MTH8408 : Méthodes d'optimisation et contrôle optimal

Laboratoire 4: Optimisation sans contraintes et méthodes itératives

Travail réalisé par Julien Pallage

Matricule: 2012861

3 mars 2024

In [ ]:
```
using Pkg
Pkg.activate(".") #Accède au fichier Project.toml
Pkg.instantiate()
Pkg.instantiate()
Pkg.status()
```

```
  Activating project at `~/Documents/code/MTH8408-Hiv24/lab4_JP`
Status `~/Documents/code/MTH8408-Hiv24/lab4_JP/Project.toml`
  [54578032] ADNLPModels v0.7.0
  [6e4b80f9] BenchmarkTools v1.5.0
  [c91e804a] Gadfly v1.4.0
  [7073ff75] IJulia v1.24.2
  [b6b21f68] Ipopt v1.6.2
  [10dff2fc] JSOSolvers v0.11.1
  [4076af6c] JuMP v1.20.0
  [ba0b0d4f] Krylov v0.9.5
  [40e66cde] LDLFactorizations v0.10.1
  [5c8ed15e] LinearOperators v2.7.0
  [a4795742] NLPModels v0.20.0
  [f4238b75] NLPModelsIpopt v0.10.1
  [792afdf1] NLPModelsJuMP v0.12.5
  [7cde8186] NLSProblems v0.5.1
^ [5049e819] OptimizationProblems v0.5.0
  [91a5bcdd] Plots v1.40.1
  [581a75fa] SolverBenchmark v0.6.0
  [ff4d7338] SolverCore v0.3.7
  [37e2e46d] LinearAlgebra
  [56ddb016] Logging
  [de0858da] Printf
Info Packages marked with ^ have new versions available and may be upgradable.
```

In [ ]:
```
using LinearAlgebra, Krylov, NLPModels, Printf, Logging, SolverCore, Test, ADNLPModels
```

In [ ]:
```
using BenchmarkTools, SolverCore, LinearOperators
using JSOSolvers, NLPModels #
using SolverBenchmark
using LinearAlgebra, NLPModels, Printf
#using OptimizationProblems.ADNLPProblems
using NLSProblems
```

Exercice 0: Introduction aux NLSModels

On a vu dans les lab précédents l'utilisation des NLPModels pour représenter un problème d'optimisation. Dans le cas de l'optimisation de moindre carrées non-linéaires, il existe un type spécifique: **NLSModel**.

$\min_x \frac{1}{2}\|F(x)\|^2$

Comme un NLPModel classique on peut faire appels aux fonctions: obj, grad, hprod ...

Mais on peut aussi utiliser des fonctions relatives à F: https://juliasmoothoptimizers.github.io/NLPModels.jl/stable/#Nonlinear-Least-Squares

L'équivalent des ADNLPModel pour ce cas est la fonction: ADNLSModel. Lien vers le site: https://juliasmoothoptimizers.github.io/ADNLPModels.jl/stable/

En utilisant les ADNLSModels écrire un modèle dont la fonction résidue est donné par FH ci-dessous.

In [ ]:
```
#Test problem:
FH(x) = [x[2]+x[1].^2-11, x[1]+x[2].^2-7]
x0H = [10., 20.]
###########################
#Utilise FH et x0H pour créer un ADNLSModel
nequ = length(FH(x0H))
himmelblau_nls = ADNLSModel(FH, x0H, nequ)
###########################


#some tests
print(residual(himmelblau_nls, x0H))
print(FH(x0H))
```

```
[109.0, 403.0][109.0, 403.0]
```

Exercice 1: Gauss-Newton

Dans cet exercice, on complète une implémentation de la méthode Gauss-Newton avec région de confiance (paramétrée par Δ) discutée en cours.

Il faut compléter les morceaux:

- utiliser les fonctions des NLSModels pour obtenir F et sa jacobienne (ici on utilise pas la jacobienne mais juste le produit jacobienne-vecteur). Parcourez la documentation de NLPModels pour déterminer la fonction adéquat, indice les fonctions pour les NLSModels indiquent des `nls` au lieu de `nlp` dans la documentation.
- Utiliser la fonction `lsmr` du package `Krylov.jl` pour résoudre le système linéaire avec une contrainte de `radius`. Lisez la documentation de `lsmr`.

In [ ]:
```
"""
Function that implements the Gauss-Newton algorithm for Nonlinear Least-square resolution.
"""
function gauss_newton(nlp      :: AbstractNLSModel,
                      x        :: AbstractVector,
                      ϵ        :: AbstractFloat = 1e-6;
                      η₁       :: AbstractFloat = 1e-3,
                      η₂       :: AbstractFloat = 0.66,
                      σ₁       :: AbstractFloat = 0.25,
                      σ₂       :: AbstractFloat = 2.0,
```

```julia
                      max_eval :: Int = 1_000,
                      max_time :: AbstractFloat = 60.,
                      max_iter :: Int = typemax(Int64)
                      )
    ######################################################
    Fx = residual(nlp, x)# le résidu
    Jx =  jac_residual(nlp, x) # operateur qui représente le jacobien du résidu
    ######################################################
    normFx = norm(Fx)

    Δ = 1.

    iter = 0

    el_time = 0.0
    tired   = neval_residual(nlp) > max_eval || el_time > max_time
    status  = :unknown

    start_time = time()
    too_small  = false
    normdual   = norm(Jx' * Fx)
    optimal    = min(normFx, normdual) ≤ ϵ

    @info log_header([:iter, :nf, :primal, :status, :nd, :Δ],
    [Int, Int, Float64, String, Float64, Float64],
    hdr_override=Dict(:nf => "#F", :primal => "‖F(x)‖", :nd => "‖d‖"))

    while !(optimal || tired || too_small)

        #################################
        #Compute a direction satisfying the trust-region constraint
        (d, stats)  = lsmr(-Jx,Fx,radius=Δ)
        #################################

        too_small = norm(d) < 1e-15
        if too_small #the direction is too small
            status = :too_small
        else
            xp      = x + d
            ###########################
            Fxp     =  residual(nlp, xp)# évalue le résidu en xp
            ###########################
            normFxp = norm(Fxp)

            Pred = 0.5 * (normFx^2 - norm(Jx * d + Fx)^2)
            Ared = 0.5 * (normFx^2 - normFxp^2)

            if Ared/Pred < η₁
                Δ = max(1e-8, Δ * σ₁)
                status = :reduce_Δ
            else #success
                x  = xp
                ###########################
                Jx = jac_residual(nlp, x)# réevalue le jacobien en x
                ###########################
                Fx = Fxp
                normFx = normFxp
                status = :success
                if Ared/Pred > η₂ && norm(d) >= 0.99 * Δ
                    Δ *= σ₂
                end
            end
        end

        @info log_row(Any[iter, neval_residual(nlp), normFx, status, norm(d), Δ])

        el_time      = time() - start_time
        iter   += 1

        many_evals   = neval_residual(nlp) > max_eval
        iter_limit   = iter > max_iter
        tired        = many_evals || el_time > max_time || iter_limit
        normdual     = norm(Jx' * Fx)
        optimal      = min(normFx, normdual) ≤ ϵ
    end

    status = if optimal
        :first_order
    elseif tired
        if neval_residual(nlp) > max_eval
            :max_eval
        elseif el_time > max_time
            :max_time
        elseif iter > max_iter
            :max_iter
        else
            :unknown_tired
        end
    elseif too_small
        :stalled
    else
        :unknown
    end

    return GenericExecutionStats(nlp; status, solution = x,
                                 objective = normFx^2 / 2,
                                 dual_feas = normdual,
                                 iter = iter,
                                 elapsed_time = el_time)
end
```

gauss_newton

On fait un premier test avec himmelblau.

