SCIENTIFIC COMPUTATION 101

Adam O. Hausknecht
University of Massachusetts Dartmouth
Mathematics Department, 285 Old Westport Road, N. Dartmouth, MA 02747-2300
ahausknecht@umassd.edu

Over the past seven years, with the hiring of new faculty, the research interests of the members of the mathematics department at UMass Dartmouth have shifted toward one that emphasizes numerical mathematics and computational mathematics. The shift toward a greater use of computational mathematics has also occurred in the physics department and other departments and programs throughout the university. This change has caused the mathematics department faculty to improve its existing undergraduate computational mathematics program by

- Applying for and receiving an NSF *CSUMS: Research in Scientific Computing in Undergraduate Education (RESCUE)* in September of 2008.
- The development of new undergraduate courses in computational mathematics, and
- Increasing the integration of mathematics software into the curriculum

The CSUMS Grant

The CSUMS grant was awarded to a team led by Prof. Sigal Gottlieb and included Prof. Steve Leon, Prof. Gary Davis, and Prof. Jae-Hun Jung (now at SUNY Buffalo). This grant has enabled the department to support the involvement of six or seven students (per semester) in undergraduate research projects. The support includes supplying the students with MacBooks, a $6000 stipend for two semesters and a summer, and the cost of travel to regional and national conferences (see http://compmath.wordpress.com/). Here are several of our recent students that attended the *Annual 2009 SIAM Conference* in Denver:

- Silvia, Leanne: "Laplacian Graph energy", Poster Presentation, Advisor: Prof. Davis
- Kimball, Ann: Without Presentation
- Ross, Aimee: "Pattern Formation Growing Sandpiles", Poster Presentation, Advisor: Prof. Davis
- Bresten, Christopher:, "Recovery of high order accuracy via Gegenbauer reconstruction in radial basis function approximation for discontinuous problems", Oral Presentation, Advisor: Prof. Gottlieb and Prof. Jung
- Callahan, Adam: "Agent Based Model of the Spread of Disease Incorporating Fear", Poster Presentation, Advisor: Dana Fine
- Perry, Andrew: "Practical Sieve Implementation in Parallel Python", Poster Presentation, and " A Deniable Chaotic Cryptosystem", Poster Presentation (joint with Chris Bresten), Advisor: Saeja Kim

Also, in **Figure 1** below, are a group of students and faculty that participated in the 15th Annual Massachusetts Undergraduate State-wide Research Conference held at UMass Amherst on May 1, 2009. The students gave poster or oral presentations of of their work.

Figure 1: The CSUMS students and faculty at the *Annual* Massachusetts Undergraduate State-wide Research Conference.

Currently students in the CSUMS program participate in a seminar running throughout the year including summer led by Professors G. Davis, S. Gottlieb, A. Heryudono, S. Kim, S. Leon, and C. Wang. The seminar is held in a technology-equipped classroom. During the seminar, each student is expected to present updates on the progress they have made on their projects and respond to questions about their projects. They are also expected to maintain their own websites where they report their progress on their research, and both students and faculty can post comments about their work. In this way, they gain practice at presenting mathematics in a friendly environment. They also learn how to write abstracts of their research and learn to which conferences they should be submitted for serious consideration. Other faculty, including myself, participate in the seminar when appropriate and time permits. Also, the seminar leaders arrange for mathematicians and scientists from other institutions and industry to give talks about their research designed for an audience of undergraduates.

New Courses in Computational Mathematics

To help prepare students for upper-level applied and numerical mathematics courses and undergraduate research projects, Prof. Leon developed a 200-level MATLAB-Octave based course on experimental mathematics and I developed a Python-based 200-level introductory course on scientific computation. These courses are project-based and assume students are taking differential equations or calculus III. These courses replace

two of the three existing programming course requirements taught by the computer science department. We did this because the computer science department's courses do not present examples or assign programming projects relevant to mathematics and science majors. After experimenting with Maple, Java, Octave and Python, I settled on Python as the core language for the course because it is free, has relatively simple syntax, is interpreted, runs on all platforms, and is widely used by the scientific computation community. Moreover, many open-source math and science modules (extensions) are available including Numpy, Visual Python, Matplotlib and Parallel Python that add many of the tools that are available in MATLAB-Octave to Python. Also, Python is the language of the open-source computer algebra system Sage. Below are two example Python programs from my scientific programming course this spring.

