Projects for NCTS Workshop on DifferentialEquations, Surface Theory, and Mathematical Visualization

February–March, 2003

Differential Geometry Computer Projects for NCTS Workshop
Part I. C. L. Terng

(1) Plane curve project

(i) Write a computer program to compute curvature of any parametrized user defined immersed plane curve, then draw osculating circles, parallel curves, focal points, involutes, and evolutes.

(ii) Use ODE solver to program the Fundamental Theorem of plane curves from a user given curvature function, then draw osculating circles, parallel curves, focal points, involutes, and evolutes.

(2) Space curve project

(i) Write a program to compute curvature and torsion of any immersed space curve.

(ii) Use ODE solver to program the Fundamental Theorem of curves in $\mathbf{R}^3$ from any user defined curvature and torsion functions, and two principal curvatures, then draw Frenet frame and parallel frame.

(3) Non-linear Schrödinger equation (NLS) and smoke ring equation project

The smoke ring equation is the curve evolution $\gamma_t = \gamma_x \times \gamma_{xx}$. It is arc length preserving flow. Geometrically, it means the space curve is moving in the direction of its binormal with the curvature as its speed. The two principal curvatures evolves according to NLS, which is a soliton equation. Hence we can use soliton theory to study this curve flow.

(i) Program the Sattinger-Lee pseudo-spectral method to solve Cauchy problem with user defined periodic initial data of NLS.

(ii) Use the inverse of Hashimoto transform to compute the cuver evolution numerically and then draw the smoke ring curve evolution.

(4) Write a program to draw all surfaces of revolutions in $\mathbf{R}^3$ that are Weingarten surfaces. Here a surface in $\mathbf{R}^3$ is called Weingarten if the mean curvature H and the Gaussian curvature K satisfy a linear relation, $aH+bK = c$ for some constants a, b, c. In particular, the case when $H =$ constant and K is constant.

(5) Program the Fundamental Theorem of surfaces in $\mathbf{R}^3$. This means:

(i) Compute the first, second fundamental forms, the principal curvatures, mean curvature, and the Gaussian curvature of a user defined parametrized immersed surface in $\mathbf{R}^3$.

(ii) Determine whether the user defined first and second fundamental forms satisfy the Gauss-Codazzi equations.

(iii) Integrate the Gauss-Codazzi equation numerically (obtained from (ii)) to get the surface and draw it.

(6) Sine-Gordon equation (SGE) and $K = -1$ surfaces

The Gauss-Codazzi equation of surfaces in $\mathbf{R}^3$ with $K = -1$ is the SGE, which is a soliton equation. Hence we can use soliton theory to construct surfaces with $K = -1$ in $\mathbf{R}^3$.

(i) Draw the $K = -1$ surfaces corresponding to n-soliton solution of SGE.

(ii) Solve the Bäcklund transformation of surface of revolution with $K = -1$ (but not the pseudosphere) repeatedly to get new surfaces of $K = -1$ and draw these surfaces.

(7) Isothermic surfaces in $\mathbf{R}^3$ and soliton equation

The Gauss-Codazzi equation for isothermic surfaces in $\mathbf{R}^3$ is a first order PDE system of three functions, which is the reduced 3-wave equation associated to $O(4,1)$. It is again a soliton equation. Hence we can use soliton theory to study these surfaces.

(i) Construct all surfaces of revolutions that are isothermic.

(ii) Construct isothermic surfaces corresponding to n-solitons (n=1,2,3)

(iii) Construct Ribaucour transformations from isothermic surfaces of revolution, and draw the new isothermic surfaces.

Differential Equations Computer Projects for NCTS Workshop
Part I. R.S. Palais

In all of the projects below you may use your favorite programming system (i.e., Maple, Mathematica, Matlab, or a compiled language like Java or C).

(1) Numerical solution of the Initial Value Problem.

Write a program to solve the initial value problem for a first order system of ODE in two variables. The user should be able to enter the ODE (as in the User Defined examples in 3D-XplorMath) and also choose initial conditions an integration method, and step-size. Your program should at least implement the Euler and Runge-Kutta methods (and perhaps an adaptive Runge-Kutta).

Of course, if your programming system has built in ODE solvers you should **not** use them but rather program the methods directly from the algorithms.

(2) Visualization of solutions of the Initial Value Problem.

Write a program to display the solution of the initial value problem for a first order system of ODE in two variables. The user should again be able to enter the ODE and also choose initial conditions. Use the Runge-Kutta method to integrate the equation, and in this case you may use a built-in solver if one is available in your programming system, but of course do not use a built-in call to a method for ODE display.

The user should have the option of showing the vector field along with the solution. Initially you should do this for time-independent vector fields, and when you get that working see if you can solve the (much harder) problem of displaying time-dependent vector fields.

(3) Visualizing multiple nearby solutions of the same IVP.

Write a program to display several neighboring solutions of the same initial value problem for first order systems of ODE in two and three variables. The user should be able to enter:

1) the ODE,
2) a central initial condition p,
3) a size S for a cubical box surrounding the central initial condition,
4) A small integer N giving the number of simultaneous solutions.
5) A step-size δ.

The program should first choose at random N initial conditions in the box of edge-length S centered at p. It should then solve the N IVPs step by step and at each step display the current solutions as different colored dots (after erasing the previous dots).

(The effect will be a small cloud of colored dots. What will be of interest, particularly for the case of "chaotic" ODE is to see how quickly the initially nearby dots spread out in time—showing visually how sensitively the solutions of the IVP depend on the initial condition.)