Second Assignment for Math 128B — Due Fri. 1 Mar. 2002

For some positive number y_0 the solution $y(x)$ of the initial value problem

$$y(0) := y_0 \quad \text{and} \quad dy/dx = x - 1/y \quad \text{for all } x \geq 0$$

remains positive and bounded; this means that some constant (bound) ß satisfies $ß > y(x) \geq 0$ for all $x \geq 0$. Your task is to estimate y_0 as accurately as you can and justify your claims scientifically. You may use any numerical techniques and software available to you provided you understand what they do and describe them well enough that your results can be replicated.

If y_0 is overestimated, the solution $y(x)$ will grow too big; if $y(X)$ is big enough for some $X > 0$ then $y(x)$ will grow at least about as fast as $(x–X)^2/2$ grows as $x \to +\infty$. Can you see why? On the other hand, if y_0 is underestimated then $y(x)$ will plunge down to $y(X) = 0$ for some $X > 0$ and then no solution $y(x)$ will exist for $x > X$; can you see why the plunge must occur? Numerical experiments may help. It is possible to deduce that when the initial value y_0 is just right then $1/(x+1) < y(x) < 1/x$ for all $x > 0$; verifying this claim mathematically leads to a better understanding of the computational challenge involved in estimating the right y_0.

Differential Inequalities

The simplest differential inequalities compare the solutions $y(x)$ and $Y(x)$ of two differential equations $dy/dx = f(x, y)$ and $dY/dx = F(x, Y)$ when $y(0) < Y(0)$ and $f(x, z) < F(x, z)$ for all z and all $x \geq 0$; then also $y(X) < Y(X)$ for all $X > 0$ provided $y(x)$ and $Y(x)$ and their first derivatives remain finite while $0 \leq x < X$.

This is so because the difference $Y–y$ is differentiable and therefore continuous and consequently cannot reverse sign without vanishing first. Let X be the infimum of positive arguments x, if any, for which $Y(x) – y(x) \leq 0$; this would imply that $Y(x) – y(x) > 0$ while $0 \leq x < X$ but $Y(X) – y(X) = 0$, contradicting the sign of the derivative: $Y'(X) – y'(X) = F(X, Y(X)) – f(X, y(X)) > 0$ wherever $Y(X) = y(X)$.

Similarly, if $y(0) > Y(0)$ and $f(x, z) > F(x, z)$ for all z and all $x \geq 0$, then $y(X) > Y(X)$ for every $X > 0$ provided *etc*. In fact, for every $x > 0$ each solution $y(x)$ of $dy/dx = f(x, y)$ is a strictly increasing function of $y(0)$ and of $f(\ldots)$ provided they determine the solution uniquely.

The differential equation we wish to solve has $dy/dx = f(x, y) := x - 1/y$. Wherever the graph of some solution $y(x)$ cuts the graph of $1/x$, the cut goes from left below to right above the latter graph because $y(x) = 1/x$ at the cut implies $y'(x) = 0 > (1/x)' = –1/x^2$ there; this means that the graph of a solution $y(x)$ can't cut the graph of $1/x$ more than once. Consequently the graph of the desired bounded solution $y(x)$ can't cut the graph of $1/x$ at all; that this solution (if it exists) must satisfy $y(x) < 1/x$ for all $x > 0$ follows from the following analysis:

Suppose there were some $X > 0$ beyond which $y(x) > 1/x$ for all $x \geq X$. Then $y'(x) = x - 1/y(x) > 0$ and so this $y(x) > y(X) > 1/X$ while $x \geq X$. Then $y'(x) = x - 1/y(x) > x – X$, so $y(x) > y(X) + \int_X^x (\xi–X)d\xi > 1/X + (x–X)^2/2$, which grows unboundedly with $x > X$. Therefore only unbounded solutions y can ever violate $y(x) < 1/x$.

If the graph of some solution $y(x)$ cuts the graph of $1/(x+1)$, the cut goes from left above to right below the latter graph since $y(x) = 1/(x+1)$ implies $y'(x) = –1 < (1/(x+1))' = –1/(x+1)^2$ there, so no solution's graph can cut the graph of $1/(x+1)$ more than once. An analysis very like the foregoing shows that $y(x) > 1/(1+x)$ if $y(x) > 0$ for all $x \geq 0$; can you verify this?

