An evolutionary algorithm for physical motion analysis

Jean Louchet

ENSTA, Laboratoire d'Electronique et d'Informatique
32 boulevard Victor
75015 PARIS, France
&
INRIA, SYNTIM project
Rocquencourt, B.P. 105
78153 LE CHESN \Y Cedex, France

e-mail: louchet@bora.inria.fr

Abstract

This paper presents an approach to motion understanding, through identification of physical parameters from image sequences. It is based on a family of particle-based physical models where deformable objects are represented as sets of weighted particles and their interactions. The interaction model presented derives from an energy potential, using dual bonds (extension springs) and ternary bonds (torsional springs).

An original dynamical motion analysis algorithm is described, which extracts physical animation parameters (springs lengths, angles, stiffness...) through the processing of an image sequence. Genetic techniques are employed to perform the fitting of parameters in an analysis-by-synthesis scheme. Experimental test results on synthetic sequences are reported.

Keywords:

motion analysis, physical modelling, model-based coding, particle models, analysis-synthesis, deformation, animation, vibration modelling, genetic algorithms.

1- INTRODUCTION

This work is a first step towards automatic interpretation of motion from image sequences, in terms of mechanical properties of the objects involved. Elastic, deformable objects are modelled using structures of particles connected by elastic bonds. A self-consistent analysis-by-synthesis method interprets the space-time behaviour of the object by fitting parameters of the model.

Many approaches to motion interpretation are restricted to a 2D or 3D geometric or kinematic view (optical flow,...) and often fail to consider whether or not motion is actually consistent or physically possible. Human skills in motion visual interpretation in kinesthesic terms (e.g. in sports training, where state-of-the-art computer vision lags far behind human ability) suggests the introduction of a general physical and empirical knowledge of the processes involved, into automatic interpretation and prediction of images.

Concerning physical motion modelling, we consider the approach of the LIFIA/ACROE team in the University of Grenoble [LJFCR91, C90] as our main reference. It aims at modelling particle-based physical processes in real time, using gesture-feedback transducers to produce synthetic images or sounds in a Virtual Reality environment. ACROE have confirmed the interest of the particle-based approach in image synthesis but have not solved satisfactorily the problem of automatic model identification as yet [SC93]. Their model is constrained by specific user interface requirements.

Our physical model uses a greater variety of primitives and an energy-based model to enable a simpler object representation, making it more suitable for automatic parameter identification. The aim of this work is to find algorithms that can perform this task on particle-based animation models.

This research may have applications in domains like physical motion understanding, motion prediction, synthetic TV, animated image synthesis.

The first part of this paper describes the animation model used, based on particles and energy-derived interaction forces. The second part deals with the identification methods we have developed to get the dynamic parameters. Some experimental test results are given in appendix.

2- PARTICLE-BASED PHYSICAL MODELLING

2-1 Physical modelling

The first objective is to introduce a dynamical animation model for use as the basis of an image sequence analysis scheme. The original meaning of "dynamics" refers to forces and force-based models, rather than to a merely kinematic approach.

One of the difficulties of this approach comes from the fact that most Computer Vision research starts from a given application to be solved, and then develops *ad-hoc* modelling methods as required. In contrast the approach adopted here takes motion modelling to be a fundamental task in itself, independent of any particular application.

Computer vision has a central interest in deriving physical properties of objects in the scene from their images, contributing to a more complete and accurate image interpretation. These physical properties are the input parameters of physical models used in the image synthesis process:

$$\begin{tikzcd}
\text{physical model} \arrow[r, shift left, "\textit{physical motion synthesis}"] & \text{positions versus time} \arrow[l, shift left, "\textit{physical motion interpretation}"] \arrow[r, shift left, "\textit{image synthesis (rendering)}"] & \text{images} \arrow[l, shift left, "\textit{detection (low-level vision)}"]
\end{tikzcd}$$

