run_oc_simple_time_sim_arbitrary(32).m
檔案

my_model(33).m
檔案

my_params(42).m
檔案

Read the codes. I carried out the sine test on my vehicle model, and I compared with Carsim model responses given the same initial condition and open loop controls (not shown here), and I found that the magnitude of tyre load of FL and FR tyres are higher than Carsim's and the tyre load of RL and RR tyres are almost the same as Carsim's. So why do you think this could happen? I have already set the main vehicle parameters the same between my model and Carsim's vehicle model, and I have already fit my tyre model to Carsim's tyre model.

Your comments narrow the diagnosis a lot: if Fx/Fy match well and steering/tyre transients are disabled, the mismatch is very likely in the **vertical-load closure**, not the horizontal tyre model.

已思考 35s ›

Your comments make sense, and they actually make the diagnosis cleaner.

Given what you clarified, I would now **de-emphasize tyre Fx/Fy, steering dynamics, and relaxation length**. If Fx/Fy match CarSim much better than Fz, then the front-load mismatch is most likely caused by the **vertical-load generation mechanism**, especially the way your current reduced model replaces suspension/tire vertical compliance with algebraic load-transfer equations.

1. Your biggest observation is correct

CarSim and your model are generating tyre vertical load in fundamentally different ways.

CarSim roughly does this:

$$F_z = f(\text{tyre compression},\ \text{tyre compression},\ K_t,\ C_t,\ \text{road geometry},\ \text{suspension kinematics})$$

Your current model instead does this:

$$F_z = f(X,\ Y_1,\ Y_2,\ \phi,\ \dot{\phi},\ \theta,\ \dot{\theta},\ H,\ q_1,\ q_2,\ k_\phi,\ c_\phi)$$

In your uploaded `my_model.m`, the four tyre loads are algebraic variables constrained by the Chapter-9-style load equations. They are not computed from tyre compression or wheel vertical displacement. The front split is directly:

$$\Delta Z_1 = \frac{Y_1 q_1 + k_{\phi 1}\phi + c_{\phi 1}\dot{\phi}}{t_1}$$

and then:

$$F_{z,FL} = 0.5 Z_1 - \Delta Z_1$$
$$F_{z,FR} = 0.5 Z_1 + \Delta Z_1$$

So if Y_1 is already matching CarSim reasonably well, then the remaining mismatch is probably from q_1, $k_{\phi 1}$, $c_{\phi 1}$, or the absence of the tyre/suspension vertical-compliance chain. my_model

2. Should you include tyre stiffness into your suspension stiffness?

Yes, but only in a very specific approximate way.

You should **not** add tyre stiffness directly to suspension stiffness as:

$$K_{\text{wrong}} = K_s + K_t$$

That would be physically wrong, because suspension spring and tyre vertical stiffness are not in parallel. They are approximately in **series** between the sprung mass and the road:

$$\text{sprung body} \to \text{suspension spring/damper} \to \text{wheel/unsprung mass} \to \text{tyre vertical stiffness} \to \text{road}$$

If you want to keep your current model structure and do a condensed approximation, the physically consistent first approximation is:

$$K_{\text{eff}} = \frac{K_s K_t}{K_s + K_t}$$

This is the equivalent vertical stiffness from sprung body to ground after removing the explicit tyre-compression/wheel-heave DOF.

With your numbers:

Front suspension stiffness:

$$K_{s,f} = 30\,000\ \text{N/m}$$

CarSim tyre vertical stiffness:

$$K_t = 325\ \text{N/mm} = 325\,000\ \text{N/m}$$

So:

$$K_{\text{eff},f} = \frac{30\,000 \times 325\,000}{30\,000 + 325\,000} \approx 27\,465\ \text{N/m}$$

Rear:

$$K_{s,r} = 43\,000\ \text{N/m}$$
$$K_{\text{eff},r} = \frac{43\,000 \times 325\,000}{43\,000 + 325\,000} \approx 37\,976\ \text{N/m}$$

So including tyre stiffness through static series condensation would reduce your effective front wheel rate by about **8.5%** and rear by about **11.7%**.

Your current `my_params.m` uses the raw wheel/suspension rates directly to build pitch and roll stiffness:

$$k_\theta = (K_{w,FL} + K_{w,FR}) l_f^2 + (K_{w,RL} + K_{w,RR}) l_r^2$$
$$k_{\phi 1} = (K_{w,FL} + K_{w,FR})(t_1/2)^2$$
$$k_{\phi 2} = (K_{w,RL} + K_{w,RR})(t_2/2)^2$$

with `Karb_f = Karb_r = 0`. my_params

So a reasonable test is to replace `Kw_FL`, `Kw_FR`, `Kw_RL`, `Kw_RR` by the condensed effective rates:

```
Kt = 325e3; % [N/m]

Kw_FL_raw = 30e3;
Kw_FR_raw = 30e3;
Kw_RL_raw = 43e3;
Kw_RR_raw = 43e3;

Kw_FL = Kw_FL_raw * Kt / (Kw_FL_raw + Kt);
Kw_FR = Kw_FR_raw * Kt / (Kw_FR_raw + Kt);
Kw_RL = Kw_RL_raw * Kt / (Kw_RL_raw + Kt);
Kw_RR = Kw_RR_raw * Kt / (Kw_RR_raw + Kt);
```

This is mathematically and physically defensible **only as a condensed low-frequency approximation**.

3. But this will probably not fully fix the mismatch

The tyre stiffness is large relative to the suspension stiffness:

$$325\ \text{kN/m} \gg 30\text{–}43\ \text{kN/m}$$

So the effective stiffness correction is only around 9–12%. That may improve the result, but if your front Fz magnitude is significantly higher than CarSim, tyre vertical stiffness alone probably does **not** explain all of it.

The bigger remaining suspect is still:

$$q_1$$

In your parameters:

$$h = 0.453\ \text{m}$$
$$d = 0.7h = 0.3171\ \text{m}$$
$$H = h - d = 0.1359\ \text{m}$$
$$q_1 = q_2 = d = 0.3171\ \text{m}$$

So your front lateral load transfer uses:

$$\Delta Z_1 = \frac{Y_1 \times 0.3171 + k_{\phi 1}\phi + c_{\phi 1}\dot{\phi}}{1.634}$$

That is quite a strong direct lateral load-transfer term. If CarSim's effective front roll-center/no-roll-center/load-transfer height is lower than $0.3171\ \text{m}$, your front load split will be too large even when Y_1 matches. my_params

4. Why Fx/Fy matching does not guarantee Fz matching

This is a key point.

In your model, Fx/Fy can match CarSim because your tyre model and slip kinematics are fitted well. But Fz is not merely an output of the tyre model. Fz is also a result of the vertical dynamics/load-transfer closure.

So you can have:

$$F_x^{my} \approx F_x^{CarSim}$$
$$F_y^{my} \approx F_y^{CarSim}$$

but still:

$$F_z^{my} \neq F_z^{CarSim}$$

because the vertical load path is different.

That is exactly what I would expect when comparing:

1. a reduced load-transfer algebraic model, against
2. a multibody/suspension/tyre-compression vertical-load model.

So your observation is not contradictory. It is actually a strong sign that your **horizontal tyre calibration is probably okay**, but your **vertical compliance/load-transfer model needs separate calibration**.

5. Can tyre stiffness be included without changing the model?

There are three levels.

Level 1 — easiest and defensible

Use tyre stiffness only to compute **effective wheel rates**:

$$K_{w,i}^{eff} = \frac{K_{s,i} K_{t,i}}{K_{s,i} + K_{t,i}}$$

Then rebuild:

$$k_{\phi 1},\ k_{\phi 2},\ k_\theta$$

using $K_{w,i}^{eff}$ instead of raw suspension spring rates.

This keeps your current model form unchanged.

This is useful, but it only approximates the static/low-frequency influence of tyre vertical compliance.

Level 2 — better within current model philosophy

Identify effective front/rear load-transfer parameters directly against CarSim:

$$q_1,\ q_2,\ H$$

and maybe:

$$k_{\phi 1},\ k_{\phi 2},\ c_{\phi 1},\ c_{\phi 2}$$

But I would start with only q_1, because your rear already matches well.

So I would test:

$$q_1 = 0.10,\ 0.15,\ 0.20,\ 0.25,\ 0.3171$$

while keeping q_2 unchanged.

This is probably more important than including tyre stiffness.

Level 3 — most physically faithful

Add heave/wheel vertical states, tyre compression, and suspension deflection explicitly.

That would make your model structurally closer to CarSim:

$$F_{z,i} = K_{t,i}\delta_{t,i} + C_{t,i}\dot{\delta}_{t,i}$$

But this changes the current model significantly. It would be more physically faithful, but less simple and less MPC-friendly.

6. My updated judgement after your comments

Given your clarifications, my ranking becomes:

1. **Most likely:** front effective lateral load-transfer height q_1 is too high.
2. **Also likely:** your current vertical-load algebraic closure is too different from CarSim's compression-based vertical model.
3. **Worth testing:** use tyre-stiffness-condensed effective wheel rates instead of raw suspension rates.
4. **Less likely now:** front Y_1 mismatch, because you said Fx/Fy match much better.
5. **Not a concern now:** tyre relaxation length and steering-system dynamics, since you disabled them in CarSim.
6. **Still worth checking:** exact CarSim output channel. Make sure it is tyre-road contact normal force, not suspension force, wheel-center vertical force, or filtered tyre load.

What I would do next

First, compute these four signals from your model and CarSim:

$$F_{zf,\text{sum}} = F_{z,FL} + F_{z,FR}$$
$$F_{zf,\text{split}} = F_{z,FR} - F_{z,FL}$$
$$F_{zr,\text{sum}} = F_{z,RL} + F_{z,RR}$$
$$F_{zr,\text{split}} = F_{z,RR} - F_{z,RL}$$

If front **sum** matches but front **split** is too high, tune q_1.

