Shear Force Drivetrain Derivations

30 lb bot, indirect (rigid) belt drive: drivetrain analysis

Kaden Dadabhoy

Shear Force (30 lb combat robot), Anteater Combat Engineering Robotics

September 2026

Abstract

This is a typeset copy of my handwritten derivation of the Shear Force drivetrain model. One motor per side drives a front wheel and a back wheel through a gearbox and two timing belts. I relate every axle speed to the motor speed, find the condition for the front and back wheels to roll at the same speed, work out the static normal loads, model the motor as a linear torque-speed line, and use a power balance to get the acceleration of the robot with a traction cap on the motor torque. The later sections list the equations that the command-line calculator (`main.cpp`) actually uses, the value of every parameter with the reason it was chosen, and how the code integrates the equations in time. Where the code differs from the notes, the derivation follows the notes and the code's version is given in a footnote.

The handwritten original is at https://www.kadendadabhoy.com/ShearForceFiles/SF_Drivetrain_Derivations_KadenDadabhoy.pdf. Page references such as "(scan p. 3)" point to that original.

Contents

1 Assumptions

The notes do not list assumptions in one place. These are the ones the derivation relies on, stated plainly.

1. I model one side of the robot. One motor, through a gearbox, drives one front wheel and one back wheel with two timing belts. Every mass, weight and force in this document is *per side* unless it says otherwise: m is half of the robot's mass, and mg is half of the robot's total weight W *(scan p. 3, 7)*. The two sides are assumed identical, which is what lets one side stand for the robot when it drives straight.

2. The belt drive is rigid: a belt does not stretch or skip teeth, so the belt speed is the same at both of its pulleys *(scan p. 1)*.

3. Each wheel rolls without slipping, so its ground speed is its angular speed times its radius, until the traction limit is reached *(scan p. 1, 6)*.

4. The wheel sizes and pulley ratios are chosen so that the front and back wheels roll at the same speed. With that condition met, one constant K_{sys} links motor speed to robot speed *(scan p. 2)*.

5. The robot drives in a straight line on flat ground. The notes have no aerodynamic drag, no rolling resistance and no loss in the gearbox or belts. The code lumps losses into two factors, η and s (Section 11).

6. The motor torque falls linearly with speed, from the stall torque at zero speed to zero at the no-load speed *(scan p. 4)*.

7. The normal loads come from statics (no load transfer while accelerating). The traction limit only uses the sum of the two normal loads, so the split between front and back does not change the result *(scan p. 3, 7)*.

8. The same friction coefficient μ applies at the front and back wheels *(scan p. 7)*.

9. The kinetic energy includes the chassis (translation), the front and back driven pulleys and the motor shaft *(scan p. 5)*. The rotational inertia of the wheels, belts and motor-axle pulleys is not included in the notes.

2 Variables and symbols

Tables 1, 2 and 3 list every symbol in this document, with its meaning and unit. The last column gives the equation, figure or section where the symbol is first used in a derivation step (a few also appear earlier in the assumptions). Every symbol is also defined in the text where it is first used. Conventions:

- Subscripts: m is the motor, f is the front, b is the back. The notes also use r (rear) for the back, so I_{rp}, ω_r and F_r belong to the back axle. p in a subscript means pulley and w means wheel (r_{fp} is the front pulley radius, r_{fw} the front wheel radius).
- Points 1, 2 and 3 are axle 1 (driven by the gearbox), axle 2 (front) and axle 3 (back), as numbered in the notes.
- Per side: m, mg, N_f, N_b, F_f, F_r, τ_m and I_{eq} all belong to one side of the robot. Only W is for the whole robot.
- The notes sometimes use two names for one quantity; I use one symbol for each here, and the tables note the other name where the notes use one.

3 System

Figure 1 is the layout I drew on the first page *(scan p. 1)*. The motor and gearbox turn axle 1 at ω_1. Two pulleys on axle 1, of pitch radius r_A and r_B, drive two timing belts. The front belt goes to the pulley of radius r_{fp} on the front axle (axle 2, turning at ω_2); the back belt goes to the pulley of radius r_{bp} on the back axle (axle 3, turning at ω_3). Each of those axles also carries a wheel: the front wheel has radius r_{fw} and the back wheel has radius r_{bw}.

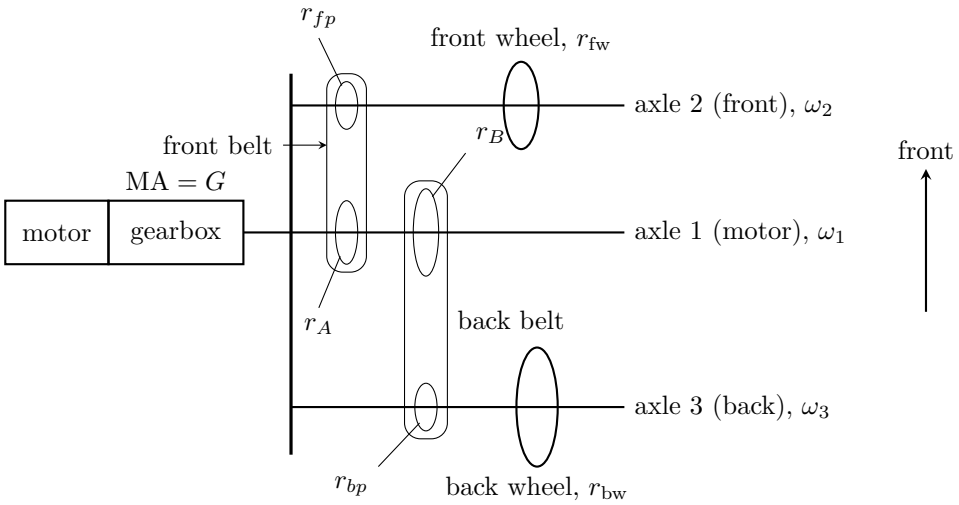

Figure 1: One side of the drivetrain, top view (after the sketch on scan p. 1). Pulleys and wheels are drawn edge-on. r_A, r_B, r_{fp} and r_{bp} are pulley pitch radii; r_{fw} and r_{bw} are wheel radii; ω_1, ω_2 and ω_3 are axle angular speeds.

Table 1: Symbols: gearing, geometry and kinematics.