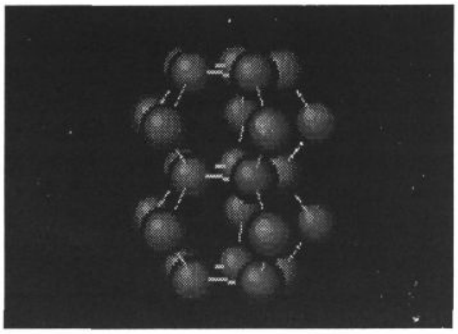
In the analysis-by-synthesis technique used here, a physical model of motion is first introduced and implemented in the form of a motion synthesis algorithm. Then the motion interpretation algorithm utilizes the synthesis algorithm as a component of an optimization process. Physical modelling thus forms the central tool in the analysis-by-synthesis process.

For the purposes of what follows, the low-level vision part of the above diagram need not be considered in detail as its specification will be dictated by higher level requirements. Similarly, image synthesis is only secondary to the main task of physical motion interpretation. It is only needed to provide the test sequences for direct input to the motion interpretation algorithm (bottom left arrow).

The first task is to choose a model containing the physics controlling the system's behaviour. Different points of view in image animation (e.g. the realism of articulations, shocks processing, interfacing gestural control...) have led the image synthesis community to a variety of choices for primitives and physical representations of behaviour and interaction [AG85, DZ93, GV89, GVP91, MZ90, TPBF87].

To this end we devised a structure description language based on point masses and their interactions. It includes both binary (pairwise) and ternary (torsional springs) interactions, to allow thin objects to be represented by simple structures, only containing physically meaningful particles, preferably located inside the objects.

The physical consistency of the system is ensured by the use of global energy potentials in this particle-based model. The total energy of the system is expressed as a sum (over all particles) of local potential energy terms associated with each particle, but retains the possibility of refinements by adding e.g. some global environmental corrective terms. The input variables are distances and angles between particles.



A mechanical structure.

Using binary bonds only, would generate two problems, both linked to the fact that modelling a given object without ternary bonds may imply the addition of extra particles into the object model, in order to mimic the ternary bonds' torsional effects.

The first is that the process of automatically extracting the positions of these extra particles from real images seems to be out of the reach of a low-level image processing algorithm. The second difficulty arises from the fact that the extra particles introduced may have low masses and generate motion aliasing effects (due to excessive force/mass ratios) which could only be solved by an increase in the sampling frequency and precision of number representation. This would cancel out any benefits derived from the simplicity of using binary interactions.

In the present phase of the project, which aims only at solving the identification problem, no dissipative energy terms have been included.

2-2 Implementing an energy-based model

As already mentioned, the total energy of the system is given by sum of local energy terms:

$$E = \sum_{particles} \left(E_{binary} + E_{ternary}\right)$$

where:

$$E_{binary}(particle\ i) = \sum_{j\ neigh.i} S_{ij}\left(dist(i,j) - L_{ij}\right)^2$$

and:

$$E_{ternary}(particle\ i) = \sum_{j,k\ neigh.\ i} T_{jik}\left(\vec{u}(i,j)\bullet\vec{u}(i,k) - M_{jik}\right)^2$$

$\vec{u}$ (i,j) and $\vec{u}$ (i,k) are the unit vectors of the *i-j* (resp. *i-k*) directions, "•" is the scalar product, L_{ij} and M_{jik} are the *i-j* link length and the *j-i-k* angle cosine at rest, S_{ij} and T_{jik} are the radial and angular stiffness coefficients.

With the preceding conventions, the force model may be derived from the gradient of the bond energy E. Its *x* coordinate may be written as:

$$f_x = f_{1x} + f_{2x} + f_{3x}$$

where:

$$f_{1x} = -\sum_{j\ neigh.i} S_{ij} \frac{x_{ij}\left(d_{ij} - L_{ij}\right)}{d_{ij}}$$

$$f_{2x} = -\sum_{\substack{j\ neigh.\ i \\ k\ neigh.\ i \\ k \neq j}} T_{jik}\left(\frac{x_{ij}x_{ik}+y_{ij}y_{ik}+z_{ij}z_{ik}+M_{jik}d_{ij}d_{ik}}{d_{ij}^4 d_{ik}^4}\right)$$
$$\left(-d_{ij}^2 d_{ik}^2\left(x_{ij}+x_{ik}\right)+x_{ij}x_{ik}\left(x_{ij}d_{ik}^2+x_{ik}d_{ij}^2\right)\right)$$

