4th GRACM Congress on Computational Mechanics
GRACM 2002
Patra, 27–29 June, 2002
©GRACM

A SPECTRAL VERSION OF PERRY'S CONJUGATE GRADIENT METHOD FOR NEURAL NETWORK TRAINING

D.G. Sotiropoulos, A.E. Kostopoulos, and T.N. Grapsa

University of Patras, Department of Mathematics,
Division of Computational Mathematics & Informatics,
GR-265 00 Rio, Patras, Greece.
e-mail: {dgs,arkostop,grapsa}@math.upatras.gr.

Keywords: back propagation, supervised training, conjugate gradient methods, Perry's method, spectral steplength, non-monotone Wolfe conditions.

Abstract. *In this work, an efficient training algorithm for feedforward neural networks is presented. It is based on a scaled version of the conjugate gradient method suggested by Perry, which employs the spectral steplength of Barzilai and Borwein that contains second order information without estimating the Hessian matrix. The learning rate is automatically adapted at each epoch, using the conjugate gradient values and the learning rate of the previous one. In addition, a new acceptability criterion for the learning rate is utilized based on non-monotone Wolfe conditions. The efficiency of the training algorithm is proved on the standard tests, including XOR, 3-bit parity, font learning and function approximation problems.*

1 INTRODUCTION

Several adaptive learning algorithms for feedforward neural networks have recently been discovered for solving approximation, pattern recognition, classification and other well known problems. Many of these algorithms are based on the gradient descent algorithm well known in optimization theory. They usually have a poor convergence rate and depend on parameters which have to be specified by the user, because no theoretical basis for choosing them exists. The values of these parameters are often crucial for the success of the algorithm. One of these algorithms is the standard backpropagation (BP) [11]. Although BP is the most common and widely used supervised training algorithm, nevertheless, because of the user depended parameters, it is usually inefficient on large scale problems.

The neural network training can be formulated as a nonlinear unconstrained optimization problem. So the training process can be realized by minimizing the error function E defined by

$$E = \frac{1}{2}\sum_{p=1}^{P}\sum_{j=1}^{N_M}\left(o_{j,p}^{M} - t_{j,p}\right)^2 = \sum_{p=1}^{P} E_p, \quad E \in C^2 \tag{1}$$

where $\left(o_{j,p}^{M} - t_{j,p}\right)^2$ is the squared difference between the actual output value at the j-th output layer neuron for pattern p and the target output value. The scalar p is an index over input-output pairs. The general purpose of the training is to search an optimal set of connection weights in the manner that the errors of the network output can be minimized.

The BP algorithm uses the steepest descent search direction [14] with a fixed step size α in order to perform the minimization of the error function. The iterative form of this algorithm is

$$w_{k+1} = w_k - \alpha g_k$$

where w denotes a column weight vector with components $w_1, w_2, \ldots, w_n$ which is defined in the n–dimensional real space $\mathbb{R}^n$, and g the gradient vector of the error function E at w, that is $g = \nabla E(w)$. E represents the batch error measure defined as the sum of squared differences error function over the entire training set.

The inefficiency of steepest descent is due to the fact that the minimization directions and step sizes are chosen poorly; if the first step size does not lead directly to the minimum, steepest descent will zig-zag with many smal steps.

The back propagation search direction d is usually augmented with a *momentum term* (Rumelhart et. al., 1986):

$$d_{k+1} = -g_{k+1} + \beta_k \cdot (w_k - w_{k-1})$$

This extra term is generally interpreted as to avoiding oscillations. It will be shown below that adding the momentum term is wise when the values α_k and β_k are well chosen; one method which chooses these parameters is known as *conjugate gradient.*

In this work we introduce and analyze a Nonmonotone Spectral Conjugate Gradient BackPropagation (NSCGBP) method. The NSCGBP is based on a scaled version of the conjugate gradient method suggested from Perry [8], [4], which employs the spectral steplength of Barzilai and Borwein [1], [5], [6], that contains second order information without estimating the Hessian matrix. The *learning rate* is automatically adapted at each epoch, using the conjugate gradient values and the learning rate of the previous one [9]. In addition, NSCGBP is combined with a new acceptability criterion for the learning rate, based on a generalization of the Wolfe's rule ([2]).

