Design tasks

10:00	10:45	DTS	M3	Design Task 1, with Compendium references and Introduction	BJ, YH, FB
11:00	11:45	DTS	M3	Design task 1	YH, FB

Administration (points, hand-in, etc): See Course Memo.

...and, these will take well care of you:

- Assistants for Design task 1, Longitudinal, 10 p:
  - **Lead:** Yansong.Huang@chalmers.se +46 73 860 47 19 ...[YH]
  - Support: Fredrik.Bruzelius@chalmers.se, +46 73 1431 365 ...[FB]
- Assistants for Design task 2, Lateral, 25 p:
  - **Lead**: Sachin.Janardhanan@chalmers.se +46 76 553 8053 ...[SJ]
  - Support: Yansong.Huang@chalmers.se +46 73 860 47 19 ...[YH]
  - For Simulator Lab at CASTER:
    - Admin: Sam.Azadi@casterchalmers.se ...[SA]
    - Instructors: 4-6 personer ...[instr]
- Assistants for Design task 3, Vertical, 15 p:
  - **Lead:** Fredrik.Bruzelius@chalmers.se, +46 73 1431 365 ...[FB]
  - Support: Sachin.Janardhanan@chalmers.se +46 76 553 8053 ...[SJ]

Yansong will soon do an introduction of Design task 1,
but Bengt will first point out useful parts in Compendium ...

Design Tasks

Learning objectives	Reading
<u>Design Task 1: Longitudinal</u> • <u>Functions</u>: Acceleration (uphill, various road friction) • <u>What to engineer</u>: Distribution of propulsion between front and rear axle (FWD/RWD) • <u>Method</u>: Simulation • <u>Tools</u>: Matlab Symbolic toolbox, "Home-coded" time-integration (for conceptual understanding of simulation)	• Figure 2-21, 2.2.3.4.1 Magic Formula Tyre Model, Eq [2.1] • 1.5.4.1.1 General Mathematics Tools • 1.5.1.1.3 Physical Modelling • 1.5.1.1.4 Mathematical Modelling, 1.5.2.1 Free-Body Diagrams • 1.5.1.1.5 Explicit Form Modelling, 1.5.1.1.6 Computation • Figure 3-24, Eq [3.13] • 3.5.2.5 Traction Control, TC *
<u>Design Task 2: Lateral</u> • <u>Functions</u>: Yaw balance in steady state high speed, Step steer response, Brake in curve • <u>What to engineer</u>: Distribution of roll-stiffness and brake force between front and rear axle • <u>Method</u>: Simulation, Driving experience, model integration and log data analysis • <u>Tools</u>: Simulink (for learning one commonly used tool for simulation), Motion platform driving simulator (for driving experience and log data analysis)	• Figure 4-11 • Figure 4-15 • Figure 4-19 • 4.3.6 Steady State Cornering Gains * • Eq [1.1][4.17] • Eq [4.18] • Figure 4-47 • 1.5.1.1.4.5 Affine and Linear form (ABCD form) • Figure 4-38, Eq [4.39] • Eq [2.47]
<u>Design Task 3: Vertical</u> • <u>Functions</u>: Comfort for stationary vibrations, Road grip due to stationary varying vertical tyre force • <u>What to engineer</u>: Wheel suspension stiffness and damping • <u>Method</u>: Frequency analysis • <u>Tools</u>: Matlab (for learning one commonly used tool for matrix computations)	• Figure 5-1, Figure 5-12, [5.44] • Figure 5-3, Eq [5.4], Eq [5.13], 4.4.3.1.1 Solution with Fourier Transform • Eq [5.45] • Figure 5-5, Figure 5-20

"Common thread" between the 3 Design tasks

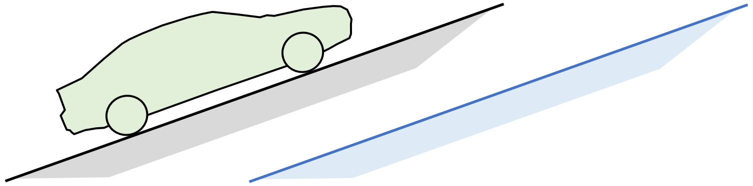
Learning objectives

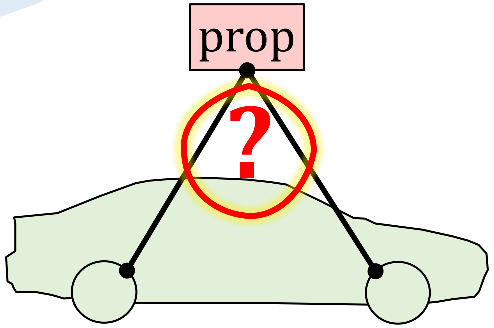
Design Task 1: Longitu
- Functions: Accelerat
- What to engineer: Di front and rear axle (
- Method: Simulation
- Tools: Matlab Symbo integration for con simulation)

