Numerical Solutions of Differential Equations (1)

- Euler method
- Sources of error
  - Truncation
  - Round-off error
- Error balance in numerical integration

Euler Method

- Many differential equations don't have an exact solution or it is very complicated, so finding exact solutions is difficult
- Often useful to be able to solve differential equations numerically
- Idea:
  - remember how we derived the differential equation for exponential growth from the discrete Malthusian model (where we took the limit $\Delta t \to 0$ of a difference equation)
  - In numerical schemes we find difference equations to approximate derivatives ...

Euler Method (2)

- Consider the general initial value problem

$$dy/dt = f(y,t),\ y(0) = y_0$$

- Recall from definition of derivative:

$$dy/dt = \lim_{\Delta t \to 0} \frac{y(t+\Delta t) - y(t)}{\Delta t}$$

- An approximation of the derivative is given by

$$dy/dt \approx \frac{y(t+\Delta t) - y(t)}{\Delta t}$$

(for sufficiently “small” Δt)

Euler Method (3)

- Putting all together ...

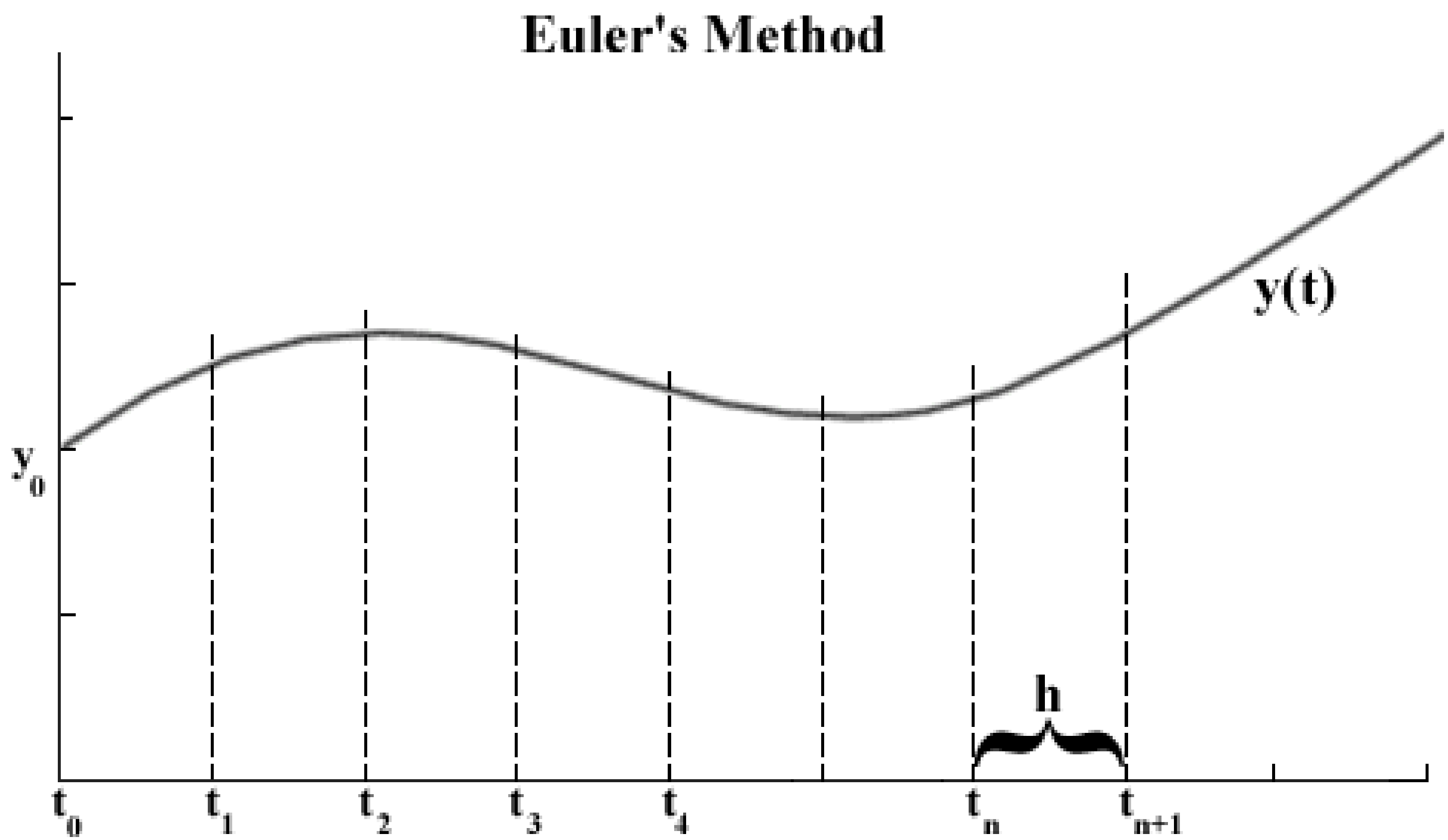
$$f(y,t) = dy/dt \approx \frac{y(t+\Delta t) - y(t)}{\Delta t}$$

