Spreadsheets in Education (eJSiE)

Volume 3 | Issue 1 Article 4

10-5-2008

Movie-like Animation with Excel's Single Step Iteration Exemplified by Lissajous Figures

Wilfried A.L. Wischniewsky
Hunan Institute of Science and Technology, weihp@web.de

Movie-like Animation with Excel's Single Step Iteration Exemplified by Lissajous Figures

Abstract
This paper presents a new interactive spreadsheet graphics animation technique which uses the spreadsheet engine's intrinsic single step iteration capability. The well-known Lissajous figures exemplify a time variant process demonstrating the virtually unlimited movie-like animation. At any time the visible motion may be interrupted, parameters may be modified, and then the calculation can be continued. First, this paper describes a simple example without VBA and then demonstrates the method's capability by a second more sophisticated Lissajous figures example using VBA. The downloadable workbook uses object blocks and comprises a pedagogically elaborated graphical user interface. It shows that the technique is applicable for both the unskilled student as well as the advanced programmer. This graphics animation method has the potential to revolutionize the spreadsheet usage for dynamical process simulations.

Keywords
graphics animation, iteration, simulation, visualisation

Cover Page Footnote
I wish to express my gratitude to Allen Kemp for his proof-reading and valuable advice. Furthermore, I want to thank my patient wife Lana for her unwavering support.

Movie-like Animation with Excel's Single Step Iteration Exemplified by Lissajous Figures

Wilfried A.L. Wischniewsky
Hunan Institute of Science and Technology
weihp@web.de

Abstract

This paper presents a new interactive spreadsheet graphics animation technique which uses the spreadsheet engine's intrinsic single step iteration capability. The well-known Lissajous figures exemplify a time variant process demonstrating the virtually unlimited movie-like animation. At any time the visible motion may be interrupted, parameters may be modified, and then the calculation can be continued. First, this paper describes a simple example without VBA and then demonstrates the method's capability by a second more sophisticated Lissajous figures example using VBA. The downloadable workbook uses object blocks and comprises a pedagogically elaborated graphical user interface. It shows that the technique is applicable for both the unskilled student as well as the advanced programmer. This graphics animation method has the potential to revolutionize the spreadsheet usage for dynamical process simulations.

Keywords: graphics animation, iteration, simulation, visualisation

1. Introduction

Baker and Sugden [4] report about various authors using spreadsheets for visualizations and simulations. As illustrated in [13] or [5], simulation results are often presented in the form of naked data tables or static diagrams, which unfortunately, do not provide the student with a visual experience of the dynamic process behind the simulation. As a matter of fact, vivid pictures and computer animations contribute significantly to the understanding of dynamical processes.

State of the art visualisations of mathematical applications in Microsoft Excel™ have been compiled by Neuwirth and Arganbright [10], as a treasury of outstanding models. For instance, the eye can follow the trajectory of the moon's orbit, the deformation of Bezier curves, or the convergence or non-convergence of Newton's algorithm. In these examples, spreadsheet formulas read a slider modified parameter and recalculate a data table which is displayed in a suitable diagram. This parameter slider push-pull method is also used by Arganbright, et al [3] to depict parametric equations dynamically.

Two other methods of visualizing time-and-motion [10, p.141ff] concern heat propagation in a metal rod or plate. The thin straight rod is modelled as consisting of I one dimensional segments. The first method calculates a complete **N x I** matrix $(x_{n,i})$, where n, $0 \le n \le N$ represents a time period and $x_{n,i}$ is the **temperature of segment i** at **period n**. A bar chart begins with the display of the initial temperature distribution $(x_{0,1}, \ldots, x_{0,i}, \ldots, x_{0,I})$. By right clicking a macro button, the user triggers a Visual Basic for Applications (VBA) subroutine which increments the period n and

then makes use of the OFFSET function to extract the respective data row **($x_{n,i}$)** out of the matrix. The bar chart automatically updates the new temperature distribution. Successive macro button hits generate a successive series of updated bar charts. This diagram series aids the study of the time-and-motion process even though the program is just reading a once pre-calculated data table.

Since the temperature of each segment at **time period n+1** depends on the temperature distribution of **period n**, the second method illustrated in [10] reduces the data matrix to only two successive rows n and n+1. Spreadsheet formulas calculate the subsequent temperature distribution **($x_{n+1,1}$, … , $x_{n+1,i}$, … , $x_{n+1,I}$)** by reading the previous state n which is also depicted by the bar chart diagram. To generate change, a macro button utilizes the *Paste-Special* command to copy only the current values of **($x_{n+1,1}$, … , $x_{n+1,i}$, … , $x_{n+1,I}$)** onto **($x_{n,1}$, … , $x_{n,i}$, … , $x_{n,I}$)** which in turn automatically recalculates a new n+1 state because the spreadsheet engine is set to *Automatic Calculation* mode. In the 2D grid model of the heat propagation in a thin plate, this second method requires two data matrices of identical size and the resulting heat distribution is presented in a 3D surface grid diagram [10, p.148ff].

Another enhanced idea is forwarded by Arganbright in [2] with some impressive examples. One depicts a function together with its derivatives. Pressing a macro button causes a short slope line representing the local first derivative to move automatically across the function graph within a fixed interval. The background VBA subroutine increases a point counter, thus generating a sequence of 200 successive pictures by one push of a button. Since in *Automatic Calculation* mode the diagram update speed is higher than the human eye can follow, a halting time delay loop interrupts the repeatedly ongoing calculation. The visual result is a vivid animation.

