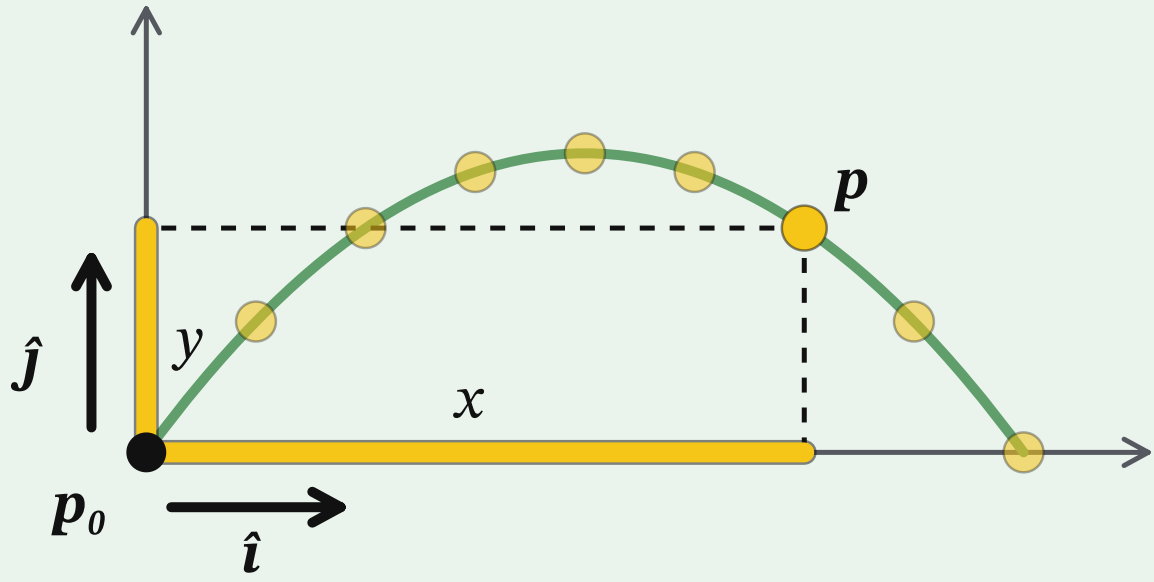
Accelab

Three Labs
The Mathematics.

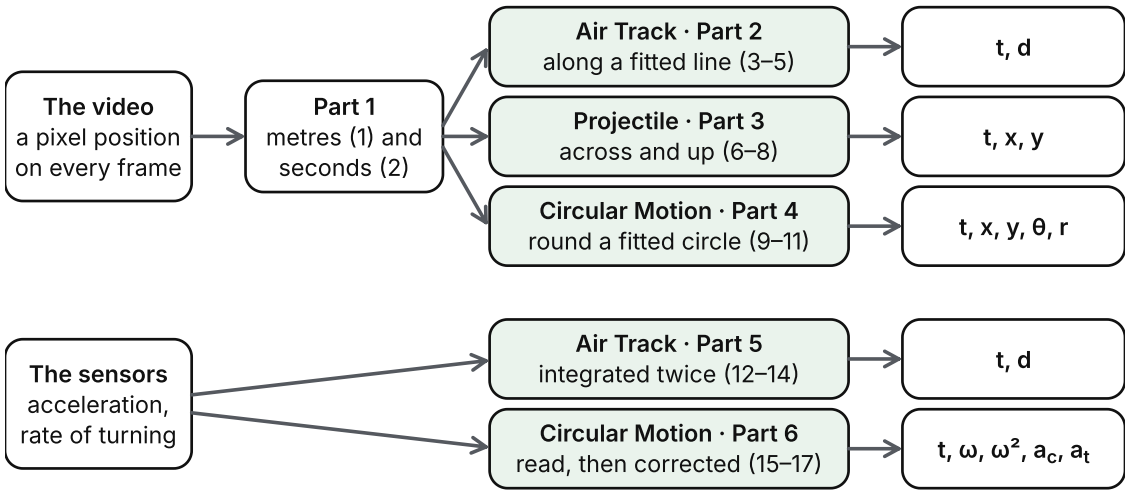


Myung Joon Kang
16 October 2026

Disclaimer. This document was generated by Claude Opus 5.5. I have reviewed every line.

Summary

Accelab collects the data for three labs: a cart on a track (**Air Track**), an object in flight (**Projectile**) and something going round (**Circular Motion**). It measures them in one of two ways. The **Camera** method films the motion, which gives a pixel position on every frame. The **Sensor** method rides along with it, which gives the iPhone's own acceleration and rate of turning. Physics wants metres, seconds and radians instead. This document explains the mathematics in between.



[Figure A] **The routes from measurement to data.** The top three are filmed, the bottom two sensed. Numbers in brackets are sections.

How the document is organised. Part 1 covers what every video shares: the scale and the clock. Parts 2 to 4 take each filmed experiment in turn. The cart in Part 2 keeps to a line, so one distance places it. The projectile in Part 3 follows a curve, so it needs two. The marker in Part 4 goes round a circle, so an angle about its centre places it. Part 5 and Part 6 are the Sensor method, which measures the cart and the turntable from the inside. The Toolbox holds the mathematics they lean on, and At a glance sets the five routes side by side.

In the code, the three filmed routes are functions in `TrackGeometry.swift`: `makeSplits`, `makePositionSplits` and `makeCircleSplits`. The sensed ones are in `MotionMeasuringManager.swift` and `RotationSampling.swift`, and the track's angle in `AngleManager.swift`. The worked numbers are illustrative, except in Section 8.4.

At a glance

For the air track the two methods give the same two columns, and Accelab steers towards the camera, which is easier to set up; Section 14.2 gives a further reason. For circular motion they measure different things, so the choice is the graph that is wanted.

Experiment	Camera	Sensor
Air Track	Measured along a line fitted to the points (Sections 3, 4)	The acceleration along the track, integrated twice (Sections 13, 14)
Projectile	Measured across and up the picture, held level (Sections 6, 7)	Not possible: the iPhone would have to fly
Circular Motion	An angle about the centre of a fitted circle (Sections 9, 10)	ω from the gyroscope, acceleration from the accelerometer (Sections 15, 16)
Before the run	A known length for the scale (Section 1); a track's angle (Section 12); a level phone for a projectile	A track's angle (Section 12); a still turntable, for the reading at rest (Section 15.1)

[Table A] **Where each experiment's numbers come from,** by method.

Run	Columns	Graph	What it gives
Air Track, either method	t, d	d against t	a, from $d = \frac{1}{2}at^2$ (Section 5)
Projectile	t, x, y	y against t	g, from $y = v_y t - \frac{1}{2}gt^2$ (Section 8)
Circular Motion, camera	t, x, y, θ, r	θ against t	ω, the gradient, and the period $2\pi / \omega$ (Section 11)
Circular Motion, sensor	t, ω, ω^2, a_c, a_t	a_c against ω^2	r, the gradient (Section 17)

[Table B] **The graph each run is for,** the one shown when its method is chosen. Each run's other graphs are in its sections.

Contents

Summary 2
At a glance 3

PART 1 · CAMERA · EVERY VIDEO 6
1 How many metres is a pixel? 7
2 How many seconds? 8

PART 2 · CAMERA · AIR TRACK 9
3 Which way does the track run? 10
3.1 The idea: the widest shadows 10
3.2 Turning "spread" into a number 10
3.3 Finding the angle where V is largest 11
3.4 Check with numbers 11
3.5 Why do it this way? 12
4 How far along it? 13
4.1 Check with numbers 13
5 Why the graph curves 14

PART 3 · CAMERA · PROJECTILE 15
6 Where is the object? 16
6.1 Two fixed directions instead of one fitted one 16
6.2 Check with numbers 17
6.3 Why do it this way? 17
7 Why must the phone be level? 18
7.1 What a tilt does 18
7.2 How much it matters 19
8 Why does the flight curve? 20
8.1 Against time 20
8.2 Against each other: the path 20
8.3 Getting g from the data 21
8.4 Check with a real run 22

Contents, continued

PART 4 · CAMERA · CIRCULAR MOTION 23

9 Where is the centre? 24

9.1 A circle as an equation 24

9.2 The radius 25

9.3 Check with numbers 25

9.4 Why do it this way? 25

10 How far round? 26

10.1 Counting whole turns 26

11 What the graphs show 27

11.1 From ω and r to the acceleration 27

11.2 Why film from straight above? 28

PART 5 · SENSOR · AIR TRACK 29

12 How steep is the track? 30

12.1 How much the angle matters 30

13 What the accelerometer feels 31

14 From acceleration to distance 32

14.1 Keeping a still cart still 33

14.2 How errors grow 33

PART 6 · SENSOR · CIRCULAR MOTION 34

15 What the sensors read 35

15.1 The reading at rest 36

15.2 Averaging into samples 36

16 A turntable that isn't level 37

16.1 Fitting the leak 38

17 The graphs 39

17.1 A phone not pointing at the centre 40

17.2 The two methods compared 40

18 TOOLBOX 41

18.1 atan2 41

18.2 Double-angle identities 42

18.3 Rotating the axes 43

18.4 Small angles 43

18.5 Least squares 44

Part 1 · Camera

Every video
Pixels to metres, frames to seconds.

Whatever is filmed, a video measures in its own units: lengths in pixels and time in frames. Turning those into metres and seconds works in exactly the same way for a cart, an object in flight and a marker going round, so it comes first. Parts 2, 3 and 4 then show where each experiment's lengths in pixels come from. The Sensor method, in Parts 5 and 6, needs no scale at all: it never sees a picture.