In fact, if $0 < y(X) \le 1/(X+1)$ for some $X > 0$ then $y(Z) = 0$ for some positive $Z < X + 1/(X+1)$ beyond which this solution $y(x)$ cannot be continued; further analysis reveals that $0 < y(x) \le \sqrt{2Z-2x}$ for $0 \le x \le Z$, which reveals how abruptly such solutions $y(x)$ plunge to zero. This revelation accords with experience of numerical solutions started from too small an initial value y_0.

To summarize: If any solution $y(x)$ exists that stays positive and bounded for all $x \ge 0$, it lies strictly between $1/(x+1)$ and $1/x$. Now, at most one such solution $y(x)$ can exist because the differential equation $dy/dx = x - 1/y$ is *stable* for *decreasing* x. This means that the difference between nearby solutions diminishes as x decreases; here is why:

Suppose $y(x)$ and $y(x) + \Delta y(x)$ are two solutions with $y(X) + \Delta y(X) > y(X) > 0$ for some $X > 0$. Provided both solutions stay positive throughout $0 \le x \le X$, the general existence and uniqueness theory of differential equations ensures that their graphs do not cross (lest solutions through the crossing point not be determined uniquely). Therefore $y(x) + \Delta y(x) > y(x) > 0$ throughout $0 \le x \le X$, and then

$$d\Delta y/dx = f(x, y+\Delta y) - f(x, y) = \Delta y(x)/(y(x)\cdot(y(x)+\Delta y(x))) > 0,$$

and thus $\Delta y(x)$ must be an increasing function of x. This implies $0 < \Delta y(0) < \Delta y(X)$.

If both solutions $y(x)$ and $y(x)+\Delta y(x)$ stayed positive and bounded for all $x \ge 0$ then, as we have seen above, $1/X > y(X) + \Delta y(X) > y(X) > 1/(X+1)$. Consequently $\Delta y(X) \to 0$ as $X \to +\infty$, forcing $\Delta y(0) \to 0$ too. In other words, both positive bounded solutions would have to be the same solution, if it exists.

To prove that the desired positive bounded solution exists, consider two solutions $y(x)$ of the differential equation $dy/dx = f(x, y) := x - 1/y$. One solution $y(x) = \overline{Y}(x; X)$ is determined by $\overline{Y}(X; X) := 1/X$ for some $X > 0$; the other solution $y(x) = \underline{Y}(x; X)$ has $\underline{Y}(X; X) := 1/(X+1)$. Because solutions $y(x)$ are monotonic functions of initial or terminal values, and because their graphs can cut the graphs of $1/x$ and $1/(x+1)$ at most once, we infer for $0 \le x < X$ that $1/(x+1) < \underline{Y}(x; X) = \underline{Y}(0; X) + \int_0^x f(\xi, \underline{Y}(\xi; X))d\xi < \overline{Y}(x; X) = \overline{Y}(0; X) + \int_0^x f(\xi, \overline{Y}(\xi; X))d\xi < 1/x$. Because of stability as x decreases, $0 < \overline{Y}(x; X) - \underline{Y}(x; X) < \overline{Y}(X; X) - \underline{Y}(X; X) = 1/((X(X+1))$. Therefore, as X increases towards $+\infty$ while x is held fixed, $\overline{Y}(x; X)$ decreases and $\underline{Y}(x; X)$ *increases* (do you see why?) towards the same limit-function $\ddot{Y}(x) = \ddot{Y}(0) + \int_0^x f(\xi, \ddot{Y}(\xi))d\xi$; this limit is the desired positive bounded solution, and $y_0 = \ddot{Y}(0)$ is the desired starting value.

Numerical Application of the Foregoing Analysis

The foregoing analysis brings to mind two ways to compute the desired starting value y_0. One way runs right-to-left: Compute $\overline{Y}(0; X)$ and $\underline{Y}(0; X)$, by numerically solving the differential equation $dy/dx = f(x, y)$ backwards from $\overline{Y}(X; X) := 1/X$ and $\underline{Y}(X; X) := 1/(X+1)$, for an increasing sequence of values X until the computed $\overline{Y}(0; X)$ and $\underline{Y}(0; X)$ approach each other closely enough that any number between them is an acceptable estimate of y_0. But what is to be made of computed values that violate the expected inequality $\underline{Y} < \overline{Y}$?

