Overview

More on functions

Function handles

Quadrature

Ordinary differential equation solvers

Bisection

Function Functions

Peter Hertel

University of Osnabrück, Germany

Lecture presented at APS, Nankai University, China

`http://www.home.uni-osnabrueck.de/phertel`

Spring 2012

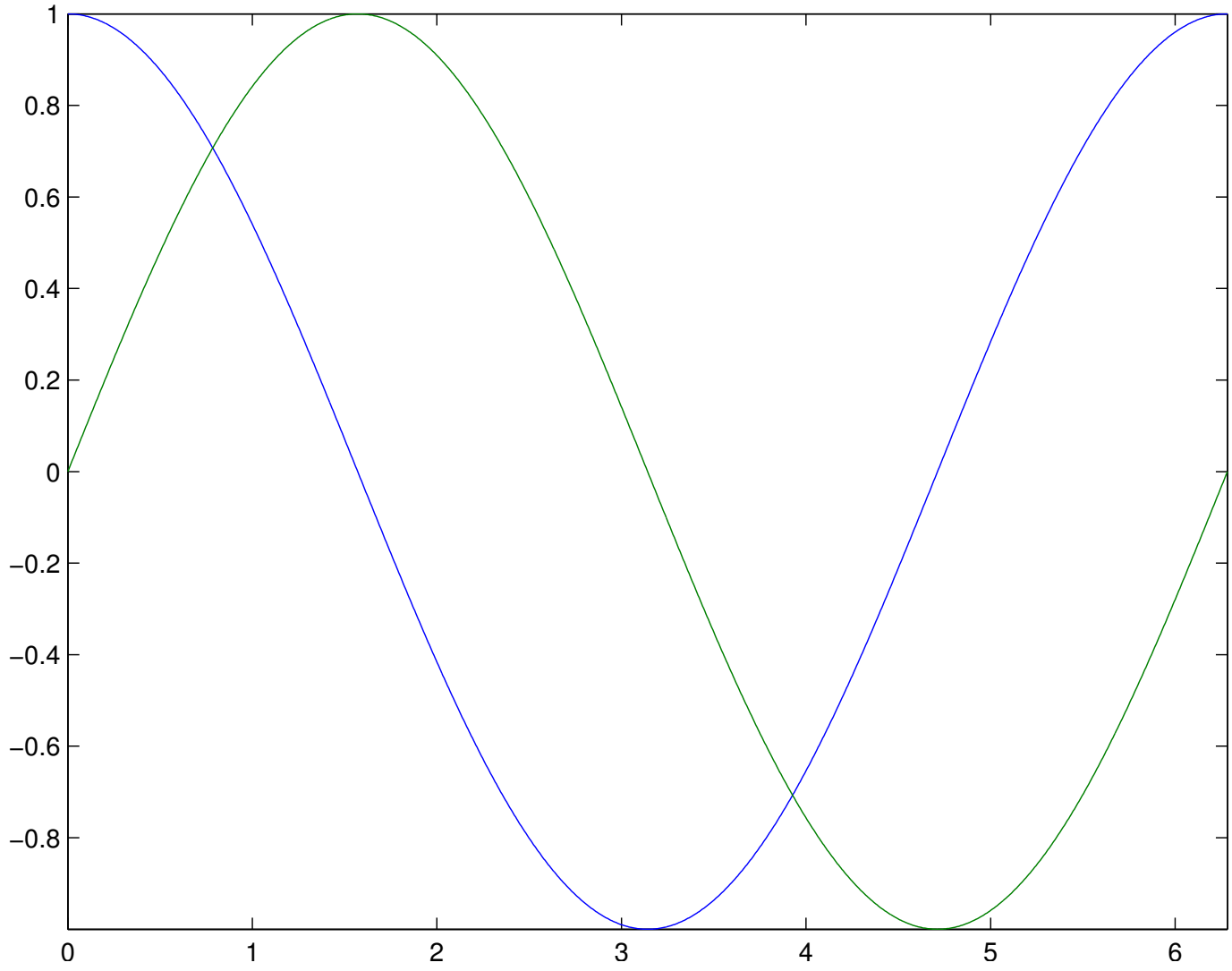
Overview

- Structure your program into functions
- Scripts are unnecessary and should be avoided
- more on functions
- functions are objects
- they may be arguments of functions !
- example: integral
- more on function functions, or functionals
- example: the bisection algorithm

Input and output values

- In mathematical language, a function $y = f(x)$ maps a real valued variable x into a real valued variable y
- In MATLAB the variable x has to be represented by a finite number of representative values
- such as `x=linspace(0,2*pi,1024);` which creates a row vector with 1024 equally spaced values in $[0, 2\pi]$. Endpoints are included.
- `y=sin(x);` returns the sine values of the representative values.
- MATLAB functions may return more than one output value
- they must be grouped by `[...]`
- such as in `[c,s]=cossin(x);`
- MATLAB functions may depend on no, one, or more arguments.
- MATLAB functions may return no, one, or more results.

```
function [c,s]=cossin(x)
c=cos(x);
s=sin(x);
end % function cossin

>> x=linspace(0,2*pi,1024);
>> [a,b]=cossin(x);
>> plot(x,a,x,b);
>> axis tight
```



Our first plot: $\cos(x)$ and $\sin(x)$ vs. x.

Function handles

- A function `fun` is a piece of software.
- Either defined in an `.m` file,
- or built-in.
- It resides somewhere in RAM
- The RAM address of the function is denoted by `@fun`
- `@` means 'located at' as is well known from Email addresses
- It is not the object itself, but a handle how to access the object
- for example, `s=@sin` says where the sine function is located in memory.
- `y=sin(x)` and `y=s(x)` return the same result, namely the sine of `x`.
- Functions can be variables themselves.

Anonymous functions

- you may declare functions by their handle only
- `fh=@(x) 1./(1+(a*x).^2)`
- the current value of a is used
- note the dots
- the handle has a name, here `fh`
- the function itself has no name (anonymous)

Functionals

- A functional is a function which depends on a function
- e.g. the integral

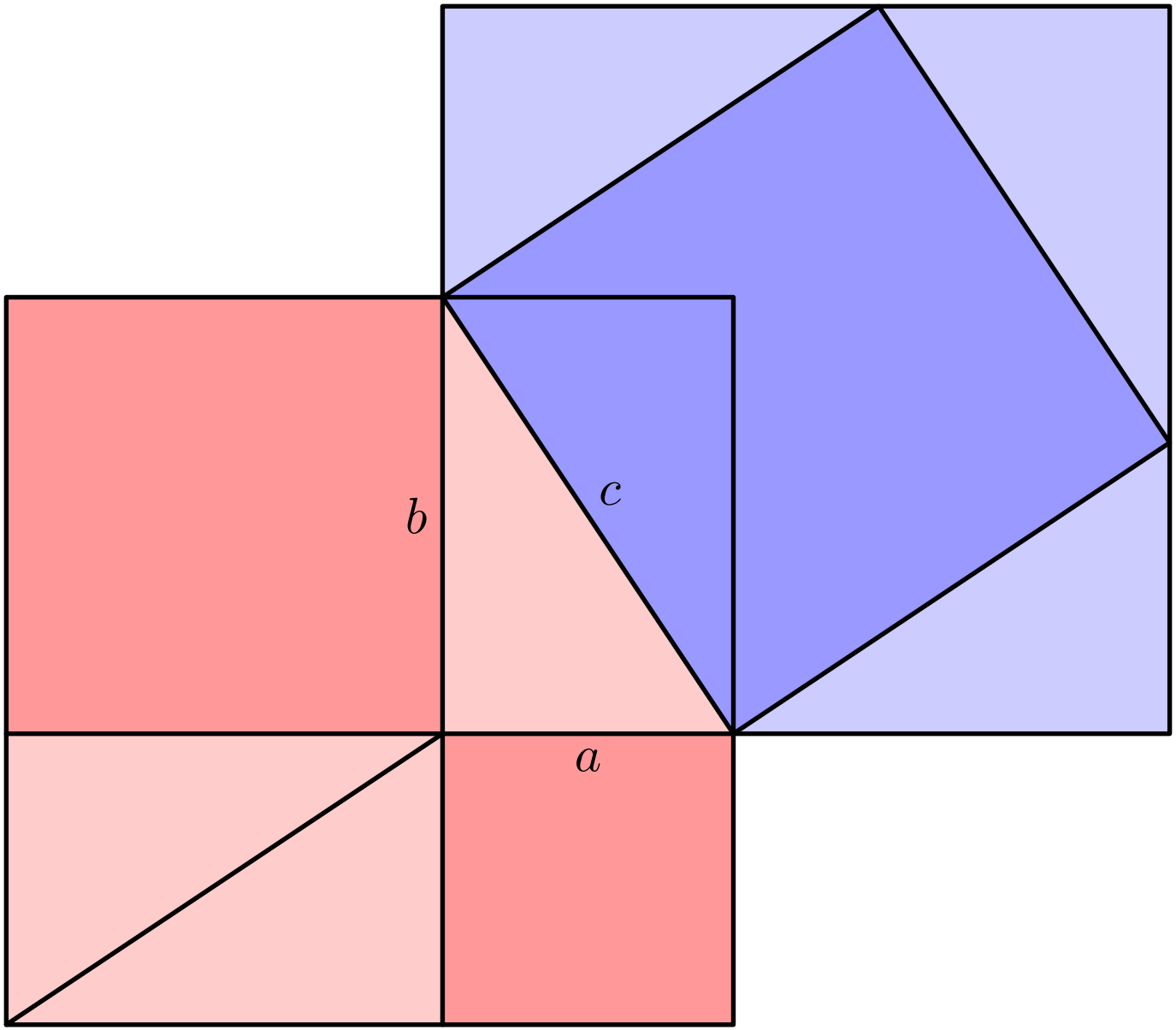
$$\mathrm{quad}(f,a,b) = \int_a^b \mathrm{d}x\, f(x)$$

