# Lect 16: n body simulation

1. Model
2. How to code it

# Basic integrals

$$v' = v + at$$

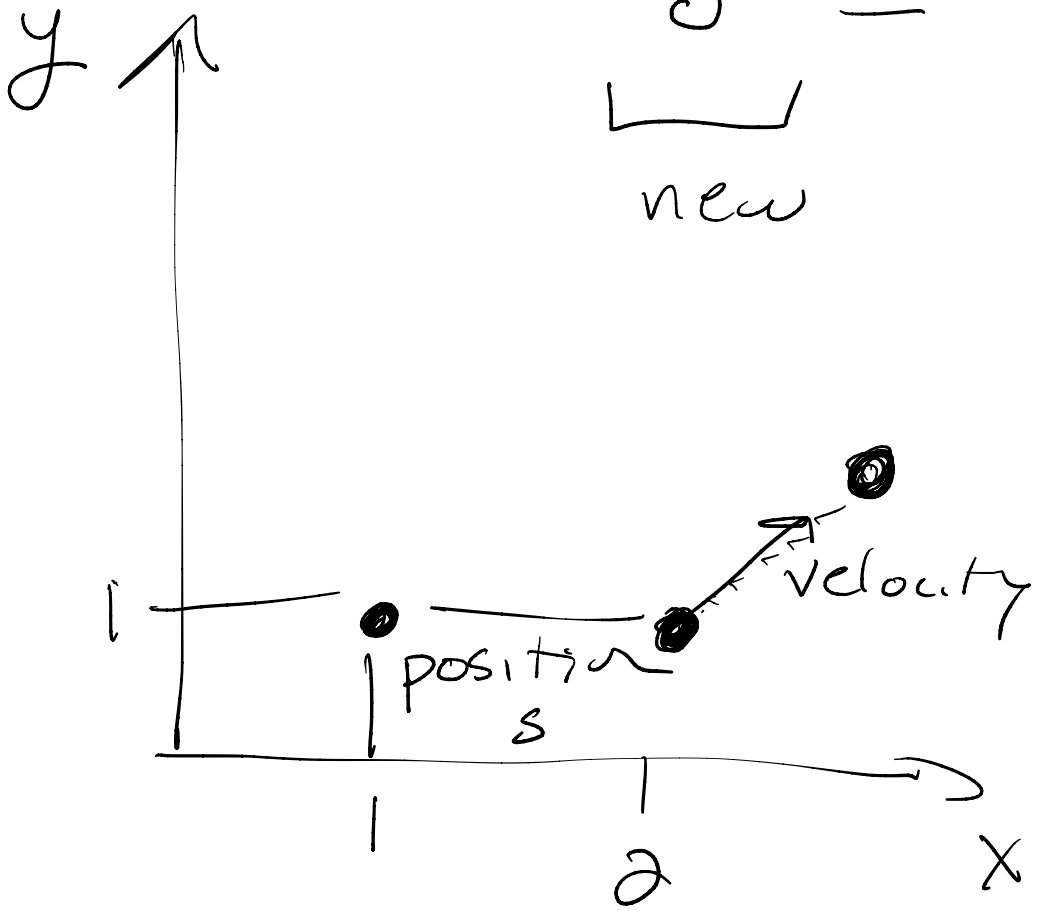
$$s' = s + vt + \frac{1}{2}at^2$$

new

time

accel.

velocity



# Newton's 2nd Law

$F = ma$

- $F$: force
- $m$: mass
- $a$: accel.

$a = \frac{F}{m}$

- $a$: accel. of that planet
- $F$: force on that planet
- $m$: mass of that planet