Symbol	Meaning	Unit	First use
MA	mechanical advantage of the gearbox: input speed divided by output speed; equal to G	dimensionless	(1)
G	gearbox reduction: turns of the motor shaft per turn of axle 1 (19 for 19:1)	dimensionless	(1)
ω_m	angular speed of the motor shaft (the notes also write ω_{motor})	$\text{rad}\,\text{s}^{-1}$	(1)
ω_{out}	angular speed of the gearbox output, the same as ω_1	$\text{rad}\,\text{s}^{-1}$	(1)
r_{in}, r_{out}	pitch radii of the gearbox input and output gears ($r_{\text{in}} = r_m$, $r_{\text{out}} = r_1$)	m	(1)
v_m, v_1	pitch-line (tangential) speeds of the gearbox input and output gears where they mesh	$\text{m}\,\text{s}^{-1}$	(2)
r_m, r_1	pitch radius of the gear on the motor shaft and of the gear on axle 1	m	(3)
ω_1	angular speed of axle 1, the axle the gearbox drives (point 1)	$\text{rad}\,\text{s}^{-1}$	(3)
ω_2	angular speed of the front axle, axle 2 (point 2)	$\text{rad}\,\text{s}^{-1}$	Fig. 1
ω_3	angular speed of the back axle, axle 3 (point 3)	$\text{rad}\,\text{s}^{-1}$	Fig. 1
r_A	pitch radius of the front-belt pulley on axle 1 (the "motor pulley" for the front)	m	Fig. 1
r_{fp}	pitch radius of the front pulley on axle 2	m	Fig. 1
r_B	pitch radius of the back-belt pulley on axle 1 (the "motor pulley" for the back)	m	Fig. 1
r_{bp}	pitch radius of the back pulley on axle 3	m	Fig. 1
r_{fw}, r_{bw}	radius of the front wheel and of the back wheel (half of d_f and d_b)	m	Fig. 1
v_{belt}	belt speed (the same at every point of one belt)	$\text{m}\,\text{s}^{-1}$	(6)
$v_{\text{belt},1}$, $v_{\text{belt},2}$	speed of the front belt where it wraps the point-1 and the point-2 pulley	$\text{m}\,\text{s}^{-1}$	(6)
R_f	front speed ratio: teeth on the front pulley divided by teeth on its motor pulley, equal to r_{fp}/r_A	dimensionless	(12)
v_{fw}, v_{bw}	ground speed of the front wheel and of the back wheel	$\text{m}\,\text{s}^{-1}$	(14), (20)
R_b	back speed ratio: teeth on the back pulley divided by teeth on its motor pulley, equal to r_{bp}/r_B	dimensionless	(18)
v	robot speed, once the front and back wheel speeds match	$\text{m}\,\text{s}^{-1}$	(26)
K_{sys}	robot travel per radian of motor rotation, v/ω_m (the "system constant")	$\text{m}\,\text{rad}^{-1}$	(28)
t	time	s	(39)
a	robot acceleration, dv/dt	$\text{m}\,\text{s}^{-2}$	(47)

Table 2: Symbols: statics, motor, energy and traction. Everything here is per side.

Symbol	Meaning	Unit	First use
m	mass of one side: half of the robot's mass	kg	(30)
g	gravitational acceleration; the code uses standard gravity (Sec. 11)	$\mathrm{m\,s^{-2}}$	(30)
mg	weight carried by one side: half of the total weight W	N	(30)
z	contact point of the back wheel with the floor; the point I take moments about	(a point)	Fig. 3
COG	centre of gravity of one side	(a point)	Fig. 3
p	horizontal distance from the back wheel contact z to the COG	m	Fig. 3
q	horizontal distance from the COG to the front wheel contact	m	Fig. 3
N_f, N_b	normal (vertical) force from the floor on the front wheel and on the back wheel (the notes also write N_r for N_b)	N	Fig. 3
$\sum M_z$	sum of moments about z, counter-clockwise positive in Fig. 3	N m	(30)
$\sum F_y$	sum of vertical forces, upward positive	N	(32)
τ_m	torque the motor delivers to the drivetrain (the notes also write τ_{motor})	N m	(35)
τ_{pot}	"potential" torque: what the motor's torque-speed line gives at speed ω_m, before any traction cap	N m	(35)
τ_0	stall torque: motor torque at $\omega_m = 0$	N m	(35)
ω_{NL}	no-load speed: motor speed at which the torque falls to zero	$\mathrm{rad\,s^{-1}}$	(35)
$I_{\max}$	maximum current through the motor (the current at stall)	A	(36)
k_t	motor torque constant: torque per ampere	$\mathrm{N\,m\,A^{-1}}$	(36)
V	battery voltage across the motor	V	(37)
K_v	motor speed constant: no-load speed per volt, as sold in $\mathrm{rpm\,V^{-1}}$	$\mathrm{rpm\,V^{-1}}$	(37)
P_{in}, P_{out}	power into the drivetrain and power out of it	W	(38)
P, τ, ω	generic power, torque and angular speed, $P = \tau\omega$	W, N m, $\mathrm{rad\,s^{-1}}$	(39)
$\mathrm{KE}_{\mathrm{total}}$	total kinetic energy of one side	J	(39)
$\mathrm{KE}_{\mathrm{chassis}}$, ...	kinetic energy of the chassis (translation), front pulley, back pulley and motor shaft	J	(40)
I_{fp}, I_{rp}	moment of inertia of the front pulley and of the back (rear) pulley about its axle	$\mathrm{kg\,m^2}$	(41)
I_m	moment of inertia of the motor shaft (rotor) about its axis	$\mathrm{kg\,m^2}$	(41)
ω_f, ω_r	angular speed of the front pulley and of the back (rear) pulley	$\mathrm{rad\,s^{-1}}$	(41)
I_{eq}	equivalent inertia: everything that moves, reflected to the motor shaft	$\mathrm{kg\,m^2}$	(44)
α_m	angular acceleration of the motor shaft, $d\omega_m/dt$	$\mathrm{rad\,s^{-2}}$	(45)
P_{traction}	power the wheels can pass to the floor through friction	W	(51)
F_f, F_r	friction (traction) force of the floor on the front wheel and on the back (rear) wheel	N	(52)
μ	coefficient of friction between the wheels and the floor, the same front and back	dimensionless	(54)
τ_{limit}	traction-limited motor torque: the largest τ_m the wheels can pass to the floor without slipping	N m	(54)

Table 3: Symbols for the code's inputs, code-only quantities and the time stepping. Code names in brackets are from `main.cpp`.

