Partial Parallelization of the Successive Projections Algorithm using Compute Unified Device Architecture

Lauro Cássio Martins de Paula,
Anderson da Silva Soares,
Telma Woerle de Lima,
Wellington Santos Martins
Institute of Informatics
Federal University of Goiás
Goiânia, Brazil

Arlindo Rodrigues Galvão Filho
Department of System and Control
Technological Institute of Aeronautics
São José dos Campos, Brazil

Clarimar José Coelho
Computer Science Departament
Pontifical University Catholic of Goiás
Goiânia, Brazil

***Abstract*—This paper proposes a partial parallelization for the Successive Projections Algorithm (SPA), which is a variable selection technique designed for use with Multiple Linear Regression. This implementation is aimed at improving the computational efficiency of SPA, without changing the outcome of the algorithm. For this purpose, a new strategy of inverse matrix calculation is employed. The advantage of the proposed implementation is demonstrated in an example involving large matrixes. In this example, gains of speedup were obtained.**

Keywords: Successive Projections Algorithm, parallelization, Multiple Linear Regression, CUDA.

I. Introduction

The Successive Projections Algorithm (SPA) is a technique that aims at selecting variables to minimize collinearity problems in Multiple Linear Regression (MLR). Originally proposed in [9], the SPA has the restriction that the variable incorporated in each iteration must be as less multicolinear possible with the previously selected variables [11], [16], [18]. Through the use of SPA, it is possible to obtain good results in various problems of multivariate analysis, such as determining sulfur in diesel samples [2], determining the quality parameters in vegetable oils [15], determining the levels of moisture and protein in wheat samples [12], among others. The SPA is composed of three stages. In phase 1 are generated chains of minimally redundant variables. Phase 2 evaluates the subsets of variables with higher predictive potential from the variable chains obtained in stage 1. Such assessment is measured by the prediction error in the multiple linear regression models. The equation 1 shows how regression coefficients are calculated, and the equation 2 shows how the predictive ability of a particular subset of variables is measured by calculating the error $RMSEP$ (Root Mean Square Error).

$$\beta = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}, \tag{1}$$

where $\mathbf{X}$ is the matrix of variables and samples, $\mathbf{y}$ is the vector of dependent variables and β is the vector of regression coefficients.

$$RMSEP = \sqrt{\frac{\sum_{i=0}^{N}(y_i - \hat{y}_i)^2}{N}}, \tag{2}$$

where $\hat{y}$ is the estimated value and y is the actual value of the property of interest.

Regarding the computational cost, phase 2 represents the highest cost compared to the other phases, because this stage involves the calculation of an inverse matrix as shown in equation 1. In [7] was proposed to reduce the cost of Phase 2 of the SPA through sequential regressions. This idea is based on updating the calculation of the inverse of the linear regression when adding a new variable instead of performing any inverse calculation. The benefit of the proposed implementation was shown through an example involving a data set near-infrared (NIR) of wheat samples. In such example, computational gains were achieved compared with the traditional SPA implementation. Despite the results obtained, such technique does not exploit the recent advances in computing power of computers, in particular the possibility of performing tasks in parallel, since sequential regressions have a sequential formulation.

In [17] was proposed the SPA parallelization in order to explore the ability of multiple processing cores (multicore) on new computer architectures. The results obtained showed that it was possible to reduce the computational cost of the algorithm as more than one processing core becomes available. However, this processing architecture currently is limited to using a maximum of eight cores.

Despite the use of sequential regressions and multicore parallelization reducing the computational time, both strategies do not make use of the latest advances in terms of processing capacity in architecture computers $Intel^{®}$. Calculating the inverse matrices using parallel programming can be more interesting due to the fact of using the parallel computing resources provided by GPUs (Graphics Processing Unit) [3], [1], [10], [20].

In this work, a new strategy for reducing the computational

cost of the SPA is proposed. In particular a partial parallelizing of phase 2 of the algorithm is proposed, involving calculation of matrix inversion by using the Compute Unified Device Architecture (CUDA) on GPUs. While the current multicore architectures have two, four or eight cores, GPUs have hundreds or even thousands of processing cores. However, unlike the parallelization on CPUs (Central Processing Unit) multicore, the organization and number of threads, which are executed independently on the GPU cores, are managed manually by the programmer.

This article is organized as follows. Section 2 details the Successive Projections Algorithm. The proposed parallelization on phase 2 of the SPA is detailed in section 3. Section 4 describes the materials and methods used. The results are discussed in section 5. Section 6 shows the conclusions.