If front **sum** is too high, look at pitch/longitudinal transfer, static axle distribution, aero, and CarSim channel definition.

If front **split** is too high while Y_1 matches, then the answer is almost certainly:

$$q_1,\ k_{\phi 1},\ c_{\phi 1}$$

not the tyre force model.

My practical suggestion is:

1. Try the tyre-stiffness series correction for `Kw_*`.
2. Then sweep/identify q_1 against CarSim front load split.
3. Keep q_2 almost unchanged initially, because your rear Fz already matches.

資料來源

As you noticed that Carsim uses different method to compute tyre load than my vehicle model: Carsim uses directly tyre compression with tyre stiffness, in which tyre compression is a consequence of vehicle model's states, but my vehicle model uses explicit vehicle states, forces, suspension parameters and so on the compute tyre load with no-roll-point height as a tunable parameter. Due to this fundalmental difference in computing tyre load, if I would use a calibrated my vehicle model for MPC, how would you suggest me to calibrate or identify the front no-roll-center height and rear no-roll-center height? What open loop tests would you suggest me to do? How long time the data should be taken for identification? Should I use driving torque too or

顯示更多

I'll frame this as an identification-design problem: which parameters are structurally identifiable, what excitation separates front/rear effects, and what CarSim signals you should export.

A useful calibration trick here is to identify from **load split**, not raw four Fz values. That removes much of the static axle-load and pitch-transfer contamination and targets q1/q2 directly.

已思考 46s

Your reasoning is correct. Because CarSim and your model generate F_z by different physical closures, I would treat q_1 and q_2 not as literal geometric roll-center heights, but as **effective front/rear lateral load-transfer heights** that make your reduced model reproduce CarSim's tyre normal-load behavior over the operating range useful for MPC.

In your model, q_1 and q_2 enter very cleanly:

$$\Delta Z_1 = \frac{Y_1 q_1 + k_{\phi 1}\phi + c_{\phi 1}\dot{\phi}}{t_1}$$
$$\Delta Z_2 = \frac{Y_2 q_2 + k_{\phi 2}\phi + c_{\phi 2}\dot{\phi}}{t_2}$$
$$F_{z,FL} = 0.5Z_1 - \Delta Z_1, \quad F_{z,FR} = 0.5Z_1 + \Delta Z_1$$
$$F_{z,RL} = 0.5Z_2 - \Delta Z_2, \quad F_{z,RR} = 0.5Z_2 + \Delta Z_2$$

So q_1 and q_2 should primarily be identified from **left-right load split**, not from the absolute four tyre loads directly. Your code also already supports changing H, q_1, q_2 at runtime through the parameter vector `pv = [kappa; H; q1; q2]`, so this calibration can be done without rebuilding the CasADi model. my_model run_oc_simple_time_sim_arbitrary

1. What I would identify first

I would first identify only:

$$q_1,\ q_2$$

and keep:

$$H,\ k_{\phi 1},\ k_{\phi 2},\ c_{\phi 1},\ c_{\phi 2}$$

fixed.

Why? Because your current mismatch is mainly front F_z, while Fx/Fy already match reasonably well. If you try to identify too many parameters at once, q_1, roll stiffness, roll damping, and even H can compensate for each other. That can give a good fit for one test but poor generalization in MPC.

Your current default is:

$$h = 0.453, \quad d = 0.7h, \quad H = h - d, \quad q_1 = q_2 = d$$

so the current model assumes the same effective no-roll-center height for front and rear. But your CarSim comparison suggests the front and rear effective heights are probably not equal. my_params

So I would use:

$$q_1 \neq q_2$$

and identify them independently.

2. Use load split, not raw Fz, as the main target

From CarSim, compute:

$$\Delta Z_{1,CS} = \frac{F_{z,FR}^{CS} - F_{z,FL}^{CS}}{2}$$
$$\Delta Z_{2,CS} = \frac{F_{z,RR}^{CS} - F_{z,RL}^{CS}}{2}$$

Then compare those against your model's front and rear load-transfer equations.

For front:

$$t_1 \Delta Z_{1,CS} \approx Y_{1,CS} q_1 + k_{\phi 1}\phi_{CS} + c_{\phi 1}\dot{\phi}_{CS}$$

For rear:

$$t_2 \Delta Z_{2,CS} \approx Y_{2,CS} q_2 + k_{\phi 2}\phi_{CS} + c_{\phi 2}\dot{\phi}_{CS}$$

So if you have CarSim $Y_1, Y_2, \phi, \dot{\phi}, F_z$, a direct least-squares estimate is:

$$q_1 = \frac{\sum_k w_k Y_{1,k}\left(t_1 \Delta Z_{1,CS,k} - k_{\phi 1}\phi_k - c_{\phi 1}\dot{\phi}_k\right)}{\sum_k w_k Y_{1,k}^2}$$
$$q_2 = \frac{\sum_k w_k Y_{2,k}\left(t_2 \Delta Z_{2,CS,k} - k_{\phi 2}\phi_k - c_{\phi 2}\dot{\phi}_k\right)}{\sum_k w_k Y_{2,k}^2}$$

This is the cleanest first-pass identification.

But this formula should only use samples where $|Y_1|$ and $|Y_2|$ are not tiny. Around zero lateral force, the division becomes ill-conditioned.

3. Signals I would export from CarSim

Export at least these:

$$F_{z,FL}, F_{z,FR}, F_{z,RL}, F_{z,RR}$$
$$F_{x,FL}, F_{x,FR}, F_{x,RL}, F_{x,RR}$$
$$F_{y,FL}, F_{y,FR}, F_{y,RL}, F_{y,RR}$$

$$u_x,\ u_y,\ r,\ \phi,\ \dot{\phi},\ \theta,\ \dot{\theta}$$

$$\delta,\ T_t,\ T_b$$

If possible, also export lateral acceleration, longitudinal acceleration, wheel speeds, suspension deflections, and tyre deflections. You may not use all of them in the first fit, but they are very useful for diagnosing whether the mismatch comes from lateral transfer, pitch transfer, static load, or signal definition.

For your model, Y_1 and Y_2 are defined as:

$$Y_1 = F_{yf}\cos\delta + F_{xf}\sin\delta$$

$$Y_2 = F_{yr}$$

where $F_{yf} = F_{y,FL} + F_{y,FR}$, $F_{xf} = F_{x,FL} + F_{x,FR}$, and $F_{yr} = F_{y,RL} + F_{y,RR}$. These are exactly the quantities feeding the front and rear lateral load-transfer equations. my_model

4. Open-loop tests I would run in CarSim

I would use a sequence of increasingly aggressive tests. Do not start with complicated throttle-plus-steering maneuvers. Start with clean lateral excitation.

Test group A — straight-line baseline

Purpose: verify static load, pitch load, and CarSim channel definition.

Run:

$$\delta = 0, \quad T_t = 0, \quad T_b = 0$$

at several initial speeds, for example:

$$40,\ 60,\ 80,\ 100\ \mathrm{km/h}$$

Use 5–8 seconds per run.

This does not identify q_1, q_2, but it checks whether:

$$F_{z,FL} + F_{z,FR}$$

and

$$F_{z,RL} + F_{z,RR}$$

already match in straight running. If they do not, you have a static/pitch/aero/load-channel problem before even touching q_1, q_2.

Test group B — slow steering ramp / quasi-steady steer

This is the most important group for identifying q_1, q_2.

Use constant initial speed, zero drive torque, zero brake torque, and a slow steering ramp:

$$\delta(t) : 0 \to +\delta_{\max}$$

then hold, then return to zero, then repeat negative direction:

$$0 \to -\delta_{\max}$$

Use several amplitudes:

$$1^\circ,\ 2^\circ,\ 3^\circ,\ 5^\circ$$

Use several speeds:

$$40,\ 60,\ 80\ \mathrm{km/h}$$

Each run can be around 15–25 seconds:

- 2–3 seconds initial straight settling.
- 4–6 seconds ramp up.
- 3–5 seconds hold.
- 4–6 seconds ramp down.
- Repeat for opposite sign, or run a separate opposite-sign test.

This test is good because $\dot{\phi}$ is not dominant, so the fitted q_1, q_2 are less contaminated by damping mismatch.

Test group C — sine steer, low to moderate frequency

After quasi-steady ramp tests, use sine steering.

I would use:

$$\delta(t) = A\sin(2\pi f t)$$

with amplitudes:

$$A = 1^\circ,\ 2^\circ,\ 3^\circ,\ 5^\circ$$

and frequencies:

$$f = 0.2,\ 0.5,\ 1.0\ \mathrm{Hz}$$

For each sine test, use enough cycles:

- 0.2 Hz: 3–5 cycles → 15–25 seconds.
- 0.5 Hz: 6–8 cycles → 12–16 seconds.
- 1.0 Hz: 8–12 cycles → 8–12 seconds.

Discard the first cycle if there is an initial transient. Use the remaining cycles for identification.

This test checks whether the same q_1, q_2 also work dynamically. If quasi-steady tests give good q_1, q_2, but sine tests still mismatch, then the problem is not only q_1, q_2; it is probably roll damping, missing tyre vertical compliance dynamics, or missing unsprung/wheel vertical DOF.

Test group D — double lane change / fishhook

Use this for validation, not initial fitting.

These tests are useful because they excite transient roll and load transfer strongly. But if you fit q_1, q_2 only to these tests, the identified values may compensate for missing dynamics and become nonphysical.

So I would use these after the fit:

- Double lane change.
- Fishhook steer.
- Step steer with smooth ramp.
- Increasing-frequency sine sweep.

5. Should you use driving torque during identification?

For the first q_1, q_2 identification:

No, I would not use meaningful driving torque.

Use either:

$$T_t = 0, \quad T_b = 0$$

or a very small constant torque only to approximately balance drag.