Symbol	Meaning	Unit	First use
W	total weight of the whole robot (both sides), entered in pounds (`totalWeightLb`)	lb	(34)
d_f, d_b	front and back wheel diameters, entered in inches (`frontWheelInches`, `rearWheelInches`)	in	(62)
T_{fp}, T_{fm}	teeth on the front pulley, and on its motor pulley (`frontPulleyTeeth`, `motorPulleyFront`)	dimensionless	(63)
T_{bp}, T_{bm}	teeth on the back pulley, and on its motor pulley (`rearPulleyTeeth`, `motorPulleyRear`)	dimensionless	(63)
I_{lim}	current limit set on the drive ESC; the code uses it in place of $I_{\max}$ (`currentLimit`)	A	(67)
η	drivetrain efficiency factor; multiplies the stall torque (`drivetrainEfficiency`)	dimensionless	(67)
s	speed factor, which I also call the slip factor; multiplies the no-load speed (`speedFactor`)	dimensionless	(66)
F	drive force of one side at the wheel contact patches, $\tau_m / K_{\mathrm{sys}}$	N	(70)
$K_{\mathrm{sys}}^{\mathrm{front}}$, $K_{\mathrm{sys}}^{\mathrm{rear}}$	K_{sys} worked out from the front axle and from the back axle, for the mismatch check (`K_sys_front`, `K_sys_rear`)	$\mathrm{m\,rad^{-1}}$	Sec. 10
v_{top}	no-load top speed of the robot, $K_{\mathrm{sys}}\omega_{\mathrm{NL}}$	$\mathrm{m\,s^{-1}}$	(72)
$a_{\max}$	largest acceleration the traction limit allows	$\mathrm{m\,s^{-2}}$	(73)
Δt	time step of the integration (`dt`)	s	(77)
t_{end}	total simulated time (`simTime`)	s	Sec. 12
n	step index, $n = 0, 1, 2, \ldots$	dimensionless	Sec. 12
t_n	time at step n, $t_n = n\,\Delta t$ (`time`)	s	(79)
$\omega_{m,n}$	motor speed at step n (`omega_m`)	$\mathrm{rad\,s^{-1}}$	(74)
$\alpha_{m,n}$	motor acceleration at step n (`alpha_m`)	$\mathrm{rad\,s^{-2}}$	(76)
$\tau_{\mathrm{pot}}^{(n)}$, $\tau_m^{(n)}$	potential torque and applied torque at step n (`torque_pot`, `torque_m`)	N m	(74), (75)
v_n	robot speed at step n (`vel_m_s`)	$\mathrm{m\,s^{-1}}$	(78)
a_n	robot acceleration at step n (`current_accel_mps2`)	$\mathrm{m\,s^{-2}}$	(76)
F_n	drive force at step n, $\tau_m^{(n)} / K_{\mathrm{sys}}$	N	Sec. 12

4 Kinematics: axle speeds and wheel speeds

4.1 Motor and gearbox (point 1)

(scan p. 1) The motor shaft turns at ω_m. The gearbox has a mechanical advantage MA, the motor speed divided by the gearbox output speed ω_{out}, which is the given reduction G. I treat the gearbox as one gear pair: an input gear of pitch radius $r_{\text{in}} = r_m$ on the motor shaft and an output gear of pitch radius $r_{\text{out}} = r_1$ on axle 1. ω_1 is the rotation of that axle, the one attached to the motor.

$$\text{MA} = \frac{\omega_m}{\omega_{\text{out}}} = \frac{r_{\text{out}}}{r_{\text{in}}} = G \qquad \text{(given)} \tag{1}$$

Where the two gears mesh, their pitch-line speeds v_m and v_1 are equal:

$$v_m = v_1 \tag{2}$$

$$r_m\,\omega_m = r_1\,\omega_1 \tag{3}$$

$$\omega_1 = \omega_m\,\frac{r_m}{r_1} \tag{4}$$

and since $r_{\text{out}}/r_{\text{in}} = r_1/r_m = \text{MA} = G$,

$$\boxed{\omega_1 = \frac{1}{G}\,\omega_m} \tag{5}$$

4.2 Point 1 to point 2: front wheel

(scan p. 1) The front belt runs over the pulley of radius r_A on axle 1 and the pulley of radius r_{fp} on axle 2 (Figure 2, left). A rigid belt has one speed v_{belt} all the way round, so its speed at the point-1 pulley, $v_{\text{belt},1}$, equals its speed at the point-2 pulley, $v_{\text{belt},2}$:

$$v_{\text{belt},1} = v_{\text{belt},2} = v_{\text{belt}} \tag{6}$$

$$v_{\text{belt}} = \omega_1\,r_A \tag{7}$$

$$v_{\text{belt}} = \omega_2\,r_{fp} \tag{8}$$

$$\omega_2\,r_{fp} = \omega_1\,r_A \tag{9}$$

$$\omega_2 = \frac{r_A}{r_{fp}}\,\omega_1 \tag{10}$$

$$\omega_2 = \frac{1}{G}\,\frac{r_A}{r_{fp}}\,\omega_m \tag{11}$$

For timing pulleys the pitch radius is proportional to the tooth count, so the radius ratio is a tooth ratio. I define the front speed ratio R_f as the teeth on the front (driven) pulley divided by the teeth on its motor (driver) pulley:

$$\frac{r_A}{r_{fp}} = \frac{\text{teeth, motor pulley (driver)}}{\text{teeth, front pulley (driven)}} = \frac{1}{\text{front speed ratio}} = \frac{1}{R_f} \tag{12}$$

which gives the front axle speed and, with the front wheel radius r_{fw}, the front wheel's ground speed v_{fw}:

$$\omega_2 = \frac{1}{GR_f}\,\omega_m \tag{13}$$

$$v_{\text{fw}} = \omega_2\,r_{\text{fw}} = \boxed{\frac{r_{\text{fw}}}{GR_f}\,\omega_m} \tag{14}$$

4.3 Point 1 to point 3: back wheel

(scan p. 1) The back belt runs over the pulley of radius r_B on axle 1 and the pulley of radius r_{bp} on axle 3 (Figure 2, right), and has its own speed v_{belt}.

$$v_{\text{belt}} = r_B\,\omega_1 = r_{bp}\,\omega_3 \tag{15}$$

$$\omega_3 = \frac{r_B}{r_{bp}}\,\omega_1 \tag{16}$$

$$\omega_3 = \frac{r_B}{r_{bp}}\left(\frac{1}{G}\,\omega_m\right) \tag{17}$$

The back speed ratio R_b is the teeth on the back (driven) pulley divided by the teeth on its motor (driver) pulley:

$$\frac{r_B}{r_{bp}} = \frac{\text{teeth, motor pulley (driver)}}{\text{teeth, back pulley (driven)}} = \frac{1}{R_b} \tag{18}$$