In [ ]:
```julia
stats = gauss_newton(himmelblau_nls, himmelblau_nls.meta.x0, 1e-6)
@test stats.status == :first_order
```

```
┌ Info:   iter      #F    ‖F(x)‖           status       ‖d‖          Δ
└ @ Main /home/julien/Documents/code/MTH8408-Hiv24/lab4_JP/Lab4-notebook.ipynb:34
┌ Info:      0       3   3.8e+02          success   1.0e+00   2.0e+00
└ @ Main /home/julien/Documents/code/MTH8408-Hiv24/lab4_JP/Lab4-notebook.ipynb:75
┌ Info:      1       4   3.1e+02          success   2.0e+00   4.0e+00
└ @ Main /home/julien/Documents/code/MTH8408-Hiv24/lab4_JP/Lab4-notebook.ipynb:75
┌ Info:      2       5   1.9e+02          success   4.0e+00   8.0e+00
└ @ Main /home/julien/Documents/code/MTH8408-Hiv24/lab4_JP/Lab4-notebook.ipynb:75
┌ Info:      3       6   4.5e+01          success   7.7e+00   8.0e+00
└ @ Main /home/julien/Documents/code/MTH8408-Hiv24/lab4_JP/Lab4-notebook.ipynb:75
┌ Info:      4       7   9.5e+00          success   3.4e+00   8.0e+00
└ @ Main /home/julien/Documents/code/MTH8408-Hiv24/lab4_JP/Lab4-notebook.ipynb:75
┌ Info:      5       8   1.6e+00          success   1.3e+00   8.0e+00
└ @ Main /home/julien/Documents/code/MTH8408-Hiv24/lab4_JP/Lab4-notebook.ipynb:75
┌ Info:      6       9   1.2e-01          success   3.5e-01   8.0e+00
└ @ Main /home/julien/Documents/code/MTH8408-Hiv24/lab4_JP/Lab4-notebook.ipynb:75
┌ Info:      7      10   8.8e-04          success   3.0e-02   8.0e+00
└ @ Main /home/julien/Documents/code/MTH8408-Hiv24/lab4_JP/Lab4-notebook.ipynb:75
┌ Info:      8      11   5.3e-08          success   2.3e-04   8.0e+00
└ @ Main /home/julien/Documents/code/MTH8408-Hiv24/lab4_JP/Lab4-notebook.ipynb:75
Test Passed
```

On essaye notre algorithme sur le problème BNST2 de la librairie NLSProblems avec 400 variables.

In [ ]:
```
nls_model = BNST2(400)
stats = gauss_newton(nls_model, nls_model.meta.x0)
@test stats.status == :first_order
```

```
┌ Info:   iter      #F    ‖F(x)‖           status       ‖d‖          Δ
└ @ Main /home/julien/Documents/code/MTH8408-Hiv24/lab4_JP/Lab4-notebook.ipynb:34
┌ Info:      0       2   1.0e+00          success   1.0e+00   2.0e+00
└ @ Main /home/julien/Documents/code/MTH8408-Hiv24/lab4_JP/Lab4-notebook.ipynb:75
┌ Info:      1       3   4.2e-16          success   1.0e+00   2.0e+00
└ @ Main /home/julien/Documents/code/MTH8408-Hiv24/lab4_JP/Lab4-notebook.ipynb:75
Test Passed
```

Exercice 2: Méthode Levenberg-Marquard inexacte

Dans cet exercice, on complète une implémentation de la méthode Levenberg-Marquardt. Pour compléter le code `lm_param` on va utiliser les fonctions suivantes:

- `dsol` qui calcul la solution du système $\min_x \frac{1}{2}\|J(x)d + F(x)\| + \lambda\|x\|^2$ avec la fonction `lsqr` du package `Krylov.jl`.
- `multi_sol` qui pour un entier nl donné et un μ va résoudre le problème de dsol pour nl valeurs de λ (autour de la valeur μ). Par exemple, pour $\mu = 10^{-6}$ et $nl = 3$, on prendra $\lambda = 10^{-7}, 10^{-6}, 10^{-5}$. Parmis les `nl` directions calculées, on retourne celle qui donne la plus petite valeur de $\|F(x+d)\|^2$.

In [ ]:
```
function dsol(Fx, Jx, λ, τ)
    (d, stats) = lsqr(-Jx, Fx, λ = λ, atol = τ)
    return (d, stats)
end
```

```
dsol (generic function with 1 method)
```

In [ ]:
```
"""
Function that generate a list of lambdas according to the requirements.
Inputs: mean lambda, number of lambda to generate
Output: List of lambdas
"""
function generate_lambdas(λ :: Float64, nl :: Int64)
    nl = (mod(nl, 2) == 0) ? nl + 1 : nl + 0
    lambda_list = []
    push!(lambda_list, λ)
    half = trunc(nl/2)
    for i in range(start = 1, stop = half, step = 1)
        plus_lamb = (λ * 10^(i))
        minus_lamb = (λ * 10^(-i))
        append!(lambda_list, plus_lamb)
        append!(lambda_list, minus_lamb)
    end
    return lambda_list

end


"""
Function that generate the best direction according to different regularization parameters.
Inputs: nlp, x, Fx, Jx, lambda, Tau
Output: best direction
"""
function multi_sol(nlp, x, Fx, Jx, λ, τ; nl = 3)
    lambda_list = generate_lambdas(λ, Int(nl))
    count = 0
    best_NormFx = 0
    best_d = 0
    for lambd in lambda_list
        (d, stats)=  dsol(Fx, Jx, lambd, τ)
        nextX = x + d
        nextFx = residual(nlp, nextX)
        next_normFx = (norm(nextFx)).^2
        if count == 0
            best_NormFx = next_normFx
            best_d = d
        elseif (next_normFx < best_NormFx)
            best_NormFx = next_normFx
            best_d = d
        else
            continue
        end

        count += 1
    end
    return best_d
end
```

```
multi_sol
```

In [ ]:
```
"""
Function that implements the Levenberg-Marquard algorithm with LSQR.
"""
function lm_param(nlp      :: AbstractNLSModel,
                  x        :: AbstractVector,
                  ϵ        :: AbstractFloat = 1e-6;
```

```julia
                  η₁       :: AbstractFloat = 1e-3,
                  η₂       :: AbstractFloat = 0.66,
                  σ₁       :: AbstractFloat = 10.0,
                  σ₂       :: AbstractFloat = 0.5,
                  max_eval :: Int = 10_000,
                  max_time :: AbstractFloat = 60.,
                  max_iter :: Int = typemax(Int64)
                  )
    ######################################################
    Fx = residual(nlp, x)# le résidu
    Jx = jac_residual(nlp, x) # operateur qui représente le jacobien du résidu
    ######################################################
    normFx   = norm(Fx)
    normdual = norm(Jx' * Fx)

    iter = 0
    λ = 0.0
    λ₀ = 1e-6
    η = 0.5
    τ = η * normdual

    el_time = 0.0
    tired   = neval_residual(nlp) > max_eval || el_time > max_time
    status  = :unknown

    start_time = time()
    too_small  = false
    optimal    = min(normFx, normdual) ≤ ϵ

    @info log_header([:iter, :nf, :primal, :status, :nd, :λ],
    [Int, Int, Float64, String, Float64, Float64],
    hdr_override=Dict(:nf => "#F", :primal => "‖F(x)‖", :nd => "‖d‖"))

    while !(optimal || tired || too_small)

        ###########################
        # (d, stats)  = lsqr(Jx, -Fx, λ = λ, atol = τ)
        d = multi_sol(nlp, x, Fx, Jx, λ, τ)
        ###########################

        too_small = norm(d) < 1e-16
        if too_small #the direction is too small
            status = :too_small
        else
            xp      = x + d
            ###########################
            Fxp     = residual(nlp, xp)# évalue le résidu en xp
            ###########################
            normFxp = norm(Fxp)

            Pred = 0.5 * (normFx^2 - norm(Jx * d + Fx)^2 - λ*norm(d)^2)
            Ared = 0.5 * (normFx^2 - normFxp^2)

            if Ared/Pred < η₁
                λ = max(λ₀, σ₁ * λ)
                status = :increase_λ
            else #success
                x  = xp
                ###########################
                Jx = jac_residual(nlp, x) # réevalue le jacobien en x
                ###########################
                Fx = Fxp
                normFx = normFxp
                status = :success
                if Ared/Pred > η₂
                    λ = max(λ * σ₂, λ₀)
                end
            end
        end

        @info log_row(Any[iter, neval_residual(nlp), normFx, status, norm(d), λ])

        el_time      = time() - start_time
        iter        += 1
        many_evals   = neval_residual(nlp) > max_eval
        iter_limit   = iter > max_iter
        tired        = many_evals || el_time > max_time || iter_limit
        normdual     = norm(Jx' * Fx)
        optimal      = min(normFx, normdual) ≤ ϵ

        η = λ == 0.0 ? min(0.5, 1/iter, normdual) : min(0.5, 1/iter)
        τ = η * normdual
    end

    status = if optimal
        :first_order
    elseif tired
        if neval_residual(nlp) > max_eval
            :max_eval
        elseif el_time > max_time
            :max_time
        elseif iter > max_iter
            :max_iter
        else
            :unknown_tired
        end
    elseif too_small
        :stalled
    else
        :unknown
    end

    return GenericExecutionStats(nlp; status, solution = x,objective = normFx^2 / 2, dual_feas = normdual, iter = iter, elapsed_time = el_time)
end
```

lm_param

We begin by testing our method on the himmelblau problem.