# Newton's Law of Gravitation

force on planet $i$: $F_i = \sum_{j=i} F_{ij}$

$F_{ij}$: force on planet $i$ due to planet $j$

$$F_{ij} = \frac{G m_i m_j}{r_{ij}^2}$$

- $G$: constant $6.67 \times 10^{-11}\ N(m/kg)^2$
- $m_i$: mass of planet $i$
- $m_j$: mass of planet $j$
- $r_{ij}$: distance from $i$ to $j$

$$a_i = \frac{F_i}{m_i}$$

accel on planet $i$ $= \dfrac{\sum_j F_{ij}}{m_i}$

$$= \sum_j F_{ij}/m_i$$

$$= \sum_j a_{ij}$$

$a_{ij}$: accel on body $i$ due to body $j$

$$= \sum_j \frac{\frac{G m_i m_j}{(r_{ij})^2}}{m_i}$$

$$= \sum_j \frac{G m_j}{(r_{ij})^2}$$

Magnitude of the accel

$$\vec{a}_i = \sum_j \hat{d}_{ij} \left( \frac{G m_j}{|d_{ij}|^2} \right)$$

accel vector on body i

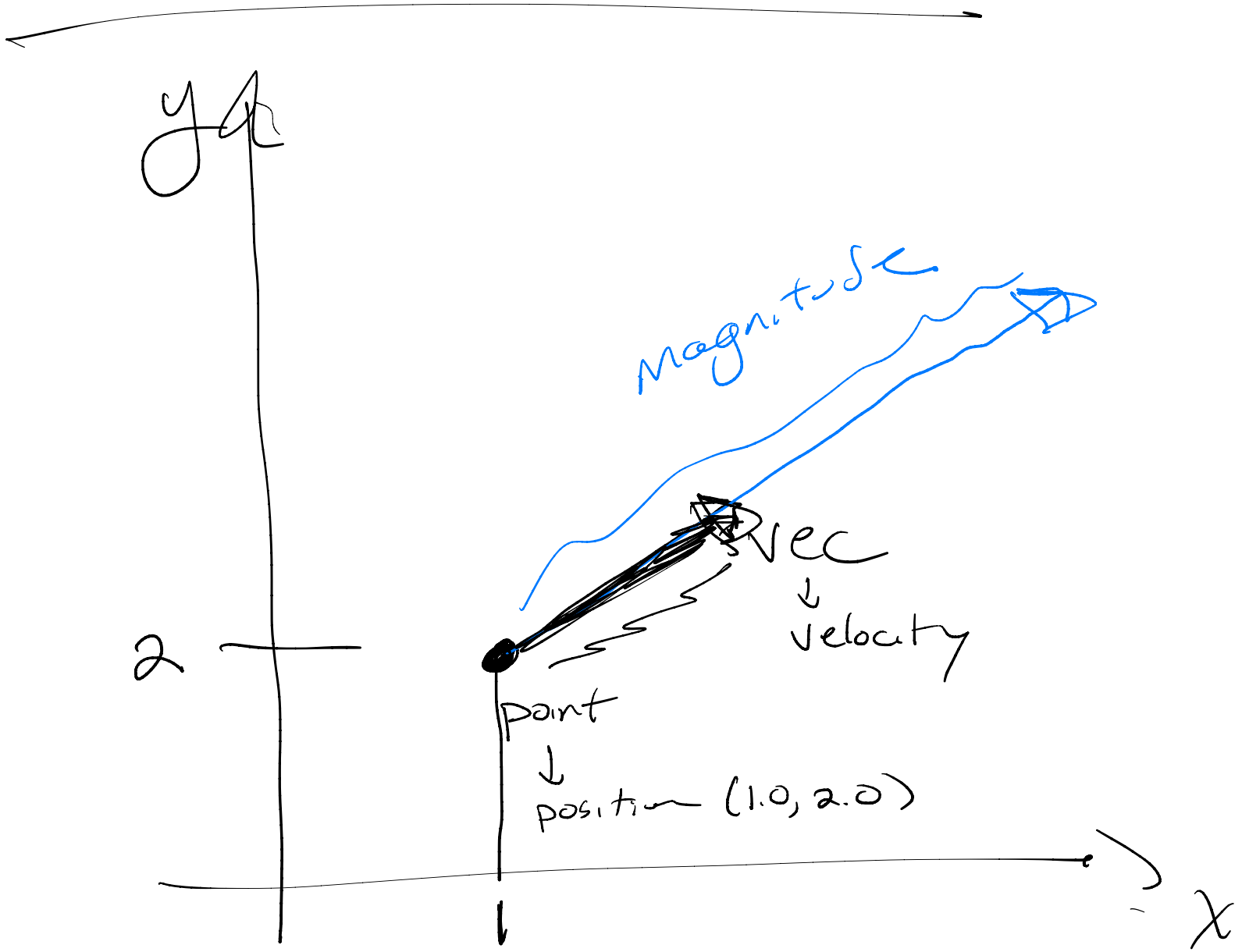
$\hat{d}_{ij}$: unit vector in dir of $d_{ij}$