With the back wheel radius r_{bw}, the back wheel's ground speed v_{bw} is

$$\omega_3 = \frac{1}{GR_b}\,\omega_m \tag{19}$$

$$v_{\text{bw}} = r_{\text{bw}}\,\omega_3 \tag{20}$$

$$v_{\text{bw}} = \boxed{\frac{r_{\text{bw}}}{GR_b}\,\omega_m} \tag{21}$$

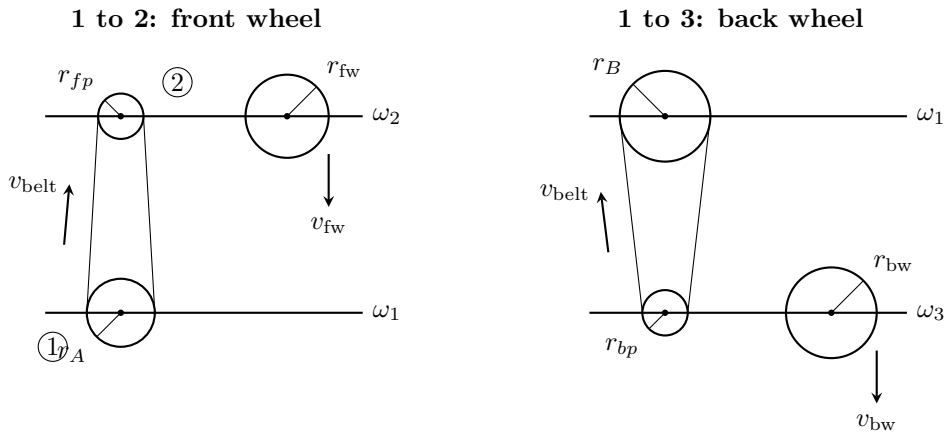

Figure 2: Belt stages (after the sketches on scan p. 1). Each belt has one speed v_{belt} at both of its pulleys; each wheel shares its axle's angular speed. v_{fw} and v_{bw} are the wheels' ground speeds.

4.4 Matching the front and back wheel speeds

(scan p. 2) We need $v_{\text{bw}} = v_{\text{fw}}$ in order to go straight. From (14) and (21):

$$\frac{r_{\text{bw}}}{R_b} \cdot \frac{\omega_m}{G} = \frac{\omega_m}{G} \cdot \frac{r_{\text{fw}}}{R_f} \tag{22}$$

$$\boxed{\frac{r_{\text{bw}}}{R_b} = \frac{r_{\text{fw}}}{R_f}} \tag{23}$$

where

$$R_b = \frac{\text{teeth, back pulley}}{\text{teeth of the corresponding motor pulley}} \tag{24}$$

$$R_f = \frac{\text{teeth, front pulley}}{\text{teeth of the corresponding motor pulley}} \tag{25}$$

Assuming (23) is satisfied, we can say that both wheels move at one robot speed v, and I name the common factor between v and ω_m the system constant K_{sys} (metres of travel per radian of motor rotation):

$$v_{\text{fw}} = v_{\text{bw}} = v \tag{26}$$

$$v = \frac{r_{\text{fw}}}{GR_f}\,\omega_m = \frac{r_{\text{bw}}}{GR_b}\,\omega_m \tag{27}$$

$$\frac{r_{\text{fw}}}{GR_f} = \frac{r_{\text{bw}}}{GR_b} \equiv K_{\text{sys}} \tag{28}$$

$$\boxed{v = K_{\text{sys}}\,\omega_m} \tag{29}$$

As a check on the code's default inputs (Table 5): the front gives $r_{\text{fw}}/R_f = 1.25\,\text{in}/(20/30) = 1.875\,\text{in}$ and the back gives $r_{\text{bw}}/R_b = 1.75\,\text{in}/(28/30) = 1.875\,\text{in}$, so the defaults satisfy (23).

5 Statics of the system

(scan p. 3) Figure 3 is the free-body diagram of one side at rest. The side has mass m (half of the robot's mass), so its weight mg acts at its centre of gravity (COG). The floor pushes up with N_b at the back wheel contact z and with N_f at the front wheel contact. The COG is a horizontal distance p in front of z and q behind the front contact. Taking moments about z (counter-clockwise positive) and summing vertical forces (upward positive):

$$\sum M_z = 0 = N_f\,(p+q) - mg\,(p) \tag{30}$$

$$N_f = \frac{mg\,p}{p+q} = \boxed{mg\left(\frac{p}{p+q}\right)} \tag{31}$$

$$\sum F_y = 0 \;\Rightarrow\; N_f - mg + N_b = 0 \tag{32}$$

$$N_b = \boxed{mg\left(1 - \frac{p}{p+q}\right)} \tag{33}$$

where mg is the weight of one side, half of the robot's total weight W:

$$mg = \frac{\text{total weight}}{2} = \frac{W}{2} \tag{34}$$

6 DC motor torque-speed curve

(scan p. 4) I take $\tau_m \equiv \tau_{\text{motor}}$, the torque the motor delivers, and $\omega_m \equiv \omega_{\text{motor}}$. The torque-speed curve is a straight line from the stall torque τ_0 (the torque at zero speed) to zero torque at the

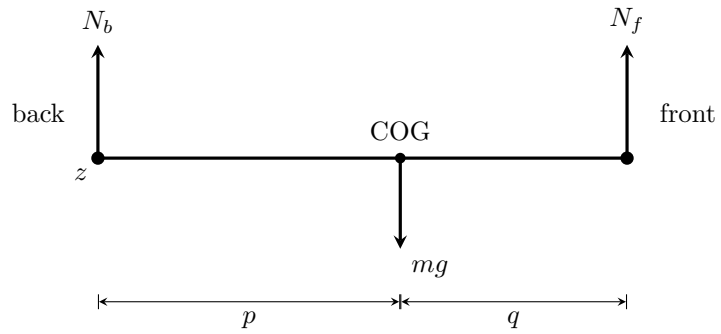

Figure 3: Free-body diagram of one side at rest (after scan p. 3). z is the back wheel contact, COG the centre of gravity of one side, mg the weight of one side, N_b and N_f the normal forces, p and q the horizontal distances.

no-load speed ω_{NL} (Figure 4). I call the torque on this line τ_{pot}, the "potential" torque, since the traction limit may not let all of it through (just renaming it for later):

$$\tau_{\mathrm{pot}} \equiv \tau_m = -\frac{\tau_0}{\omega_{\mathrm{NL}}}\,\omega_m + \tau_0 = \boxed{\tau_0\left(1 - \frac{\omega_m}{\omega_{\mathrm{NL}}}\right)} \tag{35}$$

