Forces on Ground Vehicles

ES100

March 29, 1999

E. F. Thacher

Topics

- Reducing the airfoil lift data (CLFIT.M)
- Making decisions, part I
- Ground vehicle in steady motion
- Some vehicle geometry
- Lift and drag
- Vehicle wind tunnel testing
- Rolling resistance
- Making decisions, part II
- Coast-down tests; reducing data (COAST.M)

Making Decisions I

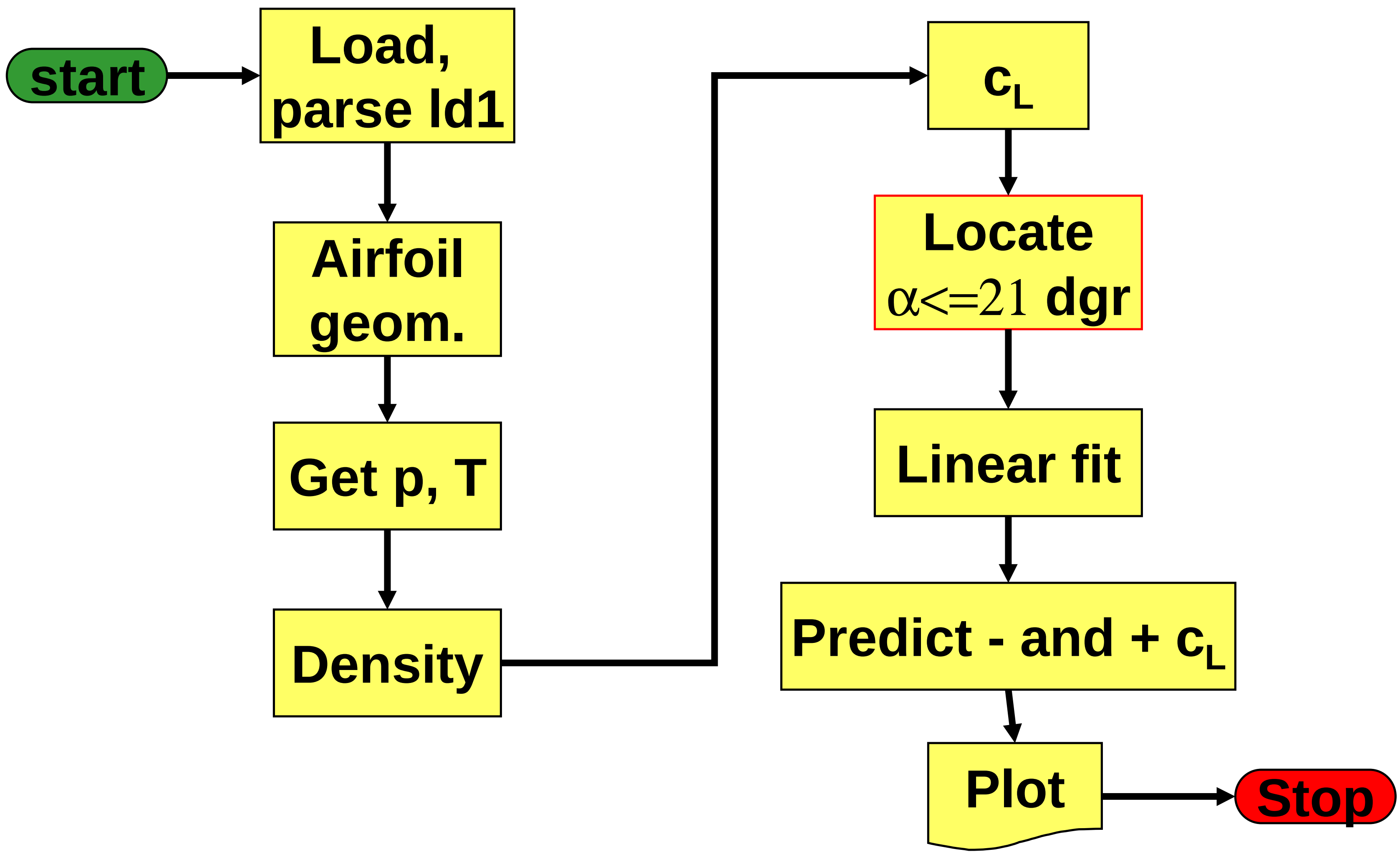
- Relational operators

Find where the angle of attack is not greater than 21 degrees. Then fit a straight line to the data.

z = (angle <= 21)

- Array z has 1's where true, zero's where false

CLFIT.M I


start
Load, parse ld1
Airfoil geom.
Get p, T
Density
c_L
Locate α<=21 dgr
Linear fit
Predict - and + c_L
Plot
Stop

CLFIT.M II

```
% Load and parse the data file
load a:\ld1.txt
V = ld1(:,4);       % vector of air speeds
                    %(m/s)
L = ld1(:,2);       % vector of lifts (N)
D = ld1(:,3);       % vector of drags (N)
angle = ld1(:,1);% vector of angles of
                    % attack (N)
% Airfoil dimensions
c = 0.10152;        % m
b = 0.254;          % m
S = b*c;            % planform area (m^2)
```

CLFIT.M III

```
% Get pressure and temperature
HmmHg = input( 'Enter air pressure
                (mm Hg): ' );
p = mmhg2pa( HmmHg ); % air pressure
                      % (N/m^2)
Tc = input( 'Enter temperature (C): ' );
T = tk( Tc );         % convert to
                      % absolute (K)

% Calculate air density
rho = airden ( p, T );
```

CLFIT.M IV

```
% Calculate the lift coefficients
q = .5*rho*V.^2; % dynamic pressures
                 % (N/m^2)
cL = L./(S*q);

% Locate angles <= 21 degrees

        z = ( angle <= 21 );

fprintf( '\nThe array z:\n %g\n', z)
limit_index = sum(z);   %count the 1's
```

CLFIT.M V

```
% Fit a straight line to the data subset
L = polyfit( angle(1:limit_index),...
     cL(1:limit_index), 1 );
fprintf( '\nThe slope is %g ;
 the intercept is %g \n\n', L(1), L(2) )

% Predict cL at - and + angles
x = [-angle(limit_index)-6:1:...
    angle(limit_index)+6];
y = polyval( L, x );
```

CLFIT.M VI

```
% Plot cL and cL fit vs. angle of attack
plot ( angle, cL, ’square’, x, y, ’-’ );
legend (’lift coefficient’,…
      '10-point linear fit', 0 );
xlabel ('angle of attack (degrees)');
ylabel ('lift coefficient');
title (‘Lift Coefficient and Linear Fit vs.
         Angle of Attack')
grid
```

Demonstrate CLFIT.M

- Note: the prediction

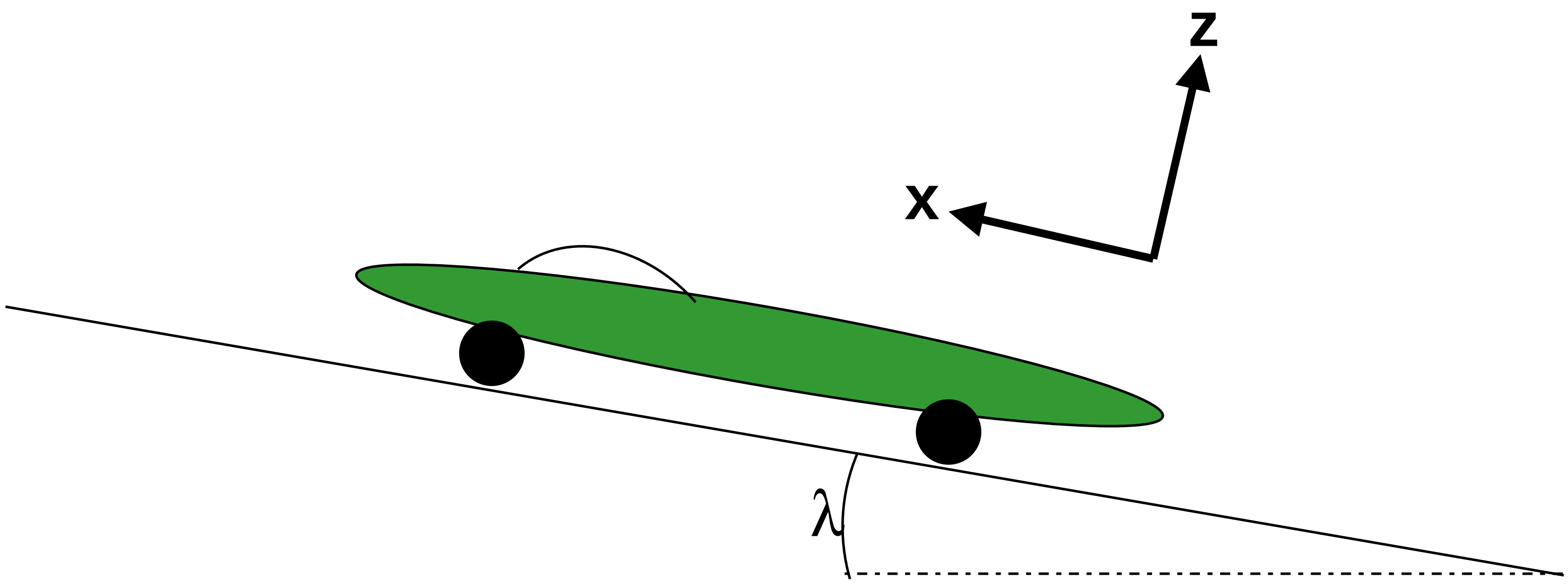
$$c_L(-\alpha) = -c_L(\alpha)$$

is true.

