ECLeS: A Flexible and General Method for Local Editing of Parameters with Linear Constraints

Elisa de Cássia Silva Rodrigues, Jorge Stolfi
Institute of Computing, University of Campinas (UNICAMP)
Campinas, Brazil
Email: {erodrigues, stolfi}@ic.unicamp.br

***Abstract*—We describe ECLeS, a general method for interactive editing of objects that are defined by real parameters subject to linear or affine constraints. In this method, the constraints and the user editing actions are combined using weighted constrained least squares instead of the usual finite element approach. We use exact integer arithmetic in order to detect and eliminate redundancies in the set of constraints and to avoid failures due to rounding. We are using this technique for the user-friendly editing of C1-continuous deformations of the plane, defined by splines of degree 5 on an arbitrary triangular mesh.**

***Keywords*-parameter editing; linear constraints; exact integer arithmetic; deformations.**

I. Introduction

Many computer applications require the interactive adjustment of parameters subject to linear or affine constraints. Typically the user adjusts the value of some parameters, and the applications is supposed to adjust other parameters in order to preserve the constraints. An example of such applications is the manual editing of spline surfaces or spline deformations subject to smoothness constraints. In this paper, our goal is to define mathematical and software tools for this task in a general, robust, and efficient way.

Contributions: This paper proposes the interactive and general method for local editing of parameters with linear constraints, called ECLeS (Editing by Constrained Least Squares). At each editing action, the user specifies a subset of the parameters that can be changed, new required value for some of them, and hint (desirable but not required) values for the others. Our algorithm uses exact integer arithmetic to detect and eliminate redundancies among the constraints that are relevant to that set and to avoid failures due to rounding. The parameters are then adjusted so as to satisfy all constraints and required values, and be as close as possible to the hints. We use weighted and constrained least squares instead of the usual finite element approach. Thanks to the use of fraction-free methods in the linear system solving, we are able to avoid the bit size explosion that is often a serious limitation of this type of algorithm.

A. Related work

There are many systems for editing objects under linear and affine constraints. Typically, they recompute all parameters at each user editing request, and use penalty terms in order to minimize the changes to non-edited parameters.

Knuth's 1979 Metafont [1] is a programming language where function parameters may be subjected to linear and affine constraints. It lets the user specify any subset of independent parameters, automatically solving for the others when enough parameters have been specified. In 1985, Nelson [2] described Juno, a constraint-based graphics system. Later, this system was extended to the drawing editor, Juno-2, by Heydon and Nelson [3]. With Juno-2 the user can define constraints graphically, which are solved by a non-linear equation solver combined with some symbolic techniques. The user could define hints for unknown parameters.

For linear and affine constraints (which we consider here), a common approach is to pre-compute a *finite element basis* of parameter change vectors that has small support and spans the space of all possible changes; and then give the user a separate "knob" for each finite element. A typical example is the editing of splines with specified continuity. Each basis element has exactly one editable Bézier control point, the other Bézier points are then computed from those. This problem was extensively studied by Schumaker [4] and others; an exposition can be found in Xu's doctoral thesis [5]. One particular case of the finite element technique is the B-spline approach [6], where there is only one control point for each patch, and the resulting spline is automatically continuous to the maximum possible order. B-splines are well defined for tensor (quadrangular or hexahedral) type meshes with regular topology [7], [8]; extending them to irregular and simplicial meshes is possible, but rather complicated [9].

Another technique for editing the Bézier control points (which we adapt and extend in this paper) uses the criterion of least squares with first-degree constraints [10]. This technique was used, for instance, by Masuda et al. [11] for surface mesh editing. It allows editing any control point and computes the remaining points by solving a linear system that combines the criterion of least squares and the constraint equations.

Our method differs from Masuda's by assuming that the application provides, at each editing event, a small subset of parameters that can be changed. Moreover, we use exact arithmetic to reliably detect inconsistent or redundant constraints.

