Feeding Your Robot

Careful Feeding Improves Control
or
What *are* those numbers?

UK Micromouse and Robotics Society

Peter Harrison
UKMARS
2022

Objectives

- Describe UKMARSBOT Control Scheme
- Examine Controller Responses
- Desirability of Feed Forward Component
- Characterise Robot Response
- Derive Robot Feed Forward Constants
- Apply Feed Forward to Robot Controller

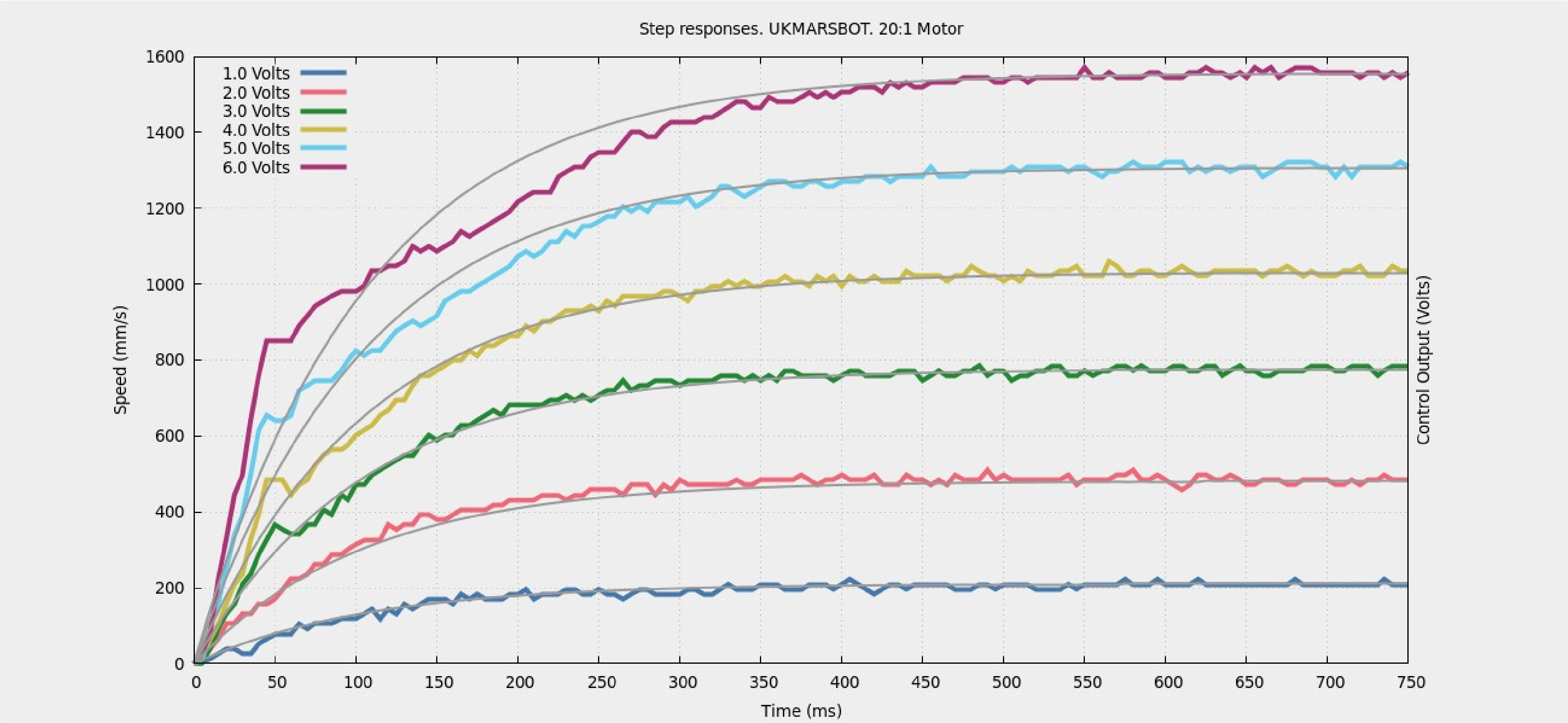
Control Objectives

- **Accurate**
  - steady state errors
- **Disturbance Rejection**
  - battery, steps and slopes
- **Responsive**
  - fast with minimal overshoot
- **Stable**
  - always reaches steady state

Why Use A ‘Controller’?

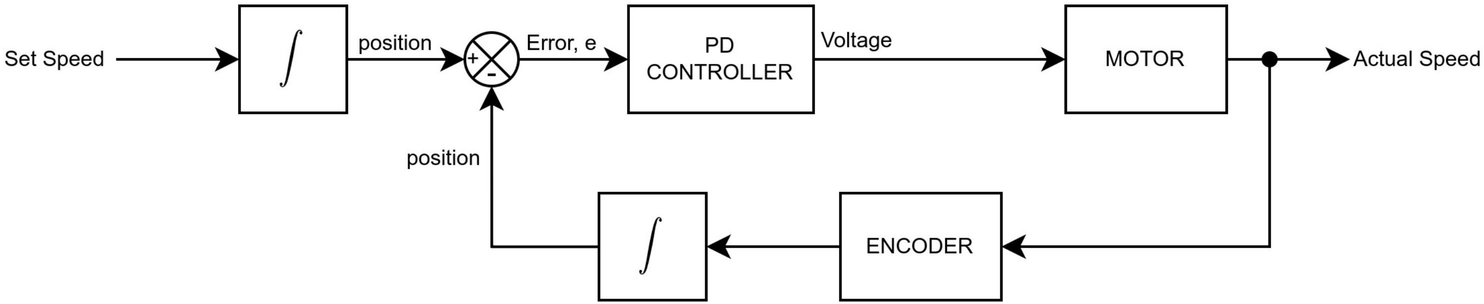
- Set a PWM duty.
  - Speed changes with battery voltage
- Set a Voltage
  - Speed changes with load
- Bang-Bang
  - Turn it on and turn it off to get an average speed
  - Hard on the bits!

