Scientific Computing

A practical Companion

1st Notebook

© Copyright 2008, Korteweg-de Vries instituut, Universiteit van Amsterdam

This notebook can be downloaded from the location:

http://staff.science.uva.nl/~walter/SC/Notebooks/SC08-1.nb

Author:

Walter Hoffmann (Korteweg-de Vries Institute for Mathematics, UvA)

January - February, 2008

Introduction

Outline of this course

The most advanced computer platform becomes

WORTHLESS when a BAD ALGORITHM is used.

We concentrate on solving problems that are stated as (large scale) Differential Equations.

Type of Problems:

I) Ordinary Differential Equations (ODE's)

(One independent variable; in general time, position (place))

IA) Initial Value Problems (IVP's)

IB) Boundary Value problems (BVP's)

II) Partial Differential Equations (PDE's)

(Two or more independent variables; in general time and position)

IIA) Elliptic Problems (Diffusion;

Laplace* equation: $u_{xx} + u_{yy} = 0$

Poisson** equation: $u_{xx} + u_{yy} = f(x, y)$

)

IIB) Parabolic Problems (Heat equation: $u_t = \kappa\, u_{xx}$)

IIC) Hyperbolic Problems (Wave equation: $u_{tt} = c^2\, u_{xx}$)

*) Pierre-Simon Laplace 1749 - 1827

**) Simeon Poisson 1781 - 1840

Solution methods

either explicit or implicit

Representation of the Solution:

Solution as linear combination of basis functions; leads to

- Finite Element Method (FEM);
- Finite Volume Method

Solution calculated in finitely many points that are chosen on a grid;
interpolation methods necessary to calculate solution in intermediate points.

- Finite Difference method (method of choice during lab-work in this course)

Class is combination of "PUSH" and "PULL":

PUSH: What every graduate student in Scientific Computing should know about solving ODE's and PDE's

PULL: What has to be explained from the concepts that are used in lab-work

Basic Tools

Mean Value Theorem (MVT):

Let f be a function that is continuous and differentiable on [a,b].

Then there exists a point $\xi \in [a,b]$ such that

$$\frac{f(b) - f(a)}{b - a} = f'(\xi),$$

or analogously:

$$\frac{f(x+h) - f(x)}{h} = f'(x + \theta h); \quad 0 \le \theta \le 1.$$

Attractive versus Repulsive Fixed-point:

Let $g: D \to D$ be a given function;

the row $\{R_n\} = \{x_0, x_1, x_2, \ldots, x_n\}$ is defined by

$$x_{k+1} = g(x_k), \text{ for given } x_0 .$$

Any value (argument, result) s with

$$s = g(s)$$

is called a fixed-point of the iteration $x_{k+1} = g(x_k)$

("what goes in comes out")

Convergence:

For any choice x_0 that is sufficiently close to s, the row R_n may or may not converge to s, depending on the derivative $g'(s)$.

For a given row R_n to converge to s, we ultimately require:

$$\left| \frac{x_{n+1} - s}{x_n - s} \right| < 1$$

From the definitions and MVT we have:

$$\left| \frac{x_{n+1} - s}{x_n - s} \right| = \left| \frac{g(x_n) - g(s)}{x_n - s} \right| = |g'(\xi)|,$$

$$\xi \text{ between } x_n \text{ and } s.$$

Convergence must (eventually) occur when `|g'(s)| < 1,`

`s` is called an *attractive* fixed-point.

Whenever `|g'(s)| > 1,` convergence is not possible; `s` is called a *repulsive* fixed-point.

For the case `|g'(s)| = 1,` no general statement is applicable.

Taylor Series Expansion (Taylor):

The following is known as Taylor's *) theorem.