$$f_{3x} = -2\sum_{\substack{j\ neigh.\ i \\ k\ neigh.\ j}} T_{ijk}\left(\frac{x_{ij}x_{jk}+y_{ij}y_{jk}+z_{ij}z_{jk}-M_{ijk}d_{ij}d_{jk}}{d_{ij}^4 d_{ik}^2}\right)$$
$$\left(-d_{ij}^2\left(x_{jk}+y_{jk}+z_{jk}\right)+x_{ij}\left(x_{ij}x_{jk}+y_{ij}y_{jk}+z_{ij}z_{jk}\right)\right)$$

$$x_{ij} = x_j - x_i$$

$$d_{ij} = \sqrt{x_{ij}^2 + y_{ij}^2 + z_{ij}^2}$$

The term f_{1x} corresponds to forces generated by binary bonds.

The terms f_{2x} and f_{3x} correspond to forces generated by ternary bonds: f_{2x} is the force component resulting from particle i being the central element of a ternary bond *(j,i,k)*, whereas f_{3x} results from i being an extremity of a ternary bond *(i,j,k)*.

Animation may then be obtained through discrete integration, using position and speed values at the preceding time step to determine interaction forces and the current position:

$$\{x(i,t)\} \rightarrow \{f(i,t)\} \rightarrow \left\{\gamma(i,t)=\frac{f(i,t)}{m(i)}\right\}$$
$$\rightarrow \{x(i,t+1)=x(i,t)+v(i,t)+\gamma(i,t)\}$$

This formula models continuous interactions between the particles of the objects. Real shocks between objects may be introduced as ideal shocks involving infinite accelerations. Therefore, as singularities in the acceleration domain, they require handling at a different level in the motion integration process.

3- LEARNING THE MODEL

3-1 Cost functions

Two physical parameter identification methods using the above basis are now considered as follows.

The basic idea is to launch the animation program, using an arbitrary object structure and parameter set, and to record the objects' coordinates from the resulting animation sequence. The learning task consists then in solving the inverse problem: assume an unknown parameter set, and reconstruct the object's mechanical parameters using the recorded animation only (i.e. the coordinates of the particles versus time).

In other terms, our problem is to find out a near-optimal mechanical representation of the system, consisting of a set of parameters (particles masses, linear bonds between particles with attributes of stiffness and length at rest, angular bonds with attributes of angular stiffness and angle at rest) which, if put back into the animation model, would give an object with a behaviour as similar as possible to the original's.

The identification problem may be represented as the optimisation of a cost function:

$$Cost(model) = \sum_{time}\sum_{particles}(dpred_{t,i,j} - dreal_{t,i,j})^2$$

where

- $dreal_{t,i,j}$ is the actual (observed) distance between particles i and j at time step t;
- $dpred_{t,i,j}$ is the distance between the same particles at time step t, as predicted by application of the physical model to actual coordinates and speeds of all

the particles at the preceding time step $(t - 1)$.

The physical model we use in the learning phase (motion analysis) is identical to the model used in the animation (motion synthesis).

The identification algorithms have been developed using the cost function above, to reconstruct the values of springs and torsional springs characteristics. In order to simplify the problem, the structure and masses values are assumed to be already known.

3-2 Simulated annealing

The first algorithm is based on the principle of simulated annealing. The philosophy of the algorithm is to randomly modify all the numerical values of the model at each identification step. It uses a Gaussian noise the standard deviation of which depends on a parameter called *temperature*. The cost function is then evaluated on the new set of parameters and compared to the preceding cost value. If the cost of the new set of values is lower, the new set will replace the old one; if not, a second random choice will be made to decide which set should be retained. The latter choice depends on the difference of costs of the old and new sets: the probability of retaining the new set rapidly decreases with the cost difference. The temperature falls exponentially at each step. The lengths of binary bonds are initialised as the average values of the distances between particles as observed on the image sequence. The other parameters are initialised with arbitrary values.

