25S 02: Reproduction of BGR Experiments and aGRAAL Implementation

JunHyun Kim

June 2025

1 Introduction

This report presents a focused study on the aGRAAL algorithm applied to variational inequality (VI) problems. The primary goal is to replicate and visualize key experiments from the BGR paper *"Beyond the Golden Ratio for VI Algorithms"*, using both provided `beyond_golden_ratio` code and a custom Julia implementation.

The experiments successfully reproduce Figures 2 through 6 from the original paper, covering a range of problem structures including linearly constrained quadratic programs, matrix games, and the so-called Polar Game and Forsaken examples. These highlight failure modes of GRAAL and OGDA and underscore the importance of parameter choices in VI solvers.

Figure 7 presents a successful reproduction of the Polar Game and Forsaken examples based entirely on a custom implementation of aGRAAL, along with trajectory visualizations.

2 Reproduction of Paper Figures

This section presents Figures 1 through 6 reproduced from the BGR paper using the provided `beyond_golden_ratio` code. Only minor adjustments were made to figure size and layout for clarity and formatting.

Each figure corresponds to a core experiment illustrating the convergence behavior of GRAAL, aGRAAL, and OGDA across various variational inequality settings. The reproduced visualizations closely match the original results and serve as validation for the implementation. Captions for Figures 1 through 6 are quoted directly from the original paper.

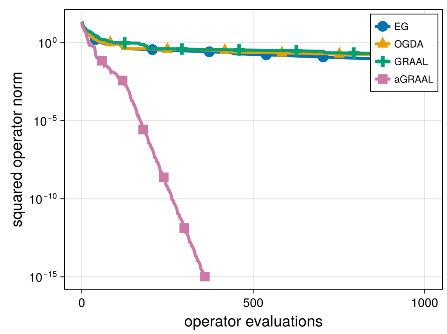

Figure 1: Policeman&Burglar matrix game example

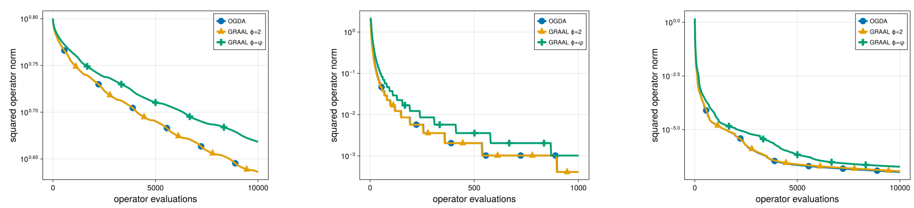

Figure 2: Left: The linearly constrained QP in (Yoon and Ryu, 2021). Middle: Test matrix given in (Nemirovski et al., 2009). Right: Randomly generated matrix game. Interestingly, even with constraints, GRAA and OGDA perform almost identically.

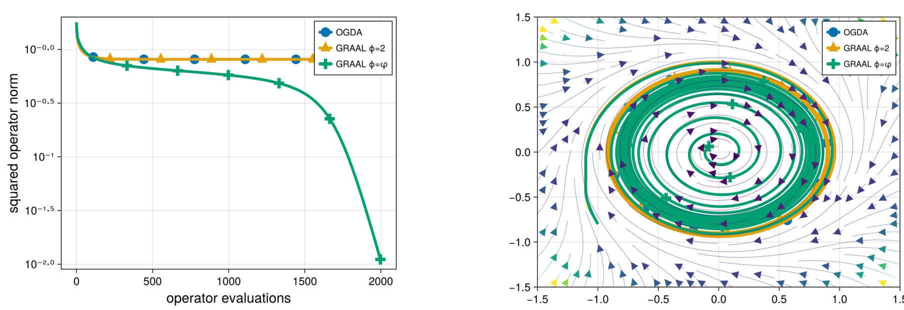

Figure 3: A special parametrization of the Polar Game example from (Pethick et al., 2022) (see Section 5.3.2 for details), showing the need to reduce ϕ for nonmonotone problems with weak Minty solutions.

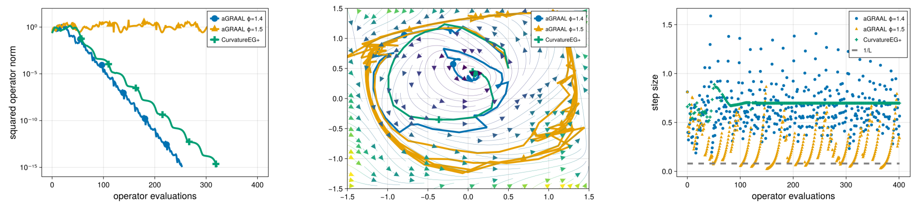

Figure 4: The "Forsaken" example of (Hsieh et al., 2021, Example 5.2), which is further explored in (Pethick et al., 2022). If ϕ is too large in aGRAAL, the method gets stuck in the limit cycle. Only for small ϕ the method converges, and does so rapidly.

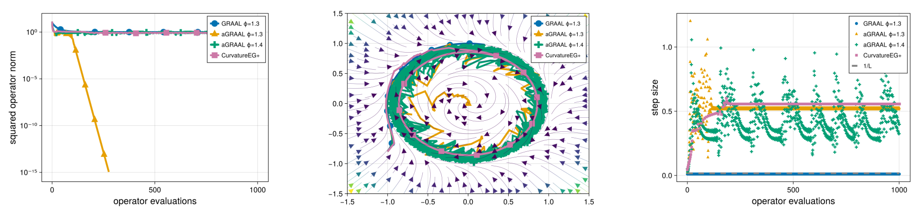

Figure 5: A different parametrization of the Polar Game, a = 3, showing the importance of adaptive step sizes. We observe that (aGRAAL) is the only method able to converge.