Ground Vehicle in Steady Motion

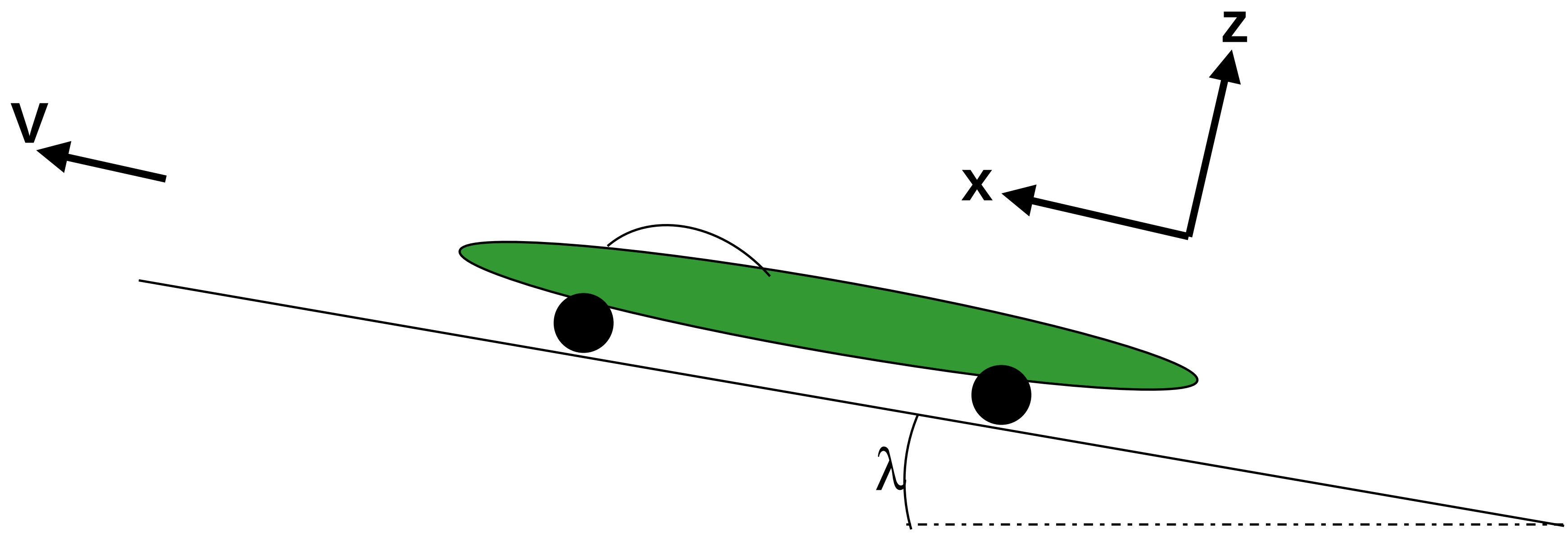
Ground Vehicle in Steady Motion

- Neglect transverse forces and moments



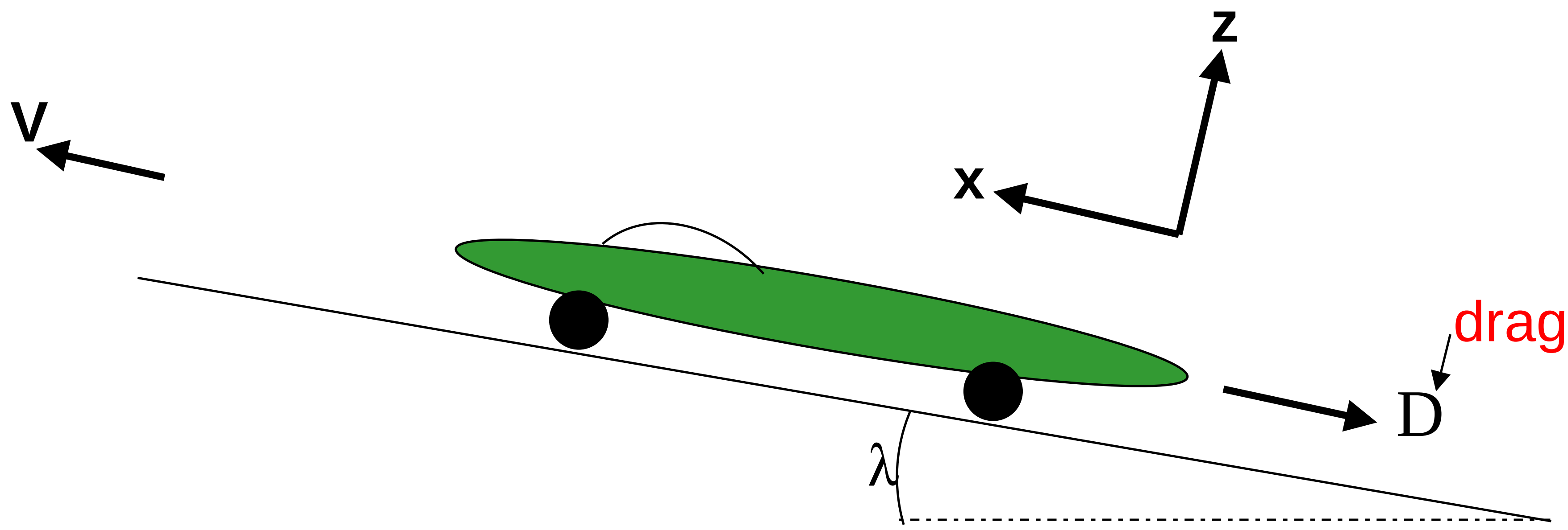
Ground Vehicle in Steady Motion

- Neglect transverse forces and moments



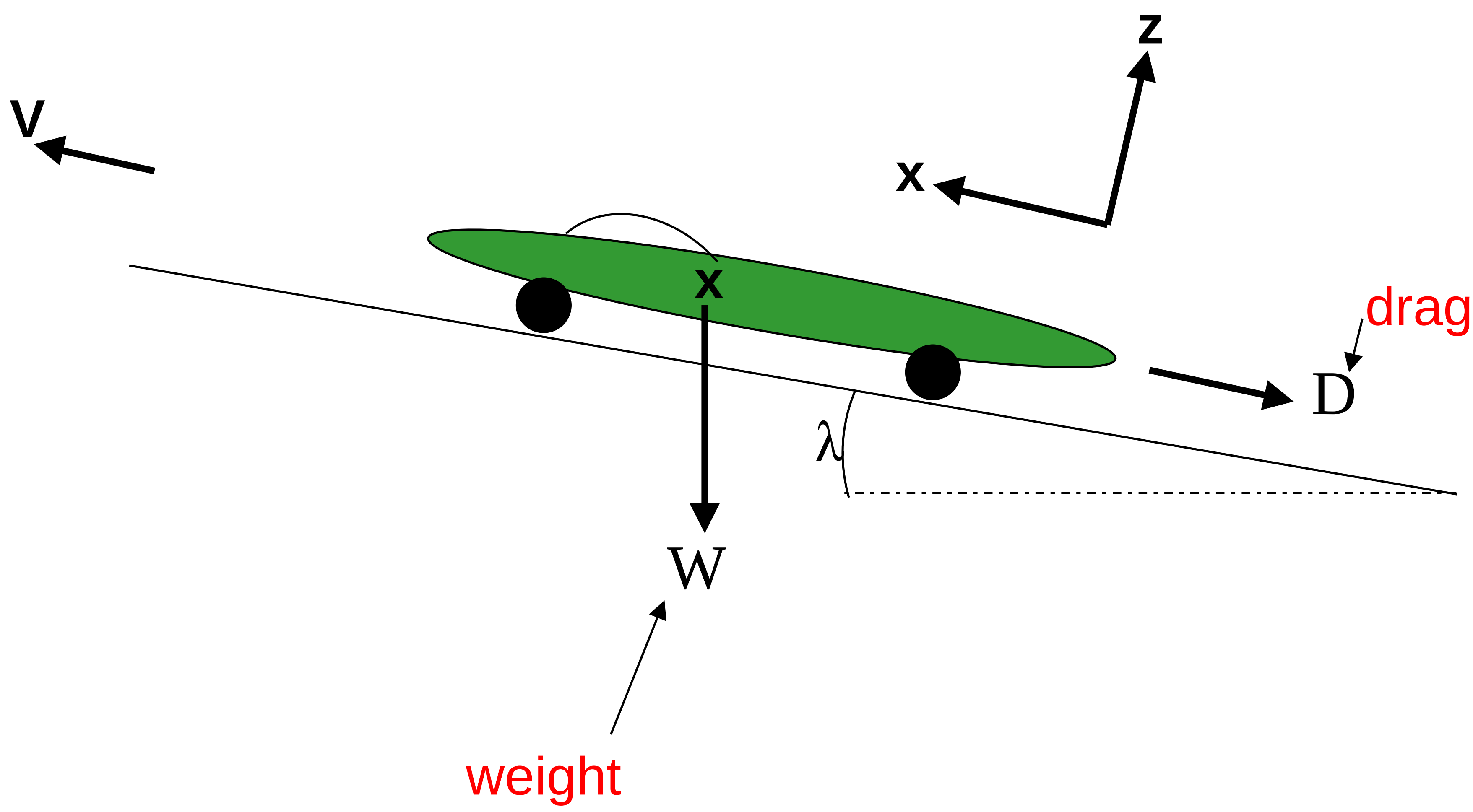
Ground Vehicle in Steady Motion

