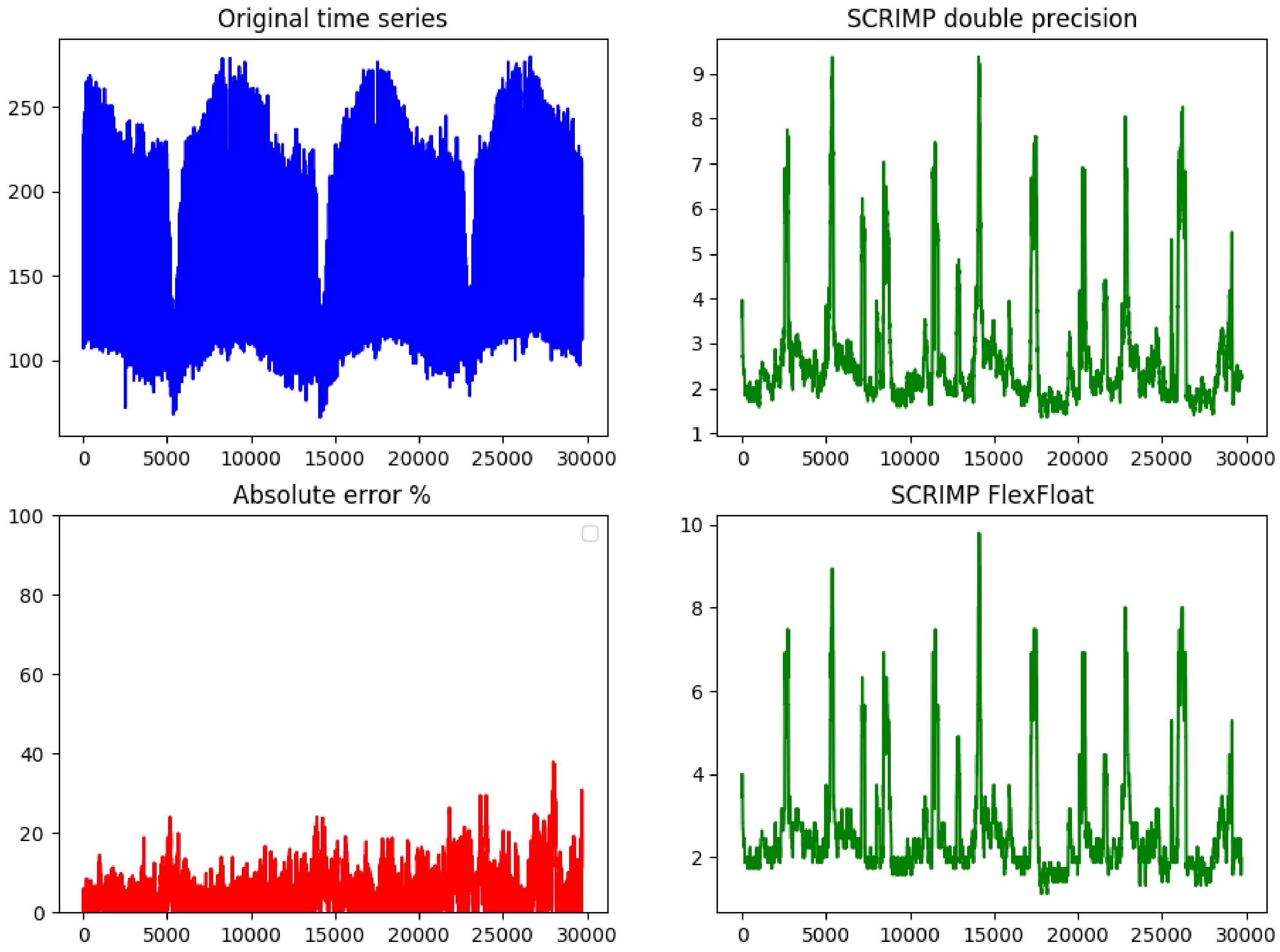


Observation

Using 32-bit precision and Flex Float we still obtain no error!!