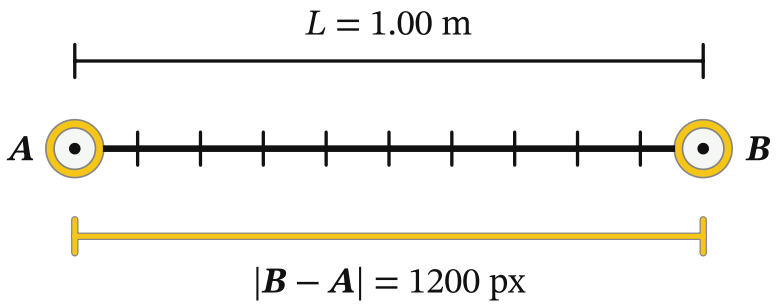
In this part

1 How many metres is a pixel?	7
2 How many seconds?	8

1 How many metres is a pixel?

A pixel has no physical size. The same metre covers few pixels when the camera is far away and many when it is close, so the video needs something of known length: the ruler.

$$k = L / |\boldsymbol{B} - \boldsymbol{A}| \qquad d = k\, s$$



[Figure 1.A] **Calibrating the scale.** The ruler's real length L above, and the same span in pixels below.

L is the ruler's real length, and $|\boldsymbol{B} - \boldsymbol{A}| = \sqrt{((x_B - x_A)^2 + (y_B - y_A)^2)}$ is the pixel distance between its two marked ends. Pythagoras is needed because the ruler may be tilted in the picture.

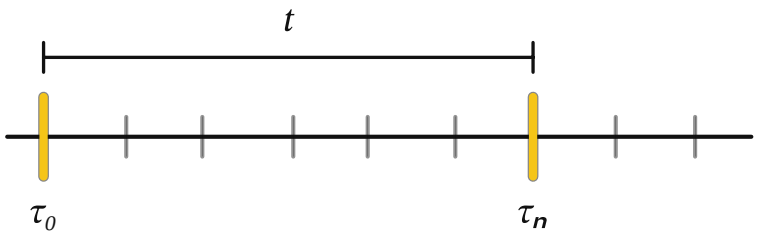
With 1 m spanning 1200 px, k = 1 / 1200 m per pixel, and a length of s = 600 px in the picture becomes d = 0.50 m.

The hidden assumption. One k is used for the whole picture. That only holds if the camera faces the motion square-on and the ruler is as far from it as the cart, the object in flight or the marker, which is what the app's filming checklist protects.

2 How many seconds?

Time comes from each frame's own timestamp in the video file, measured from the first tracked frame:

$$t = \tau_n - \tau_0$$



[Figure 2.A] **Frame timestamps.** Frames are not perfectly evenly spaced; t is the difference between two stamped times.

τ_n is the time stamped on frame n, and τ_0 is the first tracked frame's. Subtracting makes the start $t = 0$, just as every distance is measured from the first tracked point.

Why not frame number ÷ 60? iPhone video frames are not guaranteed to be exactly $1 / 60$ s apart. Counting frames would put that error straight into t.

Timing errors count twice. From $d = \frac{1}{2}at^2$, $a = 2d / t^2$, so a 1% error in t becomes about a 2% error in a.

The sensors keep time the same way. Every reading from the accelerometer and the gyroscope carries its own timestamp too, and Parts 5 and 6 subtract the time recording began. They never assume the readings arrived exactly on schedule.

Part 2 · Camera

Air Track

A cart on a track: one distance.

A cart keeps to a straight line, so a single number places it: how far along the track it has gone. The track is slanted in the picture, so the app first finds which way it runs, then measures along it. Part 1 turns that distance into metres and its frame into seconds. Part 5 measures the same run with the sensors instead.

In this part

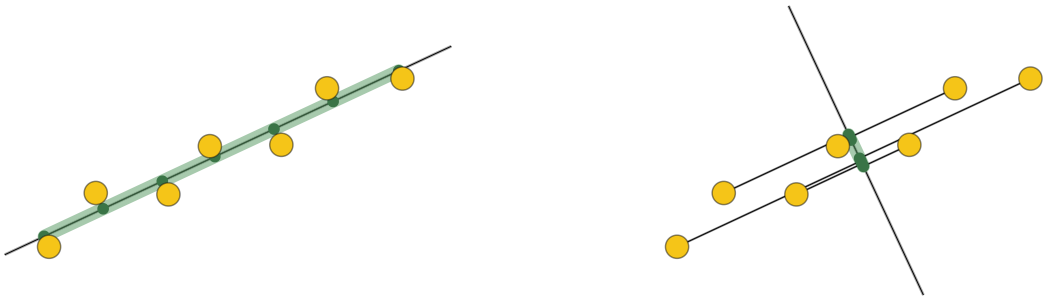
3 Which way does the track run?	10
4 How far along it?	13
5 Why the graph curves	14

3 Which way does the track run?

The camera is never perfectly level and the track is tilted, so on screen the track is a slanted line. Before measuring distance along it, the app has to know which way it points. It does this by finding the direction in which the tracked points are most spread out.

3.1 The idea: the widest shadows

Pick any direction and drop every point straight onto a line in that direction, like casting shadows. If the direction matches the track, the shadows spread out wide. If it runs across the track, they bunch together. So the track's direction is the one where the shadows are most spread out.



[Figure 3.1.A] **Shadows along and across the track.** Left: a line along the track, where the shadows (dark dots) spread wide. Right: a line across the track, where they bunch up.

3.2 Turning "spread" into a number

First subtract the average position, so the points are centred on the origin: $a = x - \bar{x}$ and $b = y - \bar{y}$. Only the direction matters, not where the line sits.

A direction at angle φ is the unit vector $(\cos\varphi, \sin\varphi)$. The shadow of the point (a, b) on it is a dot product:

$$\text{shadow} = a\cos\varphi + b\sin\varphi$$

Shadows can be positive or negative and would cancel if simply added, so square them first, then add. Call the total V:

$$V(\varphi) = \Sigma\,(a\cos\varphi + b\sin\varphi)^2 = S_{xx}\cos^2\varphi + 2S_{xy}\sin\varphi\cos\varphi + S_{yy}\sin^2\varphi$$

$$S_{xx} = \Sigma\,a^2 \qquad S_{xy} = \Sigma\,ab \qquad S_{yy} = \Sigma\,b^2$$

The three sums are not a special technique. They are what is left when the bracket is expanded: how far the points spread in x, in y, and how much x and y move together. They are computed once; after that only φ changes.

3.3 Finding the angle where V is largest

The double-angle identities (Section 18.2) turn the squares and the product into single terms in 2φ:

$$V(\varphi) = \text{constant} + A\cos 2\varphi + B\sin 2\varphi$$
$$A = (S_{xx} - S_{yy})/2 \qquad B = S_{xy}$$

The part that changes is itself a dot product of two vectors:

$$A\cos 2\varphi + B\sin 2\varphi = (A, B)\cdot(\cos 2\varphi, \sin 2\varphi)$$

$(\cos 2\varphi, \sin 2\varphi)$ is a unit vector pointing at angle 2φ. A dot product is largest when the two vectors point the same way, so 2φ must be the angle of the vector (A, B). Doubling that vector does not change its angle, which gives the formula in the code:

$$\varphi = \tfrac{1}{2}\,\text{atan2}(2S_{xy}, S_{xx} - S_{yy}) \qquad \boldsymbol{u} = (\cos\varphi, \sin\varphi)$$

The 2 and the ½ both come from the double-angle identities. $\boldsymbol{u}$ has length 1 because $\cos^2\varphi + \sin^2\varphi = 1$, which matters in Section 4. The function atan2 is explained in Section 18.1.

3.4 Check with numbers

Take three points on a line of gradient ½: (−2, −1), (0, 0) and (2, 1). Their average is already (0, 0).

$$S_{xx} = 8 \qquad S_{yy} = 2 \qquad S_{xy} = 4$$

$$\varphi = \tfrac{1}{2}\,\text{atan2}(8, 6) = \tfrac{1}{2}\times 53.13^\circ = 26.57^\circ$$

The line's real angle is $\tan^{-1}(\tfrac{1}{2}) = 26.57^\circ$. They agree.

3.5 Why do it this way?

Why all the points, not just the first and last? Every tracked point is a few pixels off. With two points, those two errors decide the direction. With all of them, the errors largely cancel.

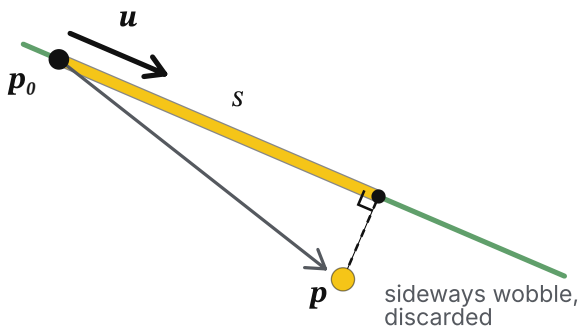
Why not an ordinary line of best fit? The usual $y = mx + c$ fit assumes x is exact and only y has error, and its gradient blows up when the track looks vertical. Here x and y are both uncertain, so the fit uses perpendicular distances and works at any tilt.

Why are widest shadows and smallest perpendicular distances the same thing? For each point, Pythagoras gives $(\text{distance from centre})^2 = (\text{shadow})^2 + (\text{perpendicular distance})^2$. The left side is fixed by where the point is, so making the shadows as large as possible makes the perpendicular distances as small as possible.