Design Task 2: Lateral
- Functions: Yaw balan Step steer response
- What to engineer: Di brake force between
- Method: Simulation, integration and log d
- Tools: Simulink (for for simulation), Moti (for driving experien

Design Task 3: Vertica
- Functions: Comfort f grip due to stationar
- What to engineer: W damping
- Method: Frequency analysis
- Tools: Matlab (for learning one commonly used tool for matrix computations)

Long	Acceleration	Prop	Simulation	Matlab
Lat	Yaw balance	Susp & Brk	Simulation	Simulink & DrivSim
Vert	Comfort & Road grip	Susp	Freq Analysis	Matlab

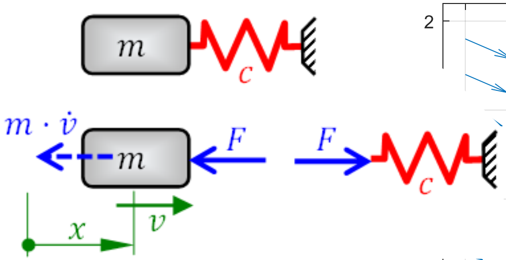


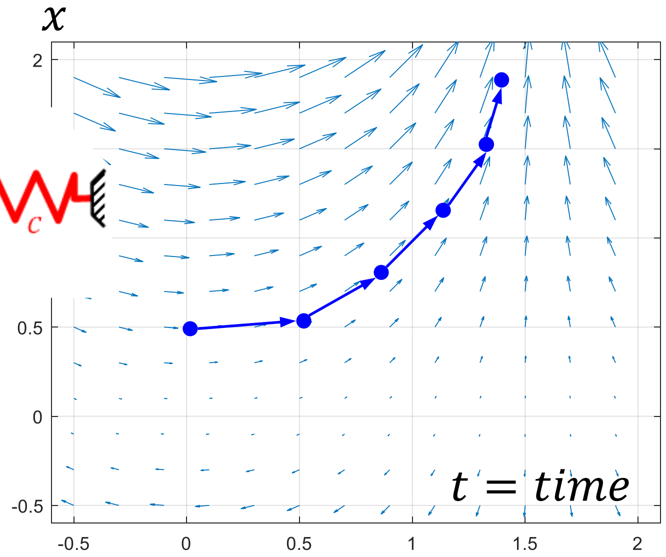


Learning objectives

<u>Design Task 1: Longitudinal</u>

- <u>Functions</u>: Acceleration (uphill, various road friction)
- <u>What to engineer</u>: Distribution of propulsion between front and rear axle (FWD/RWD)
- <u>Method</u>: Modelling and simulation
- <u>Tools</u>: “Home-solved” and “Home-coded” time-integration (for conceptual understanding of simulation)