Step Response

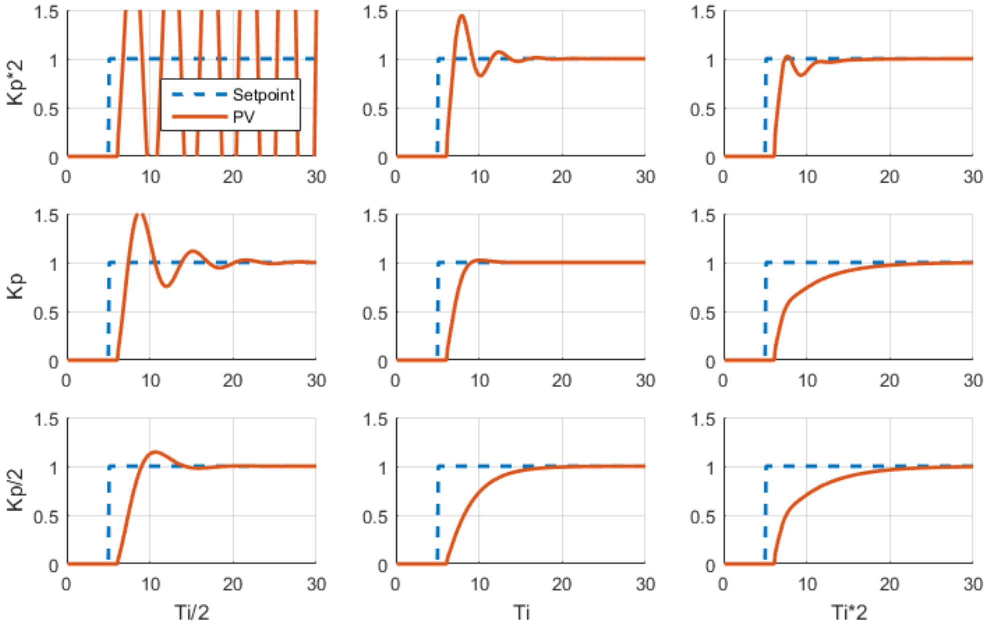


Controller Types

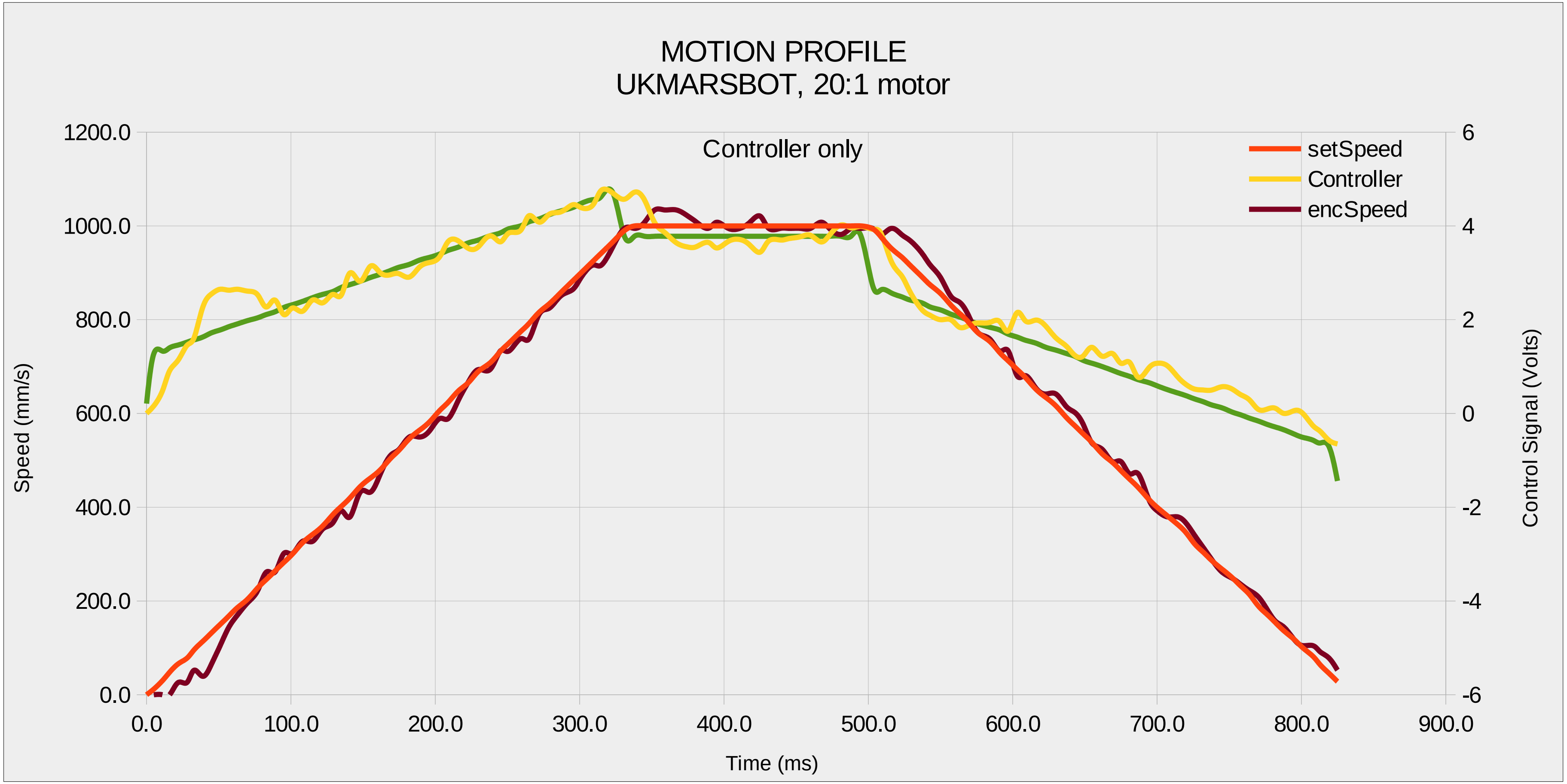
- PID - Proportional, Integral, Derivative
- Commonly Used, Often Misunderstood