Why? Because q_1, q_2 are lateral load-transfer parameters. If you add strong drive torque, you also introduce:

- longitudinal load transfer,
- pitch dynamics,
- rear longitudinal slip,
- coupling through X,
- possible changes in $F_x \sin\delta$ at the front,
- wheel-speed effects,
- combined-slip effects.

Your model's front/rear axle total load contains the longitudinal/pitch term:

$$\Delta Z = \frac{-Xh + I_y\ddot{\theta}}{l}$$

and then:

$$Z_1 = Z_{10} + \Delta Z - Z_{a1}$$

$$Z_2 = Z_{20} - \Delta Z - Z_{a2}$$

So strong driving/braking torque makes the total front/rear load distribution move around while you are trying to identify the lateral split. That makes the q_1, q_2 problem less clean. my_model

For the first pass, I would use steering-only data. Then, after getting q_1, q_2, validate with combined throttle/brake/steering data.

6. Should you use braking or acceleration at all?

Yes, but later.

I would separate the calibration into stages:

Stage 1 — identify lateral load-transfer heights

Use steering-only:

$$T_t \approx 0, \quad T_b = 0$$

Fit:

$$q_1,\ q_2$$

using load split.

Stage 2 — validate pitch/longitudinal load transfer

Use straight-line acceleration and braking:

$$\delta = 0, \quad T_t \neq 0$$

$$\delta = 0, \quad T_b \neq 0$$

Compare:

$$F_{zf,\mathrm{sum}} = F_{z,FL} + F_{z,FR}$$

$$F_{zr,\mathrm{sum}} = F_{z,RL} + F_{z,RR}$$

This checks whether H, pitch stiffness/damping, CG height, and longitudinal load-transfer behavior are acceptable.

Stage 3 — validate combined operation

Use:

- sine steer with mild drive torque,
- sine steer during coast,
- sine steer during braking,
- lane change with throttle lift,
- corner entry braking plus steering.

This is closer to MPC usage, but it should be used for validation after the clean lateral identification.

7. How long should the data window be?

For q_1, q_2, you do not need extremely long data. You need **good excitation** and **clean windows**.

My recommendation:

Quasi-steady ramp tests

Use 10–20 seconds of useful data per run.

Discard:

- the first 1–2 seconds,
- samples where $|Y_1|$ or $|Y_2|$ is very small,
- any region with obvious tyre saturation if your MPC will mostly operate below saturation.

Sine tests

Use 6–10 good cycles after initial transient.

For example:

- 0.5 Hz sine: 12–20 seconds total.
- 1.0 Hz sine: 8–12 seconds total.
- 0.2 Hz sine: 20–30 seconds total.

Longer is not automatically better. If the vehicle speed changes significantly during a coastdown test, the data becomes less clean. I would prefer several shorter tests at controlled initial speeds over one very long coastdown test.

8. My preferred identification workflow

I would do this:

Step 1 — preprocess CarSim data

Compute:

$$\Delta Z_{1,CS} = \frac{F_{z,FR}^{CS} - F_{z,FL}^{CS}}{2}$$

$$\Delta Z_{2,CS} = \frac{F_{z,RR}^{CS} - F_{z,RL}^{CS}}{2}$$

$$Y_{1,CS} = (F_{y,FL}^{CS} + F_{y,FR}^{CS})\cos\delta + (F_{x,FL}^{CS} + F_{x,FR}^{CS})\sin\delta$$

$$Y_{2,CS} = F_{y,RL}^{CS} + F_{y,RR}^{CS}$$

Step 2 — use only informative samples

Keep samples where:

$$|Y_1| > Y_{\min}$$

$$|Y_2| > Y_{\min}$$

For example:

$$Y_{\min} = 300\text{–}500\ \mathrm{N}$$

or use a percentage threshold such as 5% of peak axle lateral force.

Step 3 — direct least-squares estimate

Estimate q_1, q_2 using the load-split equations.

This gives a very good initial guess.

Step 4 — run your model with those q_1, q_2

Use your existing runtime options:

```
opts.model_q1 = q1_identified;
opts.model_q2 = q2_identified;
```

MATLAB

Your simulator already passes these through `make_model_pv` as runtime parameters.

run_oc_simple_time_sim_arbitrary

Step 5 — refine by simulation-level optimization

After the direct LS estimate, perform a nonlinear simulation-level fit:

$$\min_{q_1,q_2} \sum_k \left[w_f \left(\Delta Z_{1,my}(t_k; q_1, q_2) - \Delta Z_{1,CS}(t_k)\right)^2 + w_r \left(\Delta Z_{2,my}(t_k; q_1, q_2) - \Delta Z_{2,CS}(t_k)\right)^2 \right]$$

Use bounds:

$$0 \le q_1 \le h$$

$$0 \le q_2 \le h$$

or slightly wider if you want purely effective values:

$$-0.1h \le q_i \le 1.2h$$

but I would start with physical bounds.

Step 6 — validate, do not refit, on combined tests

Use combined steering/throttle/brake tests to check whether the identified q_1, q_2 generalize.

9. Important warning: do not identify q1/q2 from only one sine test

A single sine test can produce a misleading fit.

For example, if the frequency is high, the fitted q_1 may compensate for missing vertical tyre compliance dynamics or roll damping mismatch. Then it may work for that sine frequency but fail in a slower corner or in MPC.

So I would identify q_1, q_2 from a dataset containing:

- left and right steering,
- low and medium lateral acceleration,
- at least two speeds,
- at least one quasi-steady ramp,
- at least one sine frequency.

Then validate on tests that were not used in fitting.

10. My concrete recommendation

For your current purpose — calibrated reduced model for MPC — I would generate this dataset from CarSim:

Test	Speed	Input	Duration	Use
Straight coast	40/60/80 km/h	$\delta = 0, T_t = 0, T_b = 0$	5–8 s	static/load-channel check
Slow steer ramp	40/60/80 km/h	$0 \to \pm 2^\circ, \pm 3^\circ, \pm 5^\circ$	15–25 s	main q1/q2 fit
Sine steer	60/80 km/h	1–5°, 0.5 Hz	12–16 s	dynamic fit/check
Sine steer	60/80 km/h	1–3°, 1.0 Hz	8–12 s	dynamic validation
Double lane change	60/80 km/h	steering only	8–12 s	validation
Combined throttle/brake + steer	60/80 km/h	mild torque/brake + steer	8–15 s	final MPC-relevance validation

For the actual identification, I would start with only the slow ramps and low-frequency sine steer. I would not include aggressive acceleration/braking in the initial q_1, q_2 fit.

Final answer: **use mostly steering-only data for identifying front and rear no-roll-center heights. Use drive/brake only later for validation and possibly separate calibration of pitch/longitudinal transfer.**

資料來源

Thank you for your suggestions, could you double check your suggestions to make sure your ideas are fully rigorous and taking the ways that my model and Carsim's model of computing tyre load into account?

I'll re-check the recommendation against the actual algebraic structure of your model, especially what can and cannot be identified when CarSim's Fz comes from tyre compression.

After re-checking, I would slightly tighten the earlier advice: direct least-squares from CarSim states is good as a diagnostic and initial guess, but the final calibration should be **simulation-in-the-loop** with your model, because your model's roll/pitch states will not be exactly CarSim's states.

已思考 43s

Yes — after double-checking, I still agree with the **main direction**, but I would make the recommendation more precise and more rigorous.

The corrected version is:

Do not interpret identified q_1, q_2 as literal CarSim roll-center heights. Interpret them as effective front/rear load-transfer heights that project CarSim's compression-based tyre-load behavior into your algebraic Chapter-9-style load model.

That distinction is very important.

Your model does **not** have tyre compression, wheel heave, tyre vertical stiffness, or unsprung vertical dynamics. It lifts only the four tyre vertical loads as algebraic variables, and the model enforces algebraic load equations through `h_load`. The parameters H, q_1, q_2 are explicitly runtime load-transfer parameters, with q_1 and q_2 described as front and rear lateral load-transfer effective heights. my_model

So the identification target is not:

$$q_1, q_2 = \text{CarSim physical roll-center heights}$$

but rather:

$$q_1, q_2 = \text{effective parameters that make your reduced load model reproduce CarSim tyre loads over the MPC operating range}$$

That is rigorous and physically honest.

1. Why identifying q_1, q_2 is still sensible

In your model, the lateral load split is structurally controlled by:

$$\Delta Z_1 = \frac{Y_1 q_1 + k_{\phi 1}\phi + c_{\phi 1}\dot{\phi}}{t_1}$$

$$\Delta Z_2 = \frac{Y_2 q_2 + k_{\phi 2}\phi + c_{\phi 2}\dot{\phi}}{t_2}$$

and the four tyre loads are then:

$$F_{z,FL} = 0.5Z_1 - \Delta Z_1$$

$$F_{z,FR} = 0.5Z_1 + \Delta Z_1$$

$$F_{z,RL} = 0.5Z_2 - \Delta Z_2$$

$$F_{z,RR} = 0.5Z_2 + \Delta Z_2$$

This means q_1 and q_2 are exactly the parameters that control the **geometric/direct lateral load-transfer contribution** in your model. Therefore, if CarSim and your model have similar F_x, F_y, but different front F_z, it is rigorous to calibrate q_1, q_2, especially q_1. my_model

Your simulator also already allows H, q_1, q_2 to be changed at runtime through:

$$pv = [\kappa, H, q_1, q_2]$$

so the model structure already supports this calibration without rebuilding the CasADi graph.
run_oc_simple_time_sim_arbitrary

2. The important correction to my earlier suggestion

The direct least-squares formula using CarSim signals is useful, but it should **not be the final calibration by itself**.