magnitude

distance

$d_{ij}$ is the vector from body i to body j

# Points + vectors



```
(*   x,y - coordinate   *)
type point = real * real

(* position of the head of the vector,
     if the tail is (0,0)                *)

type vec = real * real
```

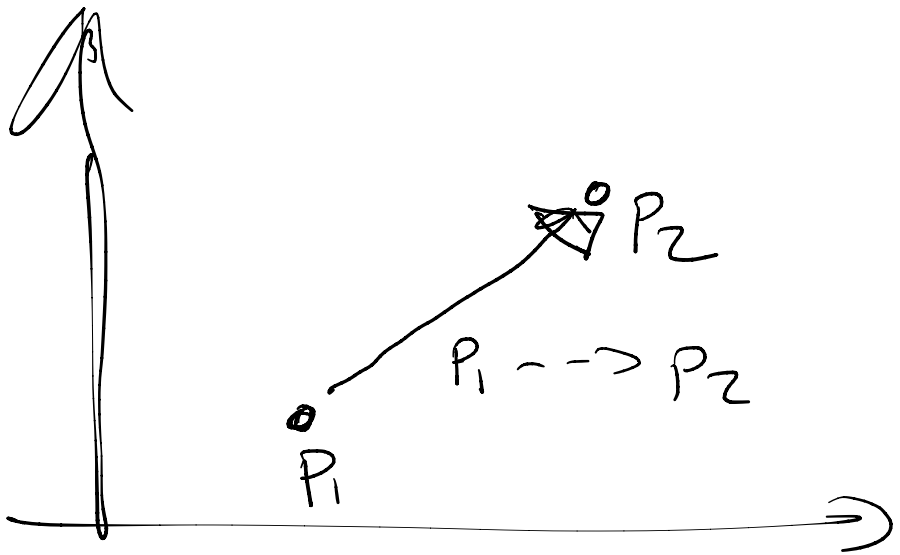
infix 3 -->

--> (P1, P2)

fun (p1 : point) --> (p2 : point) : vec =

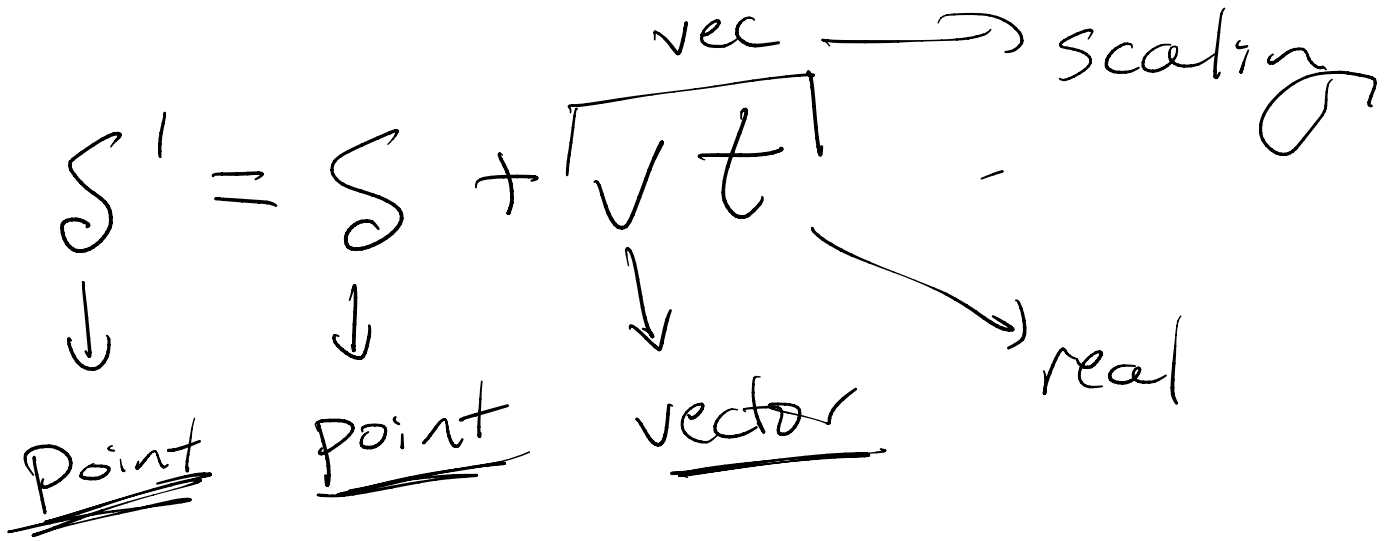
(x2, y2) (x1, y1)

