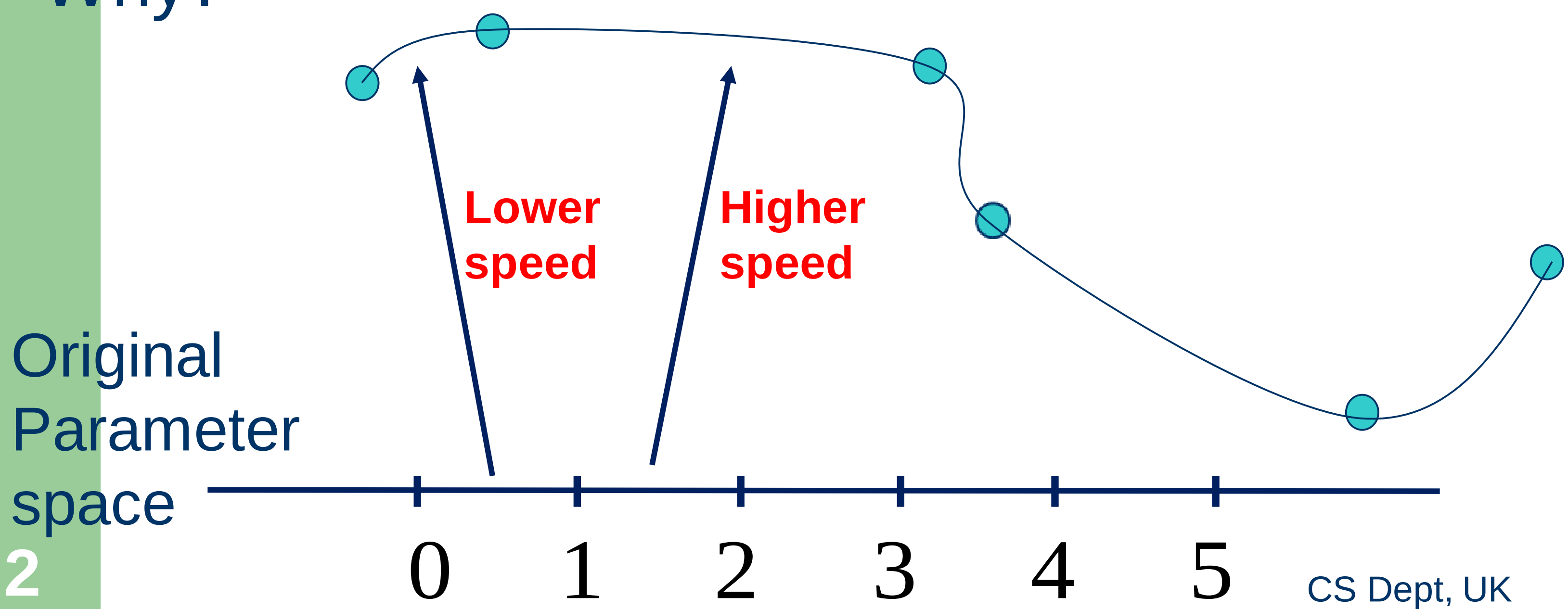
3.2 Controlling Motion Along Space Curve

- constant speed
- speeding up
- speeding down

Constant speed

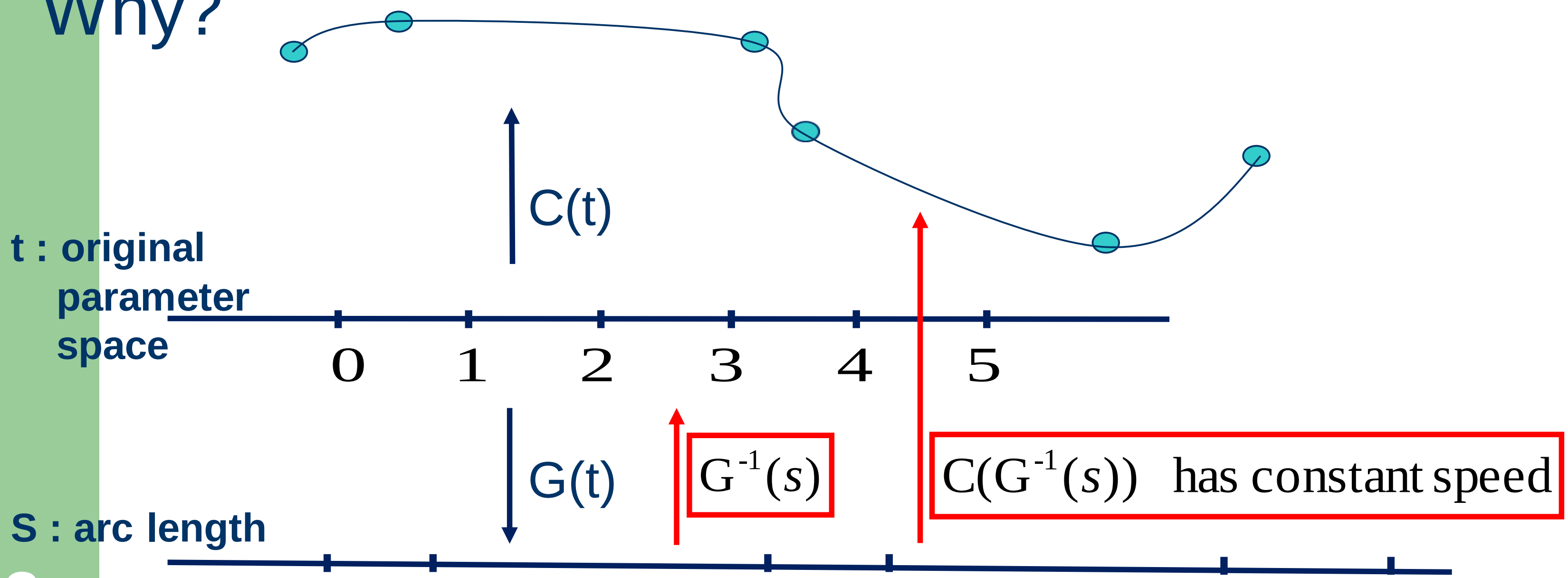
To ensure constant speed, must **parametrize** by arc length (constant *distance-time* function)

Why?



Constant speed

- To ensure constant speed, must **parametrize** by arc length (constant *distance-time* function)
- Why?



Constant speed

However, calculating *arc length*

$$s = \int_{t_1}^{t_2} \left| \frac{dC(t)}{dt} \right| dt$$

is too difficult (sometimes, impossible)

WHY?

