

## Differential equation modelling

S J Eglén

28 October 2010

### 1st order: Euler

$$\frac{dy}{dx} = f(x, y)$$

For a fixed time-step  $h$  we Taylor expand to first order.

$$y(x + h) = y(x) + h \frac{dy}{dx} + h.o.t.$$

$$y(x + h) = y(x) + hf(x, y)$$

Making time-steps too small could cause accumulation of round-off error (Strogatz).

### Differential equation modelling

#### Phase Plane analysis

### 2nd order: midpoint method

Taylor expand to 2nd order; avoid calculating  $f'()$  by chain rule, and equate coefficients.

$$k_0 = hf(x, y(x))$$

$$k_1 = hf(x + h/2, y(x) + \frac{1}{2}k_0)$$

$$y(x + h) = y(x) + k_1$$

- ▶ More evaluations, but better accuracy than Euler.
- ▶ Make a guess at an initial step
- ▶ Or take principled approach to get family of solvers for different coefficients (4 unknown with 3 equations)

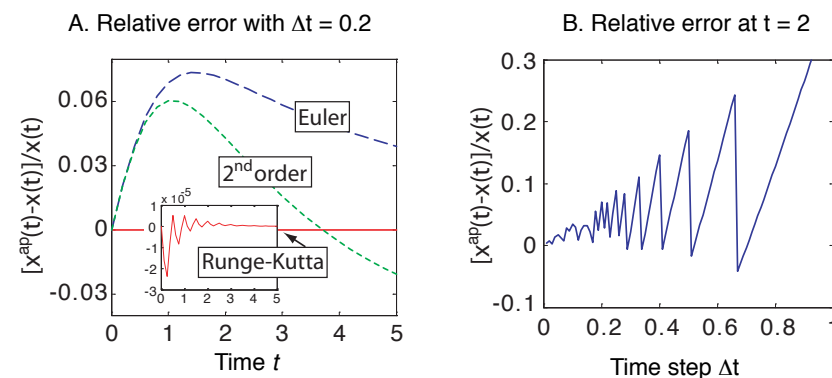
## 4th order Runge-Kutta formula (fixed step size)

$$\begin{aligned}k_0 &= hf(x, y(x)) \\k_1 &= hf(x + h/2, y(x) + \frac{1}{2}k_0) \\k_2 &= hf(x + h/2, y(x) + \frac{1}{2}k_1) \\k_3 &= hf(x + h, y(x) + k_2) \\y(x + h) &= y(x) + \frac{k_0}{6} + \frac{k_1}{3} + \frac{k_2}{3} + \frac{k_3}{6}\end{aligned}$$

- ▶  $f()$  may be expensive to compute.
- ▶ good trade off: computations performed vs accuracy

## Comparing solutions on problem with known solution

$$\frac{dx}{dt} = t - x + 1 \quad x(0) = 1$$



<http://web.cs.dal.ca/~tt/fundamentals/figures/Appendices/>

## Adaptive step size solvers

- ▶ Compare accuracy of one big jump ( $x+2h$ ), with two jumps of size  $h$ .
- ▶ Cost: 11  $f()$  evaluations, compared with 8  $f()$  evaluations for two small jumps; e.g. figure 17.2.1 of NR.
- ▶ Adjust  $h$  to ensure that  $\Delta \equiv y_2 - y_1$  within tolerance.
- ▶ popular solver: ode45
- ▶ More efficient step-size methods now available.

tolerance on error. *rk45* popular.

- ▶ Step-size increases when solution smooth.
- ▶ **Stiff** systems are a problem. (One component changing rapidly comparing to others.)
- ▶ Use industrial-strength solvers, e.g. LSODA.

## Framework for integration

- ▶ Code up function that returns integrals at any time given params.
- ▶ See DMB guide for examples.

## Example: van der Pol oscillator

IVP (initial value problems) e.g. van der Pol oscillator:

$$y'' - \mu(1 - y^2)y' + y = 0$$

Reduce to 1st order system ("normal form"):

$$\begin{aligned}x &= y' \\ x' - \mu(1 - y^2)x + y &= 0\end{aligned}$$

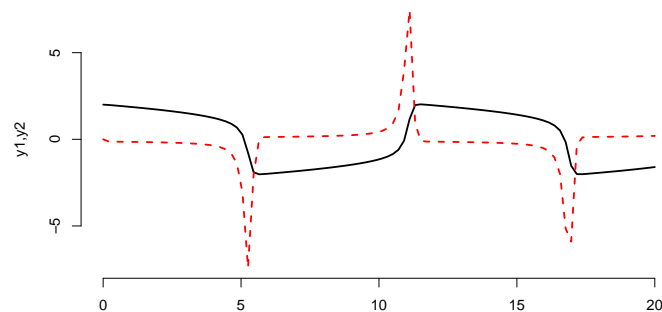
If we regard our state vector as  $\begin{pmatrix} y \\ x \end{pmatrix}$  then we have:

$$\begin{pmatrix} y' \\ x' \end{pmatrix} = \begin{pmatrix} x \\ \mu(1 - y^2)x - y \end{pmatrix}$$

