

Visualization of Numerical Data in GeoGebra

Petr Volný¹, Jana Volná², Dana Smetanová³

¹VŠB – Technical University of Ostrava

²Tomáš Baťa University in Zlín

³The Institute of Technology and Businesses in České Budějovice

Abstract

The possibilities of a visualization of numerical data in a free computer algebra system GeoGebra are studied. The tools for making interpolation and approximation curves of given discrete points are presented.

Keywords: GeoGebra, interpolation, approximation, splines

Modelling of curves

The aim of the paper is not to give a full exposition devoted to modelling curves in computer graphics but we only present a brief survey of a general theory (Alexandr 1999; Farin 2001; Ježek 2006; Sochor et al. 1996) and describe the procedure of making tools for visualization of numerical data in GeoGebra (GeoGebra; GeoGebraWiki).

The curves used most commonly in computer graphics are the polynomial curves, above all cubics. Cubics are the polynomial curves of the 3rd degree. They have many advantages, cover a sufficient variety of shapes, are easily manipulated, and their calculation is easy, etc. We often meet the curves that are piecewise polynomial; among them we include the spline curves. A spline curve is composed of polynomials of the k^{th} degree. For the spline curve it is true that there is guaranteed continuity up to the derivative of the order $k - 1$ at the connecting points of single segments. B-spline (basis spline) curve of degree k can be expressed by the following linear combination

$$P(t) = \sum_{i=0}^n P_i N_i^k(t),$$

where P_i are the position vectors of the control points and $N_i^k(t)$ are the normalized basic functions of the degree k (polynomials).

Methods that allow the user to simply specify curve or area with a predictable shape have been searched. The user usually has the task to enter only a few control points and to create a curve with given properties.

Let's say we have a finite number of control points (control polygon). We want to construct a curve that is as simple as possible and as accurately as possible replaces the control points. We distinguish between two basic approaches. In the first approach, we demand to the curve passes through the control points; we are talking about an interpolation. In the second approach, the curve may not pass through the control points, it only approaches to them in some sense; we are talking about an approximation. Both approaches have their advantages and disadvantages; it depends on the specific situation which approach can be used. Except of the control points it is possible to add a requirement on derivatives (continuity up to the order k) or so-called weight (measured data with smaller deviations have a more considerable weight) at these points.

In some situation we need to simplify a complex function. In such a case we take a few control points lying on the graph of the function and treat them as control points for interpolation or approximation curve.

The often requested properties of curves include:

- invariance with respect to the affine transformation or projection,
- convex hull property – curve lies in a convex hull of control points
- local property – changing the position of a single control point takes affect only on a part of the curve,
- curve may pass through the border points.

Let us focus on the interpolation curves in more detail. The common property of all interpolation curves is that the curves pass through the points of a control polygon. We distinguish between the following types of interpolation:

- Lagrange polynomial – take a function with $n + 1$ control points; Lagrange polynomial is the only polynomial of degree no more than n passing through all control points.
- Newton polynomial – a modification of Lagrange polynomial, it moderates some disadvantages of Lagrange polynomial.
- Hermite polynomial – the function values and the values of the derivatives up to the order k are specified at the control points.

An interpolation spline is composed from several smaller interpolation polynomials. The degree of the polynomials is not too high. But new problems arise, e.g. how to connect each single segment, interpolation polynomial so that the resulting interpolation spline was, let's say, optimal (continuously connected segments, same derivatives at the control points, etc.). E.g. the linear interpolation spline joins neighbouring control points by the lines.

E. Cattmull and R. Rom in one of their articles have shown that in the mathematical description of B-spline curves one can replace the control points by a parametric function of t , and this particular function has the same interpolation properties as the control points. This class of B-spline curves is called

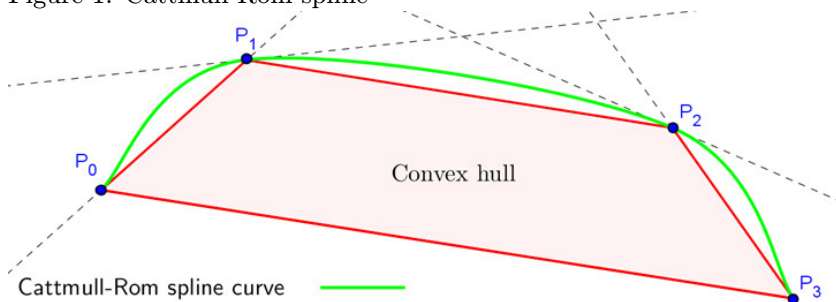
Cattmull-Rom splines. These splines are used to define the distance between objects in computer animation (Alexandr 1999).