- Neglect transverse forces and moments



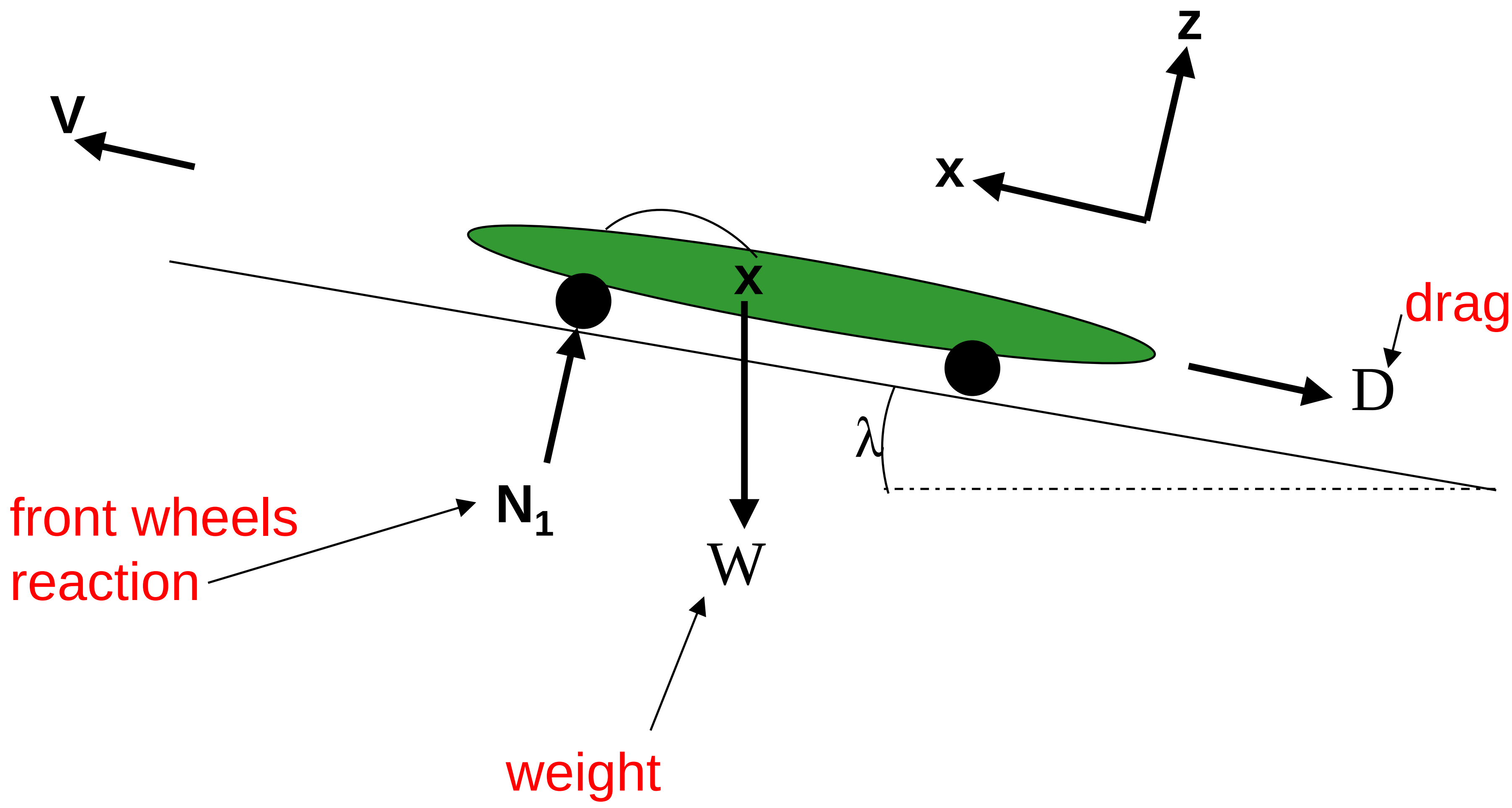
Ground Vehicle in Steady Motion

- Neglect transverse forces and moments



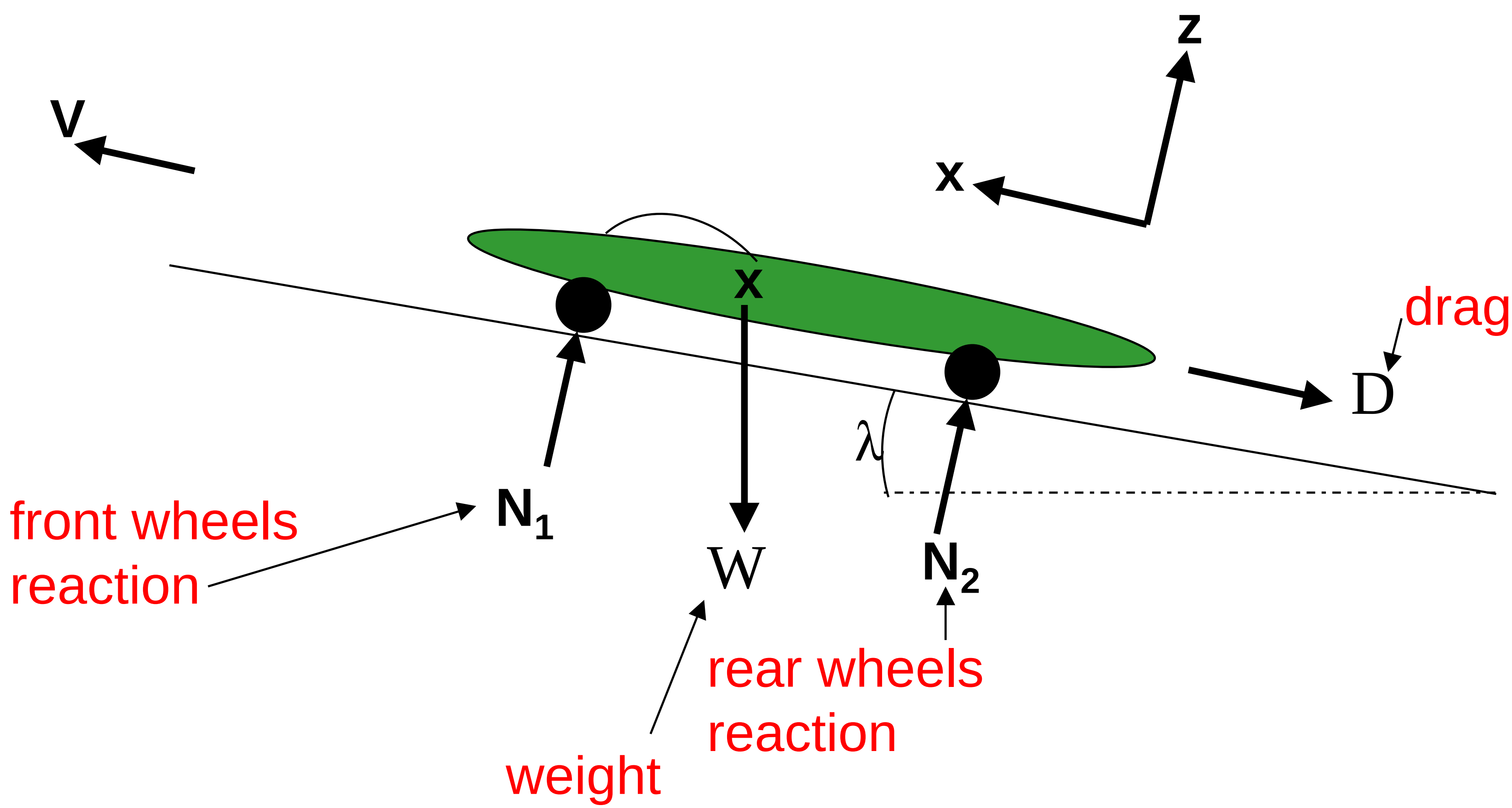
Ground Vehicle in Steady Motion

- Neglect transverse forces and moments



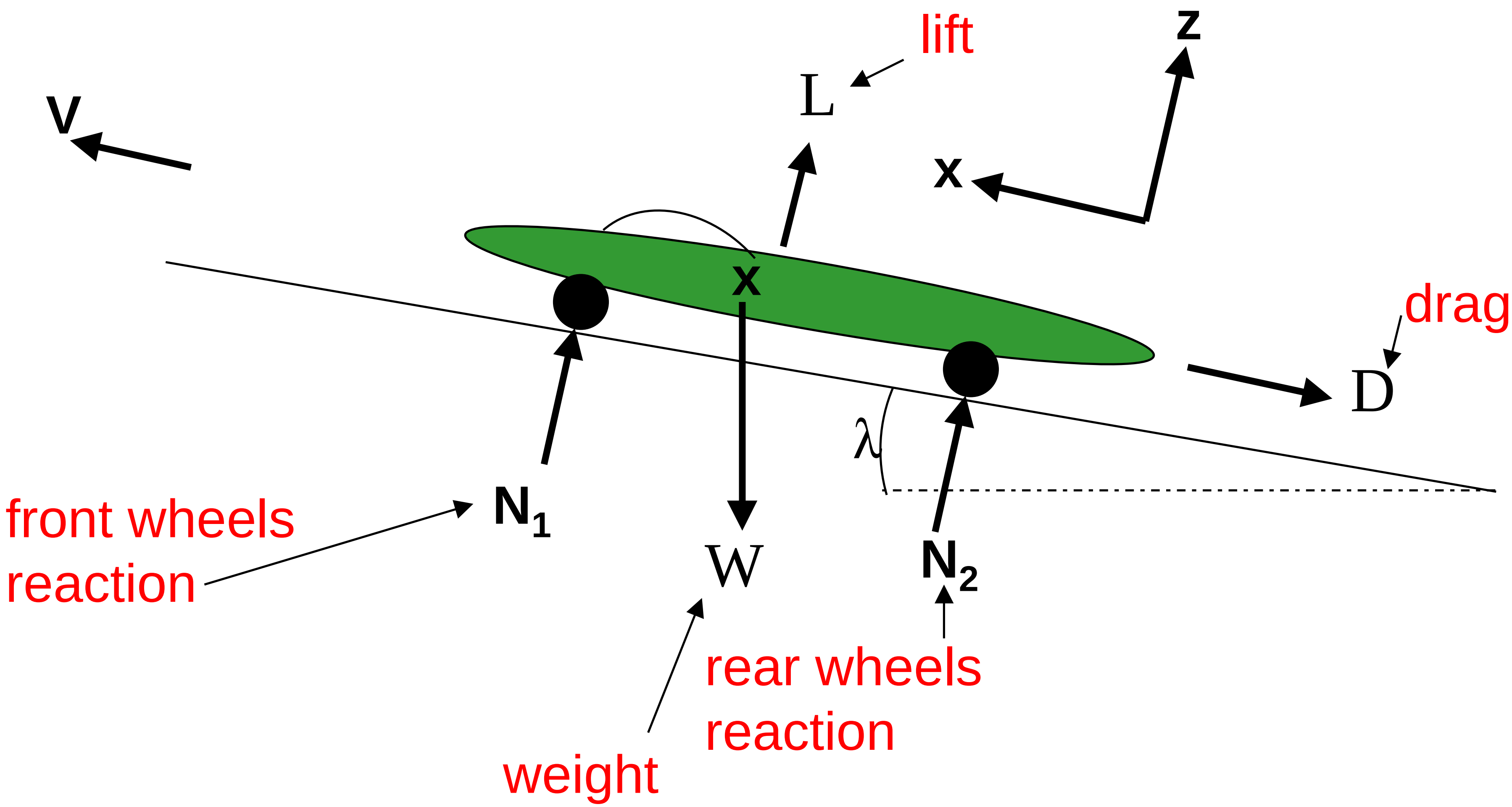
Ground Vehicle in Steady Motion