In [ ]:

```julia
stats = lm_param(himmelblau_nls, himmelblau_nls.meta.x0, 1e-6)
@test stats.status == :first_order
```

```
┌ Info:   iter      #F    ‖F(x)‖           status       ‖d‖         λ
└ @ Main /home/julien/Documents/code/MTH8408-Hiv24/lab4_JP/Lab4-notebook.ipynb:36
┌ Info:      0      16   1.2e+02          success   1.0e+01   1.0e-06
└ @ Main /home/julien/Documents/code/MTH8408-Hiv24/lab4_JP/Lab4-notebook.ipynb:77
┌ Info:      1      20   2.9e+01          success   6.0e+00   1.0e-06
└ @ Main /home/julien/Documents/code/MTH8408-Hiv24/lab4_JP/Lab4-notebook.ipynb:77
┌ Info:      2      24   5.3e+00          success   2.6e+00   1.0e-06
└ @ Main /home/julien/Documents/code/MTH8408-Hiv24/lab4_JP/Lab4-notebook.ipynb:77
┌ Info:      3      28   1.4e+00          success   7.4e-01   1.0e-06
└ @ Main /home/julien/Documents/code/MTH8408-Hiv24/lab4_JP/Lab4-notebook.ipynb:77
┌ Info:      4      32   1.7e-01          success   3.4e-01   1.0e-06
└ @ Main /home/julien/Documents/code/MTH8408-Hiv24/lab4_JP/Lab4-notebook.ipynb:77
┌ Info:      5      36   5.2e-02          success   2.5e-02   1.0e-06
└ @ Main /home/julien/Documents/code/MTH8408-Hiv24/lab4_JP/Lab4-notebook.ipynb:77
┌ Info:      6      40   1.7e-04          success   1.4e-02   1.0e-06
└ @ Main /home/julien/Documents/code/MTH8408-Hiv24/lab4_JP/Lab4-notebook.ipynb:77
┌ Info:      7      44   1.6e-09          success   4.0e-05   1.0e-06
└ @ Main /home/julien/Documents/code/MTH8408-Hiv24/lab4_JP/Lab4-notebook.ipynb:77
Test Passed
```

We then test the model on the BNST2 problem from NLSProblems.jl with 400 dimensions.

In [ ]:
```julia
nls_model = BNST2(400)
stats = lm_param(nls_model, nls_model.meta.x0)
@test stats.status == :first_order
```

```
┌ Info:   iter      #F    ‖F(x)‖           status       ‖d‖         λ
└ @ Main /home/julien/Documents/code/MTH8408-Hiv24/lab4_JP/Lab4-notebook.ipynb:36
┌ Info:      0       5   5.6e-16          success   2.0e+00   1.0e-06
└ @ Main /home/julien/Documents/code/MTH8408-Hiv24/lab4_JP/Lab4-notebook.ipynb:77
Test Passed
```

Exercice 3 - Benchmark

On benchmark nos algorithmes sur les problèmes de la librairie NLSProblems.

Voir : https://jso.dev/NLSProblems.jl/stable/benchmark/

In [ ]:
```julia
problems = (eval(problem)() for problem ∈ filter(x -> x != :NLSProblems, names(NLSProblems)))
```

```
Base.Generator{Vector{Symbol}, var"#37#39"}(var"#37#39"(), [:BNST2, :BNST3, :LVcon501, :LVcon502, :LVcon503, :LVcon504, :LVcon511, :LVcon512, :LVcon513, :LVcon514  …  :tp354, :tp355, :tp358, :tp370, :tp371, :tp372, :tp373, :tp379, :tp394, :tp395])
```

In [ ]:
```julia
solvers = Dict(
  :lm => model -> lm_param(model, model.meta.x0),
  :gn => model -> gauss_newton(model, model.meta.x0),
)

stats = bmark_solvers(
  solvers, problems,
  skipif=prob -> (!unconstrained(prob) || get_nvar(prob) > 100 || get_nvar(prob) < 5),
)
```

```
┌ Info:            Name    nvar    ncon           status      Time      f(x)      Dual    Primal
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:127
┌ Info:            NZF1      13       0      first_order   9.1e-05   6.9e-24   1.5e-11   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           mgh17       5       0      first_order   1.1e-03   2.7e-05   1.2e-08   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           mgh18       6       0         max_eval   4.3e-02   5.0e-03   1.1e-03   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           mgh19      11       0         max_eval   3.0e-01   4.4e-02   3.3e-06   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           mgh20       6       0      first_order   1.2e-03   1.1e-03   3.5e-07   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           mgh21      20       0      first_order   3.2e-04   2.6e-20   5.1e-09   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           mgh22      20       0      first_order   1.4e-04   1.0e-09   5.8e-07   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           mgh25      10       0      first_order   5.2e-05   8.0e-16   7.8e-07   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           mgh26      10       0      first_order   7.7e-04   1.4e-05   9.0e-07   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           mgh27      10       0      first_order   4.8e-05   1.4e-15   1.7e-07   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           mgh28      10       0      first_order   5.4e-05   1.5e-16   3.5e-09   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           mgh29      10       0      first_order   1.7e-04   4.1e-14   3.0e-07   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           mgh30      10       0      first_order   8.2e-05   2.6e-14   9.2e-07   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           mgh31      10       0      first_order   1.1e-04   3.1e-14   1.8e-06   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           mgh32      10       0      first_order   1.2e-05   5.0e+00   1.4e-15   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           mgh33      10       0      first_order   1.0e-05   2.3e+00   2.1e-10   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           mgh34      10       0      first_order   8.1e-06   3.1e+00   6.4e-10   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           tp266       5       0      first_order   7.0e-04   5.0e-01   7.9e-07   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           tp267       5       0         max_eval   3.8e-02   1.4e-03   1.8e-04   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           tp271       6       0      first_order   3.0e-05   6.3e-15   1.2e-06   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           tp272       6       0         max_eval   5.7e-02   5.0e-03   1.1e-03   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
```

```
┌ Info:           tp273       6       0      first_order   7.6e-05   1.3e-15   5.8e-07   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           tp282      10       0      first_order   2.0e-03   2.5e-17   4.0e-08   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           tp286      20       0      first_order   3.7e-04   2.6e-20   5.1e-09   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           tp288      20       0         max_eval   2.8e-02   8.8e-06   3.6e-04   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           tp291      10       0      first_order   1.6e-04   5.6e-10   5.0e-07   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           tp292      30       0         max_eval   3.4e-02   5.3e-08   1.2e-05   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           tp293      50       0         max_eval   2.5e-02   8.3e-06   5.2e-04   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           tp294       6       0      first_order   3.2e-04   2.8e-15   3.7e-08   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           tp295      10       0      first_order   5.9e-04   4.0e-17   1.8e-07   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           tp296      16       0      first_order   1.0e-03   7.4e-18   7.7e-08   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           tp297      30       0      first_order   2.1e-03   1.9e-16   3.4e-07   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           tp298      50       0      first_order   4.6e-03   2.4e-16   3.9e-07   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           tp299     100       0      first_order   1.3e-02   1.8e-16   3.3e-07   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           tp303      20       0      first_order   6.0e-05   4.0e-14   7.6e-06   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           tp304      50       0      first_order   1.9e-04   1.5e-22   1.8e-09   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           tp305     100       0      first_order   3.6e-04   1.5e-21   1.6e-08   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           tp370       6       0      first_order   1.4e-03   1.1e-03   3.5e-07   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           tp371       9       0         max_eval   3.9e-01   3.2e-05   1.7e-04   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           tp379      11       0      first_order   4.1e-03   2.0e-02   7.1e-07   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:            Name    nvar    ncon           status      Time      f(x)      Dual    Primal
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:127
```

```
┌ Info:            NZF1      13       0      first_order   4.1e-05   9.2e-19   5.2e-09   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           mgh17       5       0      first_order   2.0e-04   2.7e-05   1.0e-07   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           mgh18       6       0      first_order   1.2e-04   2.8e-03   3.9e-07   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           mgh19      11       0      first_order   9.0e-04   2.0e-02   3.9e-07   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           mgh20       6       0      first_order   1.7e-04   1.1e-03   8.2e-07   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           mgh21      20       0      first_order   7.0e-05   7.1e-27   2.7e-12   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           mgh22      20       0      first_order   8.3e-05   1.2e-13   2.6e-07   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           mgh25      10       0      first_order   2.7e-05   8.0e-16   7.8e-07   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           mgh26      10       0      first_order   2.4e-04   1.4e-05   3.5e-07   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           mgh27      10       0      first_order   2.2e-05   1.1e-13   1.5e-06   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           mgh28      10       0      first_order   1.5e-05   4.8e-16   6.2e-09   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           mgh29      10       0      first_order   2.6e-05   6.9e-14   4.8e-07   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           mgh30      10       0      first_order   2.5e-05   5.6e-19   3.3e-09   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           mgh31      10       0      first_order   3.8e-05   1.2e-16   9.2e-08   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           mgh32      10       0      first_order   1.6e-05   5.0e+00   2.2e-15   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           mgh33      10       0      first_order   1.1e-05   2.3e+00   1.3e-09   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           mgh34      10       0      first_order   9.1e-06   3.1e+00   5.2e-10   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           tp266       5       0      first_order   2.1e-04   5.0e-01   3.2e-07   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           tp267       5       0      first_order   1.1e-04   8.6e-15   2.8e-07   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           tp271       6       0      first_order   8.8e-06   7.9e-30   4.7e-14   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           tp272       6       0      first_order   1.4e-04   2.8e-03   3.9e-07   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           tp273       6       0      first_order   3.3e-05   2.1e-13   7.2e-06   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           tp282      10       0      first_order   1.3e-04   1.2e-13   3.0e-06   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           tp286      20       0      first_order   6.9e-05   3.5e-26   5.9e-12   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           tp288      20       0      first_order   5.1e-05   2.0e-10   2.5e-07   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           tp291      10       0      first_order   4.8e-05   2.1e-10   2.6e-07   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           tp292      30       0      first_order   3.0e-04   5.1e-10   9.3e-07   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           tp293      50       0      first_order   7.1e-04   4.3e-10   9.8e-07   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           tp294       6       0      first_order   1.0e-04   3.2e-18   3.6e-08   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           tp295      10       0      first_order   2.0e-04   1.9e-26   2.8e-12   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           tp296      16       0      first_order   3.8e-04   1.4e-23   7.7e-11   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           tp297      30       0      first_order   1.0e-03   7.1e-22   5.5e-10   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           tp298      50       0      first_order   9.2e-04   1.2e-16   2.3e-07   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           tp299     100       0      first_order   7.9e-03   3.0e-26   3.3e-12   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           tp303      20       0      first_order   3.0e-05   6.2e-14   9.5e-06   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           tp304      50       0      first_order   6.9e-05   7.6e-21   1.3e-08   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           tp305     100       0      first_order   1.5e-04   3.7e-21   2.5e-08   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           tp370       6       0      first_order   1.9e-04   1.1e-03   8.2e-07   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           tp371       9       0      first_order   2.6e-04   7.0e-07   8.1e-07   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
┌ Info:           tp379      11       0      first_order   7.6e-04   2.0e-02   7.7e-07   0.0e+00
└ @ SolverBenchmark /home/julien/.julia/packages/SolverBenchmark/YM13z/src/run_solver.jl:175
Dict{Symbol, DataFrames.DataFrame} with 2 entries:
  :lm => 40×39 DataFrame…
  :gn => 40×39 DataFrame…
```