In summary, the prevailing advanced techniques for spreadsheet graphics animation use a) the parameter slider push-pull method or b) the VBA programmed iteration or a combination of both. Both are working on a preset parameter domain and in *Automatic Calculation* mode.

The online MS Excel tutorial on iteration problems of Kardi Teknomo's Web site [17] inspired us while working on a Lissajous figures study tool. The outcome was a movie-like graphical animation similar to JAVA applets and became an illustrative demonstration of the single step iteration method's capability.

2. Spreadsheet Engines use Iteration – a Simple Example

Two fundamental questions of any spreadsheet engine are how and when to calculate the input data. In 1979 the first spreadsheet programmers Dan Bricklin and Bob Frankston [6] solved this question by letting the user choose whether a recalculation occurs automatically after each input (VisiCalc™ command */GRA*) or unsynchronized with data input, that is manually initiated (command */GRM*, recalculation started by pressing the *!* key) [7]. Then, in 1984, revolutionary roll-down menus of the Framework™ spreadsheets [11], [20] put this choice in a *Numbers* menu. In OpenOffice Calc [15], a freely downloadable open-source spreadsheet project, the user makes the same choice in the command menu *Tools – Options – OpenOffice Calc – Calculate*.

This paper is restricted to Microsoft Excel 2003 but all other versions and spreadsheet programs have the same or similar functionality. Excel addresses the iteration question in the command menu *Tools – Options – Calculation* as depicted in Figure 1. New workbooks open with *Calculation* set to *Automatic* and *Iteration* switched off. For movie-like animations we choose *Manual* calculation mode and switch on *Iteration* with *Maximum iterations* set to 1.

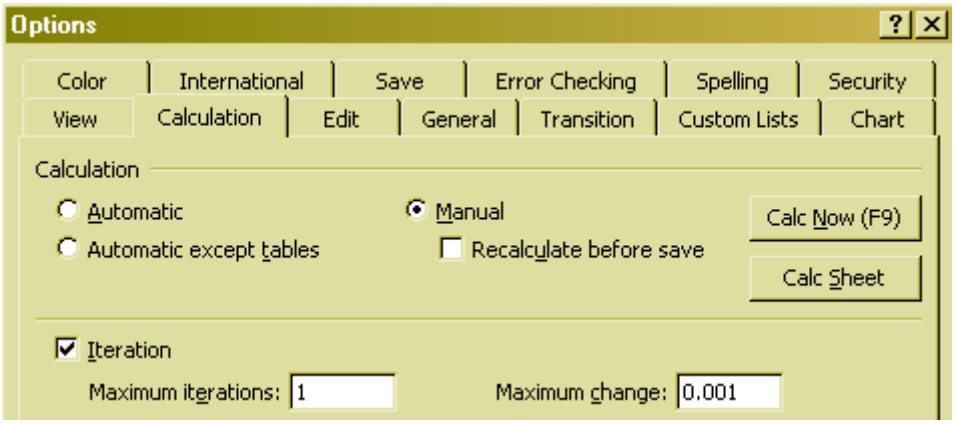

Figure 1: Command Menu - Tools - Options - Calculation Menu

With these settings the spreadsheet engine re-enters the wait-state after each one calculation. This can be tested by a self-addressing formula like A1=A1+1 which returns the value 1 in cell A1 and then waits for the next manual recalculation command. Each F9 key stroke will increase the cell value by one. Against that, *Automatic* calculation mode with switched on *Iteration* and 100 *Maximum iterations* would immediately return 100 because the spreadsheet engine does 100 recalculations in one go. Without *Iterations* switched on, Excel returns the *Circular Reference Toolbar* and opens a help window. In short, the command menu *Tools – Options – Calculation* lets us control when and how the spreadsheet engine recalculates cell formulas.

We can harness this manual iteration method to increment any variable. Figure 2 outlines the mechanism to reset a fictive time (not related to real-time). As long as B1 is not set to *s* for stop, each F9 key stroke increases the **current time t** (cell B4) by the **time step increment Δt** in B3. If the user alters B1 and presses F9 to recalculate again, the **current time t** will be reset to 0. Figure 3 reveals the underlying formulas. Note, that the formula in B4 is self-addressing which in Excel's technical terms is named a *circular reference*. Conditional formatting of cell B1 emphasizes the current mode intuitively in traffic-light manner.

	A	B
1	Type 's'+ENTER to stop and reset the time	go
2	press F9 for continued iteration	
3	time step increment Δt =	0.1
4	current time t =	1.2

Figure 2: Time Reset Mechanism Layout

=IF(B1="s","To start iteration type 'go'+ENTER","Type 's'+ENTER to stop and reset the time")	s
=IF(B1="s","then press the F9 key","press F9 for continued iteration")	
time step increment Δt =	0.1
current time t =	=IF(B1="s",0,B4+B3)

Figure 3: Formulas of the Time Reset Mechanism

Excel reserves 64 bits per numerical value. That allows the iteration steps to range from -10^{307} to 10^{308} [9] which satisfies most applications. The above mechanism does not need VBA and thus can be taught even in middle school classrooms.

