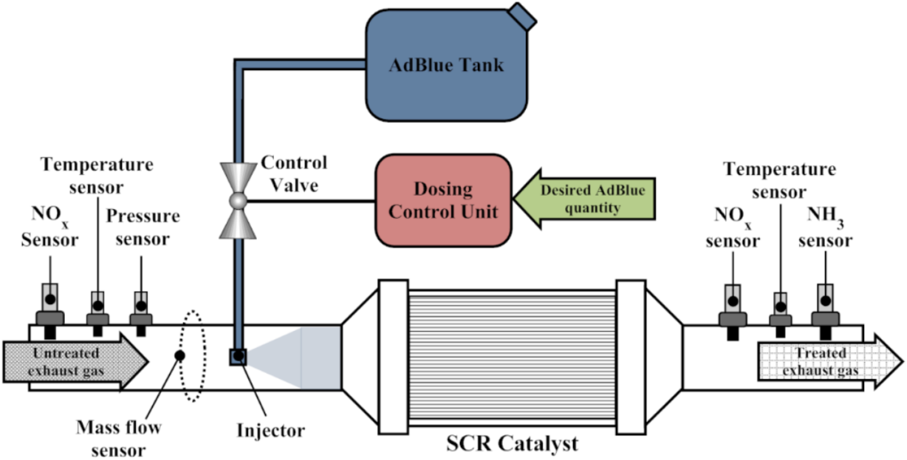
# Systematic Catalytic Reduction by Reinforcement Learning

Chris Guiver, Baruch Gutierrez, Allen Hart, James Hook, Uwe Martin, Jordan Taylor

June 14, 2019

# The Problem


AdBlue Tank
Control Valve
Dosing Control Unit
Desired AdBlue quantity
Temperature sensor
$NO_x$ Sensor
Pressure sensor
Temperature sensor
$NO_x$ sensor
$NH_3$ sensor
Untreated exhaust gas
Treated exhaust gas
Mass flow sensor
Injector
SCR Catalyst

# Original Model

$$
\begin{aligned}
\frac{d}{dt}c_{NO,k} &= \frac{n}{V_c \cdot \varepsilon_g} \cdot \frac{\overset{*}{m}_{EG} \cdot R}{p_{EG} \cdot M_{EG}} \left(T_{EG,k-1} \cdot c_{NO,k-1} - T_{c,k} \cdot c_{NO,k}\right) + \\
&\quad + a_R \left(-4 \cdot r_{std,k} - 2 \cdot r_{fst,k} - r_{NO,g,k}\right) \\
\frac{d}{dt}c_{NO_2,k} &= \frac{n}{V_c \cdot \varepsilon_g} \cdot \frac{\overset{*}{m}_{EG} \cdot R}{p_{EG} \cdot M_{EG}} \left(T_{EG,k-1} \cdot c_{NO_2,k-1} - T_{c,k} \cdot c_{NO_2,k}\right) + \\
&\quad + a_R \left(-2 \cdot r_{fst,k} - 6 \cdot r_{slw,k} + r_{NO,g,k}\right) \\
\frac{d}{dt}c_{NH_3,k} &= \frac{n}{V_c \cdot \varepsilon_g} \cdot \frac{\overset{*}{m}_{EG} \cdot R}{p_{EG} \cdot M_{EG}} \left(T_{EG,k-1} \cdot c_{NH_3,k-1} - T_{c,k} \cdot c_{NH_3,k}\right) + \\
&\quad + a_R \left(-r_{ad,k} + r_{de,k} - 4 \cdot r_{ox,g,k}\right) \\
\frac{d}{dt}c_{O_2,k} &= \frac{n}{V_c \cdot \varepsilon_g} \cdot \frac{\overset{*}{m}_{EG} \cdot R}{p_{EG} \cdot M_{EG}} \left(T_{EG,k-1} \cdot c_{O_2,k-1} - T_{c,k} \cdot c_{O_2,k}\right) + \\
&\quad + a_R \left(-0.5 \cdot r_{NO,g,k}\right) \\
\frac{d}{dt}\theta_{NH_3,k} &= \frac{1}{\Theta_{NH_3}} \left(r_{ad,k} - r_{de,k} - 4 \cdot r_{std,k} - 4 \cdot r_{fst,k} - 8 \cdot r_{slw,k} - 4 \cdot r_{ox,k}\right) \\
\frac{d}{dt}T_{c,k} &= \frac{n}{m_c \cdot c_{p,c}} \left(\overset{*}{m}_{EG} \cdot c_{p,EG} \cdot (T_{EG,k-1} - T_{c,k}) + \alpha_c \cdot a_c \cdot (T_{Amb} - T_{c,k})\right)
\end{aligned}
$$