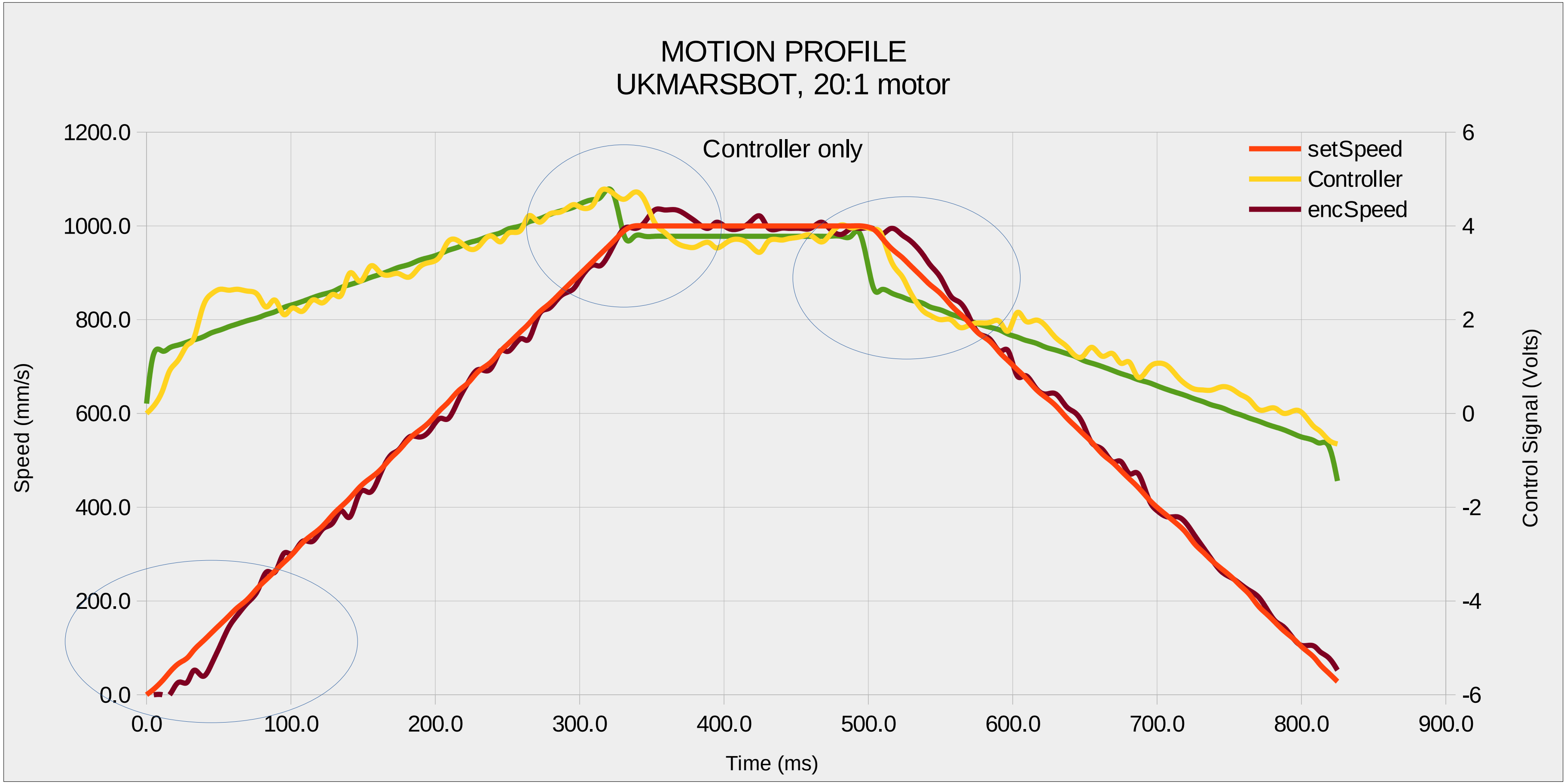
Feedback Control Tuning

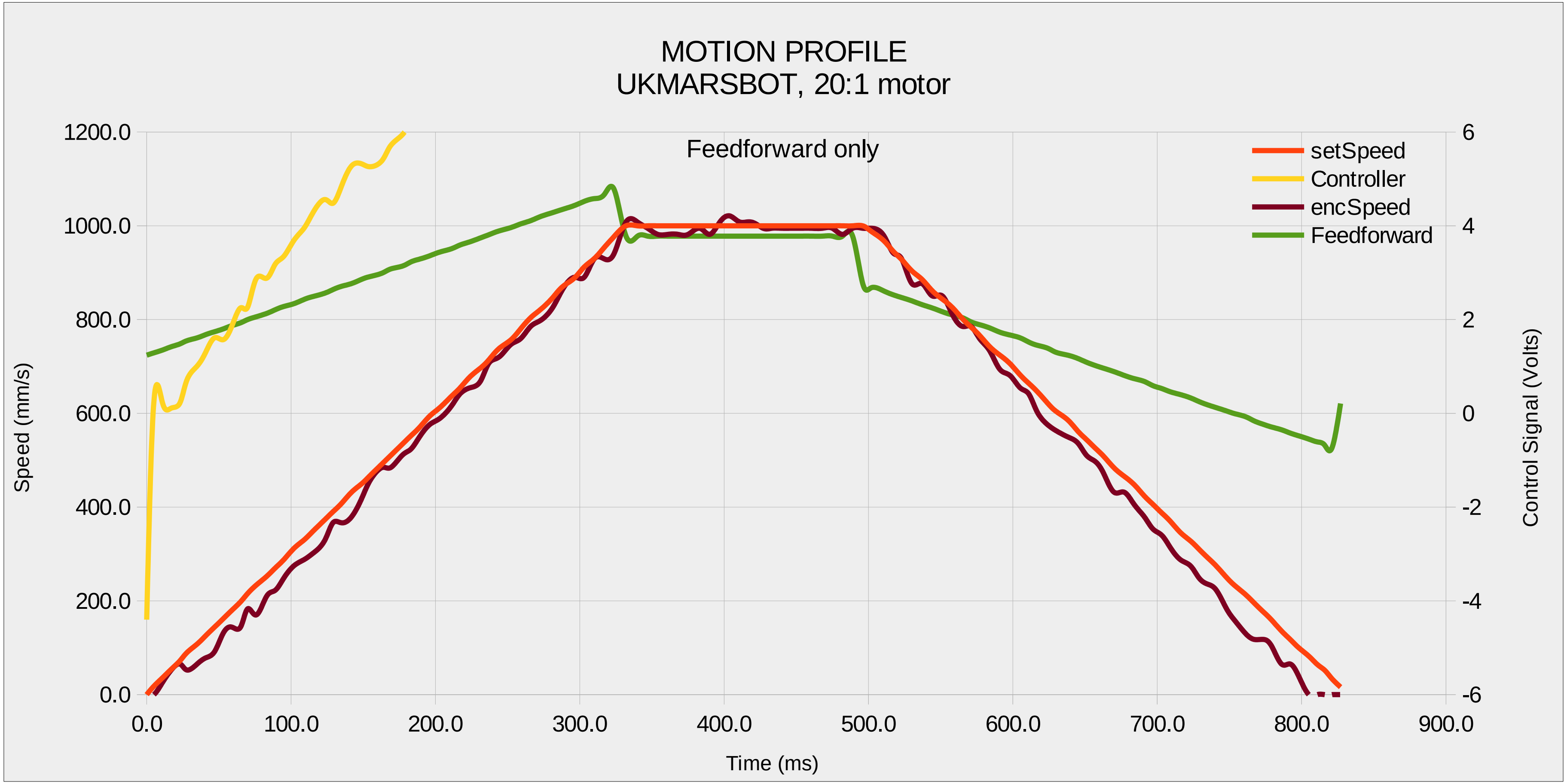


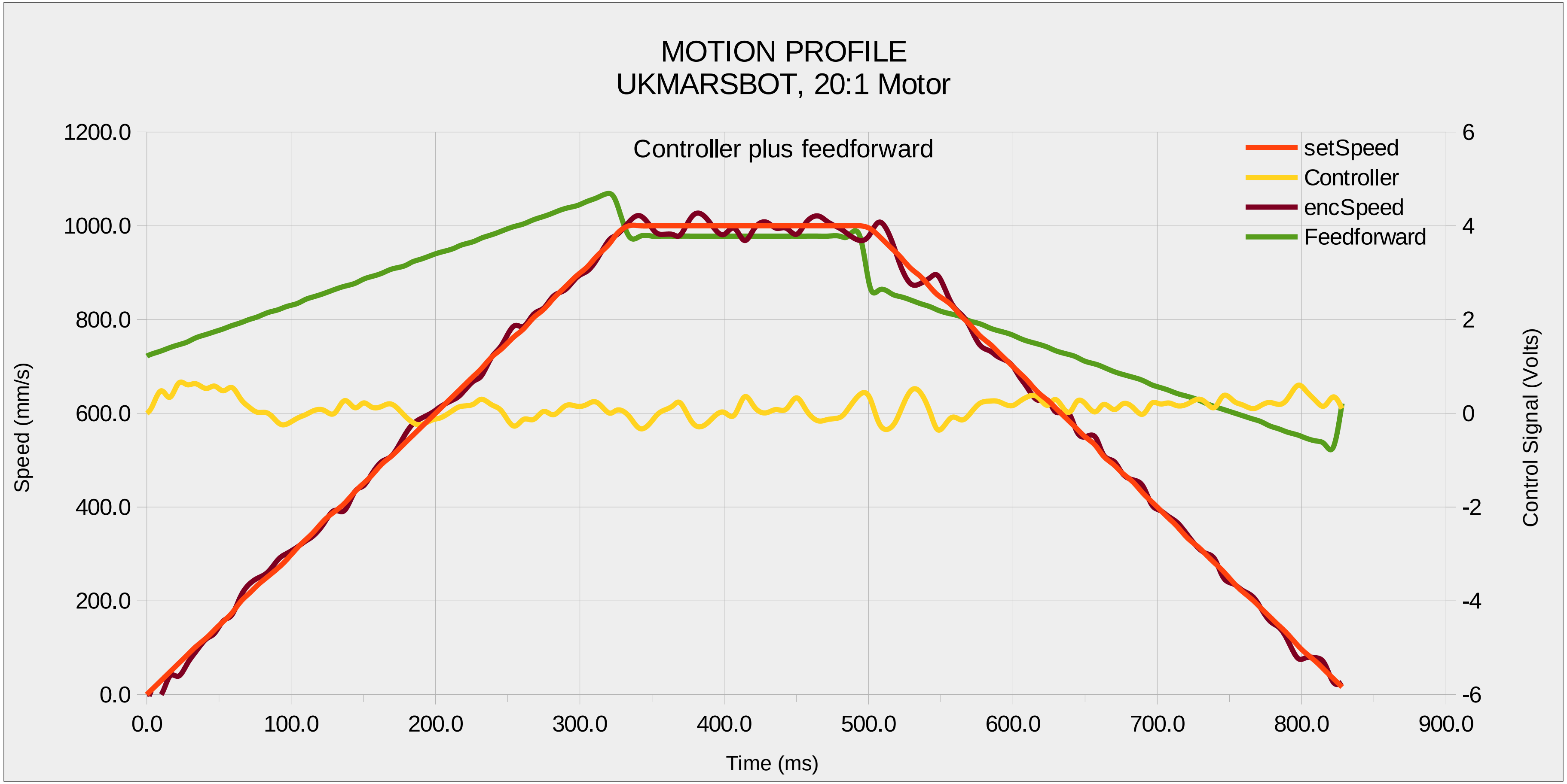
https://www.pid-tuner.com/pid-control/



Peter Harrison, UKMARS, Sep 2022


MOTION PROFILE
UKMARSBOT, 20:1 motor
Controller only
setSpeed
Controller
encSpeed
Speed (mm/s)
Control Signal (Volts)
Time (ms)
1200.0
1000.0
800.0
600.0
400.0
200.0
0.0
100.0
200.0
300.0
400.0
500.0
600.0
700.0
800.0
900.0
6
4
2
0
-2
-4
-6

