Keldysh Institute • Publication search

Keldysh Institute preprints • Preprint No. 124, 2017

ISSN 2071-2898 (Print)
ISSN 2071-2901 (Online)

Kislitsyn A.A., Kozlova A.B., Masherov E.L., Orlov Y.N.

Numerical Algorithm for Self-consistent Stationary Level for Multidimensional Non-stationary Time-series

Recommended form of bibliographic references: Kislitsyn A.A., Kozlova A.B., Masherov E.L., Orlov Y.N. Numerical Algorithm for Self-consistent Stationary Level for Multidimensional Non-stationary Time-series // Keldysh Institute Preprints. 2017. No. 124. 14 p. doi:10.20948/prepr-2017-124-e
URL: http://library.keldysh.ru/preprint.asp?id=2017-124&lg=e

Ордена Ленина
ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ
имени М.В.Келдыша
Российской академии наук

A.A. Kislitsyn, A.B. Kozlova,
E.L. Masherov, Yu.N. Orlov

Numerical Algorithm for Self-consistent Stationary Level for Multidimensional Non-stationary Time-series

Москва — 2017

Кислицын А.А., Козлова А.Б., Машеров Е.Л., Орлов Ю.Н.

Алгоритм вычисления согласованного уровня стационарности для многомерных временных рядов

В работе описывается алгоритм построения статистики, называемой согласованным уровнем стационарности, в случае многомерных нестационарных временных рядов. Практическая цель применения этой статистики – в построении индикатора разладки для нестационарных выборочных функций распределения. Отличие от классической задачи, сводящейся фактически к тесту двух выборок на стационарность, состоит в том, что надо обнаружить переход из одного нестационарного режима в другой. Алгоритм тестируется на данных электроэнцефалограмм для распознавания приступа эпилепсии.

Ключевые слова: согласованный уровень нестационарности, индикатор разладки, электроэнцефалограмма, приступ эпилепсии

Kislitsyn A.A., Kozlova A.B., Masherov E.L., Orlov Yu.N.

Numerical Algorithm for Self-consistent Stationary Level for Multidimensional Non-stationary Time-series

In this paper we consider the self-consistent stationary level of electroencephalogram time series. The practical purpose of this statistics is to construct the disorder indicator. Unlike the classical problem of stationary test of two samples, in our case one should construct an indicator to predict the change in the nonstationary regime. For example, we consider special predictor of an attack of epilepsy.

Key words: non-stationary index, disorder indicator, electroencephalogram, epilepsy attack

Работа выполнена при поддержке гранта РНФ, проект

№ 14-21-00025

Contents

1. Self-consistent stationary level

This paper is devoted to the problem of numerical algorithm construction for modeling of disorder indicator of non-stationary multi-dimensional time-series. The central point of theoretical aspect of the model is the so-called self-consistent stationary level (SCSL), introduced in [1, 2]. In one-dimensional case there is a software [3, 4] for calculating SCSL in the frame of the problem of generation ensemble of trajectories. But it is not a convenient instrument to disorder analyze. We want to construct indicator-predictor for disorder in non-stationary sample distribution function.

As it is well-known [5, 6], the problem of a sample fit can be solved with the use of non-parametric statistics of Kolmogorov-Smirnov test

$$S_N = \sup_x \left| F_{1,N}(x) - F_{2,N}(x) \right|, \tag{1}$$

for which the following asymptotic representation is valid:

$$\lim_{N\to\infty} P\left\{ 0 < \sqrt{\frac{N}{2}}\, S_N < z \right\} = K(z), \tag{2}$$

where $K(z)$ is a tabulated Kolmogorov function (see e.g. [7]) and N is a sample length. The value of $F_N(x)$ is a sample distribution function of a random value of ξ with realization values x in a sample window of length N. In formula (2) the significance level approximately is taken to be equal to $1-K(z)$. Let us consider a self-consistent significance level for the sample of N length, defined by formula

$$1 - K\left(\sqrt{\frac{N}{2}}\varepsilon \right) = \varepsilon . \tag{3}$$

The solution $\varepsilon = \varepsilon^*(N)$ of this equation (3) was tabulated in [8] (see fig. 1).

