Proceedings of the 1. Conference on
Applied Mathematics and Computation
Dubrovnik, Croatia, September 13–18, 1999
pp. 253–261

Parameter Identification Problem Solving Using Genetic Algorithm

Krešimir Seršić* and Igor Urbiha†

Abstract. We describe two methods for solving the parameter identification problem for ordinary differential equations of the second order — a genetic algorithm and a variant of Monte Carlo method. These methods were tested and compared in one practical example.

AMS subject classification: 65C05, 65L99

Key words: Monte Carlo methods, genetic algorithms, evolution programs, parameter identification, system of ordinary differential equations, optimization

1. Introduction

Many processes occurring in nature and living organisms can be modelled by a system of differential equations (see, e.g., [21, 11, 23, 22]). Models usually contain some unknown parameters, which are to be determined from the given data in order to minimize the differences between the experimental values and the values given by models. This problem is known as the Parameter Identification Problem.

There are several methods developed for solving such problems (and they are usually difficult to implement on a computer) for ordinary differential equations of the first order: finite differences (see, e.g., [16, 3]), integration of data (see [3]), initial value approach (see [3, 4, 7, 19, 24, 25, 26]), smoothing the data (see, e.g., [16, 17, 23, 24]), multiple shooting approach (see, e.g., [4, 9]). The parameter identification problem for ordinary differential equations of the second order is studied in [16, 24] and [21].

Complicated methods tend to introduce errors which, combined with errors already present in experimental data, may effect undesirable influence on computed results. Errors introduced by such methods can be diminished by using simpler methods.

Suppose a theoretical model is described by an ordinary differential equation of the second order:

$$\frac{d^2y}{dt^2} = f\big(t, y(t), y'(t), \vec{P}\big), \tag{1}$$

*Faculty of Electrical Engineering, University of Osijek, Kneza Trpimira 2b, 31000 Osijek, Croatia, e–mail: sersic@etfos.hr

†Department of Mathematics, University of Zagreb, Bijenička cesta 30, 10000 Zagreb, Croatia, e–mail: urbiha@math.hr

where $\vec{P}$ is a vector of n real parameters. Experimental data are given with pairs (t_i, y_i), $i = 1, 2, \ldots, m$, where $0 \leq t_1 < \cdots < t_m < \infty$, and $y_1, \ldots, y_m$ denote approximate values of the searched function y at the data points $t_1, \ldots, t_m$. According to given data, one has to determine the optimal parameter values $\vec{P} = (p_1, p_2, \ldots, p_n)$ which minimize the functional:

$$F(\vec{P}) = \sum_{i=1}^{m} \big(y_i - y(t_i, \vec{P})\big)^2. \tag{2}$$

The parameter identification problem for the system of ordinary differential equations of the second order is similarly defined. The problem is usually stated as follows: Find reasonable values for $\vec{P}$ so that, for suitably chosen initial conditions, the solution of (1) fits given data. Once the values for $\vec{P}$ have been estimated, suitable initial conditions $y(t_1) = \mu$, $y'(t_1) = \nu$, still have to be found in order to determine the function y from (1).

Various numerical methods usually try to find $\vec{P}$ first, and then, taking it into account, search for suitable μ and ν thus solving the problem. The algorithms described below search for $\vec{P}$, μ and ν simultaneously, so a change in (2) is needed:

$$F_p(\vec{P}, \mu, \nu) = \sum_{i=1}^{m} \big|y_i - y_{\mu,\nu}(t_i, \vec{P})\big|^p, \tag{3}$$

or

$$F_\infty(\vec{P}, \mu, \nu) = \max\Big\{\big|y_i - y_{\mu,\nu}(t_i, \vec{P})\big| \;\Big|\; i = 1, 2, \ldots, m\Big\}, \tag{4}$$

so F_∞ actually measures maximal error of the approximation.

Due to the nature of the method, L_1 (or "Manhattan") norm can be used, which also makes computations faster. Algorithms were tested with $p \in \{1, 2\}$, i.e., with L_1 and L_2 norms (squaring was avoided, as it slows down execution of a program).