In our implementation, based on the Metropolis algorithm, slightly better results were obtained through the use of a two-dimensional temperature, allowing alternate annealing of binary bonds (springs) and ternary bonds (torsional springs). Tests on several 50- and 100-image sequences of a very simple object (3 particles, 6 parameters to be identified) have shown that the algorithm converges to good estimates (typically $\varepsilon < 5\%$) of parameters in a majority of cases, but on larger objects (typically from 5 particles) or with extreme values, some quantities converge poorly or not at all. For a given image sequence, the convergence for each parameter depends then unpredictably on both their true values and the initial conditions (temperature, time constant...).

3-3 Evolutionary algorithms

For larger systems a more elaborate scheme is needed. The second method implemented uses an evolutionary algorithm, closely inspired by the 'genetic algorithms' philosophy [G89]. The principle chosen is to launch a small population of models ('*individuals*'), provide them with mutation and crossover rules, giving some sort of advantage to the 'fittest' individuals (those having lower cost function values), and let the population evolve naturally in the hope that at least a part of the population will reach a good approximation of the optimum. The individuals are described by a complete set of parameters of the structure under consideration, this is their '*genetic code*'. The aim is to overcome the limitations of simulated annealing in object size, and at the same time to exploit the empirical fact that simulated annealing may converge to parameter space points which are wrong, but some coordinates of which (i.e. the parameters of some bonds) are however accurately estimated.

The evolutionary process in the genetic algorithm implemented here contains *selection*, *mutations* and *crossovers* at each step.

Individuals are first sorted into order of fitness determined by their cost function at every step. Next a fixed proportion p_c of the individuals is modified by crossovers. Finally a proportion p_m of individuals is modified by mutations.

Coding

Unlike most coding schemes adopted in genetic algorithms ([G89]), the method adopted here codes the individuals with the same numerical values as in the annealing algorithm, i.e. a sequence of real values (rather than a Boolean code). In order to get a good dispersion, the initial population is initialised randomly.

Crossover

In the crossover step, the individuals to be modified (typically p_c = 30% to 60%) are chosen deterministically from the population as the poorer performing individuals. Their genetic content is erased; each ordered couple of values (for example the [length,-stiffness] couple corresponding to one binary bond) is replaced by the corresponding couple of values copied from another individual (parent). The parents are randomly chosen from the population, but with a bias which gives the better individuals a greater chance of being selected and hence a higher probability of propagating elements from their genetic codes.

Selection is thus performed both through this bias and the poorer performing individuals only being modified by the crossover process.

Mutations

After the crossover step, mutations are applied to a proportion p_m of the individuals (typically, p_m = 1% to 5%): for each parameter, one individual is chosen randomly from the worst 95%, and the parameter is modified for *this* individual. The mutation process consists in applying a multiplicative noise with a given standard deviation (or '*temperature*') which may decrease under the control of the cost function. General experience in genetic algorithms, showing that temperature does not play the same role as in annealing and should not be as systematically decreased, is confirmed here. An additional mutation process introduces random values for bond lengths (parameters L) calculated from the observed distances between particles. The mutations, introduced here in a non-classical way because of the continuous nature of parameters, aim (classically) at maintaining a genetic diversity in the population and allow escape from a local optimum far from the solution wanted.

Mutations preserve the 5% most performing individuals: this 'elitist' policy forces the process to retain the best configurations found, and the cost function to keep decreasing with time. Without this feature one could sometimes observe the cost function increase after typically 100 generations and the entire population converge to a poor solution.

Local vs. global cost functions

The normal cost function used in the algorithm encompasses the whole object as described above: the sum is extended to all the object's particles.