It is valid as a diagnostic:

$$q_1^{LS} = \frac{\sum_k Y_{1,k}\left(t_1 \Delta Z_{1,k}^{CS} - k_{\phi 1}\phi_k - c_{\phi 1}\dot{\phi}_k\right)}{\sum_k Y_{1,k}^2}$$

$$q_2^{LS} = \frac{\sum_k Y_{2,k}\left(t_2 \Delta Z_{2,k}^{CS} - k_{\phi 2}\phi_k - c_{\phi 2}\dot{\phi}_k\right)}{\sum_k Y_{2,k}^2}$$

But if $\phi, \dot{\phi}, Y_1, Y_2$ come from CarSim, this identifies the best q_1, q_2 **assuming your model has the same roll state and same force decomposition as CarSim**. That assumption is not exactly true because CarSim's tyre load comes from tyre compression and suspension/tyre vertical motion, while your model has no tyre compression or wheel heave.

So the rigorous workflow should be:

1. Use equation-level LS only to get an initial estimate and diagnostic.
2. Then run your own model with the same open-loop controls and initial condition.
3. Fit q_1, q_2 by minimizing the mismatch between your model's predicted load split and CarSim's load split.

That final step is necessary because the model must be calibrated in the state trajectory that **your MPC model itself will generate**, not only in the CarSim state trajectory.

3. What is and is not identifiable

This is the most important rigor point.

q_1, q_2 are identifiable from left-right load split

Define:

$$\Delta Z_1^{CS} = \frac{F_{z,FR}^{CS} - F_{z,FL}^{CS}}{2}$$

$$\Delta Z_2^{CS} = \frac{F_{z,RR}^{CS} - F_{z,RL}^{CS}}{2}$$

Then q_1, q_2 mainly affect these quantities.

So for identifying q_1, q_2, do **not** primarily fit raw:

$$F_{z,FL}, F_{z,FR}, F_{z,RL}, F_{z,RR}$$

because raw Fz also includes static load, longitudinal pitch transfer, aero load, and CarSim channel-definition differences.

The cleaner targets are:

$$F_{z,FR} - F_{z,FL}$$

$$F_{z,RR} - F_{z,RL}$$

or equivalently $\Delta Z_1, \Delta Z_2$.

H is not as cleanly identifiable from the same steering-only data

In your model, H affects roll and pitch dynamics, not only tyre-load algebra. For example, the model uses:

$$LM = -k_\phi \phi - c_\phi \dot{\phi} + mgH\phi$$

$$A_\phi = LM + HY$$

and then computes $\ddot{\phi}$, $\dot{r}$, $\dot{u}_x$, and $\dot{u}_y$. So H is coupled with roll motion and vehicle dynamics, while q_1, q_2 directly appear in the load-split algebra. my_model

Therefore I would **not** freely identify H, q_1, q_2 all together from one sine-steer test unless you have a broad dataset. It may fit the data but the parameters can compensate for each other.

For the front/rear no-roll-center problem, first identify q_1, q_2 while holding H fixed. Only later, if front/rear axle **sum loads** or roll dynamics are also wrong, consider H.

4. Tyre stiffness issue: what role should it play?

CarSim's tyre stiffness affects Fz through tyre compression. Your model does not have tyre compression. Therefore, there is no exact mathematically equivalent way to "add tyre stiffness" to your current model without adding at least one missing vertical compliance coordinate.

The only physically reasonable reduced approximation is to fold tyre stiffness into an **effective vertical wheel rate**:

$$K_{\text{eff}} = \frac{K_s K_t}{K_s + K_t}$$

not:

$$K_s + K_t$$

Your current parameters build pitch and roll stiffness from `Kw_FL`, `Kw_FR`, `Kw_RL`, and `Kw_RR`, then compute $k_{\phi 1}, k_{\phi 2}, k_\theta$ from those values. my_params

So if you want to account for CarSim tyre stiffness without changing the model structure, the defensible approximation is:

$$K_{w,i}^{eff} = \frac{K_{s,i} K_{t,i}}{K_{s,i} + K_{t,i}}$$

Then use $K_{w,i}^{eff}$ in the roll and pitch stiffness calculations.

But this is only a **low-frequency static compliance correction**. It will not reproduce CarSim's dynamic tyre compression behavior. If you do not make this correction, then the identified q_1, q_2 will partly absorb the missing tyre compliance. That is acceptable for an MPC prediction model, but you should recognize the parameters as empirical effective parameters.

5. Revised open-loop test recommendation

I still recommend steering-dominant tests first, but with a stronger reason:

Since q_1, q_2 mainly govern left-right lateral load transfer, the identification should avoid strong pitch and longitudinal load-transfer excitation at first.

So the first dataset should be:

$$T_t \approx 0, \quad T_b = 0, \quad \delta \neq 0$$

or, if the vehicle slows down too much:

$$T_t = \text{small drag-balancing torque}, \quad T_b = 0$$

Do **not** use strong drive torque in the first q_1, q_2 fit.

Reason: drive torque changes X, wheel speeds, longitudinal load transfer, pitch dynamics, and possibly the front $F_x \sin\delta$ contribution inside Y_1. Your model's front/rear axle total loads also include:

$$\Delta Z = \frac{-Xh + I_y \ddot{\theta}}{l}$$

so strong drive/brake data contaminates lateral load-transfer identification with pitch/longitudinal effects.
my_model

6. Best test set for rigorous identification

I would use the following hierarchy.

Test 0 — static and straight-line validation

Use:

$$\delta = 0, \quad T_t = 0, \quad T_b = 0$$

at 40, 60, 80, 100 km/h.

Use 5–8 seconds each.

Purpose: check whether CarSim Fz channel and your static/front-rear axle load are comparable. This test does not identify q_1, q_2.

Test 1 — slow steering ramp

This should be your main identification test.

Use:

$$0 \to +\delta_{\max}$$

and

$$0 \to -\delta_{\max}$$

at 40, 60, 80 km/h.

Use amplitudes:

$$1^\circ,\ 2^\circ,\ 3^\circ,\ 5^\circ$$

Use 15–25 seconds per run.

Purpose: identify quasi-steady q_1, q_2 with minimal contamination from roll-rate damping and missing tyre vertical dynamics.

Test 2 — low-frequency sine steer

Use:

$$0.2\ \text{Hz}, \quad 0.5\ \text{Hz}$$

with amplitudes:

$$1^\circ,\ 2^\circ,\ 3^\circ,\ 5^\circ$$

Use 15–25 seconds for 0.2 Hz and 12–16 seconds for 0.5 Hz.

Purpose: check whether the same q_1, q_2 work in dynamic but not too aggressive conditions.

Test 3 — validation only: higher-frequency sine / lane change

Use:

$$1.0\ \text{Hz}$$

or double-lane-change/fishhook-style steering.

Do not use these as the first fitting dataset. They contain more dynamic vertical-compliance effects that your current model cannot exactly represent.

Test 4 — combined torque + steering validation

After q_1, q_2 are identified, validate with:

- mild acceleration plus steering,
- throttle lift plus steering,
- mild braking plus steering,
- corner-entry-like open-loop input.

Use this to check whether the calibrated model is useful for MPC. Do not use it as the first q_1, q_2 identification dataset.

7. How long should the data be?

The right answer is not "as long as possible." It is "long enough to excite the relevant load-transfer dynamics while keeping the operating condition controlled."

For each speed/amplitude case:

- Slow ramp: 15–25 s.
- 0.2 Hz sine: 3–5 cycles, so 15–25 s.
- 0.5 Hz sine: 6–8 cycles, so 12–16 s.
- 1.0 Hz sine: 8–12 cycles, so 8–12 s.

Discard:

- the first transient second or first sine cycle,
- samples where $|Y_1|$ or $|Y_2|$ is very small,
- samples near tyre saturation unless you specifically want drift/high-limit MPC behavior,
- samples where speed has decayed too much.

8. The final calibration objective I now recommend

For the final simulation-in-the-loop fit, use your model with the same open-loop input as CarSim and minimize:

$$J(q_1, q_2) = \sum_k w_1 \left[\left(F_{z,FR}^{my} - F_{z,FL}^{my} \right) - \left(F_{z,FR}^{CS} - F_{z,FL}^{CS} \right) \right]^2$$

$$+ \sum_k w_2 \left[\left(F_{z,RR}^{my} - F_{z,RL}^{my} \right) - \left(F_{z,RR}^{CS} - F_{z,RL}^{CS} \right) \right]^2$$

Optionally add a smaller term for axle sums:

$$+\epsilon \sum_k \left[\left(F_{z,FL}^{my} + F_{z,FR}^{my} \right) - \left(F_{z,FL}^{CS} + F_{z,FR}^{CS} \right) \right]^2$$

$$+\epsilon \sum_k \left[\left(F_{z,RL}^{my} + F_{z,RR}^{my} \right) - \left(F_{z,RL}^{CS} + F_{z,RR}^{CS} \right) \right]^2$$

with ϵ small.

Why small? Because q_1, q_2 should mainly fit load split. Axle-sum errors may come from H, pitch transfer, aero, static load, or CarSim's compression-based vertical model. You do not want q_1, q_2 to compensate for everything.

Use bounds:

$$0 \le q_1 \le h$$

$$0 \le q_2 \le h$$

At least initially. Since your current `veh.h = 0.453`, this means:

$$0 \le q_1, q_2 \le 0.453 \text{ m}$$

Your current nominal value is:

$$q_1 = q_2 = d = 0.7h = 0.3171 \text{ m}$$

which may be too high for the front if your front Fz split is larger than CarSim's. my_params

9. Final double-checked conclusion

Your identification idea is rigorous **only if framed correctly**:

Wrong framing:

$$q_1, q_2 = \text{true CarSim roll-center/no-roll-center heights}$$

Correct framing:

$$q_1, q_2 = \text{effective front/rear lateral load-transfer heights for your reduced MPC model}$$

Because your model and CarSim compute tyre load fundamentally differently, exact equivalence is impossible without adding tyre compression/wheel-heave/heave dynamics. But fitting effective q_1, q_2 is still a valid model-reduction/calibration strategy.