II. Successive Projections Algorithm Review

The multivariate calibration refers to obtaining a mathematical model that allows to provide the value of a quantity y based on values measured from a set of explanatory variables x_1, x_2, ..., x_k [12]. Thus, it is possible to obtain a suitable model

$$y = \beta_0 + \beta_1 x_1 + ... + \beta_k x_k + \varepsilon, \quad (3)$$

where β_0, β_1, ..., β_k, k = 1, 2, ..., K, are the coefficients to be determined, and ε is a portion of random error. The process for obtaining coefficients is also known as MLR, typically being performed by the least squares method [8]. The Multiple Linear Regression is a statistical technique used to build models that describe reasonably relationships between several explanatory variables of a given process [14], [4].

The goal of SPA is to select a subset of variables with low collinearity that allows the construction of a MLR model with a capacity of adequate prediction. Data modeling for the SPA implementation are divided into two sets: calibration, containing N_c observations, and validation containing N_v observations, where N_c + N_v = N. The data calibration and validation are arranged respectively in matrixes X_c ($N_c \times K$) and X_v ($N_v \times K$). In SPA's phase 1 are generated K chains with M variables each, being

$$M = min(N_c - 1, K). \quad (4)$$

In step 2, the SPA uses the validation set to evaluate subsets of variables extracted from the chains generated in stage 1. As a result of phase 2, the best subset of variables is the one that leads to the smallest value of $RMSEP$ among subsets tested. The algorithm of phase 2 is shown in the algorithm 1.

Obtaining $RMSEP$ can be performed in two ways:

- If validation is used to test series, a set of validation samples must be defined. Soon after, the best subset is determined by the lowest root value of mean squared error on a validation set calculated by the equation 2 for all subsets of variables;

Algorithm 1: Step 2 of the SPA

1) Do $k = 1$
2) While $k < K$
3) Do $m = 1$
4) While $m < M$
5) Let $\mathcal{X}_{km}$ be a subset of varibles formed by m first elements of k-th chain generated on phase 1.
6) Let $\mathbf{S}_{km}^{-1}$ be the inverse of the equation 1.
7) Using the variables contained in $\mathcal{X}_{km}$, calculate the inverse $\mathbf{S}_{km}^{-1}$ and subsequently the remainder of the equation 1.
8) Calculate the error $RMSEP$ of k-th chain with m variables, according to equation 2.
9) Do $m = m + 1$
10) End While $m < M$
11) Do $k = k + 1$
12) End While $k < K$

- If the cross-validation is used, the best subset is determined by the lowest root value of mean squared error of cross validation in the calibration set, which can be obtained by an equation similar to equation 2.

The third and last phase consists of eliminating variables that do not contribute significantly to the predictive capacity of the resulting MLR model. For such, each variable selected in phase 2 is associated with a "relevancy index" given by the product of the sample standard deviation and the regression coefficient modulus of this variable [18].

III. Proposed Parallelization

A square matrix A is said to be invertible if there exists another matrix A^{-1} such that $A^{-1}A$ = I and AA^{-1} = I, where I is called identity matrix. According to the literature, A matrix has an inverse if and only if $det(A) \neq 0$.

Calculating the inverse of a matrix can require significant computational effort, especially when A is large. Therefore, using the parallel computing resources provided by a GPU can be viable. The GPU was initially developed as a driven-flow technology, optimized for calculations of intensive data use, where many identical operations can be accomplished in parallel on different data. Unlike a multicore CPU, which normally executes some threads in parallel, the GPU was designed to run thousands of threads in parallel [5].

Programming models such as CUDA [5] and OpenCL [19], allows that applications can be run more easily on the GPU. CUDA was the first architecture and interface for programming application (API), created by $NVIDIA^{®}$ in 2006 to allow the GPU could be used for a wide variety of applications. Like any technology, the GPU has its limitations. Depending on the data volume, GPU's computational performance may prove inferior when compared to CPU performance. In this case, the data amount to be transferred to the GPU memory must be taken into account, because there is an overhead associated with the parallelization of tasks on the GPU [13]. Factors regarding the access time to memory can also influence the computational performance. In other words, access to GPU global memory usually has a high latency and it is subject to a coalesced access to data in memory [6].

In this paper, a strategy for the parallelization at step of the calculation of inverse matrix used in phase 2 of the Successive Projections Algorithm is presented.

