VariationalInequality – Julia for Variational Inequalities

Nov 05, 2018

Contents

CHAPTER 1

VariationalInequality: Julia package for solving variational inequality problems

This package, VariationalInequality.jl, implements solution algorithms for solving finite-dimensional variational inequality (VI) problems of the following form:

To find $x^* \in X$ such that

$$F(x^*)^\top (x - x^*) \geq 0 \qquad \forall x \in X$$

where the set X is defined by equalities and inequalities. The problem may be called $VI(F, X)$.

This package requires `JuMP` and `Ipopt`. Use `@variable` and `@mapping` to define x and $F(x)$, respectively. Use `@innerproduct` to match each variable and mapping and add to the problem. You can use `@innerproduct` multiple times.

For variational inequality problems for traffic user equilibrium, see TrafficAssignment.jl.

1.1 Installation

Please note that currently VariationalInequality is under development.

```
Pkg.add("JuMP")
Pkg.add("Ipopt")

Pkg.add("VariationalInequality")
```

See below for a few examples. Check the example folder in the github repository for more examples.

1.2 Example 1

Example 1 from Fukushima (1986).

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$F(x) = \begin{bmatrix} 2x_1 + 0.2x_1^3 - 0.5x_2 + 0.1x_3 - 4 \\ -0.5x_1 + x2 + 0.1x_2^3 + 0.5 \\ 0.5x_1 - 0.2x_2 + 2x_3 - 0.5 \end{bmatrix}$$

$$X = \{x : x_1^2 + 0.4x_2^2 + 0.6x_3^2 \leq 1\}$$

```
using VariationalInequality
using JuMP

m = VIPModel()

@variable(m, x1)
@variable(m, x2)
@variable(m, x3)

@constraint(m, const1, x1^2 + 0.4x2^2 + 0.6x3^2 <= 1)

@mapping(m, F1, 2x1 + 0.2x1^3 - 0.5x2 + 0.1x3 - 4)
@mapping(m, F2, -0.5x1 + x2 + 0.1x2^3 + 0.5)
@mapping(m, F3, 0.5x1 - 0.2x2 + 2x3 - 0.5)

@innerproduct(m, [F1, F2, F3], [x1, x2, x3])

// or
// @innerproduct(m, F1, x1)
// @innerproduct(m, F2, x2)
// @innerproduct(m, F3, x3)


sol1, Fval1, gap1 = solveVIP(m, algorithm=:fixed_point, max_iter=1000, step_size=0.1)
@assert 0<= gap1 < 1e-6

println("x1 = ", sol1[x1] )
println("x2 = ", sol1[x2] )
println("x3 = ", sol1[x3] )
```

1.3 Example 2

The example in Section 5.8 of Friesz (2010) Chapter 5. Finite Dimensional Variational Inequalities and Nash Equilibria.

$$\sum_{p=1}^{3} F_p(h^*)(h_p - h_p^*) \geq 0 \quad \forall h \in X$$

$$X = \left\{ h : \sum_{p=1}^{3} h_p = T_{14} \right\}$$

```
using JuMP, VariationalInequality

m = VIPModel()

A = [25; 25; 75; 25; 25]
B = [0.010; 0.010; 0.001; 0.010; 0.010]
T14 = 100
p = 3

@variable(m, h[i=1:p] >= 0)

# Add constraints to construct the feasible space
# The set X as in VI(F,X)
@constraint(m, sum{h[i], i=1:p} == T14)

# Define @mapping to be used for the mapping of the VI
# The mapping F as in VI(F,X)
@mapping(m, F1, A[1]+B[1]*h[1]^2 + A[4]+B[4]*(h[1]+h[2])^2 )
@mapping(m, F2, A[2]+B[2]*(h[2]+h[3])^2 + A[3]+B[3]*h[2]^2 + A[4]+B[4]*(h[1]+h[2])^2 )
@mapping(m, F3, A[2]+B[2]*(h[2]+h[3])^2 + A[5]+B[5]*(h[3])^2 )

# The order in F and h should match.
F = [F1, F2, F3]
@innerproduct(m, F, h)

# sol = the solution x^*
# Fval = F(x^*)
# gap = value of the gap function
sol, Fval, gap = solveVIP(m, algorithm=:extra_gradient, max_iter=1000, step_size=0.01)

@show sol
```

1.4 Example 3

Problem (15) with data in Table 1, Example 1, from Nagurney et al. (2014).

```
using JuMP, VariationalInequality

m = 2; n = 1

model = VIPModel()
```

(continues on next page)

(continued from previous page)

```
@variable(model, s[i=1:m] >=0)
@variable(model, d[j=1:n] >=0)
@variable(model, Q[i=1:m, j=1:n] >= 0)
@variable(model, q[i=1:m] >= 0)

@constraint(model, supply[i=1:m], s[i] == sum{Q[i,j], j=1:n})
@constraint(model, demand[j=1:n], d[j] == sum{Q[i,j], i=1:m})

as = [5; 2]
bs = [5; 10]
@mapping(model, pi[i=1:m], as[i] * s[i] + q[i] + bs[i])

ac = [1; 2]
bc = [15; 20]
@mapping(model, c[i=1:m, j=1:n], ac[i,j] * Q[i,j] + bc[i,j] )

ad = [2]
bd = [100]
@NLexpression(model, qhat[j=1:n], sum{q[i]*Q[i,j], i=1:m} / ( sum{Q[i,j], i=1:m} + 1e-
↪6 ) )
@mapping(model, nrho[j=1:n], ad[j] * d[j] - qhat[j] - bd[j] )

aq = [5; 10]
@mapping(model, Fq[i=1:m], aq[i] * q[i] - pi[i] )


@innerproduct(model, pi, s)
@innerproduct(model, c, Q)
@innerproduct(model, nrho, d)
@innerproduct(model, Fq, q)

for i=1:m, j=1:n
    setvalue(Q[i,j], 1.0)
end

sol1, Fval1, gap1 = solveVIP(model, algorithm=:fixed_point, max_iter=10000, step_
↪size=0.1, tolerance=1e-10)
@assert 0<= gap1 < 1e-6

@show gap1

@show sol1[Q[1,1]]
@show sol1[Q[2,1]]
@show sol1[q[1]]
@show sol1[q[2]]
```