Lecture 9 Review

Poisson and Gaussian probability distributions.

Pick random numbers from arbitrary probability distribution.

First peek at fitting data.

Numerical Derivatives

We need to know to to take a derivative df(x)/dx of a function f(x) at x.

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

(from calculus land)

Taylor series expand f(x + h). Recall that h << 1:

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2} f''(x) + \ldots$$

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2} f''(\zeta)$$

(exact, also from calculus)

ζ unknown !! ($x \leq \zeta \leq x + h$)

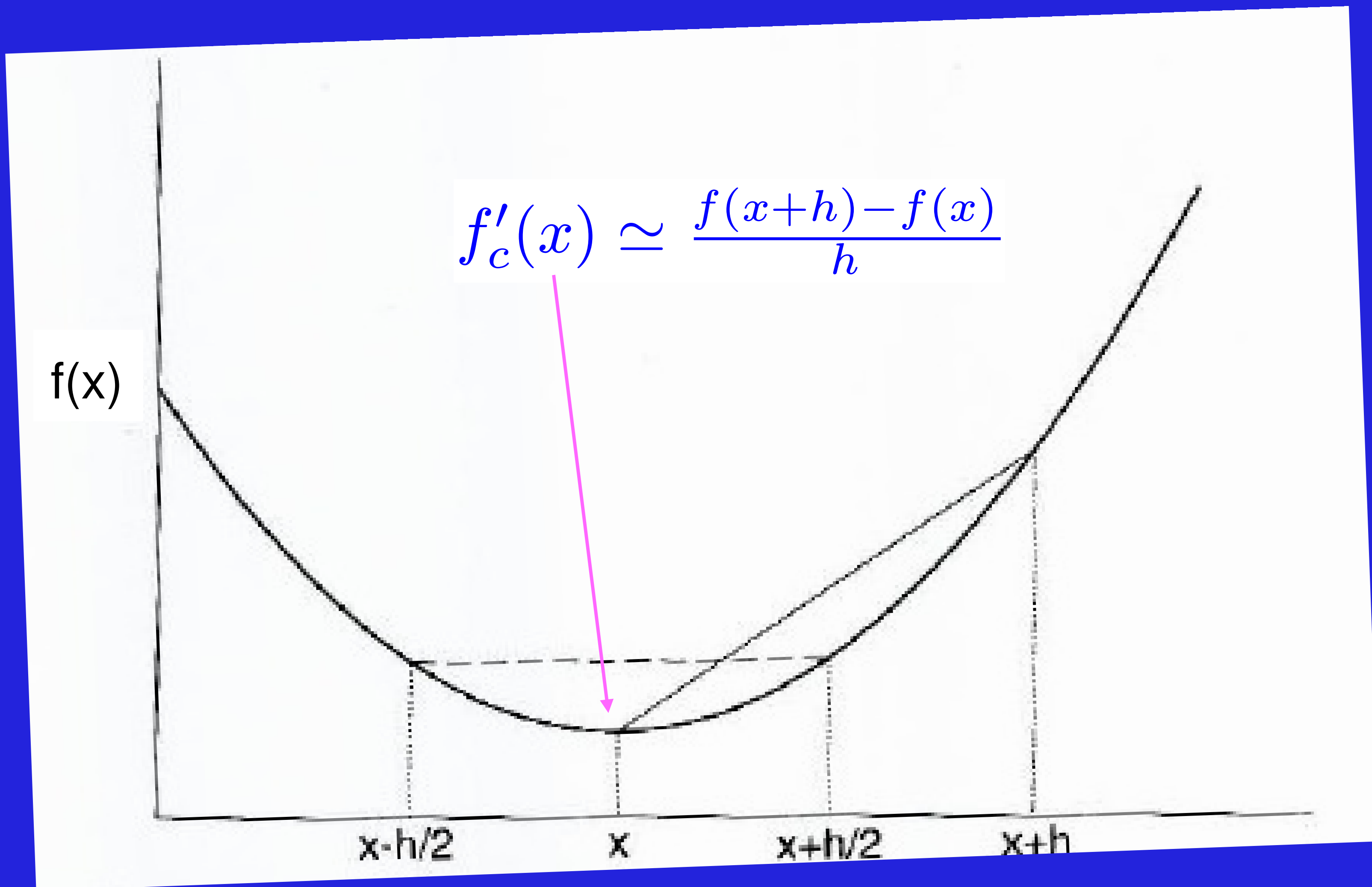
$$f'(x) = \frac{f(x+h)-f(x)}{h} - \frac{h}{2} f''(\zeta)$$

"truncation error"

$$f'_c(x) \simeq \frac{f(x+h)-f(x)}{h}$$

"forward derivative"

Forward Derivative



Not too bad (truncation error $\propto$ h), but we can do better.