3. Lissajous Figures

A Lissajous figure is a two-dimensional parametric curve (x(t), y(t)), $t \geq 0$ with

$$x(t) = a_1 \cos(\omega_1 t + \phi) \quad (1)$$

$$y(t) = a_2 \cos(\omega_2 t) \quad (2)$$

with constant parameters a_1, a_2, ω_1, ω_2 . Usually $a_1 = a_2$ so that the graph of (x(t), y(t)) fits in the square $[-a_1, a_1] \times [-a_2, a_2]$. The combination of different angular speeds ω_1, ω_2 and phase shifts ϕ create a variety of curve traces. Special regular curve traces are famous and named in honour of the French physician Jules Antoine Lissajous. Applications are widespread in physics, electronics, chemistry, and medicine [12]. In 1993 Arganbright [1, p.63-64] has already presented a short tutorial about Lissajous figures using spreadsheets but today's Internet provides several animated Lissajous figure visualisations using JAVA applets [14], [16] and others. Such vivid animations should challenge the spreadsheet programmer. It exceeds the scope of this article to explain the underlying wave theory in detail. Most academic textbooks on physics or electronics, like [8] or [19] and some Internet resources like [21], cover this topic. Differing from the introduction in section 2, the following example uses advanced spreadsheet techniques such as macros, control tools, and naming. Such Microsoft Excel™ functions are well documented in [10], [18] or similar comprehensive textbooks. This paper focuses on describing the design of the interactive **LissajousFigures** workbook to exemplify the single step iteration technique's capability to animate spreadsheet graphics.

The workbook **LissajousFigures** comprises a worksheet named **GUI** being the graphical user interface, a second worksheet **calc** with all calculations, and some VBA program modules. The following sections cover the questions of how to iterate the time (3.1), how to introduce the periodic functions x(t) and y(t) to the student (3.2), how to plot the emitted waves and how these waves combine to become the animated Lissajous figure (3.3), how the overall workbook is designed (3.4), what the VBA subroutines basically do (3.5), and finally what the rationales behind the graphical user interface are (3.6). The section sequence follows the logical structure within the functional blocks of the workbook. Since a student would start playing with the **GUI** we suggest that the reader should first have a look at Figure 16 or play with the workbook before reading the following sections.

3.1. The Time Iteration

The worksheet **calc** contains and manipulates all data that are displayed in the GUI. It starts with the time iteration which basically works as described in section 2. Cell C14 (Figure 4) is named **t** and that of cell C16 **init**. Figure 5 shows the respective cell formulas. The logical variable **init** communicates with VBA subroutines to reset the **phase shift ϕ** between ω_1 and ω_2 each time the user changes either value and also resets the time.

	A	B	C
14	current time t =		127
15			
16		init =	FALSE

Figure 4: Time Iteration Module, Cell Values

	A	B	C
14		current time t =	=IF(init=TRUE,0,C14+1)
15			
16		init =	=init

Figure 5: Time Iteration Module, Cell Formulas

Cell names ease editing subsequent formulas and are especially handy for variables with global significance. Names may be defined by the command menu *Insert - Name – Define* (Figure 6) but it is more convenient to just type the new name into the Name Box left of the *Formula Bar* followed by two *Enter* keystrokes.

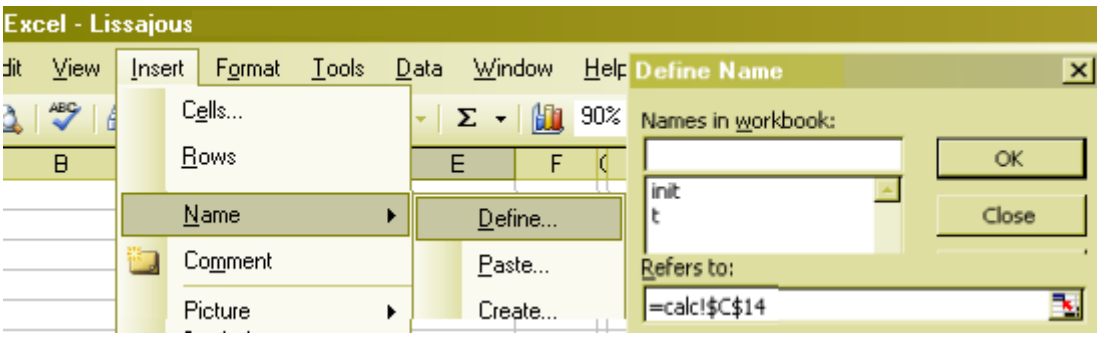

Figure 6: Time Iteration Module, Cell Naming

3.2. Generating the Periodic Movement

Since the underlying concept is based on intuitive understanding, the **GUI** (see Figure 17) presents cycling points on circles as the origin of the waves. For the sake of simplicity we elaborate the programming of equation (2). Equation (1) is done in the same manner. The **horizontal movement** block (Figure 7) uses one slider to set the **angular speed ω** ranging from 0 to 180 degrees (per fictive time unit TU) in cell L15 and a second slider to set the **amplitude a** (in length units LU) which is also the circle's radius. Sliders can be generated by the command menu *View – Toolbars – Forms*. Figure 8 reveals the respective formulas.

	H	I	J	K	L	N	O
11	horizontal movement						
14			radians		degrees		
15		angular speed ω =	0.1047	1/TU =	6		
16		current angle φ(t) =	1.676	=	96.0	slower	faster
18		amplitude a =	4	LU	40		
19						smaller	larger

Figure 7: Wave Generation Block, Cell Values

	radians		degrees
angular speed ω =	=RADIANS(L15)	1/TU =	6
current angle φ(t) =	=IF(init=TRUE,0,MOD(J16+J15,2*PI()))	=	=180/PI()*J16
amplitude a =	=L18/10	LU	40

Figure 8: Wave Generation Block, Cell Formulas

The adjustment of the **angular speed ω** using degrees matches even the young student's knowledge but for speedy animation it has been translated into radians (cell J15). ω remains constant as long as the user does not touch the parameter slider. Note that the **current angle φ(t)** uses a *circular reference* to increase its own value by ω with each iteration step. That is the same construction as the time increment above and causes visual motion in the **Wave Generator** diagram (Figure 9).