Let `f` be a function that is continuous on `[a,b]` such that the derivatives of `f` of order up to and including `n` are defined and continuous on `[a,b]` then for each $x \in [a,b]$ there exists a $\theta = \theta(x)$ with $0 \le \theta \le 1$ such that

$$f(x) = f(a) + (x-a) f'(a) + \frac{1}{2!}(x-a)^2 f''(a) + \frac{1}{3!}(x-a)^3 f'''(a) + \ldots + \frac{1}{n!}(x-a)^n f^{(n)}(a + \theta(x-a)); \qquad 0 \le \theta \le 1.$$

which, apart from the *remainder term*, is a power series in `(x - a);` the first "so many terms", form a polynomial in `x` .

With names of the variables changed appropriately, the expansion can also be written as:

$$f(x+h) = f(x) + h f'(x) + \frac{1}{2!} h^2 f''(x) + \frac{1}{3!} h^3 f'''(x) + \ldots + \frac{1}{n!} h^n f^{(n)}(x + \theta h); \quad 0 \le \theta \le 1.$$

Observe that in this formula x is assumed to be constant and h is variable resulting in a power series in *h* for the first "so many" terms.

The following well known series expansions are special cases of series expansions stemming from Taylor's theorem with the choice `a=0;` the resulting approximation of a function in the form of a powerseries is called a Maclaurin**) series:

$$\exp(x) = 1 + x + \frac{1}{2}x^2 + \frac{1}{6}x^3 + \ldots \quad ; \text{ (for all } x)$$

$$\cos(x) = 1 - \frac{1}{2}x^2 + \frac{1}{4!}x^4 - \frac{1}{6!}x^6 + \ldots \quad ; \text{ (for all } x)$$

$$\sin(x) = x - \frac{1}{6}x^3 + \frac{1}{5!}x^5 + \ldots \quad ; \text{ (for all } x)$$

$$\ln(1+x) = x - \frac{1}{2}x^2 + \frac{1}{3}x^3 - \frac{1}{4}x^4 + \ldots \quad ; -1 < x \leq 1$$

*) Brook Taylor 1685 - 1731

**) Colin Maclaurin 1698 - 1746

Solving differential equations I

Visualization of Vectorfield

Visualization of families of solutions of ODE's.

We use the package VisualDSolve (developed at Amstel Institute, UvA) that can be downloaded from http://staff.science.uva.nl/~walter/SC/Notebooks/VisualDSolve.m

Loading the package by

In[1]:=

```
<< VisualDSolve`
```

Some parameters are set.

In[2]:=

```
SetOptions[VisualDSolve,
  PlotStyle → {{Blue, Thickness[0.01]}, {Red, Thickness[0.01]},
    {Green, Thickness[0.01]}, {Magenta, Thickness[0.01]},
    {Yellow, Thickness[0.01]}}];
```

The package was developed under *Mathematica* 4; in *Mathematica* 5 a number of harmless messages are sort of a nuisance. We suppress their generation by:

In[3]:=

```
Off[NDSolve::precw]
```

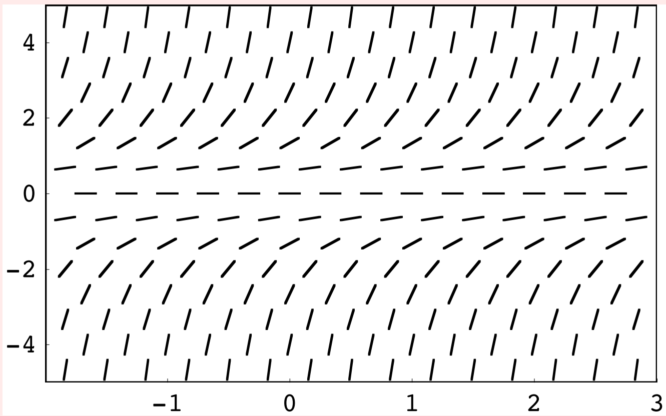
An innocent looking ODE:

$$x'(t) = x(t)^2$$

Draw its vectorfield for a finite interval `[0, 2]` for `t` and a symmetrical interval for the `x`-value:

In[4]:=

```
VisualDSolve[x'[t] == x[t]^2,
    {t, -2, 3}, {x, -5, 5}, DirectionField → True];
```



Observe that negative values of `t` also may be used:

Two examples of a solution curve having `x(0) = 1` and `x(0) = -1` respectively.