The Cattmull-Rom spline curve is composed of several parts (segments); each of these parts is represented by four control points of the leading polygon. Let's have four points P_0, P_1, P_2, P_3 . The vector equation for the parametric expression of the curve has the form

$$Q(t) = \frac{1}{2}(t^3, t^2, t, 1) \begin{pmatrix} -1 & 3 & -3 & 1 \\ 2 & -5 & 4 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 \end{pmatrix} \begin{pmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{pmatrix}.$$

This curve starts at P_1 and ends at P_2 . Tangent vector at P_1 is parallel with the vector $P_0 - P_2$, tangent vector at P_2 is parallel with the vector $P_1 - P_3$. For a given set of control points $P_0, P_1, \dots, P_n$ we construct Cattmull-Rom spline for every consecutive quadruple of control points. If we require to the curve starts at P_0 and ends at P_n , we will have to add the initial and the final control point to the given set of control points, i.e. $P_0, P_0, P_1, \dots, P_{n-1}, P_n, P_n$. The disadvantage of such curves is that the resulting spline generally does not lie in the convex hull of its control points.

Figure 1: Cattmull-Rom spline



Source: own

Approximation curves (approximation – curve fitting) are another important class of curves. The approximation curve may not pass through the control points. We most often use polynomial functions for an approximation. The degree of a polynomial generally raises with increasing number of control points thus the calculations needed for finding approximations of curve become more time-consuming. Using splines is quite common for the creation of an approximation curve. The leading polygon is divided into several smoothly (or continuously up to the order k) connected smaller curves of a lower degree.

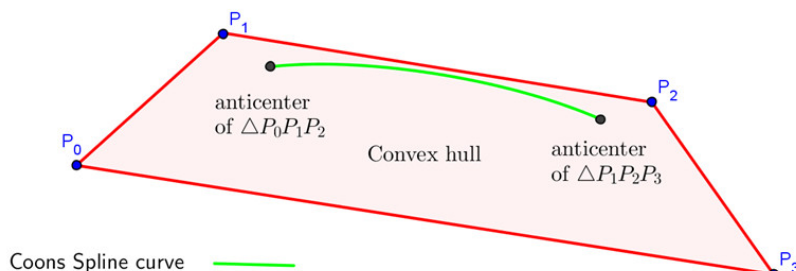
Most commonly, we use cubic splines in which the approximation curve is composed from cubic polynomials (polynomials of the 3rd degree). Each cubic polynomial is controlled by a quadruple of four consecutive leading points. The concrete examples are Bezier cubics or Coons cubics. The further examples are

Ferguson cubics or Hermite cubics, both of them are controlled by a pair of two leading points and by two tangent vectors at them.

The Coons cubics have been used quite often for generating B-spline curves. Any of these cubics doesn't start or doesn't end at the leading points. It starts or ends in a so-called anticenter of a triangle intended of first or last three leading points of a leading polygon with at least four vertices. The anticenter of triangle $P_0P_1P_2$ is a point located at $P_0 + (2/3)(P_1 - P_0) + (1/6)(P_2 - P_0)$ (Alexandr 1999).

The advantage of this kind of approximation is that we can guarantee continuity up to the order 2 at the connecting points. If we change the position of one of the leading points, the curve will change only locally. The curve lies in its convex hull. Coons B-spline curves are used for example in the aviation industry.