Example 1: A program for testing the LeftEnd Point Rule.

```
def g(x):
  return sin(x)

def leftEndPt(g, a, b, n):
  deltaX = (b-a)/n; s  = 0.0; i = 0

while i < n:
  x = a+i*deltaX
  s += g(x) # For speed, use '+=' instead of sum = sum + sin(x)
  i += 1    # For speed, use '+=' instead of i = i + 1
  return s*deltaX

for n in range(500, 10500, 500):
  result = leftEndPt(g, 0.0, pi, n)
  print 'The approximation of the Integral(sin(x),x=0..pi) using ',
  print '%d rectangles is %g' %(n, result)
  print 'The error in the approximation is %g'%(abs(result-2.0))
  print ''
```

Example 2: A student's program that uses Visual Python to demo Monte Carlo Integration.

```
## Charles Poole Friday February 26, 2010
## Method for estimating the area of an integral

## Made for MTH280, Umass Dartmouth - Edited by A.O. Hausknecht
from visual.graph import * # import graphing features
import random
from math import pi
```

```
def MonteMethod(g,n):
    funct1 = gcurve(color=color.green) # Connected curve objects
    funct2 = gcurve(color=color.blue)
    funct3 = gcurve(color=color.blue)
    for x in arange(0, 1.001, .001): # x goes from 0 to 1
        funct1.plot(pos=(x,g(x)))    # plot
        funct2.plot(pos=(x,4))
        funct3.plot(pos=(1,4*x))
    s = 0.0; Area = 4
    funct4 = gdots(color=color.white)
    for i in arange(1,n,1):
        c = random.random(); d = random.uniform(0,4)
        # Comment out to animate dots better
        print 'Monte Carlo Approximation to the area of this
integral is %g after %g runs' %(Area*(s/i),i)
        if g(c) < d:
            funct4.plot(pos=(c,d), color = color.red)
        else:
            s += 1.0; funct4.plot(pos=(c,d), color=color.white)

    print 'Area tested is %g. Total hits beneath function is %g,
after %g runs' %(Area,s,n)
    print 'The ratio of hits to misses is %g' %(s/n)
    funct1 = gcurve(color=color.green) # a connected curve object
    funct2 = gcurve(color=color.red)
    funct3 = gcurve(color=color.red)
    for x in arange(0., 1.001, 0.001): # x goes from 0 to 1
        funct1.plot(pos=(x,g(x)))      # plot
        funct2.plot(pos=(x,4))
        funct3.plot(pos=(1,4*x))

    return float(Area*s/n)

def g(x): #Function are using
    return (4.0*((1.0-x**2)**.5))

approx = MonteMethod(g,10000)

print 'Monte Carlo Approximation to the area of this integral is
%g' %(approx)
print 'Exact Value is %g and error is %g' %(pi, abs(pi-approx))
```

The student's program produced **Figure 2** below and the following output.

```
Area tested is 4. Total hits beneath function is 7871, after 10000
runs
The ration of hits to misses is 0.7871
Monte Carlo Approximation to the area of this integral is 3.1484
Exact Value is 3.14159 and error is 0.00680735
```

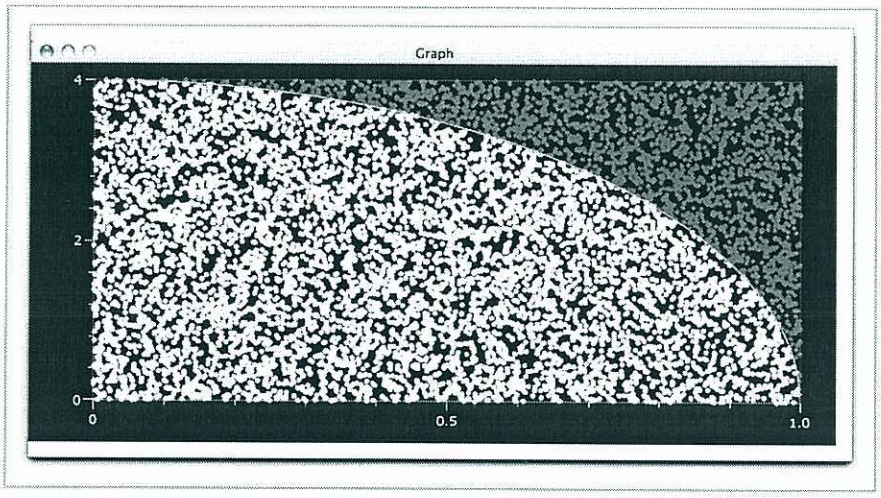


Figure 2: Using the Monte Carlo Method to approximate π

For more more information about Python and scientific programming see [2] below.