4 How far along it?

$\boldsymbol{p}_0$ is the first tracked point, the start. $\boldsymbol{p} - \boldsymbol{p}_0$ is the cart's displacement from the start, so the first point is automatically at zero. Its distance along the track is a dot product with the unit vector from Section 3:

$$s = (\boldsymbol{p} - \boldsymbol{p}_0) \cdot \boldsymbol{u}$$



[Figure 4.A] **Projecting a point onto the track.** s is measured along the track from the start.

That displacement has a part along the track and a small sideways part from tracking error. By definition $\boldsymbol{v} \cdot \boldsymbol{u} = |\boldsymbol{v}|\,|\boldsymbol{u}| \cos\alpha$, and because $|\boldsymbol{u}| = 1$ this is just $|\boldsymbol{v}| \cos\alpha$: the length of the shadow of $\boldsymbol{v}$ on the track, already in pixels. The sideways part is perpendicular to $\boldsymbol{u}$, so its dot product is zero and it drops out.

Why not just the length $|\boldsymbol{p} - \boldsymbol{p}_0|$? That would count the sideways wobble as distance, so every value would come out slightly too large.

Which way is positive? A line has two directions, $\boldsymbol{u}$ and $-\boldsymbol{u}$, and Section 3 returns one arbitrarily. If the last point comes out negative, every s is multiplied by −1: the cart only rolls downhill, so the last point marks down-slope.

4.1 Check with numbers

With $\boldsymbol{u} = (0.8, 0.6)$ and $\boldsymbol{p} - \boldsymbol{p}_0 = (468, 376)$:

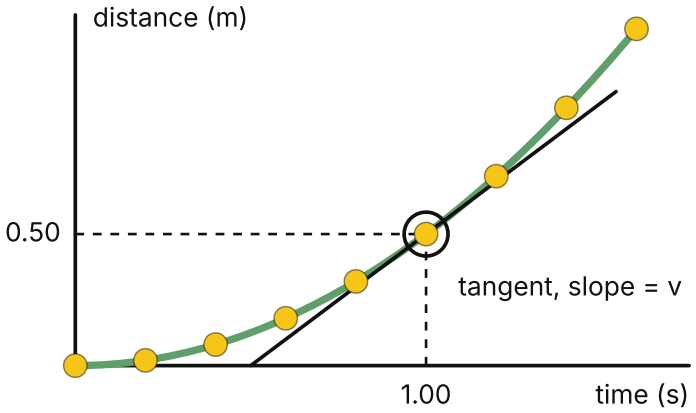
$$s = 468 \times 0.8 + 376 \times 0.6 = 374.4 + 225.6 = 600 \text{ px}$$

The displacement is really $600\boldsymbol{u} = (480, 360)$ plus a wobble of $(-12, 16)$. The wobble dotted with $\boldsymbol{u}$ is $-9.6 + 9.6 = 0$, so it has no effect.

5 Why the graph curves

Each point is now one CSV row of time and distance. On a distance–time graph the slope is the speed. The cart starts at rest and speeds up, so the graph starts flat and gets steeper.

$$a = g \sin\theta \qquad v = at \qquad d = \tfrac{1}{2}at^2$$



[Figure 5.A] **Distance against time.** Points at equal time steps on a parabola. The ringed point is 0.50 m at 1.00 s, and the straight line is the tangent there.

The slope of the graph at any instant is the derivative of distance with respect to time, which is the speed. The acceleration is the derivative of speed in the same way:

$$v = \frac{\mathrm{d}d}{\mathrm{d}t} = at \qquad a = \frac{\mathrm{d}v}{\mathrm{d}t}$$

The tangent line in the figure shows this slope at the ringed point. There t = 1.00 s, so its slope is 1 × 1.00 = 1.00 m/s. Earlier points have flatter tangents and later points steeper ones, which is what a changing speed looks like.

The only unbalanced force along the track is a constant component of gravity, so the acceleration is constant, the speed grows in proportion to time, and the distance grows with time squared: a parabola. The example point fits. With a = 1 m/s^2, t = 1.00 s gives d = 0.50 m, the row `1.0000,0.50000`.

To get a from the data, plot v against t (slope a) or d against t^2 (slope ½a), then compare with $g \sin\theta$. Accelab stops at exporting d and t; that analysis is the student's part of the lab.

Part 3 · Camera

Projectile

An object in flight: two distances.

An object in flight follows a curve, so there is no direction to fit and one number cannot place it. The app measures how far it has gone across and how far up, along the picture's own two edges, which is why the phone has to be level. Part 1's scale and clock are used unchanged.

In this part

6 Where is the object?	16
7 Why must the phone be level?	18
8 Why does the flight curve?	20

6 Where is the object?

An object in flight does not keep to a line, so Sections 3 and 4 cannot be used as they are. There is no track direction to find, and the direction of widest spread of a curved path is only a slanted compromise that means nothing physical. What physics wants instead is two numbers on each frame: how far the object has gone across, x, and how far up, y, both measured from where it started.

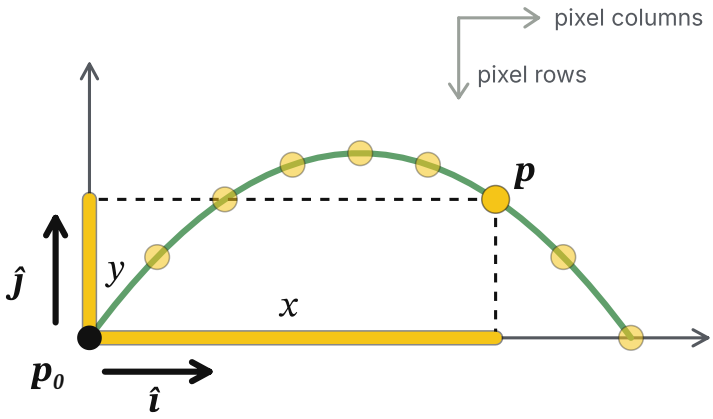
6.1 Two fixed directions instead of one fitted one

The picture has two directions built in: along its width and up its height. If the phone is level, these are the real horizontal and vertical. As unit vectors, where σ is either 1 or −1:

$$\hat{\boldsymbol{\imath}} = (\sigma, 0) \qquad \hat{\boldsymbol{\jmath}} = (0, -1)$$

Then each position is the dot product of Section 4, once with each, and the scale k from Section 1 turns pixels into metres:

$$x = k\,(\boldsymbol{p} - \boldsymbol{p}_0) \cdot \hat{\boldsymbol{\imath}} \qquad y = k\,(\boldsymbol{p} - \boldsymbol{p}_0) \cdot \hat{\boldsymbol{\jmath}}$$



[Figure 6.1.A] **Measuring across and up.** x and y are the two parts of the displacement from the start $\boldsymbol{p}_0$ to the object $\boldsymbol{p}$.

$\hat{\boldsymbol{\imath}}$ and $\hat{\boldsymbol{\jmath}}$ are perpendicular and each has length 1, so x and y are the two **components** of the displacement. Together they hold everything it does. For the cart, the part across the track was thrown away as wobble; here that part is the height, which is half the answer.

Why the −1 in $\hat{\boldsymbol{\jmath}}$? On a screen, pixel rows are numbered from the top, so the picture's own y grows downwards. Height grows upwards. $\hat{\boldsymbol{\jmath}}$ points up the picture, which turns "fewer rows down" into "more height".

What is σ? It is +1 if the last tracked point is to the right of the first and −1 if it is to the left, so x counts up in the direction the object travels, whichever way it was thrown. This is the cart's rule again: distance counts up down-slope. Height is never given a sign this way. Up is up wherever the object lands; if y took its sign from the motion, an object pushed off a table, which only ever goes down, would come out with gravity pointing upwards.

6.2 Check with numbers

Use the scale from Section 1, $k = 1 / 1200$ m per pixel. An object is thrown to the right, so $\sigma = 1$. Half a second later it is 1200 px to the right of its start and 330 px higher up the picture, which is 330 rows *fewer*: $\boldsymbol{p} - \boldsymbol{p}_0 = (1200, -330)$ px.

$$x = (1200 \times 1 + (-330) \times 0) / 1200 = 1.000 \text{ m}$$

$$y = (1200 \times 0 + (-330) \times (-1)) / 1200 = 0.275 \text{ m}$$

Thrown to the left instead, the displacement is (−1200, −330) px and $\sigma = -1$. Then $x = (-1200 \times -1) / 1200 = 1.000$ m again, and y is untouched. The CSV row is `0.5000,1.00000,0.27500` either way.

6.3 Why do it this way?

Why is the first point the origin? Subtracting $\boldsymbol{p}_0$ makes the first row (0, 0) at $t = 0$. The equations of motion then need no starting position, only a starting velocity (Section 8).

Why one scale for both directions? A pixel is as tall as it is wide, so the k measured along the ruler holds for heights too. It rests on the same hidden assumption as before: the camera square-on, and the ruler as far away as the object.

Why not let the student place the axes? Some video-analysis tools do. Fixing them to the picture means nothing to set up and nothing to get wrong on screen, at the price of one rule while filming: keep the phone level. Section 7 shows what that rule is worth.

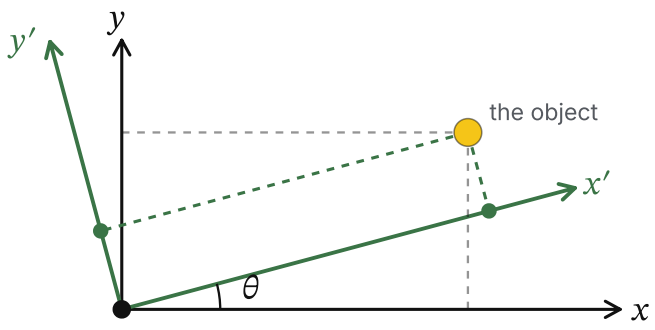
7 Why must the phone be level?