My final recommendation is:

1. Use CarSim steering-only or nearly steering-only tests first.
2. Fit mainly front/rear load split, not raw four Fz.
3. Use direct LS only as an initial diagnostic.
4. Use simulation-in-the-loop fitting as the final calibration.
5. Do not identify $H, q_1, q_2, k_\phi, c_\phi$ all together from one sine test.
6. Use drive/brake tests after q_1, q_2 calibration to validate pitch/combined behavior, not as the first lateral-load-transfer identification dataset.

資料來源

Oh, I forgot to mention, that when comparing my sine wave test with Carsim, not only the front tyre loads are magnitude of difference (highly probably due to current q1 set up value not representative as you said), but also there's phase difference between Fz front left of mine and Carsim. Would this be due to the different Fz calculation routes between my model and Carsim?

Yes — a phase difference in front F_z is actually an even stronger sign that the issue is not just the scalar value of q_1, but the **dynamic vertical-load path**.

已思考 25s ›

Yes. The phase difference can absolutely be due to the different F_z calculation routes between your model and CarSim.

But I would phrase it more carefully:

The magnitude error may be mainly due to q_1, but the phase error is probably due to dynamic differences in the vertical-load path, not just the numerical value of q_1.

In your model, front tyre load is algebraic:

$$F_{z,FL} = 0.5Z_1 - \Delta Z_1$$

$$F_{z,FR} = 0.5Z_1 + \Delta Z_1$$

where:

$$\Delta Z_1 = \frac{Y_1 q_1 + k_{\phi 1}\phi + c_{\phi 1}\dot{\phi}}{t_1}$$

and:

$$Z_1 = Z_{10} + \Delta Z - Z_{a1}$$

So front F_z is computed immediately from lateral force Y_1, roll angle ϕ, roll rate $\dot{\phi}$, pitch/longitudinal transfer, and the tunable effective height q_1. There is no explicit tyre compression state, wheel vertical displacement, unsprung vertical velocity, or tyre vertical stiffness state in this F_z calculation. my_model

CarSim, by contrast, computes tyre normal load through tyre/suspension vertical deflection. That gives the vertical-load path its own dynamics.

So yes, the phase mismatch is very plausibly caused by this structural difference.

Important point: q_1 mostly changes amplitude, not phase

If you only change q_1, you mainly change the size of the direct front lateral load-transfer term:

$$\frac{Y_1 q_1}{t_1}$$

That will strongly affect the magnitude of:

$$F_{z,FR} - F_{z,FL}$$

But by itself, q_1 does not create a new dynamic delay, because it multiplies Y_1 algebraically.

So if your front F_z has both:

1. amplitude too large, and
2. phase different from CarSim,

then the interpretation should be:

- q_1 is probably wrong for amplitude;
- the phase difference probably comes from missing or mismatched vertical dynamics: tyre compression, tyre vertical stiffness, unsprung vertical motion, suspension damper dynamics, or roll damping/roll-state mismatch.

Why your model can have different Fz phase even when Fx/Fy match

This is very possible.

Fx/Fy can match because the tyre slip and horizontal tyre force model are well fitted. But Fz is not only a tyre-force output. In your model, Fz is a **load-transfer algebraic reconstruction**.

In CarSim, Fz is closer to:

$$F_z^{CS} \approx K_t z_t + C_t \dot{z}_t$$

where z_t is tyre compression. But z_t depends on sprung body roll/pitch/heave, unsprung motion, suspension kinematics, tyre vertical stiffness, and road contact geometry.

Your model instead says, approximately:

$$F_z^{my} = f(Y_1, Y_2, X, \phi, \dot{\phi}, \theta, \dot{\theta})$$

So even if:

$$F_y^{my} \approx F_y^{CS}$$

you can still have:

$$F_z^{my}(t) \not\approx F_z^{CS}(t)$$

in both magnitude and phase.

That is not surprising. It is a direct consequence of the different vertical-load closure.

Which phase direction would make sense?

If your model's front F_z **leads** CarSim, that is very consistent with your current formulation.

Why? Because your direct term:

$$Y_1 q_1 / t_1$$

acts immediately once lateral force appears. CarSim's tyre compression/suspension path can lag because the body and wheel vertical deflections need time to develop.

Also, your damping term:

$$c_{\phi 1} \dot{\phi} / t_1$$

can create a phase-leading component relative to roll angle. Depending on how large this term is, your model's F_z can peak earlier than CarSim.

If your model's front F_z **lags** CarSim, then I would suspect the opposite: your roll dynamics, roll damping, or the $\phi, \dot{\phi}$ contribution is too slow or too dominant relative to the direct $Y_1 q_1$ term. In that case, q_1 may be too small relative to $k_{\phi 1}, c_{\phi 1}$, or your roll inertia/stiffness/damping combination may not match CarSim's effective roll response.

The key diagnostic: do not look first at FL alone

For phase, looking only at $F_{z,FL}$ can be misleading because:

$$F_{z,FL} = 0.5 Z_1 - \Delta Z_1$$

So FL contains two different dynamic pieces:

1. front axle total load Z_1, affected by longitudinal/pitch load transfer;
2. front left-right split ΔZ_1, affected by lateral load transfer.

Instead, compare these separately:

$$F_{zf,\text{sum}} = F_{z,FL} + F_{z,FR}$$

$$F_{zf,\text{split}} = F_{z,FR} - F_{z,FL}$$

The front split is the better signal for q_1:

$$F_{zf,\text{split}} = 2 \Delta Z_1$$

If the phase difference is mostly in $F_{zf,\text{split}}$, then the issue is lateral load-transfer dynamics.

If the phase difference is mostly in $F_{zf,\text{sum}}$, then the issue is pitch/longitudinal transfer, aero, or CarSim load-channel definition.

Decompose your model's front load split

For your model, plot these three components:

$$\Delta Z_{1,\text{geo}} = \frac{Y_1 q_1}{t_1}$$

$$\Delta Z_{1,\text{spring}} = \frac{k_{\phi 1} \phi}{t_1}$$

$$\Delta Z_{1,\text{damper}} = \frac{c_{\phi 1} \dot{\phi}}{t_1}$$

Then compare each one's phase against CarSim's:

$$\Delta Z_{1,CS} = \frac{F_{z,FR}^{CS} - F_{z,FL}^{CS}}{2}$$

This will tell you what is causing the phase difference.

My expectation is:

- the $Y_1 q_1 / t_1$ term will be close in phase to lateral force;
- the $k_{\phi 1} \phi / t_1$ term will lag lateral acceleration/lateral force;
- the $c_{\phi 1} \dot{\phi} / t_1$ term may lead the roll-angle term;
- CarSim's compression-based F_z will behave like a combination of these, plus missing tyre/unsprung vertical dynamics.

What this means for identification

This is important:

Do not try to fix both magnitude and phase by changing only q_1.

That can overfit one sine frequency and produce a bad MPC model.

A better approach is:

1. Use slow ramp or very low-frequency sine data to identify q_1, q_2.
   At low frequency, phase effects are smaller, so q_1, q_2 mostly represent quasi-static lateral load transfer.
2. Use medium-frequency sine data to check phase.
   If magnitude is good but phase is still wrong, the issue is dynamic: roll damping, suspension damping, missing tyre vertical compliance, or missing unsprung/heave states.
3. Only then consider fitting $c_{\phi 1}, c_{\phi 2}$, or adding a simple effective dynamic correction.

For your current reduced MPC model, I would first keep the model simple and identify:

$$q_1, q_2$$

from quasi-steady/low-frequency data.

Then validate on sine tests.

If the phase difference is unacceptable for MPC prediction, then q_1, q_2 alone are not enough. You may need either:

- better roll damping calibration $c_{\phi 1}, c_{\phi 2}$;
- effective tyre-stiffness-condensed wheel rates;
- a simple first-order vertical-load lag model;
- or, for the most physical solution, explicit wheel-heave/tyre-compression states.

My updated judgement

Yes, the phase difference is very likely due to the different Fz calculation routes.

But the detailed interpretation is:

- **Magnitude mismatch**: likely q_1, front effective load-transfer height, and possibly front roll stiffness/damping distribution.
- **Phase mismatch**: likely missing vertical compliance dynamics, different roll dynamics, tyre compression path in CarSim, and the algebraic nature of your current Fz equations.
- **Changing q_1 alone**: can fix amplitude, but usually cannot rigorously fix phase.

So your observation actually supports the conclusion that your model should be calibrated in two stages:

1. quasi-static lateral load-transfer calibration for q_1, q_2;
2. dynamic validation/calibration for roll damping or missing vertical-load dynamics.

資料來源

Please double check your ideas and arguments about Fz phase difference betwen my model and Carsim and how it will affect model calibration.

已思考 19s

Yes — after double-checking, the phase-difference argument is valid, but it needs one refinement:

The phase difference can be caused by the different F_z calculation route, but it is not caused simply by "CarSim uses tyre stiffness and your model does not." More precisely, it is caused by the fact that CarSim has a vertical compliance/dynamics path, while your current model reconstructs F_z algebraically from force balance, roll/pitch states, and effective load-transfer parameters.

Your model's tyre-load equations are algebraic constraints. The lifted loads $F_{z,FL}, F_{z,FR}, F_{z,RL}, F_{z,RR}$ are solved at every RHS evaluation so that `h_load = 0`; they are not integrated as vertical dynamic states.

run_oc_simple_time_sim_arbitrary In `my_model.m`, front load split is:

$$\Delta Z_1 = \frac{Y_1 q_1 + k_{\phi 1} \phi + c_{\phi 1} \dot{\phi}}{t_1}$$

and:

$$F_{z,FL} = 0.5 Z_1 - \Delta Z_1, \qquad F_{z,FR} = 0.5 Z_1 + \Delta Z_1$$

so front F_z is a direct mixture of Y_1, ϕ, and $\dot{\phi}$, plus the front axle total-load term Z_1. my_model

The important correction

Earlier I said q_1 mainly changes amplitude, not phase. That is mostly true, but the more rigorous statement is:

Changing q_1 cannot create a true dynamic delay, but it can change the apparent phase of the total front load split because it changes the weighting between the direct $Y_1 q_1$ term and the roll terms $k_{\phi 1} \phi + c_{\phi 1} \dot{\phi}$.