The red lined **cycle pointer** in Figure 9 points from the origin to a thick yellow point. The dashed turquoise line, named **U_z(t) pointer**, projects its y-coordinate onto the right border of the XY scatter plot diagram.

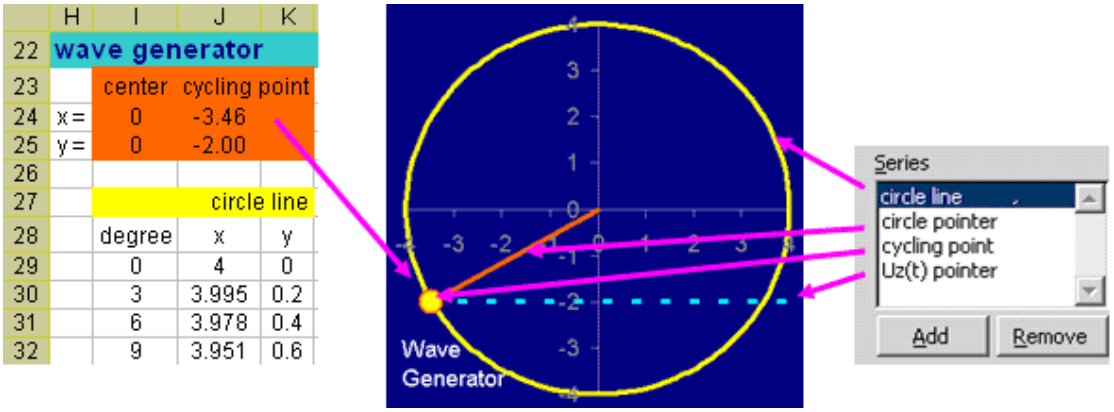

Figure 9: Wave Generator Diagram Design

During animation the yellow **cycling point** cycles around the yellow **circle line** and the **U_z(t) pointer** points to the emitted wave (compare Figure 11). The left side of Figure 9 outlines the data table of the **wave generator** block and the right side lists the diagram's *Source Data - Series – Series* names. To stress their one-to-one relationship the worksheet cells use the same colours as their corresponding graphical pendant. Note that the yellow **circle line**'s data table has been cut off in the figure. The coordinates of the **cycling point** are recalculated by the formulas in Figure 10 with each new **φ(t)**. That causes the displayed point to move with each manually triggered single iteration step. Pressing down and holding the F9 key produces a successive series of images and appears as a movie.

	H	I	J
23		center	cycling point
24	x =	0	=J18*COS(J16)
25	y =	0	=J18*SIN(J16)

Figure 10: Formulas in the Cells J24 to J25

3.3. Animation of the Emitted Wave and the Lissajous Figure

The **emitted wave $U_z(t)$ in** Figure 11 is the visual bridge between the **wave generator** and the **Lissajous figure**. In the common wave equation [8]

$$U_z(t) = a \sin [\varphi_0 + 2\pi f (t - z/c)] \qquad (3)$$

with the amplitude a and the initial phase shift φ_0, we substitute the frequency f by $\omega/2\pi$, and the radiation speed c by 1, which simplifies to

$$U_z(t) = a \sin [\varphi_0 + \omega (t - z)] . \qquad (4)$$

The data table of the **emitted wave** consists of the domain $\{z_k \epsilon \mathbf{IR} : z_k = k/2, 0 \le k \le 120, k \epsilon \mathbf{N}_0\}$ and the respective $U_{zk}(t)$ values which are recalculated with each iteration step ruled by equation (4). The start point $U_{zk=0}(t)$ is read from the y-coordinate of the yellow **cycling point**. Like a chain reaction, the updated **current time** and **cycling point's** x-y-coordinates trigger a whole new wave data table and which in turn causes a wave motion in the graphic.

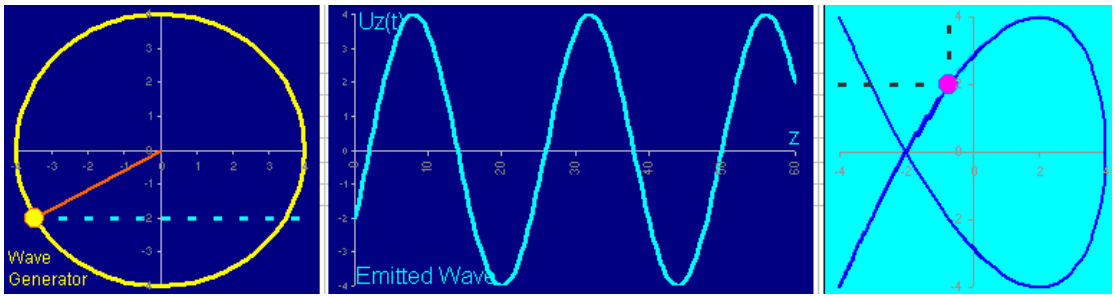

Figure 11: Emitted Wave Plot

The two perpendicular components (x(t), y(t)) of the Lissajous figure are modelled as independent copies of the **wave generator** and **emitted wave** block. Hence equation (4) doubles to

$$U_{z\_vertical}(t) = a_1 \sin [\varphi_{0,1} + \omega_1 (t - z_y)] \qquad (5)$$

$$U_{z\_horizontal}(t) = a_2 \sin [\varphi_{0,2} + \omega_2 (t - z_x)] . \qquad (6)$$