Another way runs left-to-right: starting from any guess Y_0 obtain a numerical approximation $Y(x)$ to the solution of $dY/dx = f(x, Y)$ starting from $y(0) = Y_0$ and ending with $Y(X)$ when first either $Y(X) \ge 1/X$ or $Y(X) \le 1/(X+1)$. These events can be detected during the numerical process used in the ODE-solvers in, say, Matlab 5 by invoking …

```
OPTIONS = ODESET(OLDOPTS, ..., 'events', 'on', ...)  and
function  F = odefile(x, y, 'events', ...), etc. (Look them up.)
```

Then define a function $Ø(Y_0) := 1/X$ or $-1/X$ respectively according to which event stops the ODE-solver, and use a root-finder to solve $Ø(Y_0) = 0$ for Y_0. What if $Ø$ is not monotonic?

Differential Inequalities and Error-Estimation

Attempts to solve the Ordinary Differential Equation (ODE) $dy/dx = f(x, y)$ numerically incur errors from a variety of sources:

- $f(x, y)$ is computed contaminated by roundoff and other uncertainties.
- The ODE-solver's arithmetic is contaminated by roundoff.
- The ODE-solver uses a discrete formula with a presumably tolerable error in each step.

The last source of error is normally the worst, and the only source that will be discussed here.

Ideally the computed approximation $Y(x)$ of the desired solution $y(x)$ should satisfy a nearby differential equation $dY/dx = f(x, Y) + \Delta f$ upon which a tolerance $\mu > |\Delta f|$ has been imposed by whoever invoked the ODE-solver. It would attempt to keep its discretization *error-per-unit-step* below that tolerance by keeping its stepsizes barely small enough. Assuming it succeeded, estimates that bracket the desired solution could be computed by invoking the ODE-solver upon

$$d\underline{Y}/dx = f(x, \underline{Y}) - \mu \quad \text{and} \quad d\overline{Y}/dx = f(x, \overline{Y}) + \mu$$

to obtain two numerical estimates of $\underline{Y}$ and $\overline{Y}$ that satisfied $\underline{Y} < y < \overline{Y}$ despite discretization errors-per-unit-step smaller than μ during their computation. This would be so because, for example, the computed $\underline{Y}$ would actually satisfy $d\underline{Y}/dx = f(x, \underline{Y}) - \mu + \Delta f < f(x, \underline{Y})$. The tolerance μ is unlikely to be chosen much smaller than the uncertainty that f inherits from its computation; augmenting μ by that uncertainty in the two differential equations solved for $\underline{Y}$ and $\overline{Y}$ provides an interval $\underline{Y} < y < \overline{Y}$ that indicates how f's uncertainty propagates into y.

But ODE-solvers like those in Matlab 5 do not respect an imposed tolerance μ upon their discretization errors-per-unit-step. Such a tolerance would force the ODE-solver to take too many too-tiny steps when solving either differential equations that change abruptly, or stable differential equations whose solutions become lethargic after initial transient behavior has died away. Characteristic of stable ODEs is their tendency to "forget" early errors; consequently ODE-solvers optimized for higher speed (longer and fewer steps) accept only tolerances upon their discretization *errors-per-step*. Consequently the error-estimation technique described in the previous paragraph cannot be used unless modified to take account of the ODE-solver's stepsizes. I know no straightforward way to do this with Matlab's ODE-solvers.

Instead, the simplest way to assess the accuracy of a computed solution may be to recompute it with error-tolerances at least an order of magnitude smaller than before and see how many figures agree in the two computed solutions. This technique is simple but too vulnerable to deceptive phenomena, especially when the ODE has more than one stable solution selectable by initial conditions, as happens in simulations of switching circuits. Every ODE-solver's error is biased in a direction, dependent upon the solution's derivatives, that persists when error-tolerances are diminished and may contribute decisively to the solution unless error-tolerances are far tinier than might at first have been thought sufficiently small.