That is important.

Your model's front split is:

$$\Delta Z_1 = \underbrace{\frac{Y_1 q_1}{t_1}}_{\text{direct / near lateral-force phase}} + \underbrace{\frac{k_{\phi 1} \phi}{t_1}}_{\text{roll-angle phase}} + \underbrace{\frac{c_{\phi 1} \dot{\phi}}{t_1}}_{\text{roll-rate phase}}$$

These three components do not generally have the same phase in a sine steer test. Therefore, changing q_1 changes the vector sum of these components and can move the phase of ΔZ_1 somewhat.

But it still cannot reproduce arbitrary CarSim phase behavior if CarSim's front F_z contains tyre vertical compliance, unsprung/wheel vertical motion, suspension compression dynamics, or filtering effects.

So the refined conclusion is:

q_1 can affect both magnitude and apparent phase of your model's front F_z, but it should not be used as the only parameter to force-match phase across sine frequencies.

Why the phase difference is plausible

CarSim's load route is closer to:

$$F_{z,i}^{CS} = K_{t,i} \delta_{t,i} + C_{t,i} \dot{\delta}_{t,i} + \text{contact/suspension effects}$$

where tyre compression $\delta_{t,i}$ comes from sprung-body motion, wheel/unsprung motion, suspension kinematics, and road contact geometry.

Your current model has no tyre compression state, no wheel heave, and no body heave. It has roll and pitch states, but the tyre loads themselves are algebraic outputs of the force/moment balance. Your code explicitly lifts only the four vertical loads as algebraic variables, not acceleration states or vertical-deflection states.

my_model

So if CarSim's F_z lags or leads your model's F_z, that is not surprising. It is a structural model difference.

What this means for calibration

This changes the calibration strategy.

If you fit q_1 only from one sine test using raw time-domain $F_{z,FL}$, the optimizer may choose a q_1 that partly compensates for a missing dynamic phase effect. That could give a good fit at that sine frequency but poor prediction at another frequency or steering maneuver.

So the calibration should be separated into two problems:

1. Quasi-static lateral load-transfer calibration

Use slow ramp steer or very low-frequency sine steer.

Goal:

$$q_1, q_2$$

should match the **low-frequency / quasi-static load split**:

$$F_{z,FR} - F_{z,FL}$$

$$F_{z,RR} - F_{z,RL}$$

This is where q_1, q_2 have the clearest physical meaning as effective lateral load-transfer heights.

2. Dynamic phase validation

Use medium-frequency sine steer and lane-change-type tests.

Goal: check whether the model's front load phase is acceptable after q_1, q_2 are calibrated.

If magnitude is good but phase is still wrong, then changing q_1 further is not the rigorous fix. The candidates are:

$$c_{\phi 1}, c_{\phi 2}$$

$$k_{\phi 1}, k_{\phi 2}$$

roll inertia / roll dynamics,

effective tyre-stiffness-condensed wheel rates,

or adding a simple vertical-load dynamic correction.

Your roll stiffness and damping are currently constructed from the wheel vertical stiffness/damping values, with front/rear split through `kphi1,kphi2,cphi1,cphi2` ; ARBs are set to zero in your parameters.

my_params So if the phase mismatch is significant, these parameters matter.

Do not calibrate from FL alone

This is very important.

Front-left tyre load is:

$$F_{z,FL} = 0.5Z_1 - \Delta Z_1$$

So its phase contains both:

1. front axle total load Z_1, affected by longitudinal/pitch effects;
2. front lateral split ΔZ_1, affected by lateral load transfer.

Even with zero drive torque, Z_1 can still move due to longitudinal force X, pitch acceleration, drag, and transient effects.

So for phase analysis, compare these separately:

$$F_{zf,\text{sum}} = F_{z,FL} + F_{z,FR}$$

$$F_{zf,\text{split}} = F_{z,FR} - F_{z,FL}$$

If the phase difference is mainly in:

$$F_{zf,\text{split}}$$

then it is a lateral load-transfer dynamic issue.

If the phase difference is mainly in:

$$F_{zf,\text{sum}}$$

then it is more likely pitch/longitudinal transfer, front axle total load, aero, or CarSim output-channel definition.

How phase difference should affect your identification objective

For q_1, q_2, I would **not** minimize raw time-domain FL/FR error as the only objective.

Instead, use something like:

$$J = J_{\text{quasi-static split}} + \lambda J_{\text{dynamic validation}}$$

where the main fitting target is still:

$$F_{z,FR} - F_{z,FL}$$

and:

$$F_{z,RR} - F_{z,RL}$$

For sine tests, you can also compare amplitude and phase explicitly. For example, at the steering frequency f, compute the complex Fourier coefficient of front load split:

$$G_f = \frac{\mathcal{F}\{F_{z,FR} - F_{z,FL}\}(f)}{\mathcal{F}\{\delta\}(f)}$$

Then compare:

$$|G_f^{my}| \quad \text{and} \quad |G_f^{CS}|$$

and:

$$\angle G_f^{my} \quad \text{and} \quad \angle G_f^{CS}$$

If q_1 can match low-frequency magnitude but cannot match phase at 0.5–1.0 Hz, that is evidence that q_1 alone is insufficient.

The rigorous calibration workflow I would use now

1. **First identify q_1, q_2 using slow steering ramp / low-frequency sine data.**
   Use load split, not FL alone.
2. **Keep q_1, q_2 fixed. Then check front split phase in sine tests.**
   Test 0.2 Hz, 0.5 Hz, and 1.0 Hz separately.
3. **Decompose your model's front split into three components:**

$$\frac{Y_1 q_1}{t_1}, \quad \frac{k_{\phi 1}\phi}{t_1}, \quad \frac{c_{\phi 1}\dot{\phi}}{t_1}$$

If the CarSim front split is closer in phase to Y_1, your model's roll contribution may be too dominant or too slow.

If the CarSim front split is closer in phase to ϕ, your direct $Y_1 q_1$ contribution may be too dominant.

If the phase error changes strongly with frequency, the missing tyre/suspension vertical dynamics are probably important.

4. **Only after that decide whether to calibrate $c_{\phi 1}$ or add dynamics.**

I would not jump directly to identifying $q_1, q_2, c_{\phi 1}, k_{\phi 1}$ all together from a single sine test. That will probably overfit.

Final double-checked conclusion

Your interpretation is correct: **the phase difference can very plausibly come from the different tyre-load computation routes.**

But for calibration, the key point is:

Use q_1, q_2 to fit the quasi-static or low-frequency lateral load-transfer magnitude. Do not expect q_1, q_2 alone to fix a dynamic phase mismatch.

If the phase mismatch remains after q_1, q_2 calibration, then the model is telling you that the missing dynamic vertical-load path matters. For MPC, you then have three choices:

1. accept the phase error if it is small over the MPC-relevant bandwidth;
2. calibrate roll stiffness/damping/effective wheel rates to improve phase;
3. add a simple vertical-load dynamic correction or explicit vertical DOFs if the phase error is large.

資料來源

What if later on the relaxation length of tyre is turned back on in Carsim? What if the vertical dynamics of the vehicle model's COM and wheel heaves are excited in certain maneuvers strongly in Carsim? What if the data in Carsim is added with artificial noise to mimic the real data in real world? Can the calibration methods you suggested still work or need modifications? Overall, what if Carsim's model turned on more 'high-fidelity' features but my MPC model does not consider.

I'll treat this as a robustness/generalization question: what changes when the "truth model" has dynamics/noise that the MPC prediction model intentionally omits.

已思考 15s

Yes, the calibration methods can still work, but **not unchanged**. The more high-fidelity features CarSim turns on, the more your calibration must be treated as **reduced-model projection**, not "parameter identification of the true physical system."

Your MPC model currently has no tyre relaxation states, no tyre compression states, no COM heave, and no wheel-heave DOFs. Its tyre loads are algebraic quantities constrained by the Chapter-9-style load equations, with q_1, q_2, H entering as effective load-transfer parameters. my_model So when CarSim turns on dynamics that your model does not contain, you cannot expect one pair of constants q_1, q_2 to reproduce all amplitudes and phases across all maneuvers.

The calibration still has value, but the target changes:

$$\text{not: identify true CarSim physics}$$

$$\text{but: find the best reduced MPC model over the bandwidth and maneuvers relevant to control}$$

1. If tyre relaxation length is turned back on in CarSim

Then F_x, F_y will no longer be instantaneous algebraic functions of slip. They will have lag:

$$\alpha, \kappa \to F_x, F_y$$

with frequency-dependent amplitude and phase.

That affects F_z indirectly because your front load split uses:

$$\Delta Z_1 = \frac{Y_1 q_1 + k_{\phi 1}\phi + c_{\phi 1}\dot{\phi}}{t_1}$$

and Y_1 comes from front tyre forces. my_model

So if CarSim relaxation length is on, the F_z mismatch may come from two places:

1. horizontal tyre-force lag, F_y phase/magnitude mismatch;
2. vertical-load-path mismatch.

In that case, do **not** use dynamic sine data directly to identify q_1, q_2 until you know whether F_y itself matches.

The modified workflow should be:

1. First compare F_x, F_y between your model and CarSim.
2. If F_x, F_y lag in CarSim, identify or approximate tyre relaxation separately.
3. Only then identify q_1, q_2 from low-frequency or quasi-steady load split.

If your MPC model will not include relaxation length, then do not force q_1, q_2 to compensate for relaxation lag. That would corrupt the load-transfer parameters.

2. If COM heave and wheel-heave dynamics are strongly excited

This is more serious.