The plan of this paper is as follows. In section 2 we present the proposed training algorithm. Section 3 contains our numerical examples and results. The final section contains a discussion of our results, directions for further research and some concluding remarks.

2 THE PROPOSED METHOD

2.1 PERRY'S CONJUGATE GRADIENT METHOD

The training phase of a feedforward neural network is an unconstrained nonlinear optimization problem. The goal of the training is to search an optimal set of connection weights in the manner that the errors of the network output can be minimized. Besides the classical and well known steepest descent algorithm [14], conjugate gradient algorithm is another search method that can be used to minimize network output error in conjugate directions. One of the remarkable properties of the conjugate gradient method is its ability to generate, in a very economical fashion, a set of vectors with a property known as *conjugacy* [13].

The standard conjugate gradient method is to minimize the differentiable function (1) by generating a sequence of approximation w_{k+1} iteratively according to

$$w_{k+1} = w_k + \alpha_k d_k \tag{2}$$

The scalar α_k is the steplength, known in neural network notation as *learning rate.* The steplength α_k can be determined by line search techniques in the way that $E(w_k + \alpha_k d_k)$ is minimized along the direction d_k, given w_k and d_k fixed.

The standard conjugate gradient algorithm begins the minimization process with an initial estimate w_0 and an initial search direction

$$d_0 = -\nabla E(w_0) = -g_0 \tag{3}$$

Each direction d_{k+1} is chosen to be a linear combination of the steepest descent direction $-g_{k+1}$ and the previous direction d_k. We write

$$d_{k+1} = -g_{k+1} + \beta_k d_k \tag{4}$$

where the scalar β_k is to be determined by the requirement that d_k and d_{k+1} must fulfill the conjugacy property. There are many formulae for the parameter β_k. One of them is the formula introduced by Perry [8] and is given

$$\beta_k = \frac{(y_k - s_k)^T g_{k+1}}{s_k^T y_k} \tag{5}$$

where

$$s_k = w_{k+1} - w_k \quad \text{and} \quad y_k = g_{k+1} - g_k \tag{6}$$

Conjugate gradient method has a second order convergence property without complex calculation of the Hessian matrix. A faster convergence established than first order steepest descent approach.

2.2 THE BARZILAI AND BORWEIN STEPLENGTH

In optimization theory, the classical steepest descent algorithm for the unconstrained minimization of $E : \mathbb{R}^n \to \mathbb{R}$ is given by

$$w_{k+1} = w_k + \eta_k d_k \tag{7}$$

where $d_k = -g_k = -\nabla E(w_k) \in \mathbb{R}$ is the steepest descent search direction and $\eta_k = \arg\min_\eta E(w_k + \eta d_k)$ is the steplength. This method has slow convergence and is badly affected by ill-conditioning, therefore several choices of steplengths have been proposed.

An interesting new idea is the choices of steplength that are proposed and analyzed by Barzilai and Borwein [1] for the steepest descent method for unconstrained optimization. Barzilai and Borwein seek choices of η_k that give a superlinear rate of convergence, and that can be applied in practice without the computation of second derivatives. They take the view that a steepest descent iteration is equivalent to letting w_{k+1} minimize the quadratic model when the Hessian matrix H_k is a multiple of the unit matrix, and they generate suitable multipliers for $k \geq 2$ from the secant equation.

More specifically, Barzilai and Borwein [1] proposed two new step sizes for use in conjunction with the negative gradient $-g_k$. They studied the iteration $w_{k+1} = w_k - S_k g_k$, where S_k has the form $S_k = \eta_k I$. Firstly, η_k minimizes $||s_{k-1} - \eta y_{k-1}||^2$, with $s_{k-1} = w_k - w_{k-1}$ and $y_{k-1} = g_k - g_{k-1}$. The motivation for this choice is that it provides a two-point approximation to the secant equation underlying quasi-Newton methods. This yields the iteration

$$w_{k+1} = w_k - \eta_k g_k \tag{8}$$

where η_k is given by

$$\eta_k = \frac{\langle s_{k-1}, y_{k-1}\rangle}{\langle y_{k-1}, y_{k-1}\rangle} \tag{9}$$

where $\langle\cdot,\cdot\rangle$ denotes the standard inner product.