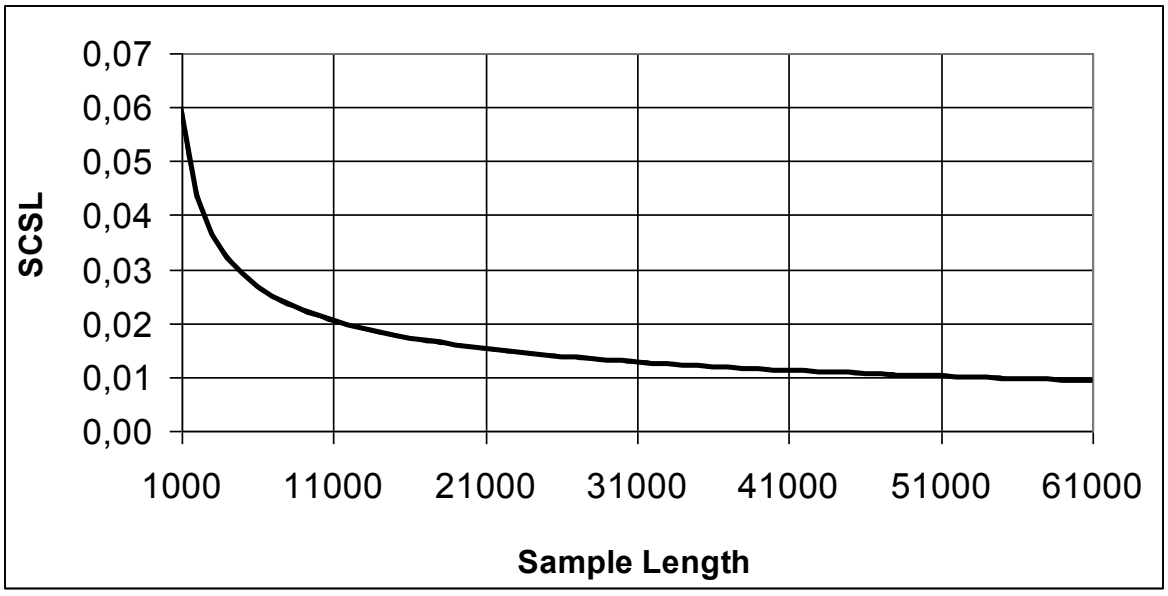

Fig. 1. SCSL for stationary case

According to non-parametric statistics (3) in stationary case the significance level $\varepsilon^*(N)$ for samples with length N is equal to probability, complementing (1) to unit. In general non-stationary case we can construct the empirical distribution function $G_N(\rho)$ for the distances $\rho(N)$ between two independent samples with length N:

$$\rho(N) = \left\| F_{1,N}(x) - F_{2,N}(x) \right\|_C . \quad (4)$$

The numerical solution with respect to ρ gives us self-consistent stationary level from the following equation:

$$G_N(\rho) = 1 - \rho . \quad (5)$$

Let us designate this level SCSL as $\rho^*(N)$. If it appears, that $\rho^*(N) > \varepsilon^*(N)$, then the series with analyzing sample distribution functions are non-stationary. The value of $\rho^*(N)$ is the correct significance level for statistical hypothesis about properties of these time-series samples.

We introduce also the value of Non-Stationary Index (NSI) as a ratio $\rho^*(N)$ and $\varepsilon^*(N)$:

$$J(N) = \frac{\rho^*(N)}{\varepsilon^*(N)} . \quad (6)$$

In practice we need to calculate the value of NSI for multi-dimensional time-series in a real time. The corresponding software is described below.

2. Numerical algorithm

Numerical algorithm is realized with the use of C++ in MSVS 2017. The data must be presented in ASCII format.

Step 1. The data reading and converting.

If initial data are presented in another format, they must be transformed to appropriate format. E.g. for EDF format the data reading is carried out with the use of initial code of the Library Module EDFbrowser to convert the data into ASCII format (github.com/Teuniz/EDFbrowser).

One should point the dimension of time-series (or the number of assigns), the volume of the main sample, the volume of tested sample (TailLength) and main sample step (StepLength).

So the file under analyze must be formatted as follows (fig. 2). The first coulomb consists of time moments or data consequent numbers and other coulombs correspond to time-series itself.

```
Time,          1,           2,           3,           4,           5,
4113.000000,  -216.831233,  -237.075880,  -291.786622,  -1878.398126, -349.676275,
4113.001953,  9.540735,     85.332679,    23.929492,    -5466.552753, -74.114833,
4113.003906,  155.770668,   330.276183,   269.374929,   -5482.447311, 140.043421,
4113.005859,  115.448684,   347.174608,   287.946465,   -5482.447311, 146.401244,
4113.007812,  38.987495,    277.740486,   217.675788,    -5482.447311, 81.651834,
4113.009766,  -69.430122,   131.510553,   78.305612,     -5482.447311, -47.846985,
4113.011719,  -296.136711,  -192.738429,  -239.920169,   -5482.447311, -334.618272,
4113.013672,  -463.113225,  -460.938180,  -512.804632,   -1837.741520, -565.340329,
4113.015625,  -503.435209,  -563.165284,  -624.735783,   212.656454,   -648.995897,
4113.017578,  -441.195466,  -516.652789,  -574.877064,   189.232895,   -602.483401,
4113.019531,  -260.164817,  -297.307889,  -362.893854,   -1157.287129, -418.608463,
```

Fig. 2. A fragment of data set

As a result we shall be given a number of vectors VSet for each assign and a vector vTime for corresponding time moments.

Step 2. The data normalizing.