In [ ]:

```
cols = [:id, :name, :nvar, :objective, :dual_feas, :neval_residual, :neval_jac_residual, :neval_hess, :iter, :elapsed_time, :status]
header = Dict(
  :nvar => "n",
  :objective => "f(x)",
  :dual_feas => "‖∇f(x)‖",
  :neval_residual => "# f",
  :neval_jac_residual => "# ∇f",
  :neval_hess => "# ∇²f",
  :elapsed_time => "t",
)

for solver ∈ keys(solvers)
  pretty_stats(stats[solver][!, cols], hdr_override=header)
end
```

id	name	n	f(x)	‖∇f(x)‖	# f	# ∇f	# ∇²f	iter	t	status
15	NZF1	13	6.93e-24	1.45e-11	41	11	0	10	9.11e-05	first_order
67	mgh17	5	2.73e-05	1.17e-08	149	30	0	37	1.08e-03	first_order
68	mgh18	6	5.03e-03	1.12e-03	10001	1917	0	2500	4.29e-02	max_eval
69	mgh19	11	4.38e-02	3.30e-06	10001	1918	0	2500	2.97e-01	max_eval
70	mgh20	6	1.14e-03	3.50e-07	61	16	0	15	1.16e-03	first_order
71	mgh21	20	2.61e-20	5.11e-09	169	33	0	42	3.24e-04	first_order
72	mgh22	20	1.01e-09	5.76e-07	61	16	0	15	1.44e-04	first_order
75	mgh25	10	7.96e-16	7.84e-07	37	10	0	9	5.20e-05	first_order
76	mgh26	10	1.40e-05	9.04e-07	105	21	0	26	7.69e-04	first_order
77	mgh27	10	1.38e-15	1.66e-07	29	8	0	7	4.82e-05	first_order
78	mgh28	10	1.53e-16	3.49e-09	17	5	0	4	5.41e-05	first_order
79	mgh29	10	4.15e-14	3.04e-07	25	7	0	6	1.71e-04	first_order
80	mgh30	10	2.63e-14	9.15e-07	37	10	0	9	8.20e-05	first_order
81	mgh31	10	3.12e-14	1.78e-06	33	9	0	8	1.08e-04	first_order
82	mgh32	10	5.00e+00	1.37e-15	5	2	0	1	1.19e-05	first_order
83	mgh33	10	2.32e+00	2.08e-10	5	2	0	1	1.00e-05	first_order
84	mgh34	10	3.07e+00	6.43e-10	5	2	0	1	8.11e-06	first_order
120	tp266	5	5.00e-01	7.94e-07	105	21	0	26	6.97e-04	first_order
121	tp267	5	1.38e-03	1.76e-04	10001	1919	0	2500	3.81e-02	max_eval
124	tp271	6	6.34e-15	1.21e-06	33	9	0	8	3.00e-05	first_order
125	tp272	6	5.03e-03	1.12e-03	10001	1917	0	2500	5.66e-02	max_eval
126	tp273	6	1.34e-15	5.77e-07	41	11	0	10	7.58e-05	first_order
127	tp282	10	2.46e-17	3.96e-08	593	115	0	148	2.03e-03	first_order
128	tp286	20	2.61e-20	5.11e-09	169	33	0	42	3.68e-04	first_order
129	tp288	20	8.78e-06	3.63e-04	10001	1917	0	2500	2.82e-02	max_eval
131	tp291	10	5.61e-10	4.99e-07	81	21	0	20	1.57e-04	first_order
132	tp292	30	5.34e-08	1.20e-05	10001	1926	0	2500	3.41e-02	max_eval
133	tp293	50	8.27e-06	5.19e-04	10001	1921	0	2500	2.53e-02	max_eval
134	tp294	6	2.78e-15	3.72e-08	197	39	0	49	3.15e-04	first_order
135	tp295	10	4.01e-17	1.79e-07	249	49	0	62	5.90e-04	first_order
136	tp296	16	7.41e-18	7.70e-08	289	59	0	72	1.01e-03	first_order
137	tp297	30	1.92e-16	3.35e-07	341	72	0	85	2.13e-03	first_order
138	tp298	50	2.42e-16	3.86e-07	437	96	0	109	4.58e-03	first_order
139	tp299	100	1.77e-16	3.29e-07	681	157	0	170	1.31e-02	first_order
140	tp303	20	4.01e-14	7.59e-06	29	8	0	7	6.01e-05	first_order
141	tp304	50	1.48e-22	1.79e-09	41	11	0	10	1.87e-04	first_order
142	tp305	100	1.46e-21	1.57e-08	49	13	0	12	3.61e-04	first_order
169	tp370	6	1.14e-03	3.50e-07	61	16	0	15	1.36e-03	first_order
170	tp371	9	3.19e-05	1.68e-04	10001	1918	0	2500	3.90e-01	max_eval
173	tp379	11	2.01e-02	7.13e-07	105	21	0	26	4.06e-03	first_order