Both end points come from motor properties: the stall torque is the maximum current $I_{\max}$ times the torque constant k_t (torque per ampere), and the no-load speed is the battery voltage V times the speed constant K_v (speed per volt)[1]:

$$\tau_0 = I_{\max}\,k_t \tag{36}$$

$$\omega_{\mathrm{NL}} = V\,K_v \tag{37}$$

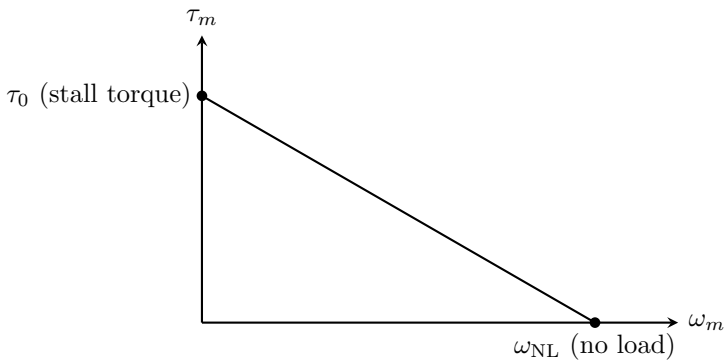

Figure 4: Linear DC motor torque-speed curve (after scan p. 4): motor torque τ_m against motor speed ω_m.

7 Work-energy: acceleration of the robot

(scan p. 5) Power in equals power out. For any shaft, power P is torque τ times angular speed ω, and here it all goes into the rate of change (with time t) of the total kinetic energy of one side,

[1]The code scales both end points: $\tau_0 = I_{\mathrm{lim}}\,k_t\,\eta$ with $k_t = 60/(2\pi K_v)$, and $\omega_{\mathrm{NL}} = K_v\,V\,s\,(2\pi/60)$, where $2\pi/60$ converts rpm to $\mathrm{rad\,s^{-1}}$. See (65) to (67) and Section 11.

$\mathrm{KE}_{\mathrm{total}}$ (in some cases this power is a force times a speed instead):

$$P_{\mathrm{in}} = P_{\mathrm{out}} \tag{38}$$

$$P = \tau\,\omega = \frac{d}{dt}\left(\mathrm{KE}_{\mathrm{total}}\right) \tag{39}$$

The kinetic energy is split into the chassis (moving at v), the front pulley, the back pulley and the motor shaft. I_{fp} and I_{rp} are the moments of inertia of the front and back (rear) pulleys, turning at ω_f and ω_r; I_m is the moment of inertia of the motor shaft, turning at ω_m:

$$\tau_m\,\omega_m = \frac{d}{dt}\left(\mathrm{KE}_{\mathrm{chassis}} + \mathrm{KE}_{\mathrm{front\ pulley}} + \mathrm{KE}_{\mathrm{back\ pulley}} + \mathrm{KE}_{\mathrm{motor\ shaft}}\right) \tag{40}$$

$$\tau_m\,\omega_m = \frac{d}{dt}\left(\tfrac{1}{2}mv^2 + \tfrac{1}{2}I_{fp}\,\omega_f^2 + \tfrac{1}{2}I_{rp}\,\omega_r^2 + \tfrac{1}{2}I_m\,\omega_m^2\right) \tag{41}$$

with $v = \omega_m K_{\mathrm{sys}}$ from (29), and the pulley speeds as written in the notes[2] $\omega_f = \omega_m/R_f$ and $\omega_r = \omega_m/R_b$:

$$\tau_m\,\omega_m = \frac{d}{dt}\left(\tfrac{1}{2}mK_{\mathrm{sys}}^2\,\omega_m^2 + \tfrac{1}{2}\frac{I_{fp}}{R_f^2}\,\omega_m^2 + \tfrac{1}{2}\frac{I_{rp}}{R_b^2}\,\omega_m^2 + \tfrac{1}{2}I_m\,\omega_m^2\right) \tag{42}$$

$$\tau_m\,\omega_m = \frac{1}{2}\left[mK_{\mathrm{sys}}^2 + \frac{I_{fp}}{R_f^2} + \frac{I_{rp}}{R_b^2} + I_m\right]\frac{d}{dt}\left(\omega_m^2\right) \tag{43}$$

The bracket is the equivalent inertia I_{eq}: the inertia of everything that moves, reflected to the motor shaft[3]:

$$\boxed{I_{\mathrm{eq}} = mK_{\mathrm{sys}}^2 + \frac{I_{fp}}{R_f^2} + \frac{I_{rp}}{R_b^2} + I_m} \tag{44}$$

Since $\frac{d}{dt}(\omega_m^2) = 2\,\omega_m\,\alpha_m$, where $\alpha_m = d\omega_m/dt$ is the motor's angular acceleration,

$$\tau_m\,\omega_m = I_{\mathrm{eq}}\,\omega_m\,\alpha_m \tag{45}$$

$$\tau_m = \boxed{I_{\mathrm{eq}}\,\alpha_m} \tag{46}$$

The robot's acceleration is $a = dv/dt$; differentiating (29) gives $a = \alpha_m K_{\mathrm{sys}}$:

$$a = \alpha_m\,K_{\mathrm{sys}} \tag{47}$$

$$\tau_m = \frac{I_{\mathrm{eq}}\,a}{K_{\mathrm{sys}}} \tag{48}$$

$$a = \boxed{\frac{\tau_m\,K_{\mathrm{sys}}}{I_{\mathrm{eq}}}} \tag{49}$$

[2] As written on scan p. 5, $\omega_f = \omega_m/R_f$ and $\omega_r = \omega_m/R_b$ have no factor of G. Equation (13) on scan p. 1 gives the front axle speed as $\omega_m/(GR_f)$, and (19) gives $\omega_m/(GR_b)$ for the back. With those, the pulley terms in (44) would be $I_{fp}/(GR_f)^2$ and $I_{rp}/(GR_b)^2$. The code does not include the pulley terms at all (see the next footnote), so this does not change any simulated result.

[3] The code keeps only the chassis term: `Inertia_eq = mass_kg_side * pow(K_sys, 2)`, that is $I_{\mathrm{eq}} = mK_{\mathrm{sys}}^2$ (`main.cpp` line 143).

8 Traction limit

(scan p. 6) However, τ_m might need to be limited because of traction. The maximum traction is the traction at the front plus the traction at the back:

$$\text{max traction} = \text{traction}_{\text{front}} + \text{traction}_{\text{back}} \tag{50}$$

Since our system is 4WD with rigid connections, torque will go to where there is traction. If one wheel has more traction than the other, the other one will stay at its traction limit and act as a free axle: the other wheel will pull it up in speed. It won't slip, because no torque above its maximum is applied to it. This is an advantage of 4WD.