Let $A_{n\times n}$ and $I_{n\times n}$ be the matrix to be calculated the inverse and identity matrix, respectively. Recursively, i = 0, 1, ..., $n-1$, through the use of two kernel functions ($kernel1$ and $kernel2$), each thread performs an operation on each element of the matrix . In the first kernel function are set $\sqrt{n}$ blocks with $\sqrt{n}$ threads each, where each thread accesses a single element and divide it by the pivot of row i of the matrix A. In the second function are created n blocks with n threads each. Each block of threads handles a line of matrices, and only threads whose its global identifier (id) divided by the number of columns ($\frac{id}{n}$) is different from index i implementing operations. For example, in the first iteration (i = 0), threads with id = 0, 1, ..., $n-1$ do not satisfy the condition $\frac{id}{n} \neq i$. Only the threads that satisfy this condition continue its execution and, after all threads have been executed, the elements below the pivot of the first column are zeroed.

Figure 1 shows the strategy used. Each arrow in the figure represents an iteration of the algorithm. Initially, there are the matrixes $A_{3\times3}$ and $I_{3\times3}$. All operations applied to the matrix elements A are also applied in parallel to the elements of the matrix I. After the last iteration, the matrix A becomes the identity matrix, and the matrix I becomes A^{-1}.

$$A = \begin{vmatrix} 5 & 6 & 9 \\ 4 & 1 & 7 \\ 1 & 7 & 6 \end{vmatrix} \Rightarrow \begin{vmatrix} 1 & 1.2 & 1.8 \\ 0 & -3.8 & -0.2 \\ 0 & 5.8 & 4.2 \end{vmatrix} \Rightarrow \begin{vmatrix} 1 & 0 & 1.73 \\ 0 & 1 & 0.05 \\ 0 & 0 & 3.89 \end{vmatrix} \Rightarrow \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

$$I = \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix} \Rightarrow \begin{vmatrix} 0.2 & 0 & 0 \\ -0.8 & 1 & 0 \\ -0.2 & 0 & 1 \end{vmatrix} \Rightarrow \begin{vmatrix} -0.05 & 0.31 & 0 \\ 0.21 & -0.26 & 0 \\ -1.42 & 1.52 & 1 \end{vmatrix} \Rightarrow \begin{vmatrix} 0.58 & -0.36 & -0.44 \\ 0.22 & -0.28 & -0.01 \\ -0.36 & 0.39 & 0.25 \end{vmatrix}$$

Fig. 1. Parallelization strategy used in the calculation of the inverse of a 3x3 matrix.

Algorithms 2 and 3 respectively show the implementation of the functions $kernel1$ and $kernel2$.

Algorithm 2: $kernel1$ implementation.

```
begin
    Parameters: A, I, index, size = number of columns;

    id ← thread global identification
    if id < size then
        pivo ← A(index, index)

        A(index, id) ← A(index,id) / pivo

        I(index, id) ← I(index,id) / pivo
    end if
end
```

Algorithm 3: $kernel2$ implementation.

```
begin
    Parameters: A, I, index, n = number of columns,
    size = number of rows × number of columns;

    id ← thread global identification
    idBlock ← block identification
    idThread ← thread local identification
    if id < size then
        if (id/n) ≠ index then
            m ← A(id/n, index)

            A(idBlock, idThread) ← A(idBlock,
            idThread) - (m × A(index, idThread))

            I(idBlock, idThread) ← I(idBlock,
            idThread) - (m × I(index, idThread))
        end if
    end if
end
```

IV. Experimental

The dataset employed in this work consists of 775 NIR spectra of whole-kernel wheat, which were used as shoot-out data in the 2008 International Diffuse Reflectance Conference (http://www.idrc-chambersburg.org/shootout.html). Each spectrum comprises 1050 variables in the range 400-2500 nm. Protein content (%) was used as the y-property in the regression calculations.

A. Computational setup

All calculations were carried out by using a desktop computer with an Intel Core i7 2600 (3.40 GHz), 8 GB of RAM memory and a $NVIDIA^{®}$ GeForce GTX 550Ti graphics card with 192 CUDA cores and 2 GB of memory config. The Matlab 7.12.0 (R2011a) software platform was employed throughout. All the matrices used in this paper were generated by using $randn()$, which is a built-in function of Matlab.