Your current model cannot reproduce vertical modal dynamics that it does not contain. CarSim may produce tyre-load oscillations from:

$$\text{sprung heave}$$

$$\text{wheel hop}$$

$$\text{tyre vertical compression}$$

$$\text{suspension compression/rebound}$$

$$\text{road vertical excitation}$$

Your model has roll and pitch states, but no explicit vertical tyre compression or wheel-heave state. The four tyre loads are algebraic variables, solved to satisfy the load equations; they are not dynamic states with their own vertical compliance. my_model

So if CarSim's wheel-hop or heave modes are strongly excited, then q_1, q_2 cannot fix the mismatch rigorously. They may reduce the average error, but they cannot reproduce frequency-dependent vertical oscillations.

In that case, use one of these modifications:

Option A — bandwidth-limited calibration

If MPC only needs low-frequency load prediction, low-pass filter CarSim F_z before calibration.

For example, fit to:

$$F_z^{CS,LP}$$

instead of raw high-frequency CarSim F_z.

Then q_1, q_2 represent the low-frequency effective load transfer useful for MPC, not wheel-hop dynamics.

Option B — exclude highly vertical maneuvers from q1/q2 identification

Use smooth steering, flat road, low vertical excitation data for identifying q_1, q_2.

Then validate separately on rougher or more aggressive maneuvers.

Option C — add a simple vertical-load dynamic correction

If phase/load oscillation matters for MPC but you do not want full wheel-heave states, add a first-order or second-order correction:

$$\tau_{Fz}\dot{F}_{z,\text{eff}} = F_{z,\text{alg}} - F_{z,\text{eff}}$$

or axle-wise:

$$\tau_f\dot{\Delta Z}_{1,\text{eff}} = \Delta Z_{1,\text{alg}} - \Delta Z_{1,\text{eff}}$$

$$\tau_r\dot{\Delta Z}_{2,\text{eff}} = \Delta Z_{2,\text{alg}} - \Delta Z_{2,\text{eff}}$$

This keeps the MPC model small but allows some phase lag.

Option D — add physical vertical DOFs

If wheel-hop/heave dynamics are important for the maneuvers you care about, the rigorous solution is to add:

- sprung heave,
- unsprung/wheel heave,
- suspension deflection,
- tyre compression,
- tyre vertical stiffness/damping.

That is the structurally correct way to match CarSim's vertical F_z path.

3. If artificial noise is added to CarSim data

Then the calibration can still work, but you need robust identification.

Noise affects this especially badly if you compute:

$$\dot{\phi}, \quad \ddot{\theta}, \quad a_x, \quad a_y$$

from numerical differentiation. Avoid differentiating noisy signals unless you filter/smooth them first.

For noisy data, modify the method like this:

1. Use repeated tests and average results.
2. Fit load split, not raw FL alone:

$$F_{z,FR} - F_{z,FL}$$

$$F_{z,RR} - F_{z,RL}$$

3. Use robust loss instead of pure least squares, for example Huber loss.
4. Use low-pass filtering consistent with MPC bandwidth.
5. Use bounds and regularization:

$$0 \le q_1, q_2 \le h$$

or a tight physically reasonable interval.
6. Do not fit high-frequency noise; fit low-frequency load-transfer trends.
7. Use train/validation split: identify on some maneuvers, validate on different maneuvers.

With noise, do not identify too many parameters at once. Start with q_1, q_2. Only add $c_{\phi 1}, c_{\phi 2}$ or dynamic lag parameters if the data quality is good enough.

4. If CarSim turns on many high-fidelity features

Then your MPC model becomes a **low-order surrogate** of CarSim, not a physically complete replica.

This is normal in MPC. The goal is not to match every high-frequency feature. The goal is to predict the states/forces that matter over the MPC horizon and bandwidth.

So you should decide what your MPC needs:

If MPC needs low-frequency handling/load-transfer prediction

Then keep the model simple. Calibrate q_1, q_2 using smooth, low-frequency steering data. Validate that it predicts the trend and peak loads reasonably.

If MPC needs aggressive transient drifting / fast load transfer

Then phase matters more. You may need to calibrate roll damping, front/rear load-transfer dynamics, or add an effective F_z lag.

If MPC must operate over bumps, kerbs, high vertical excitation, or rough roads

Then your current model is probably structurally insufficient. You need at least some vertical DOFs or an external correction/observer.

5. How I would modify the calibration workflow

The final robust workflow should be:

Stage 1 — keep CarSim simple

Disable tyre relaxation, steering dynamics, bump stops, ARB, and unnecessary vertical excitations.

Use flat-road steering-only tests.

Identify quasi-static q_1, q_2.

This gives your clean baseline.

Stage 2 — turn on one high-fidelity feature at a time

Do not turn everything on at once.

Turn on tyre relaxation only. Compare F_x, F_y.
Turn on tyre vertical compliance effects. Compare F_z.
Turn on suspension nonlinearities. Compare again.
Add noise last.

This tells you which feature breaks the reduced model.

Stage 3 — separate calibration targets by bandwidth

Use low-frequency data for q_1, q_2.

Use medium-frequency data for roll damping or F_z lag.

Use high-frequency data only for validation unless you add vertical DOFs.

Stage 4 — validate with realistic maneuvers

After calibration, validate on maneuvers not used in fitting:

- sine steer at different frequencies,
- ramp steer,
- double lane change,
- brake-and-steer,
- throttle-lift-and-steer,
- high lateral acceleration cornering.

Stage 5 — decide whether the model is good enough for MPC

Use prediction metrics over the MPC horizon, not just full-run time-domain matching.

For example, if your MPC horizon is 1 second, check:

$$F_z(t : t + 1s)$$

prediction error over many starting points.

That is more relevant than matching an entire 20-second CarSim run.

6. What should not be done

I would avoid these:

1. Do not fit q_1, q_2 to noisy high-frequency CarSim F_z directly.
2. Do not use q_1, q_2 to compensate for tyre relaxation length.
3. Do not fit one sine frequency and assume it generalizes.
4. Do not identify $H, q_1, q_2, k_\phi, c_\phi$ all together from a single maneuver.

5. Do not expect a no-heave/no-wheel-hop model to match wheel-hop load oscillations.

7. Overall answer

Yes, the calibration method still works, but it must be modified depending on which high-fidelity effects are turned on.

For your current MPC model, the best philosophy is:

$$\text{Calibrate the reduced model only over the frequency range and maneuver class that MPC needs.}$$

Use q_1, q_2 for quasi-static/low-frequency lateral load-transfer calibration.

Use tyre relaxation parameters or a force lag model for F_x, F_y phase.

Use roll damping or simple F_z lag for moderate F_z phase mismatch.

Use vertical DOFs only if CarSim's heave/wheel-hop dynamics are important for your intended MPC maneuvers.

So if CarSim becomes more high-fidelity while your MPC model remains reduced, the answer is not "calibrate harder." The answer is:

calibrate selectively, validate honestly, and add only the missing dynamics that materially affect MPC prediction.

資料來源

Please double check your suggestions, make sure your suggestions are rigorous. Think carefully and do not hurry to give me answers before you really think hard about it.

I'll re-audit the logic from first principles: model structure, identifiability, excitation design, and what changes when CarSim contains dynamics your MPC model omits.

After re-checking, the strongest correction is this: once CarSim has unmodelled dynamics/noise, the identified parameters become **frequency- and dataset-dependent pseudo-parameters**, not unique physical constants. That affects how much trust we should place in q_1, q_2.

已思考 38s

Yes. After thinking it through again, I would **keep the core idea** but state it more rigorously and with stronger limitations.

The rigorous conclusion is:

$$q_1, q_2 \text{ can still be calibrated, but only as reduced-model effective parameters, not as true CarSim physical parameters.}$$

Once CarSim has tyre relaxation, vertical tyre compression, COM heave, wheel heave, suspension nonlinearities, and noise, then your identified q_1, q_2 become **dataset-dependent and bandwidth-dependent pseudo-parameters**. They are still useful for MPC if calibrated carefully, but they are no longer unique physical quantities.

Your current model supports this interpretation: it lifts only the four tyre vertical loads as algebraic variables and uses H, q_1, q_2 as runtime load-transfer parameters, not as explicit tyre-compression or wheel-heave states. my_model The front/rear tyre loads are enforced through algebraic load equations using $Y_1, Y_2, \phi, \dot{\phi}, \theta, \dot{\theta}, X$, with front and rear split controlled by $q_1, q_2, k_{\phi 1}, k_{\phi 2}, c_{\phi 1}, c_{\phi 2}$. my_model The simulator then solves these tyre-load algebraic equations at RHS/stage evaluations rather than integrating tyre compression or wheel vertical motion. run_oc_simple_time_sim_arbitrary

1. The previous suggestions are valid only under a precise condition

The earlier method is rigorous if the goal is:

$$\text{Find the best reduced MPC model over the operating range that MPC needs.}$$

It is **not** rigorous if the goal is:

$$\text{Recover the true physical CarSim no-roll-center heights.}$$

That second goal is impossible with your current model once CarSim uses higher-fidelity vertical dynamics, because your model does not contain the same states.

So the correct mathematical interpretation is:

$$(q_1^{\backslash *}, q_2^{\backslash *}) = \arg\min_{q_1, q_2} \mathbb{E}_{\mathcal{D}} \left[\| F_{z,\text{my}}(q_1, q_2) - F_{z,\text{CarSim}} \|_W^2 \right]$$

where $\mathcal{D}$ is the chosen maneuver dataset and W is the chosen weighting/filtering. Therefore, $q_1^{\backslash *}, q_2^{\backslash *}$ depend on:

- maneuver type,
- speed range,
- steering frequency,
- lateral acceleration range,
- whether CarSim tyre relaxation is on,
- whether vertical modes are excited,
- noise level,
- filtering,
- objective weighting.

That is not a weakness if you accept them as **effective MPC-model parameters**. But it is dangerous if you interpret them as physical constants.

2. If tyre relaxation length is turned back on

Then my previous recommendation needs modification.