For each i-th assign the initial data $b_{ik} = b_i(t_k)$ are transformed to unit interval by formula

$$x_{ik} = \frac{b_{ik} + \left|\min\limits_k b_{ik}\right|}{\left|\min\limits_k b_{ik}\right| + \left|\max\limits_k b_{ik}\right|}. \qquad (7)$$

For the purpose of analysis of indicator efficiency we separate the total data volume into two part: main and tested. For EEG data the main part consists of $2 \cdot 10^6$ data and tested part just before the attack with the length $6 \cdot 10^4$ data.

Step 3. The construction of distribution function of distances between sample distribution functions (SDF) in C norm.

Step 4. Determination of the solution of equation (5).

Step 5. Formation of output data.

Now we present corresponding numerical code and block-scheme in fig. 3 below. The numerical algorithm is described according to blocks designation.

Input Data & Initial Parameters

```
const double DATA_LENGTH_const = 2000000;
const double Tail_const = 2000000 - DATA_LENGTH_const;
const double StepLength_const = 1000;
const double SegmentAmount_const = 2000;
const int ELOCTROD_AMOUNT = 64;
vector <vector <double>> AllDataSet;
vector <double> TimeSet;
int readfile(double DATA_LENGTH)
{
        char* file_name = "data.txt";
        ifstream fs(file_name, ios::in);
```

```
        while (fs)
        {
                char c;
                vector <double> DataRow(ELOCTROD_AMOUNT, 0.0f);
                fs >> time;
                TimeSet.push_back(time);
```

Data set formation

```
                for (int i = 0; i <ELOCTROD_AMOUNT; i++)
                {
                        fs >> DataRow[i];
                }
                AllDataSet.push_back(DataRow);
        };
        fs.close();
        return 0;
};
```

Data Normalization

```
int SetNormalizer(vector <vector <double>> &VectorSet)
{
        vector <double> maxSet(ELOCTROD_AMOUNT, 0);
        vector <double> minSet(ELOCTROD_AMOUNT, 0);
                for (int i = 0; i < VectorSet.size(); i++)
                for (int j = 0; j < ELOCTROD_AMOUNT; j++)
                {
                        if (VectorSet[i][j] > maxSet[j]) maxSet[j] = VectorSet[i][j];
                        if (VectorSet[i][j] < minSet[j]) minSet[j] = VectorSet[i][j];
                }
        for (int i = 0; i < VectorSet.size(); i++)
        {
                for (int j = 0; j < ELOCTROD_AMOUNT; j++)
                {

                        VectorSet[i][j] = VectorSet[i][j] + fabs(minSet[j]);
                        VectorSet[i][j] = VectorSet[i][j] / (fabs(minSet[j]) +
fabs(maxSet[j]));
                }
        }
        return 0;
}
```

Computing of cumulative distribution function

```
vector <double> distrib_data_func(vector <double> &Data, int segmentNum, vector
<double> &ExpData)
{
        vector <double> counts(StepLength);
        int counts_i, counts_j, rec;
```

```
	double start_elem = (segmentNum)*StepLength, end_elem = (segmentNum + StepLength);
	for (int i = start_elem; i < end_elem; i++)
	{
		for (int j = start_elem; j < end_elem; j++)
		{
			if (Data[j] < Data[i])
			{
				counts[counts_i] = counts[counts_i] + 1;
			}
		}
		counts_i++;
	}
	for (int counts_i = 0; counts_i < counts.size(); counts_i++)
		for (int counts_j = 0; counts_j < counts.size(); counts_j++)
		{
			if (counts[counts_i] <counts[counts_j])
			{
				rec = counts[counts_i];
				counts[counts_i] = counts[counts_j];
				counts[counts_j] = rec;
			}
		}
	for (int counts_i = 0; counts_i < counts.size(); counts_i++)
		counts[counts_i] = counts[counts_i] / StepLength;
	return counts;
}
```

The distance between sample distribution functions

```
int main()
{
	int	start_index = Tail_const,
		end_index = 2060000,
		main_length = DATA_LENGTH,
		TWOmin_length = 60000,
		Tail_Seg_Amount = TWOmin_length / StepLength_const;
	int sort;
	double rec;
	double rec_2;
	for (int rec_ind = 0; rec_ind < Iterations_for_one_file; rec_ind++)
	{
		vector <double> sample_distance;
		vector <double> distrib_distance;
		vector <double> testMainV;
		vector <double> testTailV;
		ostringstream strs;
```

```
            strs << Array_Data_size[0][rec_ind];
            string str = strs.str();
            str = str.insert(str.length(), ".txt");
            ofstream fs33(str, ios_base::app);
            for (int iter_main_index = 0; iter_main_index < ELECTROD_AMOUNT; iter_main_index++)
            {
                  DATA_LENGTH = Array_Data_size[1][rec_ind];
                  StepLength = Array_Data_size[0][rec_ind];
                  SegmentAmount = (Array_Data_size[1][rec_ind] / Array_Data_size[0][rec_ind]);
                  main_length = Array_Data_size[1][rec_ind];
                  TWOmin_length = 60000;
                  Tail_Seg_Amount = TWOmin_length / StepLength_const;
                  start_index = 2000000 - Array_Data_size[1][rec_ind];
```