9 Conclusions

(scan p. 7) At the traction limit, the power out goes into the traction power P_{traction}, the power the wheels pass to the floor through the friction forces F_f (front) and F_r (back, or rear)[4]:

$$P_{\text{out}} = \frac{d}{dt}\,(\text{KE}) = P_{\text{traction}} \tag{51}$$

$$\tau_m\,\omega_m = F_f\,v + F_r\,v \tag{52}$$

$$\tau_m\,\omega_m = (F_f + F_r)\,\omega_m\,K_{\text{sys}} \tag{53}$$

Cancelling ω_m, and using the friction law at each wheel (the largest friction force is μ times that wheel's normal force, so $F_f = \mu N_f$ and $F_r = \mu N_b$ at the limit, with μ the coefficient of friction between the wheels and the floor), gives the traction-limited torque τ_{limit}, the largest motor torque the wheels can pass on without slipping:

$$\tau_{\text{limit}} = \tau_m = \boxed{\mu\,(N_f + N_b)\,K_{\text{sys}}} \tag{54}$$

where, from (32) and (34), the two normal forces of one side add up to the weight of one side:

$$N_f + N_b = mg = \frac{\text{total weight}}{2} \tag{55}$$

So the torque that actually reaches the drivetrain is[5]

$$\boxed{\tau_m = \begin{cases} \tau_{\text{pot}} & \text{if } \tau_{\text{pot}} < \tau_{\text{limit}} \\ \tau_{\text{limit}} & \text{if } \tau_{\text{pot}} \geq \tau_{\text{limit}} \end{cases}} \tag{56}$$

and so, from (46), the motor acceleration is a function of time:

$$\alpha_m = \frac{\tau_m}{I_{\text{eq}}} \tag{57}$$

$$\omega_m = \int \alpha_m\,dt \tag{58}$$

$$v = K_{\text{sys}}\,\omega_m = \boxed{K_{\text{sys}} \int \alpha_m\,dt} \tag{59}$$

[4] The first line on scan p. 7 is partly unclear: it reads "Power out = $\frac{d}{dt}$(... KE)" followed by a relation symbol and "Power$_{\text{traction}}$". I typeset it as an equality, which is what the next line uses.

[5] The code also stops the motor torque from going negative: `torque_m = min(max(torque_pot, 0.0), Traction_Limit)` (`main.cpp` line 188). The code's "LIMIT" status flag uses $\tau_m \geq \tau_{\text{limit}} - 0.01\,\text{N m}$ (`main.cpp` line 195).

10 Final equations used in the code

The calculator is `ShearForce_DrivetrainCalcs/main.cpp` in https://github.com/kdadabhoy/ShearForce_DrivetrainCalcs; line numbers refer to commit `a741a42`. The code simulates one side. It takes the whole robot's weight W in pounds, the wheel diameters d_f and d_b in inches and the tooth counts T_{fp}, T_{fm} (front pulley and its motor pulley) and T_{bp}, T_{bm} (back pulley and its motor pulley), and converts to SI units:

$$m = \frac{W}{2} \times 0.453\,592\,37\,\mathrm{kg\,lb^{-1}} \tag{60}$$

$$mg = m \times 9.806\,65\,\mathrm{m\,s^{-2}} \tag{61}$$

$$r_{\mathrm{fw}} = \frac{d_f}{2} \times 0.0254\,\mathrm{m\,in^{-1}}, \qquad r_{\mathrm{bw}} = \frac{d_b}{2} \times 0.0254\,\mathrm{m\,in^{-1}} \tag{62}$$

$$R_f = \frac{T_{fp}}{T_{fm}}, \qquad R_b = \frac{T_{bp}}{T_{bm}} \tag{63}$$

Then it works out the constants of the model. I_{lim} is the current limit set on the drive ESC, η is a drivetrain efficiency factor and s is a speed factor; Section 11 gives their values and why:

$$K_{\mathrm{sys}} = \frac{r_{\mathrm{fw}}}{GR_f} \tag{64}$$

$$k_t = \frac{60}{2\pi K_v} \tag{65}$$

$$\omega_{\mathrm{NL}} = K_v\, V\, s\, \frac{2\pi}{60} \tag{66}$$

$$\tau_0 = I_{\mathrm{lim}}\, k_t\, \eta \tag{67}$$

$$\tau_{\mathrm{limit}} = \mu\, mg\, K_{\mathrm{sys}} \tag{68}$$

$$I_{\mathrm{eq}} = mK_{\mathrm{sys}}^2 \tag{69}$$

$$F = \frac{\tau_m}{K_{\mathrm{sys}}} \tag{70}$$

F is the drive force of one side at the wheel contact patches; the code only outputs it. Equation (65) is the usual relation for a brushless motor, $k_t = 1/K_v$ with K_v in $\mathrm{rad\,s^{-1}\,V^{-1}}$; the $60/(2\pi)$ converts from $\mathrm{rpm\,V^{-1}}$. Because the ESC limits the current to I_{lim}, the code's τ_0 is the torque at the current limit, not the motor's true stall torque. Table 4 maps each equation to its line in `main.cpp` and to the equation in the notes it comes from.

With the code's $I_{\mathrm{eq}} = mK_{\mathrm{sys}}^2$, the acceleration (49) reduces to

$$a = \frac{\tau_m\, K_{\mathrm{sys}}}{mK_{\mathrm{sys}}^2} = \frac{\tau_m / K_{\mathrm{sys}}}{m} = \frac{F}{m}, \tag{71}$$

so the coded model is a point mass m pushed by the drive force F, which is capped at μmg. The code guards the division with $\max(I_{\mathrm{eq}}, 1 \times 10^{-5}\,\mathrm{kg\,m^2})$ (lines 191 and 192).

If the front and back axles do not satisfy (23), the two values of the system constant differ. The code computes both, $K_{\mathrm{sys}}^{\mathrm{front}} = r_{\mathrm{fw}}/(GR_f)$ and $K_{\mathrm{sys}}^{\mathrm{rear}} = r_{\mathrm{bw}}/(GR_b)$ (lines 148 and 149), prints a velocity mismatch warning when $|K_{\mathrm{sys}}^{\mathrm{front}} - K_{\mathrm{sys}}^{\mathrm{rear}}| > 1 \times 10^{-4}\,\mathrm{m\,rad^{-1}}$ (lines 215 to 223), and simulates with the front value. SF_Telem, my telemetry app (part of my Cosmic engine), re-implements the same model and simulates both axles.

Table 5 lists the code's default inputs; Section 11 gives the reason for each one.

Table 4: Where each coded equation lives in `main.cpp` (commit `a741a42`) and where it comes from in the notes.