# Simplified Model

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{T} = -a(\mathrm{T} - 200) + b\mathrm{NO}_x^{in}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{NO}_x^{cat} = \mathrm{NO}_x^{in} - \mathrm{NO}_x^{out} - \alpha R(\mathrm{T}, \mathrm{NO}_x^{cat}, \mathrm{NH}_3^{load})$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{NH}_3^{load} = \mathrm{NH}_3^{in} - \mathrm{NH}_3^{out} - R(\mathrm{T}, \mathrm{NO}_x^{cat}, \mathrm{NH}_3^{load})$$

$$R(\mathrm{T}, \mathrm{NO}_x^{cat}, \mathrm{NH}_3^{load}) = f(\mathrm{T})\mathrm{NO}_x^{cat}\mathrm{NH}_3^{load}$$

# Simplified Model

$NO_x^{in}$
$NH_3^{in}$

$\longrightarrow$

Temp
$NO_x^{cat}$
$NH_3^{load}$

$\longrightarrow$

$NO_x^{out}$
$NH_3^{out}$

**Challenge:** Correctly control the amount of $NH_3^{in}$ injected into the system at every time step $t$.

**Challenge:** Correctly control the amount of $NH_3^{in}$ injected into the system at every time step $t$.

**Idea:** Use Reinforcement-Learning to train a controller capable of correctly deciding the optimal amount.

# Q-Learning

Reinforcement Learning involves an agent, moving through a state space, $\mathcal{S}$, by selecting an action from an action space, $\mathcal{A}$, at each state.

Given we are at some state $s_t \in \mathcal{S}$, taking an action $a_t \in \mathcal{A}$ provides the agent with the reward $r = r(s, a)$ and new state $s_{t+1}$.

The aim of the agent is to maximise the total (future) reward.

Idea:

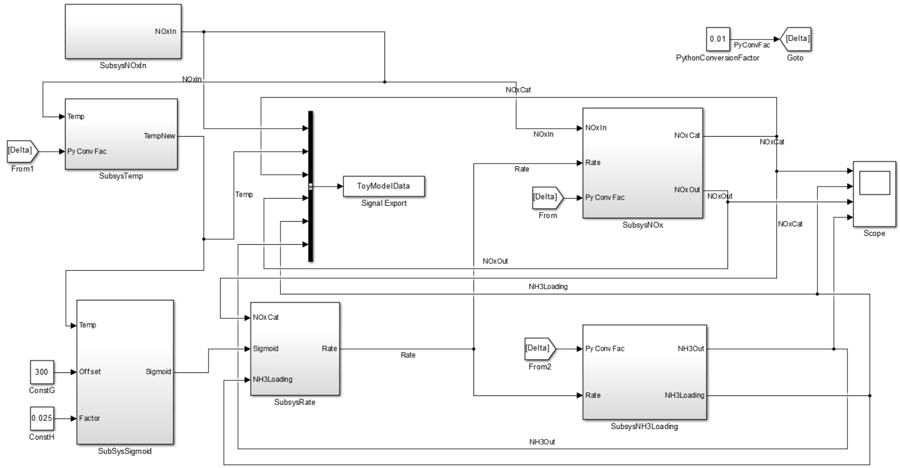
$$Q(s,a) \approx \mathbb{E}\begin{bmatrix}\text{future discounted sum of rewards if we start at state} \\ \text{and then follow current policy for the rest of time}\end{bmatrix}$$

Ideally:

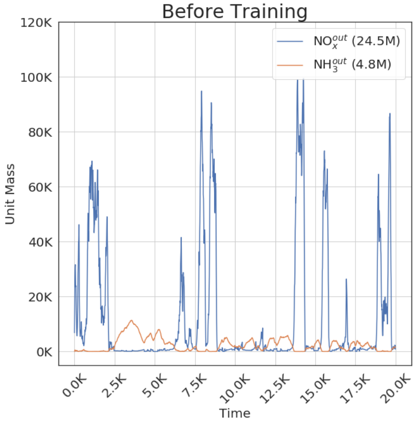
$$Q(s_t, a_t) = r_t + \gamma \max_a \mathbb{E}[Q(s_{t+1}, a)]$$