Two methods that can be used to solve the mentioned problems are given in this paper, one using Monte Carlo's "massive attack" strategy, and the other relying on genetic (or evolution, see [14, 15]) algorithms. Both methods depend on extensive use of (pseudo) random numbers.

A function which needs to be minimized (optimized) is often referred to as an objective function.

2. Algorithms

The first algorithm relies on Monte Carlo strategy of "massive attack". Suppose we need to solve an optimization problem $f(X) \to \min$, where $f : S \subseteq \mathbb{R}^m \to \mathbb{R}^+$. Let $X^* = (x_1^*, x_2^*, \ldots, x_m^*) \in S$ be the solution, i.e., $f(X^*) = \min_{X \in S} f(X)$. First, a bounded subset of the domain of the function f, which contains X^*, needs to be found. With smaller starting domain, program will find good approximations to the solution faster. A domain is given as a Cartesian product of the form $D_0 = \Pi_{i=1}^m \langle a_i, b_i \rangle$, where

$x_i^* \in \langle a_i, b_i \rangle$. The program randomly computes a large number of points $Y_i \in D_0$ belonging to the domain ("massive attack"), and keeps track of a certain number of points Y_j with lowest values $f(Y_j)$. Those points are used to determine the smallest domain D_1 which contains them all. The program repeats the process and finds a sequence of domains D_i, $i = 0, 1, \ldots, n$, where $D_0 \supseteq D_1 \supseteq \cdots \supseteq D_n$, each of which, hopefully, contains the solution, and keeps track of several points with lowest function values. The point with the minimal function value is given as the approximation to the solution.

Algorithm 1 (MC). *"Massive attack" with successive domain shrinking:*

`input:` *domain*
`repeat`
  compute sufficiently many random points belonging to the domain
  domain ← *the smallest domain containing the best points found so far*
`until` *some appropriate condition is met*
`output:` *domain and the best point*

The other algorithm uses genetic approach devised by J. H. Holland [14] and was implemented using modified program GENOCOP III, originally programmed by Z. Michalewicz [15].

The main difference between the two algorithms is in the way they handle randomly generated points. While the first algorithm leaves them alone, the second one tries to "push" them towards (or, in the general direction of) the solution, in somewhat random fashion, hoping that better approximations will be obtained. Since the solution is unknown, it uses the best approximation found, expecting that it is near the solution. Description of various "pushing" methods used by genetic and evolution algorithms is beyond the scope of this article. It suffices to say that freshly generated random point and the best current approximation are combined to produce an "offspring", a new point whose function value is then compared with currently best function values.

Algorithm 2 (GEN). *Genetic or evolution approach:*

`input:` *domain*
`repeat`
  `repeat`
    compute a random point belonging to the domain
    "push" it towards the best current point
  `until` *sufficiently many points are generated*
  domain ← *the smallest domain containing the best points found so far*
`until` *some appropriate condition is met*
`output:` *domain and the best point*

Due to the nature of generation of new points, this algorithm can "forget" the best point found so far, when making transition to a new domain. Nevertheless, it

exhibited faster convergence than the first algorithm. However, it is not clear whether the genetic approach is significantly better than the plain "massive attack" approach.

A domain given as an output can be used again as input, if necessary. It is advisable to rerun the program several times with sufficiently small domains. Due to the uncertainty involved with the use of random numbers, it may happen that some domains could be shrunk too much and thus lose the solution. In such cases, an approximation to some local optimal point would be found.

3. Example

The following example is taken from [21]. Enzyme effusion problem is described by the following system of differential equations (see, e.g., [13, 12, 11, 24]):

$$\begin{aligned} y_1' &= p_1 \cdot (27.8 - y_1) + \frac{p_4}{2.6} \cdot (y_2 - y_1) + \frac{1991}{t\sqrt{2\pi}} \exp\Big(-\frac{w^2}{2}\Big), \\ y_2' &= \frac{p_4}{2.7} \cdot (y_1 - y_2), \end{aligned} \tag{5}$$

where $w = (\ln(t) - p_2)/p_3$ is a model of enzyme effusion into the blood after a heart attack. It contains four parameters which need to be estimated according to the given data (see Table 1). The system is transformed into a second-order differential equation for a function whose values on experimental data are known. Function f mentioned in 1) for this problem is

$$f(t, y, y', \vec{P}) = -\frac{5.3}{7.02}\, p_4 y' - p_1 y' + \frac{p_1 p_4}{2.7}(27.8 - y) + \frac{1991}{t^2}\Big(\frac{p_4}{2.7}\, t - 1 - \frac{w}{p_3}\Big) \exp\Big(-\frac{w^2}{2}\Big),$$

where w is the same as in (5). Experimental data are given in Table 1.