Increased Integration of Mathematics Software into the Curriculum

Many of the faculty have stepped up the use of computer algebra systems such as Maple, Mathematica, and Sage in their courses and in undergraduate research projects. For example, Prof. Fine recently offered a course on the web, social networking and random Graphs. For this, he used Maple for in-class projects throughout the course. This semester Prof. Fine plans to use Maple for projects in his course on quantum mechanics. Prof. Davis uses Mathematica for many of the undergraduate research projects he sponsors and uses the open-source statistics package R (www.r-project.org) for his upper-level calculus-based statistics course. Because students have limited access to Maple, Mathematica and MATLAB, Prof. Heryudono, myself and others have begun using Maxima (maxima.sourceforge.net), Sage (www.sagemath.org), Octave (www.gnu.org/software/octave/) and TEMATH (www2.umassd.edu/temath) in their courses. Here are several examples.

Sage Version 4.3.3 Release Date: 2010-02-21

```
sage: var('x, y') ## Declare x and y to be variables
sage: plot(sin(1/x),(x, -1, 1)) ## Example 1, see Figure 3
sage: f(x,y)= x*sin(y)+y*sin(x) ## Example 2, see Figure 4
```

```
sage: contour_plot(x*sin(y)+y*sin(x),(x,-6.28,6.28),
(y,-6.28,6.28),plot_points=400,fill=False, linestyles=['solid',
'dashdot'],labels=True)
```

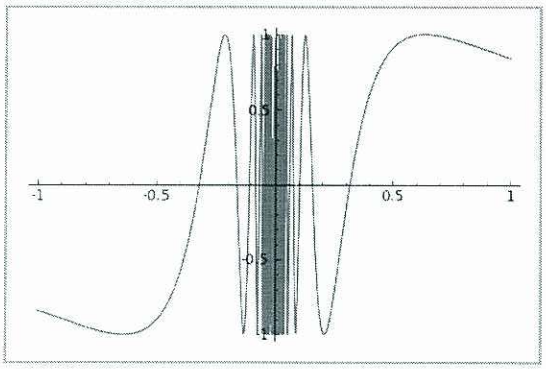


Figure 3: A Sage plot of $y = \sin(1/x)$

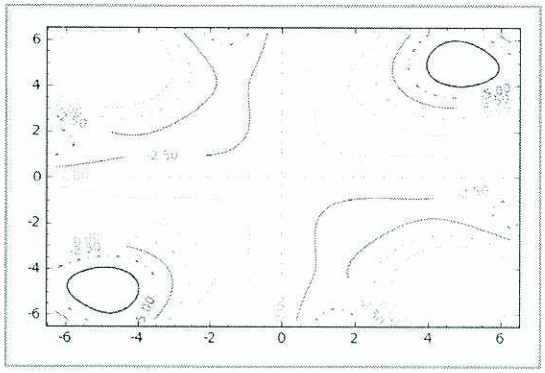


Figure 4: A Sage contour plot

Octave Version 3.2.3

The next two examples are taken from [1] with modifications.

```
> %% Example 1: A plot of the solution of dy/dt=exp(-t^2), y(0)=1
> x = zeros(31); y = zeros(31); f = inline('exp(-t.^2)','t');
> for i=0:30 t(i+1) = i/10; y(i+1)= 1+quad(f,0,t(i+1)); end;
> plot(t,y); text(1, 1.5, "y(t) = 1 + integral(exp(-u^2), 0, t)");
>
> %% Example 2: A plot of the solution of x' = t-x^2
> xdot = inline('t-x^2','x','t');
> odes(xdot,[0 5], 0, 0.1); %% My replacement for MATLAB's ode45.
```