The cell ranges N29 to N149 and O29 to O149 in Figure 13, and the columns W, X, respectively, show their data tables. The **GUI** displays one **emitted wave** horizontally as sketched in Figure 11 and the second one vertically. The iteration mechanism animates both waves simultaneously. Figure 12 shows that their amplitudes at z=60, i.e., $U_{z\_horizontal=60}(t)$ and $U_{z\_vertical=60}(t)$, coincide with the coordinates of the purple **track point** which pulls the Lissajous figure's curve as a blue tail. The curve's data table $\{(z_m, x_{zm}(t), y_{zm}(t)): 0 \le m \le 189, m \epsilon \mathbf{N}_0\}$, is just the downward continuation of the emitted waves' data tables $\{(z_k, U_{zk\_vertical}(t), U_{zk\_horizontal}(t)): 0 \le k \le 120\}$ in the columns N (identical with W), O, and X. The curve's parameter domain $\{z_m : z_m = 60 + m \cdot d, 0 \le m \le 189\}$, varies with the adjustable **tail segment length d**. The data table redundancies have been included for the didactical purpose of visualizing the downward relationship.

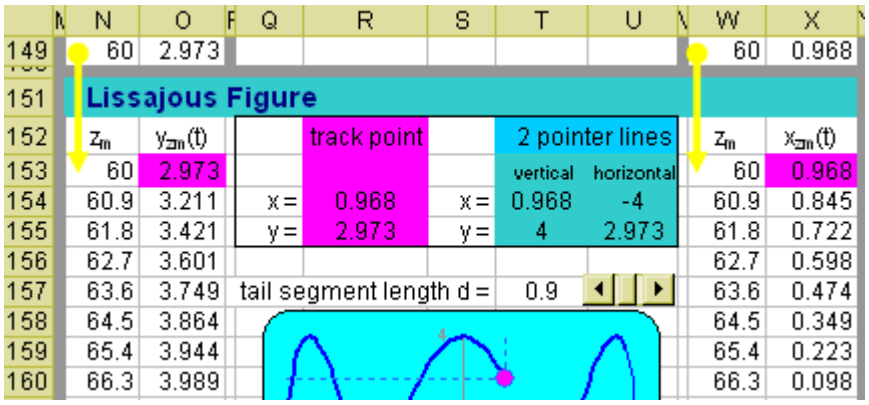

Figure 12: Lissajous Figure with Data Tables

Again the **track point** cell range R152 to R155 is marked in the same purple colour as the visible **track point**, alike the two dashed light blue **pointer lines**.

3.4. Workbook Architecture and Module Co-play

Figure 13 outlines the overall structure of the worksheet **calc** though the long data columns have been truncated. The concept reflects an object oriented design method. Cell range A1 to F10 contains the worksheet header information. The variables **current time t**, **init**, and **phase shift ϕ** are of global use and placed below this header. Headlined with **horizontal movement** one cell area contains all respective objects in one block. It displays all parameters of the **wave generator** and **emitted wave** together with their parameter controls. So does the **vertical movement** block, respectively.