FF parameters [exp, man] => distance=[8, 23]; dotprod=[8, 23]; stats=[8, 23]; profile=[8, 23]

Power Demand Data - Reduced

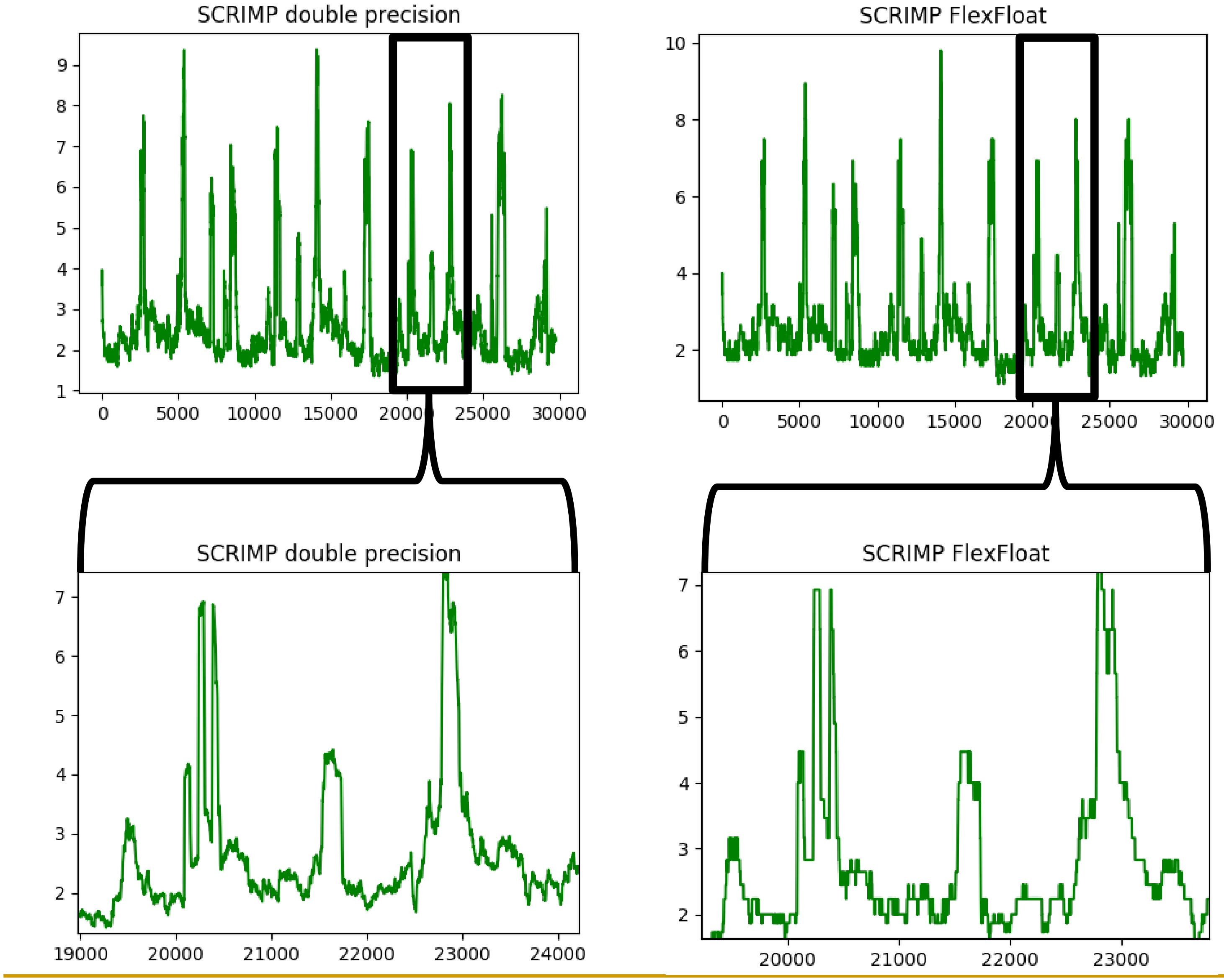


FF parameters [exp, man] => distance=[6, 17]; dotprod=[6, 17]; stats=[6, 17]; profile=[5, 2]