- Neglect transverse forces and moments



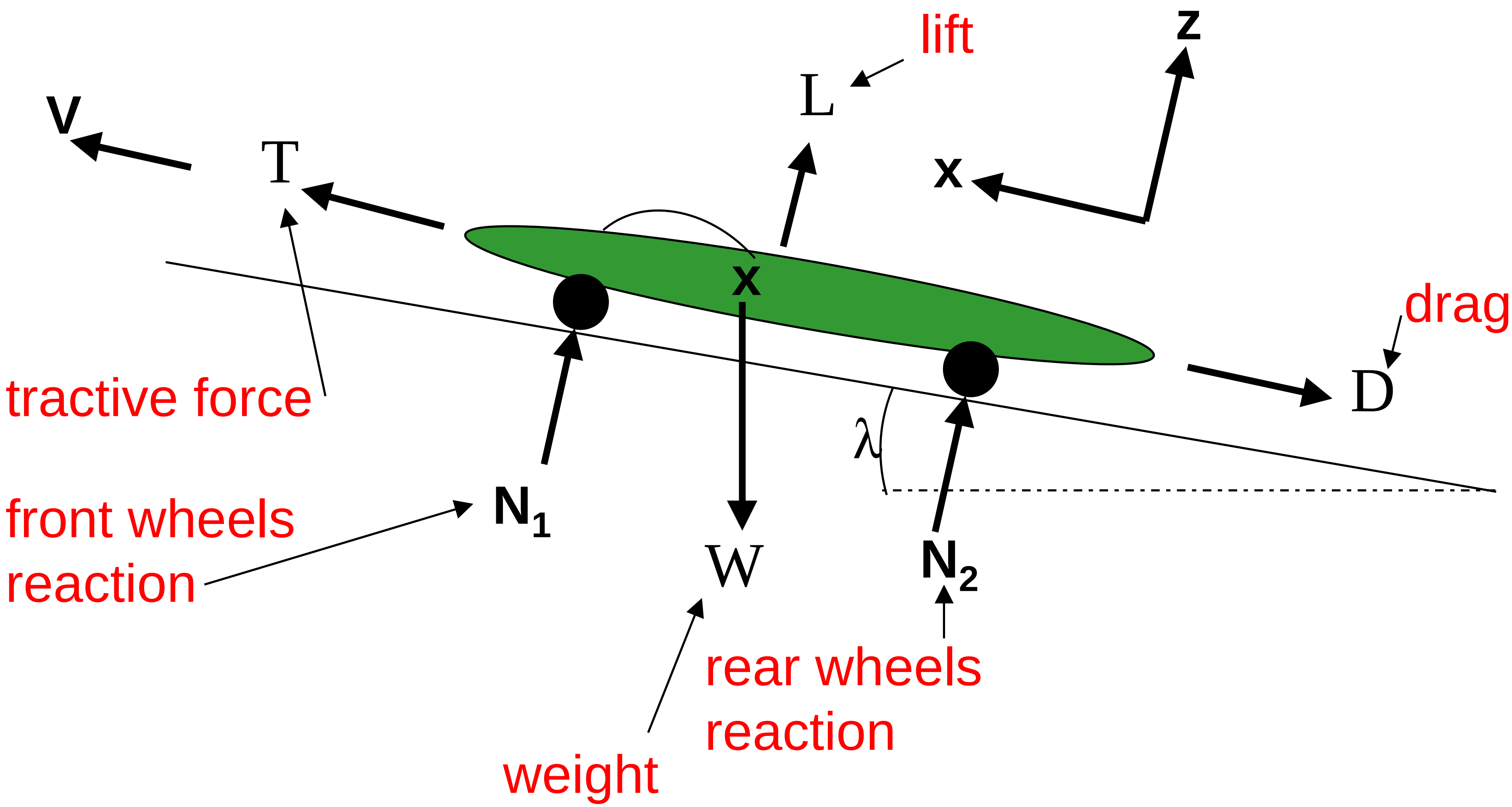
Ground Vehicle in Steady Motion

- Neglect transverse forces and moments



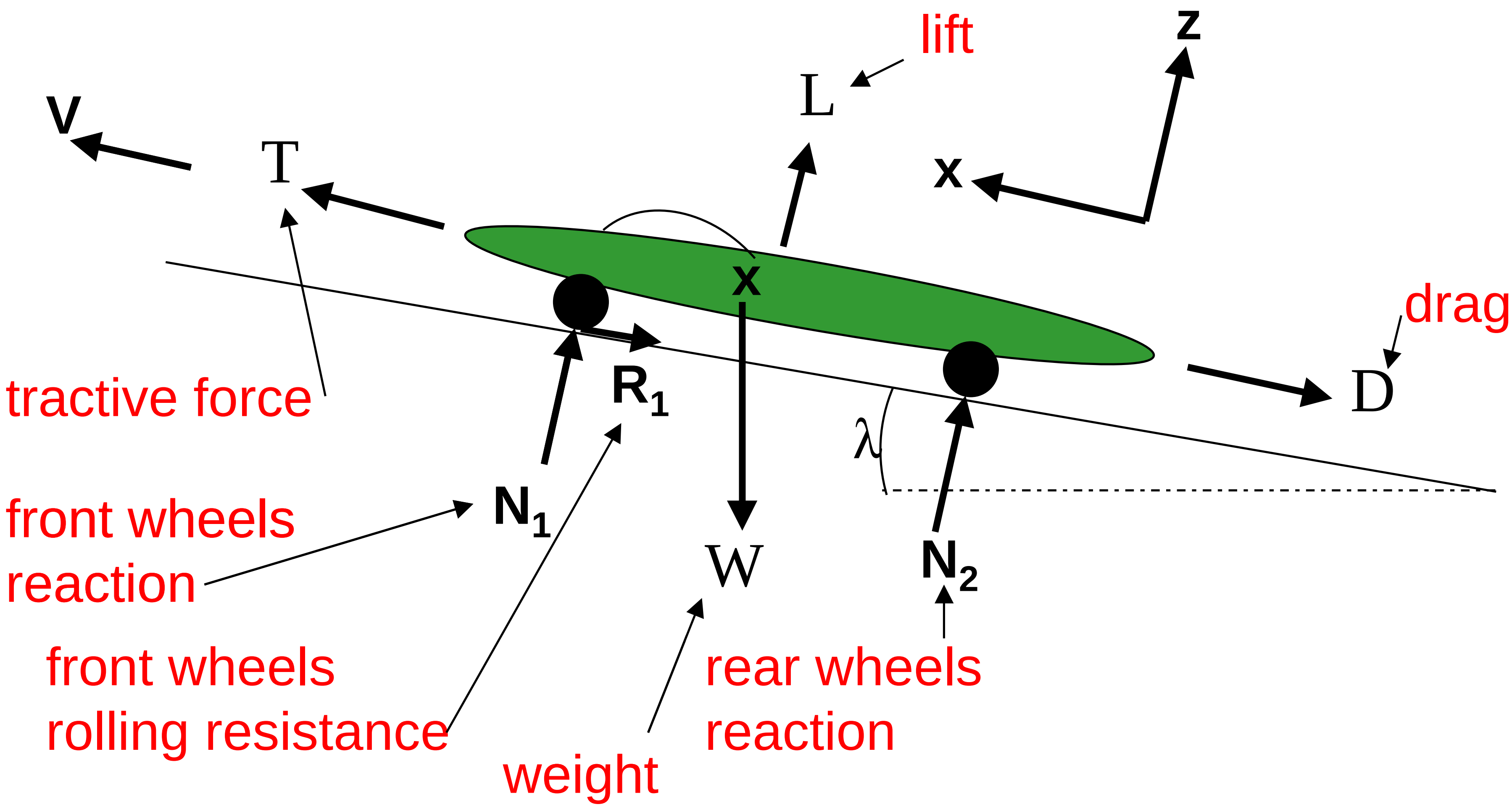
Ground Vehicle in Steady Motion

- Neglect transverse forces and moments



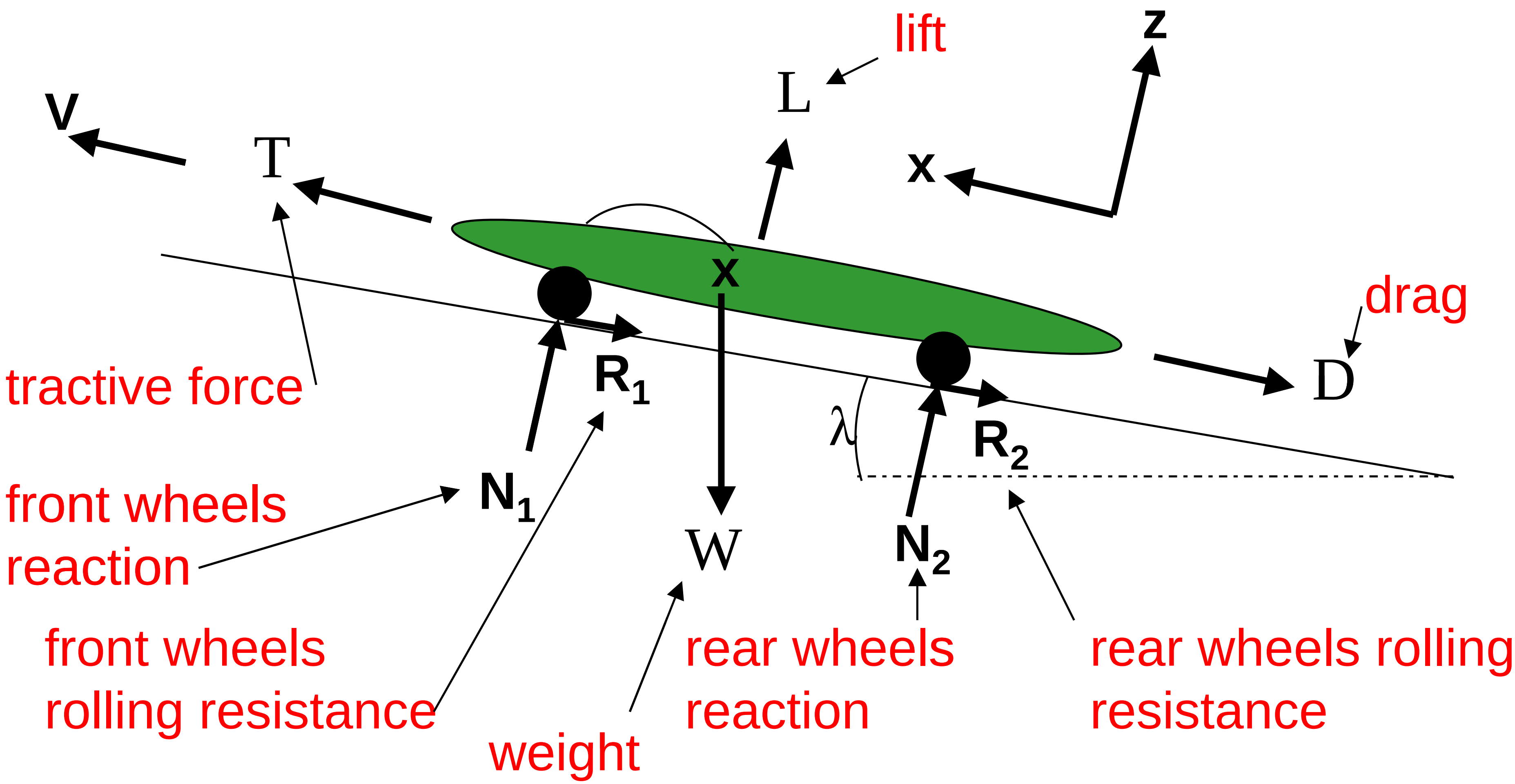