One of the most important features of this algorithm is the introduction of alternative ***'local' cost functions.*** To each individual particle, a local cost function is associated, and evaluated from the positions of the first and second neighbours of this particle in the object. We use them in a *local version of the algorithm*:

- at each generation, instead of using a single cost function, the entire collection of cost functions corresponding to each particle is evaluated to get multiple sorting of the population;

- for each particle, local mutation and crossover processes are used, restricted to the bonds which participate to the local cost function; this allows the selection process to be based on local rather than global fitness of individual parameters.
- the global cost function (sum of the local cost functions) is still used to get the final result at each generation.

This exploits the fact that a particle's position at instant *t* is entirely determined by its first- and second-order neighbours' positions and speeds at instant *t*-1, and on the values of the (binary and ternary) bonds it belongs to. The cost function uses an estimation over a single time step, which allows the optimisation process to be based on local properties[1].

4-EXPERIMENTAL RESULTS

Experimental results on our evolutionary algorithms show a good convergence, using the 'local' version of the algorithm (see appendix).

Using a population of 100 to 200 individuals and 5-particle objects (4 binary bonds and 3 ternary bonds, 14 parameters to be identified), convergence is obtained with the global evolutionary algorithm, generally from about 100 generations, with a good success rate and for a mechanical parameter dynamic range significantly greater than when using annealing. On such small objects, the local version is roughly equivalent to the global one, as the neighbourhoods cover nearly all the object: it only requires a greater number of generations, but the total calculations are equivalent.

On more complex objects (the number of particles and bonds is related to the object's complexity rather than to its physical size), the global evolutionary algorithm becomes extremely slow in convergence. With the local version, the number of calculations at each generation increases linearly with the number of bonds, but the number of generations needed no longer varies significantly with the object's complexity. Some results on 8-particle objects (8 binary bonds, 10 ternary bonds, total 36 parameters to identify) are given in appendix.

Fine tuning the mutation process is essential. Generally, ternary bonds parameters are more easily estimated than binary bonds. Introducing binary bond lengths around their observed average values through mutations improves the convergence, and suggests to introduce as much physical knowledge as possible concerning the physical process, into the algorithm.

Typical results are shown in appendix. In order to obtain an objective evaluation of the algorithm, we evaluate the convergence quality using an "external" cost function which is global and calculated on the whole image sequence (1000 images); the identification process's internal cost function only uses a small number of images (typically 50 to 100: lower numbers of time samples in the internal cost function result in loss of performance). In all cases, the external cost function gives a final assessment

[1] If we were using a 'deeper' cost function, e.g. a motion integration over 2 time steps, this would imply extending the neighbourhood up to the 4th order neighbours. Cost functions based on longer-term predictions (rather than the one-step prediction presently used) might help the final convergence of the algorithm but are not compatible with local evolution.

of the ability of the model to extrapolate the motion of the object, which is a better criterion than would be a mere comparison of the mechanical parameters found, against the values used when creating the original image sequence.

5 - CONCLUSION

Our aim is to build particle-based real-time physical animation tools, using mechanical object descriptions obtained from the geometry of motion in real image sequences.

The identification algorithm developed here, converges in a number of generations normally independent of the number of particles involved, which results in the total cost of identification being proportional to the number of particles. This is confirmed by limited scale experiments.

Besides possible applications such as scene understanding in physical terms, or sound and vibration modelling, the main aim of this work is physical model-based animation, enabling the user of an image synthesis workstation to get physically realistic animations by using empirical models built directly from real image data ("Synthetic TV"). Such models, suitable both for real-time animation and (non-real time) identification, may also be of benefit to Virtual Reality applications, allowing to build mechanically realistic behaviours of the objects interacting with the user.

References

[AG85] W.W. Armstrong, M.W. Green, "The Dynamics of Articulated Rigid Bodies for Purposes of Animation", *Proc. of Graphics Interface*, 1985.

[C90] C. Cadoz, "Connaître pour simuler, simuler pour connaître", *Coll. Mod. Phys. Grenoble*, 1990.

[DZ93] P. Dworkin, D. Zeltzer, "A New Model for Efficient Dynamic Simulation", *4th Eurographics Animation and Simulation Workshop*: pp.135-148, 1993.

[G89] D.A.Goldberg, "Genetic Algorithms in Search, Optimization and Machine Learning", Addison-Wesley 1989.