- quadrature (of the circle)
- in MATLAB: `q=quad(f,a,b)`
- `f` must be a function handle
- `@sin, @my_function, or anonymous`
- `>> f=@(x) sqrt(1-x.^2);`
- `>> pa=4*quad(f,0,1)`
- `pa = 3.1416`

Squaring the circle

- About 1500 BC the Chinese used the modern decimals system
- About 450 BC, the Chinese mathematician and astronomer Zu Chongzhi approximated π by 355/113
- This is correct for first seven digits
- About the same time, in ancient Greece, mathematicians (philosophers) tried to find an exact expression
- Quadrature: reshape a figure so that it has the same area until it becomes a square , or a quadrate
- The Greek mathematician never reached their goal because this is not possible in a finite number of steps
- Only modern mathematics introduced infinite series and their limit values
- The integral is such a limit.

Thabit ibn Qurra lived (arond 800 AD) and taught in Baghdad, today Irak. He invented a very elegant proof of Phythagoras's theorem.



By moving equally sized triangles, Thabit proved $a^2 + b^2 = c^2$.

Accuracy

- A computer has only a finite number of states and a finite time to perform a calculation
- e.g. numbers like $\sqrt{2}$ or π are never exact
- Since a computer uses the binary number system, even $1/3$ will be approximated
- therefore the quad function can calculate the integral with a finite accuracy only
- default is five digits
- this can be improved
- `pa=4*quad(f,0,1,1e-12)` is accurate for 11 digits
- note that quad can be called with three and with four arguments