Ground Vehicle in Steady Motion

- Neglect transverse forces and moments



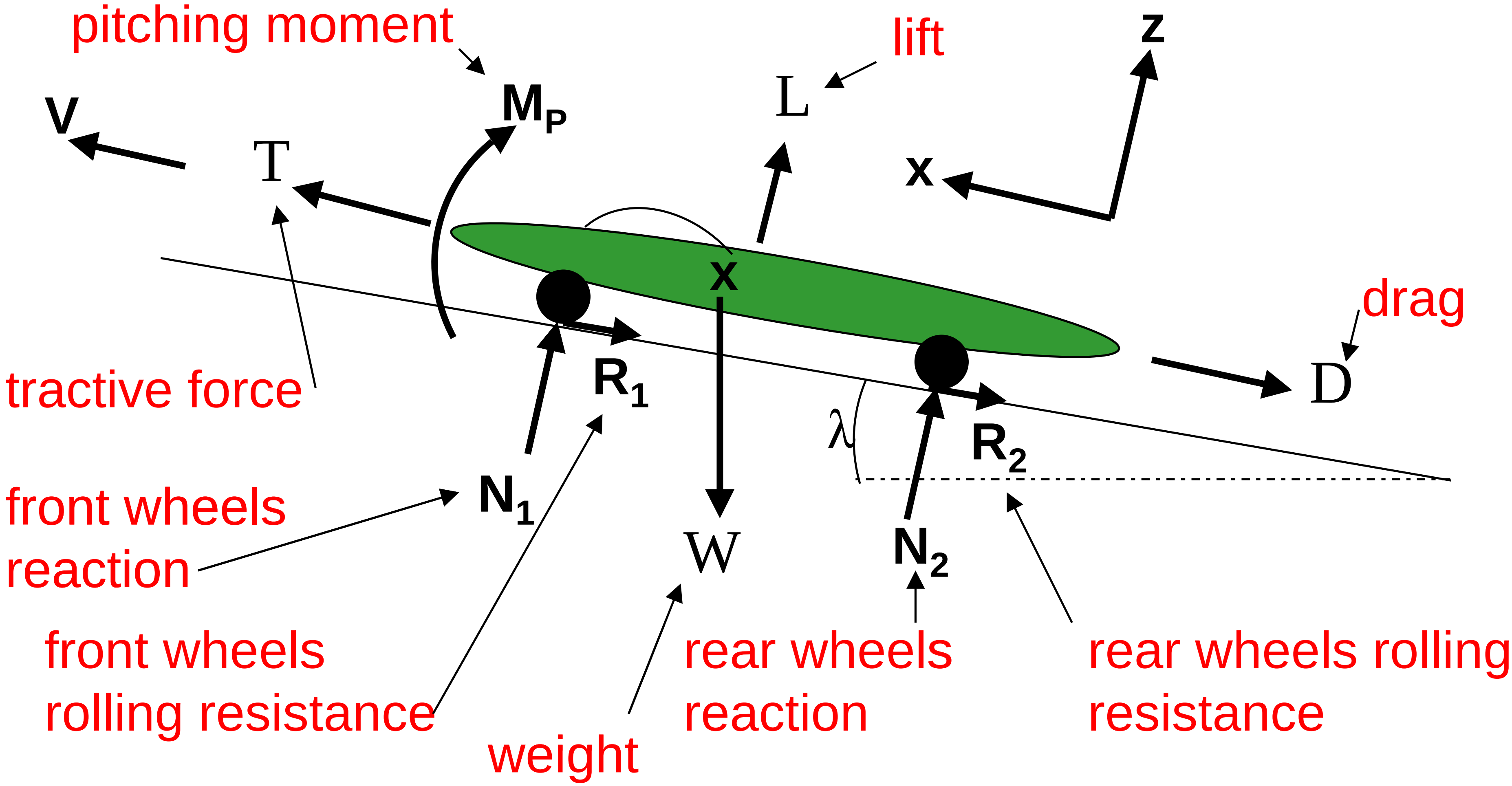
Ground Vehicle in Steady Motion

- Neglect transverse forces and moments



Ground Vehicle in Steady Motion

- Neglect transverse forces and moments



Ground Vehicle in Steady Motion

Summing Forces in x- and z-directions:

$$\sum F_x = T - R - D - W\sin\lambda = 0$$

$$\sum F_z = L + N - W\cos\lambda = 0$$

$R = R_1 + R_2$
$N = N_1 + N_2$

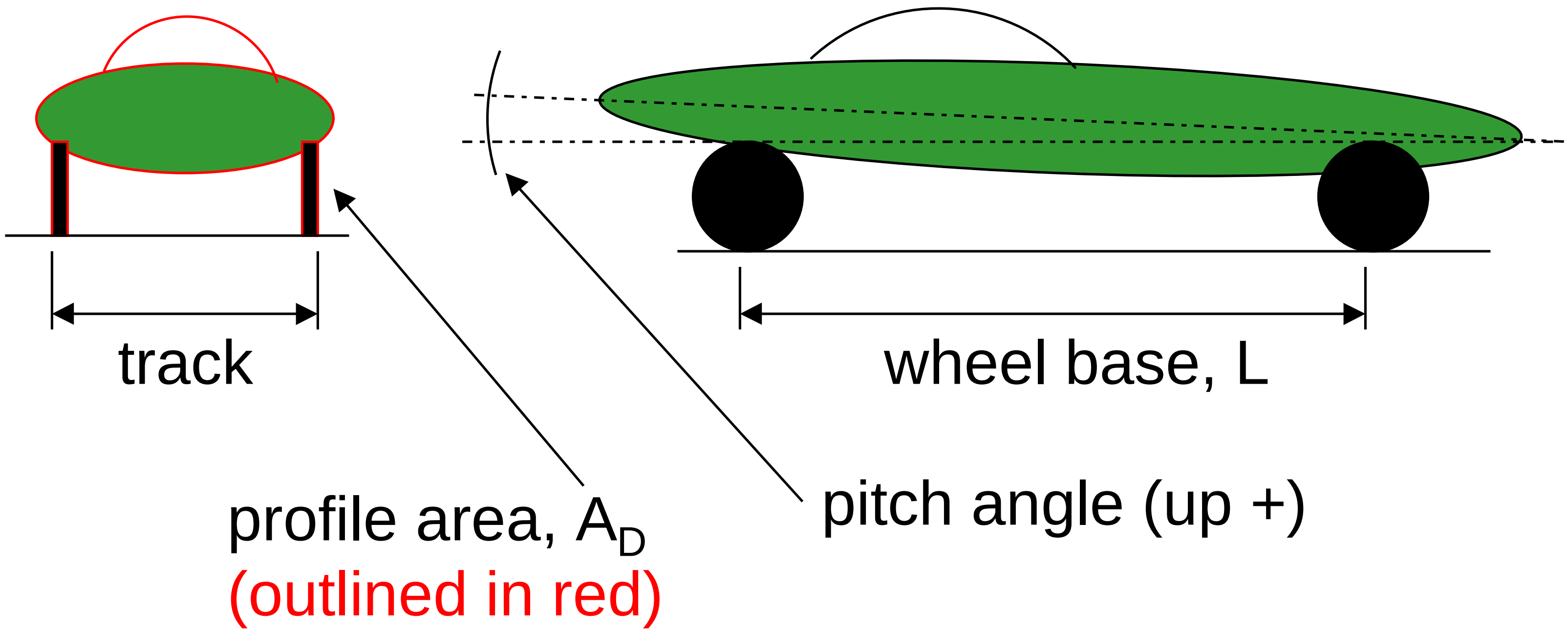
Accelerating in X-Direction

$$T - R - D - W\sin\lambda = M_e \frac{dV}{dt}$$

$$L + N - W\cos\lambda = 0$$

The effective mass, M_e, accounts for the rotational inertia of the wheels. It is about 1.03 M, typically.

Some Vehicle Geometry



“drag area” is $c_D A_D$, [m^2]