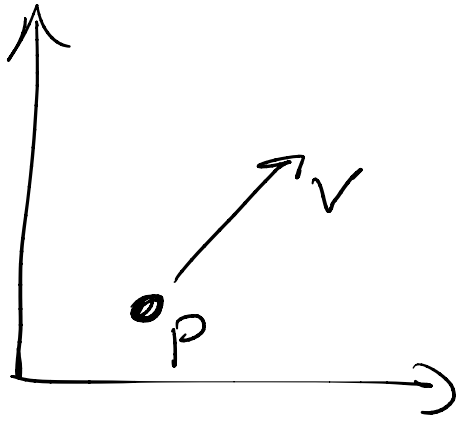
( x2 - x1 , y2 - y1 )




Vector from P1 to P2

```
fun collided ((x1,y1):point, (x2,y2):point):bool=
    Real.==(x1, x2) andalso
    Real.==(y1, y2)
```




(* compute head of v, if tail is at p *)

fun displace((x,y): point, (x',y'): vec): point =

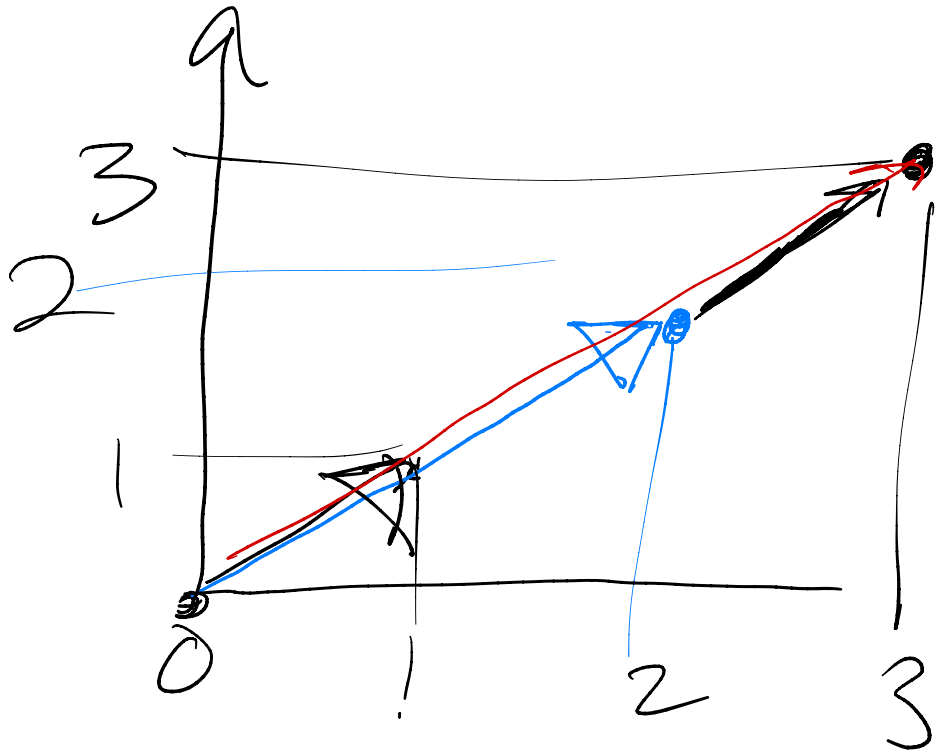
(x + x', y + y')

