# Numerical differentiation

# Announcement

- there is a script and some functions available for this lecture on the course web site

# Numerical differentiation

- numerical differentiation attempts to estimate the value of the derivative of a function without computing the analytic form of the derivative
- why would you want to do this?
  - the function is difficult or impossible to differentiate
  - the derivative is expensive to compute
  - you don't actually know the function that you are trying to differentiate

# Differentiating polynomials

- polynomials are often used in numerical calculations, and MATLAB provides a function **`polyder`** that computes the derivative of a polynomial
  - also computes the derivative of a product of polynomials or a quotient of polynomials

# Differentiating polynomials

```
poly1 = [3 6 9];               % 3x^2 + 6x + 9
d = polyder(poly1)
d =
     6     6

poly2 = [1 2 0];               % x^2 + 2x + 0
d = polyder(poly1, poly2)  % derivative of product
d =
     12    36    42    18
```

# Forward finite difference

- the mathematical definition of a derivative is

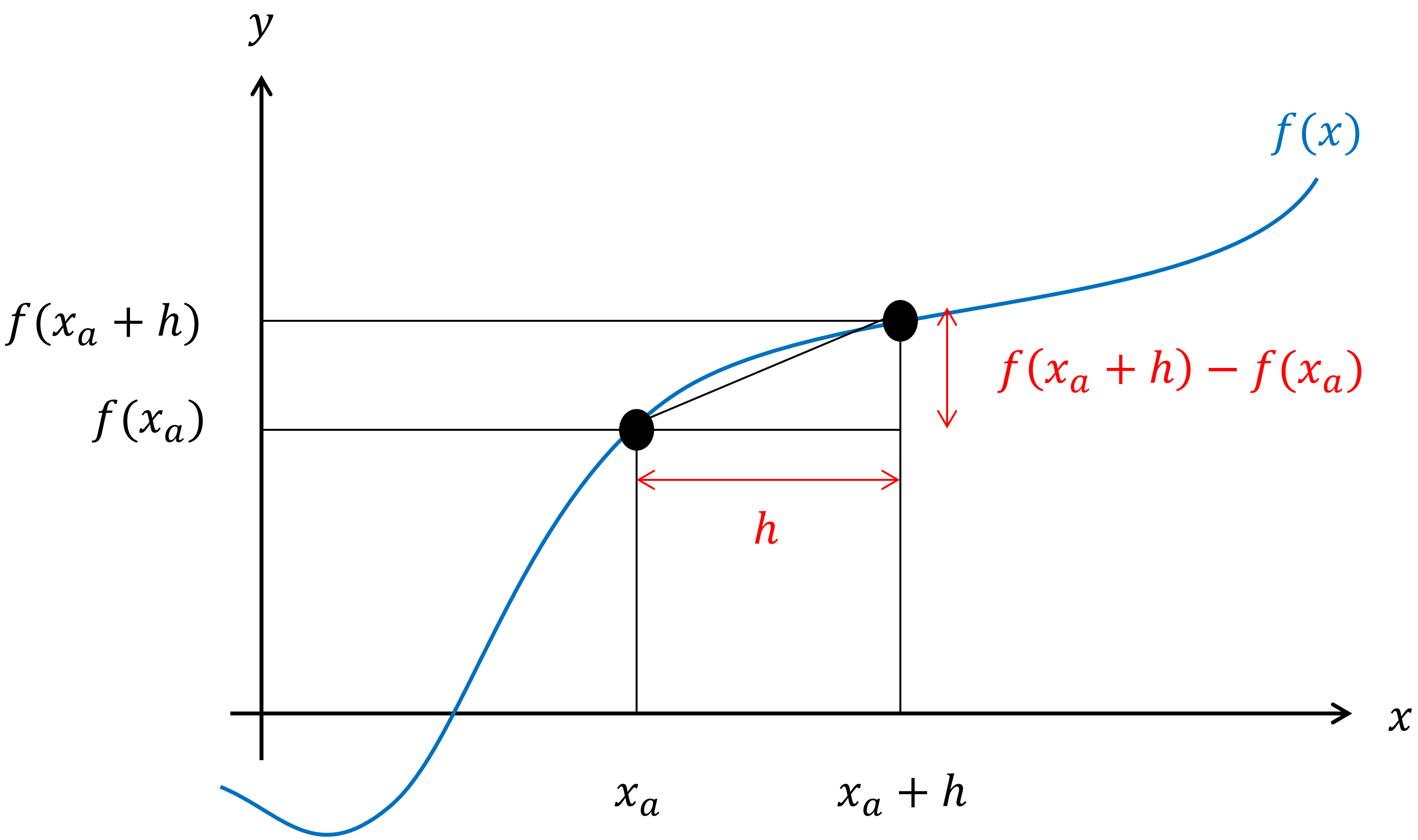
$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