The cart's direction is found from the cart's own points. Tilt the phone and the fitted $\boldsymbol{u}$ tilts with it, so nothing is lost. A projectile has no such protection: its axes are the picture's edges, and if the phone is rolled by an angle θ, those edges are not the real horizontal and vertical.

7.1 What a tilt does

Call the true coordinates (x, y) and what the tilted picture measures (x', y'). Turning the axes by θ mixes the two (Section 18.3):

$$x' = x\cos\theta + y\sin\theta \qquad y' = y\cos\theta - x\sin\theta$$



[Figure 7.1.A] **A tilted picture.** Its axes (green) are turned by θ from the true ones, so it reads a slightly different x and y for the same object.

Put in the true motion from Section 8, $x = v_x t$ and $y = v_y t - \tfrac{1}{2}gt^2$:

$$y' = (v_y\cos\theta - v_x\sin\theta)\,t - \tfrac{1}{2}(g\cos\theta)\,t^2$$
$$x' = (v_x\cos\theta + v_y\sin\theta)\,t - \tfrac{1}{2}(g\sin\theta)\,t^2$$

Two things have happened. In y', g has become $g\cos\theta$. In x', a t^2 term has appeared from nowhere: the object seems to accelerate sideways at $g\sin\theta$.

7.2 How much it matters

The size of each effect depends only on the tilt.

Tilt θ	Measured $g \cos\theta$	Error in g	Sideways $g \sin\theta$
1°	9.809 m/s^2	0.02% low	0.17 m/s^2
2°	9.804 m/s^2	0.06% low	0.34 m/s^2
5°	9.773 m/s^2	0.4% low	0.85 m/s^2
10°	9.661 m/s^2	1.5% low	1.70 m/s^2

[Table 7.2.A] **The effect of a tilted phone,** taking g = 9.81 m/s^2.

For a small tilt cos θ stays very close to 1 while sin θ grows at once (Section 18.4). So g from the y–t graph survives a careless phone, but the x–t graph, which should be a straight line, bends. **A curved x–t graph is the sign of a tilted phone.**

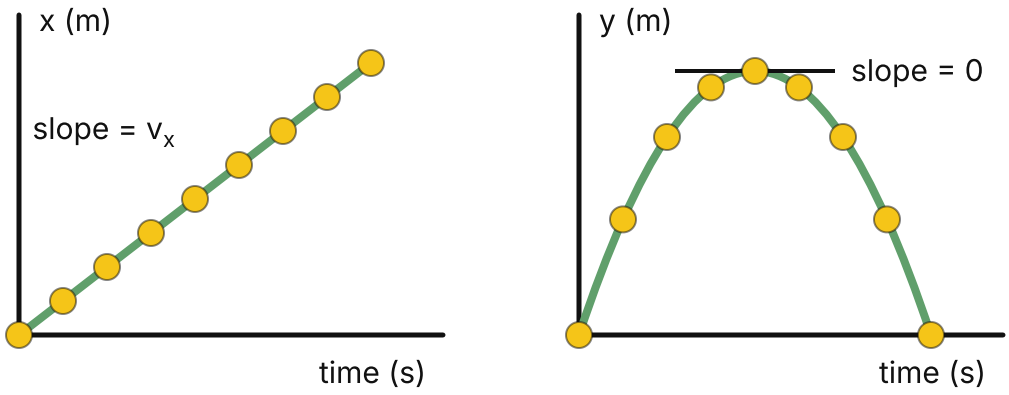
8 Why does the flight curve?

Each point is now one CSV row of time, x and y. Two different graphs come out of it.

8.1 Against time

Once the object has left the hand, the only force on it is gravity, straight down (ignoring the air). Nothing changes its sideways speed v_x, and its upward speed, which starts at v_y, falls by g every second:

$$x = v_x t \qquad y = v_y t - \tfrac{1}{2} g t^2$$



[Figure 8.1.A] **Against time.** Points at equal time steps. x climbs a straight line; y climbs, stops and falls along a parabola.

The y–t graph is the cart's graph from Section 5 turned upside down, with g in place of $g \sin\theta$: a track tilted to 90° would simply be a drop. Its slope is the upward speed, $dy / dt = v_y - gt$. At the top that slope is zero, which gives the time and height of the peak:

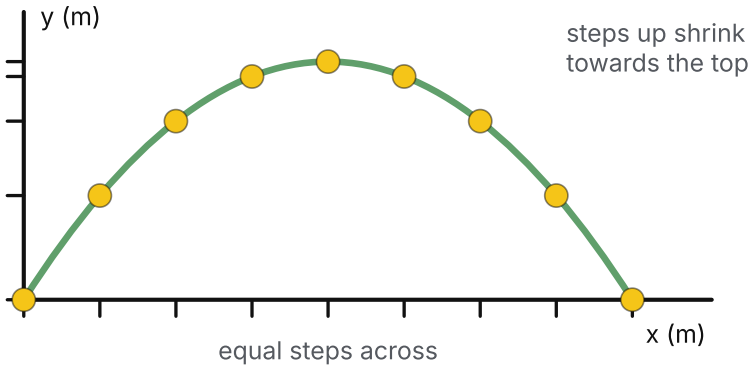
$$t_{top} = v_y / g \qquad y_{top} = v_y^2 / 2g$$

8.2 Against each other: the path

From the first equation $t = x / v_x$. Putting that into the second removes time altogether:

$$y = (v_y / v_x)\, x - (g / 2v_x^2)\, x^2$$

This is a quadratic in x, so the path through the air is itself a parabola, in space this time rather than in time. It is the yellow trail on the video, and the curve the Analysing screen draws.



[Figure 8.2.A] **Height against distance.** The same points as Figure 8.1.A. That screen uses one scale for both axes, the smaller of width ÷ range of x and height ÷ range of y, so the path keeps its true shape.

8.3 Getting g from the data

Accelab stops at exporting t, x and y; finding g is the student's part of the lab. Three ways, from the same rows:

Fit the parabola. Fit $y = bt - ct^2$ to the y–t graph. Then $g = 2c$.

Plot the speed. Between two rows the upward speed is about $\Delta y / \Delta t$. Plotted against time it is a straight line of slope $-g$.

Difference twice. At equal time steps each Δy is smaller than the one before by $g\,\Delta t^2$. At 60 frames a second that is $9.81 / 3600 \approx 2.7$ mm, which in most shots is only a pixel or two, about the size of the tracking error itself, so this way is the noisiest. A fit uses every point, and the errors largely cancel, as in Section 3.5.

8.4 Check with a real run

Unlike the other numbers here, these were measured. A generated clip of an object launched at (2.4, 3.3) m/s under g = 9.81 m/s^2, drawn at 600 px per metre and 60 frames a second beside a 1 m ruler, was put through the app: imported, calibrated on the ruler, tracked automatically over 46 frames and exported.

Clip	g from $y(t)$	v_x from $x(t)$	Largest error
Thrown to the right	9.798 m/s^2	2.399 m/s	2.1 mm
Thrown to the left	9.814 m/s^2	2.399 m/s	3.1 mm

[Table 8.4.A] **Recovered from the exported CSV.** The true values are 9.81 m/s^2 and 2.4 m/s; 1 px is 1.7 mm.

A generated clip is the kindest possible test: a sharp object, a level camera and no perspective. Real footage will do worse, mostly through blur and a phone that is not quite square-on.

Part 4 · Camera

Circular Motion

A marker going round: one angle.

A marker on a turntable, filmed from straight above, keeps to a circle. As with the cart's track, nobody tells the app where that circle is, so it first fits one to the tracked points, then places every point by its angle about the centre. Part 1's scale and clock are used unchanged, though the angle needs no scale at all: a radian is one length divided by another.

In this part

9 Where is the centre?	24
10 How far round?	26
11 What the graphs show	27

9 Where is the centre?

Every number in this part is measured from the centre of the circle, but nobody marks the centre: it may be hidden under the turntable's spindle, or not be on anything at all. So the app finds it from the marker's own points, much as Section 3 found the track's direction from the cart's.

9.1 A circle as an equation

A circle with centre (a, b) and radius r is every point whose distance from the centre is r. Multiplying out and collecting the unknowns:

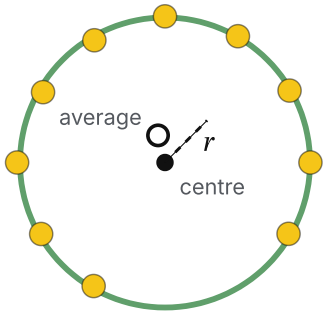
$$(x-a)^2+(y-b)^2=r^2 \quad\Longleftrightarrow\quad x^2+y^2+Dx+Ey+F=0$$

$$D=-2a \qquad E=-2b \qquad F=a^2+b^2-r^2$$

The point of the rewrite is that D, E and F appear only to the first power: the equation is a circle, but it is **linear in its unknowns**. A tracked point never lies exactly on the circle, so its left side is not quite zero; call that misfit e. Least squares (Section 18.5) picks the D, E and F that make Σe^2 smallest, and setting its three derivatives to zero gives three linear equations, with $z = x^2 + y^2$ for each point:

$$\begin{pmatrix} \Sigma x^2 & \Sigma xy & \Sigma x \\ \Sigma xy & \Sigma y^2 & \Sigma y \\ \Sigma x & \Sigma y & n \end{pmatrix} \begin{pmatrix} D \\ E \\ F \end{pmatrix} = \begin{pmatrix} -\Sigma xz \\ -\Sigma yz \\ -\Sigma z \end{pmatrix}$$

Solving them is ordinary elimination. The centre is then $(-D/2, -E/2)$, straight from the definitions of D and E.