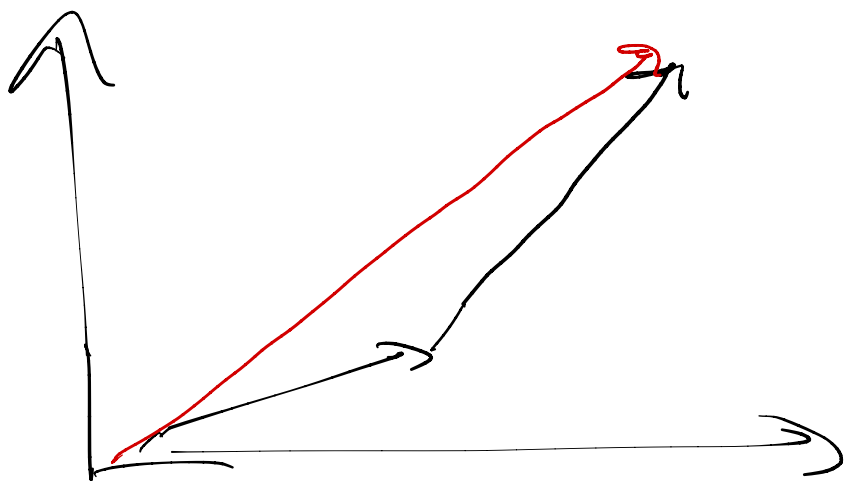
$$s' = s + vt + \frac{1}{2}at^2$$

point

vec

$$v_1 + v_2$$



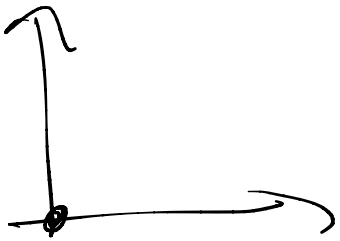



infix / 3 ++

fun ((x1, y1) : vec) ++ ((x2, y2) : vec) : vec =

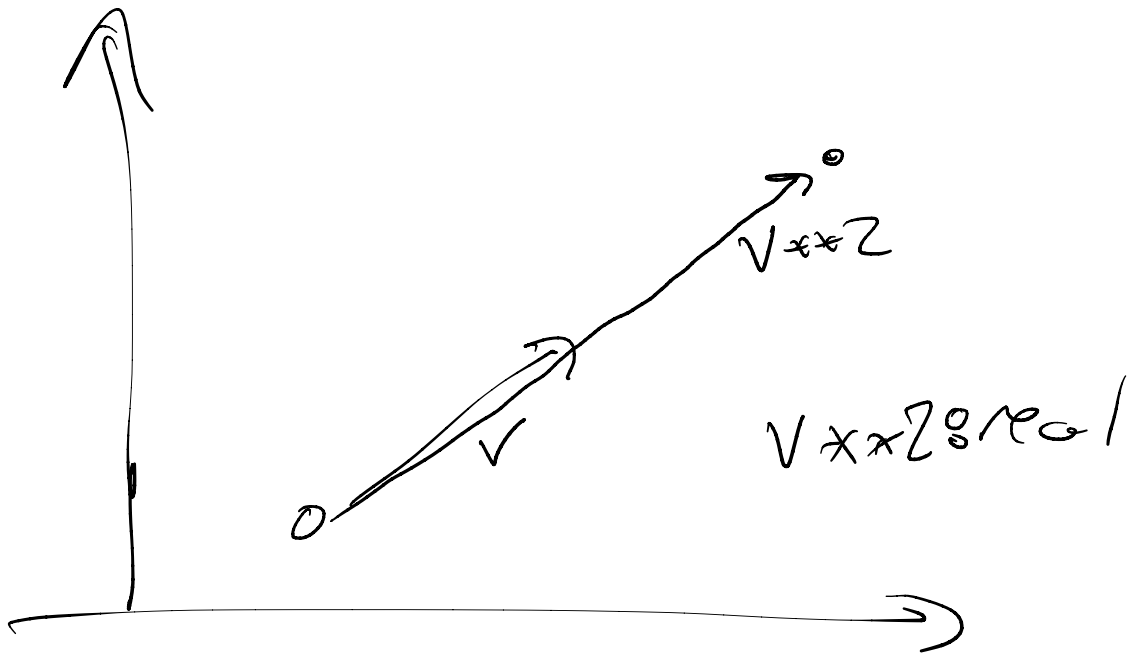
(x1 + x2, y1 + y2)