The data splitting into the main and test parts

```
                  for (int i = start_index; i <= end_index; i++)
                  {
                        testMainV.push_back(AllDataSet[i][iter_main_index]);
                  }
                  if (end_index != 0)
                  {
                        for (int i = end_index + 1; i <= AllDataSet.size() - 1; i++)
                        {
      testTailV.push_back(AllDataSet[i][iter_main_index]);
                        }
                  }
```

Determining the distance between two nearest distribution functions (distance_vector_forming)

```
sample_distance = distance_vector_forming(testMainV, SegmentAmount);
```

Computing of cumulative distribution function for g_n(Rho)

```
                  distrib_distance = distribFunction(sample_distance);
                  for (int i = 0; i < sample_distance.size(); i++)
                        for (int j = 0; j < sample_distance.size(); j++)
                        {
                              if (sample_distance[i] < sample_distance[j])
                              {
                                    rec = sample_distance[i];
                                    sample_distance[i] = sample_distance[j];
                                    sample_distance[j] = rec;
                              }
                              if (distrib_distance[i] < distrib_distance[j])
                              {
                                    rec = distrib_distance[i];
                                    distrib_distance[i] = distrib_distance[j];
```

```
                                        distrib_distance[j] = rec;
                                }
                        }
```

Solving the equation for Rho

```
                        while (sample_distance[ind_i] + distrib_distance[ind_i] < 1)
                        if (sample_distance[ind_i] + distrib_distance[ind_i] == 1)  rec = distrib_distance[ind_i];
                        else
                        {
                                double X1, X2, X3, X4, Y1, Y2, Y3, Y4;
                                X1 = distrib_distance[ind_i];
                                X2 = distrib_distance[ind_i + 1];
                                Y1 = sample_distance[ind_i];
                                Y2 = sample_distance[ind_i + 1];
                                X3 = 1 - Y1;
                                X4 = 1 - Y2;
                                Y3 = 1 - X1;
                                Y4 = 1 - X2;
                                rec = (X1*Y2 - Y1*X2)*(X3 - X4) - (X1 - X2)*(X3*Y4 - Y3*X4);
                                rec = rec / ((X1 - X2)*(Y3 - Y4) - (Y1 - Y2)*(X3 - X4));
                                rec = 1 - rec;
                        }
                        ind_i++;
```

Output statistics: Rho&NSI

```
                        sample_distance.clear();
                        distrib_distance.clear();
                        vector <double> tail_distance;
                        SegmentAmount = Tail_Seg_Amount;
                        StepLength = Tail_Seg_Length;
                        double rhoBig = rec;
                        tail_distance = distance_vector_forming(testTailV, SegmentAmount);
                        for (int i_ind = 0; i_ind < tail_distance.size(); i_ind++)
                        {
                                if (tail_distance[i_ind] > rhoBig) { rec_2 = rec_2 + 1; }
                        }
                        rec_2 = rec_2 / tail_distance.size();
        return 0;
}
```