[Figure 9.1.A] **Fitting a circle.** The points cover three quarters of a turn. The fitted centre (black) is the circle's; their plain average (ringed) is pulled towards the side the points cover.

9.2 The radius

The definition of F gives $r^2 = D^2 / 4 + E^2 / 4 - F$, but there is a clearer view of it. Each point's misfit is exactly the square of its distance d from the fitted centre, less r^2, and the third equation (the derivative with respect to F) says the misfits add to zero:

$$\Sigma\,(d^2 - r^2) = 0 \quad \Longrightarrow \quad r = \sqrt{(\,\Sigma d^2 / n\,)}$$

So the fitted radius is the **root mean square** of the points' distances from the centre. It is the Radius shown when the run finishes. The CSV's radius column keeps each point's own d instead, so a wobble in the circle stays visible.

9.3 Check with numbers

Take three points on a circle of radius 5 about the origin: (5, 0), (0, 5) and (−5, 0). Every z is 25.

$$\Sigma x^2 = 50 \qquad \Sigma y^2 = 25 \qquad \Sigma xy = 0 \qquad \Sigma x = 0 \qquad \Sigma y = 5 \qquad n = 3$$
$$\Sigma xz = 0 \qquad \Sigma yz = 125 \qquad \Sigma z = 75$$

The first equation is $50D = 0$. The other two are $25E + 5F = -125$ and $5E + 3F = -75$, so $E = 0$ and $F = -25$. The centre is (0, 0) and $r = \sqrt{25} = 5$, as it should be. Their plain average, (0, 5 / 3), is a third of the radius off.

9.4 Why do it this way?

Why not just average the points? The average is the centre only when the points are spread evenly over whole turns. A run that stops part-way round, or a marker that slows down, crowds one side, and the average drifts towards it.

Why this misfit, and not the gap to the circle? The true gap is $d - r$, and minimising it means guessing and improving, because the unknowns sit inside a square root. The misfit used here is $d^2 - r^2 = (d - r)(d + r) \approx 2r(d - r)$ for points near the circle: the same gap scaled by $2r$, solved in one step. Its known weakness is a radius too small, sometimes badly, when noisy points cover only a short arc. A few full turns avoid that.

Why work from the average position? Pixel positions run to a thousand or more, so the sums of z run to billions and lose digits. Subtracting the average position first moves the origin, not the circle.

10 How far round?

With the centre ***c*** known, each point ***p*** is measured from it in metres, with the picture's y turned round to point up, as in Section 6:

$$x = k\,(p_x - c_x) \qquad y = k\,(c_y - p_y)$$

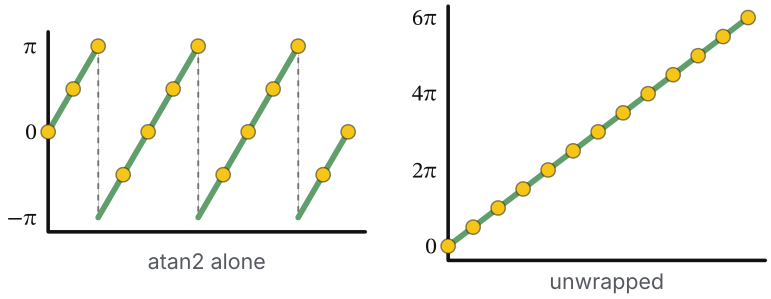
Its direction from the centre is atan2(y, x), which reads the signs of both and so knows the quadrant (Section 18.1). The answer lies between −π and π.

10.1 Counting whole turns

atan2 says which way, not how far round. Each time the marker crosses the negative x-axis its answer jumps by 2π, between π and −π, and the angle–time graph becomes a sawtooth (Figure 10.1.A). The app undoes each jump by comparing every angle with the one before and taking off whole turns:

$$\theta_n = \theta_{raw} - 2\pi \times \text{round}(\,(\theta_{raw} - \theta_{n-1})/2\pi\,)$$

The rounding counts how many whole turns the raw step is from zero. A real step between frames is far less than half a turn, so anything bigger is a wrap. At 60 frames a second that holds unless the marker goes round more than 30 times a second.



[Figure 10.1.A] **Three steady turns, two ways.** Left: atan2 alone, which wraps at ±π. Right: the same points with the wraps taken out, one straight line.

Check with numbers. The last angle was 3.10 rad and the next raw angle is −3.10 rad. Then (−3.10 − 3.10) / 2π = −0.987, which rounds to −1, so θ = −3.10 + 2π = 3.183 rad: a step of 0.083 rad onwards, not 6.20 rad back.

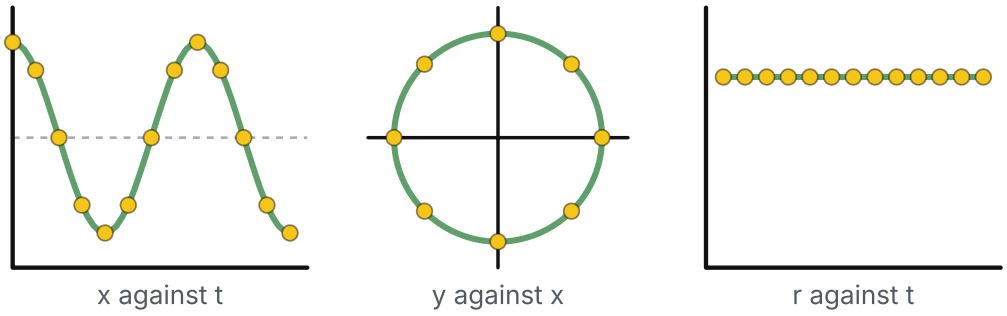
Last, the angles are measured from the first point and given a sign, $\theta = \sigma(\theta_n - \theta_0)$, with σ = −1 if they shrank. So θ starts at 0 and climbs through 2π, 4π, … in the direction of travel, whichever way the marker went round: the cart's rule again (Section 4). The Turns shown at the end are the last θ / 2π.

11 What the graphs show

Each point is now one CSV row of time, x, y, θ and r. The lab's own graph is θ against t:

$$\theta = \omega t \qquad T = 2\pi / \omega$$

At a steady spin the angle grows in proportion to time, so the graph is a straight line through the origin (Figure 10.1.A, right) and its gradient is the angular velocity ω in rad/s. One turn is 2π, so a turn takes $T = 2\pi / \omega$. A turntable that slows gives a line that bends over, and its gradient at any instant is ω at that instant, just as the tangent in Section 5 gave the cart's speed.



[Figure 11.A] **The other three graphs of a steady spin,** with points at equal time steps.

x against t is a wave of amplitude r and period T: seen edge-on, circular motion is simple harmonic motion. y against x is the circle itself, the curve the Analysing screen draws to one scale on both axes so it stays round. r against t is flat if the radius stayed the same; a slope or a wobble means a marker that slipped, or a camera that was not looking straight down (Section 11.2).

11.1 From ω and r to the acceleration

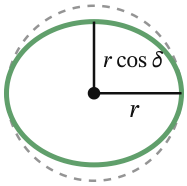
The video gives the two things circular motion is built from, ω from the gradient and r from the fit, and everything else follows:

$$v = \omega r \qquad a_c = \omega^2 r = v^2 / r$$

A marker 0.20 m from the centre goes round 3 times in 2.0 s. Then $\omega = 3 \times 2\pi / 2.0 = 9.42$ rad/s and $T = 0.667$ s, so $v = 9.42 \times 0.20 = 1.88$ m/s and $a_c = 9.42^2 \times 0.20 = 17.8$ m/s^2, nearly twice g. Part 6 measures that acceleration directly instead, which is how the two methods can check each other (Section 17.2).

11.2 Why film from straight above?

The phone can be off in two ways. **Turned** about its own line of sight, it only turns the picture of the circle, which changes nothing: a turned circle is the same circle, and the angle is measured from the first point anyway. **Tipped** by an angle δ away from straight down, it sees the circle at a slant. Lengths in the direction of the tip shrink by cos δ, and the circle becomes an ellipse.



[Figure 11.2.A] **The circle a tipped camera sees,** drawn for a tip of 35° so that it shows.

Small angles (Section 18.4) say how little a real tip matters: the shrinking is $1 - \cos\delta \approx \frac{1}{2}\delta^2$, which is tiny for a small δ.

Tip δ	One direction short by	Fitted radius short by	θ gained per turn
2°	0.06%	0.03%	2π exactly
5°	0.38%	0.19%	2π exactly
10°	1.5%	0.76%	2π exactly

[Table 11.2.A] **The effect of a tipped camera** on a circle filmed from above.

The fitted radius loses only about half the shrinking: one direction is squeezed and the other is not, and a root mean square (Section 9.2) of the two gives $r\sqrt{((1 + \cos^2\delta) / 2)}$. Round an ellipse the angle no longer runs evenly; it runs a little fast and a little slow twice a turn, and r against t wobbles between r and $r\cos\delta$ at the same rate. But each whole turn still adds exactly 2π, so ω taken over whole turns is untouched.

That is why this experiment has no levelling step, unlike the projectile's, where a tilt bends a whole graph (Section 7.2). A phone looking straight down is lying flat, too, which the angle reading of Section 12 cannot use. What still matters is Section 1's assumption: the known length must lie at the marker's height, or it sets the wrong k. The table also assumes no perspective, and a known length laid across the tip; laid along it, k comes out $1 / \cos\delta$ too large.

Part 5 · Sensor

Air Track

The cart, felt from inside.