By symmetry, they minimize $||\eta s_{k-1} - y_{k-1}||^2$ with respect to η. The corresponding step size turns out to be

$$\eta_k = \frac{\langle s_{k-1}, s_{k-1}\rangle}{\langle s_{k-1}, y_{k-1}\rangle} \tag{10}$$

The step size η_k given by (10) is the inverse of the Rayleigh quotient

$$s_{k-1}^T\left[\int_0^1 \nabla^2 E(w_k + t s_{k-1})dt\right]s_{k-1}/s_{k-1}^T s_{k-1} \tag{11}$$

which, of course, lies between the largest and the smallest eigenvalue of the Hessian average $\int_0^1 \nabla^2 E(w_k + t s_{k-1})dt$. This implies that the step size contains second order information without estimating the Hessian matrix. The results shown that this choice of the step size is very efficient [1].

2.3 THE NSCGBP TRAINING METHOD

In this section we will introduce and analyze the nonmonotone spectral conjugate gradient training method. As it is mentioned before, the problem we have to deal with is to minimize the error function (1). Let the family of gradient training algorithms having the iterative form

$$w_{k+1} = w_k + \alpha_k d_k \tag{12}$$

where w_k is the current point, d_k is the search direction and α_k is the steplength.

In our method, by adopting the ideas from Birgin and Martinez in [4], the directions are generated by

$$d_{k+1} = -\eta_k g_{k+1} + \beta_k s_k \tag{13}$$

for $k = 0, 1, 2, \ldots$, where g_k denotes $\nabla E(w_k)$, $w_0 \in \mathbb{R}^n$ is arbitrary and $d_0 = -g_0$.

Suppose, for a moment, that $E(w) \in \mathbb{C}^2$ and $H \equiv \nabla^2 E(w)$ is positive definite. This implies that $y_k \neq 0$. Therefore, the true minimizer w_* satisfies

$$w_* = w_{k+1} + d_*$$

where

$$Hd_* = -g_{k+1}$$

Pre-multiplying by s_k^T, this gives

$$s_k^T H d_* = -s_k^T g_{k+1}$$

Therefore

$$y_k^T d_* = -s_k^T g_{k+1}$$

Thus the hyper-plane

$$\mathcal{H}_k \equiv \{d \in \mathbb{R}^n | y_k^T d = -s_k^T g_{k+1}\}$$

contains the optimum increment d_*, which gives $w_* = w_{k+1} + d_*$. Observe that the null direction $d = 0$ belongs to $\mathcal{H}$ only if $s_k^T g_{k+1} = 0$ which is not our assumption at all.

By the discussion above, it is natural to impose, for the search direction d_{k+1},

$$d_{k+1} \in \mathcal{H}_k \tag{14}$$

Then, by (13), we have

$$\beta_k = \frac{(\eta_k y_k - s_k)^T g_{k+1}}{s_k^T y_k} \tag{15}$$

For $\eta_k = 1$ the above formula is reduced to the formula (5) introduced by Perry in [8].

In this paper, as Birgin and Martinez in [4], we decided to replace the classical choice $\eta_k = 1$ with the spectral gradient choice

$$\eta_k = \frac{\langle s_k, s_k \rangle}{\langle s_k, y_k \rangle} \tag{16}$$

which is the Barzilai and Borwein steplength that we discussed in section 2.2.

The learning rate α_k is a modification of Shanno's choice which has been used in CONMIN [9]. Therefore

$$\alpha_k = \begin{cases} \frac{1}{||g_0||_2}, & \text{if } k = 0; \\ \frac{\alpha_{k-1}||d_{k-1}||_2}{||d_k||_2}, & \text{otherwise.} \end{cases} \tag{17}$$

where d_k is the conjugate gradient direction, d_{k-1} is the previous one, and α_{k-1} is the previous learning rate. The initial learning rate α_0 is $1/||g_0||_2$ where g_0 is the initial steepest descent direction.

As a learning rate acceptability criterion we will utilize the nonmonotone Wolfe criterion introduced by Jiye Han, Jie Sun and Wenyu Sun in [2]. We suggest that the learning rate α_k can be computed along the search direction d_k by a Wolfe nonmonotone line search, which does not enforce the common sense requirement $E(w_{k+1}) < E(w_k)$, but uses a learning rate acceptability criterion which is a generalization of Wolfe's rule. Specifically, we impose that the error function value E of each new epoch must satisfy the Wolfe's conditions with respect to the maximum error function value of a prefixed number M of previous epochs. The nonmonotone Wolfe conditions is given by

$$E(w_k + \alpha_k d_k) - \max_{0 \le j \le M} E(w_{k-j}) \le c_1 \alpha_k \nabla E(w_k)^T d_k \tag{18}$$

$$\nabla E(w_k + \alpha_k d_k)^T d_k \ge c_2 \nabla E(w_k)^T d_k \tag{19}$$

where $0 < c_1 \le c_2 < 1$ and M is a non-negative integer.