B. Overview of the method

The general method described in this paper operates on a set $\mathcal{P}$ of variables (the *control parameters*) of an application, subject to a fixed set $\mathcal{R}$ of linear equations with integer

coefficients (the *constraints*). We assume that the set $\mathcal{R}$ of all equality constraints is denoted by a matrix equation

$$RP = Q \tag{1}$$

where P is the vector of all control parameters, R is a constant coefficient matrix and Q a constant vector (often zero).

The ECLeS method is used when the user has attempted modify some of the control parameters through some editing software (the *user interface*). Its goal is to adjust other control parameters so that all constraints remain satisfied.

The method consists of two procedures, *Initialize* and *Update*. The *Initialize* procedure is called when the user chooses the parameters to be adjusted. The *Update* procedure is then called one or more times to modify those parameters.

Specifically, the *Initialize* procedure receives the set $\mathcal{R}$ and two disjoint subsets of $\mathcal{P}$:

- $\mathcal{A}$ (*anchor*): one or more control parameters whose values will be set by the user;
- $\mathcal{D}$ (*derived*): zero or more control parameters that may be adjusted if necessary to satisfy the constraints.

The procedure then identifies the set $\mathcal{E}$ of *relevant constraints* consisting of the equations of $\mathcal{R}$ that depend on any parameter in $\mathcal{A} \cup \mathcal{D}$. The set $\mathcal{F}$ is the set of all *fixed relevant* parameters, that occur in some equation of $\mathcal{E}$ but are not in $\mathcal{A}$ or $\mathcal{D}$.

The set $\mathcal{D}$ of derived parameters must be large enough to allow all the relevant constraints in $\mathcal{E}$ to be satisfied. That is, for any combination of new values assigned to the control parameters in $\mathcal{A}$ it must be possible to satisfy all constraints in $\mathcal{E}$ (and therefore in $\mathcal{R}$) by assigning appropriate new values to all control points in $\mathcal{D}$. We refer to this requirement as the *solvability condition*. If this condition is not satisfied, the *Initialize* procedure fails and the user interface is expected to take appropriate action. Otherwise, *Initialize* pre-computes certain matrices and other information to be used by the *Update* procedure.

For each control parameter p in $\mathcal{A} \cup \mathcal{D}$ the *Update* procedure receives a new value p'. For the parameters in $\mathcal{A}$, the new values are specified by the user and assumed to be mandatory. For the parameters in $\mathcal{D}$, the new values are only suggestions. The *Update* procedure then computes, for each control parameter p in $\mathcal{D}$, a new value p'' that is close or equal to p', in such away that all constraints are satisfied. The procedure then updates each parameter p with the new values p' or p''. For every parameter $p \notin \mathcal{A} \cup \mathcal{D}$, the desired value p' and final value p'' are assumed to be equal to the current value p.

II. Solving Linear Systems

In this section, we describe the technique used to solve linear systems with integer coefficients. We assume in general that the system is

$$Ax = b \tag{2}$$

where A is a known rectangular $m \times n$ integer matrix, x is an unknown column vector of n rational numbers, and b is a known vector of m integers. We need to solve the system exactly in order to check for linearly dependent equations.

A straightforward implementation of the LU decomposition algorithm with exact rational arithmetic is very inefficient because the sizes of the numerators and denominators grow exponentially with the number of equations m. This problem was solved by the so called fraction-free method of E. H. Bareiss [12]. This method was extended by others authors [13], [14], [15], [16].

Most of these methods are restricted to square invertible or full-rank matrices. In this paper, we use the method proposed by D. J. Jeffrey [17] which can perform fraction-free Gaussian LU factoring of arbitrary rectangular matrices of any rank.

A. Factoring the matrix

Specifically, let r be the rank of A. Factoring the rectangular matrix A gives five integer matrices L ($m \times r$), U ($r \times n$), D ($r \times r$, diagonal), T_r ($m \times m$, permutation matrix of rows) and T_c ($n \times n$, permutation matrix of columns) such that

$$A = T_r L D^{-1} U T_c. \tag{3}$$

In this equation, the structures of the matrices L and U are

$$L = \begin{pmatrix} \hat{L} \\ \widetilde{L} \end{pmatrix} \quad \text{and} \quad U = \begin{pmatrix} \hat{U} & \widetilde{U} \end{pmatrix} \tag{4}$$

where $\hat{L}$ is an $r \times r$ lower triangular matrix, $\hat{U}$ is an $r \times r$ upper triangular matrix, both invertible, and $\widetilde{L}$, $\widetilde{U}$ are arbitrary integer matrices with sizes $(m-r) \times r$ and $r \times (n-r)$, respectively.