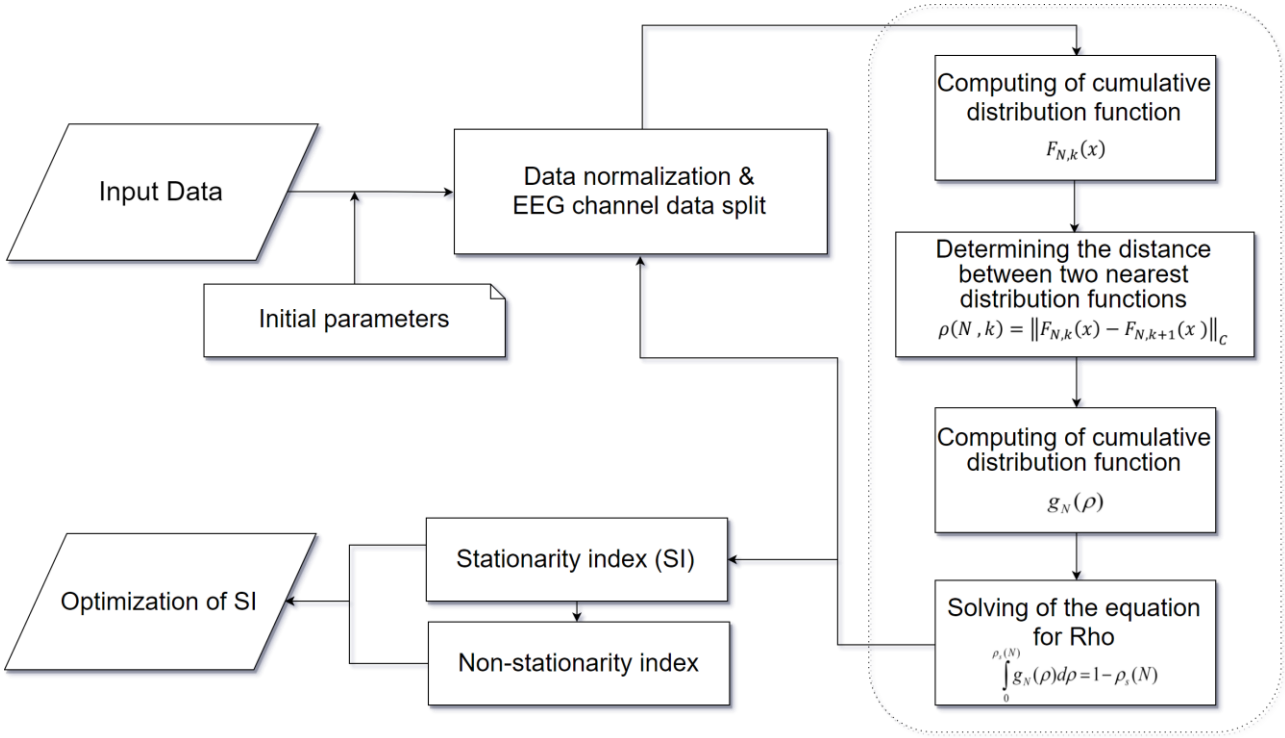

Fig. 3. The Scheme of SCSL and NSI calculation algorithm

3. EEG data

We apply further this algorithm to disorder indicator construction for EEG time-series of epilepsy attack. The data were accumulated by scientific personal of Burdenko Institute. We consider several data sets for patients during 1 hour before epilepsy attack. For each case we analyzed 32 assigns with frequency of data reading 512 Hz. The typical fragment of data is presented in fig. 4.

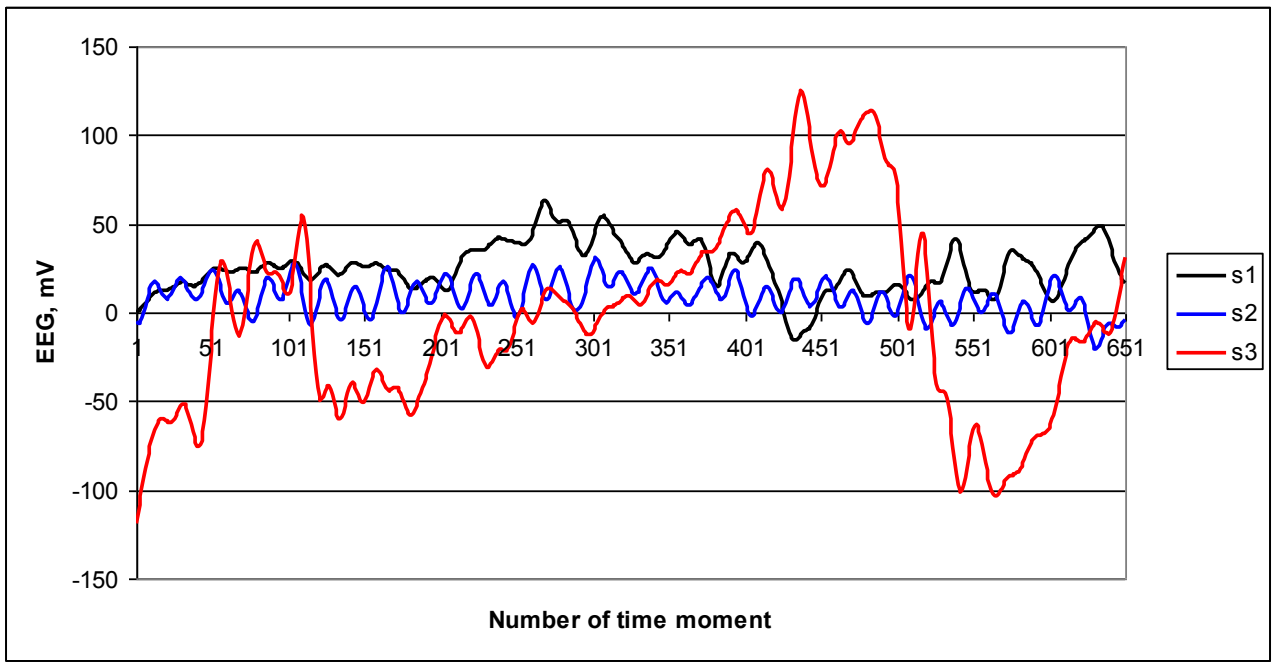

Fig. 4. Typical behaviour of EEG synchronous signals by several assigns

It appears, that signals from various assigns are not coherent. So they can be studied separately.

One should emphasize, that traditional method of correlation analysis and power spectrum estimation [8] is correct for stationary process only. So firstly we analyze Kolmogorov-Smirnov statistics (4) depending on data length and use indicator (6) to estimate the stationary level of these time-series.

The typical dependences of SCSL and NSI from the sample length of the main part of data set are presented in fig. 5 and 6. From these figures it follows, that series are non-stationary. The value of SCSL gives us a typical accuracy of statistical estimation of sample identity.