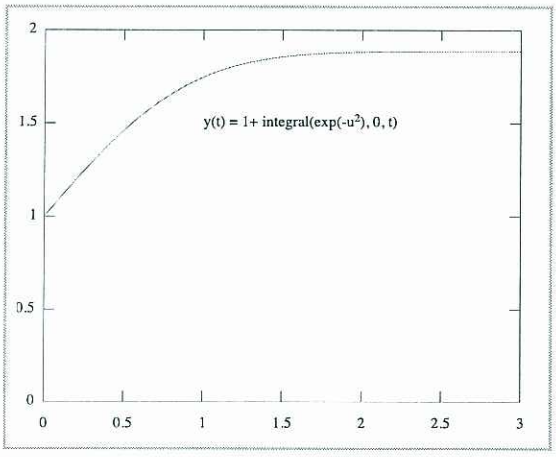


Figure 5: A solution using Octave's *quad*

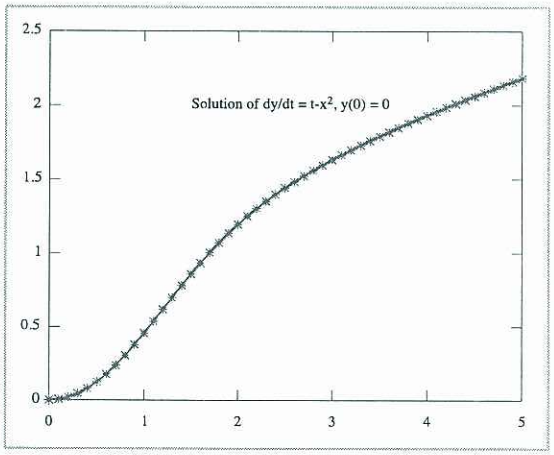


Figure 6: A solution using *odes*

The code for *odes* used above is shown below and is posted at

http://www.umassd.edu/cas/mathematics/people/hausknecht/

```
function [x,y] = odes(ydot,interval, y0, h)
% An Octave replacement for MATLAB's ode45
 x0  = interval(1); % The minimum x-value
 x1  = interval(2); % The maximum x-value
 x = x0:h:x1;
 % Generate an approximate solution for the differential
 % equation with the initial condition y(x0)= y0.
 y = lsode(ydot,[y0],x);
 hold on; % Enable plot overlay so plots will be combined
 % Generate a vector of x-values using a set size h = 0.1
 % Plot the solution as a curve
 size=2; % linewidth and marker size
 plot(x,y,'linewidth', size);
 % Overlay a the solution points
 plot(x,y,'*', 'markersize', size+1);
 % Overlay a the solutions initial point in red
 plot(x0, y0,'+r','markersize',size+2);
 hold off; % Disable plot overlay
 % Use the transpose operator to the row vector of
 % x-values into a column vector (to match MATLAB's ode45).
 x = x';
 end
```

TEMATH (Developed by A.O. Hausknect and Robert E. Kowalczyk)

Example A: In calculus courses, TEMATH's Limit tool can be used to introduce the concept of a limit and the $\varepsilon - \delta$ definition of the continuity of a function. In **Figure 6** , the function $f(x) = x^2 - 4$ is plotted along with a visual representation of the the largest delta neighborhood (the darker shaded rectangles) of $a = 3$ for a given ε (the large light shaded rectangle).

Example B: In a calculus III course, TEMATH's Parametric Tangent tool can be used to dynamically plot the unit velocity vector, the unit acceleration vector and the osculating circle of an object whose position is given by a parametric curve in the plane (see **Figure 7** below). Note that we have recently added a feature that makes it possible to represent the moving object using predefined icons such as a the green car shown in **Figure 7**.

Example C: In a differential equation course, TEMATH can be used to visualize the qualitative properties of a first-order autonomous differential equation, as well as its equilibrium solutions, its phase diagram, and its direction field (see **Figure 8** below).

Note that the solution is dynamically regenerated as the mouse is dragged, which helps to show students that the qualitative nature of a solution depends on the location of its initial-condition relative to the equation's equilibrium solutions.