Another way to assess accuracy is to use a few sufficiently different numerical methods to solve the same ODE, and see how well they agree. This way is not foolproof either, but it appears to be the way the designers of Matlab's ODE-solvers had in mind. By practically precluding the imposition of a tolerance upon error-per-unit-step, they have inhibited the development and use of reliable (and costly) error-bounding techniques based upon differential inequalities that can take account of the ODE's intrinsic uncertainty too. Speed drives all else out of mind.

Matlab Programs to Compute Solutions Right-to-Left

Two numerical approximations $[\overline{Y}(x), \underline{Y}(x)]$ of the solution $y(x)$ of the ODE $dy/dx = x - 1/y$ were computed simultaneously over an interval $0 \le x \le Xend$ starting at $x = Xend := 5$ with $[\overline{Y}(Xend), \underline{Y}(Xend)] = 1.0 ./ [Xend, 1+Xend]$ and running x down to 0. Other values Xend greater than 4 gave very similar results. The ODEfile defining $f(x, y) = x - 1/y$ for both solutions $y = \overline{Y}$ and $y = \underline{Y}$ simultaneously was called f2.m :

```
function  f = f2(x, y, flag)
%  f2  is a  Matlab ODEfile  to solve  y' = x - 1/y  for two solutions
%  [y1(x), y2(x)]  simultaneously,  differing in their initial values.
if (nargin<3)|isempty(flag),
     f = x - 1.0 ./ y(:) ;
  else switch(flag)
    case 'jacobian',
       f = diag( 1.0 ./ (y.*y) ) ;
    otherwise
         f2flag = flag,
         error('f2(x, y, Unknown Flag)')
   end,   end
```

Note how invoking f2(x, y, 'jacobian') delivers the Jacobian matrix of first partial derivatives of $[f(x, \overline{Y}); f(x, \underline{Y})]$ with respect to $[\overline{Y}; \underline{Y}]$, namely $diag(1.0 ./ [\overline{Y}^2, \underline{Y}^2])$, for use by some of Matlab's five ODE-solvers. Scripts were written to invoke each of these with the same given Xend and error-tolerances; the script called f2_113.m invokes ODE-solver ode113 :

```
%  Matlabscript to run  ODE-solver  ode113  on  f2
Xend,  Atol, %  RelTol = 100*eps  is the minimum  Matlab  allows.
OPTIONS = odeset('RelTol',100*eps, 'AbsTol',Atol,  ...
                 'Jacobian','on', 'Vectorized','on') ;
Xspan = [Xend, 0] ;   Yend = 1.0 ./[Xend, 1+Xend] ;
[X,Y] = ode113('f2', Xspan, Yend, OPTIONS) ;
n = length(X),  XYY = [X(n), Y(n,:)]
plot(X,Y(:,1),'r', X,Y(:,2),'g')
```

Four more scripts f2_15s.m , f2_45.m , f2_23.m , f2_23s.m invoked their respective ODE-solvers. Each script first displays previously assigned values Xend and tolerance Atol upon absolute error-per-step; the tolerance $100*eps \approx 2.2204\mathbf{e}{-14}$ upon relative error-per-step was the smallest Matlab's ODE-solvers would accept. Then the script sets Xspan to tell the ODE-solver to run independent variable x down from Xend to 0, initializing $[\overline{Y}(Xend), \underline{Y}(Xend)]$ to Yend so that $\overline{Y} > \underline{Y}$ and the true solution y lies between them except for numerical errors. After invoking the ODE-solver to generate arrays X and Y , the script displays the number n of steps taken, the final value $X(n) = 0$ of x, the computed solutions' $Y(n,:) = [\overline{Y}(0), \underline{Y}(0)]$, and the solutions' graphs.

Each ODE-solver tried to keep the error-per-step in each computed solution $Y(x)$ smaller than $100*eps*|Y(x)| + Atol$. Each such error tends to be forgotten exponentially, as x decreases, at a rate determined by $\partial f/\partial y \approx 1./Y(x)^2$. Numerical experiments indicated that $Y(x)$ is slightly bigger than $1/(1+x)$, so errors have to decay extremely rapidly when x is near Xend, slowly when x is near 0. Without knowing the ODE-solver's stepsizes when x is near 0 we can't say how much bigger than $100*eps*|Y(0)| + Atol$ is the error in $Y(0)$; not much bigger, we hope.