Lift Coefficient of Car

$$c_L = \frac{L}{A_D \frac{1}{2} \rho V^2}$$

$$\frac{N}{(m^2)\left(\frac{N}{m^2}\right)} \Rightarrow$$ c_L is dimensionless

Lift Coefficient of Car

lift coefficient

$$c_L = \frac{L}{A_D \frac{1}{2} \rho V^2}$$

$$\frac{N}{(m^2)\left(\frac{N}{m^2}\right)} \Rightarrow$$ c_L is dimensionless

Lift Coefficient of Car

lift coefficient

$$c_L = \frac{L}{A_D \frac{1}{2} \rho V^2}$$

lift force

$$\frac{N}{(m^2)\left(\frac{N}{m^2}\right)} \Rightarrow$$ c_L is dimensionless

Lift Coefficient of Car

lift coefficient

$$c_L = \frac{L}{A_D \frac{1}{2} \rho V^2}$$

lift force

profile area

$$\frac{N}{(m^2)\left(\frac{N}{m^2}\right)} \Rightarrow$$ c_L is dimensionless

Lift Coefficient of Car

lift coefficient

$$c_L = \frac{L}{A_D \frac{1}{2} \rho V^2}$$

lift force

profile area

air density

$$\frac{N}{(m^2)\left(\frac{N}{m^2}\right)} \Rightarrow$$ c_L is dimensionless

Lift Coefficient of Car

lift coefficient

$$c_L = \frac{L}{A_D \frac{1}{2} \rho V^2}$$

lift force

profile area

air density

relative air speed

$$\frac{N}{(m^2)\left(\frac{N}{m^2}\right)} \Rightarrow$$ c_L is dimensionless

Drag Coefficient of Car

drag coefficient

$$c_D = \frac{D}{A_D \frac{1}{2} \rho V^2}$$

drag force

profile area

air density

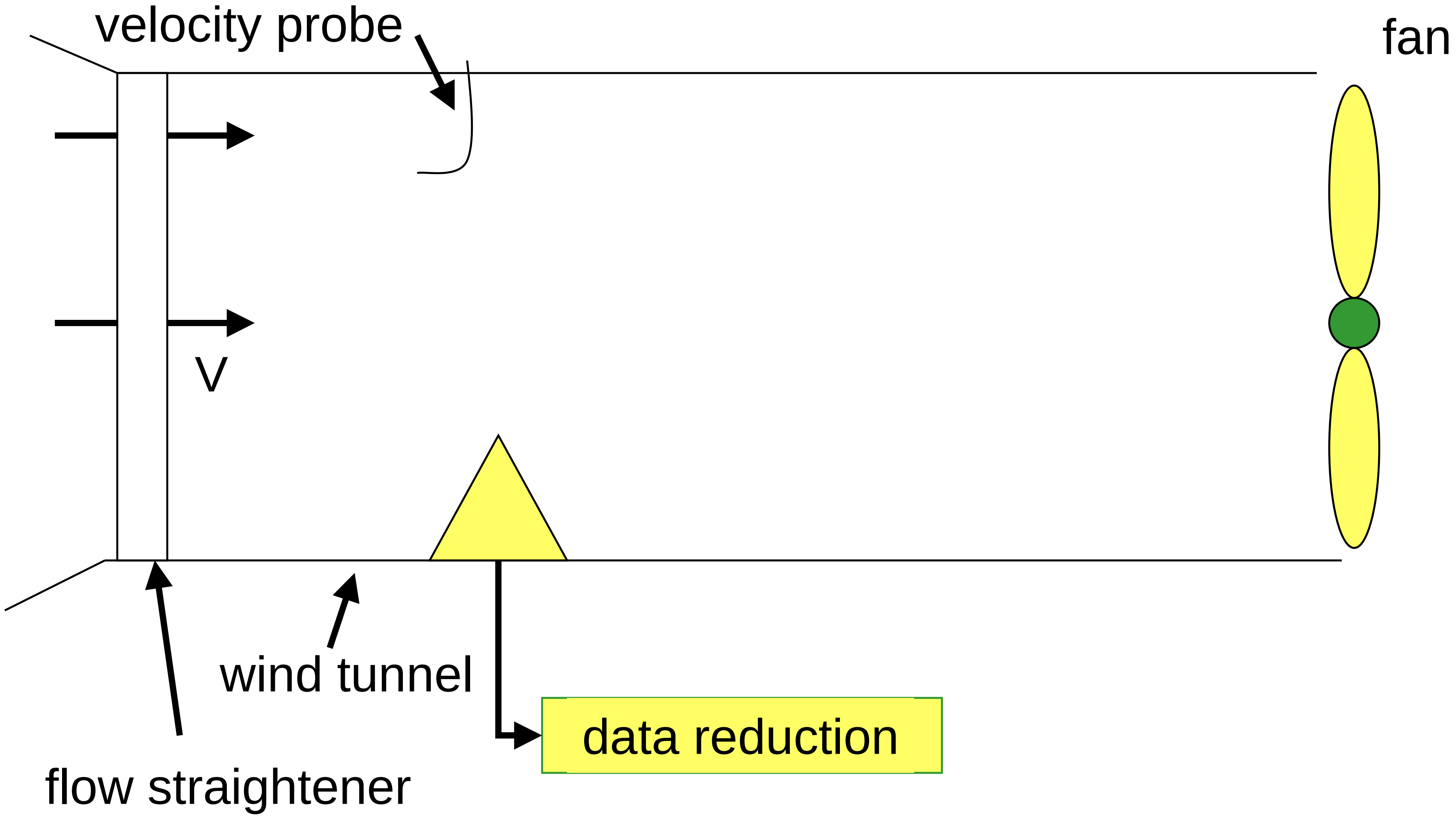
relative air speed

$$\frac{N}{(m^2)\left(\frac{N}{m^2}\right)} \Rightarrow$$ c_D is dimensionless

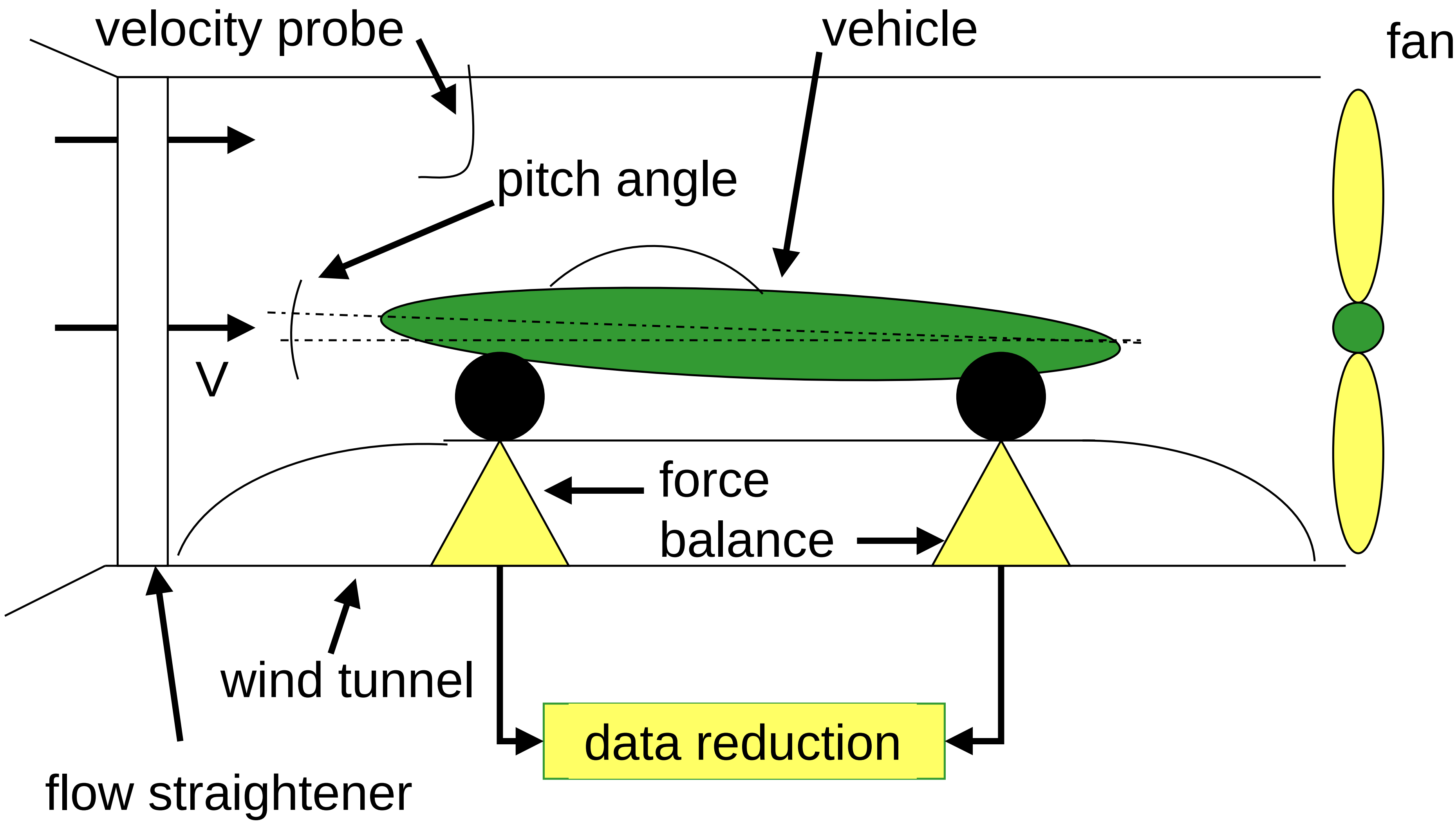
Lift, Drag in Straight Ahead Flow

- Lift and drag depend on
  - The relative airspeed
  - The viscosity and density of the air
  - The shape and smoothness of the vehicle
    - Including wheels and wheel wells, and the
    - Ventilation system
  - The pitch angle of the vehicle
  - The proximity of the ground

Measuring c_L and c_D in Tunnel



Measuring c_L and c_D in Tunnel



Utility of Model Tests

- Use c_L and c_D *for full scale car* **if**
  - Geometrically similar car
  - Same Reynolds number

- However, drag of actual car usually greater than model
  - Model usually simplified
  - Some full-scale test facilities exist

Reynolds Number

Proportional to the

ratio of dynamic to viscous forces

acting on a fluid element

$$\boxed{Re = \rho VL/\mu}$$

L is wheel base

$$\frac{\left(\frac{kg}{m^3}\right)\left(\frac{m}{\sec}\right)(m)}{\frac{N\sec}{m^2}} = \frac{\frac{N}{m^2}}{\frac{N}{m^2}} \Rightarrow n.d.$$

Rolling Resistance

- Contact patch force
  - Proportional to N
- Bearing friction (small)
  - Proportional to V and W
- Wheel rotational drag (smallest)
  - Proportional to V

$$\boxed{R = \mu W = (\mu_1 \cos\lambda + \mu_2 V)W}$$

Coast-Down Test to Measure $c_D A_D$, μ_1, μ_2

- Measure W, p (pressure), and T (temp. Celcius)
- No wind, horizontal road (ideally)
- Initial speed greater than about 40 mph
- Coast down in straight line and record V at intervals
- [Note: μ means “rolling resistance coefficient,” in this case -- not viscosity]

Reducing Coast Down Data

- Fit V(t), find $\dot{V}(t)$, fit a quadratic in V to it:

$$\boxed{\dot{V}(t) = a_2 V^2 + a_1 V + a_0}$$

- Where

$$\boxed{a_2 = \frac{c_D A_D \rho}{2M_e}, \quad a_1 = \frac{\mu_2 W}{M_e}, \quad a_0 = \frac{\mu_1 W}{M_e}}$$

Making Decisions II

- If the user selects numeric derivative
  - …then do that method and go on
- Else she wants to smooth the data first
  - …then do that and go on
- End of decision

First Derivative Approximations

Forward difference

$$\frac{dV}{dt} \approx \frac{V_{t+\Delta t} - V_t}{\Delta t} = \frac{V_{i+1} - V_i}{\Delta t}$$