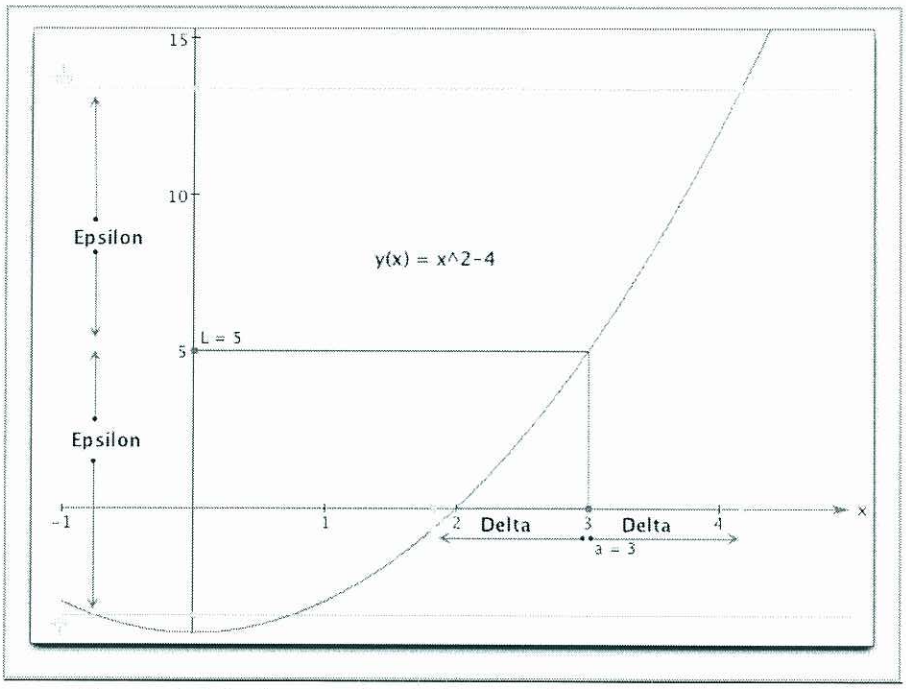


Figure 6: The largest δ -neighborhood for a given ε at $a = 3$

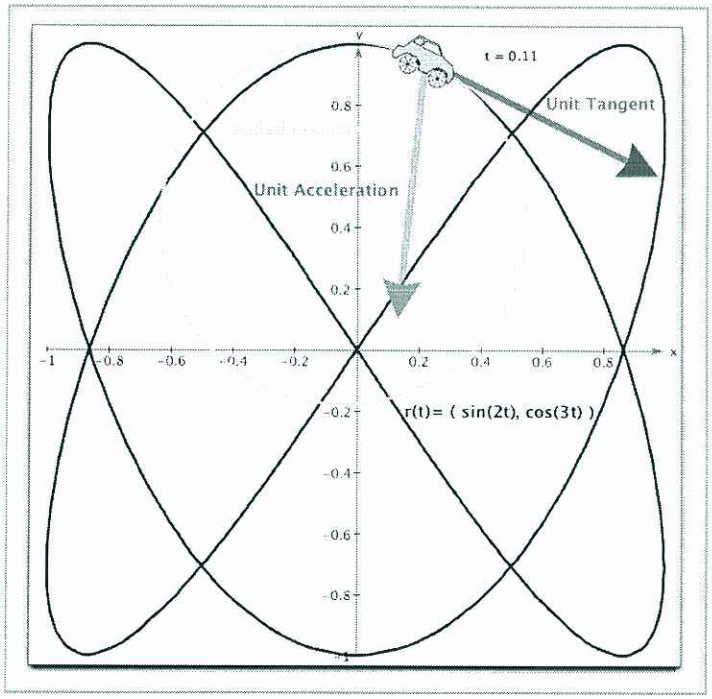


Figure 7: An object (the green car) whose position is given by $r(t) = \left(\sin(2t), \cos(3t)\right)$

Example D: TEMATH's First-order Differential Equation tool can also be used to demonstrate both the Euler and the Runge-Kutta methods for numerically solving initial-value problems (see **Figure 9** and and **Figure 10** below).

