Derivation of Butler-Volmer Equation

$$R \rightarrow O + ne^-$$

where $R = \sum s_{iR} R_i^{z_{iR}}$ = Reduced State

$O = \sum s_{iO} O_i^{z_{iO}}$ = Oxidized State

$\Rightarrow$ s_i : Stoichiometric coefficient of species i ($+s_i \rightarrow R$; $-s_i \rightarrow O$)

$\Rightarrow$ $R_i(O_i)$: Symbol for chemical formula

$\Rightarrow$ Z_i : Charge number of species i.

$$R = k_a C_R e^{(1-\alpha) ne\Delta\phi/k_BT} - k_c C_O e^{-\alpha ne\Delta\phi/K_BT} = R_a - R_c = \frac{\#\text{reactions}}{\text{time site}}$$

where $C_R = \prod_i c_{iR}^{s_{iR}}$, $C_O = \prod_i c_{iO}^{s_{iO}}$, $\Delta\phi = \phi_e - \phi$ electrod-electrolyte potential

α = Symmetry factor (transfer coefficient)

$$\frac{R_a}{R_c} = \frac{k_a C_R}{k_c C_O} e^{ne\Delta\phi/k_BT}$$ {Anode cathode reaction rate ratio does not depend on α or any properties of transition state.}

$$\Delta\phi_{eq} = \frac{k_BT}{ne} \ln\left(\frac{k_c C_O}{k_a C_R}\right) = V^\circ - \frac{k_BT}{ne} \ln\left(\frac{\prod_i c_{iR}^{s_{iR}}}{\prod_i c_{iO}^{s_{iO}}}\right)$$ where $V^\circ = \frac{k_BT}{ne} \ln\left(\frac{k_c}{k_a}\right)$

$\hookrightarrow$ Nernst Equation

* Reaction Rate in terms of Activation Overpotential

$$\eta = \Delta\phi - \Delta\phi_{eq}$$

$$R = k_a C_R e^{(1-\alpha)ne(\eta + \Delta\phi_{eq})/k_BT} - k_c C_O e^{-\alpha ne(\eta+\Delta\phi_{eq})/k_BT}$$

using $\Delta\phi_{eq} = \frac{k_BT}{ne} \ln\left(\frac{k_c C_O}{k_a C_R}\right)$

$$R = k_a C_R \left(\frac{k_c C_O}{k_a C_R}\right)^{1-\alpha} e^{(1-\alpha)ne\eta/k_BT} - k_c C_O \left(\frac{k_c C_O}{k_a C_R}\right)^{-\alpha} e^{-\alpha ne\eta/k_BT}$$

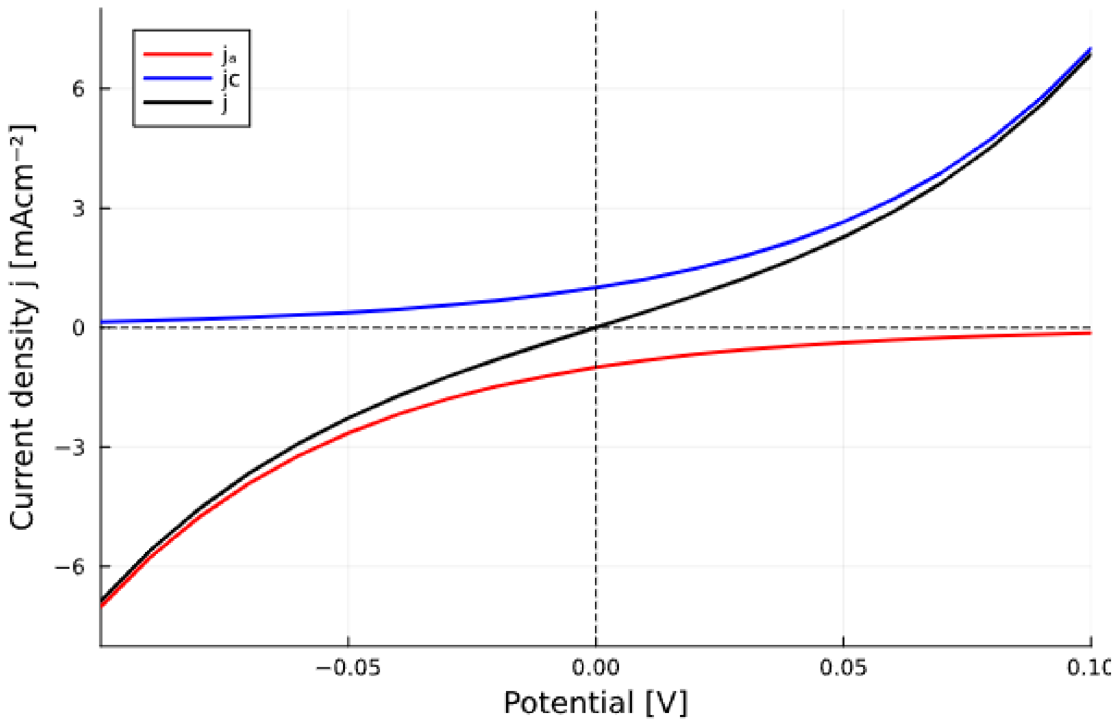
$$R = (k_c C_O)^{1-\alpha} (k_a C_R)^{\alpha} \left[e^{(1-\alpha)ne\eta/k_BT} - e^{-\alpha ne\eta/k_BT}\right]$$

