1 Benchmarking Available LCP Solvers

1.1 Load Dependencies

```
using BenchmarkTools, ComplementaritySolve, DataFrames, PyPlot, StableRNGs
```

1.2 Basic LCP

```
A_1 = [2.0 1; 1 2.0]
q_1 = [-5.0, 6.0]
```

```
2-element Vector{Float64}:
 -5.0
  6.0
```

1.2.1 Unbatched Version

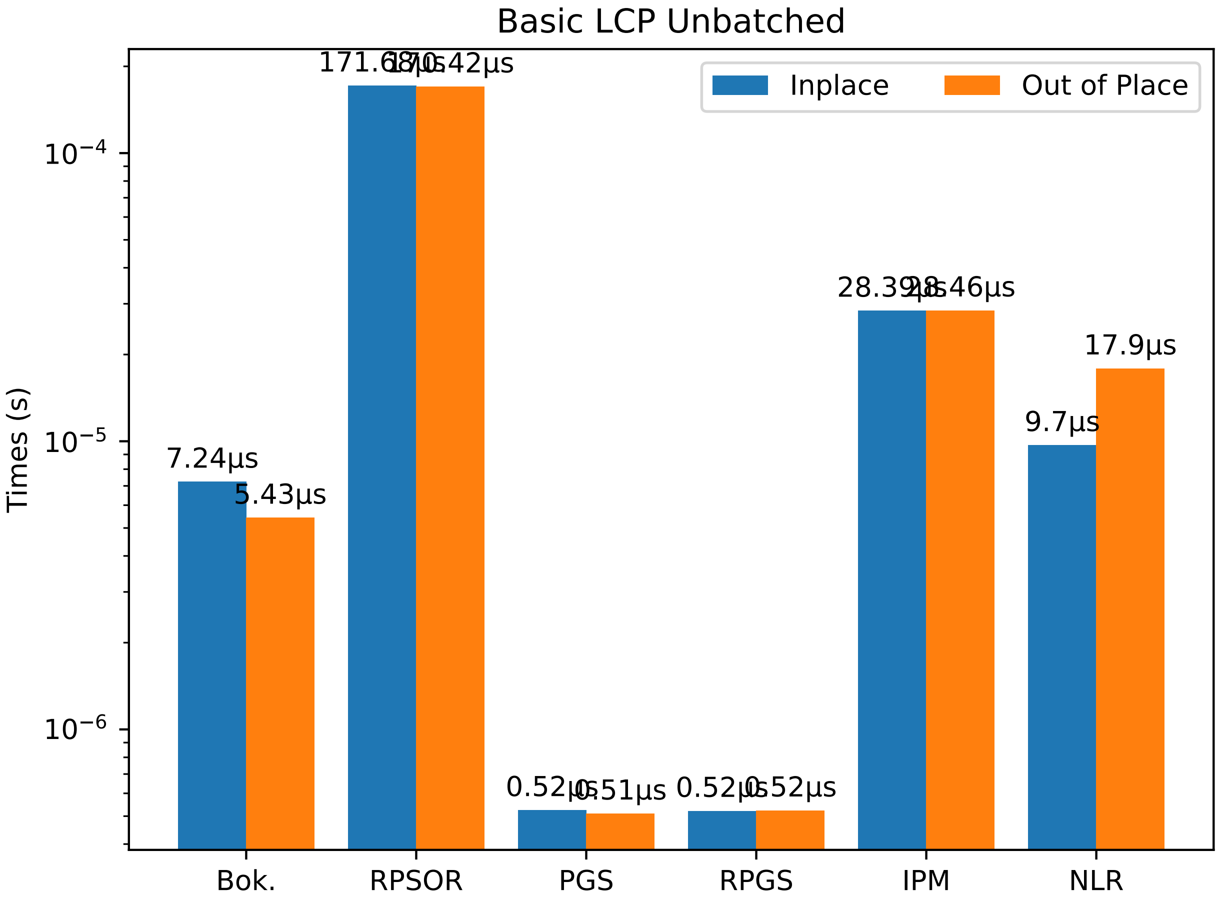
```
SOLVERS = [BokhovenIterativeAlgorithm(),
    RPSOR(; ω=1.0, ρ=0.1),
    PGS(),
    RPGS(),
    InteriorPointMethod(),
    NonlinearReformulation(),
]
times = zeros(length(SOLVERS), 2)
solvers = ["Bok.", "RPSOR", "PGS", "RPGS", "IPM", "NLR"]

for (i, solver) in enumerate(SOLVERS)
    prob_iip = LCP{true}(A_1, q_1, rand(StableRNG(0), 2))
    prob_oop = LCP{false}(A_1, q_1, rand(StableRNG(0), 2))
    times[i, 1] = @belapsed solve($prob_iip, $solver)
    times[i, 2] = @belapsed solve($prob_oop, $solver)
end

xloc = 1:length(solvers)
width = 0.4  # the width of the bars
multiplier = 0
fig, ax = subplots(layout="constrained")
ax.set_yscale("log")

for (i, group) in enumerate(["Inplace", "Out of Place"])
    global multiplier
    offset = width * multiplier
    rects = ax.bar(xloc .+ offset, times[:, i], width, label=group)
    for (j, rect) in enumerate(rects)
        height = rect.get_height()
        ax.annotate("$(round(times[j, i] * 10^6; digits=2))μs",
                    xy=(rect.get_x() + rect.get_width() / 2, height),
                    xytext=(0, 3),  # 3 points vertical offset
                    textcoords="offset points",
                    ha="center", va="bottom")
    end
    multiplier += 1
end
```

```
ax.set_ylabel("Times (s)")
ax.set_title("Basic LCP Unbatched")
ax.set_xticks(xloc .+ width ./ 2, solvers)
ax.legend(loc="upper right", ncols=3)
fig.tight_layout()
fig
```