vector addition



val zero_vec = (0.0, 0.0)

```
infixr 4 **
fun ((x1, y1)) ** (c:real) : vec =
    (x1 * c,  y1 * c)
```

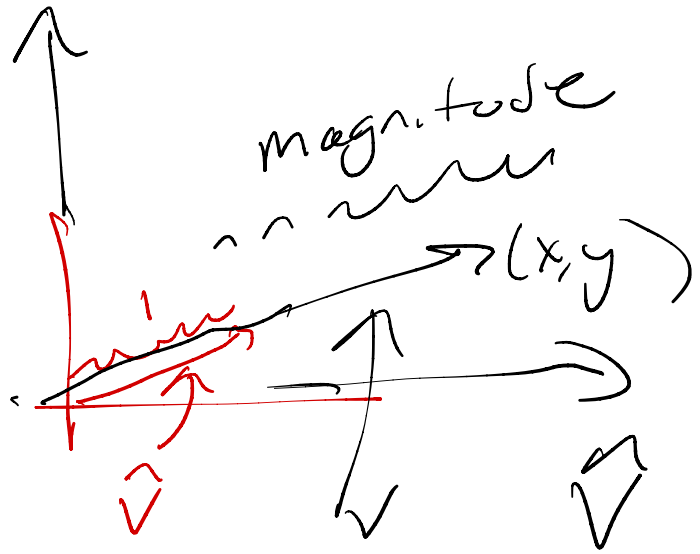



```
s++(v**t)

1+(x*5)
```

```
fun mag((x,y): vec): real =
    Math.sqrt(x*x + y*y)

fun unitVec(v:vec): vec =
    V**(1/mag v)
```




$\sqrt{x^2+y^2}$

$\hat{v}$ unit vec

```
type point
type vec

val --> : point * point → vec
val collided : point * point → bool
val displace : point * vec → point
val ++ : vec * vec → vec
val ** : vec * real → vec

val zero : vec
val mag : vec → real
val unitVec : vec → vec
```

$$s' = s + vt + \frac{1}{2}at^2$$

$$v' = v + at$$

```
                        mass * position * velocity
type body =             real * point *    vec

fun stepBody ((m, s, v) : body, a : vec, t : real) : body =
    (m,
     displace (s, (v ** t) ++ (a ** (0.5 * t * t))),
     v ++ (a ** t))
```

(* accel on body1 dzto body2 *)

```
fun accOn((m1, p1, v1): body, (m2, p2, v2): body) : vec =
    case collideδ(p1, p2) of
      tve => zero
    | false => let val d12 = p1 --> p2
               in
                 (unitVec d12) **
                 ((G * m2) // ((mag d12) * (mag d12)))
               end
```

code

math

$$\hat{d}_{12}\left(\frac{G m_2}{|d_{12}|^2}\right)$$

body seq.seq ] all bodies

```
fun accelerations (bodies: body seq.seq):
                                  vec seq.seq =
(* given <b1, b2, ----->
   make <a1, a2, ----->  *)
```

Math:

$$\left\langle \left\langle \sum_j \vec{a}_{1j}, \sum_j \vec{a}_{2j}, \text{---} \right\rangle \right\rangle$$