Differential equation

Approximation of derivative

- We find: $y(t+\Delta t) \approx y(t) + \Delta t\, f(y,t)$
- If we discretise an interval of time T into N+1 little intervals of length h:

Euler Method (4)



$$\frac{dy}{dt} = f(t,y)$$

$$y(t_0) = y_0$$

y(t) is the solution of this differential equation

This is Euler's formula to approximate the solutions.

$$\boxed{y_{n+1} = y_n + hf(t_n, y_n)}$$

Example: Malthusian Growth

- Remember equation for Malthusian growth

$$dP/dt = rP, P(0) = P_0$$

has the solution:

$$P(t) = P_0 \exp(rt)$$

- What if we apply Euler's method?

$$P_{n+1} = P_n + hrP_n \qquad t = hn, n = 0,1,\ldots$$
$$P_0 = P(0)$$

- Can rewrite this as:

$$P_{n+1} = (1+hr)P_n = (1+hr)^2 P_{n-1} = \ldots = (1+hr)^{n+1} P_0$$

Euler Scheme for Malthusian Growth

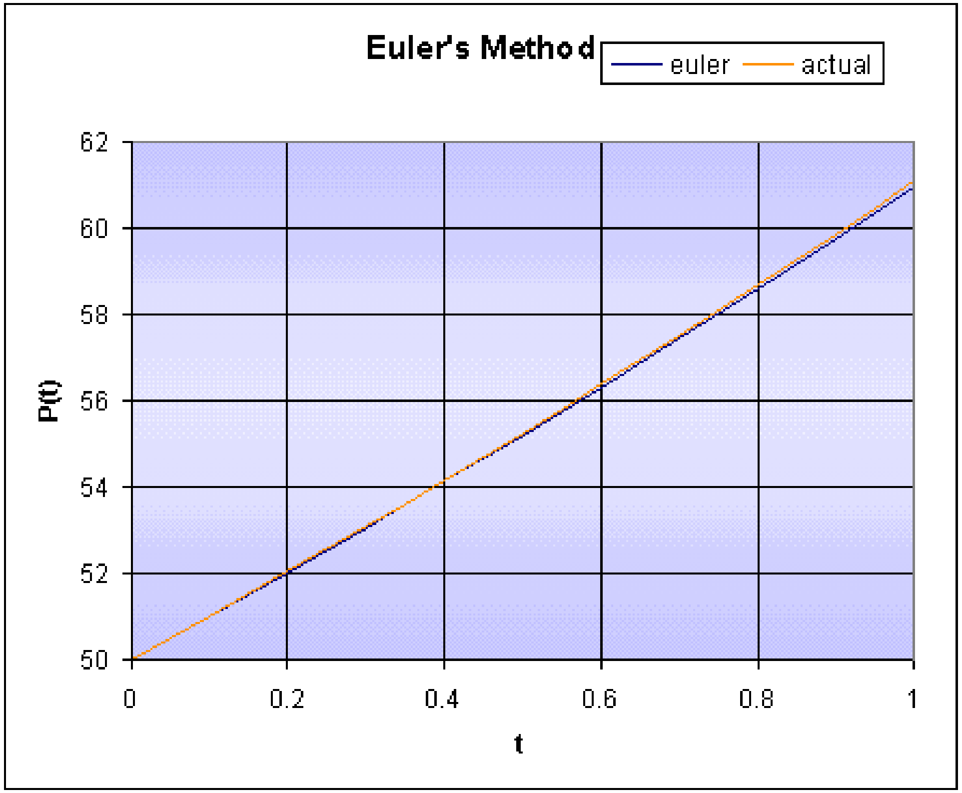
- Normally the discrete scheme is solved via computer simulation
- Here we find:

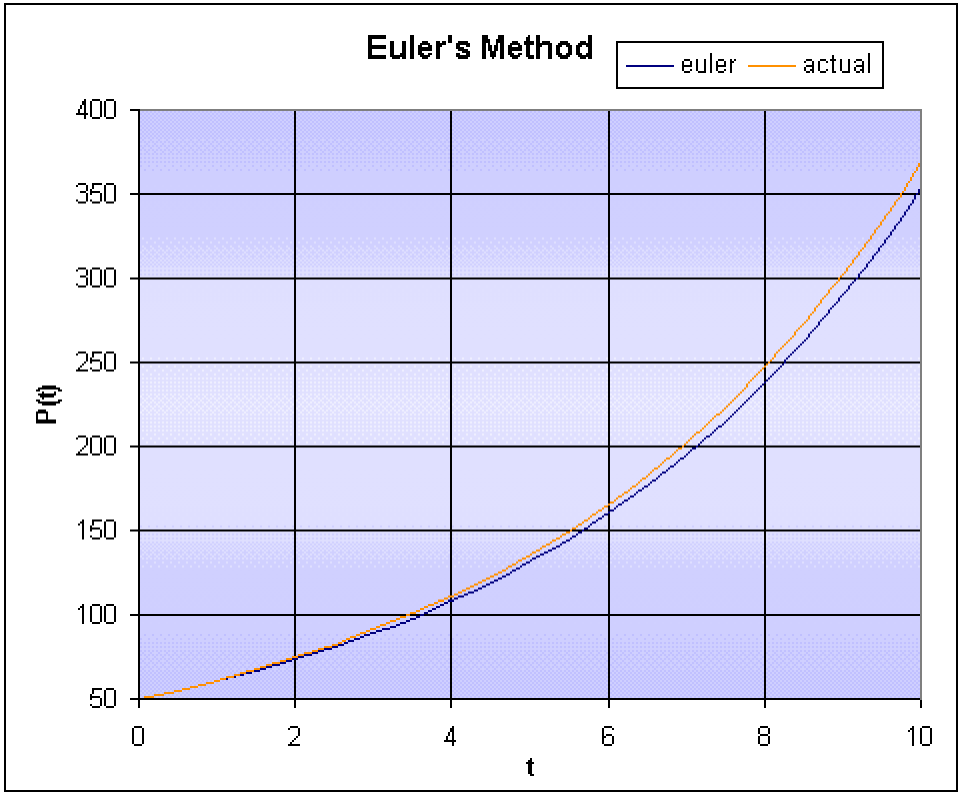
$$P_n = (1 + hr)^n P_0$$

$$P^{Euler}(t) = (1 + hr)^{t/h} P_0$$

- Compared to: $P(t) = P_0 e^{rt}$
  - I.e. the numerical scheme only becomes exact in the limit of infinitely small step size (show it!)
  - For any finite step size there are deviations, the larger, the larger the step size ...