## Output

```
> out[1:5,]
```

	time	1	2
[1,]	0.0000000	2.000000	0.0000000
[2,]	0.2020202	1.981466	-0.1279646
[3,]	0.4040404	1.954426	-0.1371399
[4,]	0.6060606	1.926339	-0.1408267
[5,]	0.8080808	1.897523	-0.1444858



## Calling the solver

```
library(deSolve)
vanderpol = function(t, y, parms) {
  ## T = current time
  ## Y = current state vector
  ## PARMS = vector of parameters for system.
  mu = parms[1]
  derivs = c( y[2], mu*(1-y[1]^2)*y[2] - y[1])
  ## Return
  list(derivs)
}

init.cond = c(2, 0);
times = seq(from=0, to=20, length=100)
parms = (mu=5.0)
out = lsoda(init.cond, times, vanderpol, parms)
pdf(file='vanderpol.pdf', width=8, height=5); par(bty='n')
matplot(out[,1], out[,-1], type='l', xlab='time',
        vlab='v1.v2'.lwd=2)
```

## Limitations

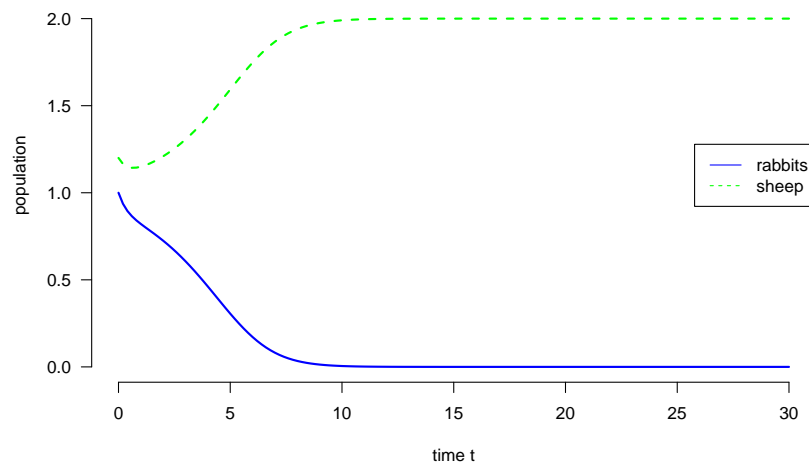
- ▶ How to add spikes into framework, if just returning derivative?
- ▶ or more generally, sudden changes to variables?
- ▶ how to debug?

## Outline

Differential equation modelling

Phase Plane analysis

1. Computing any trajectory over time, as before.



i.e. use numerical integration

Example: rabbits vs sheep (Strogatz, p155)