The first condition (18) allows any point to be accepted if it improves sufficiently the function value compared with the largest of the $M+1$ (or k if $k \leq M$) most recent function values. The integer M controls the amount of monotonicity that is allowed. The second condition (19) ensures that the denominator of spectral gradient choice (16) is well defined and always positive, since it implies that $s_k^T y_k > 0$. Both conditions allows an increase in the error function values without affecting the global convergence properties as have been proved in [2].

At this point we will summarize the NSCGBP training algorithm.

Algorithm 2.1

```
1. Initialization:
    1.1 Number of epochs $k = 0$.
    1.2 Error goal:= eg.
    1.3 Parameters $M \geq 1$ and $0 < c_1 \leq c_2 < 1$.
    1.4 Weight vector:=$w_0$.
2. Calculate the gradient $g_0 = \nabla E(w_0)$. Calculate the learning rate $\alpha_0$ using the
   relation (17). Compute the weight vector $w_1$ according to relation (12) and set
   $k = k + 1$.
3. Compute the Barzilai and Borwein spectral gradient choice $\eta_k$ using the relation
   (16). Check if $1/\eta_k < e$ or $1/\eta_k < 1/e$. If the relations holds then accept $\eta_k$
   else set $\eta_k = 1$.
4. Calculate $\beta_k$ by Perry's formula given by the relation (15). Calculate the
   new search direction $d$ according to relation (13). If the condition $d^T g_{k+1} \leq$
   $-10^{-3}||d||_2||g_{k+1}||_2$ holds then set $d_{k+1} = d$, otherwise set $d_{k+1} = -\eta_k g_{k+1}$.
5. Calculate the new learning rate $\alpha_k$ according to relation (17)
6. Update the weight vector $w_{k+1}$ according to relation (12). Calculate the gradient
   $g_{k+1} = \nabla E(w_{k+1})$.
    6.1 If the learning rate acceptability condition (nonmonotone
        line search) (18) and (19) is fullfilled goto Step 7.
    6.2 Set $\alpha_k = \alpha_k/2$ and goto Step 6.
7. Check if $E(w_{k+1}) > eg$, set k=k+1 and goto Step 3. Otherwise, get the final
   weight vector $w_*$ and the corresponding value of $E$.
```

Remark 2.1 *Parameter e is the one that Raydan in [7] introduces in order to avoid having the spectral gradient choice very large or too small.*

Remark 2.2 *The search direction computed in Step 4 according to relation (13) can fail to be a descent direction. In our algorithm, when the angle between d and $-g_{k+1}$ is not acute enough we "restart" the algorithm with the spectral gradient direction $-\eta_k g_{k+1}$, in the same manner as in [4].*

Remark 2.3 *As we can see from the above algorithm, the method needs the same function and gradient evaluations, because of the nonmonotone Wolfe conditions, in order to accept the learning rate. These conditions as it is obvious from (18) and (19) need a function and a gradient evaluation.*

3 NUMERICAL EXAMPLES

The nonmonotone spectral conjugate gradient backpropagation method (NSCGBP), as described in the previous section, was tested in four problems in order to study and evaluate the performance. The problems have been tested, are the eXclusive OR Problem, the 3-bit Parity Problem, the Font Learning Problem and the Continuous Function Approximation Problem. On each problem, four algorithms have been simulated. These algorithms are the standard backpropagation (BP), the momentum backpropagation (MBP) [3], the adaptive backpropagation (ABP) [12] and the NSCGBP. For the simulations an IBM PC compatible with Matlab Version 5.3 has been used. We have utilized Matlab Neural Network

Toolbox Version 3.0 for the BP, MBP and ABP algorithms. For the heuristic parameters of the previous three algorithms, Toolbox default values are used, unless stated otherwise. The NSCGBP algorithm has been implemented in Matlab environment and the values of parameters M, c_1, c_2 and e have been fixed to 10, 10^{-4}, 0.5 and 10^{-3}, respectively. At the start of each simulation, the weights of the network have been initialized by the Nguyen - Widrow method [10]. Each algorithm has been tested for the same initial weights. During training of the network, each time step is called an epoch and is defined to be a single sweep through all training patterns. At the end of each epoch, the weights of the network have been updated.