B. Solving the system

Substituting formulas (3) and (4) into system (2), we have

$$T_r \begin{pmatrix} \hat{L} \\ \widetilde{L} \end{pmatrix} D^{-1} \begin{pmatrix} \hat{U} & \widetilde{U} \end{pmatrix} T_c x = b. \tag{5}$$

Letting $B = T_r^{-1} b$, $X = T_c x$ and $y = UX$, equation (5) becomes

$$\begin{pmatrix} \hat{L} \\ \widetilde{L} \end{pmatrix} D^{-1} y = \begin{pmatrix} \hat{B} \\ \widetilde{B} \end{pmatrix} \tag{6}$$

where $\hat{B}$ and $\widetilde{B}$ consist of the first r and last $m-r$ elements of B, respectively. We can split the system (6) into two systems

$$\hat{L} D^{-1} y = \hat{B} \quad \text{and} \quad \widetilde{L} D^{-1} y = \widetilde{B}. \tag{7}$$

Since $\hat{L} D^{-1}$ is an $r \times r$ invertible matrix, we can solve the first system for y

$$y = D \hat{L}^{-1} \hat{B}. \tag{8}$$

The second system $\widetilde{L} D^{-1} y = \widetilde{B}$ in equation (7) is non-empty only if the rank r of A is less than the number of rows m, in which case either some constraints are redundant, or there are incompatible constraints. To verify whether the original system (2) is consistent, it is necessary and sufficient to check if the bottom half of the equation (6) is satisfied with this value y, that is

$$\widetilde{L} \hat{L}^{-1} \hat{B} = \widetilde{B}. \tag{9}$$

To solve (5), we can find X by solving the system

$$\begin{pmatrix} \hat{U} & \widetilde{U} \end{pmatrix} \begin{pmatrix} \hat{X} \\ \widetilde{X} \end{pmatrix} = y \tag{10}$$

that is, by solving

$$\hat{U}\hat{X} = y - \widetilde{U}\widetilde{X} \tag{11}$$

where $\widetilde{X}$ can be chosen arbitrarily. Setting $\widetilde{X} = 0$, we get

$$\hat{X} = \hat{U}^{-1}y \tag{12}$$

and then

$$x = T_c^{-1}X. \tag{13}$$

III. Constrained Least Squares

The other main tool that we use is the quadratic optimization (least squares) method with affine constraints, that we describe in this section. Given a vector $x' = (x'_1, ..., x'_n)$ of desired values, we want to find a vector $x = (x_1, .., x_n)$ of values that minimizes the distance between each new value x_s and the desired value x'_s, while satisfying a set of constraints $Ax = b$, where A is any $m \times n$ matrix and b is a vector of m elements. More precisely, we want minimize the goal function

$$S(x) = \sum_{s=1}^{n} w_s[x_s - x'_s]^2 \tag{14}$$

where the coefficient w_s is a user defined weight that indicates the importance of the desirable value x'_s (the bigger the weight value w_s is, compared to the other weights, the more the algorithm will try to honor the value x'_s).

We use the technique of section II-A to perform Gaussian factoring of the matrix A and the vector b. If this process terminates with success, we obtain the rank r and integer matrices U, L, D and permutation matrices T_r and T_c, such that $A = T_r L D^{-1} U T_c$. If $r < m$, the system $Ax = b$ has redundant equations, and can be replaced by

$$\hat{A}x = \hat{B} \tag{15}$$

where $\hat{A}$ is the first r rows of $LD^{-1}UT_c$ and $\hat{B}$ is the first r rows of $T_r^{-1}b$.

