Report about Homotopy Continuation

I solved $f(x) = 0$ with the homotopy $h(t, x) = tf(x) + (1 - t)g(x)$ by integrating the differntial equation

$$\begin{cases} x'(t) = -\dfrac{h_t(t, x)}{h_x(t, x)} \\ x(0) = x_0 \end{cases}$$

using Euler's method with stepsize $\Delta t = 1/10$.

First, a Maple script computes the terms h_t and h_x appearing in the differential equation. In our test run we choose

$$f(x) = \cos x - x, \qquad g(x) = f(x) - f(x_0) \qquad \text{and} \qquad x_0 = 1.$$

The script `func.mpl` follows:

```
# func.mpl -- compute forcing term for the differential equation
# used in the homotopy continuation method for solving f(x)=0
restart;
f:=x->cos(x)-x;

x0:=1.0;
getx0:=unapply(x0,x);
g:=unapply(f(x)-f(x0),x);
df:=unapply(diff(f(x),x),x);
dg:=unapply(diff(g(x),x),x);
h:=unapply(t*f(x)+(1-t)*g(x),[t,x]);
dhdx:=unapply(diff(h(t,x),x),[t,x]);
dhdt:=unapply(diff(h(t,x),t),[t,x]);
F:=unapply(-dhdt(t,x)/dhdx(t,x),[t,x]);
with(CodeGeneration,C);
C(F);
fname:="func.i";
fd:=open(fname,WRITE);
fprintf(fd,"/* func.c -- by Maple */\n");
close(fd);
myopt:=output=fname,
       declare=[x::numeric,t::numeric];
C(F,myopt);
C(f,myopt);
C(g,myopt);
C(h,myopt);
C(df,myopt);
C(dg,myopt);
C(dhdt,myopt);
C(dhdx,myopt);
C(getx0,myopt);
```

The Maple script creates a file `func.i` containing the needed C subroutines for computing a solution to the differential equation.

```
/* func.c -- by Maple */
#include <math.h>

double F(double t, double x)
{
    return (0.4596976941e0 / (t * (-sin(x) - 0.1e1) + (0.1e1 -
                t) * (-sin(x) - 0.1e1)));
}

#include <math.h>

double f(double x)
{
    return (cos(x) - x);
}

#include <math.h>

double g(double x)
{
    return (cos(x) - x + 0.4596976941e0);
}

#include <math.h>

double h(double t, double x)
{
    return (t * (cos(x) - x) + (0.1e1 - t) * (cos(x) - x +
            0.4596976941e0));
}

#include <math.h>

double df(double x)
{
    return (-sin(x) - 0.1e1);
}

#include <math.h>

double dg(double x)
{
    return (-sin(x) - 0.1e1);
}

double dhdt(double t, double x)
```

```
{
    return (-0.4596976941e0);
}

#include <math.h>

double dhdx(double t, double x)
{
    return (t * (-sin(x) - 0.1e1) + (0.1e1 - t) * (-sin(x) - 0.1e1));
}

double getx0(double x)
{
    return (0.10e1);
}
```

The program `topy.c` includes the file `func.i` and then solves the differential equation using Euler's method.

```
/*
    topy.c -- Solve f(x)=0 by the homotopy method
    Written Septeber 29, 2012 by Eric Olson for Math 701
*/

#include <stdio.h>
#include "func.i"

#define N 10

main(){
    printf("# topy -- Solve f(x)=0 by the homotopy method\n"
    "# Written September 29, 2012 for Math 701 by Eric Olson\n\n");

    int i;
    double dt=1.0/N;
    double x=getx0(0.0);
    printf("# %2s %8s %24s\n","n","t_n","x_n");
    for(i=0;i<N;i++){
        double t=i*dt;
        printf("%4d %8.4f %24.15e\n",i,t,x);
        x=x+dt*F(t,x);
    }
    printf("%4d %8.4f %24.15e\n",N,N*dt,x);
    printf("\n# f(x(1))=%24.15e\n",f(x));
    return 0;
}
```

The resulting output

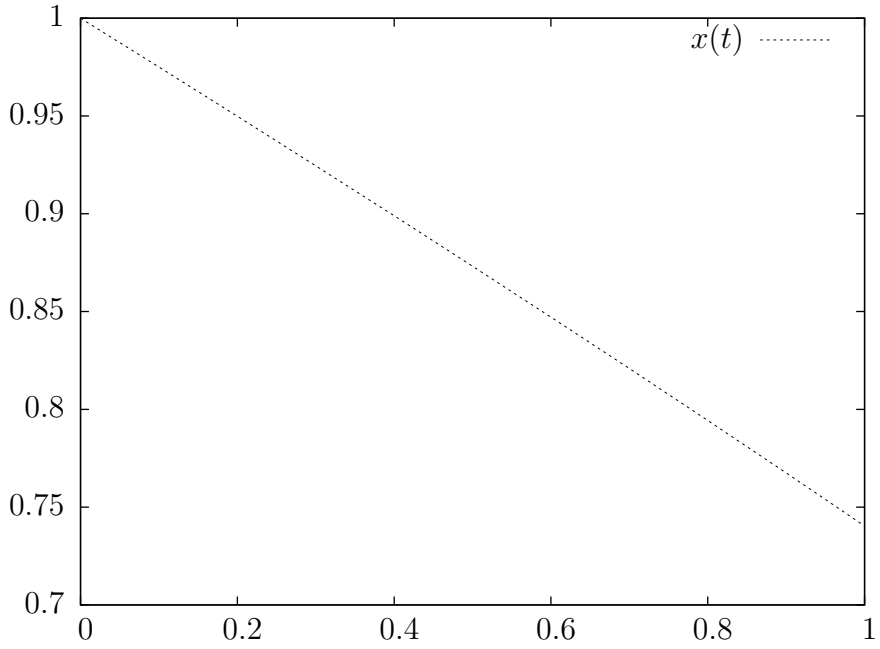
```
# topy -- Solve f(x)=0 by the homotopy method
# Written September 29, 2012 for Math 701 by Eric Olson

#  n      t_n                      x_n
   0   0.0000    1.000000000000000e+00
   1   0.1000    9.750363867857545e-01
   2   0.2000    9.498849897486028e-01
   3   0.3000    9.245342274005630e-01
   4   0.4000    8.989718975600242e-01
   5   0.5000    8.731851168990508e-01
   6   0.6000    8.471602536636458e-01
   7   0.7000    8.208828525619060e-01
   8   0.8000    7.943375506400665e-01
   9   0.9000    7.675079827553092e-01
  10   1.0000    7.403766749982386e-01

# f(x(1))=  -2.162156049025259e-03
```

shows that a solution to $f(x) = 0$ is $x \approx 0.7404$.

Output from `topy.c` is stored in the file `x.dat` and processed with Gnuplot to create the graph



using the Gnuplot script

```
1  set terminal pslatex
2  set output "graph.tex"
```

```
3 set style data lines
4 plot "x.dat" using 2:3 lt 3 ti "$x(t)$"
```

Note that all text output in `x.dat` is preceded by a '#' sign which Gnuplot treats as comments when processing the data.