Quantity	Here	`main.cpp` line	From the notes
m, mg	(60), (61)	129, 130	(34)
r_{fw}, r_{bw}	(62)	131, 132	(input conversion)
R_f, R_b	(63)	135, 136	(25), (24)
K_{sys} (front axle)	(64)	138	(28)
k_t	(65)	139	(not in the notes)
ω_{NL}	(66)	140	(37), times s
τ_0	(67)	141	(36), times η
τ_{limit}	(68)	142	(54), (55)
I_{eq}	(69)	143	(44), chassis term only
τ_{pot}	(74)	187	(35)
τ_m	(75)	188	(56), clamped at 0
a, α_m	(76)	191, 192	(49), (57)
F	(70)	204	(output only)
v	(78)	208	(29)

Table 5: The code's default inputs, from `DrivetrainInputeTemplate.txt`. These are inputs, not results.

Input	Code's default	Symbol
Total simulation time	5.0 s	t_{end}
Time step	0.02 s (template); 0.002 s (`main.cpp` struct)	Δt
Gearbox reduction	19 (19:1)	G
Front wheel diameter	2.5 in	$d_f = 2r_{\text{fw}}$
Front pulley teeth / its motor pulley teeth	20 / 30	T_{fp} / T_{fm}
Rear wheel diameter	3.5 in	$d_b = 2r_{\text{bw}}$
Rear pulley teeth / its motor pulley teeth	28 / 30	T_{bp} / T_{bm}
Speed factor	0.85	s
Drivetrain efficiency	0.80	η
Motor speed constant	1400 rpm V^{-1}	K_v
Current limit	80.0 A	I_{lim}
Battery voltage	22.2 V	V
Coefficient of friction	1.0	μ
Total weight (whole robot)	30.0 lb	W

11 Parameter values and why I chose them

Each item gives the value and why it has that value. Commit hashes refer to the calculator repository, https://github.com/kdadabhoy/ShearForce_DrivetrainCalcs; "slide" refers to my June 2026 Shear Force presentation.

11.1 Robot and gearing

Total weight $W = 30\,\text{lb}$ (whole robot). Shear Force is a 30 lb combat robot, and staying at or under 30 lb was our most important requirement (slide 4). I model the robot at the class limit. The finished robot weighed 29 lb 14 oz, so the model's mass is slightly high. Per side, $m = W/2$, see (60).

Gearbox reduction $G = 19$ (19:1). This is the reduction of the drive gearbox, a property of the part rather than a tuned number. It has been 19 since the first commit of the calculator (`2925192`, 2026-02-16, comment "Gearbox reduction"). Together with the pulleys and wheels it sets K_{sys} (64), and so the top speed and the pushing torque. My self-imposed targets were a drive top speed of 15 mph or more with a "reasonable torque curve" (slide 4), and I left the drive traction-limited at low speed on purpose, as a factor of safety for pushing and for when another robot is on top of ours (slide 10). The drive gearboxes are Repeat Robotics Magnum 12/30 lb planetary gearboxes, bought with Repeat's 3536 motor. 19:1 is the ratio the gearbox comes with, so it was a fixed constraint rather than a design choice: I sized the belt stages and wheels around it, and the pulley system could accommodate it.

Motor speed constant $K_v = 1400\,\text{rpm}\,\text{V}^{-1}$. K_v is the catalogue speed constant of the drive motor. The first version of the code used a BadAss 3515 motor at $1130\,\text{rpm}\,\text{V}^{-1}$ (commit `2925192`, `main.cpp` line 55); commit `688e8ca` (2026-02-27) changed it to $1400\,\text{rpm}\,\text{V}^{-1}$. The change follows the parts list: a BadAss motor was first planned as an option with the Repeat Robotics Magnum, and in the end we bought the Magnum with Repeat's own motor and no BadAss, so the model moved from the BadAss K_v to the Repeat motor's. The Repeat 3536 is rated at $1400\,\text{rpm}\,\text{V}^{-1}$ and has 12 poles (6 pole pairs); the pole count does not enter this model, but it is the factor between the ESC's electrical rpm and shaft rpm.

Wheel diameters $d_f = 2.5\,\text{in}$, $d_b = 3.5\,\text{in}$. I found these by iterating in the simulator; my presentation calls them the finalized wheel diameters (slide 10). They had to meet two conditions. First, the front and back wheels must roll at the same speed (23). Second, the rear axle mount sits 0.5 in higher on the chassis than the front mount, so the robot is only level when the rear wheel radius is 0.5 in larger than the front: $1.75\,\text{in} - 1.25\,\text{in} = 0.5\,\text{in}$. The axle heights are fixed by the parts, so I cannot mount the wheels differently on the chassis.

Pulley teeth: 30, 20 and 28. I kept both motor pulleys at 30 teeth, $T_{fm} = T_{bm} = 30$, to reduce complexity (one part for both belts), then chose $T_{fp} = 20$ and $T_{bp} = 28$, that is 20 teeth at the front and 28 at the back, to synchronise the wheels. That gives $R_f = 20/30$ and $R_b = 28/30$, which satisfy (23) with the wheel sizes above (the check after (29)).

11.2 Motor, battery and loss factors

Battery voltage $V = 22.2\,\text{V}$. The drive battery is a 6S high-voltage LiPo (LiHV), 22.8 V nominal at 3.8 V per cell. I deliberately model it as a standard 6S pack, 6 cells at 3.7 V, which gives 22.2 V. It is a conservative sanity assumption: nominal is already a mid-discharge value, and

the pack voltage drops as it drains over a match. Since $\omega_{\mathrm{NL}} \propto V$ (66), the 0.6 V difference is a margin of about 2.6 % on the predicted no-load and top speeds. The model has used 22.2 V since the first commit.

Current limit $I_{\mathrm{lim}} = 80$ A. This is the current limit on the drive ESC. In the model it sets the stall torque, $\tau_0 = I_{\mathrm{lim}} k_t \eta$ (67), because the ESC never lets the motor draw its true stall current. Limiting the ESC current is how I keep full-throttle inputs from spinning the wheels (slide 8: "In real life can limit the current to the ESC"). It has been 80 since the first commit. The drive ESCs are SQESC 12100 units rated 100 A. 80 A is the limit we actually ran on the drive ESCs, including at Open Sauce, set from the specs of the parts. The other drive ESC settings were a start-up power of 120 % and no stall or stuck-rotor protection. (The weapon ESC is separate and ran at 250 A; it is not part of this model.)

Drivetrain efficiency $\eta = 0.80$. The code comment calls it "Drivetrain efficiency" (commit `60cdce9`, `main.cpp` line 81). A drivetrain efficiency factor usually stands for the torque lost between the motor and the floor: gear mesh, belt and bearing friction. Here it multiplies only the stall torque (67), so it lowers the torque available at every speed but leaves the no-load speed alone. It has been 0.80 since the first commit. It is an assumption, not a measured value: a round estimate of the losses, which I have not measured on the robot.