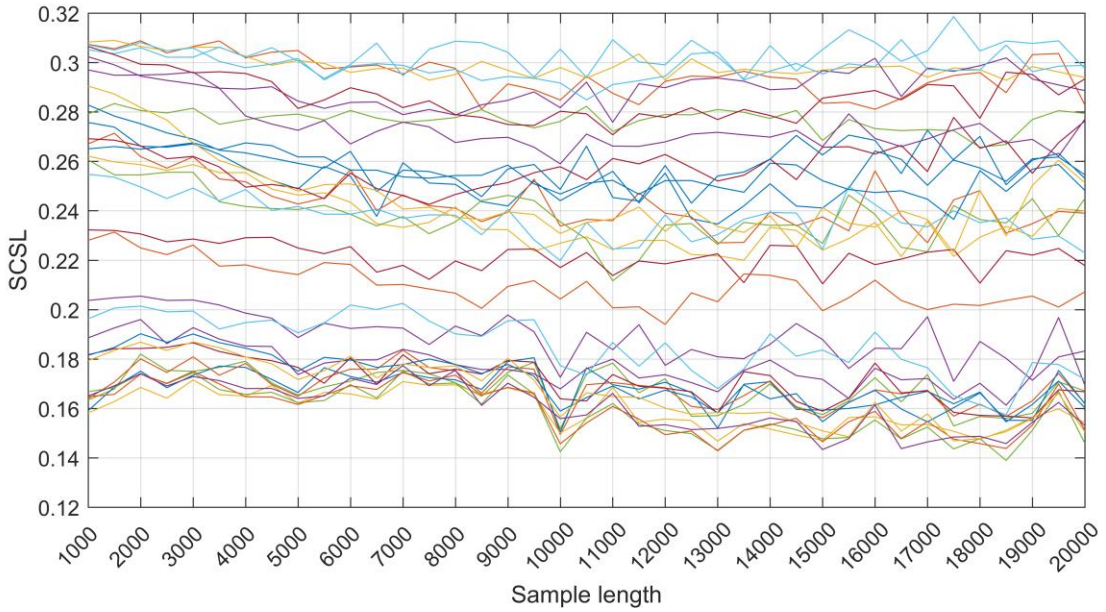

Fig. 5. Dependence of SCSL from the sample length

From the next fig. 6 one can see, that stabilization of Non-Stationary Index occurs approximately at the sample length more than 12000 data points (24-25 seconds). The value of NSI sufficiently more, than unit, so the EEG time-series are non-stationary.