$\therefore$ $I = neAR$ where $A \Rightarrow$ Electrode Area

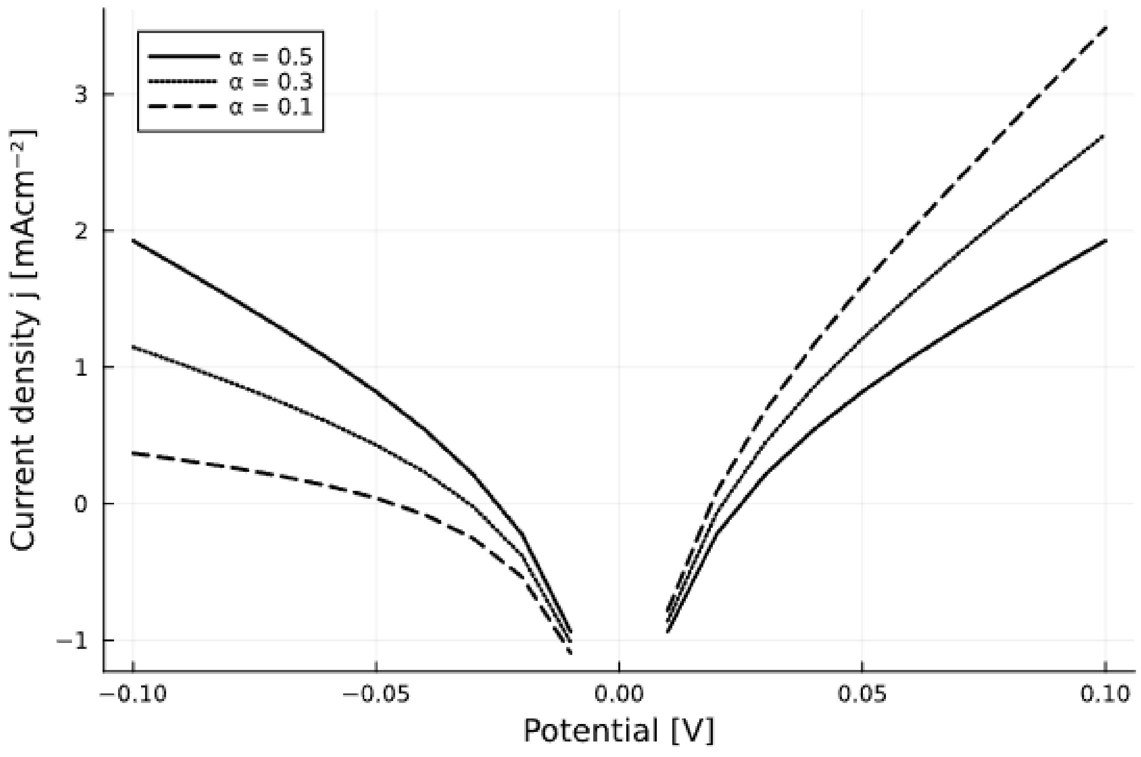
$$\boxed{I = I_0 \left[e^{(1-\alpha)ne\eta/k_BT} - e^{-\alpha ne\eta/k_BT}\right]}$$ Butler-Volmer Equation

where $I_0 = neA(k_c C_O)^{1-\alpha} (k_a C_R)^{\alpha}$ is exchange current in dilute solution.