Euler Scheme for Malthusian Growth





Comparison Euler Method—analytical result for

$$P' = 0.2\,P, P_0 = 50, h = 0.1$$

Sample C code for an Euler Scheme

```
#include<stdio.h>
float fun(float x,float y)
{  float f;
  f=x+y;
return f;}
main() {
 float a,b,x,y,h,t,k;
printf("\nEnter x0,y0,h,xn: ");    scanf("%f%f%f%f",&a,&b,&h,&t,);
x=a;    y=b;    printf("\n  x\t  y\n");
while(x<=t)    {
        k=h*fun(x,y);        y=y+k;        x=x+h;
        printf("%0.3f\t%0.3f\n",x,y);    }
}
}
```

Which diff. eq. does this code integrate?

$$\frac{d}{dt}x = ?$$

Truncation Error of Euler Scheme

- Local error
  - We approximate a derivative by the differential quotient, this is not exact

$$y(t+h) \approx y(t) + h\,y'$$

  - Error estimate from Taylor series

$$y(t+h) = y(t) + h\,y' + \frac{h^2}{2!}\,y'' + \frac{h^3}{3!}\,y''' + \dots$$

  - i.e. per iteration step the local error is proportional to h^2

Truncation Error of Euler Scheme (2)

- Global error:
  - If we integrate for a time t we need t/h steps
  - Per step we accumulate an error proportional to h^2
  - -> overall error is proportional to $h^2 \, t/h = t \, h$
- The Euler scheme is a so-called **first order scheme**, which implies
  - Local error scales prop. to step size squared
  - Global error is linear in step size
- Accuracy can be improved by decreasing step length h

Errors in Numerical Calculations

- Truncation error:
  - results from an approximation of an exact mathematical procedure
  - E.g.:

$$(1+x)^{\alpha} \approx 1+\alpha x+\frac{\alpha(\alpha-1)}{2!}x^2+\frac{\alpha(\alpha-1)(\alpha-2)}{3!}x^3+\cancel{rest}$$

- Round-off error:
  - Results from having numbers with limited significant digits representing exact numbers

Round-off Errors

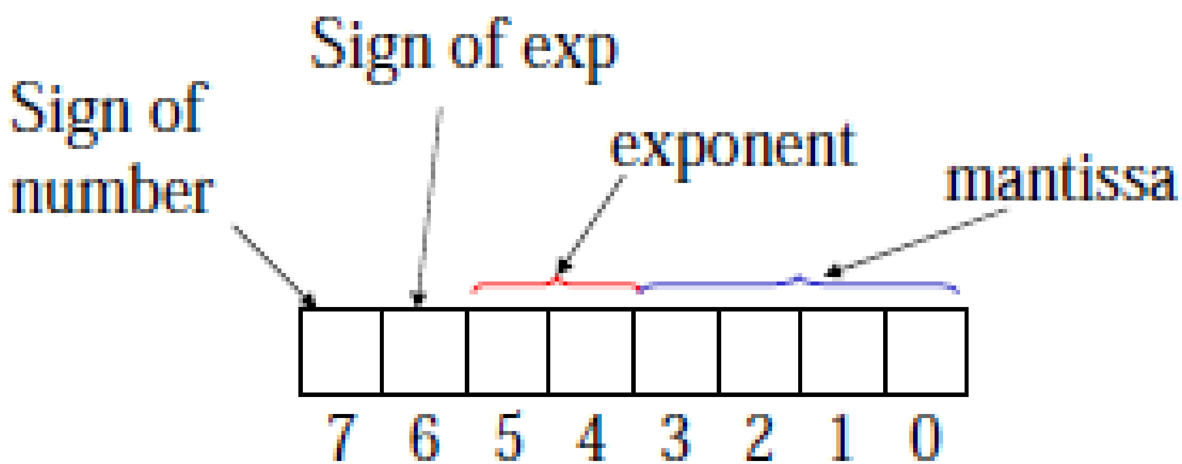
- Floating point representation

exponent

$$N = m\, b^{e}$$

Base of number system (10)

Mantissa
fractional part



Round-off Errors (2)

- Example: floating point representation of

  1/27=0.037037037037037...

- Using 4 digits this could be stored as

  $0.0370*10^{0}$

- Better: “Normalizing” (i.e. mantissa is limited to 1/b<m<1)

  $0.3703*10^{-1}$

- But still ... we lose accuracy! -> round-off errors

Round-off Errors (3)

- Floating point representation allows to handle very large and very small numbers ...