Constant speed

$$C(t) = (x(t), y(t), z(t))^T = at^3 + bt^2 + ct + d$$

$$= \begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix} t^3 + \begin{pmatrix} b_x \\ b_y \\ b_z \end{pmatrix} t^2 + \begin{pmatrix} c_x \\ c_y \\ c_z \end{pmatrix} t + \begin{pmatrix} d_x \\ d_y \\ d_z \end{pmatrix}$$

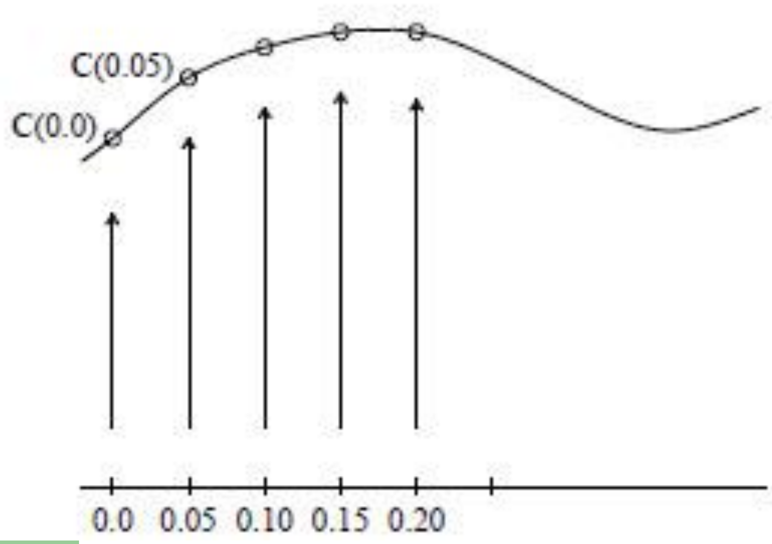
$$\left| \frac{dC(t)}{dt} \right| = \left[\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2 + \left(\frac{dz}{dt}\right)^2 \right]^{1/2}$$

$$= \sqrt{At^4 + Bt^3 + Ct^2 + Dt + E}$$

Sometime impossible to compute

Constant speed

Remedy I: estimate arc length by *forward differencing*



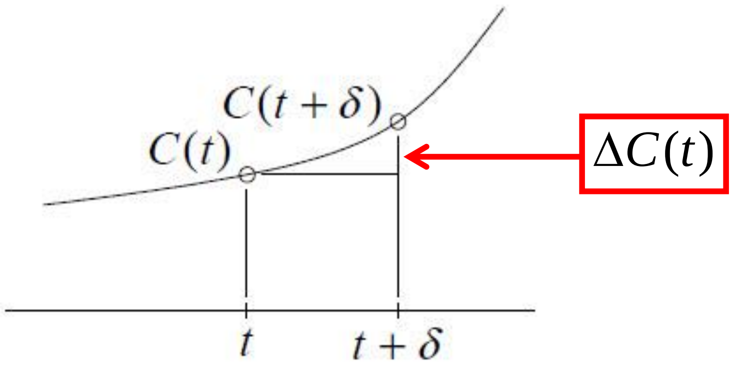
Create a table

Index	Parametric Value	Arc Length
0	0.00	0.000
1	0.05	0.080
2	0.10	0.150
3	0.15	0.230
4	0.20	0.320
5	0.25	0.400

Forward Differencing

$$C(t) = at^4 + bt^3 + ct^2 + dt + e$$

$$C(t+\delta) = C(t) + \Delta C(t)$$



Given

$C(0)$	$C(\delta)$	$C(2\delta)$	$\cdots$
$\Delta C(0)$	$\Delta C(\delta)$	$\Delta C(2\delta)$	$\cdots$
$\Delta^2 C(0)$	$\Delta^2 C(\delta)$	$\Delta^2 C(2\delta)$	$\cdots$
$\Delta^3 C(0)$	$\Delta^3 C(\delta)$	$\Delta^3 C(2\delta)$	$\cdots$
$\Delta^4 C(0)$ =	$\Delta^4 C(\delta)$ =	$\Delta^4 C(2\delta)$ =	$\cdots$

Forward Differencing

To compute a new point, only 4 **additions** are needed. Why?

$$\Delta C(t) \equiv C(t+\delta) - C(t)$$
$$= (4a\delta)t^3 + (6a\delta^2 + 3b\delta)t^2$$
$$+ (4a\delta^3 + 3b\delta^2 + 2c\delta)t$$
$$+ (a\delta^4 + b\delta^3 + c\delta^2 + d\delta)$$

$$\Rightarrow \quad C(t+\delta) = C(t) + \Delta C(t)$$

$$\Delta^2 C(t) \equiv \Delta C(t+\delta) - \Delta C(t)$$
$$= (12a\delta^2)t^2 + (24a\delta^3 + 6b\delta^2)t$$
$$+ (14a\delta^4 + 6b\delta^3 + 2c\delta^2)$$

$$\Rightarrow \quad \Delta C(t+\delta) = \Delta C(t) + \Delta^2 C(t)$$

Forward Differencing

constant

$$\Delta^3 C(t) = \Delta^2 C(t+\delta) - \Delta^2 C(t)$$
$$= (24a\delta^3)t + (36a\delta^4 + 6b\delta^3)$$

$$\Rightarrow \Delta^2 C(t+\delta) = \Delta^2 C(t) + \Delta^3 C(t)$$

$$\Delta^4 C(t) = \Delta^3 C(t+\delta) - \Delta^3 C(t)$$
$$= 24a\delta^4$$

$$\Rightarrow \Delta^3 C(t+\delta) = \Delta^3 C(t) + \Delta^4 C(t)$$

We have

$$C(t+\delta) = C(t) + \Delta C(t)$$
$$\Delta C(t+\delta) = \Delta C(t) + \Delta^2 C(t)$$
$$\Delta^2 C(t+\delta) = \Delta^2 C(t) + \Delta^3 C(t)$$
$$\Delta^3 C(t+\delta) = \Delta^3 C(t) + \Delta^4 C(t)$$

$\Rightarrow$

$C(t) \rightarrow C(t+\delta)$

$\Delta C(t) \rightarrow \Delta C(t+\delta)$

$\Delta^2 C(t) \rightarrow \Delta^2 C(t+\delta)$

$\Delta^3 C(t) \rightarrow \Delta^3 C(t+\delta)$

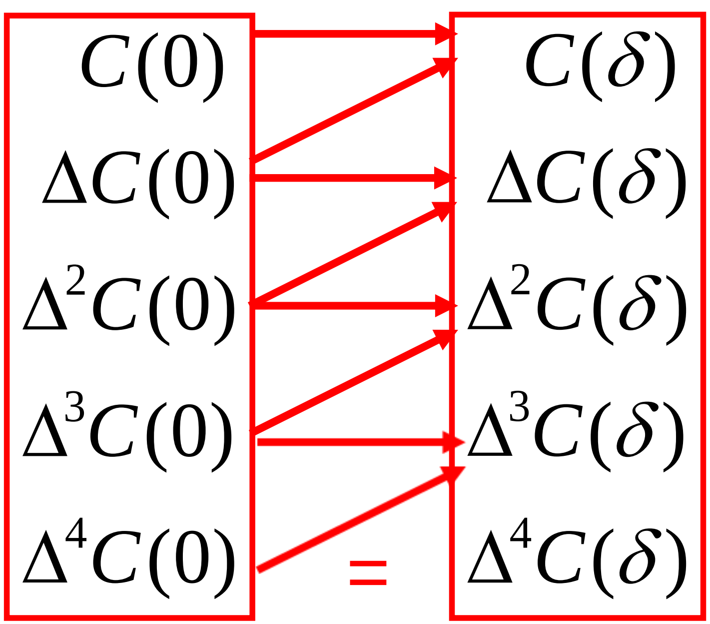
$\Delta^4 C(t) = \Delta^4 C(t+\delta)$

Forward Differencing

Hence, if we know

$$C(0), \quad \Delta C(0), \quad \Delta^2 C(0), \quad \Delta^3 C(0), \quad \Delta^4 C(0)$$