for $t = 0..t_{end}$ do $x(t + \Delta t) = x(t) + \dot{x}(t) \cdot \Delta t;$

where $\dot{x}(t) = f_{ODE}(x, t);$

where f_{ODE} is solved from $f_{DAE}(\dot{x}, x, t) = f_{DAE}(x(t), t) = 0;$

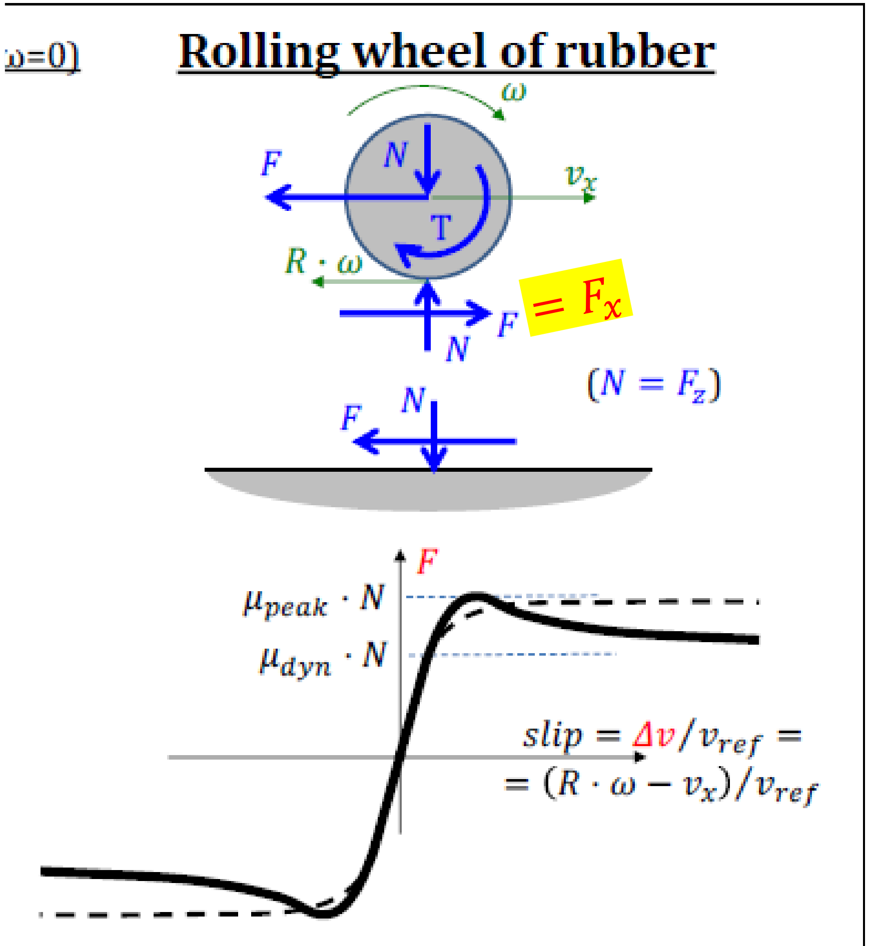
The following slides about these ”recommended readings”.

Quickly now, but more on lectures.

Reading
• Figure 2-21, 2.2.3.4.1 Magic Formula Tyre Model, Eq [2.1] • 1.5.1.1.3 Physical Modelling • 1.5.1.1.4 Mathematical Modelling, 1.5.2.1 Free-Body Diagrams • 1.5.1.1.5 Explicit Form Modelling, 1.5.1.1.6 Computation • Figure 3-24, Eq [3.13] • 3.5.2.5 Traction Control, TC * • 1.5.4.1.1 General Mathematics Tools

Task 1: Tyre Model

In Task 1, we explore a tyre model: A rolling tyre can only develop force if it has "slip, s_x"



- No *stick* mode (if rolling, i.e. $\omega > 0$).
- The friction force, F, is depending on *relative sliding speed*, $\Delta v / v_{ref}$.

: *Friction characteristics.*

2.2.3.4.1 Magic Formula Tyre Model

The most well-known curve fit tyre model is probably sor Hans Pacejka, 1934-2017. It is described, e.g., in (B