Speed factor (slip factor) $s = 0.85$. The code comment says it accounts for "air resistance/losses" (commit `60cdce9`, `main.cpp` line 77). A factor like this usually stands for the motor not reaching $K_v V$ under load: battery sag, ESC and winding losses, and drag. Here it multiplies only the no-load speed (66), so it scales the no-load top speed

$$v_{\mathrm{top}} = K_{\mathrm{sys}}\, \omega_{\mathrm{NL}} = K_{\mathrm{sys}}\, K_v\, V\, s\, \frac{2\pi}{60} \tag{72}$$

and lowers the torque at every speed above zero, but it does not change τ_0. Despite the name "slip factor", it does not model wheel slip: in this model wheel slip is handled by the traction limit (54). It was 0.90 in the first commit (`2925192`, line 59) and 0.85 from commit `60cdce9` (2026-02-27). Like η, it is an assumption, not a measured value; lowering it from 0.90 to 0.85 makes the predicted top speed more conservative (by 5.6 %, since $v_{\mathrm{top}} \propto s$).

Coefficient of friction $\mu = 1.0$. I set $\mu = 1$ so that the traction limit caps the robot's acceleration at 1 g. With the code's $I_{\mathrm{eq}} = m K_{\mathrm{sys}}^2$, (76) and (68) give

$$a_{\max} = \frac{\tau_{\mathrm{limit}}\, K_{\mathrm{sys}}}{I_{\mathrm{eq}}} = \frac{\mu\, m g\, K_{\mathrm{sys}}^2}{m K_{\mathrm{sys}}^2} = \mu\, g. \tag{73}$$

In my presentation I wrote that the model assumes the maximum traction is 1.0 g of the robot and caps the motor torque there (slide 8), and that a robot cannot accelerate above 1 g without magnets or some other downforce (slide 10). It is an assumed cap, not a measured friction coefficient for our wheels on the arena floor.

11.3 Simulation settings and fixed constants

Time step $\Delta t = 0.02$ s. This is a numerical setting, not a physical one. The explicit Euler update (Section 12) is only accurate when Δt is short compared with the time the robot takes to reach speed. In the default run in my presentation the speed levels off after about 1.5 s (slide 10), so 0.02 s gives about 75 steps over the spin-up. The value has changed: the early code

used 0.002 s (still the `main.cpp` struct default, line 23), slide 9 shows a run at 0.05 s, and the template uses 0.02 s (from commit `d1ca73e`). No convergence check between these step sizes is recorded.

Simulated time $t_{\text{end}} = 5\,\text{s}$. Long enough for the speed to level off (it does by about 2 s on slide 10). The first commit simulated 3 s (comment "3-second dash").

$g = 9.806\,65\,\text{m}\,\text{s}^{-2}$**.** Standard gravity (`main.cpp` line 126).

$0.453\,592\,37\,\text{kg}\,\text{lb}^{-1}$ **and** $0.0254\,\text{m}\,\text{in}^{-1}$**.** The exact definitions of the pound and the inch (lines 124 and 129).

$2\pi/60$**.** Converts rpm to $\text{rad}\,\text{s}^{-1}$ (lines 139 and 140).

$2.237\,\text{mph}$ **per** $\text{m}\,\text{s}^{-1}$**.** Output conversion only; the exact factor is 2.23694, rounded (line 199).

$1 \times 10^{-5}\,\text{kg}\,\text{m}^2$**.** Lower bound on I_{eq} in the division, to prevent division by zero (code comment, line 190). With valid inputs I_{eq} is far larger, so it never acts.

$0.01\,\text{N}\,\text{m}$**.** Tolerance for the "LIMIT" status flag (line 195). No reason is recorded. It only has to be small compared with τ_{limit} so that floating-point round-off does not flip the label; it does not change the physics.

$1 \times 10^{-4}\,\text{m}\,\text{rad}^{-1}$**.** Tolerance on $|K_{\text{sys}}^{\text{front}} - K_{\text{sys}}^{\text{rear}}|$ for the mismatch warning (line 215). No reason is recorded. It only has to be larger than round-off and smaller than any mismatch that matters.

12 How the code integrates the equations

The notes write the motion as integrals, (58) and (59). The code steps them forward in time with a fixed time step Δt using the explicit (forward) Euler method (`main.cpp` lines 181 to 210). The step index is $n = 0, 1, 2, \ldots$ and $t_n = n\,\Delta t$ is the time at step n; a subscript or superscript n on any quantity means its value at step n. Starting from rest, $t_0 = 0$, $\omega_{m,0} = 0$ and $v_0 = 0$, and for each step while $t_n \le t_{\text{end}}$ (the total simulated time):

$$\tau_{\text{pot}}^{(n)} = \tau_0 \left(1 - \frac{\omega_{m,n}}{\omega_{\text{NL}}}\right) \tag{74}$$

$$\tau_m^{(n)} = \min\left(\max\left(\tau_{\text{pot}}^{(n)}, 0\right),\ \tau_{\text{limit}}\right) \tag{75}$$

$$\alpha_{m,n} = \frac{\tau_m^{(n)}}{I_{\text{eq}}}, \qquad a_n = \frac{\tau_m^{(n)} K_{\text{sys}}}{I_{\text{eq}}} \tag{76}$$

$$\omega_{m,n+1} = \omega_{m,n} + \alpha_{m,n}\,\Delta t \tag{77}$$

$$v_{n+1} = K_{\text{sys}}\,\omega_{m,n+1} \tag{78}$$

$$t_{n+1} = t_n + \Delta t \tag{79}$$

Each row of output is written before the update, so row n holds t_n, v_n (in mph), a_n/g, the drive force $F_n = \tau_m^{(n)}/K_{\text{sys}}$, $\tau_m^{(n)}$ and a status of "LIMIT" or "MOTOR". The method is first order in Δt.

The loop in `main.cpp` (lines 185 to 210, comments and output statements removed):

```
while (time <= cfg.simTime)
{
    double torque_pot = torque_stall * (1.0 - (omega_m / omega_no_load));
    double torque_m = min(max(torque_pot, 0.0), Traction_Limit);
    double current_accel_mps2 = (torque_m * K_sys) / fmax(Inertia_eq, 0.00001);
    double alpha_m = torque_m / fmax(Inertia_eq, 0.00001);
    string status = (torque_m >= Traction_Limit - 0.01) ? "LIMIT" : "MOTOR";
    omega_m += alpha_m * cfg.dt;
    vel_m_s = omega_m * K_sys;
    time += cfg.dt;
}
```