Here the iPhone rides on the cart. It never sees where the cart is, only how it accelerates, so the distance has to be built up from the acceleration a step at a time, which is the opposite of Part 2's route. First, though, the sensors set the track's angle, for both methods, because that angle decides the acceleration to expect.

In this part

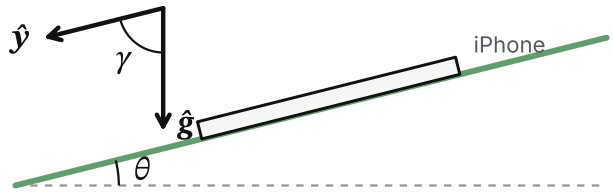
12 How steep is the track?	30
13 What the accelerometer feels	31
14 From acceleration to distance	32

12 How steep is the track?

Both methods begin an air-track run by setting the track's angle (a step that can be skipped), and both read it with the iPhone standing on the track on its long edge. CoreMotion gives the direction of gravity in the phone's own axes as a unit vector $\hat{\boldsymbol{g}}$, and the angle γ between it and the long edge $\hat{\boldsymbol{y}}$ is a dot product, as in Section 4:

$$\cos\gamma = \hat{\boldsymbol{g}} \cdot \hat{\boldsymbol{y}} \qquad\qquad \theta = 90° - \gamma$$

On a level track the long edge is at 90° to gravity, and every degree of slope takes a degree off γ. Since $\cos\gamma = \sin(90° - \gamma)$, this is just $\sin\theta = \hat{\boldsymbol{g}} \cdot \hat{\boldsymbol{y}}$: the share of gravity that lies along the track, the same $g \sin\theta$ that will pull the cart. The app shows the size, $|90° - \gamma|$, so the phone can face either way.



[Figure 12.A] **Reading the slope from gravity,** drawn steeper than any real track.

Smoothing. The angle is read 60 times a second, and each new reading counts for 15% of the shown value, the running value for 85%. So the shown angle follows roughly the last tenth of a second. It smooths the signed angle, −90° to 90°: on a 0° to 180° scale, a level phone flickering between 0.1° and 179.9° would average to 90°.

Not flat. The angle is only read with the phone on its edge. It counts as flat once the part of gravity through the screen, $|\hat{g}_z|$, passes 0.9, and as upright again only below 0.8, so it cannot flicker at the boundary.

12.1 How much the angle matters

The acceleration to expect is $g \sin\theta$. Differentiating, a small error $\Delta\theta$ in radians changes it by a fraction

$$\Delta(g \sin\theta)/(g \sin\theta) = (\cos\theta/\sin\theta)\,\Delta\theta = \Delta\theta/\tan\theta$$

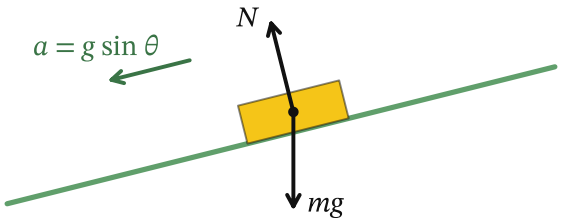
which is large for a shallow track. An error of just 0.1° is a 10% error at 1°, 5% at 2°, 2% at 5° and 1% at 10°. That is why the angle is only accepted within a small margin of the target, 0.1° unless changed in Settings, and as little as 0.05°.

13 What the accelerometer feels

An accelerometer does not measure acceleration. It measures the push on the phone from everything except gravity, per kilogram, which is called the specific force:

$$\boldsymbol{f} = \boldsymbol{a} - \boldsymbol{g}$$

Gravity pulls equally on every part of the phone, the sensor's own tiny test mass included, so it cannot be felt; only pushes can. A phone at rest on a table has $\boldsymbol{a} = 0$ and reads $\boldsymbol{f} = -\boldsymbol{g}$, the table's push upwards of 1 *g*. A phone in free fall reads nothing.



[Figure 13.A] **The forces on a cart with no friction.** The track's push *N* is all the accelerometer can feel, and none of it lies along the track.

On the track. A frictionless cart feels gravity and the track's push, and the push is at right angles to the track. So along the track the accelerometer reads *zero*, all the while the cart speeds up at $g \sin\theta$. By itself the sensor cannot see the very acceleration the lab is about.

So it comes from a difference. CoreMotion keeps its own estimate of which way is down, from the gyroscope and accelerometer together, and reports what is left once gravity is taken off, `userAcceleration`. CoreMotion's readings have the opposite sign to $\boldsymbol{f}$ (a phone at rest face up reads −1 *g* along *z*), and with that sign the leftover is $-\boldsymbol{a}$ in units of *g*:

$$a = -g_0 \times u_y \qquad g_0 = 9.80665\ \text{m/s}^2$$

with u_y the leftover along the long edge, which lies along the track. That is the minus sign in the code.

Check with numbers. On a 2° track, $a = 9.81 \times \sin 2° = 0.342\ \text{m/s}^2$. CoreMotion reports $u_y = -0.342 / 9.81 = -0.0349\ g$, and the app turns it back into $-9.81 \times (-0.0349) = 0.342\ \text{m/s}^2$.

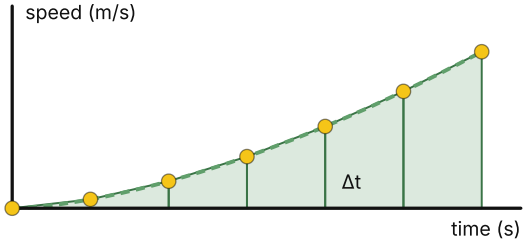
What it means. The acceleration is only as good as CoreMotion's idea of down. Were that off by 0.1°, the leftover would be off by $g \sin 0.1° = 0.017\ \text{m/s}^2$, which is 5% of a 2° track's $0.342\ \text{m/s}^2$: the same 5% as Section 12.1, for the same reason.

14 From acceleration to distance

The sensor gives a ten times a second. Speed is the area under the a–t graph and distance the area under the v–t graph, so the app adds up those areas a step at a time, each step a trapezium:

$$v_n = v_{n-1} + \tfrac{1}{2}(a_{n-1} + a_n)\,\Delta t$$
$$s_n = s_{n-1} + \tfrac{1}{2}(v_{n-1} + v_n)\,\Delta t$$

Δt is the gap between two readings' own timestamps (Section 2). Each step joins two neighbouring readings with a straight line and takes the area of the strip beneath it.



[Figure 14.A] **Adding up strips.** The dashed curve is the true speed; each strip joins two readings with a straight line. Their total area is the distance.

Why the trapezium is exact here. Under a constant acceleration the a–t graph is flat and the v–t graph a straight line, and the area under a straight line is exactly a trapezium. So neither step adds any error of its own, and a constant a comes out as exactly $d = \tfrac{1}{2}at^2$.

Check with numbers. Take $a = 1.00\ \text{m/s}^2$ throughout, $\Delta t = 0.1$ s, and the cart at rest at $t = 0$.

t (s)	v (m/s)	s (m)	$\tfrac{1}{2}at^2$ (m)
0.1	0.100	0.0050	0.0050
0.2	0.200	0.0200	0.0200
0.3	0.300	0.0450	0.0450

[Table 14.A] **The two steps against the formula.** They agree at every row.

The last row, for instance: $v = 0.200 + \tfrac{1}{2}(1 + 1)(0.1) = 0.300$ and $s = 0.0200 + \tfrac{1}{2}(0.200 + 0.300)(0.1) = 0.0450$.

14.1 Keeping a still cart still

Adding up has a weakness: speed, once added, stays until an equal and opposite acceleration takes it away. Noise and the sensor's small offset leave a little speed behind, so a cart that has stopped at the end of the track would seem to creep on for ever. So whenever a reading is tiny, below 0.03 m/s^2 either way, the app pulls the speed towards zero:

$$a \quad \rightarrow \quad a - 0.02\, v/\Delta t$$

Over a step of Δt the extra term changes v by about $-0.02v$, so the speed loses about 2% at every reading: with ten a second, $0.98^{10} = 0.82$, so it loses close to a fifth of its speed each second. The threshold is chosen to leave the run alone. $g \sin\theta$ is only 0.03 m/s^2 on a slope of 0.18°, so on all but the shallowest tracks the cart's own acceleration stays well above it while it runs, and only a cart that is held, or has stopped, is pulled to rest.

14.2 How errors grow

The camera measures the cart's position afresh on every frame, so an error on one frame stays on that frame. The sensor measures acceleration, so every error is carried into all the distances after it. A constant offset ε in a, such as a gravity estimate a fraction of a degree out (Section 13), grows as

$$\Delta v = \varepsilon t \qquad\qquad \Delta s = \tfrac{1}{2}\varepsilon t^2$$

which is the same shape as the real distance, so no graph can tell the two apart: fitting $d = \tfrac{1}{2}at^2$ to the data simply returns $a + \varepsilon$. With ε = 0.01 m/s^2, about a thousandth of g, a 2.0 s run down a 2° track ends $\tfrac{1}{2} \times 0.01 \times 2.0^2 = 0.020$ m out of 0.684 m, and the acceleration comes out 0.01 / 0.342 = 2.9% high.

Random noise is kinder. Its errors push both ways and partly cancel, though what is left still grows the longer the run goes on.

Which method, then? Both give d against t, but only the camera's errors stay put, a further reason, beside its easier set-up, that Accelab recommends filming an air track. The sensor's strengths are that it needs no ruler, no good light and no tracking, and that it reads the motion directly rather than from a picture.

Part 6 · Sensor

Circular Motion

The turntable, felt from inside.