$$Force = y(x) = D \cdot \sin(C \cdot \arctan(B \cdot x - E$$

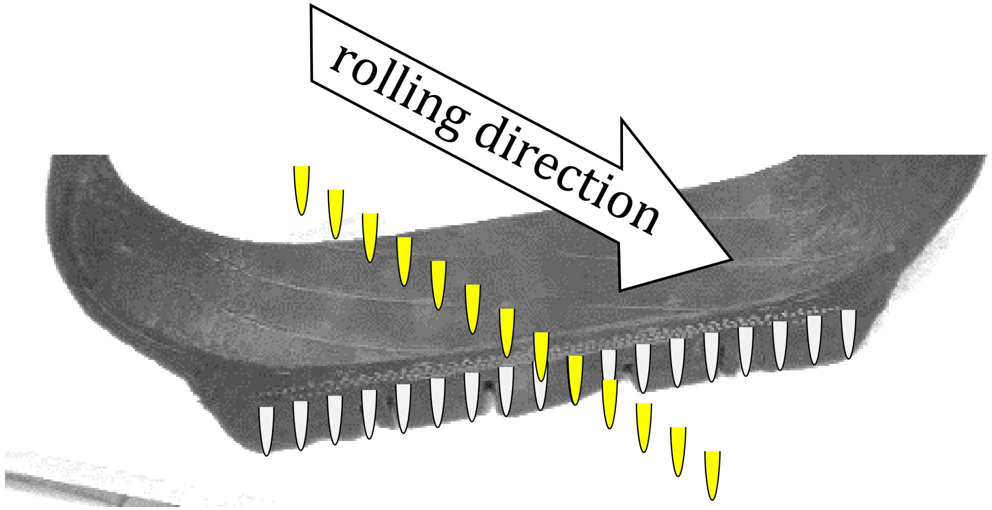
$\frac{F_x}{F_z}$

s_x=longitudinal slip

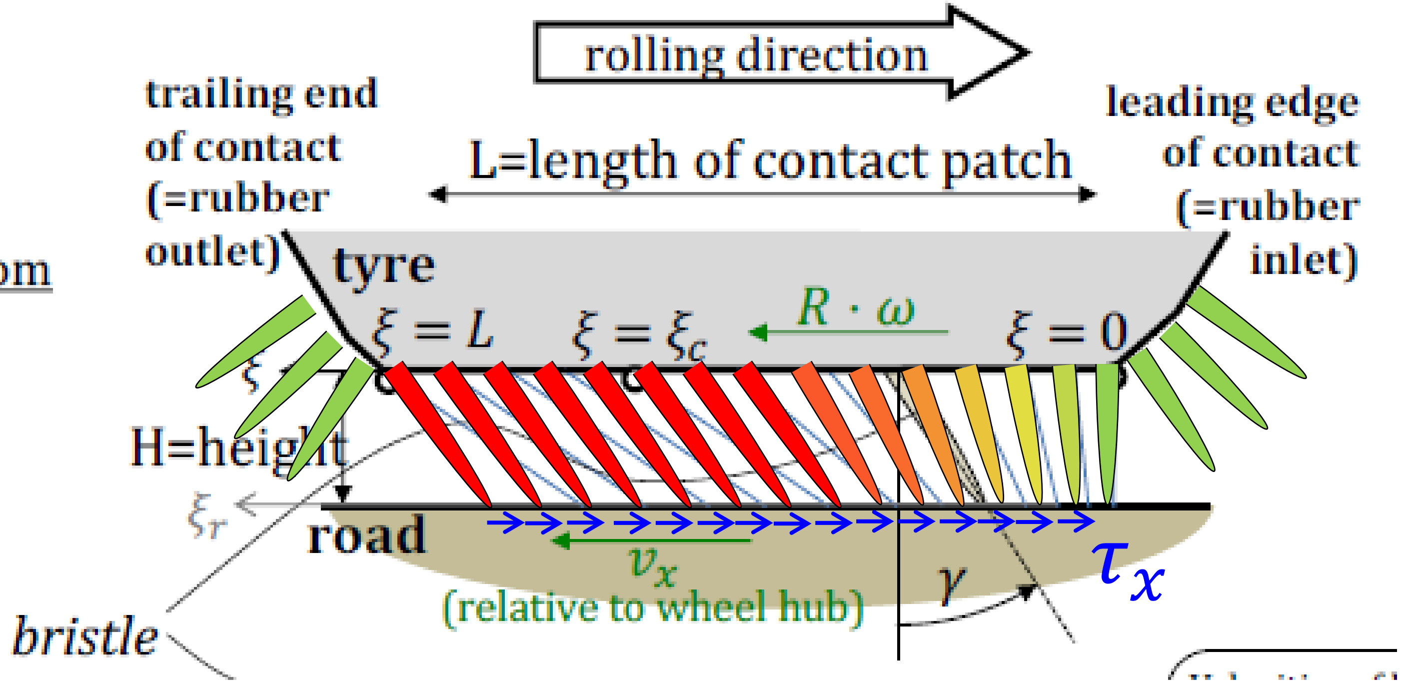
$$s_x = \frac{R \cdot \omega - v_x}{|R \cdot \omega|}; \quad [2.1]$$

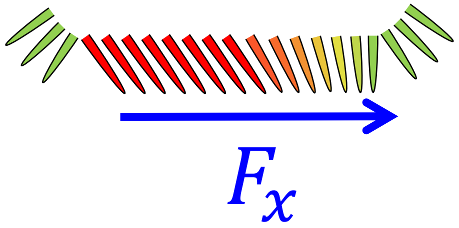
...so far, the Tyre Model is only a Mathematical formula.
So, **why** is $F_x = func(s_x)$?