Figure 2: Coons spline



Source: own

GeoGebra

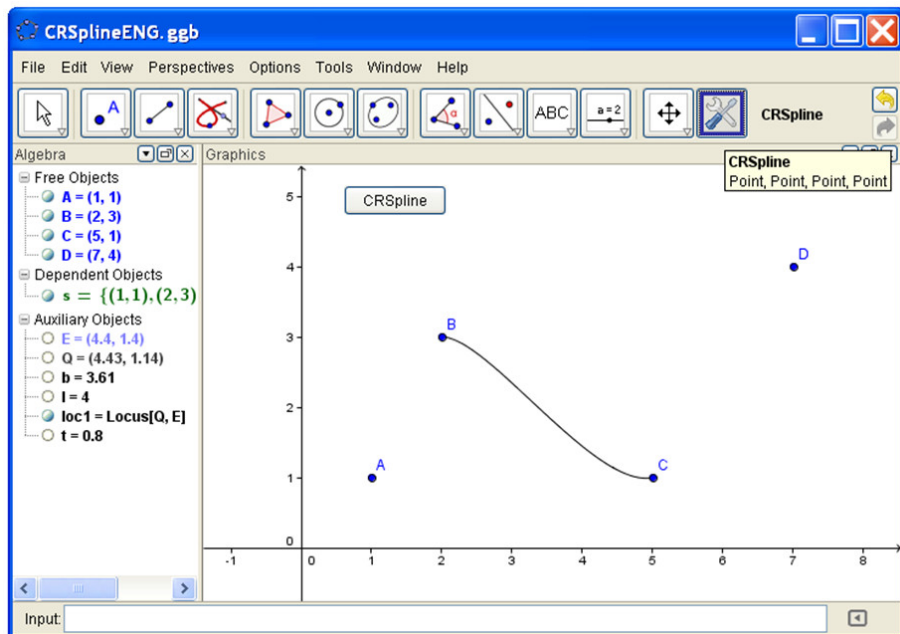
GeoGebra is a great free tool for learning and education of mathematics, statistic and descriptive geometry. It can be used at every level of the school education. GeoGebra provides us with interactive graphics, algebra and spreadsheets. GeoGebra is developed by many people around the world. For software, manuals, tips and tricks we refer to (GeoGebra; GeoGebraWiki).

Our aim is to discuss the possibilities of visualization of numerical data in GeoGebra. If we have a set of given numerical data, will it be possible to connect or represent these points by a smooth curve in GeoGebra? We create two tools, one for interpolation and the second one for approximation curves of discrete points.

Interpolation tool called CRSpline

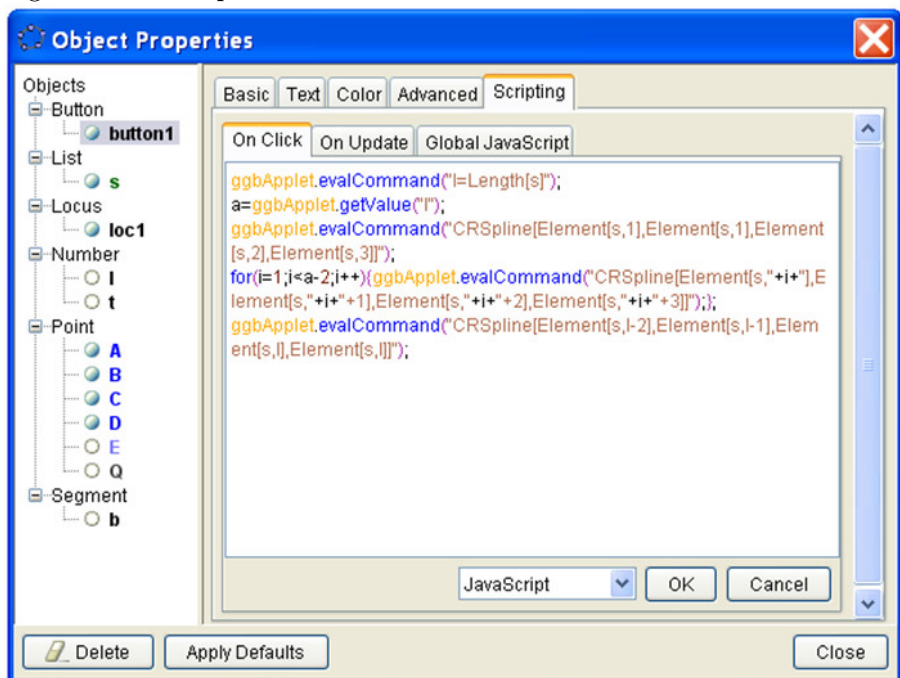
We create a tool which interpolates set of four given discrete points by Cattmull-Rom spline. We use the vector equation from the preceding section. We create a button in GeoGebra which can be used for interpolation of an arbitrary finite set of discrete points by Cattmull-Rom spline. This button is based on CRSpline tool.