$$\frac{dr}{dt} = r(3 - r - 2s)$$

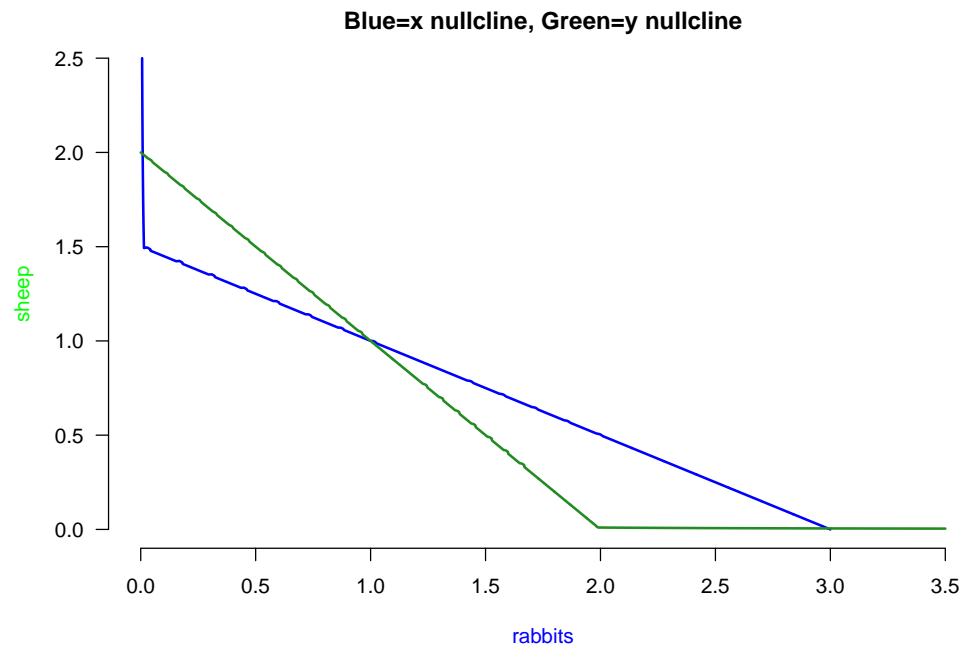
$$\frac{ds}{dt} = s(2 - r - s)$$

1. Compute trajectory over time
2. Plot phase planes
3. Nullclines
4. Examining stability

Finding nullclines and steady-states

- ▶ A nullcline for a variable  $x$  is where its DE  $\frac{dx}{dt}$  is zero.
- ▶ For a two-variable system  $x$  and  $y$ , steady-states are found where the two nullclines intersect.

Poor-man's phase space: check either side of nullclines.



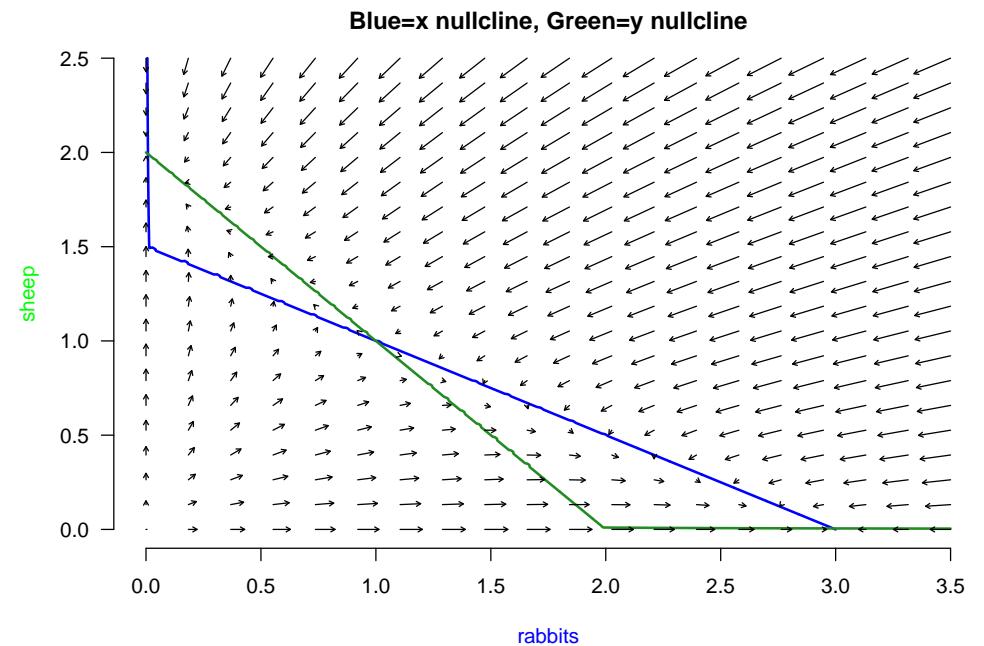
Steady-states: intersection of nullclines

Careful: not all nullclines are drawn on the previous page. This is an implementation issue with `contour()` for finding zeros.

► Find the steady-states

- 1.
- 2.
- 3.
- 4.

2. Phase-plane and 3. Nullclines



Classification of steady-states

Why take the Jacobian around the steady-state?

$$J(r, s) = \begin{pmatrix} 3 - 2r - 2s & -2r \\ -s & 2 - r - 2s \end{pmatrix}$$

Recall: for a 2x2 matrix:

- $\text{Tr}(J)$  = sum of eigenvalues
- $\text{Det}(J)$  = product of eigenvalues.

## Starting points

- ▶ deSolve package
- ▶ phase planes and nullclines (DMBpplane.r from DMB site, modified from Daniel Kaplan)
- ▶ integrate() – quadrature
- ▶ D() – symbolic differentiation
- ▶ optimize() (1d) and optim() (n-d)
- ▶ Steven Strogatz. Nonlinear dynamics and chaos.
- ▶ NR: William Press et al. Numerical Recipes in C/C++