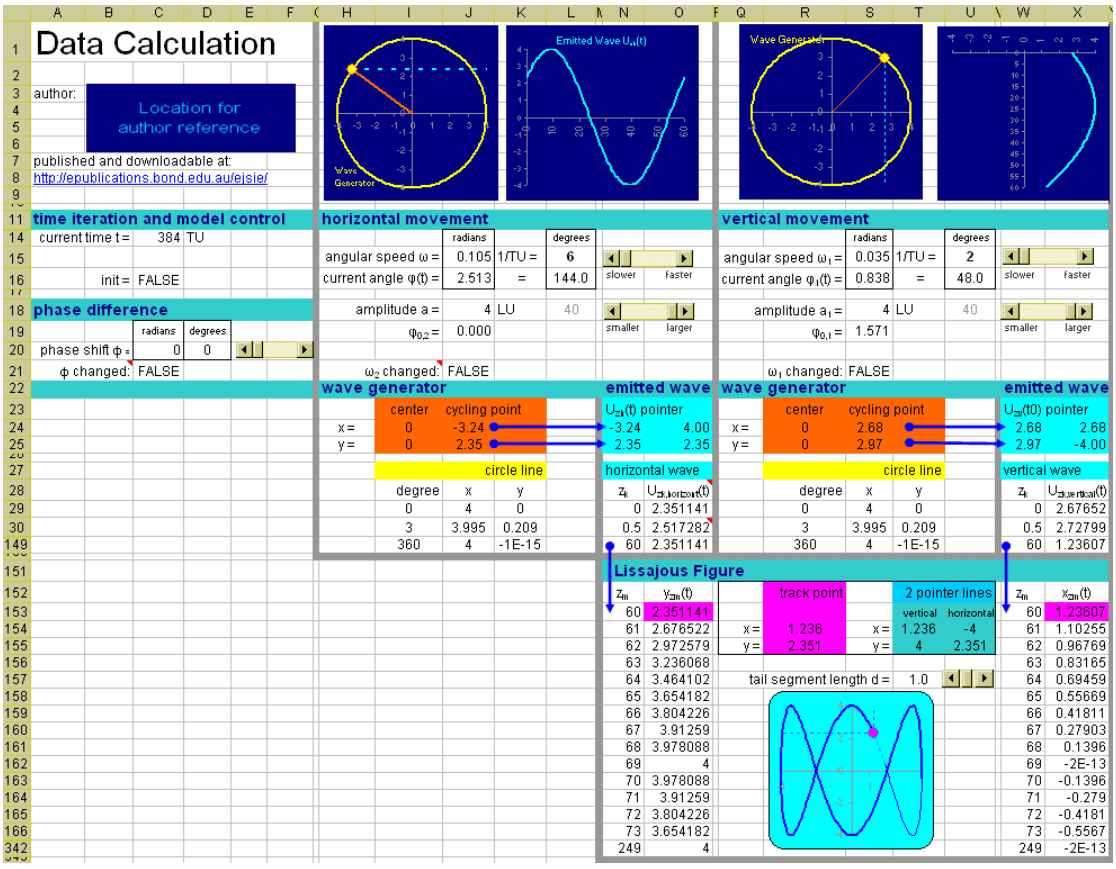

Figure 13: Layout Overview of the Worksheet **calc**

Data tables of the graphical elements follow vertically in the respective columns from line 23 to 149. Cell names and colours are the same as those of the associated dark blue XY scatter plot diagrams located in the range H1 to O10. The Lissajous figure's data columns are positioned exactly below the emitted wave data columns so that the student perceives the mathematical relationship as an obvious continuation.

The right order of the cell dependencies is a key of the proper functioning of the single step iteration method. Concerning the horizontal wave, each F9 keystroke iterates the **current time t** (Figure 3) and the **current angle $\varphi_2(t)$** in cell J16 of the **circle pointer** (Figure 8). The x-y-coordinates of the **cycling point** in J24, J25 read **$\varphi_2(t)$** and the **wave generator** diagram reads the updated values. If the angular changes are sufficiently small, pressing and holding the F9 key creates a visual impression of continuous movement of the graphical elements, i.e., a movie-like animation. The same mechanism applies to the **emitted wave** graphics and the **Lissajous figure** diagram.

Since the spreadsheet engine halts after each single iteration step, the user may interact, for instance to change parameter values or call macros before the next iteration starts. This kind of stop, modify, and go simulation allows the intense study of visualized processes. The animation speed varies with the computer's operating system, program load, and hardware equipment, such as CPU and internal data bus.

3.5. VBA Modules

Though not necessary for movie-like animations, VBA widens the application scope and allows for sophisticated user interactions. Pressing [Alt]F11 opens the editor exhibiting the VBAProject structure (Figure 14).

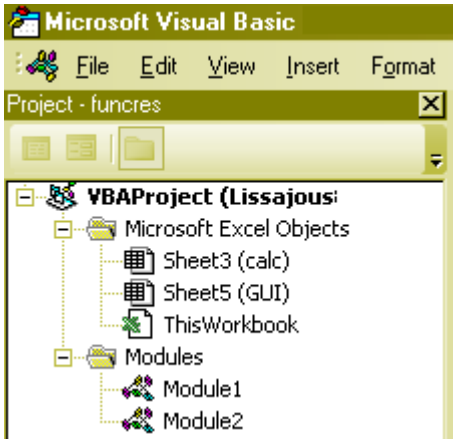

Figure 14: Microsoft Visual Basic Window Fragment

Double clicking on **ThisWorkbook** unhides the VBA intrinsic procedure **Private Sub Workbook_Open()** which presets the parameters r_1, r_2, ω_1, ω_2, ϕ at start-up, i.e., workbook open event. **Sheet5 (GUI)** contains another VBA intrinsic subroutine **Private Sub Worksheet_Calculate()** that checks for user modifications of the trigonometric function arguments in the equations (1) and (2). In this case it re-initializes the time to prevent erroneous phase shift displays. **Module1** contains the

subroutine **ZoomIn** which magnifies the Lissajous figure diagram by simply widening column K and row 23. A respective **ZoomOut** subroutine shrinks them again. A short subroutine **Automatic** (Figure 15) is nothing more than the VBA analogy of manually pressing the F9 key for as many times as the VBA variable **duration** dictates. The intention to include this simple subroutine is to demonstrate how the single step iteration method can be used as well by VBA modules to produce movie-like animations.

```
Sub Automatic()
'    VBA procedure called by the "automatic" button,
'    running an automatic animation sequence
     duration = 300  'number of consecutive iteration steps
     For i = 1 To duration
         Calculate   'performs one single iteration step
     Next i
End Sub
```

Figure 15: VBA Procedure Called by the automatic Button

3.6. The Graphical User Interface GUI

Factoring in that fun and curiosity are the strongest learning motivators; the **GUI** addresses the student in the first place as a player and only secondarily as a researcher. For that very reason the **GUI** is protected by the command menu *Tools – Protection – Protect Sheet* and the mathematical calculations are invisibly put in the separate worksheet **calc**. The author's Chinese students are all familiar with the beloved children's cartoon adventures of the clever detective Kenan (© 株式会社小学馆). They regularly cheer up spontaneously when seeing him giving instructions on the upper left corner of the screen.