t	y	t	y	t	y	t	y
0.1	27.8	21.3	331.9	42.4	62.3	81.1	23.5
2.5	20.0	22.9	243.5	44.4	58.7	91.2	24.8
3.8	23.5	24.9	212.0	47.9	41.9	101.9	26.1
7.0	63.6	26.8	164.1	53.1	40.2	115.4	33.3
10.9	267.5	30.1	112.7	59.0	31.3	138.7	17.8
15.0	427.8	34.1	88.1	65.1	30.0	163.2	16.8
18.2	339.7	37.8	76.2	73.1	30.6	186.7	16.8

Table 1. Data for enzyme effusion problem.
Independent variable t is measured in seconds.

Standard Runge–Kutta algorithm (from [20]) was used to calculate $y_{\mu,\nu}(t_i, \vec{P})$ needed in (3) and (4).

In the following two tables the results obtained using programs coded after algorithms **MC** and **GEN** are shown. The values of function F_p in (3) and (4) for $p \in \{1, 2, \infty\}$ of found approximations are given in the columns denoted by F_1, F_2 and F_∞ (max norm). The numbers in F_∞ column are actually maximal errors.

alg.	p_1	p_2	p_3	p_4	$y(0.1)$	$y'(0.1)$	F_1	F_2	F_∞
MC	0.272	2.639	0.359	0.251	27.707	−2.605	214.4	4426.3	47.1
GEN	0.279	2.647	0.367	0.237	27.800	−2.720	214.0	4554.0	47.6

Table 2. Results obtained by Monte Carlo method and genetic approach using L_1 norm (the objective function in both programs was (3) with $p = 1$).

alg.	p_1	p_2	p_3	p_4	$y(0.1)$	$y'(0.1)$	F_1	F_2	F_∞
MC	0.273	2.654	0.373	0.196	27.119	−2.980	224.1	3954.7	36.6
GEN	0.271	2.650	0.370	0.208	25.985	−2.322	225.3	3955.4	37.0

Table 3. Results obtained by Monte Carlo method and genetic approach using L_2 norm (the objective function in both programs was (3) with $p = 2$).

Results obtained with the use of L_∞ norm in the objective function were expectedly poor, so they were not included here.

Although norms L_1 and L_2 are equivalent, their use in the objective function (3) caused different consequences, i.e., different best approximations were found. The best approximation obtained with the use of one norm does not need to be the best approximation in some other norm (compare the last three columns in Tables 1 and 3).

The best approximations obtained when using L_1 norm had higher errors at few points and lower average of other errors. The best approximations obtained when using L_2 norm had lower highest errors and higher average of other errors (compare with Figure 1). It is difficult to say which best solution is better, without knowing something about the accuracy of data and model.

If most of the experimental data are accurate, then it is better to use L_1 norm. If data are less accurate, but none of them highly inaccurate, then L_2 norm could be a better choice (compare with Figure 1). Of course, errors introduced by a model must also be taken into account. In the extreme case when we have inaccurate data or model, then L_∞ (max norm) could be a reasonable choice.

Borders of Figure 1 and Figure 3 are drawn in the global picture, given with Figure 2.

The experimental data for this problem seem to have a kind of periodic behaviour not captured by the model used. (See Figure 2 – points tend to go “upward” above the curve, and then suddenly “drop” down below it.)