In order to minimize the goal function (14) while satisfying the constraints (15), it is necessary that the gradient ∇S of the goal function be perpendicular to the solution set of the constraints. The gradient consists of the derivatives

$$\frac{\partial S}{\partial x_s} = 2w_s[x_s - x'_s] \tag{16}$$

for $s = 1, 2, \ldots, n$. To be perpendicular to the constraint solution space, the gradient ∇S must satisfy

$$\frac{\partial S}{\partial x_s} = -\sum_{k=1}^{n} \lambda_k \hat{A}_{ks} \tag{17}$$

where each variable λ_k is an indeterminate real coefficient, the *Lagrange multiplier* [18] of the constraint expressed by row k of the system (15). Therefore

$$2w_s x_s + \sum_{k=1}^{n} \lambda_k \hat{A}_{ks} = 2w_s x'_s. \tag{18}$$

Equation (18) can also be written in matrix form

$$Mx + \hat{A}^\top \lambda = Mx' \tag{19}$$

where M is the $n \times n$ diagonal matrix with $M_{ss} = 2w_s$; and λ is a column vector with the Lagrange multipliers $\lambda_1, ..., \lambda_m$.

We can combine equation (19) with the constraints (15) obtaining the *least squares linear system*

$$Mx = Mx' - \hat{A}^\top \lambda \tag{20}$$

$$\hat{A}x = \hat{B}. \tag{21}$$

The new values x of the variables can be computed by solving the system (21) in two steps; namely, solving

$$\hat{A}M^{-1}\hat{A}^\top \lambda = \hat{A}x' - \hat{B} \tag{22}$$

for λ, and then computing x by solving

$$Mx = Mx' - \hat{A}^\top \lambda. \tag{23}$$

IV. Details of the ECLeS Method

In this section, we describe in detail the *Initialize* and *Update* procedures of the ECLeS method.

We can partition the matrix R, and the vectors P and Q into sub-matrices and sub-vectors according to their rows and/or columns being in $\mathcal{E}$, $\mathcal{A}$, $\mathcal{D}$ or $\mathcal{F}$, and then write the relevant constraint equations of $\mathcal{E}$ in matrix form $Ax = b$, where $A = R_{\mathcal{E}\mathcal{D}}$, $x = P''_{\mathcal{D}}$ and $b = Q_{\mathcal{E}} - R_{\mathcal{E}\mathcal{A}}P'_{\mathcal{A}} - R_{\mathcal{E}\mathcal{F}}P_{\mathcal{F}}$.

The matrices $R_{\mathcal{E}\mathcal{D}}$, $R_{\mathcal{E}\mathcal{A}}$ and $R_{\mathcal{E}\mathcal{F}}$ are $m \times n$, $m \times s$ and $m \times t$ matrices of coefficients of the derived parameters $\mathcal{D}$, anchor parameters $\mathcal{A}$ and the fixed relevant parameters $\mathcal{F}$ in the equations $\mathcal{E}$, respectively. The vector $P''_{\mathcal{D}}$ will consist of the n computed values p'' of the derived parameters. The vector $Q_{\mathcal{E}}$ contains the m elements of the constant vector Q corresponding to the equations $\mathcal{E}$. The vector $P'_{\mathcal{A}}$ contains the user-specified values p' of the s anchor parameters; and $P_{\mathcal{F}}$ is a vector with the values p of the t fixed relevant parameters. Each row of $R_{\mathcal{E}\mathcal{D}}$ is a subset of the elements of some row of R, corresponding to the $\mathcal{D}$ parameters. The same row of $R_{\mathcal{E}\mathcal{A}}$ consists of the elements of that row of R corresponding to the anchors. Finally, the same row of $R_{\mathcal{E}\mathcal{F}}$ consists of the elements of that row of R corresponding to the $\mathcal{F}$ parameters.

A. The Initialize procedure

The *Initialize* procedure receives the sets $\mathcal{A}$, $\mathcal{D}$ and $R_{\mathcal{E}\mathcal{D}}$. It then uses the method described in section II-A to isolate redundant or inconsistent equations and obtain the system (15).