[GV89] M.-P. Gascuel, A. Verroust, "Animation à l'aide de la dynamique: état de l'art". *Rapport de recherche du LIENS*, 89(5), mai 1989.

[GVP91] M.-P. Gascuel, A. Verroust, C. Puech, "A Modelling System for Complex Deformable Bodies Suited to Animation and Collision Processing". *Journal of Visualization and Computer Animation*, 2 (3), August 1991.

[L93] J. Louchet, "A Dynamical Model for Molecule Animation", *Image'Com*, Bordeaux, March 1993.

[LJFCR91] A. Luciani, S. Jimenez. J.L. Florens, C. Cadoz, O. Raoult, "Computational Physics: a Modeller Simulator for Animated Physical Objects", *Proc. Eurographics Conference*, Wien, Sep. 1991, Elsevier.

[MZ90] M. McKenna , D. Zeltzer, "Dynamic Simulation of Autonomous Legged Motions", *Computer Graphics (Siggraph 90)*, 24 (4): pp. 29-38, 1990.

[SC93] N. Szilas, C. Cadoz, "Physical Models That Learn", *International Computer Music Conference*, Tokyo, 1993.

[T90] V. Torres, "Interatomic Potentials and Defect Modelling in Semi-Conductors", *Ph.D. thesis*, King's College London, 1990.

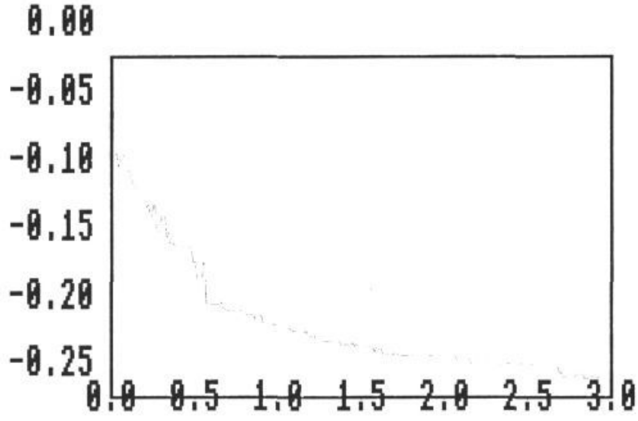
[TPBF87] D. Terzopoulos, J. Platt, A. Barr, K. Fleischer, "Elastically Deformable Models", *Computer Graphics-Siggraph 87*, 21(4): pp. 205-214, 1987.

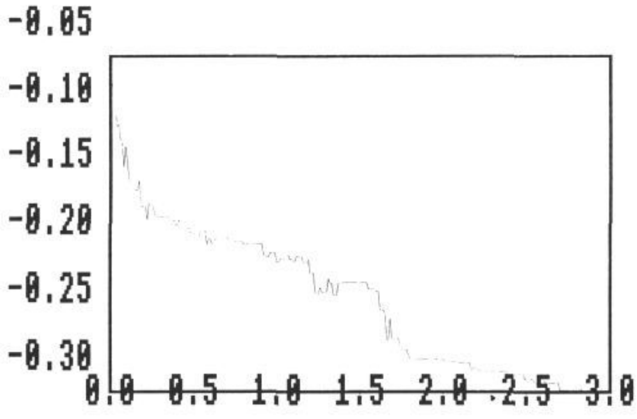
Appendix: Convergence Test Results

The following diagrams show the evolution of the cost function logarithm along the genetic identification processes (local and global) of a 8-particle object (36 parameters), using 80 time samples and 150 individuals. The cost function shown is calculated on the whole image sequence (1000 images), and is different from the internal cost function used in the optimization process, which uses a small number of time samples from the sequence (typically 50 to 100).

The x-axis represents the number of generations (in hundreds).

The y-axis represents the tenth of the logarithm of the cost function calculated from the whole sequence. The curve measures how the model found is able to extrapolate motion.