Central Difference Derivative

Use alternative, but entirely equivalent, definition of df(x)/dx at x.

$$f'(x) \equiv \lim_{h\to 0} \frac{f(x+h/2)-f(x-h/2)}{h}$$

Taylor series expand f(x + h/2):

$$f(x+h/2) = f(x) + \frac{h}{2} f'(x) + \frac{h^2}{8} f''(x) + \frac{h^3}{48} f'''(x) + \ldots$$

and f(x - h/2):

$$f(x-h/2) = f(x) - \frac{h}{2} f'(x) + \frac{h^2}{8} f''(x) - \frac{h^3}{48} f'''(x) + \ldots$$

$$f'(x) = \frac{f(x+h/2)-f(x-h/2)}{h} - \frac{h^2}{24} f'''(\zeta)$$

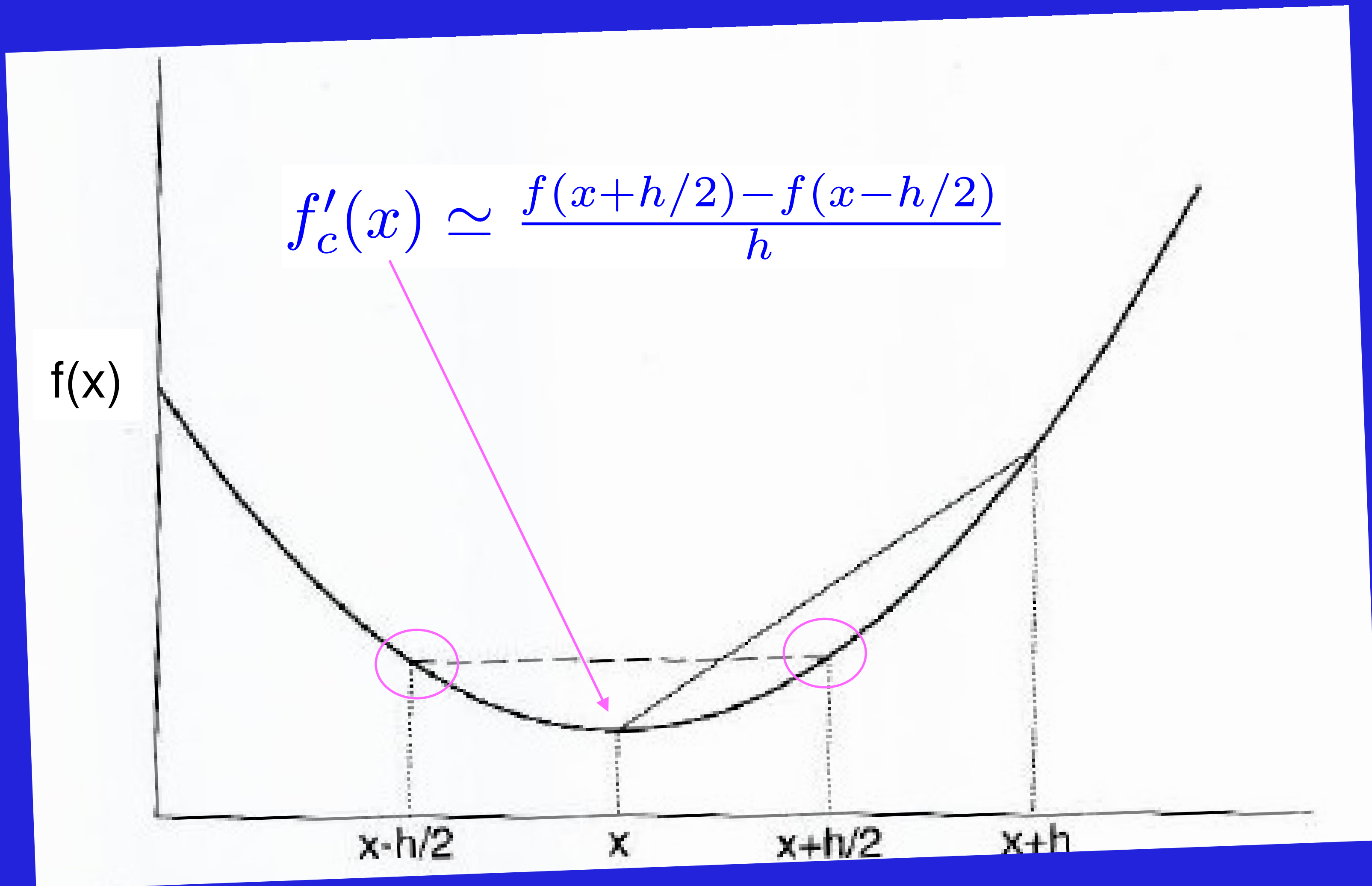
$$f'_c(x) \simeq \frac{f(x+h/2)-f(x-h/2)}{h}$$