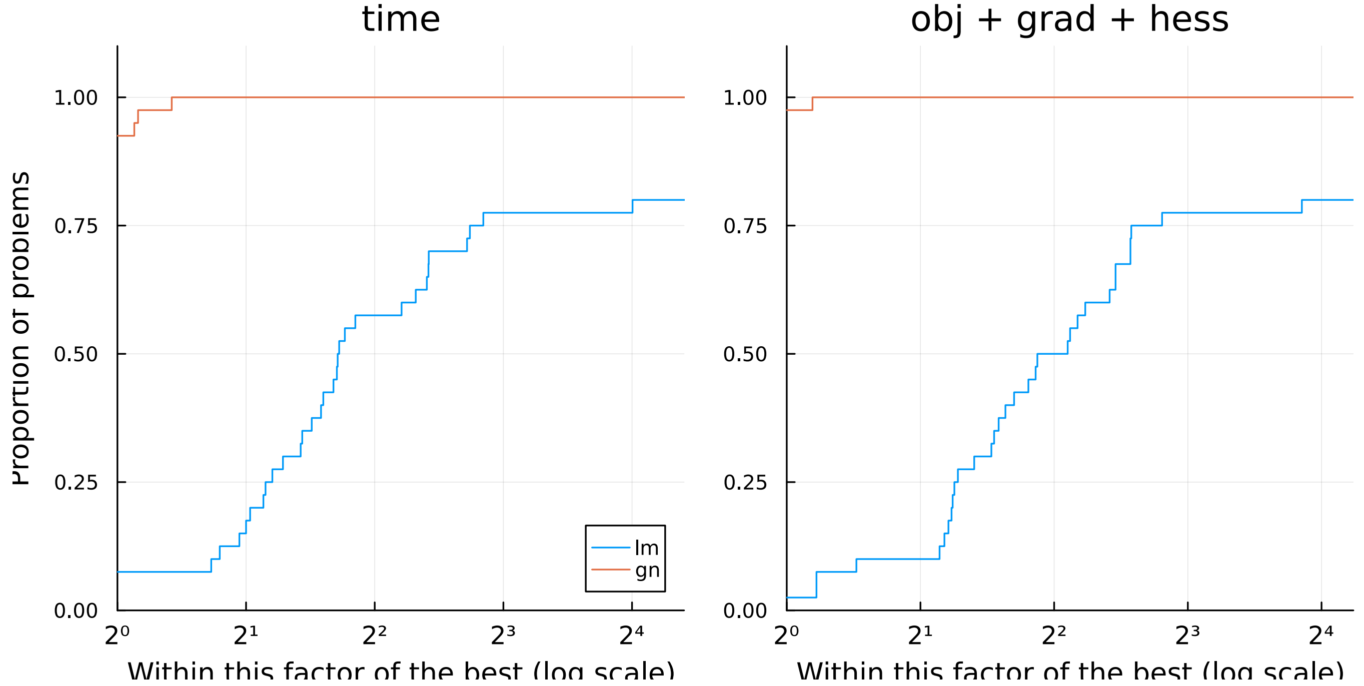
id	name	n	f(x)	‖∇f(x)‖	# f	# ∇f	# ∇²f	iter	t	status
15	NZF1	13	9.22e-19	5.21e-09	9	9	0	8	4.10e-05	first_order
67	mgh17	5	2.73e-05	1.02e-07	17	13	0	16	2.04e-04	first_order
68	mgh18	6	2.83e-03	3.87e-07	16	14	0	15	1.22e-04	first_order
69	mgh19	11	2.01e-02	3.91e-07	20	17	0	19	8.98e-04	first_order
70	mgh20	6	1.14e-03	8.16e-07	7	7	0	6	1.73e-04	first_order
71	mgh21	20	7.14e-27	2.67e-12	19	15	0	18	7.01e-05	first_order
72	mgh22	20	1.20e-13	2.62e-07	19	15	0	18	8.30e-05	first_order
75	mgh25	10	7.96e-16	7.84e-07	10	10	0	9	2.69e-05	first_order
76	mgh26	10	1.40e-05	3.47e-07	27	15	0	26	2.40e-04	first_order
77	mgh27	10	1.06e-13	1.46e-06	7	7	0	6	2.19e-05	first_order
78	mgh28	10	4.84e-16	6.19e-09	3	3	0	2	1.50e-05	first_order
79	mgh29	10	6.94e-14	4.79e-07	3	3	0	2	2.60e-05	first_order
80	mgh30	10	5.64e-19	3.31e-09	5	5	0	4	2.50e-05	first_order
81	mgh31	10	1.20e-16	9.20e-08	6	6	0	5	3.79e-05	first_order
82	mgh32	10	5.00e+00	2.23e-15	4	4	0	3	1.60e-05	first_order
83	mgh33	10	2.32e+00	1.34e-09	3	3	0	2	1.10e-05	first_order
84	mgh34	10	3.07e+00	5.22e-10	3	3	0	2	9.06e-06	first_order
120	tp266	5	5.00e-01	3.18e-07	28	15	0	27	2.11e-04	first_order
121	tp267	5	8.64e-15	2.80e-07	18	14	0	17	1.14e-04	first_order
124	tp271	6	7.89e-30	4.73e-14	3	3	0	2	8.82e-06	first_order
125	tp272	6	2.83e-03	3.87e-07	16	14	0	15	1.38e-04	first_order
126	tp273	6	2.10e-13	7.22e-06	8	8	0	7	3.29e-05	first_order
127	tp282	10	1.19e-13	3.01e-06	26	23	0	25	1.27e-04	first_order
128	tp286	20	3.47e-26	5.90e-12	19	15	0	18	6.89e-05	first_order
129	tp288	20	2.03e-10	2.51e-07	12	12	0	11	5.10e-05	first_order
131	tp291	10	2.07e-10	2.61e-07	21	21	0	20	4.82e-05	first_order
132	tp292	30	5.12e-10	9.33e-07	83	74	0	82	2.99e-04	first_order
133	tp293	50	4.28e-10	9.76e-07	142	131	0	141	7.14e-04	first_order
134	tp294	6	3.20e-18	3.64e-08	32	23	0	31	1.04e-04	first_order
135	tp295	10	1.88e-26	2.82e-12	48	34	0	47	1.97e-04	first_order
136	tp296	16	1.39e-23	7.73e-11	66	46	0	65	3.78e-04	first_order
137	tp297	30	7.10e-22	5.54e-10	111	76	0	110	1.04e-03	first_order
138	tp298	50	1.20e-16	2.29e-07	60	58	0	59	9.17e-04	first_order
139	tp299	100	2.98e-26	3.32e-12	349	235	0	348	7.91e-03	first_order
140	tp303	20	6.22e-14	9.46e-06	8	8	0	7	3.00e-05	first_order
141	tp304	50	7.56e-21	1.27e-08	11	11	0	10	6.91e-05	first_order
142	tp305	100	3.73e-21	2.51e-08	13	13	0	12	1.48e-04	first_order
169	tp370	6	1.14e-03	8.16e-07	7	7	0	6	1.89e-04	first_order
170	tp371	9	7.00e-07	8.14e-07	5	5	0	4	2.62e-04	first_order
173	tp379	11	2.01e-02	7.68e-07	16	13	0	15	7.59e-04	first_order

In [ ]:

```
first_order(df) = df.status .== :first_order
unbounded(df) = df.status .== :unbounded
solved(df) = first_order(df) .| unbounded(df)
costnames = ["time", "obj + grad + hess"]
costs = [
  df -> .!solved(df) .* Inf .+ df.elapsed_time,
  df -> .!solved(df) .* Inf .+ df.neval_residual .+ df.neval_jac_residual,
]

using Plots
gr()

profile_solvers(stats, costs, costnames)
```



Pour faire ce benchmark, nous avons utilisé 40 problèmes. Nous avons fait un profil de performance en fonction du temps d'exécution et du nombre d'évaluation du résidual + le nombre d'évaluation du Jacobien. On remarque que l'algorithme de gauss-newton résous l'ensembles des problèmes alors que l'algorithme de Levenberg-Marquard ne réussis pas à résoudre 8 problèmes parmi les 40. On peut déterminer ce chiffre en regardant le nombre de problèmes ayant le status max_eval. On observe une performance nettement supérieur pour la méthode de Gauss-Newton.

