

PEPit.jl tutorial: Accelerated proximal point for monotone inclusions

John Doe

December 11, 2025

Before we do anything let us load the necessary packages!

```
using PEPit, OrderedCollections, Mosek, MosekTools, OffsetArrays
```

Let us start with defining an empty PEP, which we will construct step by step.

```
problem = PEP()
```

Consider the monotone inclusion problem

$$\text{Find } x : 0 \in Ax,$$

where A is maximally monotone. We denote $J_A = (I + A)^{-1}$ the resolvent of A . Let us define this operator class in PEPit next.

```
A = declare_function!(problem, MonotoneOperator, OrderedDict())
```

Algorithm: The algorithm that we want to investigate is the accelerated proximal point is described as follows, for $t \in \{0, \dots, n-1\}$

$$\begin{aligned} x_{t+1} &= J_{\alpha A}(y_t), \\ y_{t+1} &= x_{t+1} + \frac{t}{t+2}(x_{t+1} - x_t) - \frac{t}{t+2}(x_t - y_{t-1}), \end{aligned}$$

where $x_0 = y_0 = y_{-1}$, where α is a positive scalar.

Our goal is to compute the smallest possible $\tau(n, \alpha)$ such that the guarantee

$$\|x_n - y_n\|^2 \leq \tau(n, \alpha) \|x_0 - x_\star\|^2,$$

is valid, where $Ax_\star \ni 0$ and x_0 is the initial point that we pick.

For our tutorial, let us take $\alpha = 2.0$ and $n = 10$.

```
alpha = 2

n = 10
```

We next define the optimal point $x_\star \equiv \text{xs}$ that satisfies $0 \in Ax_\star$. and the initial point $x_0 \equiv \text{x0}$,

```
xs = stationary_point!(A)

x0 = set_initial_point!(problem)
```

The next stage is defining the initial condition $\|x_0 - x_\star\|^2$.

```
set_initial_condition!(problem, (x0 - xs)^2 <= 1)
```

Now it is time to implement our algorithm in a loop.

```
x = OffsetVector(fill(x0, n + 1), 0:n)
y = OffsetVector(fill(x0, n + 1), 0:n)

for i in 0:(n - 2)
    x[i + 1], _, _ = proximal_step!(y[i + 1], A, alpha)
    y[i + 2] = x[i + 1] + i / (i + 2) * (x[i + 1] - x[i]) - i / (i + 2) * (x[i] - y[i])
end
x[n], _, _ = proximal_step!(y[n], A, alpha)
```

Not it is time to define our performance metric, which is $\|x_n - y_n\|^2$.

```
set_performance_metric!(problem, (x[n] - y[n])^2)
```

We have everything to build up the PEP. Now time to solve!

```
τ_PEPit = solve!(problem, solver=Mosek.Optimizer, verbose=true)
```

If everything is well, we should get $\tau_{\text{PEPit}} = 0.01$ for $n = 10$ and $\alpha = 2$.

What does the theory say? A tight theoretical worst-case guarantee can be found in [1, Theorem 4.1], for $n \geq 1$,

$$\|x_n - y_{n-1}\|^2 \leq \frac{1}{n^2} \|x_0 - x_\star\|^2.$$

```
τ_theory = 1 / (n^2)

@info "PEPit guarantee: τ_PEPit = $(round(τ_PEPit, digits=6))"

@info "Theoretical guarantee: τ_theory = $(round(τ_theory, digits=6))"
```

We should see:

```
PEPit guarantee:  $\tau_{\text{PEPit}} = 0.01$ 
```

```
Theoretical guarantee:  $\tau_{\text{theory}} = 0.01$ 
```

So the guarantee that we found via PEPit is the same as theoretical guarantee!

Reference:

[1] D. Kim (2021). Accelerated proximal point method for maximally monotone operators. Mathematical Programming, 1-31. <<https://arxiv.org/pdf/1905.05149v4.pdf>>