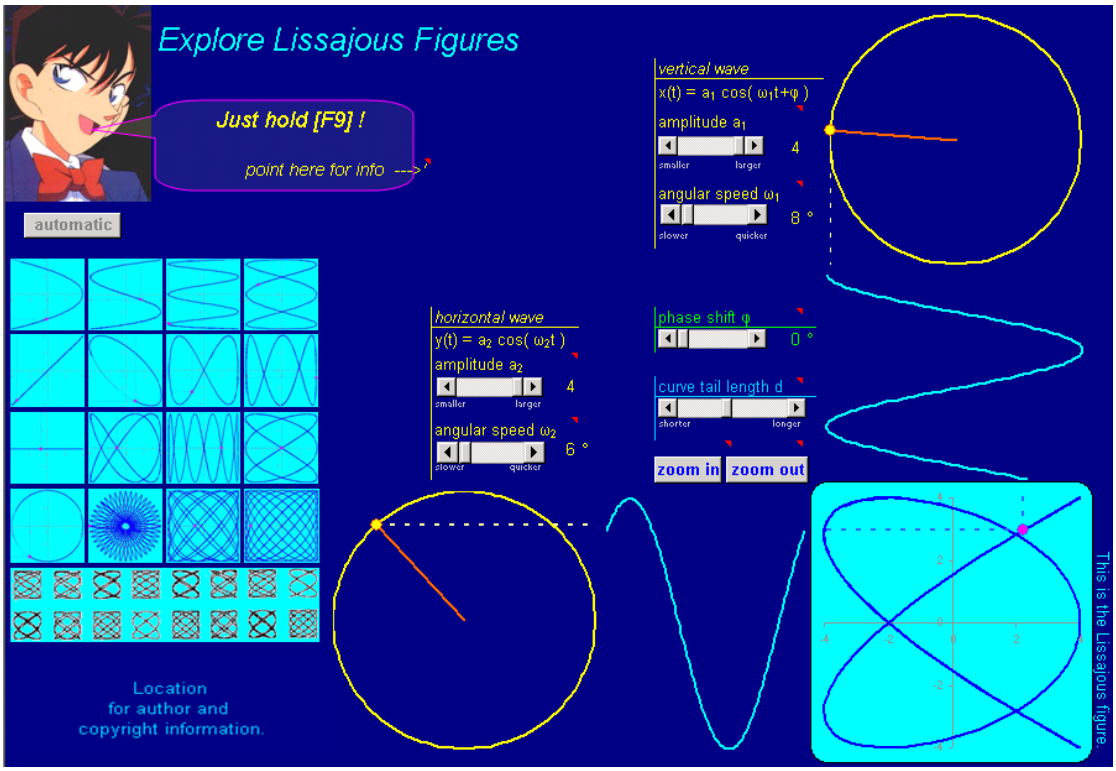

Figure 16: Graphical User Interface GUI

In the accustomed reading direction of western culture, the two independent wave animations flow from top to bottom and left to right. At their perpendicular intersection area they combine as the **Lissajous figure** in the lower right of the display. Since the whole worksheet background is coloured in the same dark blue as the five independent diagrams which are simply copies of those in the worksheet **calc**, the observer's eye catches the composition as a whole rather than as single graphical elements. The associated parameter sliders are positioned close to each diagram so that the student can watch the effect of parameter modifications easily. The left side of the screen exhibits a set of small example diagram icons. Not all are Lissajous figures. Some are products of numerical errors caused by too large angular speeds, i.e. step sizes. For pure pedagogical reasons they are included since the author discovered that such diagrams encourage the student's playfulness and thus support the educator's target. On mouse-over, a screen tip reveals each icon's parameter setting but there are no macros engaged because the student should be motivated to explore the diagram variety using a manual trial and error method. Such handy screen tips can be linked to pictures by the right mouse button context menu *Edit Hyperlink – Screen Tip*. Figure 17 depicts one of the **GUI**'s supplementary help texts when the cursor moves over one of the red triangles which indicate a cell comment. The right click context menu *Insert Comment* offers the necessary programming tool.

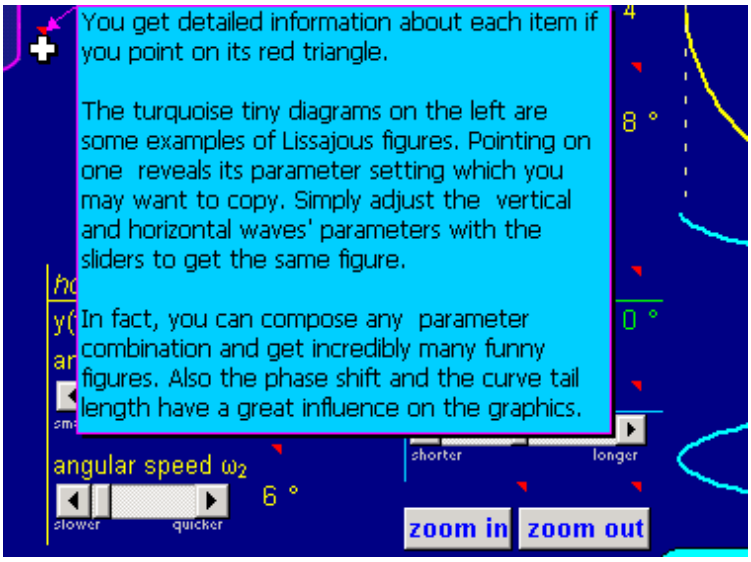

Figure 17: Clipping Depicting a Help Text Example

A **zoom in** button magnifies the **Lissajous figure** diagram to expose details. It is complemented by a **zoom out** button which leads back to the overall view. The tricky idea behind these macros is revealed in section 3.5. Though the **GUI**'s purpose is to activate the student's manual exploration, an unobtrusive **automatic** macro button has been added to facilitate automatic uninterruptible demonstrations. Additionally, a graphical user interface should exhibit a meaningful and motivating title and discreet information about the author's reference.