← Our workhorse derivative

Truncation error $\propto h^2$

"central difference derivative"

Central Difference Derivative (2)



Again, our workhorse (truncation error $\propto h^2$)

GSL Routine (see derivative.cc)

```
#include <iostream>
#include <iomanip>
#include <gsl/gsl_math.h>
#include <gsl/gsl_deriv.h>

using namespace std;

double f (double x, void * params)
{
  return pow (x, 1.5);   // <----- CHANGE ME
}

int main ()
{
  gsl_function F;
  double result, abserr;

  F.function = &f; // no touch
  F.params = 0;    // no touch
```

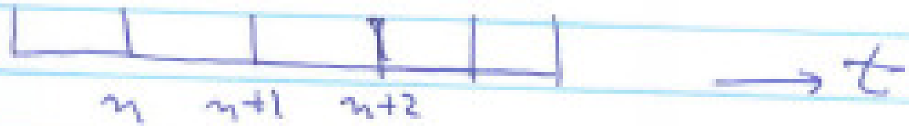
```
cout << "f(x) = x^(3/2)" << endl;

 gsl_deriv_central (&F, 2.0, 1e-8,
&result, &abserr);
  cout << "x = 2.0" << endl;
  cout << "f'(x) = " <<
setprecision(11)<< result << " +/- "
<< abserr << endl;
  cout << "exact = " <<
setprecision(11) << 1.5 * sqrt(2.0)
<< endl << endl;

  gsl_deriv_forward (&F, 0.0, 1e-8,
&result, &abserr);
  printf ("x = 0.0\n");
  printf ("f'(x) = %.10f +/- %.10f\n",
result, abserr);
  printf ("exact = %.10f\n", 0.0);

  return 0;
}
```

RUNGE-KUTTA (2ND ORDER)

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

DISCRETIZING,

$$y_{n+1} = y_n + \int_{t_n}^{t_{n+1}} f(t,y)\,dt$$



APPROXIMATE,

$$f(t,y) \simeq f(t_{n+1/2}, y_{n+1/2})$$

$$+ (t - t_{n+1/2}) \frac{df}{dt}(t_{n+1/2})$$

$$+ \mathcal{O}(h^2)$$

SUB LAST EQN INTO FIRST AND NOTE:

$$\int_{t_n}^{t_{n+1}} (t - t_{n+1/2})\, dt = \left. \frac{(t - t_{n+1/2})^2}{2} \right|_{t_n}^{t_{n+1}}$$

$$= 0$$

SO,

$$\int f(t,y)\, dt \simeq f(t_{n+1/2}, y_{n+1/2})\, h$$

$$\Rightarrow y_{n+1} \simeq y_n + h f(t_{n+1/2}, y_{n+1/2})$$

SO FAR SO GOOD, EXCEPT WE DON'T KNOW WHAT $y_{n+1/2}$ IS.