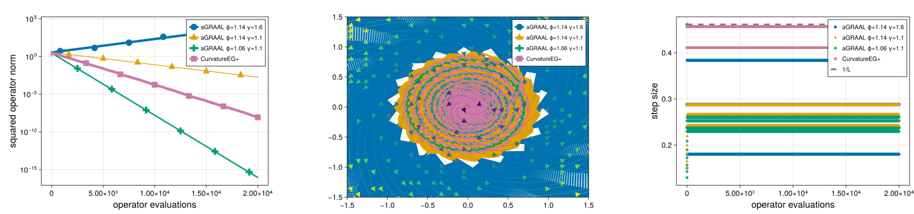

Figure 6: A special parametrization ($a^2 = 3.7, b = -1$) of (Pethick et al., 2022, Example 5), illustrating the fact that sometimes γ does indeed need to be chosen smaller as predicted by theory.

3 Custom Implementation

This section introduces a Julia implementation of the aGRAAL algorithm and its application to the Polar Game and Forsaken examples. The results confirm that aGRAAL's convergence behavior remains sensitive to the parameter ϕ, consistent with the findings reported in the BGR paper.

aGRAAL Implementation

The following Julia function implements the aGRAAL algorithm with adaptive step size, based on the formulation presented in the BGR paper.

```
function aGRAAL(F, proj, z0; ϕ=1.618, γ=1.2, α0=0.1, max_iter=500)
    z1 = proj(z0 - α0 * F(z0))
    z̄0 = z0

    zs = [z0, z1]
    zbars = [z̄0, ]
    alphas = [α0, ]
    thetas = [ϕ, ]

    for k in 2:max_iter

        diff_z = zs[k] - zs[k-1]
        diff_F = F(zs[k]) - F(zs[k-1])
        diff_F_norm = norm(diff_F)^2

        α1 = γ*alphas[k-1]
        α_candidate = diff_F_norm == 0 ? α1 : (ϕ * thetas[k-1] / (4 *
            alphas[k-1])) * norm(diff_z)^2 / diff_F_norm
        αk = min(α1, α_candidate)

        z̄k = (ϕ - 1)/ϕ * zs[k] + 1/ϕ * zbars[k-1]
        zk_1 = proj(z̄k - αk*F(zs[k]))
        θk = αk/alphas[k-1] * ϕ

        push!(zs, zk_1)
        push!(zbars, z̄k)
        push!(alphas, αk)
        push!(thetas, θk)
    end

    return zs, alphas, thetas
end
```

Visualization of Iterates

The code below visualizes the trajectory of iterates along with the underlying vector field defined by the VI problem. It highlights rotational behavior and convergence patterns.

```
function plot_iterates(zss::Vector{Tuple{Vector{Vector{Float64}},
    String}}, F::Function; minval=-1.5, maxval=1.5, title="aGRAAL
    Iterates")

    fig = Figure(size=(800, 600))
    ax = Axis(fig[1, 1],
              xlabel = "x", ylabel = "y", title = title,
              xticks = -1.5:0.5:1.5,
              yticks = -1.5:0.5:1.5
        )

    for zs in zss
```

```
        n = length(zs[1][1])
        xs = [z[1] for z in zs[1]]
        ys = [z[2] for z in zs[1]]

        lines!(ax, xs, ys, linewidth=2, label=zs[2])
        scatter!(ax, xs, ys, markersize=5)
    end

    f(z) = Point2f(-F(z)...)
    streamplot!(ax, f, minval..maxval, minval..maxval, colormap=[:grey
        ], linewidth=0.4)

    xlims!(minval, maxval)
    ylims!(minval, maxval)

    axislegend(ax)
    return fig
end
```

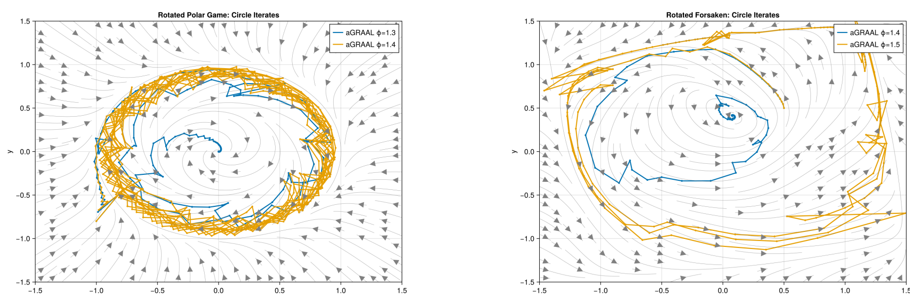

Figure 7: Left: Reproduction of the Polar Game experiment (Figure 5 in the BGR paper) with initial point $(-1,\ -0.8)$, where aGRAAL with $\phi = 1.3$ successfully converges, while $\phi = 1.4$ results in divergence. Right: Reproduction of the Forsaken example (Figure 4) with initial point $(0.5,\ 0.5)$, where $\phi = 1.4$ converges, but increasing to $\phi = 1.5$ leads to divergence. Both figures were generated using a custom Julia implementation, illustrating that the empirical behavior of aGRAAL remains sensitive to the choice of ϕ, despite its theoretically adaptive step size mechanism.

4 Conclusion

This report reproduced key experiments from the BGR paper and implemented the aGRAAL algorithm independently to examine its convergence behavior. The results confirm that aGRAAL can diverge for large values of ϕ, consistent with the original findings. These observations highlight the importance of careful parameter tuning, particularly when applying aGRAAL to nonmonotone variational inequality problems.