then

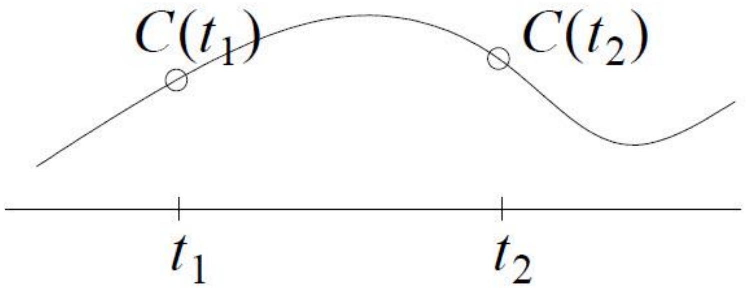


Disadvantage: large numerical error

Error would propagate all the way from the start to the last point

Constant Speed

Let *LENGTH*(t_1, t_2) be the length of the space curve from $C(t_1)$ to $C(t_2)$

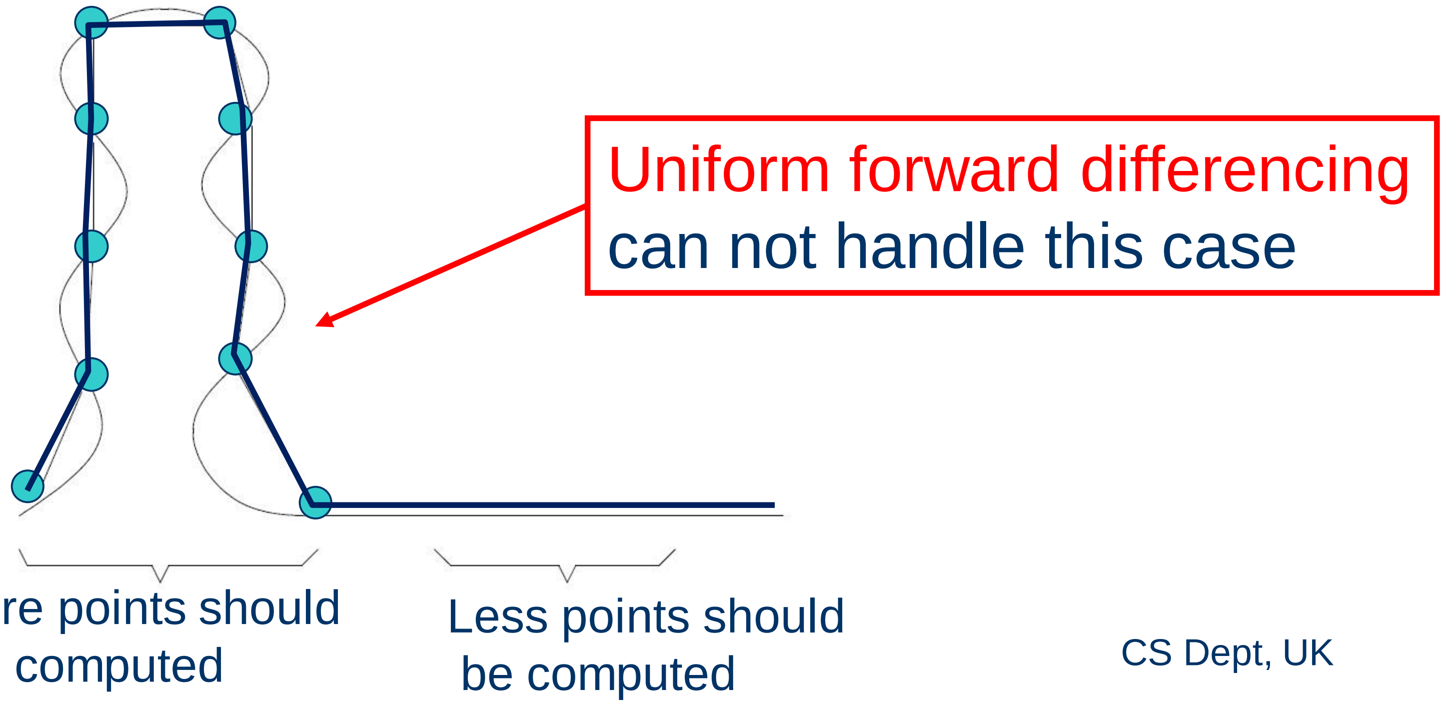


Need to solve two problems:

1. Given t_1 *and* t_2, find *LENGTH*(t_1, t_2)
2. Given the arc length s and a parameter value t_1, find t_2 so that *LENGTH*$(t_1, t_2) = s$

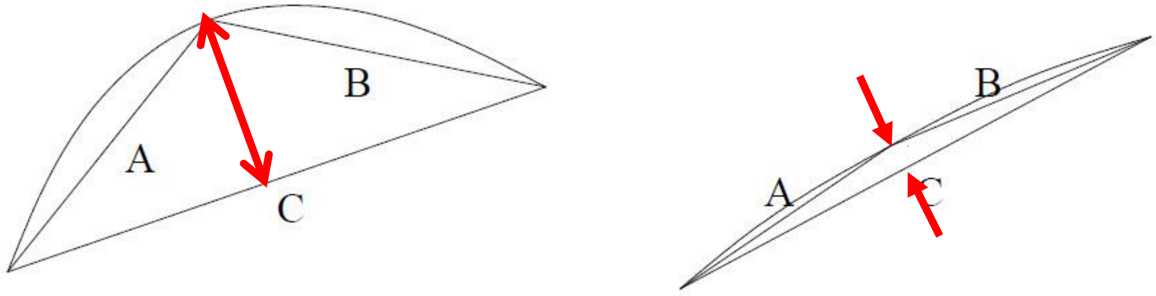
APPROACH II: estimate arc length by *adaptive subdivision*

(uniform forward differencing is easy, fast and intuitive, but generates large numerical error) and



APPROACH II: estimate arc length by *adaptive subdivision*

Idea: using *chordal deviation* to determine if a region should be further subdivided



If *Length(A) + Length(B) - Length(C)* > ε

further subdivide the segment

otherwise

stop subdivision of the segment

APPROACH II: estimate arc length by *adaptive subdivision*

Algorithm:

i = 0; s = 0;