Exercice 4: Rocket Control

Dans les cellules ci-dessous nous introduisons un modèle de contrôle optimal (cf. https://en.wikipedia.org/wiki/Optimal_control) pour le contrôle d'une fusée dont une version discrétisée a été modélisé avec JuMP:

Le lien vers le tutoriel: https://nbviewer.jupyter.org/github/jump-dev/JuMPTutorials.jl/blob/master/notebook/modelling/rocket_control.ipynb

In [ ]:
```julia
using JuMP, Ipopt

# Create JuMP model, using Ipopt as the solver
rocket = Model(optimizer_with_attributes(Ipopt.Optimizer, "print_level" => 0))

# Constants
# Note that all parameters in the model have been normalized
# to be dimensionless. See the COPS3 paper for more info.
h_0 = 1    # Initial height
v_0 = 0    # Initial velocity
m_0 = 1    # Initial mass
g_0 = 1    # Gravity at the surface

T_c = 3.5  # Used for thrust
h_c = 500  # Used for drag
v_c = 620  # Used for drag
m_c = 0.6  # Fraction of initial mass left at end

c     = 0.5 * sqrt(g_0 * h_0)  # Thrust-to-fuel mass
m_f   = m_c * m_0            # Final mass
D_c   = 0.5 * v_c * m_0 / g_0    # Drag scaling
T_max = T_c * g_0 * m_0        # Maximum thrust

n = 800   # Time steps

@variables(rocket, begin
    Δt ≥ 0, (start = 1/n) # Time step
    # State variables
    v[1:n] ≥ 0            # Velocity
    h[1:n] ≥ h_0          # Height
    m_f ≤ m[1:n] ≤ m_0    # Mass
    # Control
    0 ≤ T[1:n] ≤ T_max    # Thrust
end)

# Objective: maximize altitude at end of time of flight
@objective(rocket, Max, h[n])

# Initial conditions
@constraints(rocket, begin
    v[1] == v_0
    h[1] == h_0
    m[1] == m_0
    m[n] == m_f
end)

# Forces
# Drag(h,v) = Dc v^2 exp( -hc * (h - h0) / h0 )
@NLexpression(rocket, drag[j = 1:n], D_c * (v[j]^2) * exp(-h_c * (h[j] - h_0) / h_0))
# Grav(h)   = go * (h0 / h)^2
@NLexpression(rocket, grav[j = 1:n], g_0 * (h_0 / h[j])^2)
# Time of flight
@NLexpression(rocket, t_f, Δt * n)

# Dynamics
for j in 2:n
    # h' = v

    # Rectangular integration
    # @NLconstraint(rocket, h[j] == h[j - 1] + Δt * v[j - 1])

    # Trapezoidal integration
    @NLconstraint(rocket,
        h[j] == h[j - 1] + 0.5 * Δt * (v[j] + v[j - 1]))

    # v' = (T-D(h,v))/m - g(h)

    # Rectangular integration
    # @NLconstraint(rocket, v[j] == v[j - 1] + Δt *(
    #                 (T[j - 1] - drag[j - 1]) / m[j - 1] - grav[j - 1]))

    # Trapezoidal integration
```

```julia
    @NLconstraint(rocket,
        v[j] == v[j-1] + 0.5 * Δt * (
            (T[j] - drag[j] - m[j] * grav[j]) / m[j] +
            (T[j - 1] - drag[j - 1] - m[j - 1] * grav[j - 1]) / m[j - 1]))

    # m' = -T/c

    # Rectangular integration
    # @NLconstraint(rocket, m[j] == m[j - 1] - Δt * T[j - 1] / c)

    # Trapezoidal integration
    @NLconstraint(rocket,
        m[j] == m[j - 1] - 0.5 * Δt * (T[j] + T[j-1]) / c)
end
```

In [ ]:
```julia
# Solve for the control and state
println("Solving...")
status = optimize!(rocket)

# Display results
# println("Solver status: ", status)
println("Max height: ", objective_value(rocket))
```

```
Solving...
Max height: 1.0128340648308058
```

In [ ]:
```julia
value.(h)[n]
```

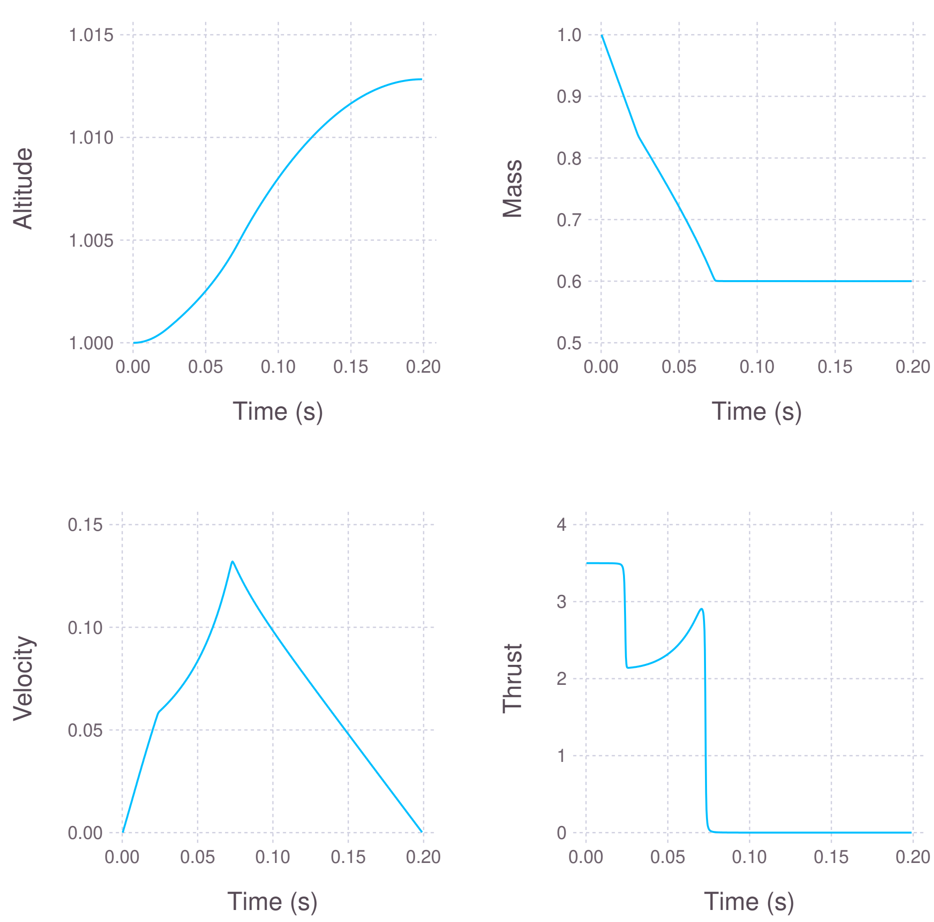
```
1.0128340648308058
```

In [ ]:
```julia
# Can visualize the state and control variables
using Gadfly
```

In [ ]:
```julia
h_jump = value.(h)[:]
m_jump = value.(m)[:]
v_jump = value.(v)[:]
T_jump = value.(T)[:]
delta_jump = value.(Δt)




h_plot = Gadfly.plot(x = (1:n) * value.(Δt), y = h_jump, Geom.line,
                Guide.xlabel("Time (s)"), Guide.ylabel("Altitude"))
m_plot = Gadfly.plot(x = (1:n) * value.(Δt), y = m_jump, Geom.line,
                Guide.xlabel("Time (s)"), Guide.ylabel("Mass"))
v_plot = Gadfly.plot(x = (1:n) * value.(Δt), y = v_jump, Geom.line,
                Guide.xlabel("Time (s)"), Guide.ylabel("Velocity"))
T_plot = Gadfly.plot(x = (1:n) * value.(Δt), y = T_jump, Geom.line,
                Guide.xlabel("Time (s)"), Guide.ylabel("Thrust"))
draw(SVG(6inch, 6inch), vstack(hstack(h_plot, m_plot), hstack(v_plot, T_plot)))
```



Questions:

```
- i) Transformer le modèle JuMP utilisé ci-dessus en un NLPModel en utilisant le package `NLPModelsJuMP`.
- ii) Résoudre ce nouveau modèle avec `Ipopt` en utilisant `NLPModelsIpopt`.
- iii) Calcul séparément la différence entre les h,v,m,T, Δt calculés.
- iv) Est-ce que le contrôle T atteint ses bornes ?
- v) Reproduire les graphiques ci-dessous avec la solution calculée via `NLPModelsIpopt`.
```

In [ ]:
```julia
using NLPModels, LinearAlgebra, NLPModelsJuMP, NLPModelsIpopt
```

Ici, nous transformons le problème Jump en problème NLPModel et le résolvons avec ipopt.