Task 1: Tire Model



View from side:





Task 2a & 2b: Vehicle Model

1.5.1.1.2 Physical Modelling

1.5.2.1 Free-Body Diagrams

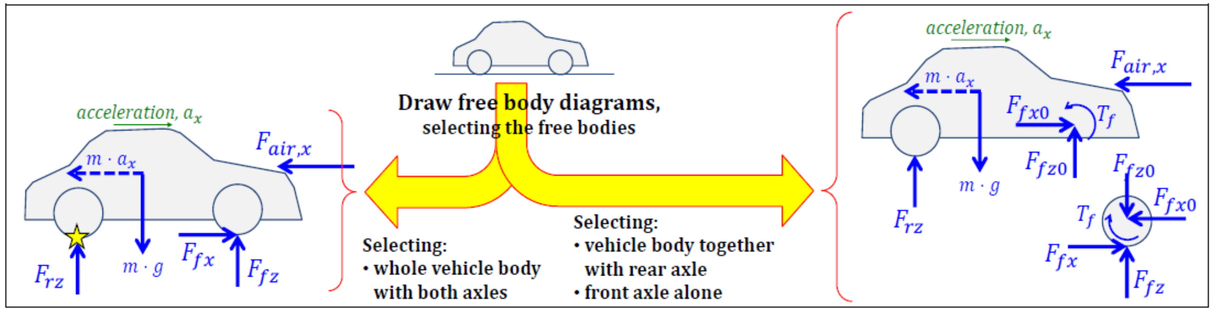
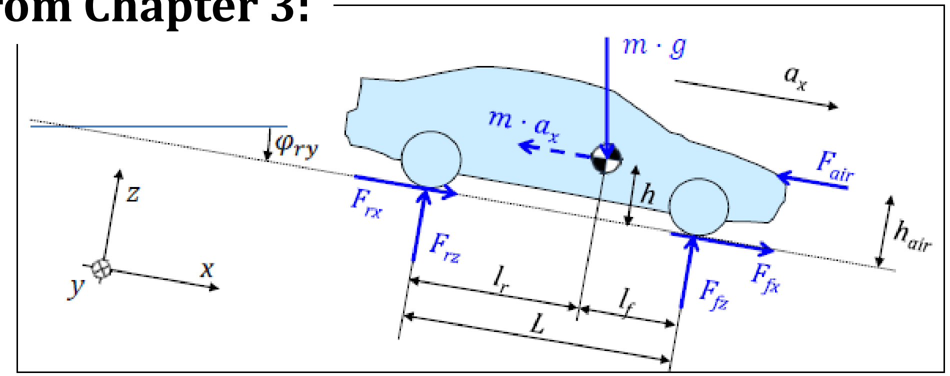


Figure 1-12: Free body diagram. The dashed arrow is a "fictive force". The star is a way to mark around which point(s) moment equilibrium is taken in the later stage "Mathematical model".

From Chapter 3:



1.5.1.1.3 Mathematical Modelling

- $f_{DAE}(\dot{z}, z, t) = 0;$

where $a_x = \dot{v}_x$

From Chapter 3:

Moment equilibrium, around rear contact with ground:

$$-F_{fz} \cdot L + m \cdot g \cdot \left(l_r \cdot cos(\varphi_{ry}) + h \cdot sin(\varphi_{ry})\right) - R_{air} \cdot h_{air} - m \cdot a_x \cdot h = 0; \Rightarrow$$

$$\Rightarrow F_{fz} = m \cdot \left(g \cdot \frac{l_r \cdot cos(\varphi_{ry}) + h \cdot sin(\varphi_{ry})}{L} - a_x \cdot \frac{h}{L}\right) - R_{air} \cdot \frac{h_{air}}{L};$$

Moment equilibrium, around front contact with ground:

$$+F_{rz} \cdot L - m \cdot g \cdot \left(l_f \cdot cos(\varphi_{ry}) - h \cdot sin(\varphi_{ry})\right) - R_{air} \cdot h_{air} - m \cdot a_x \cdot h = 0; \Rightarrow$$

$$\Rightarrow F_{rz} = m \cdot \left(g \cdot \frac{l_f \cdot cos(\varphi_{ry}) - h \cdot sin(\varphi_{ry})}{L} + a_x \cdot \frac{h}{L}\right) + R_{air} \cdot \frac{h_{air}}{L};$$