B. The Update procedure

If *Initialize* succeeds, the values $x = P''_{\mathcal{D}}$ of the derived parameters are computed by the *Update* procedure, each time the suggested values $x' = P'_{\mathcal{D}}$ are given by the user interface.

When the solvability condition (9) of the system is not satisfied, choices of the anchor parameters by the user must have included incompatible constraints in the original system. In this case, the *Update* procedure returns a message to user interface notifying that the specified anchor parameter values are not valid, that is, the solvability condition is not satisfied.

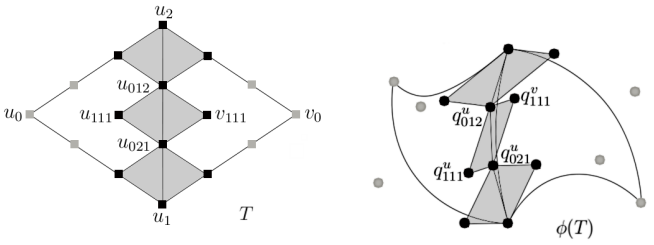
Then, for example, the procedure cancels the editing action and the user must select a new set of anchors.

The *Update* procedure solves the least squares system, as described in section III, equations (22) and (23), obtaining the new computed values $P''_{\mathcal{D}}$ for the derived parameters.

V. Application: 2D Deformation

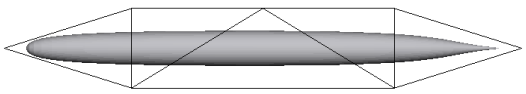
We are using the ECLeS method as part of an editor for 2D deformations that are defined by triangular splines of degree 5 with C^1 smoothness constraints [19]. A deformation is edited by displacing the Bézier control points of the spline. The constraints are quadrilateral conditions which represent C^1 continuity constraints. If the coordinates of the vertices of the domain mesh are rational, each quadrilateral constraint can be expressed by a linear equation with integer coefficients.

In Fig. 1, each C^1 continuity constraint is represented by a gray quadrilateral. The quadrilateral formed by the control points $q^u_{111}, q^u_{012}, q^u_{021}, q^v_{111}$ on the deformed mesh must be an affine image of the quadrilateral formed by their nominal positions $u_{111}, u_{012}, u_{021}, v_{111}$ on the original mesh. Fig. 2 shows a deformation specified by the preliminary version of our editor.

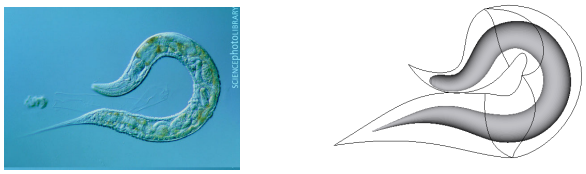


(a) Original mesh. (b) Deformed mesh.

Fig. 1. Quadrilateral C^1 continuity constraints for the Bézier control points of a spline deformation ϕ of degree 3.



(a) Original mesh.



(b) Actual image [20]. (c) Deformed mesh.

Fig. 2. Example of deformation specified by our editor.

VI. Conclusion

In this paper, we described the general ECLeS method for interactive editing of parameters subject to linear or affine constraints. We use exact integer arithmetic in order to detect and eliminate redundancies among constraints and avoid rounding failures. In the ECLeS algorithm, the constraints and the user editing actions are combined using weighted constrained least squares, instead of the usual finite element approach, thus providing more flexible control to the user.

Acknowledgment

This work is supported by Brazilian government grants CNPq 140780/2013-0 and 301016/92-5 (NV).

References

[1] D. E. Knuth, *Tex and Metafont, New Directions in Typesetting*. Stanford: American Mathematical Society and Digital Press, 1979.

[2] G. Nelson, "Juno, a Constraint-based Graphics System," *SIGGRAPH Comput. Graph.*, vol. 19, no. 3, pp. 235–243, Jul. 1985. [Online]. Available: http://doi.acm.org/10.1145/325165.325241

[3] A. Heydon and G. Nelson, "The Juno-2 Constraint-Based Drawing Editor," in *Technical Report 131a, Digital Systems Research*, 1994.