Numerical Results

Exhibited here are results for two values 10^{-6} and 10^{-12} of Atol , the tolerance upon absolute error-per-step. Tabulated for each of Matlab's five ODE-solvers ode… are the number n of steps (actually the number-plus-one of steps not retried with a smaller stepsize to keep the local error below the assigned tolerance), and the computed values $\overline{Y}(0)$ and $\underline{Y}(0)$ between which the desired y_0 would lie but for numerical errors.

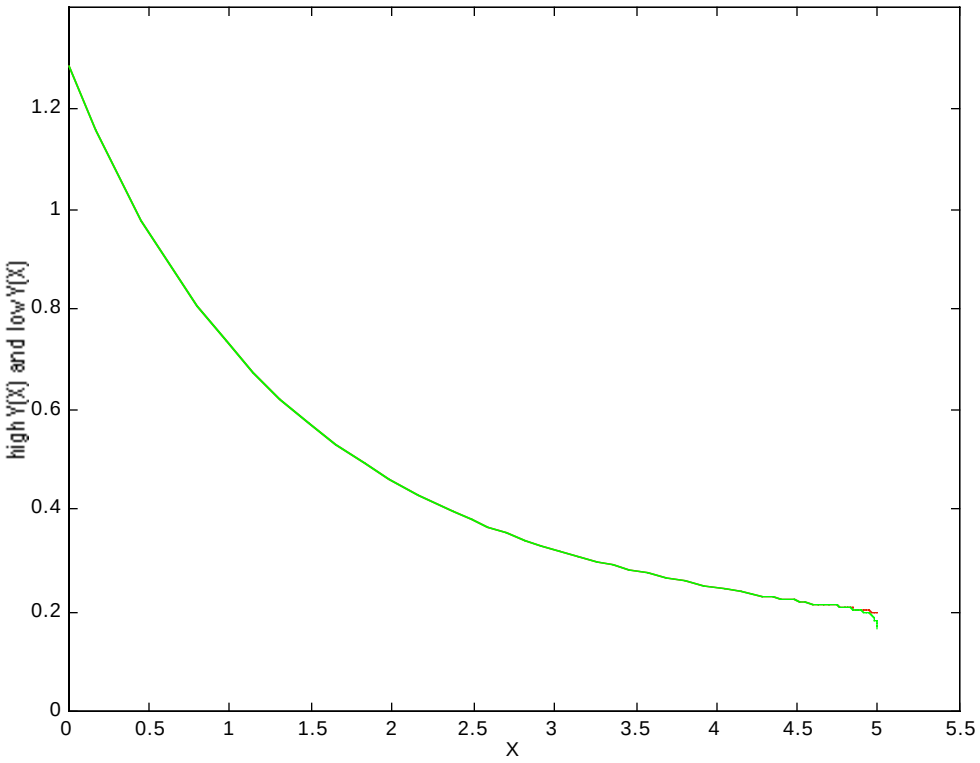
These results come from Matlab 5.3 run on a Wintel PC; results from a Mac Quadra (68040) are almost the same.

Atol = 0.000001 :

ode:	ode23s	ode23	ode15s	ode45	ode113
n:	246	134	89	217	75
$\overline{Y}(0)$:	1.28358085363422	1.28359687743776	1.28359822084272	1.28359816310455	1.28359918899254
$\underline{Y}(0)$:	1.28358085363422	1.28359687743776	1.28359820787955	1.28359816310455	1.28359918905259

The last column's $\overline{Y}(0) < \underline{Y}(0)$ is not a typo; ode113 really did compute solutions $\overline{Y}(x)$ and $\underline{Y}(x)$ whose graphs crossed instead of preserving the relationship $\overline{Y}(x) > \underline{Y}(x)$ established at x = Xend . This occurred because our ODE is too stiff for ode113 to treat accurately when x is very near Xend ; the sharp upward surge in $\underline{Y}(x)$ was extrapolated too far. Here are graphs:

Plots of $\underline{Y}(x)$ (green) and $\overline{Y}(x)$ (red)



Barely visible near x = 4.93 is the point where the graphs of $\underline{Y}(x)$ and $\overline{Y}(x)$ cross.