Observation

We obtain error in lower values, but anomalies are still detectable

Power Demand Data - Profile Zoom



Observation

The anomalies are still detectable using Flex Float

Conclusions and Future Work

Conclusions and Future Work

- Matrix profile can be useful for **many** time series motif discovery **applications**

- SCRIMP FlexFloat benchmark allows the **exploration of reduced precision** computation of Matrix Profile

- Architects could design accelerators using the exact amount of precision needed for each application, **maximizing performance** and **minimizing energy consumption**

- **Future work** comprises evaluating time series analysis using a non emulated transprecision computing environment as pulp-platform

References

- Some of the examples are taken from the **Matrix Profile tutorial** available at https://www.cs.ucr.edu/~eamonn/MatrixProfile.html

- **SCRIMP**:
  - Zhu, Y., Yeh, C. C. M., Zimmerman, Z., Kamgar, K., & Keogh, E. (2018, November). Matrix profile XI: SCRIMP++: time series motif discovery at interactive speeds. In *2018 IEEE International Conference on Data Mining (ICDM)* (pp. 837-846). IEEE.
  - https://sites.google.com/site/scrimpplusplus/

- **FlexFloat**:
  - G. Tagliavini, A. Marongiu and L. Benini, "FlexFloat: A Software Library for Transprecision Computing," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*.
  - https://github.com/oprecomp/flexfloat