V. Results and Discussion

Figure 2 presents the time required for completion of Phase 2 depending on the maximum number M of variables to be selected. For M = 100, for instance, regressions involving one up to 100 variables are carried out and matrix inversion in the same order (100x100). As can be observed, the computational time increases with the matrix size, but the increase is less pronounced if the parallel regression procedure is used. For M = 1000, for example, this procedure reduces the time by a factor of two. Although the computational gains obtained for the sizes of small matrices time required for execution of Phase 2 is lower by using CPU. This case can be observed in Figure 3. The implementation using GPU is more efficient for matrices from 300x300. Although the proposal is feasible for matrices larger than 300x300, we believe that the proposal

is important once the devices used in this type of problem have generated data with ever larger. Until five years ago the appliances generated matrixes with few hundreds of variables, while recently it is in the thousands. In this sense, the development of computational algorithms used in this type of problem is important to computational not to become unviable.

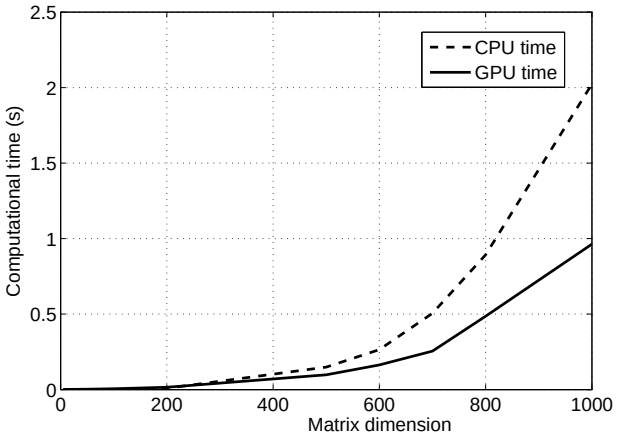

Fig. 2. Comparison of computational performance between CPU and GPU.

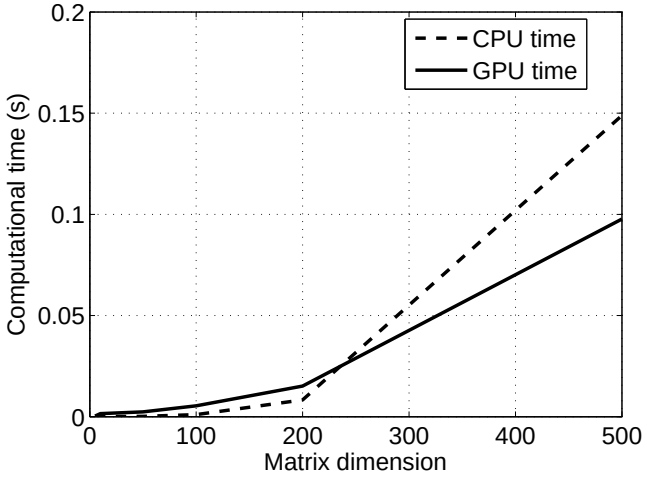

Fig. 3. Detail of Figure 2, showing a comparison of computational performance between CPU and GPU for matrix size up to 500x500.

VI. Conclusion

This paper proposed a partial parallelization of the Successive Projections Algorithm based on Compute Unified Device Architecture. This procedure was employed in Phase 2 of SPA, which is the computational bottleneck of the overall algorithm. The results obtained by using a large dataset of NIR spectra (775 samples and 1050 variables) revealed that substantial gains in computational efficiency can be obtained by using the proposed implementation.

Acknowledgment

The authors thank the research agencies CAPES and FAPEG for the support provided to this research.

References

[1] Nesrin Aydin Atasoy, Baha Sen, and Burhan Selcuk, *Using gauss - jordan elimination method with {CUDA} for linear circuit equation systems*, Procedia Technology **1** (2012), no. 0, 31 – 35.

[2] Mrcia C. Breitkreitz, Ivo M. Raimundo, Jr., Jarbas J. R. Rohwedder, Celio Pasquini, Heronides A. Dantas Filho, and Mrio C. U. Jos, Gledson E.and Arajo, *Determination of total sulfur in diesel fuel employing nir spectroscopy and multivariate calibration*, The Analyst **128** (2003), 1204–1207.

[3] Lau Mai Chan and Rajagopalan Srinivasan, *A graphic processing unit (gpu) algorithm for improved variable selection in multivariate process monitoring*, 11th International Symposium on Process Systems Engineering (Iftekhar A. Karimi and Rajagopalan Srinivasan, eds.), Computer Aided Chemical Engineering, vol. 31, Elsevier, 2012, pp. 1532 – 1536.