ode45

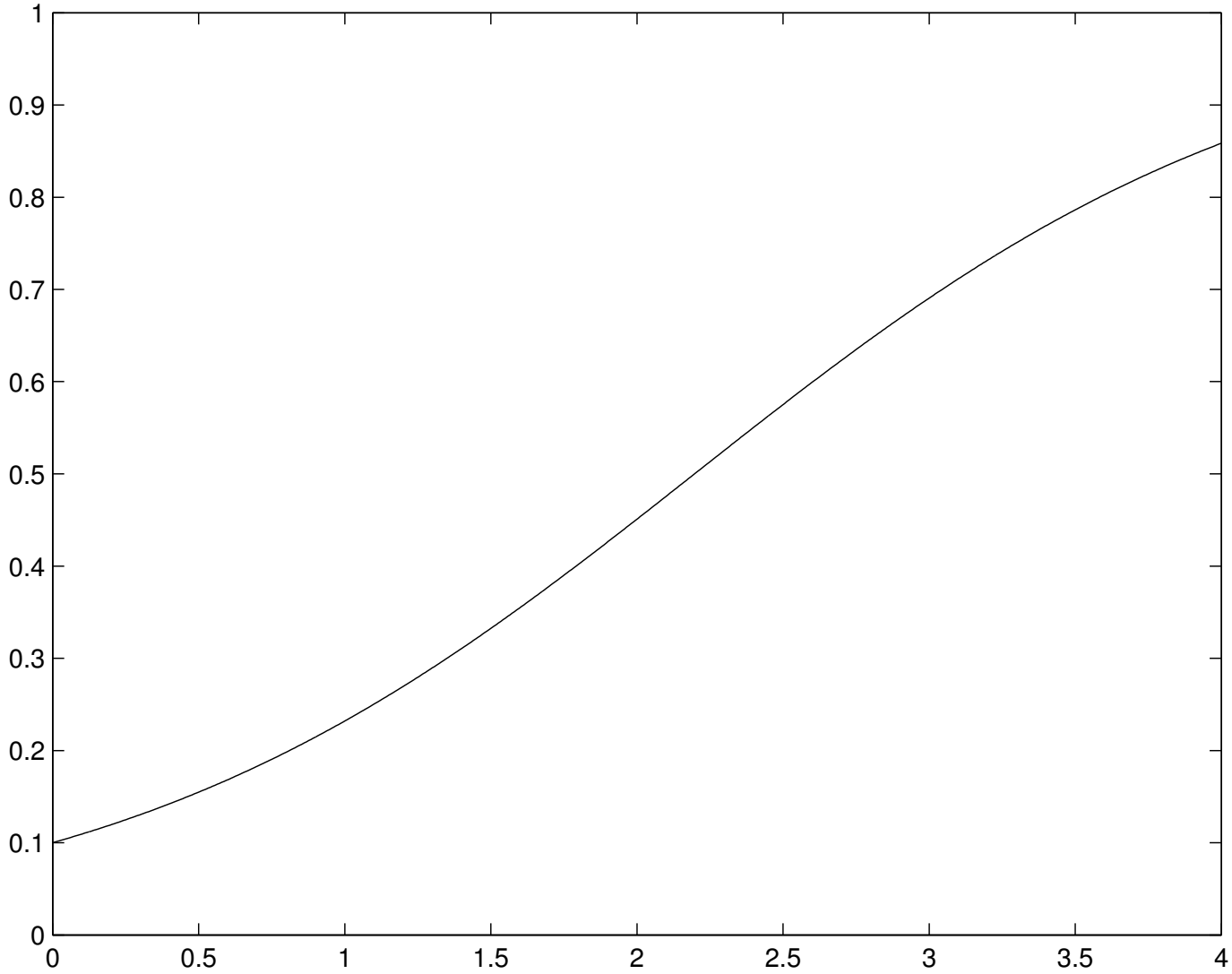
- solve the differential equation

  $\dot{y} = f(t, y)$
- first, specify the differential equation by a function

  $f = f(t, y)$
- specify the time span `tspn`
- specify the initial condition y_0
- `[tout,yout]=ode45(@f,tspn,y0)` does the job
- example: standard logistic curve

  $\dot{y} = y(1 - y)$
- currently, the market share is 10%

```
% calculate the standard logistic curve
% for a given initial condition

% setup the differential equation
ydot=@(t,y) y.*(1-y);

% time span
ts=linspace(0,4,512);

% initial condition
y0=0.10;

% solve it by ode45
[t,y]=ode45(ydot,ts,y0);

% plot it
plot(t,y);
axis([0,4,0,1]);

% export it
print -deps slc.eps
```



The standard logistic curve as solved numerically by `ode45`.

Bisection algorithm

- Let f be a continuous function and $[x_1, x_2]$ an interval. If $f(x_1)$ and $f(x_2)$ have different sign, then there is an $x \in [x_1, x_2]$ such that $f(x) = 0$.
- In MATLAB, `f` is a function handle.
- `>> pa=2*bisection(@cos,0,2,1e-12);`
- `>> pa-pi`
- `    5.7909e-13`

```
function x=bisection(f,x1,x2,tol)
maxit=50;
f1=f(x1);
f2=f(x2);
if f1*f2>0
  error('bad initial interval');
end;
it=1;
while (abs(x2-x1)>abs(tol)) && (it<maxit)
  it=it+1;
  xc=(x1+x2)/2;
  fc=f(xc);
  if f1*fc<0
    x2=xc;
    f2=fc;
  else
    x1=xc;
    f1=fc;
  end;
end
x=(x1+x2)/2;
end % bisection
```