The results of all four algorithms will be presented. For each problem, we present a table which summarizes the performance of the algorithms for simulations that have reached solution. The resulted parameters are: *min* the minimum number, *max* the maximum number, *mean* the mean value, *s.d.* the standard deviation of function and gradient evaluations and *succ.* the simulations succeeded out of 1000 simulations. As it is already known, each algorithm, including the NSCGBP, has the same number of function and gradient evaluations. If an algorithm fails to converge, it is considered that it fails to train the FNN, but its epochs, function and gradient evaluations are not included in the statistical analysis of the algorithms. This fact clearly favors BP, MBP and ABP that require too many epochs to complete the task and/or often fail to converge.

3.1 THE EXCLUSIVE - OR PROBLEM

The first problem we have been encountered is the eXclusive - OR (XOR) Boolean function problem, which is considered as a classical problem for the FNN training. The XOR function maps two binary inputs to a single binary output. As it is well known this function is not linearly separable.

Algorithm	**min**	**max**	**mean**	**s.d.**	**succ.**
BP	31	975	103.53	132.05	69.9%
MBP	24	922	125.59	156.94	73.9%
ABP	19	865	49.11	74.34	69.6%
NSCGBP	29	998	145.27	185.40	84.6%

Table 1: The XOR Problem

The selected architecture of the FNN is 2-2-1 (six weights and three biases) with logistic neurons with biases in the hidden layer and with a linear output neuron with bias. The error goal has been set to 0.01 and the maximum epochs to 1000. For the BP and MBP algorithms the learning rate is chosen to be 0.1 instead of the default value 0.01 to accelerate their convergence, since they converge slowly with the default value in this problem. The results of the simulations are presented in Table 1.

3.2 THE 3-BIT PARITY PROBLEM

In this simulation we have been considered the 3-bit parity problem, which can be considered as the 3-bit version of the XOR problem. This problem maps three binary inputs to a single binary output. The target of the output is 1, if the number of 1 bits in the input is odd, and 0 otherwise. The selected architecture of the FNN is 3-2-1 (eight weights and three biases) with logistic neurons with biases in the hidden layer and with a linear output neuron with bias. The error goal has been set to 0.01 and the maximum epochs to 1000. For the BP and MBP algorithms the learning rate is chosen to be 0.1 instead of the default value 0.01 to accelerate their convergence, since they converge slowly with the default value in this problem. The results of the simulations are presented in Table 2.

Algorithm	**min**	**max**	**mean**	**s.d.**	**succ.**
BP	-	-	-	-	0.0%
MBP	174	991	487.94	205.01	56.2%
ABP	435	972	563.46	113.56	48.4%
NSCGBP	161	994	424.07	171.64	75.6%

Table 2: The 3-bit Parity Problem

3.3 THE FONT LEARNING PROBLEM

The third test problem we have been encountered is the font learning problem. We present to the network 26 matrices with the capital letters of the English alphabet. Each letter has been defined in terms of binary values on a grid of size 5×7. The selected architecture of the FNN is 35-30-26 (1830 weights and 56 biases) with logistic neurons with biases in the hidden layer and with a linear output neuron with bias.

Algorithm	**min**	**max**	**mean**	**s.d.**	**succ.**
BP	1070	1992	1550.8	184.76	75.1%
MBP	1226	1801	1521	156.27	4.2%
ABP	1285	1998	1785.6	160.73	36.2%
NSCGBP	325	1250	713.32	131.09	100.0%

Table 3: The Font Learning Problem

Also, in order to improve the performance of the methods with fixed learning rate (i.e. BP and MBP), the weights have been initialized following the Nguyen - Widrow method, as we have stated in the beginning of this section, but afterwards the weights and biases of the output layer have been multiplied by 0.01. The error goal has been set to 0.1 and the maximum epochs to 2000. The results of the simulations are presented in Table 3.