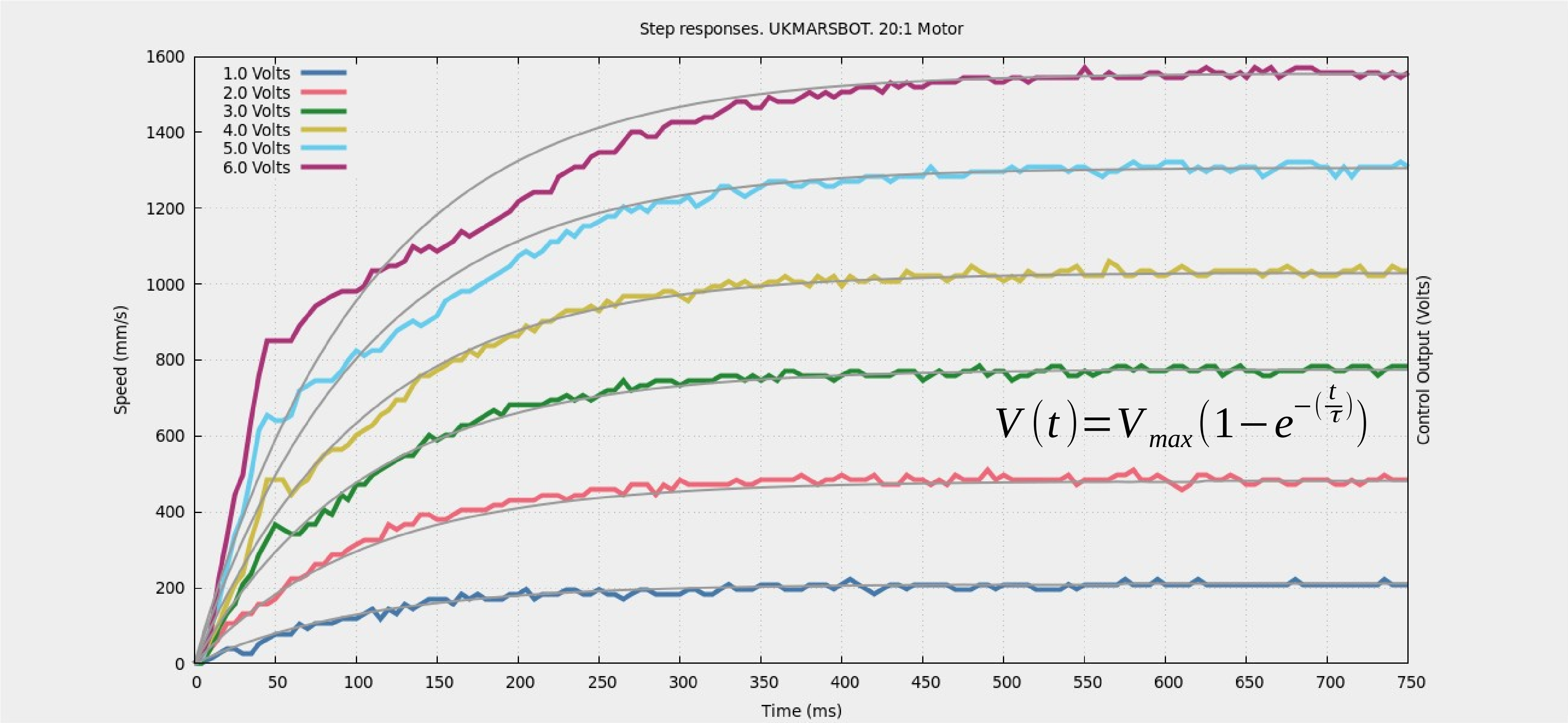

MOTION PROFILE
UKMARSBOT, 20:1 motor
Feedforward only
setSpeed
Controller
encSpeed
Feedforward
Speed (mm/s)
Control Signal (Volts)
Time (ms)
1200.0
1000.0
800.0
600.0
400.0
200.0
0.0
6
4
2
0
-2
-4
-6
0.0
100.0
200.0
300.0
400.0
500.0
600.0
700.0
800.0
900.0


MOTION PROFILE
UKMARSBOT, 20:1 Motor
Controller plus feedforward
setSpeed
Controller
encSpeed
Feedforward
Speed (mm/s)
1200.0
1000.0
800.0
600.0
400.0
200.0
0.0
0.0
100.0
200.0
300.0
400.0
500.0
600.0
700.0
800.0
900.0
Time (ms)
Control Signal (Volts)
6
4
2
0
-2
-4
-6

