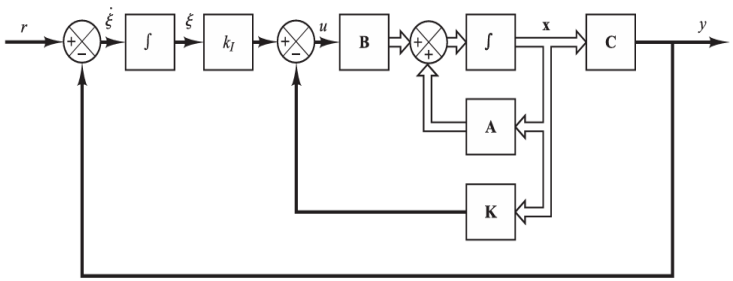
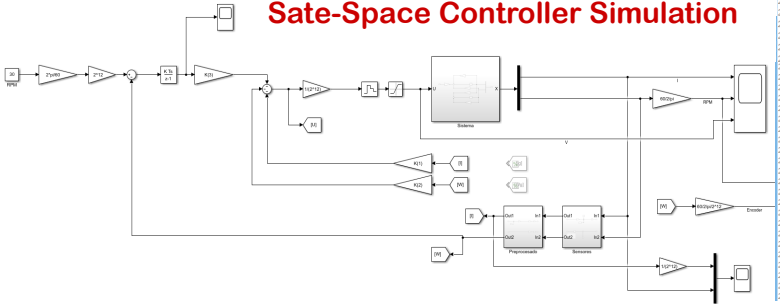
Theroretical Sate-Space Controller



Sate-Space Controller Simulation



Sate-Space Controller Implementation

```
void StateSpaceController(control_param_t* param, current_values_t* current, data_shared_t* data)
{
    char zero_r, zero_l;
    //Check the speed reference, for very low speed impose 0 V output
    if (data->speed_ref_r == 0){
        zero_r = 1;
    } else {
        zero_r = 0;
    }
    if (data->speed_ref_l == 0){
        zero_l = 1;
    } else {
        zero_l = 0;
    }
    //Calculates Speed error
    int error_r = (data->speed_ref_r - data->speed_r);
    int error_l = (data->speed_ref_l - data->speed_l);
    //Apply antiwindup saturation action
    error_r = error_r - 10*param->windup_r;
    error_l = error_l - 10*param->windup_l;
    //Integrates the error (Servo System Strategy "Ogata" -> error ~= 0 in steady state)
    IntegralBackwardEuler(param, error_r, error_l, zero_r, zero_l);
    //Windup Action
    IntegralWindup(param, INTEGRAL_SATURATION);
    //param->x0w_r = param->x0_r;
    //param->x0w_l = param->x0_l;
    //Summation of the State Space Controller terms before control action
    //Summation from Integral part
    int sum_r = - SignedRightShift(param->KI*param->x0w_r, 11); //Control parameter was 11 bits left shifted
    int sum_l = - SignedRightShift(param->KI*param->x0w_l, 11);
    //Summation from x2 State (Wheel Speed) feedback
    sum_r += - SignedRightShift(param->K_speed*data->speed_r, 11); //Control parameter was 11 bits left shifted
    sum_l += - SignedRightShift(param->K_speed*data->speed_l, 11);
    //Summation from x1 State (Motor Current) feedback
    sum_r += - SignedRightShift(param->K_current*current->I_r, 11); //Control parameter was 11 bits left shifted
    sum_l += - SignedRightShift(param->K_current*current->I_l, 11);
    //Transforming control action from voltage to PWM (1023/14 ~= 73 -> 0.9% error)
    //Force the output voltage to be set at 0 V (if the reference speed is 0)
    if (zero_r)
    {
        param->V_r = 0;
    }else{
        //Compensating the 12 left bits originally shifted (in current and speed)
        param->V_r = SignedRightShift(73*sum_r, 12);
    }
    if (zero_l)
    {
        param->V_l = 0;
    }else{
        //Compensating the 12 left bits originally shifted (in current and speed)
        param->V_l = SignedRightShift(73*sum_l, 12);
    }

    //OPEN LOOP
    /*
    param->V_r = data->speed_ref_r;
    param->V_l = data->speed_ref_l;
    */
    //It makes the changes to apply the desired voltage in the motors
    VoltageApplication(param);
}
```