[4] J. Cortina, *Interaction, nonlinearity, and multicollinearity: Implications for multiple regression*, Journal of Management **19** (1993), no. 4, 915–922.

[5] NVIDIA CUDA, *Nvidia cuda c programming guide*, 4.0 ed., NVIDIA Corporation, 2701 San Tomas Expressway Santa Clara, CA 95050, 2011.

[6] NVIDIA $CUDA^{TM}$, *Nvidia cuda c programming best practices guide*, NVIDIA Corporation, 2701 San Tomas Expressway Santa Clara, CA 95050, 2009.

[7] Anderson da Silva Soares, Arlindo Rodrigues Galv ao Filho, Roberto Kawakami Harrop Galv ao, and Mário César Ugulino Araújo, *Improving the computational efficiency of the successive projections algorithm by using a sequential regression implementation: A case study involving nir spectrometric analysis of wheat samples*, Journal of the Brazilian Chemical Society **21** (2010), 760–763.

[8] Norman Richard Draper and Harry Smith, *Applied regression analysis*, 1998.

[9] Mario C. U. Araújo et al, *The successive projections algorithm for variable selection in spectroscopic multicomponent analysis*, Chemometrics and Intelligent Laboratory Systems (2001), 57–65.

[10] Fabio Fabris and Renato A. Krohling, *A co-evolutionary differential evolution algorithm for solving minmax optimization problems implemented on {GPU} using c-cuda*, Expert Systems with Applications **39** (2012), no. 12, 10324 – 10333.

[11] Roberto Kawakami Harrop Galvao, Mario Cesar Ugulino Araujo, Wallace Duarte Fragoso, Edvan Cirino Silva, Gledson Emidio Jose, Sofacles Figueredo Carreiro Soares, and Henrique Mohallem Paiva, *A variable elimination method to improve the parsimony of {MLR} models using the successive projections algorithm*, Chemometrics and Intelligent Laboratory Systems **92** (2008), no. 1, 83 – 91.

[12] Arlindo R. Galvao Filho, Roberto K. H. Galvao, and Mario Cesar U. Araujo, *Effect of the subsampling ratio in the application of subagging for multivariate calibration with the successive projections algorithm*, Journal of the Brazilian Chemical Society (en).

[13] David B. Kirk, *Nvidia cuda software and gpu parallel computing architecture*, NVIDIA Corporation, 2008.

[14] T. Naes and B. H. Mevik, *Understanding the collinearity problem in regression and discriminant analysis*, Journal of Chemometrics **15** (2001), no. 4, 413–426.

[15] Alessandra Félix Costa Pereira, Márcio Jose Coelho Pontes, Francisco Fernandes Gambarra Neto, Sergio Ricardo Bezerra Santos, Roberto Kawakami Harrop Galvao, and Mario Cesar Ugulino Araujo, *Nir spectrometric determination of quality parameters in vegetable oils using ipls and variable selection*, Food Research International **41** (2008), 341–348.

[16] Marcio Jose Coelho Pontes, Roberto Kawakami Harrop Galvao, Mario Cesar Ugulino Araujo, Pablo Nogueira Teles Moreira, Osmundo Dantas Pessoa Neto, Gledson Emidio Jose, and Teresa Cristina Bezerra Saldanha, *The successive projections algorithm for spectral variable selection in classification problems*, Chemometrics and Intelligent Laboratory Systems **78** (2005), no. 1, 11 – 18.

[17] Anderson da Silva Soares, Roberto K. H Galvao, Mario C. U. Araujo, S. F. C Soares, and Luiz Alberto Pinto, *Multi-core computation in chemometrics: case studies of voltammetric and NIR spectrometric analyses*, Journal of the Brazilian Chemical Society **21** (2010), 1626 – 1634 (en).

[18] Sófacles F. Soares, Adriano A. Gomes, Mario C. Araujo, Arlindo R. Filho, and Roberto K. Galvo, *The successive projections algorithm*, TrAC Trends in Analytical Chemistry **42** (2012), 94–98.

[19] Ryoji Tsuchiyama, Takashi Nakamura, Takuro Iizuka, Akihiro Asahara, Jeongdo Son, and Satoshi Miki, *The opencl programming book*, Fixstars, 2010.

[20] Ahmet Artu Yldirim and Cem Ozdogan, *Parallel wavelet-based clustering algorithm on {GPUs} using {CUDA}*, Procedia Computer Science **3** (2011), no. 0, 396 – 400.