Plots:



j_a
jc
j
Current density j [mAcm⁻²]
6
3
0
−3
−6
−0.05
0.00
0.05
0.10
Potential [V]



α = 0.5
α = 0.3
α = 0.1
Current density j [mAcm⁻²]
3
2
1
0
−1
−0.10
−0.05
0.00
0.05
0.10
Potential [V]

Source Code:

```julia
# load in some dependency packages
using DelimitedFiles
using Statistics
using ElectrochemicalKinetics

α = 0.5
α1 = 0.3
α2 = 0.1
j₀ = 1
ΔV = range(-0.1, 0.1; step=0.01)
bv = ButlerVolmer(j₀,α)
bv1 = ButlerVolmer(j₀,α1)
bv2 = ButlerVolmer(j₀,α2)

iF = bv.(ΔV) # full cell
iF1 = bv1.(ΔV) # full cell
iF2 = bv2.(ΔV) # full cell
ic = bv.(ΔV,true) # Cathodic current
ia = iF - ic # anodic current
j0=log.(abs.(iF))
j1=log.(abs.(iF1))
j2=log.(abs.(iF2))
using Plots
# ΔV: Potential [V]
# ia: Anodic current density [mAcm⁻²]
# ic: Cathodic current density [mAcm⁻²]
# iF: Total current density [mAcm⁻²]

# Create the plot
plot(ΔV, ia,
    label="jₐ",
    color="red",
    xlabel="Potential [V]",
    ylabel="Current density j [mAcm⁻²]",
    legend=:topleft, # Adjust the legend position as needed
    linewidth=2
)

plot!(ΔV, ic,
    label="jc",
    color="blue",
    linewidth=2
)
```

```julia
plot!(ΔV, iF,
    label="j",
    color="black",
    linewidth=2
)

# Add a horizontal line at j = 0
hline!([0], linestyle=:dash, color="black", label="") # remove the line label
#Add a vertical line at V = 0
vline!([0], linestyle=:dash, color="black", label="")
# Set axis limits (optional, based on the image)
xlims!(-0.1, 0.1)
ylims!(-8, 8)
# Display the plot
display(plot!())

# Create the plot
plot(ΔV, reverse(j0),
    label="α = 0.5",
    color="black",
    xlabel="Potential [V]",
    ylabel="Current density j [mAcm⁻²]",
    legend=:topleft, # Adjust the legend position as needed
    linewidth=2
)
plot!(ΔV, reverse(j1),
    label="α = 0.3",
    color="black",
    xlabel="Potential [V]",
    ylabel="Current density j [mAcm⁻²]",
    linestyle=:dot,
    legend=:topleft, # Adjust the legend position as needed
    linewidth=2
)
plot!(ΔV, reverse(j2),
    label="α = 0.1",
    color="black",
    xlabel="Potential [V]",
    ylabel="Current density j [mAcm⁻²]",
    linestyle=:dash,
    legend=:topleft , # Adjust the legend position as needed
    linewidth=2
)
```

Nernst Potential:

```julia
using IdealGas
function
Nernst_Potential(SMF::Vector{Float64},therm_file::String,T::Float64)
    # species used to calculate the Nernst potential
    species = ["H2", "H2O", "O2","CO","CO2"]
    #SMF     = [FE["ImF"][1],FE["ImF"][2],0.21]
    T0      = 25.0+273.0 # T0 is used to calculate the reference Gibbs
free energy

    # sp_trd  = create_transport_data(species, tr_file) # species
transport data
    thermo_obj = create_thermo(species, therm_file)   # thermal
properties for Cp

    E0H2 = IdealGas.E0_H2(thermo_obj,T0)
    OCVH2 = IdealGas.nernst(E0H2, T;pH2 = SMF[1], pO2 = SMF[3], pH2O =
SMF[2])

    E0CO = IdealGas.E0_CO(thermo_obj,T0)

    # Here note that the mole fractions in the fuel and air electrode
are used for the partial pressures.
    OCVCO = IdealGas.nernst_co(E0CO, T0;pCO = 0.9, pO2 = 0.21, pCO2 =
0.1)

    return OCVH2, OCVCO
end
```