In[5]:=

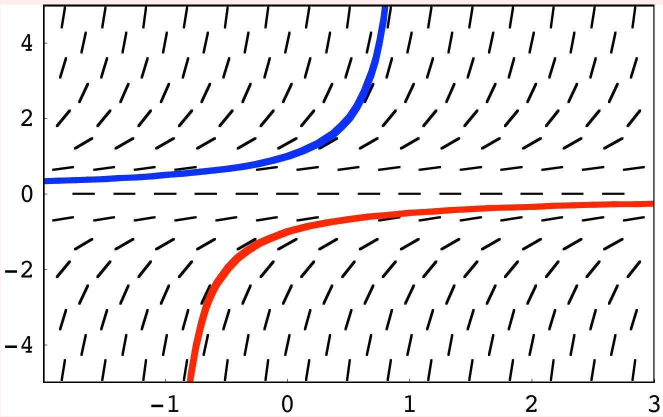
```
VisualDSolve[x'[t] == x[t]^2, {t, -2, 3}, {x, -5, 5},
    DirectionField → True, InitialValues → {{0, 1}, {0, -1}}];
```

```
NDSolve::mxst : Maximum number of 500 steps
   reached at the point t == 0.9999997870126447`. More…

VDS::mxstep :
 NDSolve ran into a MaxSteps limitation for the initial value
   {0., 1.}  and only computed the orbit out to t =
   0.9999997870126447`. If a larger domain is
   needed,  increase the MaxSteps option setting.

NDSolve::mxst :
 Maximum number of 500 steps reached at the point t == -1.. More…

VDS::mxstep :
 NDSolve ran into a MaxSteps limitation for the initial value
   {0., -1.}  and only computed the orbit out to t = 3.`. If a
   larger domain is needed,  increase the MaxSteps option setting.
```



What do we conclude about the existence of solutions for a given ODE ?

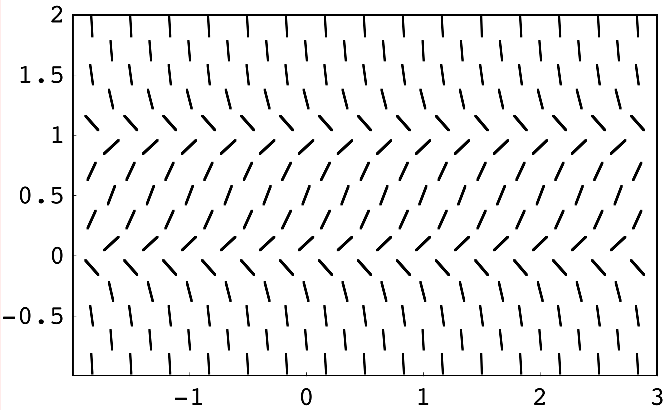
Another ODE for illustration :

Consider

$$x'(t) = 10\,x(t) - 10\,x(t)^2$$

In[6]:=

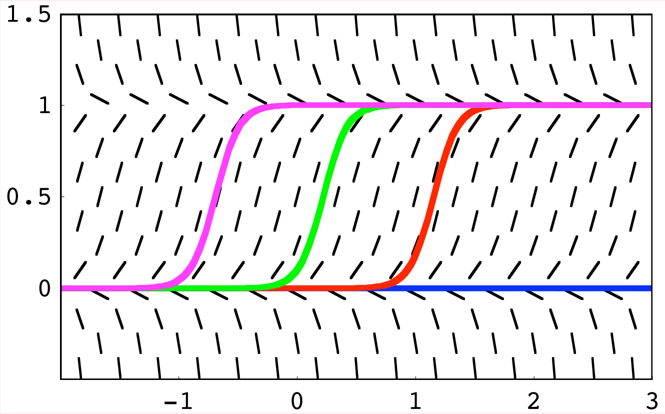
```
VisualDSolve[x'[t] == 10 x[t] - 10 x[t]^2,
    {t, -2, 3}, {x, -1, 2}, DirectionField → True];
```