Time Series Analysis
Using Transprecision Computing

Ivan Fernandez Vega

NiPS Summer School

4 September 2019

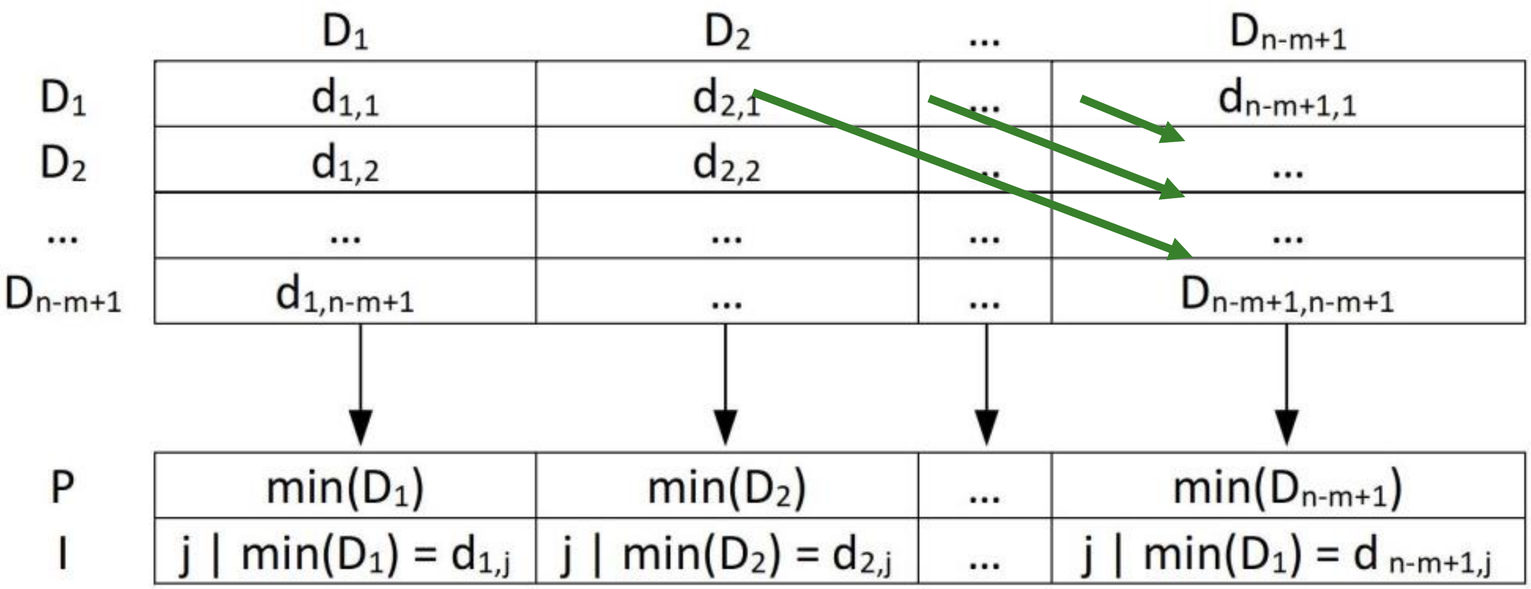
UNIVERSIDAD DE MÁLAGA

Backup Slides

SCRIMP

- Matrix Profile implementation (SCRIMP)
  - Takes advantage of the dot product from the previous step performing the calculations in **diagonals** instead of columns or rows:

$$Q_{i,j} = Q_{i-1,j-1} - t_{i-1}t_{j-1} + t_{i+m-1}t_{j+m-1}$$



	D_1	D_2	...	D_{n-m+1}
D_1	$d_{1,1}$	$d_{2,1}$	...	$d_{n-m+1,1}$
D_2	$d_{1,2}$	$d_{2,2}$	...	...
...	...	...	...	...
D_{n-m+1}	$d_{1,n-m+1}$	...	...	$D_{n-m+1,n-m+1}$

P	$\min(D_1)$	$\min(D_2)$	...	$\min(D_{n-m+1})$
I	$j \mid \min(D_1) = d_{1,j}$	$j \mid \min(D_2) = d_{2,j}$	...	$j \mid \min(D_1) = d_{n-m+1,j}$

Parallelization

- SCRIMP is highly parallelizable (no calculus dependency between diagonals)

- However, elements inside diagonals need results from the previous step

- Two possible computation approaches:
  - **Random order for the diagonals**
    - Benefit: allows the possibility of obtaining partial (maybe enough accurate) results if the program is interrupted
    - Drawback: less performance if complete solution needed
  - **Sequential order for the diagonals**
    - Benefit: better performance in complete solution
    - Drawback: if the program is interrupted, only part of the time series is explored

Code

- SCRIMP FF code transformation examples

Original Code

```
for (int w = 0; w < win; w++)
{
     lastz += tSeries[w + subseq]
           * tSeries[w];
}
```

FlexFloat code

```
for (int w = 0; w < win; w++)
{
    ff_fma(&lastz,  &tSeries[w + subseq],
          &tSeries[w], &lastz);
}
```

```
distance = 2 * (windowSize
    - (lastz - windowSize *
     AMean[j] * AMean[i]) /
    (ASigma[j] * ASigma[i]));
```

```
ff_mul(&sigma_prods, &ASigma[subseq], &ASigma[0]);
ff_mul(&mean_prods,  &AMean[subseq],  &AMean [0]);
ff_cast(&mean_cast,  &mean_prods,
           (flexfloat_desc_t) {dist_exp, dist_man});
ff_cast(&sigma_cast, &sigma_prods,
           (flexfloat_desc_t){dist_exp, dist_man});
ff_mul(&distance, &mean_cast,  &windowSize);
ff_sub(&distance, &lastz_cast, &distance);
ff_div(&distance, &distance,   &sigma_cast);
ff_sub(&distance, &windowSize, &distance);
ff_mul(&distance, &distance,   &constant_2);
```