In [1]:
```
using CSV
using DataFrames
using JuMP
using Plots
using Random
using Statistics
using LinearAlgebra
using Distributions
using BipartiteMatching
using Gurobi
using LinearAlgebra
using SymPy
using NLsolve
using LaTeXStrings
```

# Proof of Theorem 4

## Goal

The notebook aims to verify if

$$1-(1-B)\cdot e^{-2\alpha} < 2(1-B/2)\left[1-(1-(B-\delta)/2)\alpha - e^{-2(\alpha-\delta)(1-B/2)}\right] + (B-\delta) - 2/\alpha^f \left((e^{B/2\cdot\alpha^f}-1)\cdot e^{-(1-B/2)\alpha^f\cdot e^{-0.1\cdot(1-(B-\delta)/2)}}\right)$$

for $B \in \{0.4000, 0.4001, \ldots, 0.7999, 0.8000\}, \alpha \in \{0.0100, 0.0101, \ldots, 0.0499, 0.0500\}$ when $\alpha^f = 22$. That is, we verify

$$(1-B)\cdot e^{-2\alpha} + 2(1-B/2)\left[1-(1-(B-\delta)/2)\alpha - e^{-2(\alpha-\delta)(1-B/2)}\right] - 1 + (B-\delta) - 2/\alpha^f \left((e^{B/2\cdot\alpha^f}-1)\cdot e^{-(1-B/2)\alpha^f\cdot e^{-0.1\cdot(1-(B-\delta)/2)}}\right) > 0$$

for $B \in \{0.4000, 0.4001, \ldots, 0.7999, 0.8000\}, \alpha \in \{0.0100, 0.0101, \ldots, 0.0499, 0.0500\}$ when $\alpha^f = 22$.

## Arguments

- `alpha` : Value of $\alpha$.
- `alphaf` : Value of $\alpha^f$ .
- `B` : Value of $B$.
- `delta` : Small positive increment $\delta$.

## Functions

- `comp_f_func(B, alpha, delta)` returns the local lower bound of $\mu(B/2, B/2) - \mu(B, 0)$.
- `calculate_lb_comparison_matrix(delta)` examines all $B$ and $\alpha$ values within the claimed region.

## Outputs

- `calculate_lb_comparison_matrix(delta)` should print nothing if all values of $B$ and $\alpha$ pass the verification. This is indeed observed when delta = 0.0001.

In [2]:
```
# Define a Julia function to compute local lower bound of mu(B/2,B/2) - mu(B,0)
function comp_f_func(B, alpha, delta)
    lb_comparison_val = (1-B)*exp(-2*alpha) + 2*(1-B/2)*(1-(1-(B-delta)/2)*alpha-exp(-2*(alpha-delta)*(1-B/2))) - 1 + (B-delta) - (
                        2*((exp(B/2*alphaf)-1)*exp(-(1-B/2)*alphaf*exp(-0.1*(1-(B-delta)/2))))/alphaf)
    return lb_comparison_val
end

# This function iterates over B and alpha, printing out cases where the inequality is not satisfied.
function calculate_lb_comparison_matrix(delta)

    budget_range = 0.4:delta:0.8
    alpha_range = 0.01:delta:0.05

    for (i, B) in enumerate(budget_range)
        for (j, alpha) in enumerate(alpha_range)
            comparison = comp_f_func(B, alpha, delta)
            if comparison <= 0
                println(B, "\t", comparison)
            end
        end
    end
    return
end
```

Out[2]:
```
calculate_lb_comparison_matrix (generic function with 1 method)
```

In [3]:
```
alphaf = 22
delta = 0.0001
results_matrix = calculate_lb_comparison_matrix(delta)
```

In [ ]: