Function handles and additional parameters

Function handle example:

```
clear
f=@(x) p*x
```

```
f =
    @(x)p*x
```

```
%{
Undefined function or variable 'p'.
Error in @(x)p*x
%}
```

```
p=1;f=@(x) p*x
```

```
f =
    @(x)p*x
```

```
f(1)     % ans = 1
```

```
ans = 1
```

Change parameter p

```
p=2;
f(1)     % ans = 1
```

```
ans = 1
```

```
% No affect on f.
```

Change p, redifine f

```
p=2;f=@(x) p*x
```

```
f =
    @(x)p*x
```

```
f(1)     % ans = 2
```

```
ans = 2
```

```
% Conclusion:
```

```
% Function f uses the value p had on definion time of f.
```

Computer algebra example: Maple has two modes of definitions:

```
%{
> p:=1, f:=x -> p*x;          (Matlab: f=@(x) p*x)
                        x->p*x
> p:=2; f(x)
                        2*x
%}
% Maple's x-> value_at(x) definition works "run-time, calling time", Matlab's doesn't.
%
```

The second Maple-mode, unapply, works in the Matlab-way.

```
%{
p:=2
g := unapply(p*x, x)
                        x-> 2*x
%}
% Works at definition time, puts the value of p into the code.
%
% Matlab doesn't write the value of p into the code, but uses the
% value p had when f was defined, just like Maple's unapply.
%
```

Example: Finite difference derivative

```
type fd_deriv
```

```
function y = fd_deriv(f,x,h)
% Finite difference derivative of f
if nargin < 3, h=sqrt(eps); end;
y=(f(x+h)-f(x))/h;
end
```

Let f be Newton iteration function for sqrt:

```
type sqrtiter
```

```
function [x,niter] = sqrtiter(a,tol)
%
if nargin < 2, tol=eps, end;
x=a;
niter=0;
xreldiff=inf;

while xreldiff > tol
    niter=niter+1;
    xold=x;
    x=(x+a/x)/2;
    xreldiff=abs(x-xold)/abs(x);
    if niter > 50
        error('Did not converge after 50 iterations')
    end
```

```
end
```

```
%
% sqrtiter has a tol-argument, but fd_deriv takes a function with only 1
% input argument.
%
% The most elegant way is to use an anonymous function, i.e. include
% the function definition on the command line:
%
```

```
fder=fd_deriv(@(x) sqrtiter(x,0.0001),2)
```

```
fder = 0.3536
```

Check with symbolic:

```
syms x
dsqrt=diff(sqrt(x),x)
```

```
dsqrt =
```

$$\frac{1}{2\sqrt{x}}$$

```
pretty(dsqrt)
```

```
    1
---------
2 sqrt(x)
```

```
latex(dsqrt)
```

```
ans = \frac{1}{2\, \sqrt{x}}
```

$$\frac{1}{2\sqrt{x}}$$

```
d_at_2=subs(dsqrt,x,2.0)
```

```
d_at_2 =
```

$$\frac{\sqrt{2}}{4}$$

```
eval(d_at_2)
```

```
ans = 0.3536
```

```
fder
```

```
fder = 0.3536
```

Example: ode45