1.5.1.1.4 Explicit Form Modelling

- $\dot{x} = f_{ODE}(x, u(t), t);\; y = g(x, u(t), t);$

1.5.1.1.5 Computation

$$\dot{x}_{now} = f_{ODE}(x(t_{now}), u(t_{now}), t_{now}); \text{ (Explicit form model)}$$
$$x(t_{now} + \Delta t) = x(t_{now}) + \Delta t \cdot \dot{x}_{now}; \text{ (Derivative approximation)}$$

Your model might first contain an algebraic loop…

Task 2-3: Model of Traction Control, TC

TC is good for maximum F_x and keeping margin for F_y.

3.5.2.5 Traction Control, TC *

*Function definition: **Traction Control** prohibits driver to spin the driven axle(s) in positive direction while accelerating. An extended definition of TC also includes vehicle acceleration requested by other functions than pedal braking, such as CC. TC uses both friction brakes and propulsion system as actuators.*

The purpose of Traction Control is to maximise traction AND to leave some friction for lateral forces for steering and cornering. Traction control is similar to ABS, but for keeping slip below a certain value, typically +(15..20)%.

Traction control can use different ways to control slip, using different actuators. One way is to reduce engine torque, which reduces slip on both wheels on an axle if driven via differential. Another way is to apply friction brakes, which can be done on each wheel individually. Vehi
eral axles can also redistribute propulsion from one a

There is a TC model in the handed out code, but it requires 2 "disclaimers":

1. Our TC is **<u>not</u>** modelled as a controll**<u>er</u>**; you will **<u>not</u>** find a slip error anywhere in the model.
   - Instead, our TC is modelled as if the slip was **ideally controlled** to the exact value of an optimal slip. (I.e. **not** the "natural causality". Or **not** replicating a controller's input and output signals.)

2. Generally, TC can be actuated as either **propulsion torque** reduction, **friction brake** intervention, changed **distribution** of torques between axles or a combination of those.
   - Our TC model has to be thought of as actuated with **axle-individual wheel torque reduction**, and neither with reduction of total torque nor changed distribution of torque between the axles. It can be understood as the function "TC by **friction brake**".

Avoid error prone manual algebraic manipulations

It is good practice to solve from Mathematical model to Explicit form model by hand this one first time. But for future:

1.5.4.1.1 General Mathematics Tools

Contributions from Mats Jonasson, Volvo Cars and Vehicle Dynamics at Chalmers
Examples of tool: Matlab, Matrixx, Python

We will take Matlab as example. Matlab is a commercial computer program for general mathematics. It is developed by Mathworks Inc. Compendium will use some simple Matlab code to describe models in this compendium. The following are useful for dynamic models:

General help function, here shown for function "`inv`" `>> help inv`

Solve linear systems of equations, $\boldsymbol{A} \cdot \boldsymbol{x} = \boldsymbol{b}$: `>> x=inv(A)*b;`

Solve non-linear systems of equations, $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$: `>> x=fsolve('f',...);`

Solve ODE as initial value problems, $\dot{\boldsymbol{x}} = \boldsymbol{f}(t, \boldsymbol{x})$: `>> x=ode23('f',x0,...);`

Find Eigen vectors (V) and Eigen values (D) to systems: $\boldsymbol{D} \cdot \boldsymbol{V} = \boldsymbol{A} \cdot \boldsymbol{V}$: `>> [V,D]=eig(A);`

Find $\boldsymbol{x}$ which minimizes $f(\boldsymbol{x})$ under constraints $\boldsymbol{A} \cdot \boldsymbol{x} \leq \boldsymbol{b}$: `>> x=fmincon(f,x0,A,b)`

Find the $\boldsymbol{x}$ which minimizes $0.5 \cdot \boldsymbol{x}^T \cdot \boldsymbol{H} \cdot \boldsymbol{x} + \boldsymbol{f}^T \cdot \boldsymbol{x}$: `>> x=quadprog(H,f);`

Matlab is mainly numerical, but also has a symbolic toolbox:

```
>> syms x a; Eq=a/x+x==0; solve(Eq,x)   %symbolically solve equation
ans = +(-a)^(1/2)
      -(-a)^(1/2)
```

Solve the system of equations before simulation!

Not in each time step! (Simulation becomes very slow)