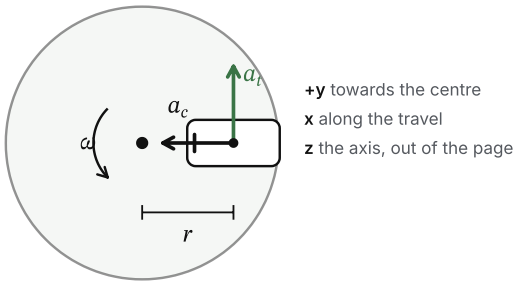
Here the iPhone lies flat on a turntable, its top edge pointing at the centre. Unlike the cart, nothing is added up to make the data: the gyroscope reads the angular velocity outright and the accelerometer the acceleration. The work is in keeping those readings honest, against the sensor's own offset, its noise, and a table that is not quite level.

In this part

15 What the sensors read	35
16 A turntable that isn't level	37
17 The graphs	39

15 What the sensors read

With the phone flat and its top edge pointing at the centre, its own axes line up with the motion: +*y* points at the centre, *x* runs along the direction of travel, and *z* is the turntable's axis.



[Figure 15.A] **The phone on the turntable, from above.** The bar marks its top edge. The dot is the accelerometer, whose distance from the axis is *r*.

Angular velocity. The gyroscope reads the rate of turning about *z* directly, in rad/s. Every part of a rigid turntable turns at the same rate, so ω does not depend on where the phone sits. The app keeps its size, $|\omega|$, so it does not matter which way the table was spun.

Acceleration. A point at radius *r* going round has an acceleration towards the centre, from its changing direction, and one along its travel, from any change in speed:

$$a_c = \omega^2 r \qquad a_t = r\alpha, \qquad \alpha = \mathrm{d}\omega / \mathrm{d}t$$

With the phone lined up, a_c falls on its *y* axis and a_t on its *x*. From the readings A_x and A_y, in *g*:

$$a_c = -g_0(A_y - b_y) \qquad a_t = -\sigma g_0(A_x - b_x)$$

b is the reading at rest (Section 15.1), and $\sigma = \pm 1$ is the sign of the gyroscope's reading, so that speeding up counts as positive whichever way the table turns. The minus signs are CoreMotion's reversed sign again (Section 13); the code still marks the turntable's as to be confirmed on a device.

The raw reading, not `userAcceleration`. *A* is the accelerometer's own reading, gravity and all. CoreMotion's estimate of down cannot tell a steady sideways acceleration from a tilt, so after some seconds of going round it could start to count a_c as part of down and take it away. Lying flat, gravity is along *z*, so *x* and *y* hold the motion with no estimate in them. A cart's run is over in a couple of seconds; a turntable can go round for a minute.

15.1 The reading at rest

Even standing still, an accelerometer reads a little in x and y: its own offset, a few hundredths of a g on some phones. That is 0.2 to 0.3 m/s^2, more than many a real tangential acceleration. So the sensors are read all through the 3-second countdown, while the turntable stands, and that reading b is subtracted from the run. Recording, and $t = 0$, start the moment the countdown ends, when the table is spun.

Why the median, not the mean? The countdown's haptic ticks jolt the phone. A median ignores a few wild readings; a mean is dragged by them. Five readings of 0.020, 0.021, 0.019, 0.020 and, during a tick, 0.090 g have a mean of 0.034 g but a median of 0.020 g. The one tick would move the mean by 0.014 g, or 0.14 m/s^2; it does not move the median at all.

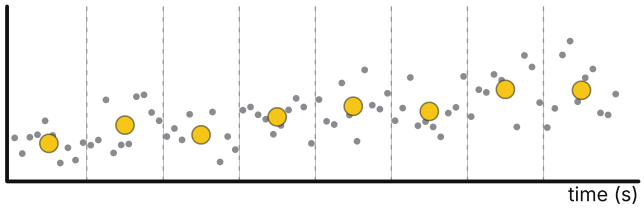
The gyroscope's reading at rest is taken the same way. ω is reported as read, but the rest reading is taken off before the gyroscope is added up into an angle (Section 16), where even a tiny offset would grow with time, as in Section 14.2.

15.2 Averaging into samples

The sensors are read 100 times a second, but the data has 10 samples a second: each one is the mean of the readings in its tenth of a second, about ten of them. Averaging n readings whose noise is independent and of size σ leaves noise of

$$\sigma/\sqrt{n} = \sigma/\sqrt{10} \approx 0.32\,\sigma$$

so each sample is about three times steadier than a single reading.



[Figure 15.2.A] **Readings into samples.** Grey: readings at 100 a second. Yellow: each tenth of a second's mean, the sample that goes into the data.

A sample's time is the mean of its readings' times, so the last one, cut short when Done is tapped, is still placed where it belongs. Its ω^2 is the square of its mean ω, squared after averaging: squaring first would add the noise's own square to every sample and push them all slightly high.

16 A turntable that isn't level

Tilt the turntable by a small angle ψ and gravity no longer lies along z: a share $g \sin \psi$ of it lies in the plane of the table, pointing downhill. That share is fixed to the room, but the phone turns, so in the phone's own axes it seems to go round once a turn, the other way. With φ the angle the phone has turned since it stood still, rotating the axes (Section 18.3) gives each channel a **leak** of

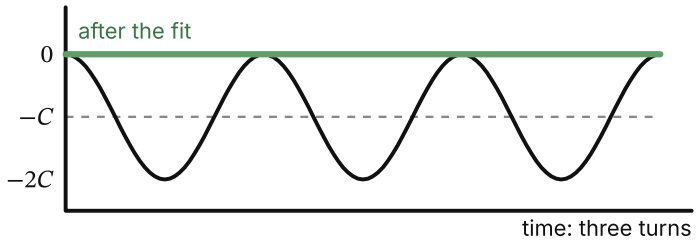
$$C \cos \varphi + S \sin \varphi \qquad\qquad \sqrt{(C^2 + S^2)} = g \sin \psi$$

where C and S depend on which way downhill was at the start. A tilt of only 1° leaks $9.81 \times \sin 1° = 0.17$ m/s^2, the size of a real tangential acceleration. φ comes from adding up the gyroscope's readings with the trapezium rule of Section 14, its rest reading taken off first.

Why the reading at rest isn't enough. It was taken at $\varphi = 0$, when the leak was C. Subtracting it leaves

$$C \cos \varphi + S \sin \varphi - C = C(\cos \varphi - 1) + S \sin \varphi$$

Over whole turns cos φ and sin φ average to zero, so what is left averages to $-C$: a steady offset as large as the leak was at rest, up to $g \sin \psi$, with a wobble once a turn on top. Going round does not average it away.



[Figure 16.A] **What is left of the leak,** taking $S = 0$. Black: after the reading at rest alone, averaging $-C$ (dashed). Green: after the fit of Section 16.1.

For a phone 0.15 m out turning at 3 rad/s, $a_c = 3^2 \times 0.15 = 1.35$ m/s^2, so a 1° tilt left in would be an error of up to $0.17 / 1.35 \approx 13\%$.

16.1 Fitting the leak

Taking the leak off means knowing C and S, which nothing measures directly. So once the run is over, the app fits them to the whole recording by least squares (Section 18.5). Each channel is fitted on its own, with four unknowns:

$$A_y \approx c_0 + c_1\omega^2 + c_2 \cos\varphi + c_3 \sin\varphi$$
$$A_x \approx c_0 + c_1(\mathrm{d}\omega/\mathrm{d}t) + c_2 \cos\varphi + c_3 \sin\varphi$$

c_0 takes the fixed offset, and c_2 and c_3 are the leak's C and S. c_1 takes the motion that channel really carries: ω^2 for the centripetal one, since $a_c = \omega^2 r$, and $\mathrm{d}\omega / \mathrm{d}t$ for the tangential one, since $a_t = r\alpha$. Without that term, the readings changing as the table slows down would be something the fit could only explain by bending the sinusoids. $\mathrm{d}\omega / \mathrm{d}t$ is the gyroscope's change across half a second, so that its noise does not swamp the real changes of speed.

Then the leak's *change since rest* is taken off every reading, and the reading at rest takes off what was there at the start, with each channel's own c_2 and c_3:

$$A_y \quad \rightarrow \quad A_y - [c_2(\cos\varphi - 1) + c_3 \sin\varphi]$$

On a level table the fit finds c_2 and c_3 close to zero and changes nothing. Its $\sqrt{(c_2^2 + c_3^2)}$, in g, is $\sin\psi$: the app works it out, though it does not show it.

Why at least two turns? Over less than a turn, $\cos\varphi$ and $\sin\varphi$ look much like a constant and a steady slope, so the fit cannot tell the leak from the offset and the motion; the equations become nearly dependent. With fewer than two turns the app keeps the reading at rest alone.

Why only at the end? The fit needs the whole run. So the numbers on screen while recording use the reading at rest alone, and tapping Done remakes every sample from the kept readings with the leak taken off.

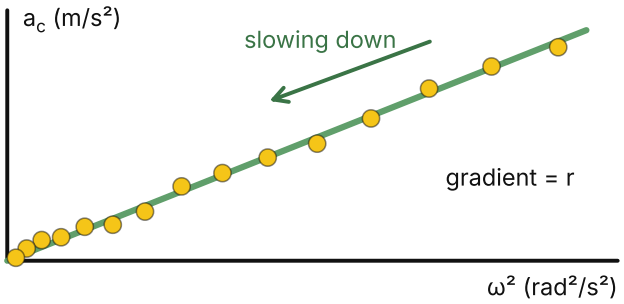
Why fit each channel on its own? The leak in x is the leak in y turned by a quarter of a turn, so one could be worked out from the other. But which way is a quarter turn depends on the handedness of the phone's axes and the sign conventions, and fitting both separately rests on none of that.