Figure 3: CRSpline tool



Source: own

Figure 4: JavaScript code for button1



Source: own

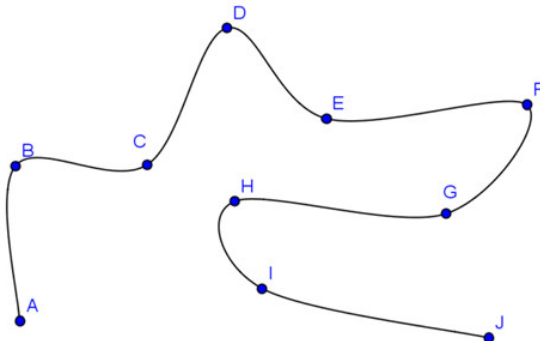
Table 1: Step by step construction of Cattmull-Rom spline in GeoGebra

1.	Input:	Insert the point $A=(1,1)$ into the input bar. Analogously insert the points: $B=(2,3)$, $C=(5,1)$, $D=(7,4)$.
2.		Connect points B and C using tool Segment between two points.
3.		Insert an arbitrary point E on the segment BC using tool New point.
4.	Input:	Compute relative distance of E from B with respect to the length b. Insert $t=\text{Length}[E-B]/\text{Length}[C-B]$. If E belongs to segment BC, then parameter t is set from 0 to 1.
5.	Input:	Define the point Q lying on the Cattmull-Rom spline for the parameter t. Insert $Q=(\text{Element}[1/2 \text{ Transpose}[\{\{t^3\}, \{t^2\}, \{t\}, \{1\}\}]\{\{-1, 3, -3, 1\}, \{2, -5, 4, -1\}, \{-1, 0, 1, 0\}, \{0, 2, 0, 0\}\}\{x(A), \{x(B)\}, \{x(C)\}, \{x(D)\}\}, 1, 1), \text{Element}[1/2 \text{ Transpose}[\{\{t^3\}, \{t^2\}, \{t\}, \{1\}\}]\{\{-1, 3, -3, 1\}, \{2, -5, 4, -1\}, \{-1, 0, 1, 0\}, \{0, 2, 0, 0\}\}\{y(A), \{y(B)\}, \{y(C)\}, \{y(D)\}\}, 1, 1))$ (using vector equation from the previous theory)
6.		Define the Cattmull-Rom spline using tool Locus on Q and E. Spline is formed by all positions of Q depending on point E passing the segment BC.
7.		Create new tool CRSpline. Choose Tools from Menu Bar, choose Create new Tool... Output Objects – Locus loc1 :Locus[Q,E]; Input Objects – Point A, Point B, Point C, Point D; Name & Icon – Tool name: CRSpline, Command name: CRSpline.
8.	Input:	Create list of points, insert $s=\{A,B,C,D\}$.
9.		Insert Button, Caption: CRSpline, Apply. Choose Object Properties... of the button, then Scripting, On Click, Java Script. Insert (symbol // introduces comment): <code>ggbApplet.evalCommand("l=Length[s]"); //length of list "s" in GeoGebra a=ggbApplet.getValue("t"); //value "t" is transferred into Java ggbApplet.evalCommand("CRSpline[Element[s,1],Element[s,1],Element[s,2], Element[s,3]]"); //the first part of spline is created and it starts at the first point of list "s" for(i=1;i<a-2;i++){ggbApplet.evalCommand("CRSpline[Element[s,"+i+"], Element[s,"+i"+1],Element[s,"+i"+2],Element[s,"+i"+3]]");}; //cycle creates //single parts of spline for every consecutive quadruple of control points ggbApplet.evalCommand("CRSpline[Element[s,i-2],Element[s,i-1],Element[s,i], Element[s,i]]"); //the last element of spline is created and it ends at the last point of list "s"</code>

Source: own

Finally, if you have a set of discrete points, make a list of these points called s and apply the button CRSpline.