[4] M.-J. Lai and L. L. Schumaker, *Spline Functions On Triangulations*. New York, NY, USA: Cambridge University Press, 2007.

[5] D. Xu, "Incremental Algorithms for the Design of Triangular-based Spline Surfaces," PhD in Computer and Information Science, Faculties of the University of Pennsylvania, Philadelphia, PA, USA, 2002.

[6] G. Farin, *Curves and Surfaces for CAGD: A Practical Guide*, 5th ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2002.

[7] C. de Boor, "Splines as Linear Combinations of B-splines. A Survey," New York, N, USA, pp. 1 – 47, 1976.

[8] Y. Cui and J. Feng, "Technical Section: Real-time B-spline Free-form Deformation via GPU acceleration," *Comput. Graph.*, vol. 37, no. 1-2, pp. 1–11, Feb. 2013. [Online]. Available: http://dx.doi.org/10.1016/j.cag.2012.12.001

[9] Y. He, X. Gu, and H. Qin, "Fairing Triangular B-splines of Arbitrary Topology," in *In Proceedings of Pacific Graphics 05*, 2005, pp. 153 – 156.

[10] W. H. Press, *Numerical Recipes: The Art of Scientific Computing*, 3rd ed. Nova Iorque, NY, EUA: Cambridge University Press, set 2007.

[11] H. Masuda, Y. Yoshioka, and Y. Furukawa, "Interactive Mesh Deformation Using Equality-constrained Least Squares," *Comput. Graph.*, vol. 30, no. 6, pp. 936–946, Dec. 2006. [Online]. Available: http://dx.doi.org/10.1016/j.cag.2006.08.012

[12] E. H. Bareiss, "Sylvester's Identity and Multistep Integer-preserving Gaussian Elimination," *Mathematics of Computation*, vol. 22, pp. 565–565, 1968.

[13] P. R. Turner, "A Simplified Fraction-free Integer Gauss Elimination Algorithm," Tech. Rep., 1995.

[14] G. C. Nakos, P. R. Turner, and R. M. Williams, "Fraction-free Algorithms for Linear and Polynomial Equations," *SIGSAM Bull.*, vol. 31, no. 3, pp. 11–19, Sep. 1997. [Online]. Available: http://doi.acm.org/10.1145/271130.271133

[15] W. Zhou and D. Jeffrey, "Fraction-free Matrix Factors: New Forms for LU and QR Factors," *Frontiers of Computer Science in China*, vol. 2, no. 1, pp. 67–80, 2008. [Online]. Available: http://dx.doi.org/10.1007/s11704-008-0005-z

[16] D. Dureisseix, "Generalized Fraction-free LU Factorization for Singular Systems with Kernel Extraction," *Linear Algebra and its Applications*, Jul. 2011. [Online]. Available: http://dx.doi.org/10.1016/j.laa.2011.06.013

[17] D. J. Jeffrey, "LU Factoring of Non-invertible Matrices," *ACM Commun. Comput. Algebra*, vol. 44, no. 1/2, pp. 1–8, Jul. 2010. [Online]. Available: http://doi.acm.org/10.1145/1838599.1838602

[18] C. E. Pearson, *Handbook of Applied Mathematics: Selected Results and Methods*, 2nd ed. New York, NY, US: Van Nostrand Reinhold, 1990.

[19] E. C. S. Rodrigues, A. Gomide, and J. Stolfi, "A User-editable C^1-Continuous 2.5D Space Deformation Method For 3D Models," *Electronic Notes in Theoretical Computer Science*, vol. 281, no. 0, pp. 159 – 173, 2011, selected papers of the 2011 Latin American Conference in Informatics (CLEI). [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1571066111001812

[20] S. P. Library, "LM of the Nematode Worm, Caenorhabditis Elegans," available at: http://www.sciencephoto.com/media/366424/enlarge. Accessed on Jun 29, 2015.