In [ ]:
```julia
nlp = MathOptNLPModel(rocket)
stats = ipopt(nlp)
print(stats)
```

```
This is Ipopt version 3.14.14, running with linear solver MUMPS 5.6.2.

Number of nonzeros in equality constraint Jacobian...:    15185
Number of nonzeros in inequality constraint Jacobian.:        0
Number of nonzeros in Lagrangian Hessian.............:    45543

Total number of variables............................:     3201
                     variables with only lower bounds:     1601
                variables with lower and upper bounds:     1600
                     variables with only upper bounds:        0
Total number of equality constraints.................:     2401
Total number of inequality constraints...............:        0
        inequality constraints with only lower bounds:        0
   inequality constraints with lower and upper bounds:        0
        inequality constraints with only upper bounds:        0

iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du alpha_pr  ls
   0  1.0100000e+00 3.96e-01 2.13e+00  -1.0 0.00e+00    -  0.00e+00 0.00e+00   0
   1  1.2110479e+00 7.40e-03 6.00e+03  -1.0 4.97e-01    -  1.32e-02 9.84e-01f  1
   2  1.2048591e+00 5.86e-03 1.11e+04  -1.0 3.15e+00    -  1.44e-01 1.57e-01f  1
   3  1.2629237e+00 5.19e-03 1.46e+04  -1.0 1.82e+00    -  7.26e-02 1.13e-01f  1
   4  1.4170550e+00 5.07e-03 3.15e+03  -1.0 1.67e+01   0.0 1.17e-02 2.18e-02f  1
   5  1.1124928e+00 2.20e-03 5.06e+05  -1.0 5.28e-01   1.3 2.12e-01 5.77e-01h  1
   6  1.1282562e+00 1.79e-03 1.44e+06  -1.0 1.83e+01    -  1.25e-02 1.82e-01f  1
   7  1.0529956e+00 3.50e-04 3.37e+05  -1.0 4.64e-01   0.9 9.47e-01 7.93e-01h  1
   8  1.0386949e+00 4.62e-04 2.32e+05  -1.0 9.23e+00    -  4.08e-02 3.63e-01f  1
   9  1.0298777e+00 3.71e-04 1.76e+05  -1.0 7.48e+00    -  2.05e-01 3.07e-01h  1
iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du alpha_pr  ls
  10  1.0232606e+00 2.51e-04 1.34e+05  -1.0 4.79e+00    -  3.50e-01 3.18e-01h  1
  11  1.0175352e+00 1.63e-04 9.98e+04  -1.0 5.79e+00    -  4.28e-01 3.88e-01h  1
  12  1.0143410e+00 1.05e-04 1.03e+05  -1.0 4.65e+00    -  1.00e+00 3.36e-01h  1
  13  1.0080148e+00 2.85e-05 5.04e+04  -1.0 1.45e+00    -  1.00e+00 9.90e-01h  1
  14  1.0078402e+00 4.17e-06 2.33e+03  -1.0 4.80e-01    -  1.00e+00 1.00e+00h  1
  15  1.0078153e+00 1.55e-08 1.82e+01  -1.0 3.60e-02    -  1.00e+00 1.00e+00f  1
  16  1.0078153e+00 4.53e-13 1.31e+01  -2.5 8.59e-05    -  1.00e+00 1.00e+00h  1
  17  1.0078190e+00 3.01e-10 1.01e-03  -2.5 2.80e-03    -  1.00e+00 1.00e+00h  1
  18  1.0078229e+00 3.09e-10 3.15e+02  -5.7 2.98e-03    -  9.99e-01 1.00e+00h  1
  19  1.0094243e+00 8.64e-05 3.87e+00  -5.7 2.27e+00    -  9.88e-01 9.72e-01f  1
iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du alpha_pr  ls
  20  1.0111368e+00 6.10e-05 2.94e+00  -5.7 2.05e+00    -  1.00e+00 9.00e-01f  1
  21  1.0111174e+00 4.64e-07 2.00e-02  -5.7 7.19e-01    -  1.00e+00 1.00e+00f  1
  22  1.0111207e+00 3.30e-09 6.81e-06  -5.7 6.70e-02    -  1.00e+00 1.00e+00h  1
  23  1.0122700e+00 2.31e-05 3.00e+01  -8.6 8.96e-01    -  7.15e-01 8.77e-01f  1
  24  1.0127033e+00 1.51e-05 8.16e+00  -8.6 8.55e-01    -  7.50e-01 7.80e-01h  1
  25  1.0128033e+00 1.01e-05 3.13e+00  -8.6 1.26e+00    -  6.60e-01 7.31e-01h  1
  26  1.0128269e+00 5.34e-06 1.23e+00  -8.6 1.39e+00    -  6.46e-01 7.34e-01h  1
  27  1.0128326e+00 2.56e-06 3.91e-01  -8.6 1.36e+00    -  7.03e-01 7.73e-01h  1
  28  1.0128339e+00 1.11e-06 5.08e-03  -8.6 1.18e+00    -  9.71e-01 8.96e-01f  1
  29  1.0128341e+00 2.45e-07 4.92e-05  -8.6 9.30e-01    -  1.00e+00 1.00e+00f  1
iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du alpha_pr  ls
  30  1.0128341e+00 3.51e-09 1.19e-06  -8.6 3.15e-01    -  1.00e+00 1.00e+00h  1
  31  1.0128341e+00 8.93e-11 5.13e-09  -8.6 3.70e-02    -  1.00e+00 1.00e+00h  1

Number of Iterations....: 31

                                   (scaled)                 (unscaled)
Objective...............:  -1.0128340648308058e+00    1.0128340648308058e+00
Dual infeasibility......:   5.1309253582343877e-09    5.1309253582343877e-09
Constraint violation....:   8.9276780412816947e-11    8.9276780412816947e-11
Variable bound violation:   0.0000000000000000e+00    0.0000000000000000e+00
Complementarity.........:   2.5098720261156027e-09    2.5098720261156027e-09
Overall NLP error.......:   5.1309253582343877e-09    5.1309253582343877e-09


Number of objective function evaluations             = 32
Number of objective gradient evaluations             = 32
Number of equality constraint evaluations            = 32
Number of inequality constraint evaluations          = 0
Number of equality constraint Jacobian evaluations   = 32
Number of inequality constraint Jacobian evaluations = 0
Number of Lagrangian Hessian evaluations             = 31
Total seconds in IPOPT                               = 0.245

EXIT: Optimal Solution Found.
Generic Execution stats
  status: first-order stationary
  objective value: 1.0128340648308058
  primal feasibility: 8.927678041281695e-11
  dual feasibility: 5.130925358234388e-9
  solution: [0.0002487563718099509  2.9314730041006646e-41  0.0006225585100650303  0.001246602795464594 ⋯ 0.002331886219071902]
  multipliers: [-0.4248952799888991  -4.868945544805185  0.1883670260188766  -0.24615504933665058 ⋯ -0.004435298993160076]
  multipliers_L: [-1.0073321170573435e-5  -0.25059035596800616  -4.025104898765545e-6  -2.0101699331154446e-6 ⋯ -1.0746205337886829e-6]
  multipliers_U: [0.0  0.0  0.0  0.0 ⋯ -7.164497754177925e-10]
  iterations: 31
  elapsed time: 0.245
  solver specific:
    real_time: 0.24546313285827637
    internal_msg: :Solve_Succeeded
```

In [ ]:

```julia
test = stats.solution
print(length(test))
```

```
3201
```

Nous extractons les variables optimales à partir des statistiques de résolution.