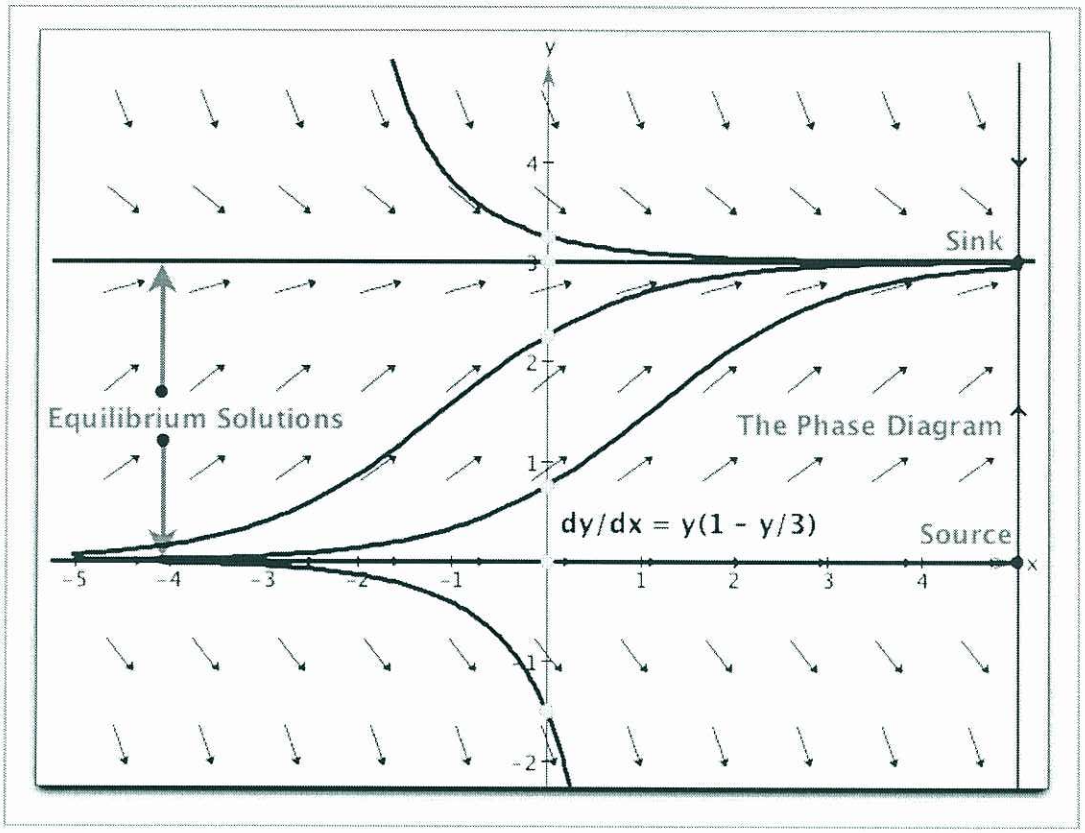
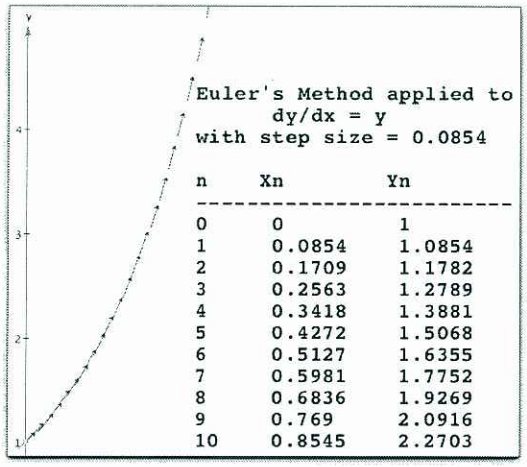


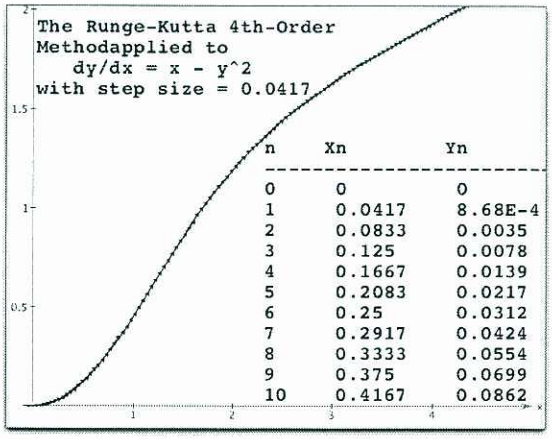
Figure 8: Features of the autonomous differential equation $dy/dx = y(1-y/3)$



Euler's Method applied to
dy/dx = y
with step size = 0.0854

n	Xn	Yn
0	0	1
1	0.0854	1.0854
2	0.1709	1.1782
3	0.2563	1.2789
4	0.3418	1.3881
5	0.4272	1.5068
6	0.5127	1.6355
7	0.5981	1.7752
8	0.6836	1.9269
9	0.769	2.0916
10	0.8545	2.2703