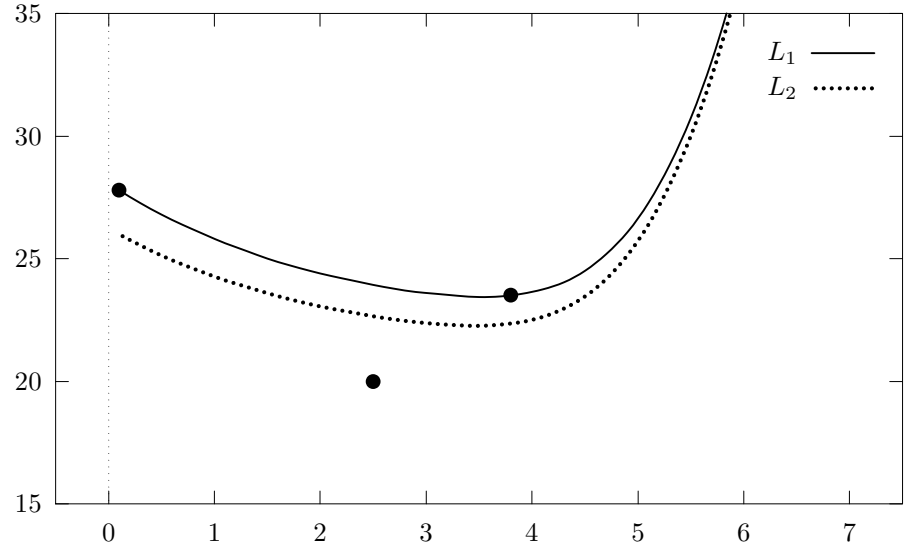

Figure 1. Approximations to enzyme effusion problem found with algorithm GEN.

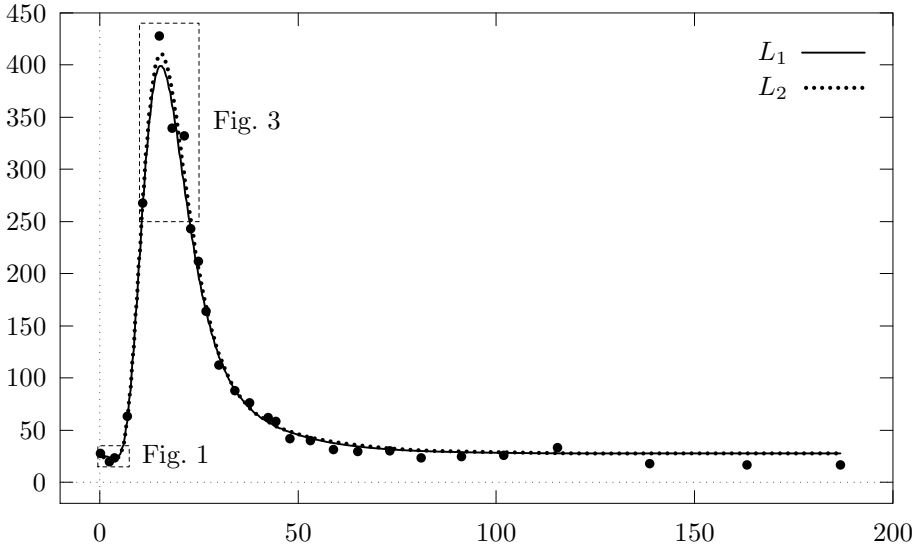

Figure 2. Approximations to enzyme effusion problem found with algorithm GEN. The curve with higher maximum was found using L_2 norm.