"i" ==> t

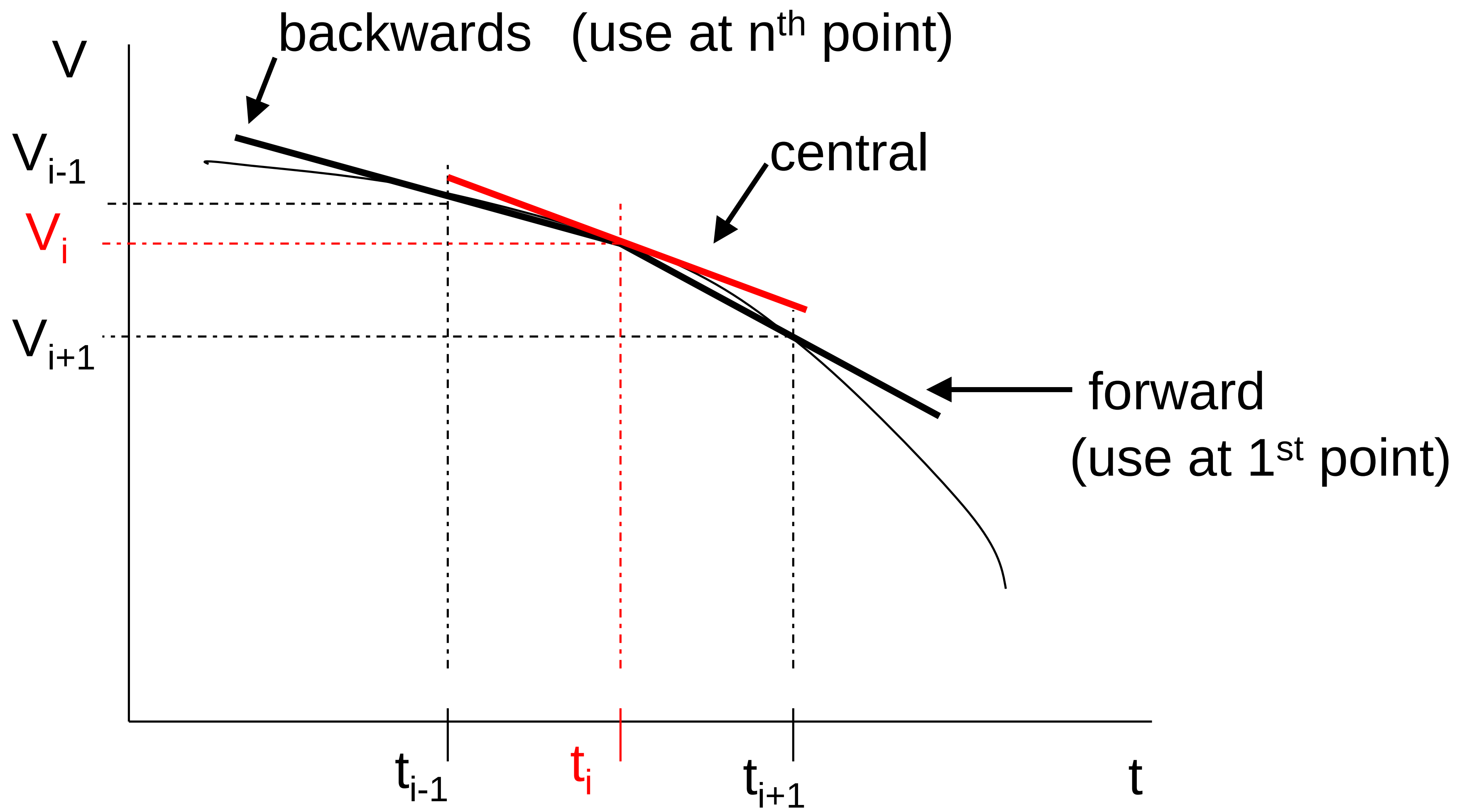
Central difference

$$\frac{dV}{dt} \approx \frac{V_{i+1} - V_{i-1}}{2\Delta t}$$

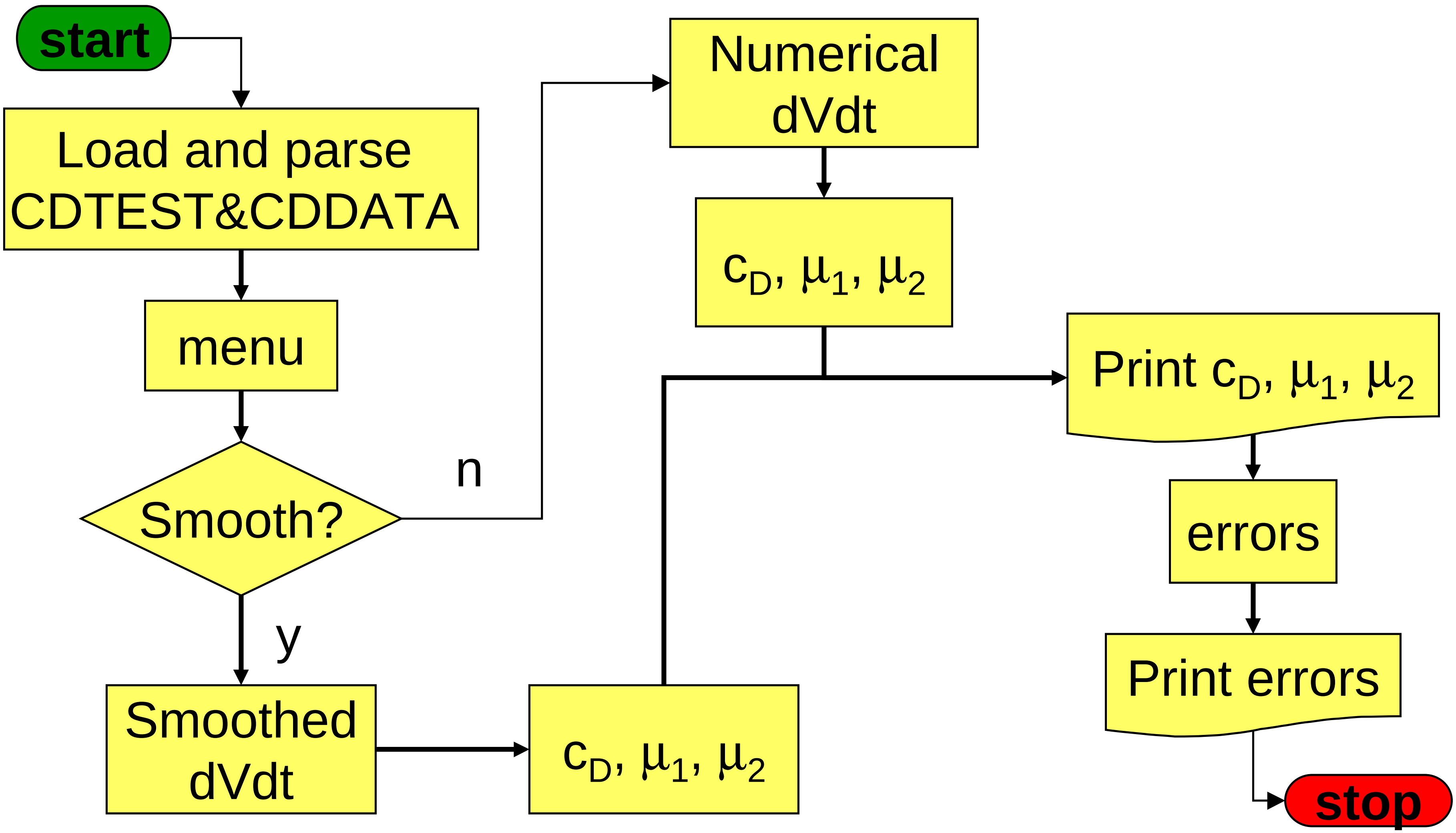
Backward difference

$$\frac{dV}{dt} \approx \frac{V_i - V_{i-1}}{\Delta t}$$

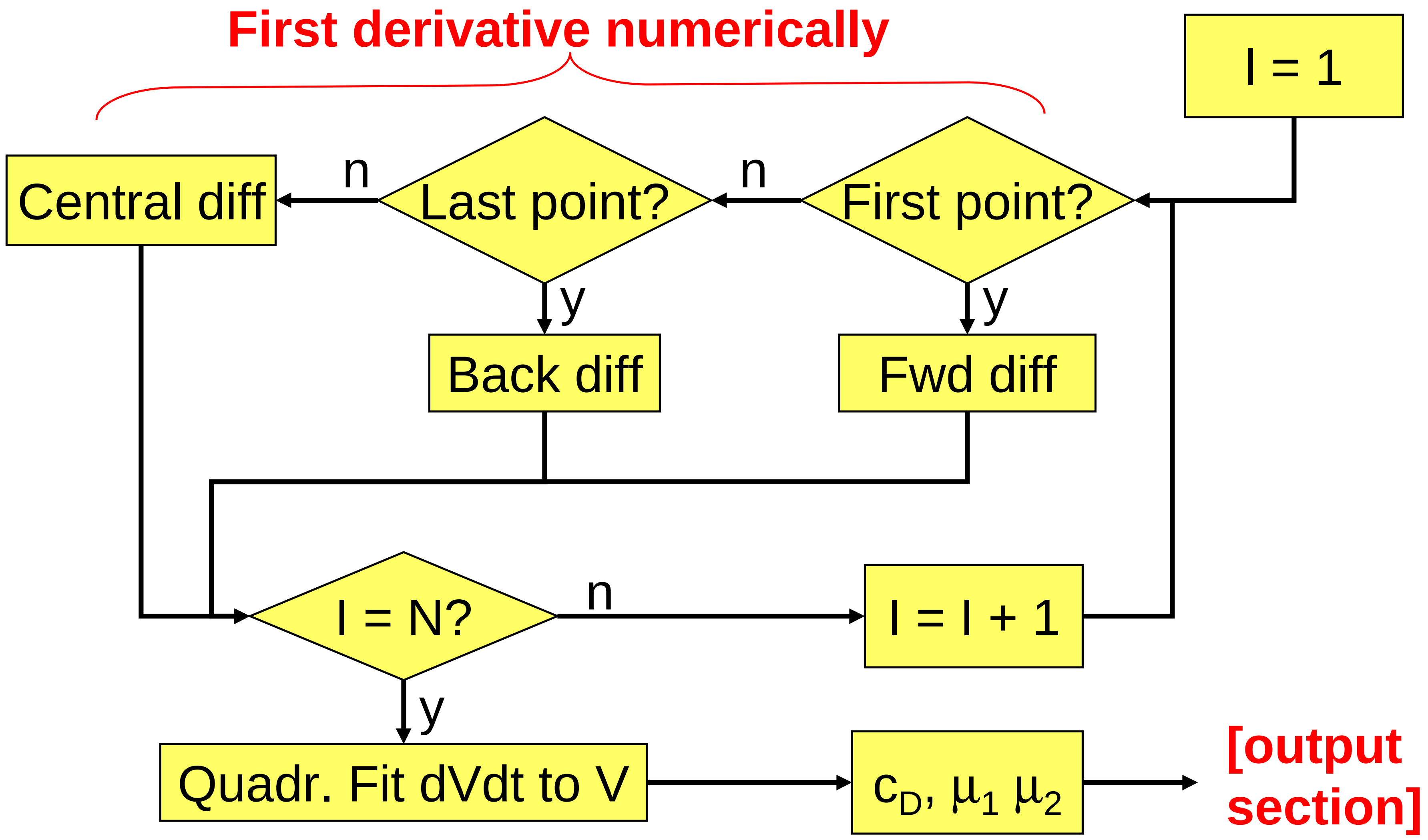
Numerical Derivatives II



COASTD.M Flow Chart I



Numerical dV/dt



COASTD.M I

```
% Load and parse the data file
load a:\cddata -ascii;
t = ld1(:,1);           % vector of times (sec)
V = ld1(:,2);           % vector of car speeds at
                        % times (m/s)

% Car mass and profile area
load a:\cdtest -ascii;
Ap = cdtest(2);              % m^2
M = cdtest(5);               % kg
p = cdtest(6);
Tc = cdtest(7);
T = tk (Tc);
rho = airden (p*1000, T);
```

COASTD.M II

```
% Menu
fprintf ('\nMENU:\n\t1. Unsmoothed data,
             numeric derivative.\n\t2.
             Cubic smoothing.\n\n\t');

choice = input ('Enter number of
                 choice: ');
```

The “choice” if-block

If choice == 1

[un-smoothed data, numeric derivative statements]

else

[cubic-polynomial fitted to data, derivative is derivative of polynomial]

end

COASTD.M III

```
% Start of “choice” if-block
if choice == 1

   % Un-smoothed data, numeric derivative
   n = length(V);
   clear dVdt
   dt = t(2)-t(1);

   % Loop through all data points

     [next slide]
```

[if choice == 1, continued]

```
for i = 1:n
    if i == 1
        dVdt(i) = ( V(i+1)-V(i) )/dt;
    elseif i == n
        dVdt(i) = ( V(i)-V(i-1) )/dt;
    else
        dVdt(i) = ( V(i+1)-V(i-1) )/( 2*dt);
    end
end        % End of loop through data
% Reduce the results
   a = polyfit (V, dVdt', 2);
   [cDm mu1m mu2m] = reduce (a, cdtest);
```

COASTD.M IV

```
else
% Try cubic smoothing
    b3 = polyfit (t, V, 3);
    Vf3 = b3(1)*t.^3 + b3(2)*t.^2 + …
          b3(3)*t + b3(4);
    dVdt3 = 3*b3(1)*t.^2 + 2*b3(2)*t +…
                 b3(3);
% Reduce the results
    a3 = polyfit (Vf3, dVdt3, 2);
    [cDm mu1m mu2m] = reduce (a3, cdtest);
end      % End of “choice” if-block
```

COASTD.M V

```
% Print results
fprintf ('\nResults:\n\tcD = %g\n
   \tmu1 = %g\n\tmu2 = %g (s/m)\n\n',...
    cDm, mu1m, mu2m);

% Calculate errors
errors (cDm, mu1m, mu2m, cdtest);
```

New Functions in COASTD.M I

```
function errors ( cD, mu1, mu2, cdtest )

ecD = abs( (cD/cdtest(1)) - 1 )*100;
emu1 = abs( (mu1/cdtest(3)) - 1 )*100;
emu2 = abs( (mu2/cdtest(4)) - 1 )*100;
fprintf ('\n\tError in cD = %g %%\n
    \tError in mu1 = %g %%\n
    \tError in mu2 = %g %%\n\n',...
    ecD, emu1, emu2)
```

New Functions in COASTD.M II

```
function [cD, mu1, mu2] =
                        reduce(a, cdtest);
% Assign convenient names
AD = cdtest(2);
M = cdtest(5);
p = cdtest(6);
Tc = cdtest(7);
% Calculate T, rho, Me, and W
T = tk (Tc);
rho = airden (p*1000, T);
Me = 1.03*M;
W = M*9.807;
```

Function REDUCE.M, cont.

```
% Find cD, mu1, mu2.  Recall that:
%     a(1) = -.5*cD*AD*rho/Me;
%     a(2) = -mu2*W/Me;
%     a(3) = -mu1*W/Me;
cD = -a(1)*2*Me/(AD*rho);
mu1 = -a(3)*Me/W;
mu2 = -a(2)*Me/W;
```

Results from COASTD.M

- Effect of number of measurements
- Effect of standard deviation