For solving application problems the choice of the most suitable computation tool determines the success. In single step iteration mode spreadsheets provide virtually endless calculation series but may be interrupted at any step to analyse the

system state; modify parameter values; call VBA macros to launch complex model changes; and continuation from the interrupted system state. This opens a wide range of new spreadsheet applications in dynamic simulations.

4. Summary

This article first discusses the prevailing methods to visualize time-and-motion with Excel graphics. Then it addresses the spreadsheet engine's user adjustable calculation modes, i.e., automatic or manual. The manual calculation mode can be set up for single step iteration which is being used to animate Lissajous figures. The article explains in detail the object oriented design and functioning of the worksheet, focusing on how iteration produces the visual movement in diagrams. Then it elucidates the pedagogical rationales of the graphical user interface which provides the parameter controls and displays a composition of five synchronized animated diagrams.

The elaborated example demonstrates how this technique uses the very concept of the spreadsheet engine's intrinsic single step iteration capability to generate continuous and virtually unlimited animations of spreadsheet graphics. The user can produce continuous or single-step slow motion controlled by just one keyboard key. At any iteration step, the calculations may be interrupted, parameters changed, VBA macros activated, and then continued which makes this technique even interactive.

Animation with single step iteration has the potential to revolutionize the usage of spreadsheets to simulate, analyse, and study dynamical processes. It opens a huge field of movie-like interactive graphical animations for time or space variant dynamical process simulations.

References

[1] Arganbright, D. (1993). *Practical Handbook of Spreadsheet Curves and Geometric Constructions*, CRC Press, Boca Raton, FL, USA

[2] Arganbright, D. (2005). Enhancing Mathematical Graphical Displays in Excel through Animation, *Electronic Journal of Spreadsheets in Education* 2(1). Available online at http://epublications.bond.edu.au/ejsie/vol2/iss1/8/

[3] Arganbright, D., Neuwirth, E., and Smith, R.S.(2005). Workshop in Excel. *Proceedings of KAIST International Symposium on Enhancing University Mathematics Teaching :*183-194, Daejeon, Korea

[4] Baker, J.E. and Sugden, S.J. (2003). Spreadsheets in Education – The First 25 Years, *Electronic Journal of Spreadsheets in Education* 1(1):18-43, Available online at http://epublications.bond.edu.au/ejsie/vol1/iss1/2

[5] Bourg, D.M. (2006). *Excel Scientific and Engineering Cookbook*. OReilly Media, Inc., Sebastopol, CA, USA

[6] Bricklin, D. VisiCalc: Information from its creators, Dan Bricklin and Bob Frankston. *Web Resource*. http://bricklin.com/visicalc.htm

[7] Bricklin, D., Frankston, B. VisiCalc Reference Card. *Web Resource*. http://www.bricklin.com/history/refcard3.htm

[8] Höfling, O. (1985). *Physik*. F.Dümmler Verlag, Bonn, Germany

[9] Microsoft Corporation. Excel 2003 Specifications and Limits. *Web Resource.* http://office.microsoft.com/en-us/excel/HP051992911033.aspx?pid=CH062527721033

[10] Neuwirth, E. and Arganbright D. (2004). *The Active Modeller: Mathematical Modeling with Microsoft Excel.* Thompson/Brooks-Cole Publishers, Belmont, CA, USA

[11] Pressler, R. (1985). *Framework, Developer's Handbook*. Ashton-Tate, Culver City, California, USA

[12] Saeki, Y., Sagawa, K., Suga, H. (1978). Dynamic Stiffness of Cat Heart Muscle in Ba2+-Induced Contracture. *Circ. Res*. 1978; 42; 324-333. Available online at http://circres.ahajournals.org

[13] Seal, K.C. and Przasnyski, Z.H. (2005). Illustrating Probability through Roulette: A Spreadsheet Simulation Model. *Electronic Journal of Spreadsheets in Education* 2(1): 73-94. Available online at http://epublications.bond.edu.au/cgi/viewcontent.cgi?article=1028&context=ejsie

[14] St Andrews, University, Scotland. (1996). Lissajous Figures Applet. *Web Resource.* http://www-history.mcs.st-andrews.ac.uk/Java/Lissajous.html

[15] Sun Microsystems Inc.. OpenOffice. *Web Resource*. http://www.sun.com/software/star/openoffice/index.xml

[16] Surendranath R.B. Lissajous Figures Applet. *Web Resource.* http://pagesperso-orange.fr/olivier.granier/meca/simul/lisajou/simul.html

[17] Teknomo, K. (2006). Excel On-Line Tutorial. *Web Resource*. http://people.revoledu.com/kardi/tutorial/Excel/Iteration.html

[18] Walkenbach, J.(2003*). Excel 2003 Bible*. Wiley Publishing, Inc., Indianapolis, Indiana, USA

[19] Westphal, W.H. (1952). *Physikalisches Wörterbuch*. Springer-Verlag, Berlin, Germany

[20] Wikimedia Foundation Inc.. Framework Office Suite. *Web Resource.* http://en.wikipedia.org/wiki/Framework_%28office_suite%29

[21] Wolfram Research Inc.. Lissajous Curve. *Web Resource.* http://mathworld.wolfram.com/LissajousCurve.html