WE ONLY CALCULATE STUFF AT $t_n, t_{n+1}, \ldots$ TO GET $f(t_n, y_n)$.

HOWEVER, WE USE APPROXIMATION

$$y_{n+1/2} \simeq y_n + \frac{dy}{dt} h/2$$

$$\simeq y_n + \frac{1}{2} h f(t_n, y_n)$$

FINALLY (OR ALMOST ANYWAY),

$$\underset{\sim}{y}_{n+1} \simeq \underset{\sim}{y}_n + \underset{\sim}{k}_2$$

w/ $\underset{\sim}{k}_2 = h \underset{\sim}{f}(t_n + \frac{h}{2}, \underset{\sim}{y}_n + \underset{\sim}{k}_1/2)$

$\underset{\sim}{k}_1 = h \underset{\sim}{f}(t_n, \underset{\sim}{y}_n)$

"2ND-ORDER" RUNGE-KUTTA ALGORITHM FOR ODE SOLUTION.

THIS LOOKS VERY COMPLICATED.

LOOKS CAN BE DECEIVING !

WRITE DIFFEQ's IN STD FORMAT

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

$\tilde{y}$ & $\tilde{f}$ ARE N-DIMENSIONAL <u>VECTORS</u>

$$\tilde{y} = \begin{pmatrix} y^{(1)}(t) \\ y^{(2)}(t) \\ \vdots \\ y^{(N)}(t) \end{pmatrix}, \quad \tilde{f} = \begin{pmatrix} f^{(1)}[t, \tilde{y}] \\ f^{(2)}[t, \tilde{y}] \\ \vdots \\ f^{(N)}[t, \tilde{y}] \end{pmatrix}$$

<u>WHY?</u>

$$\frac{dy^{(1)}}{dt}(t) = f^{(1)}[t, \tilde{y}]$$

$$\frac{dy^{(2)}}{dt}(t) = f^{(2)}[t, \tilde{y}]$$

← NO y-DERIVATIVES

CONSIDER NEWTON'S 2ND LAW

$$d^2x/dt^2 = F(t, dx/dt, x)$$

WRITE IN STD FORM

$$y^{(1)}(t) \equiv x(t)$$ EASY

$$dx/dt = dy^{(1)}/dt \equiv y^{(2)}(t)$$ EASY.

SO,

$$\frac{dy^{(1)}}{dt}(t) = y^{(2)}(t)$$

$$\frac{dy^{(2)}}{dt}(t) = F(t, y^{(1)}, y^{(2)})$$

EQUATIONS IN VECTOR FORM

w/

$$f^{(1)} = y^{(2)}(t)$$

$$f^{(2)} = F(t, y^{(1)}, y^{(2)})$$

EXAMPLE:

$$m\,d^2x/dt^2 = -kx$$

$$d^2x/dt^2 = -\frac{k}{m}x$$

$$y^{(1)}(t) = x(t)$$

$$\frac{dy^{(1)}}{dt} = \boxed{f^{(1)} = y^{(2)}(t)}$$

$$d^2x/dt^2 = \frac{d}{dt}\left(\frac{dy^{(1)}}{dt}\right) = -\frac{k}{m}x$$

$$\frac{d}{dt}y^{(2)}(t) = -\frac{k}{m}x$$

$$\boxed{f^{(2)} = -\frac{k}{m}y^{(1)}}$$

4TH-ORDER RUNGE-KUTTA

WE USE

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \frac{1}{6}\left(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4\right)$$

w/

$$\mathbf{k}_1 = h\,\mathbf{f}(t_n, \mathbf{y}_n)$$

$$\mathbf{k}_2 = h\,\mathbf{f}(t_n + h/2, \mathbf{y}_n + \mathbf{k}_1/2)$$

$$\mathbf{k}_3 = h\,\mathbf{f}(t_n + h/2, \mathbf{y}_n + \mathbf{k}_2/2)$$

$$\mathbf{k}_4 = h\,\mathbf{f}(t_n + h, \mathbf{y}_n + \mathbf{k}_3)$$

YIKES !

Summary

Numerical derivatives: forward and central difference.

ODE solver: 4th order Runge-Kutta

Don’t suffer in silence. Scream for help!!!