In [ ]:

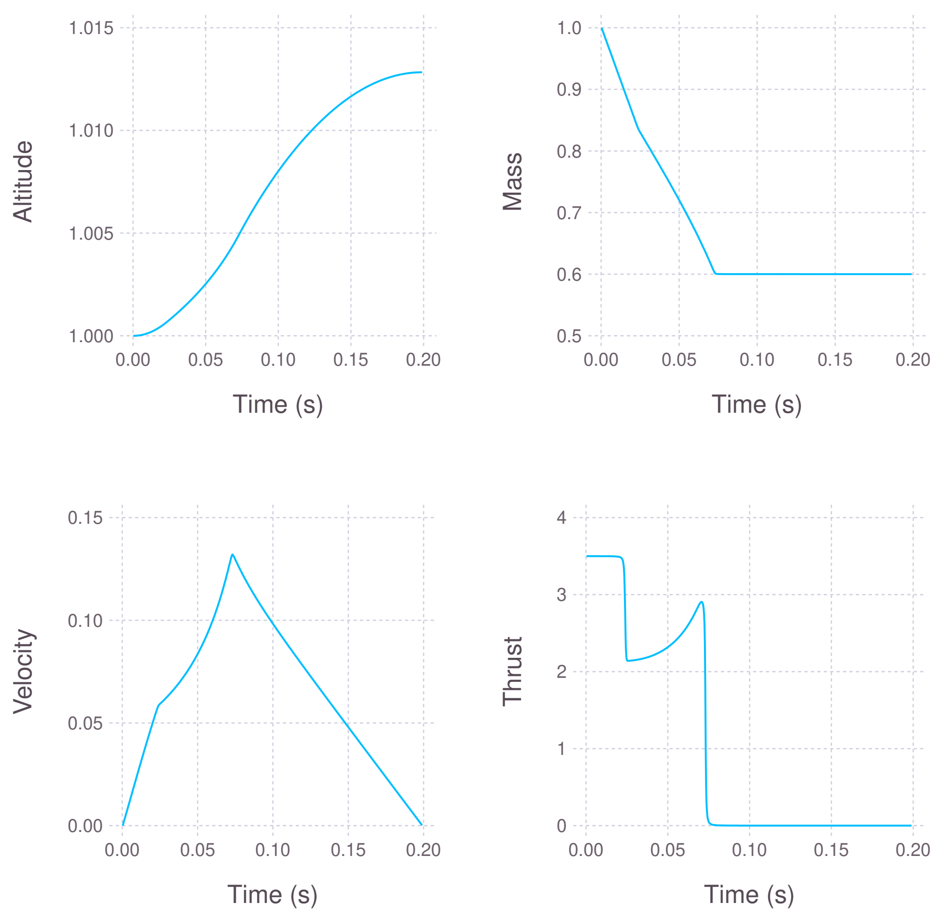
```julia
delta = stats.solution[1]
v = stats.solution[2:801]
h =  stats.solution[802:1601]
m = stats.solution[1602:2401]
T = stats.solution[2402:3201]

println(length(T))
```

```
800
```

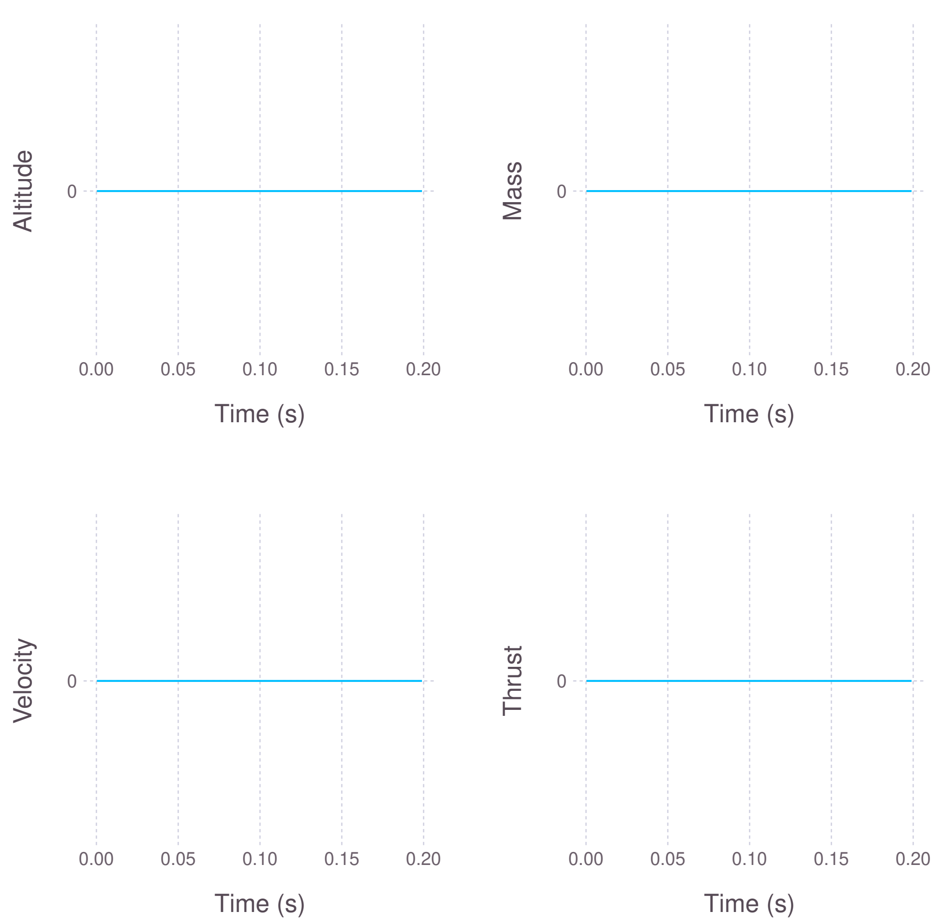
Nous reproduisons les graphiques comme demandé.

```
In [ ]: h_plot = Gadfly.plot(x = (1:n) * delta, y = h, Geom.line, Guide.xlabel("Time (s)"), Guide.ylabel("Altitude"))
        m_plot = Gadfly.plot(x = (1:n) * delta, y = m, Geom.line,Guide.xlabel("Time (s)"), Guide.ylabel("Mass"))
        v_plot = Gadfly.plot(x = (1:n) * delta, y = v, Geom.line, Guide.xlabel("Time (s)"), Guide.ylabel("Velocity"))
        T_plot = Gadfly.plot(x = (1:n) * delta, y = T, Geom.line, Guide.xlabel("Time (s)"), Guide.ylabel("Thrust"))
        draw(SVG(6inch, 6inch), vstack(hstack(h_plot, m_plot), hstack(v_plot, T_plot)))
```



Ici, on observe la différence entre les deux méthodes de résolution, on remarque qu'il n'y a vraiment aucune différence entre les deux.

```
In [ ]: # Differences
        h_plot = Gadfly.plot(x = (1:n) * delta, y = h - h_jump, Geom.line, Guide.xlabel("Time (s)"), Guide.ylabel("Altitude"))
        m_plot = Gadfly.plot(x = (1:n) * delta, y = m - m_jump, Geom.line,Guide.xlabel("Time (s)"), Guide.ylabel("Mass"))
        v_plot = Gadfly.plot(x = (1:n) * delta, y = v - v_jump, Geom.line, Guide.xlabel("Time (s)"), Guide.ylabel("Velocity"))
        T_plot = Gadfly.plot(x = (1:n) * delta, y = T - T_jump, Geom.line, Guide.xlabel("Time (s)"), Guide.ylabel("Thrust"))
        draw(SVG(6inch, 6inch), vstack(hstack(h_plot, m_plot), hstack(v_plot, T_plot)))
```



La norme des différences donne exactement la même réponse.

```
In [ ]: println(norm(h - h_jump))
        println(norm(v - v_jump))
        println(norm(m - m_jump))
        println(norm(T - T_jump))
        println(norm(delta - delta_jump))
```

```
0.0
0.0
0.0
0.0
0.0
```

Finalement, on remarque que T atteint sa valeur maximale de 3.5 en début d'ascension puis sa valeur minimale, zéro, vers 0.07 secondes.