ArcLengthTable[i] ← *(0, s) ;*

STACK ← Push [0, 1] ;

while (STACK not empty) {

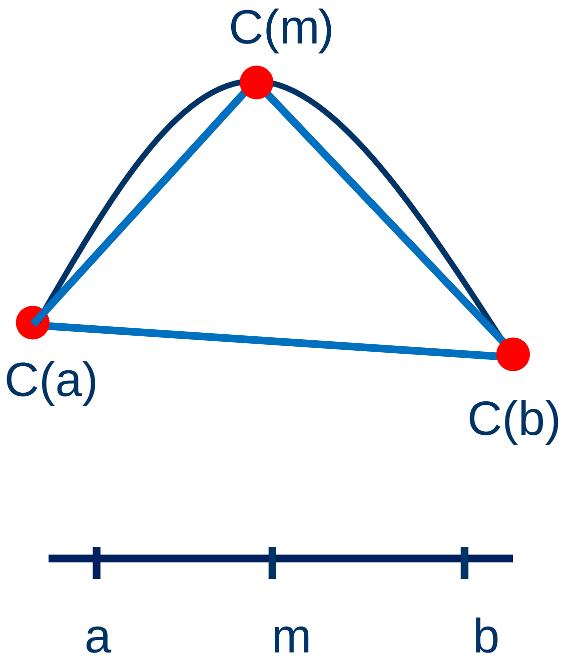
[*a, b*] ← *Pop STACK;*

m ← *(a + b)/2 ;*

L1 ← *Length(C(a), C(m));*

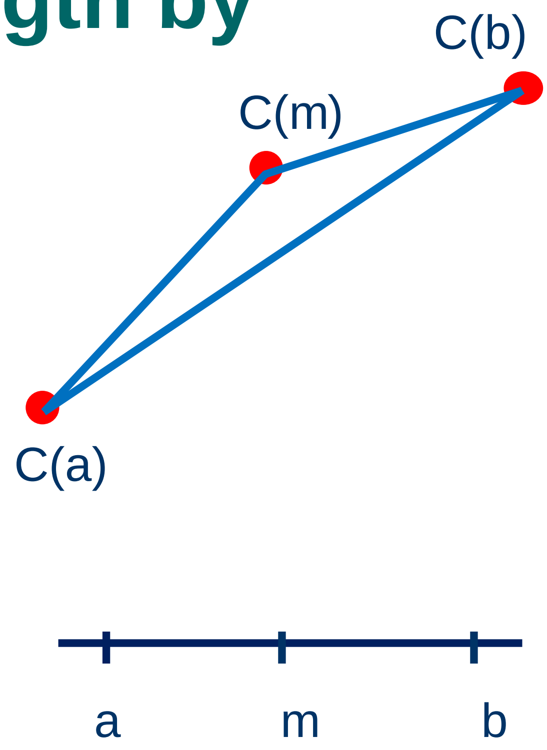
L2 ← *Length(C(m), C(b));*

L3 ← *Length(C(a), C(b));*



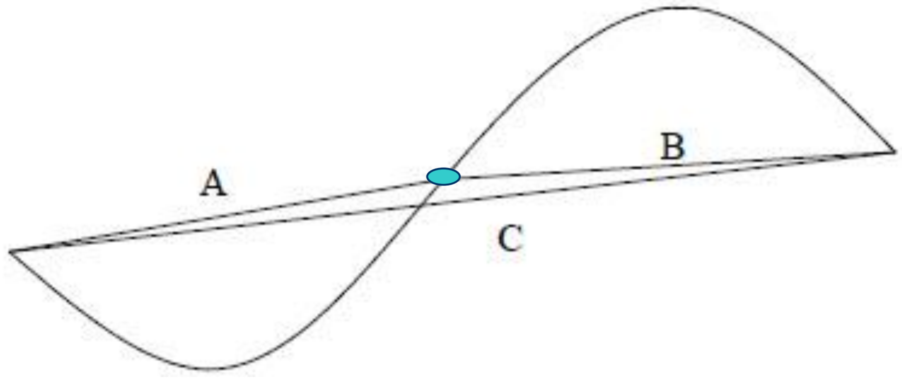
APPROACH II: estimate arc length by *adaptive subdivision*

```
if (L1 + L2 - L3 > ε) {
    STACK ← Push [m, b];
    STACK ← Puch [a, m];
}
else {
    s = s + L1;
    ArcLengthTable[i + +] ← (m, s);
    s = s + L2;
    ArcLengthTable[i + +] ← (b, s);
}
}
```



APPROACH II: estimate arc length by *adaptive subdivision*

Potential Problem: can not detect the following situation



Possible solution: force the subdivision down to a certain level then embark the adaptive subdivision t(u) = ((b-a)/2)u + (b+a)/2

APPROACH III: computing arc length numerically

$$s = \int_a^b \left|\frac{dC(t)}{dt}\right| dt = \int_a^b f(t)dt$$

$$= \int_{-1}^{1} \frac{dt(u)}{du} f(t(u))\, du$$

$$= \int_{-1}^{1} \frac{b-a}{2} f(t(u))\, du$$

$$= \int_{-1}^{1} F(u)\, du$$

$$= \sum_{i=1}^{n} w_i F(x_i)$$

$$w_i = \int_{-1}^{1} l_i(x)dx$$

w_i : Gaussian weights

$$l_i(x) = \prod_{j=1, j\neq i}^{n} \left(\frac{x - x_j}{x_i - x_j}\right)$$

x_i : Gaussian nodes

APPROACH III: computing arc length numerically

If $C(t) = at^3 + bt^2 + ct + d$

$$= \begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix} t^3 + \begin{pmatrix} b_x \\ b_y \\ b_z \end{pmatrix} t^2 + \begin{pmatrix} c_x \\ c_y \\ c_z \end{pmatrix} t + \begin{pmatrix} d_x \\ d_y \\ d_z \end{pmatrix}$$

$$= \begin{pmatrix} a_x t^3 + b_x t^2 + c_x t + d_x \\ a_y t^3 + b_y t^2 + c_y t + d_y \\ a_z t^3 + b_z t^2 + c_z t + d_z \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

APPROACH III: computing arc length numerically

we have

$$\left(\frac{dx}{dt}\right)^2 = \left(3a_x t^2 + 2b_x t + c_x\right)^2$$

$$= 9a_x t^4 + 12a_x b_x t^3 + (6a_x c_x + 4b_x^2)t^2$$

$$+ 4b_x c_x t + c_x^2$$

Then

$$f(t) = \left|\frac{dC(t)}{dt}\right| = \left[\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2 + \left(\frac{dz}{dt}\right)^2\right]^{1/2}$$

$$= \sqrt{At^4 + Bt^3 + Ct^2 + Dt + E}$$

APPROACH III: computing arc length numerically

where

$$A = 9(a_x^2 + a_y^2 + a_z^2)$$

$$B = 12(a_x b_x + a_y b_y + a_z b_z)$$

$$C = 4(b_x^2 + b_y^2 + b_z^2) + 6(a_x c_x + a_y c_y + a_z c_z)$$

$$D = 4(b_x c_x + b_y c_y + b_z c_z)$$

$$E = c_x^2 + c_y^2 + c_z^2$$

APPROACH III: computing arc length numerically

How to build an *arc length table?*

- uniform Gaussian integration
- adaptive Gaussian integration

similar to adaptive subdivision except

$$Length(a,b) = \int_a^b \left| \frac{dC(t)}{dt} \right| dt$$

APPROACH III: computing arc length numerically

Once the *arc length table* is created, then

1) for given t_1 *and* t_2*, how to find* $Length(t_1, t_2)$ *?*

2) for given t_1 *and* s*, how to find* t_2 *so that*

$$s = Length(t_1, t_2) ?$$

For (ii), use Newton-Raphson method

Let $f(t) = s - Length(t_1, t),$ construct a sequence of points $\{ p_n \}$ as follows