Figure 5: Example of Cattmull-Rom spline



Source: own

Approximation tool called CoonsSpline

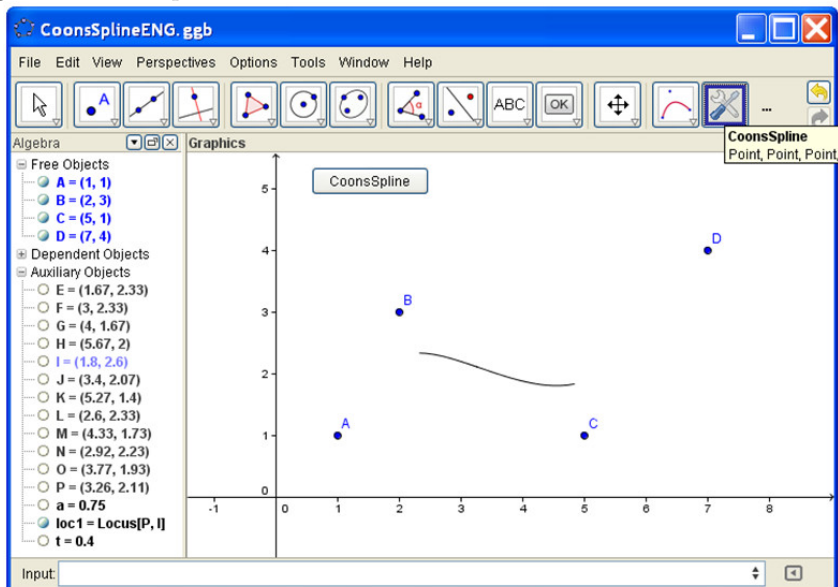
We use Coons cubic for the approximations of points. We continue analogously as we did in the previous case but with one exception. We didn't present a vector equation for Coons spline at this paper; nevertheless we use a different algorithm. At first, we created the tool which operates on four given points. Then we created the button which for an arbitrary list of points constructs Coons spline. We slightly modified the procedure of button for our purpose. The resulting spline begins at the first point of the list and it ends at the last point of the list.

Tab. 2: Step by step construction of Coons spline in GeoGebra

1.		Insert the point $A=(1,1)$ into the input bar. Analogously insert the points: $B=(2,3)$, $C=(5,1)$, $D=(7,4)$.
2.		Insert point $E=A + 2/3 (B - A)$ on the segment AB, then insert $F=B + 1/3 (C - B)$, $G=B + 2/3 (C - B)$ on segment BC and $H=C + 1/3 (D - C)$ on segment CD.
3.		Connect points E and B using tool Segment between two points.
4.		Insert an arbitrary point I on the segment EB using tool New point.
5.		Compute relative distance of I from E with respect to the length $b=B-E$. Insert $t=Length[I-E]/Length[B-E]$. If I belongs to segment EB, then parameter t is set from 0 to 1.
6.		Insert point $J=F + t (G - F)$ on the segment FG, insert point $K=C+t(H-C)$ on the segment CH. Points have the same relative distance t on particular segments.
7.		Insert point L, midpoint of IJ using tool Midpoint or Center, insert point M, midpoint of JK using tool Midpoint or Center.
8.		Insert point $N=L + t (J - L)$ on the segment LJ, then insert $O=J + t (M - J)$ on the segment JM, insert $P=N + t (O - N)$ on the segment NO.
9.		Define the Coons spline using tool Locus on P and I. Spline is formed by all positions of P depending on point I passing the segment EB.
10.		Create new tool CoonsSpline. Choose Tools from Menu Bar, choose Create new Tool... Output Objects – Locus loc1 :Locus[P,I]; Input Objects – Point A, Point B, Point C, Point D; Name & Icon – Tool name: CoonsSpline, Command name: CoonsSpline.
11.		Create list of points, insert $s=\{A,B,C,D\}$.
12.		Insert Button, Caption: CoonsSpline, Apply. Choose Object Properties... of the button, then Scripting, On Click, Java Script. Insert (symbol // introduces comment): <pre> ggbApplet.evalCommand("l=Length[s]");<i>//length of list "s" in GeoGebra</i> a=ggbApplet.getValue("l");<i>//value "l" is transferred into Java</i> ggbApplet.evalCommand("Aux1=(Element[s,1]-(Element[s,2]-Element[s,1]))"); <i>//auxiliary point "Aux1" is created, i.e. spline will begin at first point of list "s"</i> ggbApplet.setVisible("Aux1",false);<i>//auxiliary point "Aux1" is hidden</i> ggbApplet.evalCommand("CoonsSpline[Aux1,Element[s,1],Element[s,2],Element[s,3]]");<i>//the first part of spline is created and it starts at the first point of list "s"</i> for(i=1;i<a-2;i++){ggbApplet.evalCommand("CoonsSpline[Element[s, "+i+"],Element[s, "+i+"+1],Element[s, "+i+"+2],Element[s, "+i+"+3]]");}<i>//cycle creates single parts of spline for every consecutive quadruple of control points</i> ggbApplet.evalCommand("Aux2=(Element[s,l]-(Element[s,l-1]-Element[s,l]))"); <i>//auxiliary point "Aux2" is created, i.e. spline will end at the last point of list "s"</i> ggbApplet.setVisible("Aux2",false);<i>//the auxiliary point "Aux2" is hidden</i> ggbApplet.evalCommand("CoonsSpline[Aux2,Element[s,l],Element[s,l-1],Element[s,l-2]]"); <i>//the last element of spline is created and it ends at the last point of list "s"</i> </pre>