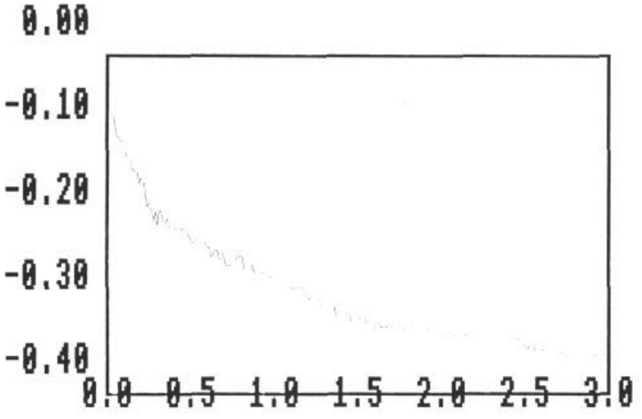
```
75  1.232e-02    0.158  1.033    0.227  1.004    0.112  0.941    0.108  0.609    -1.015 0.937    -1.092 0.678    -1.198 1.099
129 1.228e-02    0.141  1.023    0.212  0.990    0.112  0.942    0.108  0.608    -1.012 0.937    -1.103 0.663    -1.198 1.081
101 1.227e-02    0.158  1.033    0.212  0.973    0.115  0.942    0.110  0.604    -1.013 0.937    -1.104 0.667    -1.198 1.099
10  1.224e-02    0.158  1.034    0.212  1.004    0.112  0.958    0.116  0.609    -1.002 0.952    -1.103 0.670    -1.198 1.088
94  1.224e-02    0.158  1.033    0.212  1.004    0.111  0.941    0.101  0.604    -1.012 0.945     1.103 0.670    -1.198 1.088
50  1.219e-02    0.158  1.031    0.212  1.004    0.119  0.941    0.110  0.608    -1.012 0.945    -1.103 0.667    -1.198 1.088
136 1.215e-02    0.158  1.031    0.212  1.004    0.119  0.941    0.108  0.609    -1.013 0.937    -1.103 0.667    -1.198 1.088
18  1.206e-02    0.143  1.052    0.213  1.004    0.112  0.941    0.110  0.609    -1.017 0.937    -1.103 0.670    -1.198 1.099
26  1.170e-02    0.158  1.034    0.212  1.004    0.116  0.941    0.111  0.604    -1.012 0.945    -1.103 0.670    -1.198 1.099
93  1.155e-02    0.158  1.034    0.227  1.004    0.119  0.941    0.110  0.608    -1.015 0.952    -1.103 0.670    -1.198 1.081
contrib    2.70%          7.67%           6.06%           8.77%           2.94%     worst = no. 3;
real values     0.100  1.000    0.200  1.000    0.150  1.000    0.100  0.600    -1.000 1.000    -1.000 0.700    -1.300 1.000
error factor    1.580  1.034    1.133  1.004    1.263  1.063    1.102  1.013    1.015  1.050    1.103  1.045    1.086  1.081
generation no 300   temperature=0.300470 cost function calculated on 80 images  FLOI = 4 coeff= 10000.000
solution found: position error: 2.1809e-06 mutual distance error 0.0010156, log = -2.993294     local alg. 1
```

The final data above show some numerical results (not all) corresponding to the preceding graph:

- the first 10 lines are the internal cost and parameter values found for the 10 best individuals;
- 11th line gives each particle's contribution to the total cost of the best individual;
- 12th line gives the true values originally used for the animation;
- 13th line, the error factors between true and found values;
- 14th line, some parameters used in the identification algorithm;
- 15th line, the final error calculated as the sum (on the whole sequence) of squares of differences between predicted and real positions/mutual distances between particles.

The first column contains local cost values for the ten individuals; the 10 next columns display lengths and stiffness of 5 binary bonds, the 6 last columns contain the parameters from 3 ternary bonds. The object used in the test contains 8 binary bonds and 10 ternary bonds (36 parameters to identify).

Here is another typical test result, on a different object with the same number of particles.


0.00
-0.10
-0.20
-0.30
-0.40
0.0
0.5
1.0
1.5
2.0
2.5
3.0