  ... but ...

  - More storage required than for integers
  - Longer processing time
  - Round-off error is introduced since the mantissa holds a finite number of digits
  - Round-off error increases with x, e.g. 4 digit mantissa
    - $0.3516 * 10^4 \rightarrow \Delta x = 1$
    - $0.3516 * 10^0 \rightarrow \Delta x = 0.0001$

Arithmetic Manipulation Errors: +

- Consider computer with 4 digit mantissa
- Add 2.365 and 0.01234

0.2365*10^1 → 0.2365*10^1

+0.1234*10^-1 match exponents 0.001234*10^1

0.237734*10^1

Chop to fit floating point representation ↓

0.2377*10^1

→ Last two digits have been lost!
Relative error proportional to magnitude

Arithmetic Manipulation Errors (2)

- Also matter when
  - Adding large and small numbers
  - When subtracting nearly equal numbers
  - When performing a large number of arithmetic manipulations
- Can be minimized by using extended precision (at the cost of run time)
- Total error = Truncation error + Round-off errors

Round-off Errors in Euler Scheme

- Assume machine precision is ε
- In step n of Euler scheme rounding off error is ϵy_n
- In N steps roughly $error \sim N \epsilon y_0$ (if all round-off errors are of the same sign)
- More realistically, round-off errors are independent, and thus

$$round-off\ error \sim \sqrt{N} \epsilon y_0$$

- Techniques to reduce round-off error, e.g. compensated summation -> e.g. Kahan summation

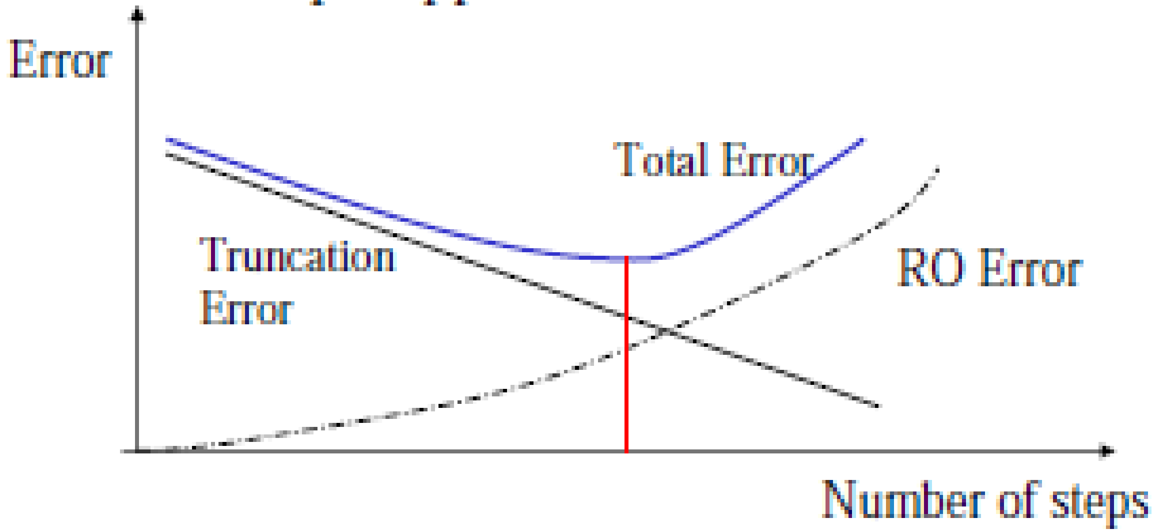
Error Balance for Euler Scheme

2 Sources of Error:

1. Truncation – Taylor Series
2. Round-Off – significant Digits

Truncation Error: 2 parts

1. Local – method application over 1 step
2. Global – accumulated additive error over multiple applications



Just using smaller step lengths h is not enough!

Summary

- Important points to remember:
  - Idea of the Euler scheme
  - Order of Euler scheme
  - Various sources of numerical error –
    - Truncation error
    - Round-off error
  - Trade-offs of errors in numerical integration