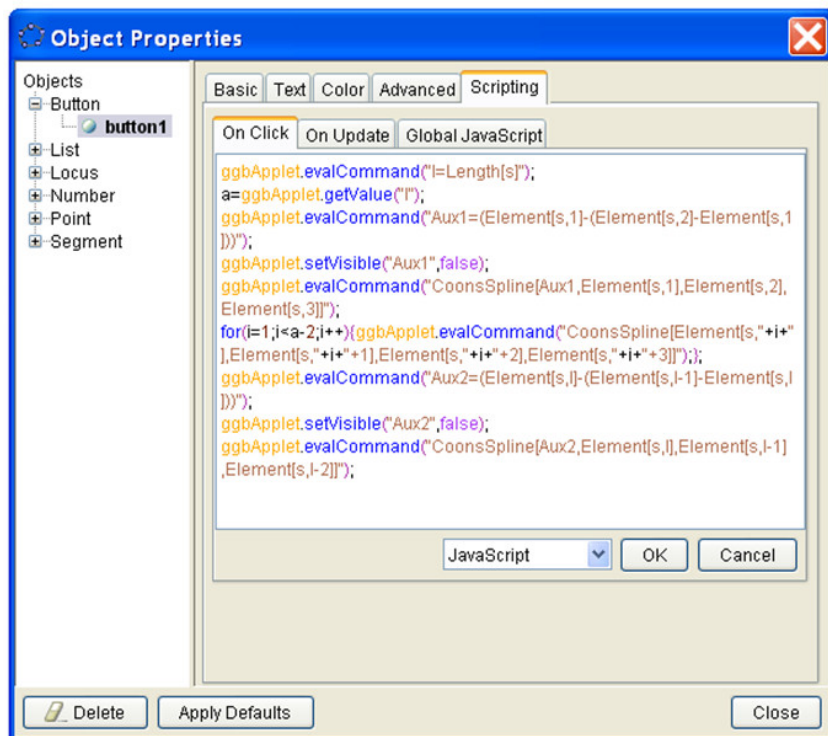
Source: own

Figure 6: CoonsSpline tool



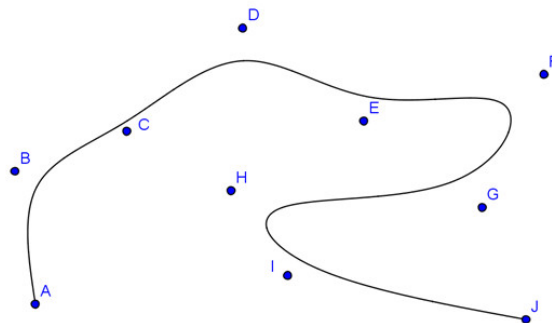
Source: own

Figure 7: JavaScript code for button1



Source: own

Figure 8: Example of Coons spline

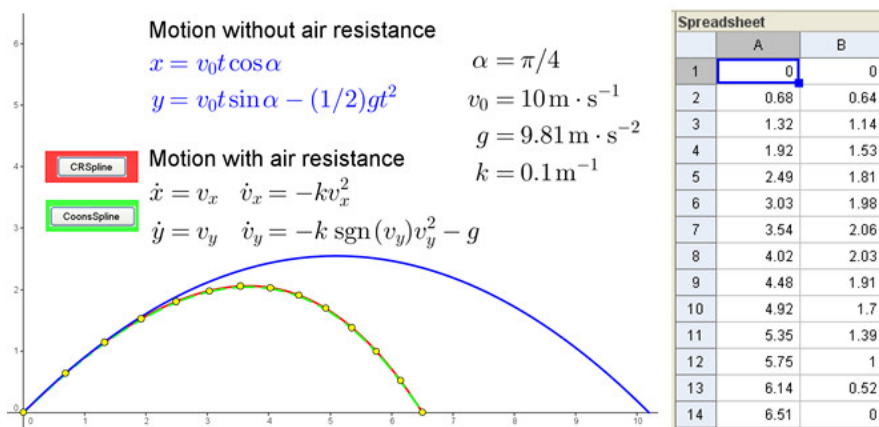


Source: own

Example

We demonstrate our tools on the following example: we will visualize the motion of a particle in the gravitational field with and without air resistance. The first case is very simple and students can become familiar with such a mechanical system during secondary school education. The second case, air resistance takes effect on the moving particle, is quite complicated because appropriate system of differential equations which models the motion can be solved only numerically. For details we refer to (Morávková and Paláček 2011).

Figure 9: Comparison of the motion of a particle in the gravitational field with and without air resistance



Source: own

The differential equations have been solved in an external programme (it can be used any software with numerical solver like MAPLE, MATLAB or MAX-IMA) and the concrete numerical solution has been imported (copy – paste)

into GeoGebra spreadsheet. The list of points has been created using “Create List of Points” tool and spline curves have been constructed. An accuracy of the approximation depends on the density of the control points.

Conclusion

When we have studied mechanical systems described by differential equations solvable only numerically, a question was raised there how to create the trajectories of such systems. We wanted to take advantage of many useful possibilities of the exports provided by GeoGebra, especially with respect to LaTeX and post script graphics format. We had to solve how to construct the appropriate trajectory in GeoGebra.

Acknowledgement