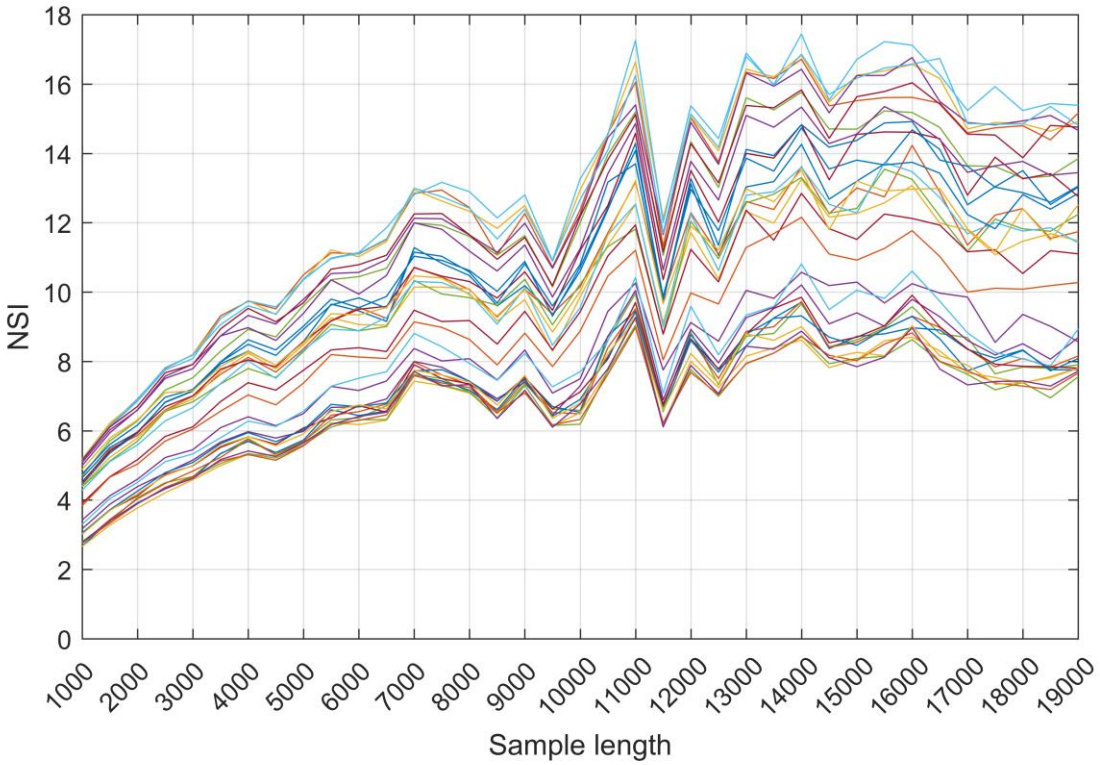

Fig. 6. Dependence of NSI from the sample length

4. Efficiency of disorder predictor

Because of non-stationary character of EEG time-series the indicator of deviation from stationary state is not an indicator of disorder. We have to compare several non-stationary cases with each other or one sample from the first non-stationary distribution function and the sample with the same length, following just after the previous sample, from the second distribution function. Our new indicator of disorder is constructed in a similar way [1, 2]. We suppose, that there is a normal non-stationary level of EEG, which can be determined from the main part of data set. And just before epilepsy attack the deviation of SCSL of some optimal sample volume in a small but statistical sufficient time period from SCSL in normal state can be treated as predictor of disorder.

We define the value of predictor index K as a ratio of SCSL of tested part and the main part of data set:

$$K = \frac{J_{test}}{J_{main}}. \tag{8}$$

Here J is defined in (6).

The value of (8) was calculated for all assigns. It was found, that there are several assigns (consisting approximately 30 % from the total number of electrodes), which have reliable exceeding the value of J_{main}. Corresponding results for 10 electrodes are presented in fig. 7.

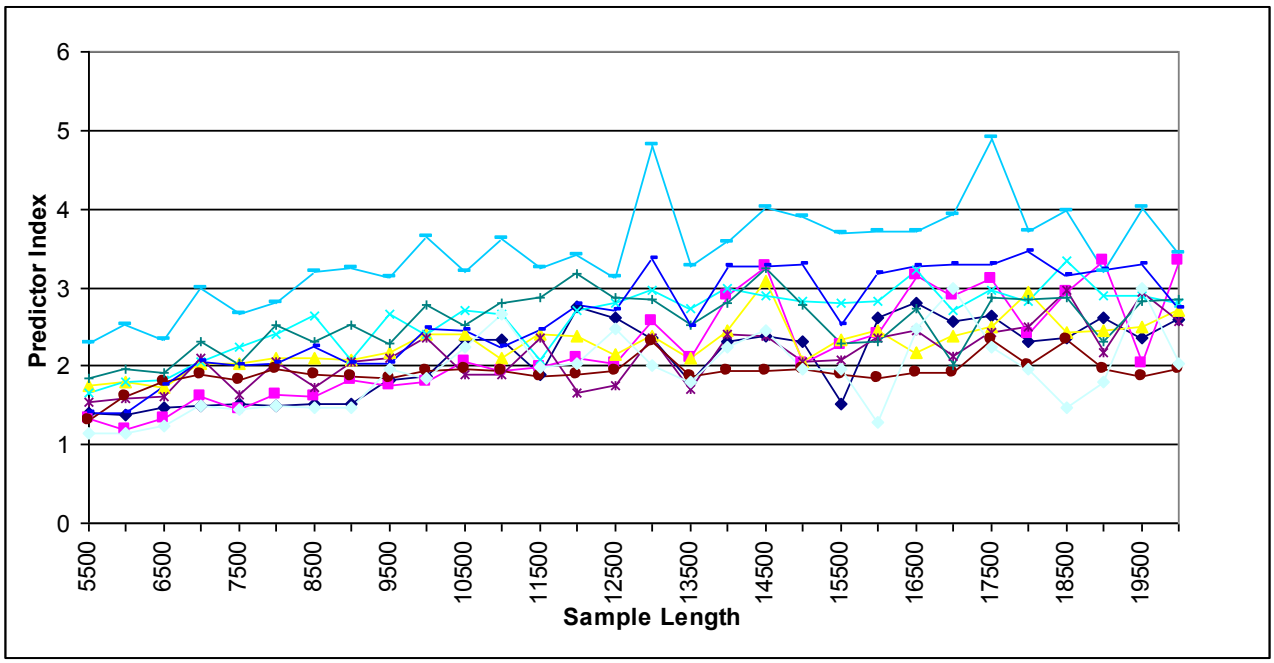

Fig. 7. Dependence of Predictor Index from the sample length

One can see, that predictor index grows from increasing of sample length. But it is interesting to obtain minimal length to reliable prediction of disorder (see fig. 8).