Each ODE-solver's $\overline{Y}(0)$ and $\underline{Y}(0)$ agree to at least 8 dec., but they cannot all be correct because the different solvers' results agree to at most 5 dec., or 6 if ode23s is deemed an outlier. Had only one ODE-solver been invoked, its results' agreement would have misled us into overestimating their accuracy.

Atol = 0.000000000001 :

ode:	ode23s	ode23	ode15s	ode45	ode113
n:	24656	10432	613	3033	231
$\overline{Y}(0)$:	1.283598708681	1.28359871046153	1.28359871045841	1.2835987104635	1.28359871046502
$\underline{Y}(0)$:	1.283598708681	1.28359871046153	1.28359871045967	1.2835987104635	1.28359871046502

Once again a solver, this time ode15s, has crossed the graphs of $\overline{Y}$ and $\underline{Y}$. Once again, each solver's results agree rather more closely, better than 11 dec., than the 8 dec. to which all five solvers' results agree. Once again, ode23s seems to be the outlier on the low side; the other solvers' results agree to almost 11 dec. How many digits would you trust now?

Ch. 8 of *Using Matlab*, `<…/matlab/help/pdfdocs/using_ml.pdf>`, gives advice about Matlab's ODE-solvers, describing the circumstances for which each is recommended. Still, prudent scientists and engineers will trust their numerical results only if corroborated by mathematical proofs, when they are not too tedious, and always by recomputations using diverse numerical methods. Proofs are precious, and so are computable error-bounds; but they are often more expensive than the results they corroborate are worth. For instance, error-bounds for ODEs' initial-value problems rather more general than this assignment's can be computed, within limits posed by severe nonlinearity, by a method outlined very roughly in class note *Ellipsoidal Error Bounds for Trajectory Calculations* `<http://www.berkeley.edu/~wkahan/Math128/Ellipsoi.pdf>` .
The method augments a given ODE dy/dt = f(t, y) of dimension N by an auxiliary ODE of dimension N(N+1)/2 to compute the error-bound simultaneously with the computed solution y , and requires among other things that $\partial f/\partial y$ be made available. The price is daunting when N is big. If this method is the best available except in special cases, then error-bounds for solutions of ODEs are unlikely to become commonplace. Corroboration by recomputation may turn out to be the best that can be done except in special cases.

The Answer: $y_0 = 1.283598710463599523\ldots$.

The foregoing task, to find the critical initial value y_0 , was posed by A. Tissier as problem #6551, p. 694 in the *Amer. Math. Monthly* **94** (1987), solved on pp. 631-5 and 657-9 of **96** (1989) with the aid of an *Airy* function.

When x is big enough, y(x) can be approximated by the first few terms of a series expansion

$$y(x) \approx 1/x - 1/x^4 + 5/x^7 - 44/x^{10} + 539/x^{13} - 8337/x^{16} + O(1/x^{19})$$

which could not be obtained directly by Maple V r3 on my Mac Quadra because of a division-by-zero error. Another approximation useful only when x is big enough is a continued fraction

$$y(x) \approx \cfrac{1}{x + \cfrac{1}{x^2 + \cfrac{4}{x + \cfrac{19}{4x^2 + \cfrac{535}{19x + \cfrac{79432}{535x^2 + O(\frac{1}{x})}}}}}} \ .$$

Both expansions are obtained by treating 1/x instead of x as the independent variable, and seeking a solution y(x) representable formally as a (possibly nonconvergent) power series in 1/x around 1/x = 0 ; then it turns out that $x\cdot y(x) \to 1$ as $1/x \to 0$, and that the power series for $x\cdot y(x)$ involves only powers of x^{-3} . But neither expansion reveals y(0) , for which the only method I know to work well is numerical integration of the ODE backwards from an estimate for sufficiently large x obtained from one of those expansions. The result from a Runge-Kutta method carried out by Maple V with extravagant precision is $y_0 = 1.2835987104635995\ldots$; and a Fortran program running a similar method carefully on an IBM PC carrying 64 sig. bits got $y_0 = 1.2835987104635995234$.