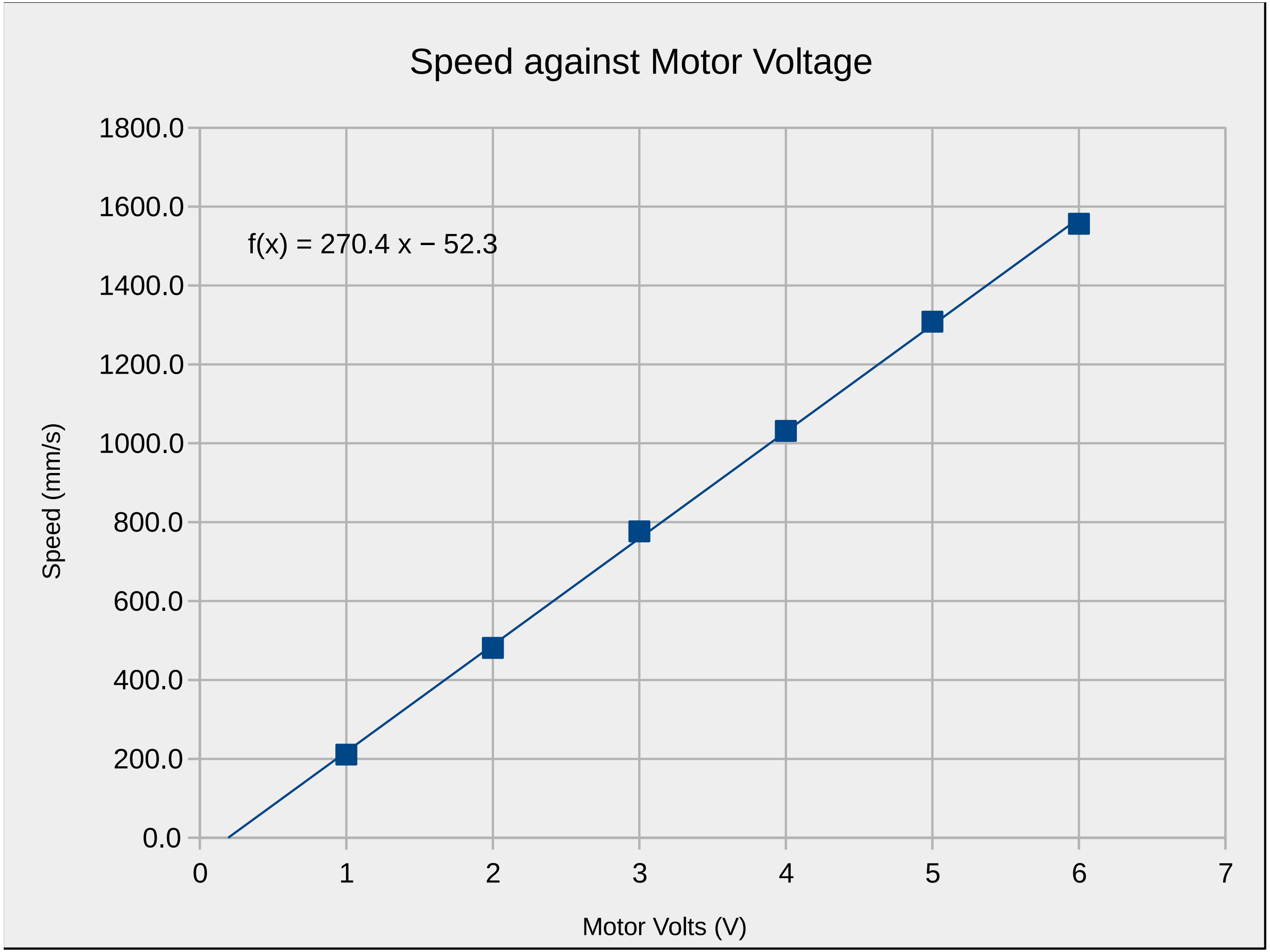
Characterise the Drive system

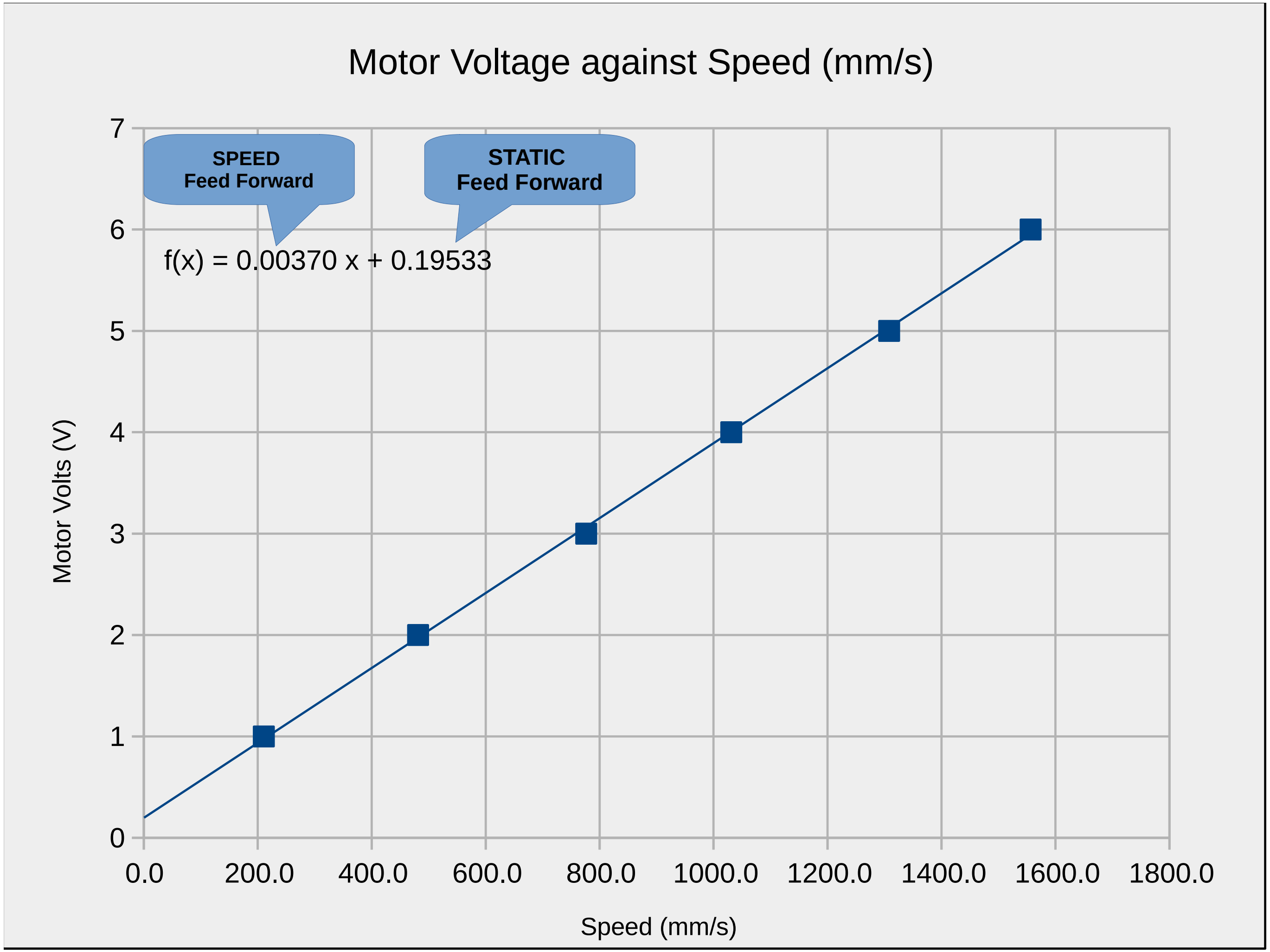
- Voltage => Speed
- Current => Torque
- Torque => Acceleration
- Speed response is simple 1st Order