Draw solution curves through (0,0), (0,0.00001), (0,0.1), (0,0.999)

In[7]:=

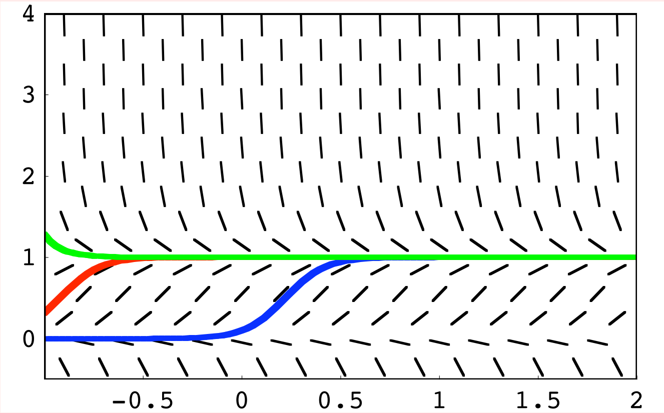
```
VisualDSolve[x'[t] == 10 x[t] - 10 x[t]^2,
    {t, -2, 3}, {x, -0.5, 1.5}, DirectionField → True,
    InitialValues → {{0, 0}, {0, 0.00001}, {0, 0.1}, {0, 0.999}}];
```



Some values x(0) > 1 give :

In[8]:=

```
VisualDSolve[x'[t] == 10 x[t] - 10 x[t]^2,
    {t, -1, 2}, {x, -0.5, 4}, DirectionField → True,
    InitialValues → {{0, 0.1}, {0, 0.9999}, {0, 1.00001}}];
```

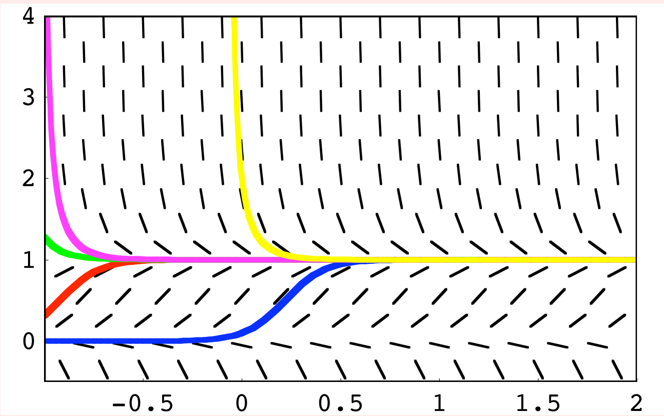


In[9]:=

```
VisualDSolve[x'[t] == 10 x[t] - 10 x[t]^2, {t, -1, 2},
    {x, -0.5, 4}, DirectionField → True, InitialValues →
      {{0, 0.1}, {0, 0.9999}, {0, 1.00001}, {0, 1.00004}, {0, 2}}];
```

```
NDSolve::mxst : Maximum number of 500
   steps reached at the point t == -0.0693147. More…

VDS::mxstep :
 NDSolve ran into a MaxSteps limitation for the initial value
   {0., 2.}  and only computed the orbit out to t = 2.`. If a
   larger domain is needed,  increase the MaxSteps option setting.
```



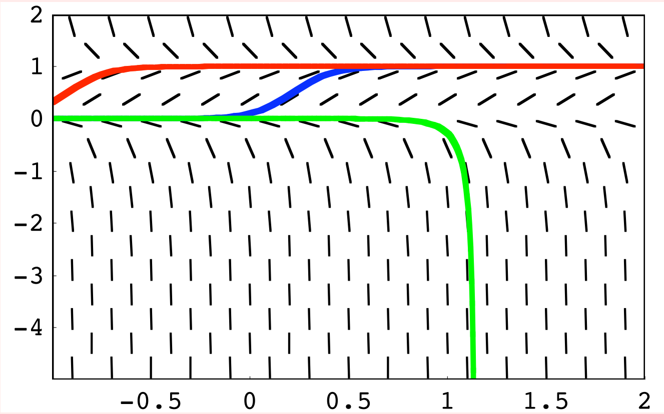
A negative value for `x(0)`:

In[10]:=

```
VisualDSolve[x'[t] == 10 x[t] - 10 x[t]^2,
    {t, -1, 2}, {x, -5, 2}, DirectionField → True,
    InitialValues → {{0, 0.1}, {0, 0.9999}, {0, -0.00001}}];
```

```
NDSolve::mxst : Maximum number of 500 steps
   reached at the point t == 1.1511399822095694`. More…

VDS::mxstep :
 NDSolve ran into a MaxSteps limitation for the initial value
   {0., -0.00001}  and only computed the orbit out
   to t = 1.1511399822095694`. If a larger domain
   is needed,  increase the MaxSteps option setting.
```



Simple numerical algorithm for solving an ODE ("Forward Euler").

The method of solution we use, is known as Forward Euler*) ; it is based on the following approximation:

$$\frac{x_{n+1} - x_n}{\Delta t} \approx x'(t_n).$$

From this we arrive at:

$$x_{n+1} = x_n + \Delta t\, x'(t_n).$$

For the given ODE

$$x'(t) = 10\, x(t) - 10\, x(t)^2$$

this yields

$$x_{n+1} = x_n + \Delta t\left(10\,x_n - 10\,x_n^2\right) = x_n\,(1 + 10\,\Delta t - 10\,\Delta t\,x_n).$$

*) Leonhard Euler 1707 – 1783

For evaluating the required iterations, we use the *Mathematica* routines NestList and ListPlot.

We set `a = 10 Δt` :

In[11]:=

```
IterFunction[a_][x_] := x*(1 + a - a*x);
```

We use Forward Euler to get a numerical solution for the case `x(0) = 0.1` .

We try various values for the increment `Δt` ;

a choice `Δt`= 0.08 leads to a = 0.8 giving

In[12]:=

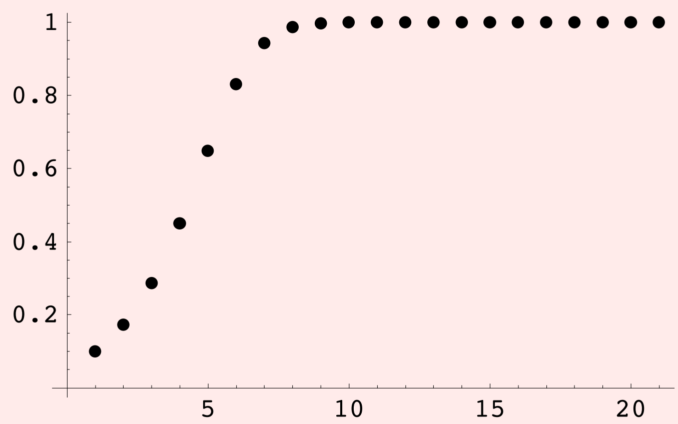
```
g[x_] := IterFunction[0.8][x];
```

In[13]:=

```
NestList[g, 0.1, 20]
ListPlot[%, PlotStyle -> PointSize[0.02]];
```

Out[13]=

```
{0.1, 0.172, 0.285933, 0.449273,
 0.647214, 0.829877, 0.942822, 0.985949,
 0.997032, 0.999399, 0.99988, 0.999976,
 0.999995, 0.999999, 1., 1., 1., 1., 1., 1., 1.}
```



Larger values for the increment `Δt` lead to larger values for a as is shown:

In[15]:=

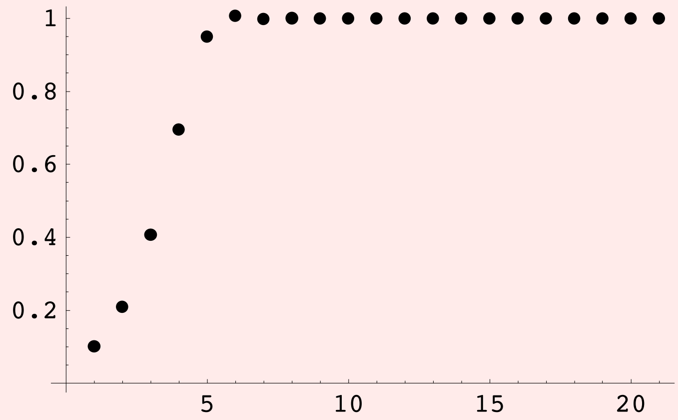
```
g[x_] := IterFunction[1.2][x];
```

In[16]:=

```
NestList[g, 0.1, 20]
ListPlot[%, PlotStyle -> PointSize[0.02]];
```

Out[16]=

```
{0.1, 0.208, 0.405683, 0.695008, 0.949374,
 1.00705, 0.99853, 1.00029, 0.999942, 1.00001,
 0.999998, 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.}
```



The choice $\Delta t = 0.2$ leads to a = 2 giving:

In[18]:=

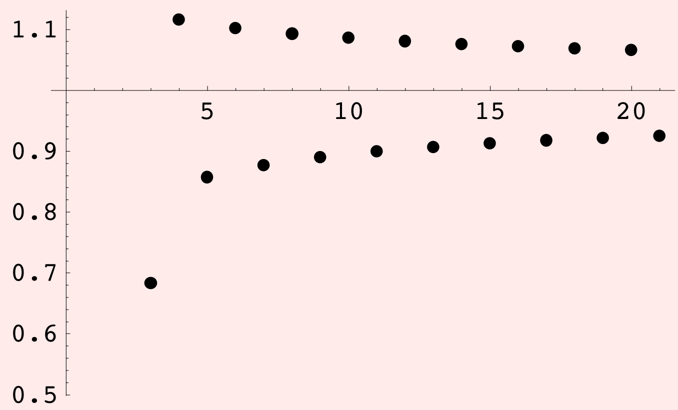
```
g[x_] := IterFunction[2.0][x];
```

In[19]:=

```
NestList[g, 0.1, 20]
ListPlot[%, PlotStyle -> PointSize[0.02]];
```

Out[19]=

```
{0.1, 0.28, 0.6832, 1.11608, 0.856977, 1.10211,
 0.877035, 1.09272, 0.89008, 1.08576, 0.899537,
 1.08028, 0.906834, 1.07581, 0.9127, 1.07206,
 0.917558, 1.06885, 0.921671, 1.06606, 0.925215}
```



Experimenting with several choices for even larger values of Δt (and therefore a) leads to the following results:

In[21]:=

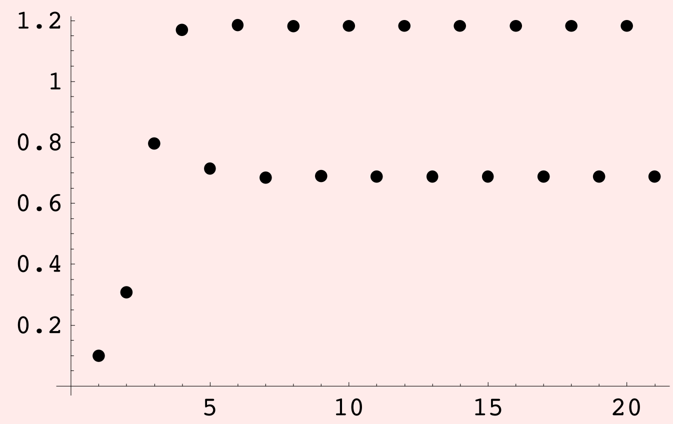
```
g[x_] := IterFunction[2.3][x];
```

In[22]:=

```
NestList[g, 0.1, 20]
ListPlot[%, PlotStyle -> PointSize[0.02]];
```