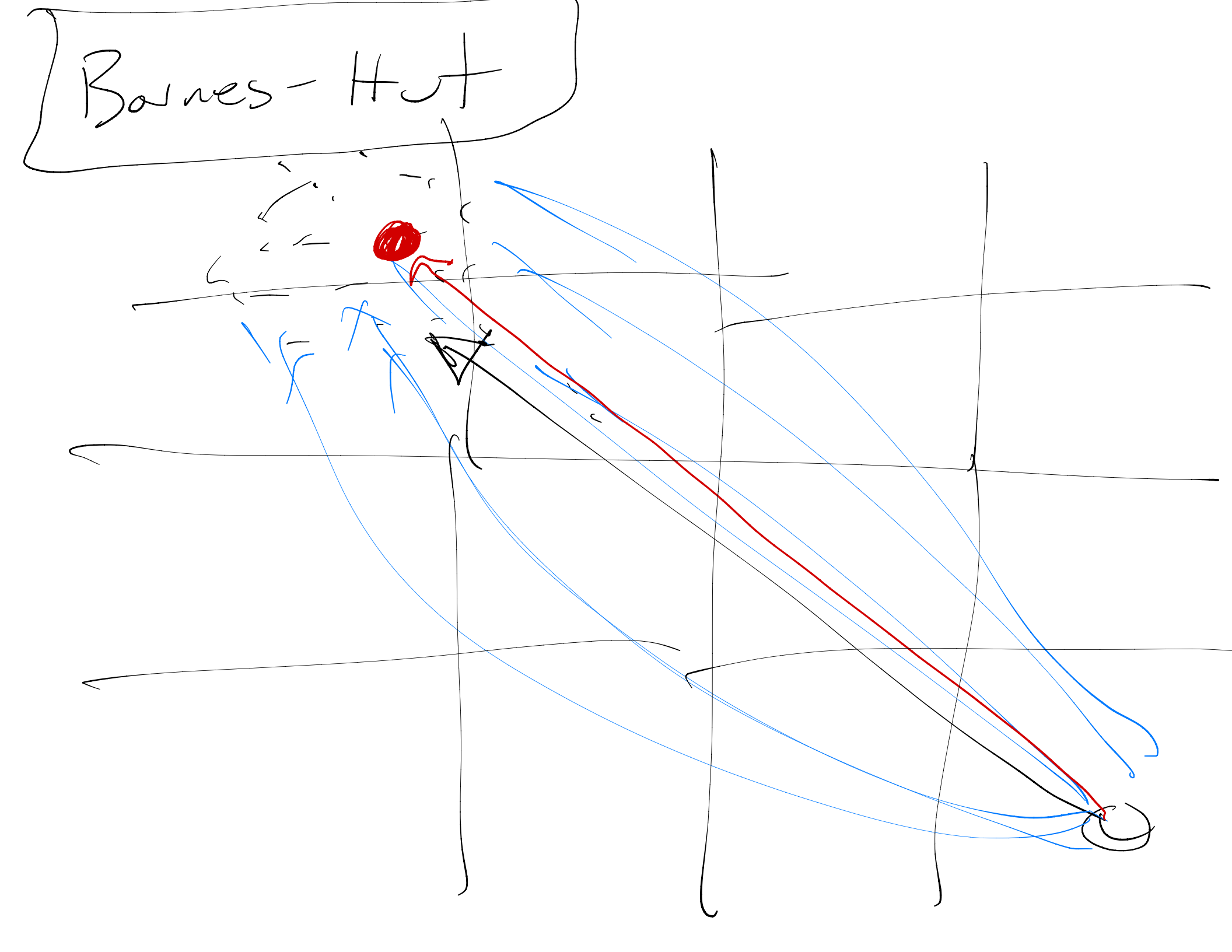
*)

```
fun accelerations(bodies) =
  Seq.map(fn bodyi =>
    Seq.reduce(fn (a1,a2) => a1++a2,
               zero,
               Seq.map(fn bodyj =>
                         accOn(bodyi, bodyj),
                       bodies)),
    bodies)
```

$$\sum_{j=1}^{n} \vec{a_{ij}}$$

Work + span — n bodies

| | input | work | span |
|---|---|---|---|
| outer map | n | $O(n^2)$ | $O(\log n)$ |
| inner map | n | $O(n)$ | $O(1)$ |
| reduce | n | $O(n)$ | $O(\log n)$ |


Barnes - Hut