3.4 THE CONTINUOUS FUNCTION APPROXIMATION PROBLEM

The last problem we have been considered is the approximation of the continuous transcendental function $F(x) = \sin(x)\cos(x)$. This problem maps one real input to a single real output. The input values are 20 equally spaced points $x_i \in [0, 2\pi]$ and the target values are the mapping of these points from function $F(x)$. As it is cleared, we have 20 patterns and each pattern is consisted of one input $x \in [0, 2\pi]$ and one target value $F(x)$.

Algorithm	**min**	**max**	**mean**	**s.d.**	**succ.**
BP	366	984	793.01	157.11	7.7%
MBP	349	994	791.65	162.28	7.8%
ABP	125	999	619.27	217.87	32.5%
NSCGBP	130	999	487.78	193.38	69.8%

Table 4: The Cont. Function Approx. Problem

The selected architecture of the FNN is 1-10-1 (twenty weights and eleven biases) with logistic neurons with biases in the hidden layer and with a linear output neuron with bias. The error goal has been set to 0.1 and the maximum epochs to 1000. The results of the simulations are presented in Table 4.

4 CONCLUSIONS

In this paper, a new training method is introduced for fast supervised learning. It is shown that this method is very promising with several advantages. Our experimental results shown that the NSCGBP method clearly outperforms the classical training algorithms (BP, MBP and ABP). It has improved the average number of function and gradient evaluations, and better convergence rates have established. Also, it is observed an increment in the convergence speed.

Further work must be done in order to evaluate the generalization of the method. Moreover, it would be interesting to replace the learning rate adaption with more sophisticated learning rate adaptions. On the other hand, we can substitute Perry's formula with other well known formulae as Fletcer-Reeves and Polac-Ribiére, in order to test how the NSCGBP will perform.

References

[1] J. Barzilai and J.M. Borwein, Two point step size gradient methods, *IMA J. Numer. Anal.*, **8**, 141–148, (1988).

[2] J. Han, J. Sun and W. Sun, Global Convergence of Non-Monotone Descent Methods for Unconstrained Optimization Problems, to appear in *Journal of Computational and Applied Mathematics.*

[3] R.A. Jacobs, Increased rates of convergence through learning rate adaptation, *Neural Networks*, **1**, 295–307, (1988).

[4] E.G. Birgin and J.M. Martinez, A spectral conjugate gradient method for unconstrained optimization, *Applied Mathematics and Optimization*, **43**, 117–128, (1999).

[5] V.P. Plagianakos, D.G. Sotiropoulos, and M.N. Vrahatis, A Nonmonotone Backpropagation Training Method for Neural Networks, Dept. of Mathematics, Univ. of Patras, Technical Report No.98-04, (1998).

[6] V.P. Plagianakos, D.G. Sotiropoulos and M.N. Vrahatis, Automatic adaptation of learning rate for backpropagation neural networks In: Recent Advances in Circuits and Systems, N.E. Mastorakis ed., pp.337-341, World Scientific Publishing Co. Pte. Ltd., (1998).

[7] M. Raydan, The Barzilai and Borwein gradient method for the large scale unconstrained minimization problem, *SIAM J. Optim.*, **7**, 26–33, (1997).

[8] A. Perry, A modified conjugate gradient algorithm, *Operations Research*, **26**, 26–33, (1978).

[9] D.F. Shanno and K.H. Phua, Minimization of unconstrained multivariate functions, *ACM Transactions on Mathematical Software*, **2**, 87–94, (1976).

[10] D. Nguyen and B. Widrow, Improving the learning speed of 2-layer neural network by choosing initial values of the adaptive weights, IEEE First International Joint Conference on Neural Networks, **3**, 21–26, (1990).

[11] D.E. Rumelhart and J.L. McClelland(eds), Parallel distributed processing: explorations in the microstructure of cognition, Vol. 1: Foundations, MIT Press, 1986.

[12] T.P. Vogl, J.K. Mangis, J.K. Rigler, W.T. Zink and D.L. Alkon, Accelerating the convergence of the back-propagation method, *Biological Cybernetics*, **59**, 257–263,(1988).

[13] J. Nocedal and S.J. Wright, *Numerical Optimization*, Springer Series in Operations Research, (1999).

[14] A. Cauchy, Méthode générale pour la résolution des systéms d'équations simultanées, *Comp. Rent. Sci. Paris*, **25**, 536–538, (1847).