With relaxation length off, F_x, F_y are closer to algebraic functions of slip, like your model. With relaxation length on, CarSim has additional dynamics:

$$\alpha, \kappa \to F_x, F_y$$

with amplitude attenuation and phase lag.

Because your load-transfer equation uses Y_1 and Y_2,

$$\Delta Z_1 = \underset{t_1}{Y_1 q_1 + k_{\phi 1}\phi + c_{\phi 1}\dot{\phi}}$$

$$\Delta Z_2 = \underset{t_2}{Y_2 q_2 + k_{\phi 2}\phi + c_{\phi 2}\dot{\phi}}$$

a phase difference in F_y will automatically create a phase difference in F_z. my_model

So if relaxation length is on, do **not** immediately recalibrate q_1, q_2 from dynamic sine data. That would allow q_1, q_2 to compensate for missing tyre-force dynamics, which is physically wrong and may not generalize.

The rigorous procedure should be:

1. First compare F_x, F_y phase and magnitude.
2. If F_x, F_y mismatch because of relaxation, either:
   - add a simple tyre-force lag/relaxation approximation to the MPC model, or
   - restrict q_1, q_2 identification to very low-frequency/quasi-steady data where relaxation effects are small.
3. Only then identify q_1, q_2 from load split.

If the MPC model will **not** include relaxation length, then q_1, q_2 should be fitted to the **low-frequency projected behavior**, not to high-frequency CarSim F_z phase.

3. If COM heave and wheel-heave dynamics are strongly excited

Then q_1, q_2 alone are structurally insufficient.

This is the strongest limitation.

CarSim's vertical load may contain components like:

$$F_z^{CS} = F_{z,\text{load transfer}} + F_{z,\text{heave}} + F_{z,\text{wheel hop}} + F_{z,\text{tyre compression}} + F_{z,\text{suspension damper}}$$

Your model can represent the first part approximately, but not the rest. Your current state vector includes roll and pitch states, but no sprung heave, unsprung heave, tyre compression, or wheel vertical velocity. The only algebraic variables are the four tyre vertical loads. my_model

So if vertical modes are strongly excited, calibration of q_1, q_2 can at best match the **average/low-frequency envelope**, not the instantaneous high-frequency F_z.

Therefore, if the CarSim maneuver strongly excites wheel hop or heave, use one of these strategies:

Case A — MPC does not need high-frequency vertical load

Low-pass filter CarSim F_z and identify q_1, q_2 from the low-frequency load split.

Then the target is:

$$F_{z,\text{CarSim}}^{LP}$$

not raw F_z.

This is rigorous if your MPC bandwidth is also low and you do not need to predict wheel-hop-scale load oscillations.

Case B — MPC needs dynamic Fz phase but not full vertical dynamics

Add a small effective load dynamics model, for example:

$$\tau_f \dot{\Delta Z}_{1,\text{eff}} = \Delta Z_{1,\text{alg}} - \Delta Z_{1,\text{eff}}$$

$$\tau_r \dot{\Delta Z}_{2,\text{eff}} = \Delta Z_{2,\text{alg}} - \Delta Z_{2,\text{eff}}$$

or a second-order axle load-transfer lag. This is not fully physical, but it is a controlled reduced-order approximation.

Case C — MPC must handle bumps, kerbs, wheel hop, or strong vertical load oscillation

Then your model needs more states. You would need at least some of:

- sprung heave,
- unsprung heave/wheel vertical motion,
- suspension deflection,
- tyre compression,
- tyre vertical stiffness/damping.

In this case, trying to "calibrate harder" with only q_1, q_2 is not rigorous.

4. If artificial noise is added

The previous suggestion also needs modification under noise.

There are two different cases.

Noise only on measured Fz

If noise is added only to CarSim F_z, then least squares on load split is still basically reasonable, especially with enough data and repeated tests. The estimator variance increases, but the estimate is not necessarily biased.

You should use:

- repeated runs,
- averaging,
- robust loss,
- physically bounded q_1, q_2,
- validation data not used for fitting.

Your existing code already contains a noise Monte Carlo structure for identification, which is a useful direction for checking sensitivity. run_oc_simple_time_sim_arbitrary

Noise on regressors such as $F_y, \phi, \dot{\phi}$

This is more dangerous.

The closed-form diagnostic formula uses quantities such as $Y_1, Y_2, \phi, \dot{\phi}$. Your code computes diagnostic instantaneous/LS q_1, q_2 from $F_z, F_x, F_y, \delta, \phi, \dot{\phi}$, with thresholds to avoid tiny Y denominators. run_oc_simple_time_sim_arbitrary

If $Y_1, Y_2, \phi, \dot{\phi}$ are noisy, ordinary least squares can become biased because the regressors are noisy. This is the classic errors-in-variables problem.

So with noisy CarSim/real data, I would not rely only on closed-form LS. I would use it as a diagnostic, then perform simulation-in-the-loop fitting with:

- robust loss, for example Huber loss;
- low-pass filtering matched to MPC bandwidth;
- bounds $0 \le q_i \le h$ initially;
- multiple maneuvers;
- validation on withheld maneuvers;
- no numerical differentiation of noisy signals unless smoothed carefully.

Offline zero-phase filtering is okay for calibration analysis, but do not confuse that with what a causal online estimator would know.

5. What changes if CarSim becomes more high-fidelity but MPC model stays simple?

Then the calibration problem becomes a **model reduction problem**, not a parameter-identification problem.

That means you should no longer ask:

"Which q_1, q_2 are true?"

You should ask:

"Which q_1, q_2, and possibly which small extra dynamic corrections, give the best MPC-relevant prediction over the horizon and bandwidth I care about?"

For MPC, this is the right criterion:

$$\text{prediction error over MPC horizon}$$

not:

$$\text{perfect matching of a long high-fidelity CarSim time history}$$

For example, if your MPC horizon is 0.5–1.0 s, evaluate many short prediction windows:

$$t_0 \to t_0 + T_h$$

instead of only fitting one 20 s sine test.

6. Revised rigorous calibration workflow

I would now recommend the following workflow.

Stage 0 — define the intended MPC bandwidth

Decide the frequency range your MPC model must predict.

For example:

- low-frequency path tracking: maybe up to 0.5 Hz load-transfer behavior is enough;
- aggressive transient drifting: maybe 1–2 Hz matters;
- kerb/rough-road/load-hop control: wheel-hop frequencies matter, and your current model is not enough.

This decision should come before calibration.

Stage 1 — clean baseline calibration

Turn off CarSim features that your model does not contain:

- tyre relaxation off,
- steering dynamics off,
- ARB off if your model has ARB off,
- bump stops off,
- flat road,
- no strong vertical excitation.

Use slow steering ramp and low-frequency sine.

Identify q_1, q_2 from load split:

$$F_{z,FR} - F_{z,FL}$$

$$F_{z,RR} - F_{z,RL}$$

This gives the clean quasi-static reduced-model calibration.

Stage 2 — turn high-fidelity features on one at a time

Do not turn everything on at once.

1. Turn tyre relaxation on. Check whether F_x, F_y phase changes.
2. Turn vertical tyre/suspension features on. Check F_z phase and high-frequency content.
3. Add noise. Check parameter variance and robustness.
4. Add combined torque/brake/steer. Check generalization.

This isolates which missing physics matters.

Stage 3 — decide whether q_1, q_2 are enough

After Stage 2:

- If only magnitude changes: adjust q_1, q_2.
- If phase changes with frequency: q_1, q_2 alone are not enough.
- If F_y phase changes: add/approximate tyre relaxation or stay low-frequency.
- If F_z high-frequency oscillations appear: add effective F_z dynamics or vertical DOFs.
- If noise dominates: use robust/regularized identification and uncertainty margins.

Stage 4 — simulation-in-the-loop final fit

Use your model, with the same open-loop inputs, and optimize the reduced parameters against CarSim outputs.

But the objective should be bandwidth-aware:

$$J = J_{\text{low-frequency load split}} + \lambda_1 J_{\text{MPC horizon prediction}} + \lambda_2 J_{\text{state/force validation}}$$

Do not heavily penalize high-frequency CarSim F_z that the MPC model structurally cannot reproduce.

Stage 5 — validation

Validate on maneuvers not used for fitting:

- different sine frequencies,
- ramp steer,
- double lane change,
- brake plus steer,
- throttle lift plus steer,
- higher lateral acceleration,
- the maneuvers expected in NMPC tracking.

If the model only fits the calibration maneuver, the identified q_1, q_2 are not useful.

7. Final corrected recommendation

After double-checking, I would state the recommendation like this:

1. **For clean, low-frequency CarSim data**, identifying q_1, q_2 from front/rear load split is rigorous and appropriate.
2. **With tyre relaxation turned on**, do not let q_1, q_2 compensate for F_x, F_y phase lag. Either add tyre relaxation to the MPC model or identify q_1, q_2 only from low-frequency/quasi-steady data.
3. **With strong heave/wheel-hop dynamics**, q_1, q_2 cannot reproduce raw F_z. Use low-pass/projected F_z, add effective load dynamics, or add physical vertical DOFs.
4. **With noise**, closed-form LS is only a diagnostic. Use robust, bounded, simulation-in-the-loop identification and validation. Be especially careful if $Y_1, Y_2, \phi, \dot{\phi}$ are noisy.
5. **With many high-fidelity CarSim features on**, the problem becomes reduced-order model calibration. The goal is not exact CarSim matching, but MPC-relevant prediction over the horizon and bandwidth.

So the previous ideas are directionally correct, but the rigorous version is more conditional:

$$q_1, q_2 \text{ should be identified from low-frequency load-transfer behavior; unmodelled high-frequency dynamics should not be absorbed blindly into } q_1, q_2.$$

If the omitted high-fidelity features materially affect the MPC-relevant prediction bandwidth, then the MPC model itself must be extended, at least with tyre relaxation or effective F_z dynamics, and possibly with vertical DOFs.

資料來源