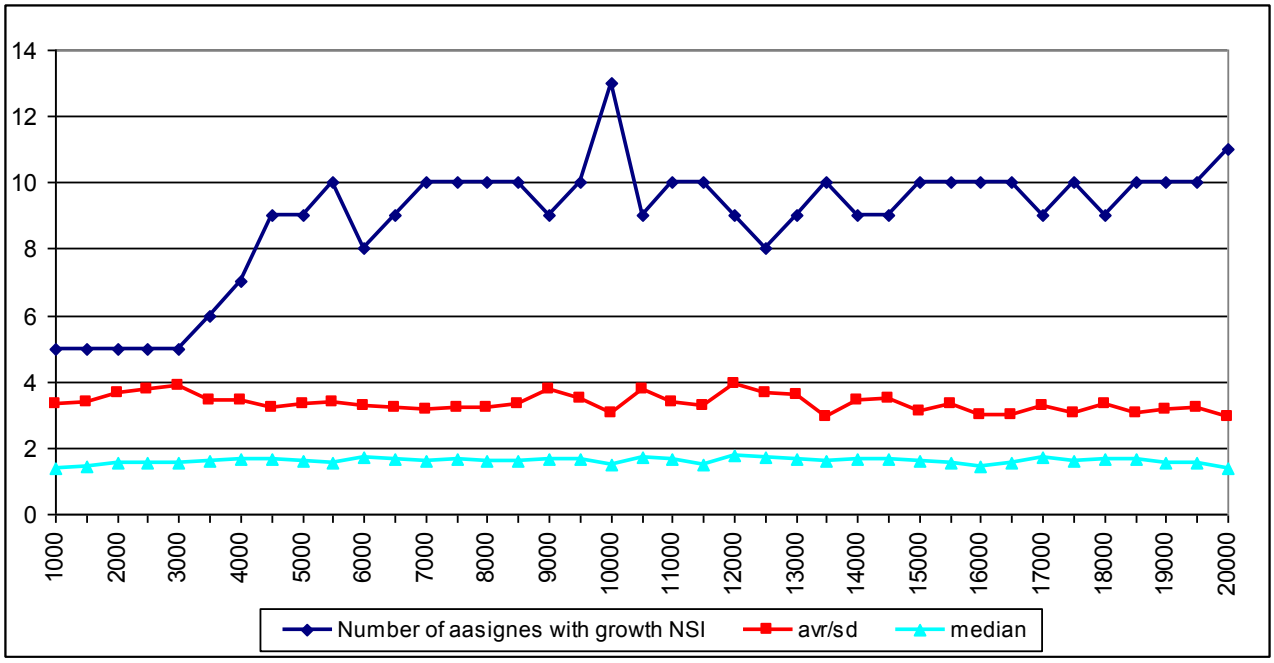

Fig. 8. Statistic properties of Predictor Index depending on the sample length

From the results presented above it follows, that sufficient number of assigns for indication of disorder can be obtained in the window more then 5500 data. The stability of the deviation *K* is approximately constant (the value of ratio between average deviation value over this amount of assigns and corresponding standard deviation). The value of median of deviation *K* is also approximately equal to constant. So the time period of 11-12 seconds is minimal for disorder indication.

Conclusion

In this work we demonstrated the efficiency of new statistical method of disorder indication – the so-called self-consistent stationary level deviation. The principal aspect of this method is the analysis of distribution function of distances between the sample distribution functions in C-norm. Multidimensional time-series is a natural object for testing of non-stationary random process analytical framework, one block of which was presented in this paper.

EEG data set may allows sufficiently complete information for construction various indicators: patterns of the typical quasi-stationary states, patterns of corresponding distribution functions, a lot of materials for testing various stochastic control operators and also a lot of problems with construction of disorder indicators.

The aim of our future researches is to construct software specifically for medicine problems in the frame of statistical analysis of electroencephalography. The application of new non-stationary statistics, presented in this paper, may be rather fruitful.

Cited literature

1. Orlov Yu.N. Kineticheskie metody issledovanija nestatsionarnykh vremennykh riadov (in Russian). – Moscow: MIPT. 2014.

2. Fedorov S.L., Orlov Yu.N. Metody chislennogo modelirovaniya processov nestatsionarnogo sluchajnogo bluzhdaniya (in Russian). – Moscow: MIPT. 2016.

3. Fedorov S.L., Orlov Yu.N. Distribution functional modeling on the ensemble of trajectories of non-stationary random process (in Russian) // Keldysh Institute Preprints. 2016. № 101. 14 p.

4. Fedorov S.L., Orlov Yu.N. Software for statistical analysis and modeling of non-stationary time-series / State Registration of Software № 2017619117 from 15.08.2017.

5. Justel A., Pena D., Zamar R. A multivariate Kolmogorov – Smirnov test of goodness of fit. // Statistics & Probability Letters, 1997. Vol. 35, pp. 251–259.

6. Drew J.H., Glen A.G. and Leemis L.M. Computing the cumulative distribution function of the Kolmogorov – Smirnov statistic. // Computational Statistics and Data Analysis, 2000. Vol. 34, pp. 1-15.

7. Durbin J. Distribution Theory for Tests Based on The Sample Distribution Function. – Society for Industrial & Applied Mathematics, Philadelphia, 1972.

8. Schomer D.H., Lopes da Silva F.H. Nidermeyer's Electroencephalography. – Philadelphia, Lippincott Williams & Wilkins, 2011. – 1296 p.