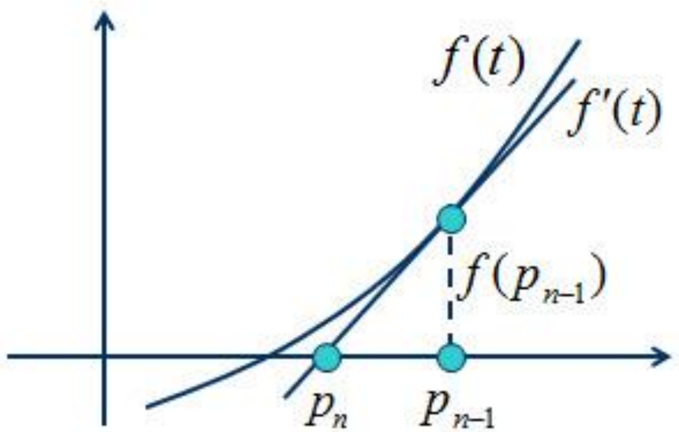
$$p_n = p_{n-1} - \frac{f(p_{n-1})}{f'(p_{n-1})}$$

APPROACH III: computing arc length numerically

the limit point of { p_n }

$$\lim_{n\to\infty} p_n = t_2$$

is the t_2 that we are looking for.



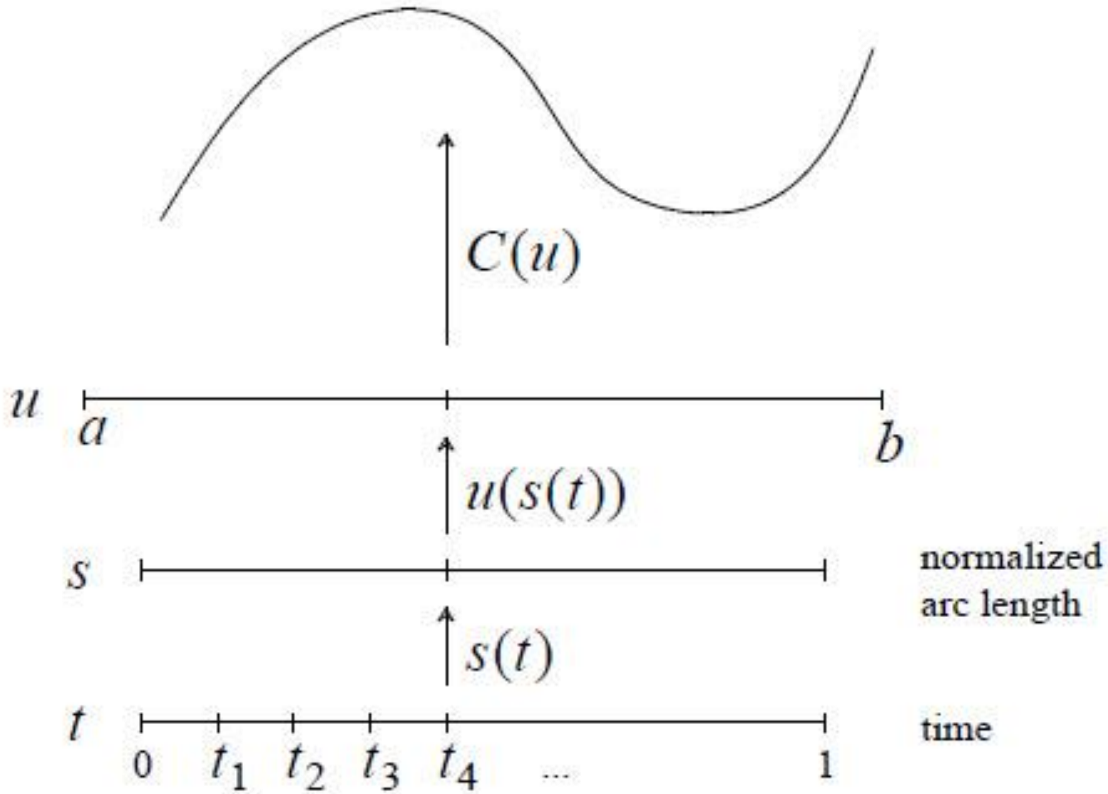
$$f'(p_{n-1}) = \frac{f(p_{n-1})}{p_{n-1} - p_n}$$

However, at any point, if $f'(p_{n-1}) = 0$ or is very close to zero, use *binary subdivision* to find t_2.

Speed Control: general idea

Time is always uniformly subdivided.

For each t_i, if we know $s(t_i)$, then use the arc length table to find the corresponding u, then compute $C(u)$ to find the location of the object at time t_i



Speed Control: general idea

The **core** of speed control is the construction of the *distance-time function*, *s(t).*

Constraints:

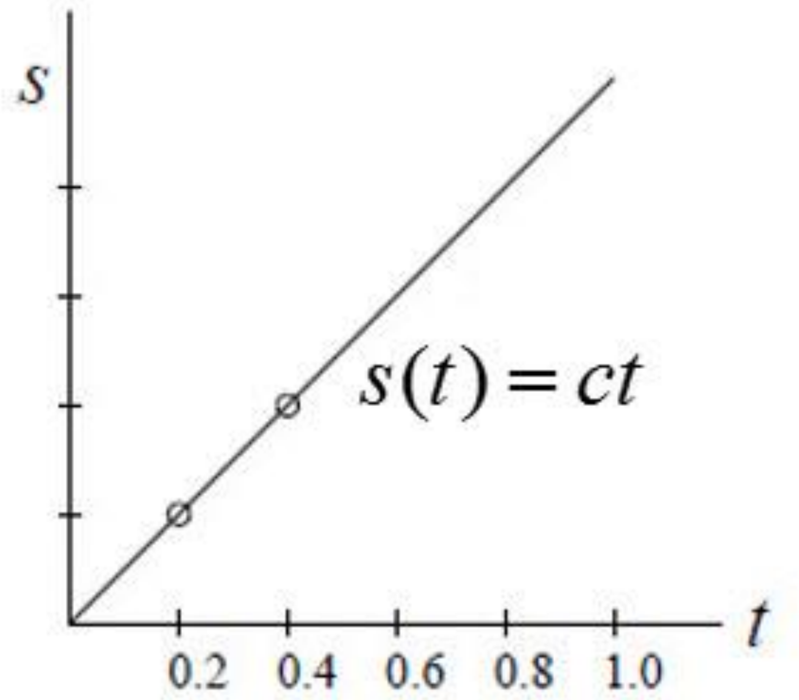
1. *s(t)* is monotonic in *t* ← Strictly increasing
2. *s(t)* is continuous ← No jumps

Possible Choices:

1. constant speed
2. ease-in/ease-out ← start slowly, be fastest at the middle of the animation, then finish slowly
3. constant acceleration
4. general *distance-time functions*