Out[22]=

{0.1, 0.307, 0.796327, 1.16936, 0.713852, 1.18367, 0.683646, 1.18108, 0.689186, 1.18187, 0.687501, 1.18164, 0.687982, 1.18171, 0.687842, 1.18169, 0.687883, 1.18169, 0.687871, 1.18169, 0.687874}



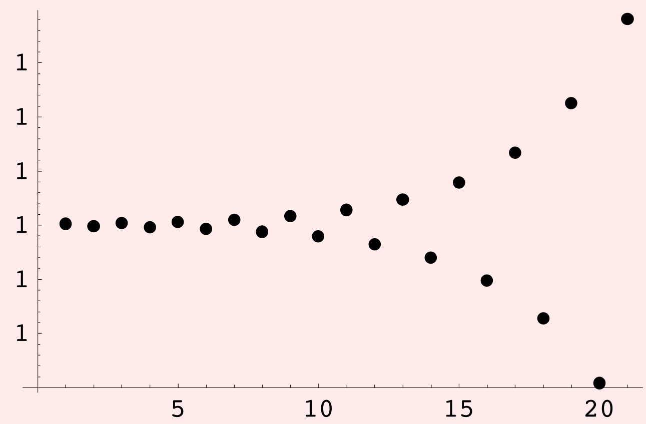
Let us start with a value of x0 which is close to the limit value 1; this leads to:

In[24]:=

```
NestList[g, 1.000000001, 20]
ListPlot[%, PlotStyle -> PointSize[0.02]];
```

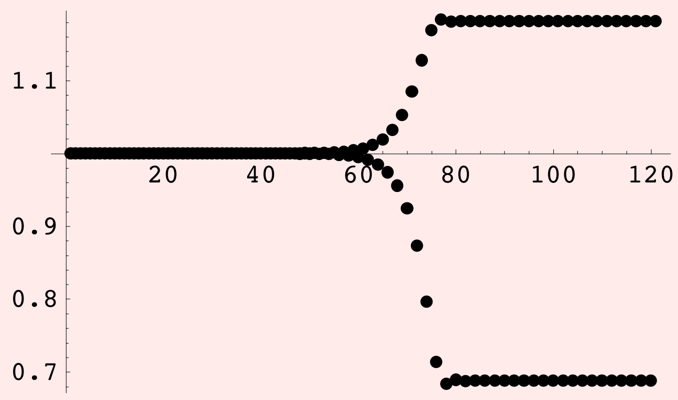
Out[24]=

{1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.}



In[26]:=

```
NestList[g, 1.000000001, 120];
ListPlot[%, PlotStyle -> PointSize[0.02]];
```

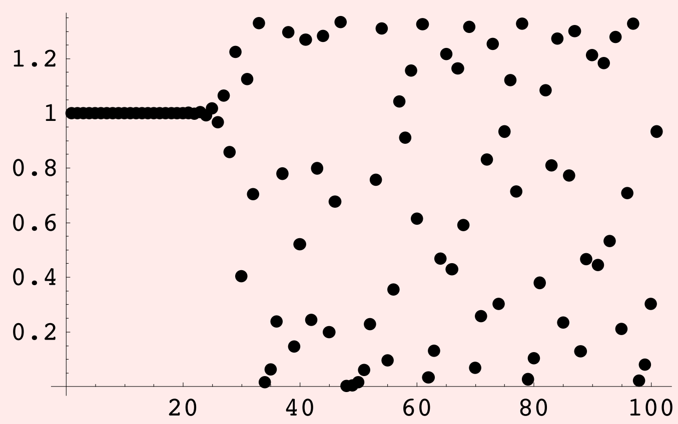


In[28]:=

```
g[x_] := IterFunction[3][x];
```

In[29]:=

```
NestList[g, 1.000000001, 100];
ListPlot[%, PlotStyle -> PointSize[0.02]];
```



Close all sections