which suggests the approximation

$$f_{\mathrm{f}}'(x) \approx \frac{f(x+h) - f(x)}{h}$$

- $f_{\mathrm{f}}'(x)$ is called the *forward finite difference* approximation of the function $f(x)$

# Forward finite difference



# Forward finite difference

```
function fprime = fdiff(f, x, h)
%FDIFF Forward finite difference derivative.
%   FPRIME = FDIFF(F, X, H) computes the derivative FPRIME of the
%   function F at all values in X using a step size of H.

fplus = f(x + h);
fx = f(x);
fprime = (fplus - fx) / h;
```

# Forward finite difference

- see **`day21.m`** for demonstration

# Backward finite difference

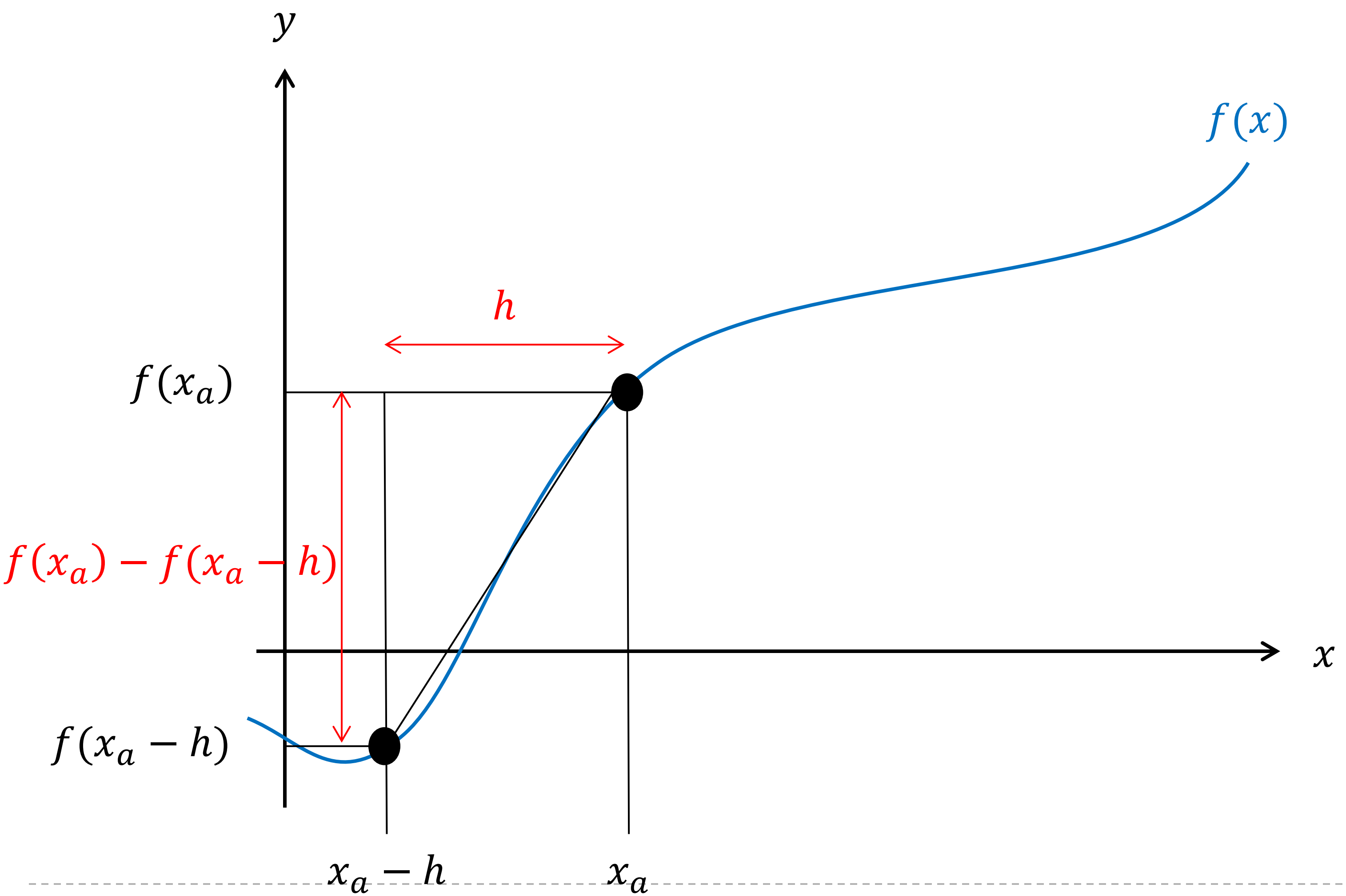
- if we choose a negative value for $h$ our approximation becomes