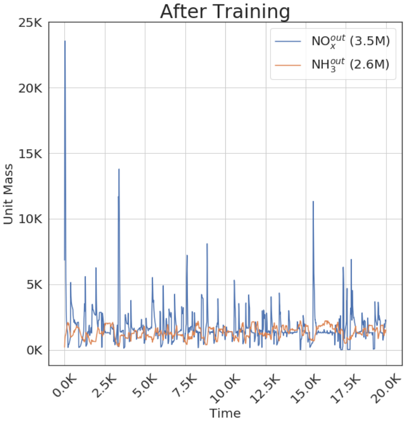
The objective of the training is to update $Q$ iteratively to take into account future values of $Q$, i.e. to correctly reflect the value of rewards available after multiple actions.

$$Q(s_t, a_t) \longleftarrow Q(s_t, a_t) - \alpha\left(Q(s_t, a_t) - r_t - \gamma \cdot \max_a \mathbb{E}[Q(s_{t+1}, a)]\right)$$

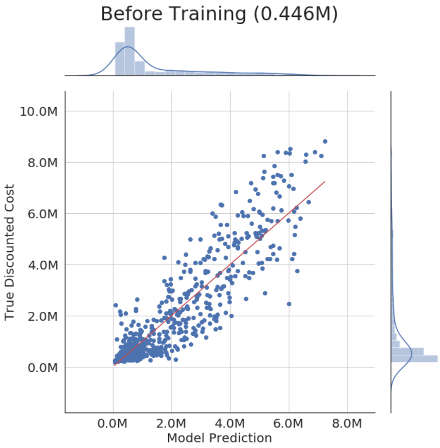

NOxIn
SubsysNOxIn
NOxIn
Temp
TempNew
[Delta]
Py Conv Fac
From1
SubsysTemp
NOxCat
Temp
ToyModelData
Signal Export
NOxIn
NOxCat
Rate
[Delta]
Py Conv Fac
NOxOut
From
SubsysNOx
NOxIn
NOxCat
Rate
NOxOut
0.01
PyConvFac
[Delta]
PythonConversionFactor
Goto
Scope
NOxCat
NOxOut
NH3Loading
Temp
300
Offset
ConstG
0.025
Factor
ConstH
Sigmoid
SubSysSigmoid
NOx Cat
Sigmoid
Rate
NH3Loading
SubsysRate
Rate
[Delta]
From2
Py Conv Fac
NH3Out
Rate
NH3Loading
SubsysNH3Loading
NH3Out

# Results I - Output Gases

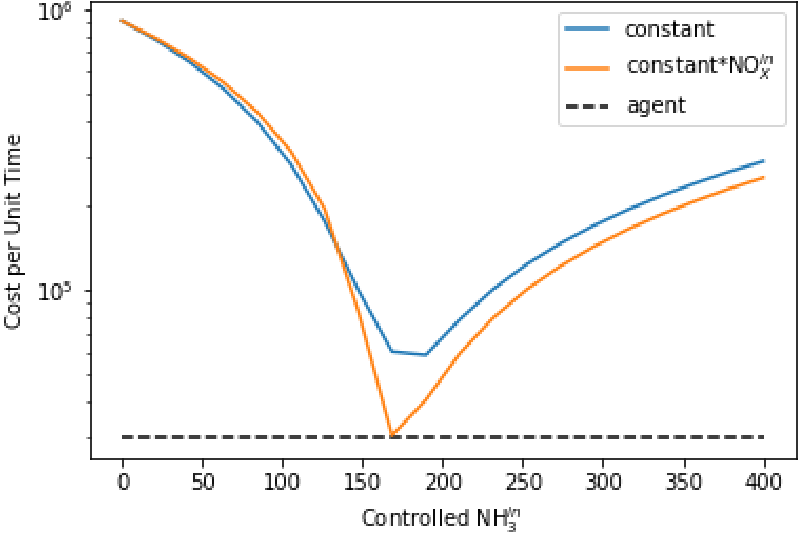


# Results II - Predicted vs True Discounted Cost



# Results III - Comparison of Policies



# Remarks

Pros

- Our agent can learn a policy governed by a system of differential equations without seeing them.
- Ability to have "online" learning to cater policy to the user.
- Cheap evaluation to determine appropriate control.

Cons

- A parameter space to search i.e. discount factor $\gamma$, learning rate $\alpha$, and exploration rate $\epsilon$.
- Long training time.

Future

- Tune the toy model to be more realistic.
- Use the original set of differential equations.
- Allow noisy measurements.