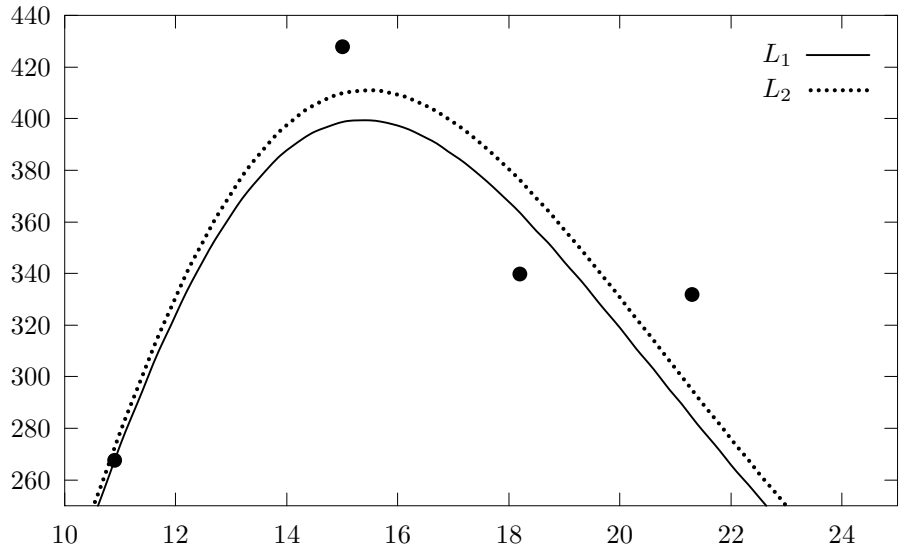

Figure 3. Approximations to enzyme effusion problem found with algorithm GEN.

It seems that we have many less accurate data without large errors, and best approximations obtained with the use of L_2 norm could be better than other. However, the use of L_1 norm is regarded as a better choice, as it is less sensitive to problems which may occur at extreme values of given data (see [10]; compare with Figure 3).

Figures 1, 2, and 3 were made with Gnuplot MS–DOS version 3.7.1. Source codes, written in C, of programs used in the example are available upon request.

4. Some statistics about executions of programs

There are many parameters influencing both programs, so it is difficult to compare their efficiency. Each program (**MC** and **GEN**) has been run three times in a row: the first time using L_1 norm and following starting domain:

$$y(0.1) \in [24.0, 29.0], \quad y'(0.1) \in [-4.0, 4.0],$$
$$p_1 \in [0.2, 0.4], \quad p_2 \in [2.5, 2.9], \quad p_3 \in [0.3, 0.44], \quad p_4 \in [0.1, 0.3],$$

the second and the third time using the output domain of the previous execution as the input domain. Such "three times invocation" has been repeated ten times. The program **MC** generated twice as many approximations (2000), compared to the program **GEN** (1000), to compensate for **GEN**'s approximations generating technique. Namely, upon generating one new approximation (via crossing-over operator, using two "old" approximations), it undergoes a transformation (via mutation operator) which gives one more approximation. So, actually, two new approximations are gen-

erated at each generation, hence the number of generations for the program **MC** was doubled.

The execution times for each program are given in the Table 4, together with the best, average, and the worst values of the objective function (3) with $p = 1$, using L_1 norm of found approximations. Each execution of **MC** lasted about 19 minutes while each execution of **GEN** lasted about 24 minutes, both with negligible deviations.

alg.	best	average	worst	time (min)
MC	216.13	218.68	220.91	19
GEN	214.35	218.00	228.82	24

Table 4. The values of the objective function and execution times for the "three times invocation" of programs **MC** and **GEN**.

The program **MC** is expectedly faster, as it generates an approximation and does nothing else. Since the program **GEN** does much more work to generate one approximation, it is slower, but also gives better approximation, as it tries to "push" every new approximation towards the best solution.

Programs were run on a PC with 300 MHz processor in a DOS window within Windows 95 operating system.

5. Conclusion

Two general stochastic methods for solving optimization problems with their application in the parameter identification problems were presented in this article. The methods are simple in comparison to the deterministic ones and easier to code.

It was shown that if both experimental data and model are accurate, with exception of few data, then it is advisable to use L_1 norm in the objective function. If data or model is less accurate, then it is better to use L_2 norm. Finally, if it is necessary to use inaccurate model or data, then the use of L_∞ norm in the objective function could be useful.

Acknowledgement. The authors wish to thank R. Scitovski, T. Marošević and D. Jukić for helpful suggestions and improvements of the paper, and Z. Michalewicz for his GENOCOP III program intended for solving numerical optimization problems with linear and nonlinear constraints. The program can be freely downloaded at URL `http://www.coe.uncc.edu/~zbyszek/gchome.html`.

References

[1] F. S. Acton, *Numerical Methods That Usually Work*, The Mathematical Association of America, 1990.

[2] T. Bäck, D. B. Fogel, and Z. Michalewicz, *Handbook of Evolutionary Computation*, Oxford University Press, New York, 1996.

[3] Y. Bard, *Nonlinear Parameter Estimation*, Academic Press, New York, 1974.