$$f_{\mathrm{b}}'(x) \approx \frac{f(x-h)-f(x)}{-h}$$

$$= \frac{f(x)-f(x-h)}{h}$$

- $f_{\mathrm{b}}'(x)$ is called the *backward finite difference* approximation of the function $f(x)$

# Backward finite difference



# Backward finite difference

```
function fprime = bdiff(f, x, h)
%BDIFF Backward finite difference derivative.
%   FPRIME = BDIFF(F, X, H) computes the derivative FPRIME of the
%   function F at all values in X using a step size of H.

fminus = f(x - h);
fx = f(x);
fprime = (fx - fminus) / h;
```

# Backward finite difference

- see **`day21.m`** for demonstration

# Central finite difference

- if we average the forward and backward difference approximations we get

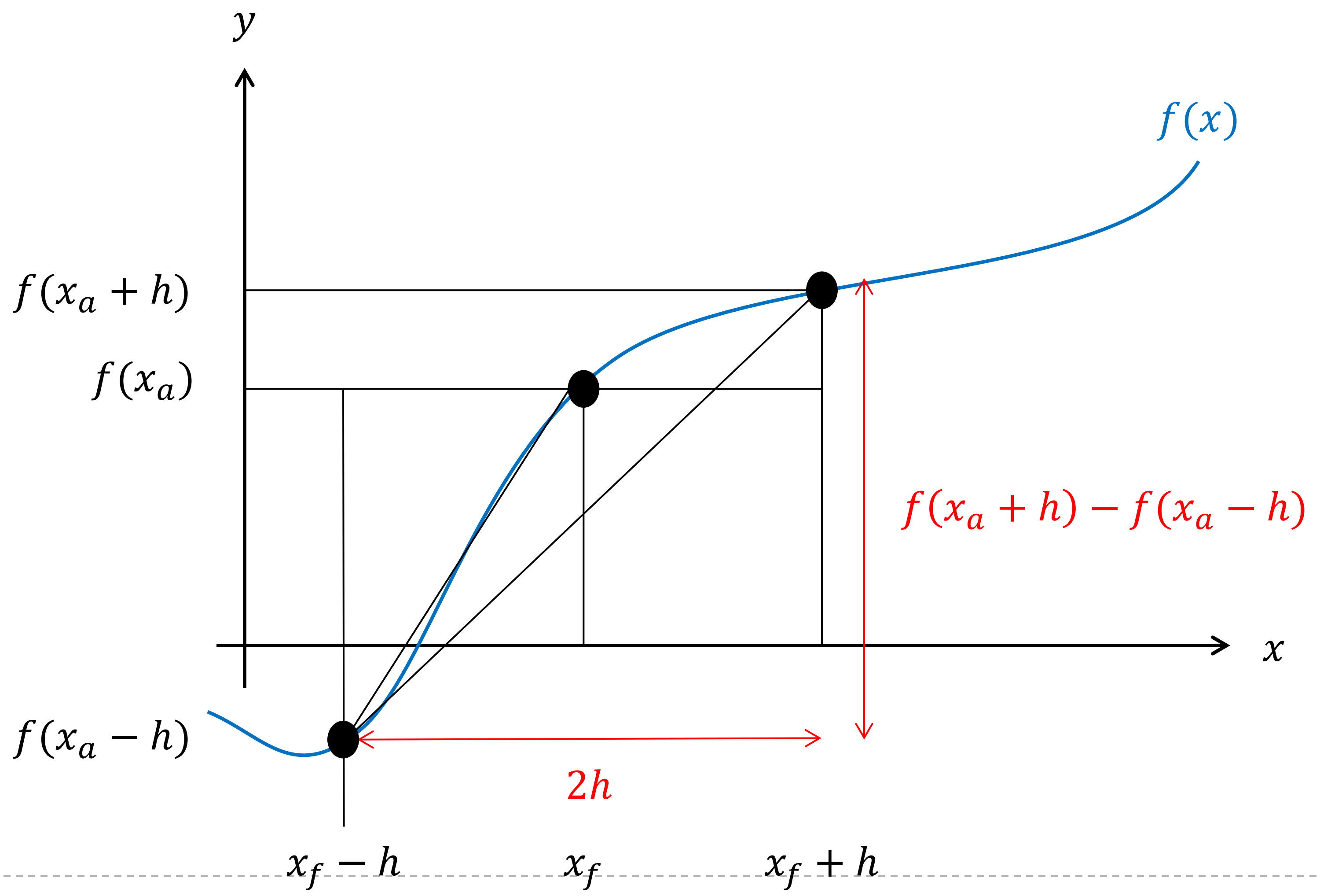
$$f_{\mathrm{c}}'(x) \approx \frac{f_{\mathrm{f}}'(x) + f_{\mathrm{b}}'(x)}{2}$$

$$= \frac{f(x+h) - f(x)}{2h} + \frac{f(x) - f(x-h)}{2h}$$

$$= \frac{f(x+h) - f(x-h)}{2h}$$

- $f_{\mathrm{c}}'(x)$ is called the *central finite difference* approximation of the function $f(x)$

# Forward finite difference



# Central finite difference

```
function fprime = cdiff(f, x, h)
%CDIFF Central finite difference derivative.
%   FPRIME = CDIFF(F, X, H) computes the derivative FPRIME of the
%   function F at all values in X using a step size of H.

fminus = f(x - h);
fplus = f(x + h);
fprime = (fplus - fminus) / (2 * h);
```

# Central finite difference

- see **`day21.m`** for demonstration

# The effect of noise

- a significant problem with computing derivatives using finite differences is that finite differences are very sensitive to noise
  - e.g., consider a sine function with a small amount of additive Gaussian noise

```
function y = noisysin(x)

y = sin(x) + randn(size(x))*0.001;

end
```

- see **`day21.m`** for demonstration

# Higher order derivatives

- higher order derivatives can be obtained by computing the finite difference of a finite difference
  - e.g., the second-order forward finite difference

$$f_{\mathrm{f}}'' \approx \frac{\frac{f(x+2h)-f(x+h)}{h} - \frac{f(x+h)-f(x)}{h}}{h}$$

$$= \frac{f(x+2h)-2f(x+h)-f(x)}{h^2}$$