The authors appreciate support of their departments.

Reference

- ALEXANDR, L., 1999. *Výuka počítačové grafiky cestou WWW*. Brno. Diplomová práce. Vysoké učení technické v Brně. Available from: <http://lubovo.misto.cz/curves/index.html>
- FARIN, G., 2001. *Curves and surfaces for CAGD*. Elsevier. ISBN 9781558607378.
- JEŽEK, F., 2006. *Geometrické a počítačové modelování*. Plzeň: Západočeská univerzita. Pomocný učební text.
- MORÁVKOVÁ, Z. and R. PALÁČEK, 2011. Šikmý vrh v odporovém prostředí. In: *Sborník příspěvků 5. konference Užití počítačů ve výuce matematiky*. České Budějovice: JČU. ISBN 978-80-7394-324-0.
- SOCHOR, J., J. ŽÁRA and B. BENEŠ, 1996. *Algoritmy počítačové grafiky*. Praha: ČVUT. ISBN 978-80-01-01406-1.
- GEOGEBRA, 2012a. *GeoGebra* [online]. Last update 2012-10-14 [cit. 2012-11-23]. Available from: www.geogebra.org
- GEOGEBRA, 2012b. *GeoGebraWiki* [online]. [cit. 2012-11-23]. Available from: wiki.geogebra.org

Možnosti vizualizace numerických dat v GeoGebře

Jsou studovány možnosti vizualizace a zpracování numerických dat v GeoGebře s využitím interpolačních a aproximačních křivek. Prezентujeme nástroje pro interpolaci, resp. aproximaci soustavy diskrétních bodů.

Klíčová slova: GeoGebra, interpolace, aproximace, splajny

Kontaktní adresa:

RNDr. Petr Volný, Ph.D., Katedra matematiky a deskriptivní geometrie, Vysoká škola báňská – Technická univerzita Ostrava, 17. listopadu 15, 708 33 Ostrava – Poruba, e-mail: petr.volny@vsb.cz

RNDr. Jana Volná, Ph.D., Ústav matematiky, Fakulta aplikované informatiky, Univerzita Tomáše Bati ve Zlíně, Nad Stráněmi 4511, 760 05 Zlín, e-mail: volna@fai.utb.cz

RNDr. Dana Smetanová, Ph.D., Katedra přírodních věd, Vysoká škola technická a ekonomická v Českých Budějovicích, Okružní 517/10, 370 01 České Budějovice, e-mail: smetanova@mail.vstecb.cz