Steps Again

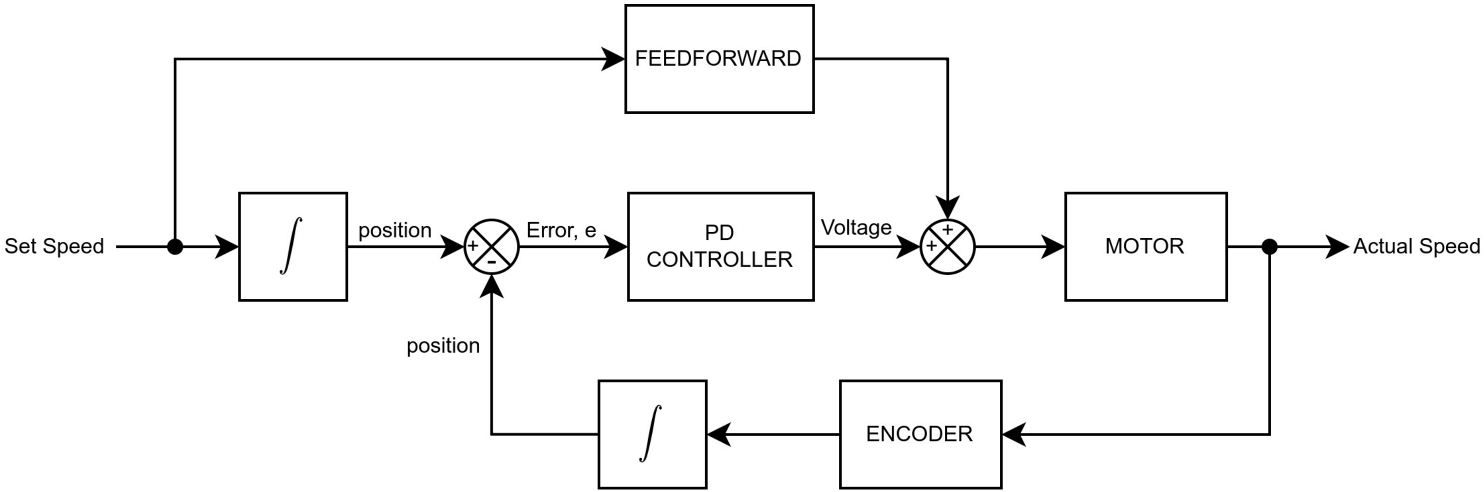


Step responses. UKMARSBOT. 20:1 Motor
1.0 Volts
2.0 Volts
3.0 Volts
4.0 Volts
5.0 Volts
6.0 Volts
Speed (mm/s)
Control Output (Volts)
Time (ms)
0
200
400
600
800
1000
1200
1400
1600
0
50
100
150
200
250
300
350
400
450
500
550
600
650
700
750
$V(t) = V_{max}\left(1 - e^{-\left(\frac{t}{\tau}\right)}\right)$


Speed against Motor Voltage
f(x) = 270.4 x − 52.3
1800.0
1600.0
1400.0
1200.0
1000.0
800.0
600.0
400.0
200.0
0.0
Speed (mm/s)
0
1
2
3
4
5
6
7
Motor Volts (V)


Motor Voltage against Speed (mm/s)
SPEED
Feed Forward
STATIC
Feed Forward
f(x) = 0.00370 x + 0.19533
Motor Volts (V)
Speed (mm/s)
0
1
2
3
4
5
6
7
0.0
200.0
400.0
600.0
800.0
1000.0
1200.0
1400.0
1600.0
1800.0

Feedback + Feed Forward



Implementation - UKMARSBOT(ish)

```
void update_motor_controllers() {
  pos_output = position_controller(); // the PD Controller
  left_output = pos_output;
  right_output = pos_output;
  v_fwd = forward_speed();
  v_left = v_fwd - (PI / 180.0) * MOUSE_RADIUS * v_rot;
  v_right = v_fwd + (PI / 180.0) * MOUSE_RADIUS * v_rot;
  left_output += SPEED_FF * v_left + BIAS_FF;
  right_output += SPEED_FF * v_right + BIAS_FF;
  set_right_motor_volts(right_output);
  set_left_motor_volts(left_output);
}
```

This is a simplification of the actual code

Summary

- Feedback controllers are essential for accuracy
- But can be hard to tune
- Feedforward controllers are simple to use
- But not as dependable
- Combine the two to get good overall control with least effort.

Thank You

Feedforward function complete

```
// For the right wheel
float get_right_feedforward(float speed) {
  static float old_speed = 0;
  float acc = (speed - old_speed) * LOOP_FREQUENCY;
  old_speed = speed;
  if (speed > 0) {
    right_FF = ACC_FF * acc + SPEED_FF * speed + BIAS_FF;
  } else {
    right_FF = ACC_FF * acc + SPEED_FF * speed - BIAS_FF;
  }
  return rightFF;
}
```

Acceleration Feed Forward Constant

- Response $V(t) = V_{max}\left(1 - e^{-\left(\frac{t}{\tau}\right)}\right)$
- Acceleration $A(t) = \frac{V_{max}}{\tau} e^{-\left(\frac{t}{\tau}\right)}$
- At t = 0: $A_o = \frac{V_{max}}{\tau}$
- Plot Voltage against Acceleration
- Slope is Acceleration Constant $K_{acc} = K_{ff}\,\tau$

Position Controller

Implementing a PD controller is not hard

```
float position_controller() {
  s_fwd_error += forward.increment() - robot_fwd_increment();
  float diff = s_fwd_error - s_old_fwd_error;
  s_old_fwd_error = s_fwd_error;
  float output = FWD_KP * s_fwd_error + FWD_KD * diff;
  return output;
}
```