Figure 9: Euler's method applied to $dy/dx = y, y(0) = 1$



The Runge-Kutta 4th-Order
Methodapplied to
dy/dx = x - y^2
with step size = 0.0417

n	Xn	Yn
0	0	0
1	0.0417	8.68E-4
2	0.0833	0.0035
3	0.125	0.0078
4	0.1667	0.0139
5	0.2083	0.0217
6	0.25	0.0312
7	0.2917	0.0424
8	0.3333	0.0554
9	0.375	0.0699
10	0.4167	0.0862

Figure 10: Runge-Kutta applied to $dy/dx = x - y^2, y(0) = 0$

Example E: Finally, TEMATH's differential equation tool can also be used to show the long-term behavior of a solution of a first-order of differential equation such as one that models the cooling of a cup of coffee in a large room (see **Figure 11** below). Here the solution is found by using the adaptive Runge–Kutta–Fehlberg method (RKF). Note that RKF is TEMATH's default method for solving differential equations.

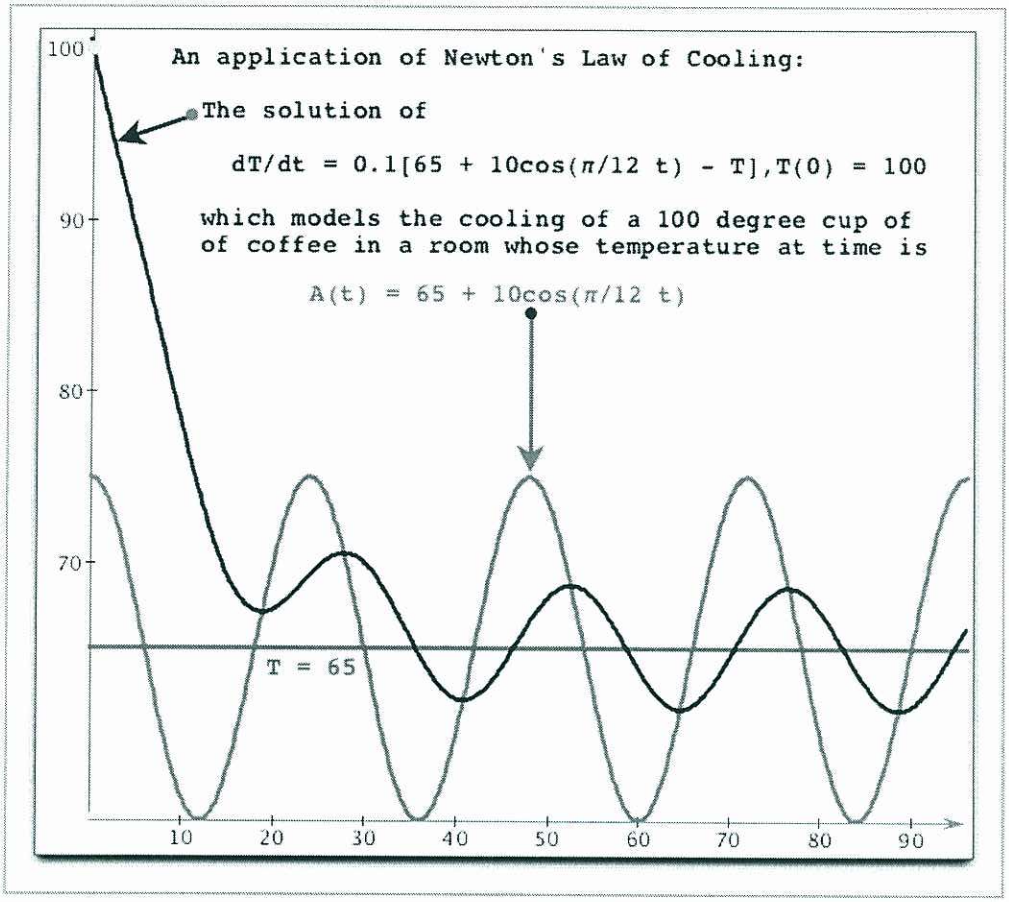


Figure 11: Solution of $dT/dt = 0.1\big(65 + 10\cos(\pi/12\, t) - T\big), T(0) = 100$

Bibliography

[1] James C. Robinson, *Ordinary Differential Equations*, Cambridge University Press, 2007.

[2] Hans Petter Langtangen, *A Primer on Scientific Programming with Python*, Springer, *2009*.

[3] Adam Hausknecht and Robert Kowalczyk, *TEMATH - Tools for Exploring Mathematics Version 3.0a*, 2010.