[4] H. G. Bock, *Recent advances in parameter identification techniques for O.D.E.*, in Numerical Treatment of Inverse Problems in Differential and Integral Equations, P. Deuflhard, ed., Heidelberg, 1983, pp. 95–121.

[5] P. T. Boggs, R. H. Byrd, and R. B. Schnabel, *A stable and efficient algorithm for nonlinear orthogonal distance regression*, SIAM J. Sci. Statist. Comput., 8 (1987), pp. 1052–1078.

[6] D. Bokal, *Evolutionary algorithms for cluster geometry*, Preprint Series, 36 (1998), p. 628.

[7] D. W. Brewer, J. A. Burns, and E. M. Cliff, *Parameter identification for an abstract Cauchy problem by quasilinearization*, Quart. Appl. Math., 51 (1993), pp. 1–22.

[8] J. E. Dennis, Jr. and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice–Hall, Englewood Cliffs, NJ, 1983.

[9] P. Deuflhard and G. Bader, *Multiple shooting techniques revisited*, in Numerical Treatment of Inverse Problems in Differential and Integral Equations, P. Deuflhard, ed., Heidelberg, 1983, pp. 74–94.

[10] Y. Dodge, ed., L_1 *Statistical Procedures And Related Topics*, Institute of Mathematical Statistics, Hayward, California, 1997.

[11] B. van Domselaar, *Een Mathematische Analyse van het Hertinfarct*, Report NN 4/74, Mathematisch Centrum, Amsterdam, 1974, pp. 19.

[12] B. van Domselaar and P. W. Hemker, *Nonlinear parameter estimation in initial value problems*, Report NW 18/75, Mathematisch Centrum, Amsterdam, 1977, pp. 49.

[13] P. W. Hemker and J. Kok, *A project on parameter identification in reaction kinetics*, Report NM–R9301, Centrum voor Wiskunde en Informatica, Amsterdam, 1993, pp. 38.

[14] J. H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, MI, 1975.

[15] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, third ed., Springer–Verlag, Berlin, 1996.

[16] H. Mühlig, *Parameteridentifikation bei Differentialgleichungen mit Hilfe von B-splines*, Wiss. Z. Tech. Univ. Dresden, 41 (1992), pp. 3–6.

[17] H. Mühlig, *Lösung praktischer Appriximationsaufgaben durch Parameteridentifikation*, Z. Angew. Math. Mech., 73 (1993), pp. 837–839.

[18] J. M. Ortega, *Numerical Analysis, A Second Course*, SIAM, Philadelphia, 1990.

[19] K. Salkauskas, *Moving least squares interpolation with thin-plate splines and radial basis function*, Comput. Math. Appl., 24 (1992), pp. 177–185.

[20] F. Scheid, *Numerical Analysis*, Schaum's Outline Series in Mathematics, McGraw–Hill, New York, 1968.

[21] R. Scitovski and D. Jukić, *A method for solving the parameter identification problem for ordinary differential equations of the second order*, Appl. Math. Comput., 74 (1996), pp. 273–291.

[22] R. Scitovski, T. Marošević, and D. Jukić, *Estimation of the optimal initial conditions in mathematical model*, in Proceedings of the 17th International Conference on Information Technology Interfaces ITI, June 13–16, 1995., Pula, Croatia, pp. 478–479.

[23] J. Swartz and H. Bremermann, *Discussion of parameter estimation in biological modelling: Algorithms for estimation and evaluation of the estimates*, J. Math. Biol., 1 (1975), pp. 241–257.

[24] J. M. Varah, *A spline least squares method for numerical parameter estimation in differential equations*, SIAM J. Sci. Statist. Comput., 3 (1982), pp. 28–46.

[25] J. M. Varah, *On the conditioning of parameter estimation problems*, in Reliable Numerical Computation, M. G. Cox and S. Hammarling, eds., Oxford Science Publications, New York, 1990, pp. 187–195.

[26] J. Williams, *Approximation and parameter estimation in ordinary differential equations*, in Algorithms for Approximation II, J. C. Mason and M. G. Cox, eds., Chapman and Hall, London, 1990, pp. 395–403.