Speed Control

1. Constant speed

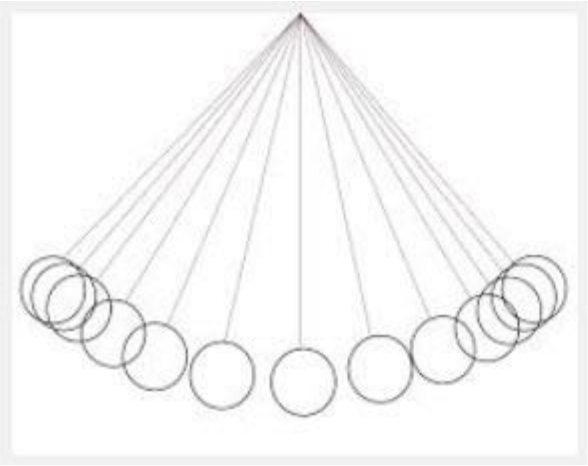


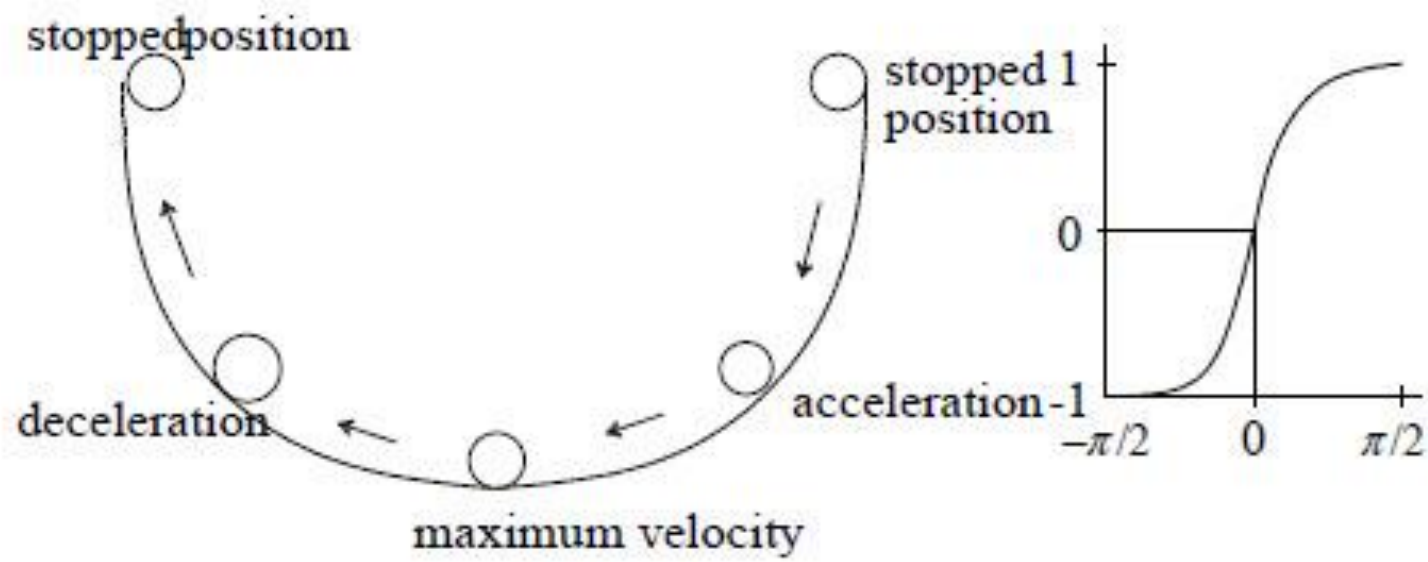
Speed Control

2. Easy-in/easy-out

Example:

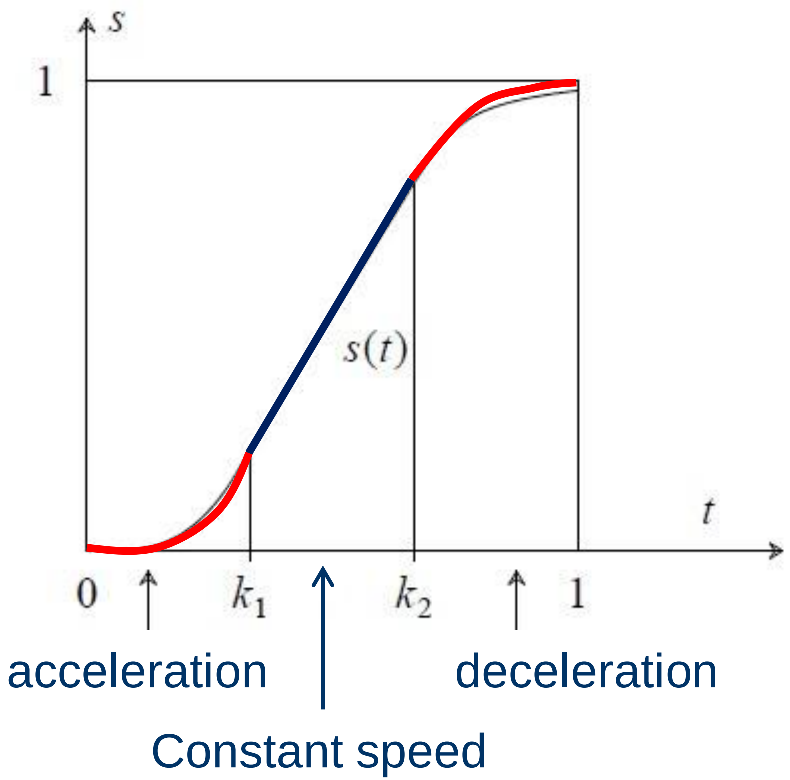
Movement of a pendulum





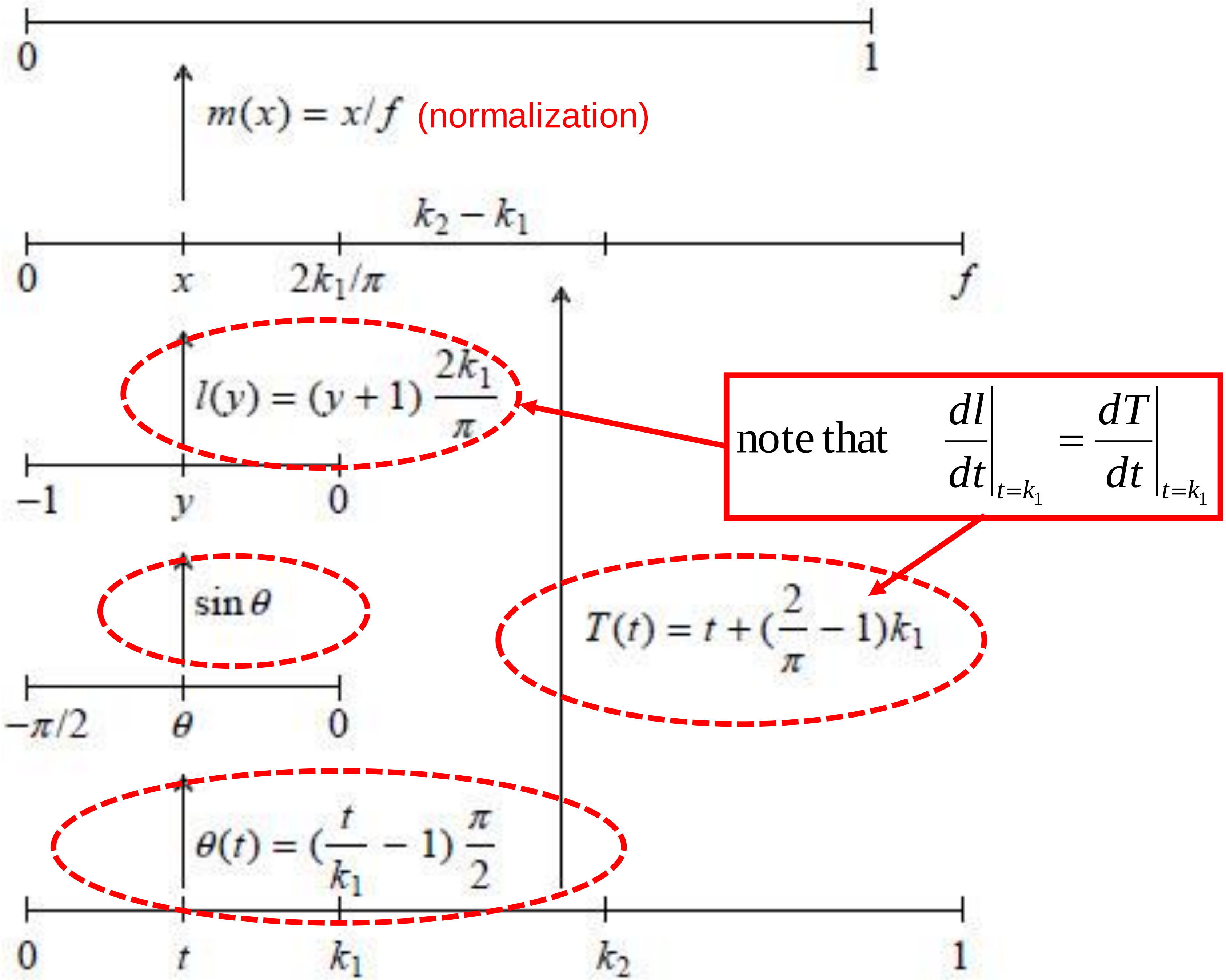
Speed Control

Using *Sinusoidal pieces*



How to design a distance-time function for this case?

(3 pieces: sine [- π/2, 0], straight line segment, sine [0, π/2])


0
1
$m(x) = x/f$ (normalization)
$k_2 - k_1$
0
x
$2k_1/\pi$
f
$l(y) = (y+1)\frac{2k_1}{\pi}$
note that $\left.\frac{dl}{dt}\right|_{t=k_1} = \left.\frac{dT}{dt}\right|_{t=k_1}$
-1
y
0
$\sin\theta$
$T(t) = t + (\frac{2}{\pi} - 1)k_1$
$-\pi/2$
θ
0
$\theta(t) = (\frac{t}{k_1} - 1)\frac{\pi}{2}$
0
t
k_1
k_2
1

Speed Control

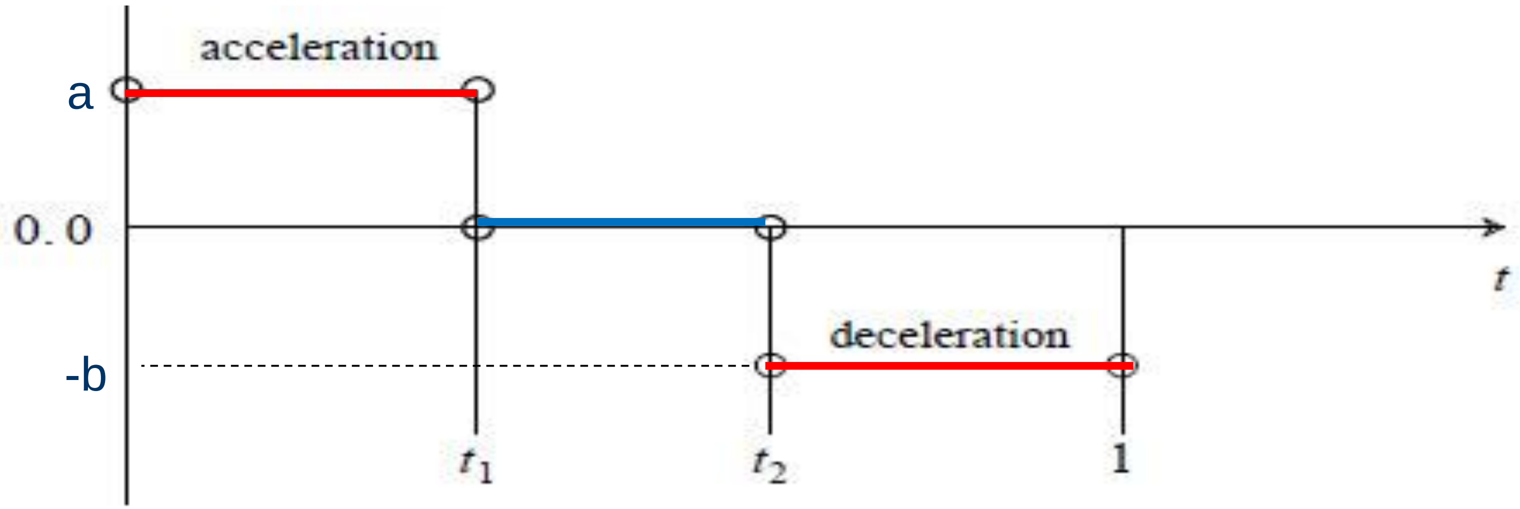
$$f=\frac{2(1-k_2)}{\pi}+k_2-k_1+\frac{2k_1}{\pi}$$

Hence,

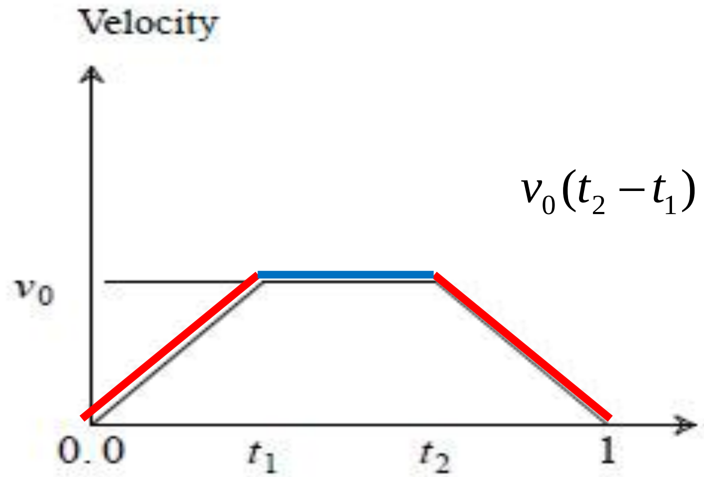
$$s(t)=\begin{cases}\frac{2k_1}{\pi}[\sin((\frac{t}{k_1}-1)\frac{\pi}{2})+1]/f, & 0\le t\le k_1\\ (t+\frac{2k_1}{\pi}-k_1)/f, & k_1\le t\le k_2\\ ?????, & k_2\le t\le 1\end{cases}$$