17 The graphs

Each sample is one CSV row of t, ω, ω^2, a_c and a_t. The lab's own graph is a_c against ω^2:

$$a_c = r \times \omega^2$$

a straight line through the origin whose gradient is r. That is why the export carries an ω^2 column: plotted against ω itself the points would lie on a parabola, from which r is much harder to read. As the turntable slows, its points slide down the line towards the origin, so a single spin-down draws the whole line. A steady spin gives one cluster instead.



[Figure 17.A] **A spin-down, one sample every tenth of a second.** The fast samples are far apart and the slow ones crowd towards the origin, all on one line.

Check with numbers. With r = 0.15 m: at ω = 4 rad/s, ω^2 = 16 and a_c = 2.40 m/s^2; at ω = 2, ω^2 = 4 and a_c = 0.60. The gradient is (2.40 − 0.60) / (16 − 4) = 0.15 m.

Which *r*? The distance from the axis to the accelerometer itself, a chip somewhere inside the phone, not to the phone's middle. A gradient a centimetre or so from a tape-measured radius may be the chip's position, not an error.

The other two graphs. ω against t has the angular acceleration α as its gradient, negative as friction slows the table. a_t against t is zero while the spin is steady, negative while it slows and positive while it speeds up. Because $a_t = r\alpha$, dividing a_t by the gradient of the ω–t graph gives r a second way.

17.1 A phone not pointing at the centre

If the top edge points a small angle β away from the centre, the phone's axes are turned by β from the directions of a_c and a_t, and rotating the axes (Section 18.3) mixes the two, with signs that depend on which way it is turned:

$$y \text{ reads} \quad a_c \cos\beta \pm a_t \sin\beta$$
$$x \text{ reads} \quad a_t \cos\beta \mp a_c \sin\beta$$

This is the tilted projectile of Section 7 again, with the same lopsided result (Section 18.4). a_c, read through $\cos\beta$, barely notices. a_t picks up a share $\sin\beta$ of a_c, which is usually far larger than a_t itself. At $\beta = 5°$, a_c reads only 0.4% low, but a_t gains 8.7% of a_c: 0.21 m/s^2 when $a_c = 2.40$ m/s^2.

An a_t that follows ω^2, large while the table is fast and fading as it slows, is the sign of a phone not pointing at the centre. The fit of Section 16.1 cannot take it off: it does not go round with φ, it follows ω^2.

17.2 The two methods compared

	Camera (Part 4)	Sensor (Part 6)
Measures	Where a marker is, on every frame	How fast the phone turns, and the push it feels
Gives directly	θ and r	ω, a_c and a_t
Works out	ω from the gradient of θ–t, then $a_c = \omega^2 r$	r from the gradient of a_c–ω^2

[Table 17.2.A] **Circular motion by each method.** Each measures what the other works out.

So on the same turntable the two check each other. Film a marker taped where the phone will lie, and the camera's $\omega^2 r$ at a given ω should match the sensor's a_c at the same ω, as long as the marker and the accelerometer are at the same radius.

18 Toolbox

The same few pieces of mathematics turn up again and again, so they are stated once, here: atan2 (Sections 3 and 10), the double-angle identities (Section 3), rotating the axes (Sections 7, 16 and 17), small angles (Sections 7, 11 and 17) and least squares (Sections 9, 16 and 17).

18.1 atan2

atan2(y, x) returns the angle of the vector (x, y) measured from the x-axis. It is a programming function, not a maths symbol, and y comes first.

$\tan^{-1}(y / x)$ has two problems. Dividing throws information away: (1, 1) and (−1, −1) point in opposite directions, but both give $y / x = 1$. And when $x = 0$ the division fails. atan2 takes x and y separately, reads their signs, and covers all four quadrants, from −180° to 180°.

Vector (x, y)	$\tan^{-1}(y / x)$	atan2(y, x)
(1, 1)	45°	45°
(−1, 1)	−45°, wrong	135°
(−1, −1)	45°, wrong	−135°
(0, 1)	cannot be computed	90°

[Table 18.1.A] **Inverse tangent against atan2** for four vectors.

Both problems really happen in Section 3. If the points spread more vertically than horizontally, $S_{xx} - S_{yy}$ is negative and $\tan^{-1}$ would pick the narrowest direction instead of the widest. If the track sits at exactly 45°, $S_{xx} = S_{yy}$ and the division is by zero.

In Section 10 the first problem is the whole point: a marker going round passes through all four quadrants on every turn, and only atan2 can tell them apart.

18.2 Double-angle identities

These write the sine and cosine of 2θ in terms of θ. They come from the compound-angle identities for $\sin(A + B)$ and $\cos(A + B)$ with $A = B = \theta$.

$$\sin 2\theta = 2 \sin \theta \cos \theta$$

$$\cos 2\theta = \cos^2\theta - \sin^2\theta = 2\cos^2\theta - 1 = 1 - 2\sin^2\theta$$

Rearranging the last two gives the forms used in Section 3:

$$\cos^2\theta = (1 + \cos 2\theta)/2 \qquad \sin^2\theta = (1 - \cos 2\theta)/2$$

A quick check at $\theta = 30°$: $\sin 60° = 0.866$ and $2 \times 0.5 \times 0.866 = 0.866$; $\cos 60° = 0.5$ and $0.75 - 0.25 = 0.5$.

In Section 3 they replace $\cos^2\varphi$, $\sin^2\varphi$ and $\sin\varphi \cos\varphi$ with single terms in 2φ. That is what lets V be read as a dot product, and it is why 2φ and ½ appear in the final formula.

18.3 Rotating the axes

Turn a pair of axes anticlockwise by θ. The new x-axis points along the unit vector $(\cos\theta, \sin\theta)$ and the new y-axis along $(-\sin\theta, \cos\theta)$. A point's new coordinates are its dot products with those two, exactly as in Section 4:

$$x' = x\cos\theta + y\sin\theta$$
$$y' = -x\sin\theta + y\cos\theta$$

The point has not moved; only its description has changed. Its distance from the origin is the same either way, because $x'^2 + y'^2 = x^2 + y^2$.

A quick check at $\theta = 30°$ with the point $(2, 1)$: $x' = 2 \times 0.866 + 1 \times 0.5 = 2.232$ and $y' = -2 \times 0.5 + 1 \times 0.866 = -0.134$, and $2.232^2 + 0.134^2 = 5.00 = 2^2 + 1^2$.

In Section 7 the new axes are the edges of a tilted picture, and the true motion is rewritten in them.

18.4 Small angles

Measure a small angle θ in radians (degrees $\times\ \pi / 180$). Then, to a very good approximation:

$$\sin\theta \approx \theta \qquad\qquad \cos\theta \approx 1 - \tfrac{1}{2}\theta^2$$

θ	θ in radians	$\sin\theta$	$\tfrac{1}{2}\theta^2$	$1 - \cos\theta$
1°	0.01745	0.01745	0.00015	0.00015
2°	0.03491	0.03490	0.00061	0.00061
5°	0.08727	0.08716	0.00381	0.00381
10°	0.17453	0.17365	0.01523	0.01519

[Table 18.4.A] **The small-angle approximations at four angles.** Each pair of columns agrees.

The sine is in proportion to the angle, but the cosine falls short of 1 only by the angle *squared*, which for a small angle is far smaller. That is the whole of Section 7.2: a slightly tilted phone bends the x–t graph, through $\sin\theta$, long before it spoils g, through $\cos\theta$. The same split decides Sections 11.2 and 17.1.

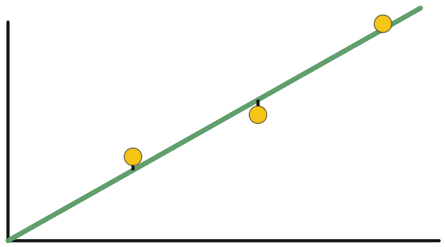
18.5 Least squares

With more equations than unknowns, no choice fits them all: a hundred tracked points do not all lie exactly on one circle. Least squares picks the unknowns that make the total of the squared misfits as small as it can be. Squaring does what it did in Section 3.2: misfits of opposite sign cannot cancel, and one big misfit counts for more than two of half its size.

The simplest case. Fit a line through the origin, $y = mx$, to points (x, y). The misfits are $y - mx$, and their squared total is smallest where its derivative with respect to m is zero:

$$-2\,\Sigma\, x(y - mx) = 0 \quad \Longrightarrow \quad m = \Sigma xy \,/\, \Sigma x^2$$

Check with numbers. For (1, 2.4), (2, 3.6) and (3, 6.2), $\Sigma xy = 2.4 + 7.2 + 18.6 = 28.2$ and $\Sigma x^2 = 14$, so $m = 2.014$. This is exactly how to take r from the a_c–ω^2 graph of Section 17, a line through the origin.



[Figure 18.5.A] **The best line through the origin.** The dark segments are the misfits; the third is too small to show.

More unknowns. In general each misfit is a measured value less a sum of unknowns, each multiplying something known. Setting every derivative to zero gives one linear equation per unknown, the **normal equations**. Section 9 has three, for D, E and F; each channel in Section 16.1 has four. The app solves them by elimination, moving the largest entry left onto the diagonal at each step so that rounding errors do not grow. Section 3's fit is a cousin that measures misfits at right angles to the line, because there x and y are equally uncertain.

When it fails. If two of the known quantities move together, the equations cannot decide between their unknowns and have no single solution. The app checks for this and gives up rather than guess: points in a line fit no circle (Section 9). Below two turns it does not even try the leak fit, keeping the reading at rest alone (Section 16.1).