1.2.2 Batched Version

Here we are batching the Problem with N starting values (typically batching LCPs involves batching multiple M and q)

```
SOLVERS = [BokhovenIterativeAlgorithm(),
    RPSOR(; ω=1.0, ρ=0.1),
    PGS(),
    RPGS(),
    InteriorPointMethod(),
    NonlinearReformulation(),
]
BATCH_SIZES = 2 .^ 1:2:11
times = zeros(length(SOLVERS), length(BATCH_SIZES), 2)
solvers = ["Bok.", "RPSOR", "PGS", "RPGS", "IPM", "NLR"]

for (i, solver) in enumerate(SOLVERS)
    for (j, N) in enumerate(BATCH_SIZES)
        prob_iip = LCP{true}(A_1, q_1, rand(StableRNG(0), 2, N))
        prob_oop = LCP{false}(A_1, q_1, rand(StableRNG(0), 2, N))
        times[i, j, 1] = @elapsed solve(prob_iip, solver)
        times[i, j, 2] = @elapsed solve(prob_oop, solver)
    end
end
```

```
Error: DimensionMismatch: arguments must have the same number of rows
```

```
xloc = 1:length(solvers)
width = 0.4  # the width of the bars
multiplier = 0
fig, ax = subplots(layout="constrained")
ax.set_yscale("log")

for (i, group) in enumerate(["Inplace", "Out of Place"])
    global multiplier
    offset = width * multiplier
    rects = ax.bar(xloc .+ offset, times[:, i], width, label=group)
    for (j, rect) in enumerate(rects)
        height = rect.get_height()
        ax.annotate("$(round(times[j, i] * 10^6; digits=2))μs",
                    xy=(rect.get_x() + rect.get_width() / 2, height),
                    xytext=(0, 3),  # 3 points vertical offset
                    textcoords="offset points",
                    ha="center", va="bottom")
    end
    multiplier += 1
end

ax.set_ylabel("Times (s)")
ax.set_title("Basic LCP Unbatched")
ax.set_xticks(xloc .+ width ./ 2, solvers)
ax.legend(loc="upper right", ncols=3)
fig.tight_layout()
fig
```

```
Error: BoundsError: attempt to access 6×5×2 Array{Float64, 3} at index [1:6
, 1]
```