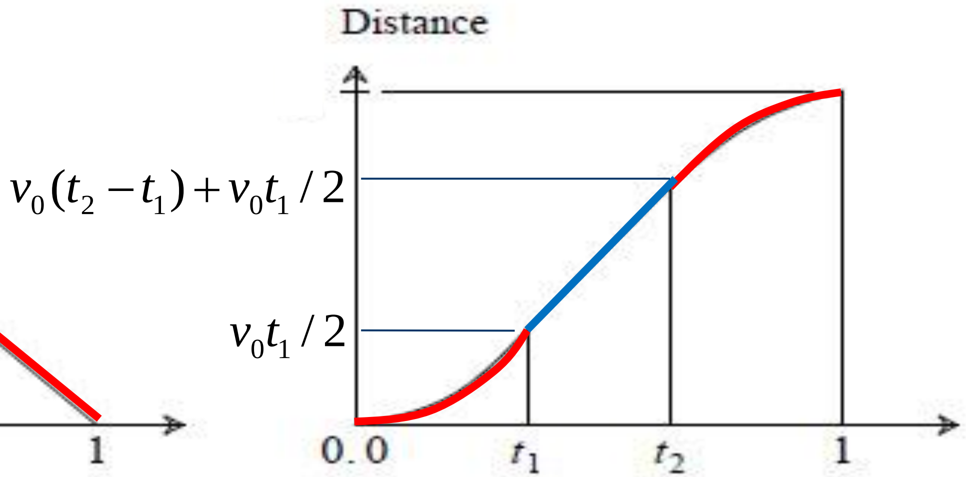
$$\left[\frac{2(1-k_2)}{\pi}\sin\left(\frac{\pi}{2(1-k_2)}(t-k_2)\right)+k_2-k_1+\frac{2k_1}{\pi}\right]/f$$

(iii) Constant acceleration



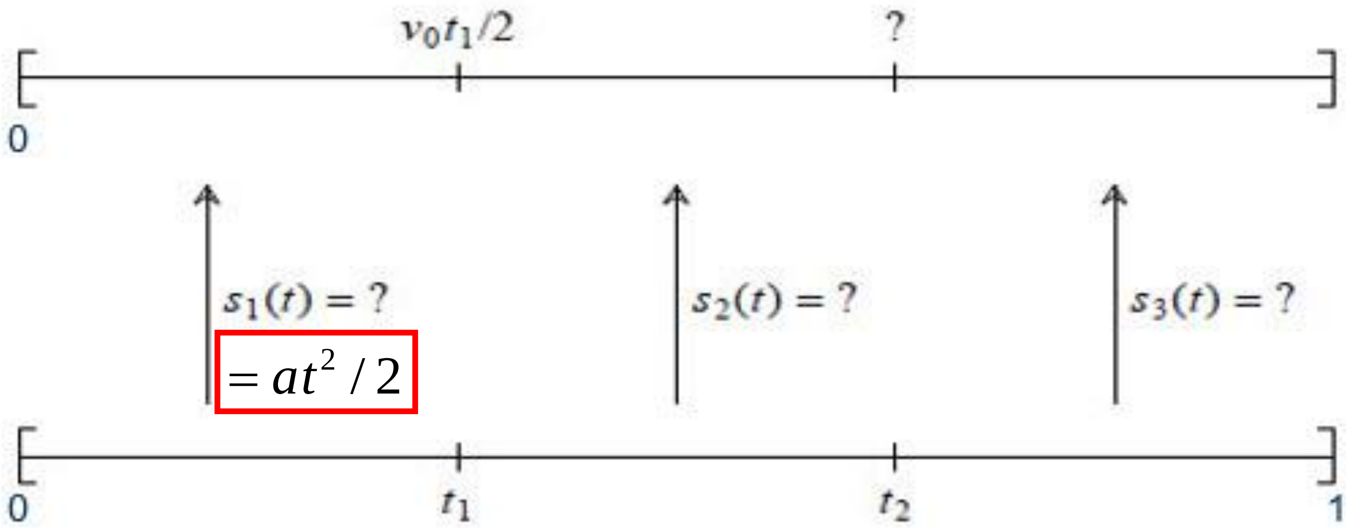
$$a = \frac{v_0}{t_1}$$





(iii) Constant acceleration

$v_0(t_2 - t_1) + v_0 t_1 / 2$

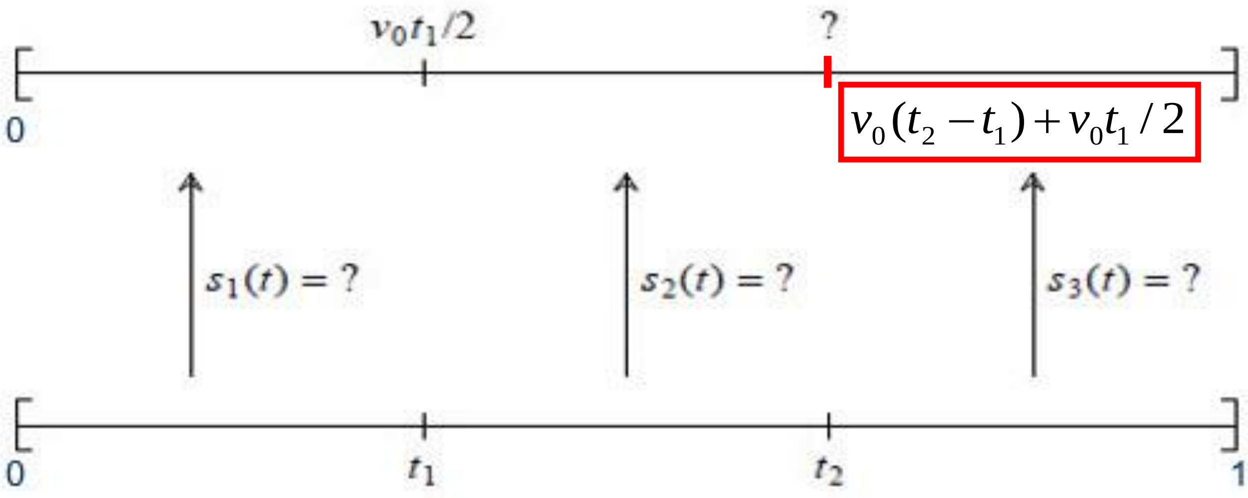


$$s_1(t) = \frac{v_0 t^2}{2t_1}, \qquad 0 \le t \le t_1$$

$$s_2(t) = v_0(t - t_1) + \frac{v_0 t_1}{2}, \qquad t_1 \le t \le t_2$$

$$s_3(t) = v_0(t_2 - t_1) + \frac{v_0 t_1}{2} + \left[v_0 - \frac{v_0(t - t_2)}{2(1 - t_2)}\right](t - t_2), \qquad t_2 \le t \le 1$$

Why?



Speed for $[t_2, 1]$: $-b(1-t_2) = v_0(1-t)/(1-t_2)$

$$\therefore \; s_3(t) = -\frac{v_0(1-t)^2}{2(1-t_2)} + d$$

$$d = \frac{v_0(1-t_2)}{2} + v_0(t_2 - t_1) + \frac{v_0 t_1}{2}$$

$$\therefore \; s_3(t) = -\frac{v_0(1-t)^2}{2(1-t_2)} + \frac{v_0(1-t_2)}{2} + v_0(t_2 - t_1) + \frac{v_0 t_1}{2}$$

$$= \mathrm{v}_0(t - t_2)\left[1 - \frac{t - t_2}{2(1-t_2)}\right] + v_0(t_2 - t_1) + \frac{v_0 t_1}{2}$$

End of Interpolation II