A simple introduction to PANOC

Willem Melis

December 18, 2017

Abstract

This is a simple introduction to solving an NMPC problem with the PANOC algorithm. First the proximal operator is introduced with the proximity gradient decent algorithm. Next the link is made between these algorithms and the PANOC algorithm.

Contents

1 MPC

1.1 System

A physical system can be represented as $\dot{x} = F_c(x, u)$, where x is the current state and u is the current input. By using a discrete integrator the system can be written as $x^{k+1} = F_d(x^k, u^k)$. F_d should be expressible as an series of elementary operations.

1.2 Problem definition

1.2.1 Problem form

The goal is to define the problem as equation 1, and then solve it for u given x_0, the current state of the system. Sometimes x_0 will be assumed to be part of the function f and the form of equation 2 will be used.

$$\underset{u}{\text{minimize}}\ f(x_0, u) + g(u) \tag{1}$$

$$\underset{u}{\text{minimize}}\ f(u) + g(u) \tag{2}$$

1.2.2 Single shoot

If the horizon is N then the problem is solved for the inputs $u = [u_0, u_1, ...u_{N-1}]$ where each u_k is a vector of all the inputs of the system. This means that the vector u is of size (Horizon-1)*dimension_input .

Define the cost for each step in the horizon as 3.

$$\begin{aligned} l_k(x_0, u) = \quad & x_k^T Q x_k + u_k^T R u_k \\ \text{subject to} \quad & x_0 = \bar{x} \\ & x_{n+1} = F_d(x_n, u_n), n = 0...N-1 \end{aligned} \tag{3}$$

Define the terminal cost as equation 4.

$$\begin{aligned} l_N(x_0, u) = \quad & x_N^T S x_N \\ \text{subject to} \quad & x_0 = \bar{x} \\ & x_{n+1} = F_d(x_n, u_n), n = 0...N-1 \end{aligned} \tag{4}$$

$f(x_0, u)$ can then be defined as in equation 5 the sum of the cost of all iterations plus the terminal cost.

$$f(x_0, u) = \sum_{k=1}^{N-1} l_k(x_0, u) + l_N \tag{5}$$

As a side note, an other term can be added to equation 5 to represent the obstacle avoidance.

1.3 Constraints

An other important aspect of a MPC problem are the constraints. In practice inputs have to comply with the physical properties of the devices. Absurdly high or low input values might in theory lead to a fast solution, but are not feasible in practice.

An major advantage of the PANOC algorithm is that it can take non linear or non convex constraints. As longs as the proximal operation is analytically defined on the constraint.

A simple example is the indicator box function, which allows to set a maximum and minimum value on the inputs. (The indicator box function is defined in the appendix) This means that every feasible solution lies within the bounds of the user defined box.

2 Proximal gradient method

2.1 Proximal mapping

The proximal operator is defined as $\text{prox}_g(x) = \arg\min_u (g(u) + \frac{1}{2}||u - x||_2^2)$.

- if $h(x) = 0$ then $\text{prox}_h(x) = x$
- if $h(x) = I_c(x)$ where I_c is define in Equation 6, the proximity operator on an indicator function is the orthogonal projection on the set.

The indicator function: is defined in equation 6.

$$I_c = \begin{cases} 0 & x \in C \\ \infty & x \notin C \end{cases} \tag{6}$$

The proximal mapping can be seen as a more generalized form of projection. The Appendix contains an other important example, the indicator box function. Which is an simple way to put constraints on the input.

2.2 Gradient projected method

$$\begin{aligned} &\underset{x}{\text{argmin}} && f_0(x) \\ &\text{subject to} && g(x) = 0 \end{aligned} \tag{7}$$

The classical gradient decent method cannot be used to solve the problem of equation 7. As this problem has a condition that must be met by the algorithms solution. The gradient decent iterations are defined in equation 8 . If in each iteration the solution is projected on the space spanned by $g(x) = 0$ the iterations become equation 9. This algorithm is called the gradient projection method.

$$x^k = x^{k-1} - \nabla f(x^{k-1}) \tag{8}$$

$$x^k = \text{project}_{g(x)=0}[x^{k-1} - \nabla f(x^{k-1})] \tag{9}$$

If $g(x)$ is an indicator function then equation 9 can be written as

$$x^k = \text{prox}_g[x^{k-1} - \nabla f(x^{k-1})] \tag{10}$$

2.3 The proximal gradient method

Equation 11 can be solved with the proximal gradient method (sometimes called forward backward splitting - FBS) . Where the proximal operator on $g(x)$ is analytically defined.

$$\arg\min_x = f(x) + g(x) \tag{11}$$

Inspired by the projected gradient method the proximal gradient method is define in equation 12. The γ variable is the step size, in order to have a convergence of O(1/k) $\gamma \in (0, 1/L)$. If if $\gamma \in (1/L, 2/L)$ the algorithm will still converge but then its no longer a majorization-minimization method. (more on this see [4])

$$x^k = \text{prox}_g\left(x^{(k-1)} - \gamma \nabla f(x^{(k-1)})\right) \tag{12}$$

2.4 Proximal minimization algorithm

[5] contains a shot proof that illustrates the proximal operator itself as a fixed point minimization algorithm. Or simply put take $f(x) = 0$. A property of the conjugated function is used to derive the gradient, see appendix for the theorem.

Proof. the iteration $x^{k+1} = \mathrm{prox}_g(x^k)$ will minimize the smoothed version of function f(x)

$$\begin{aligned}
f_\mu &= \inf_y \Big\{ |y| + \frac{1}{2\mu}(x-y)^2 \Big\} \\
&= \frac{1}{2\mu}||x||^2 + \frac{1}{2\mu}\inf_y \Big\{ 2\mu f(y) - 2x^T y + ||y||^2 \Big\} \\
&= \frac{1}{2\mu}||x||^2 + \frac{1}{\mu}\sup_y \Big\{ x^T y - \mu f(y) - \frac{1}{2}||y||^2 \Big\} \\
&= \frac{1}{2\mu}||x||^2 \frac{1}{\mu}\Big(\mu f + \frac{1}{2}||\cdot||^2 \Big)^*(x) \\
\nabla f_\mu &= \frac{x}{\mu} - \frac{1}{\mu}\operatorname*{argmax}_y \Big\{ x^T y - \mu f(y) - \frac{1}{2}||y||^2 \Big\} \\
&= \frac{1}{\mu}(x - \mathrm{prox}_{\mu f}(x)) \\
\mathrm{prox}_{\mu f}(x) &= x - \mu \nabla f_\mu(x)
\end{aligned}$$

□

This means thats the iteration $x^{k+1} = \mathrm{prox}_g(x^k)$ will minimize the smoothed version of function g(x).

3 Proximal gradient method with line search

3.1 Starting value gamma

3.1.1 Estimating Lipschitz value

The Lipschitz of $\nabla f(x)$ value is a non negative number that complies with equation 13.

$$L = \sup_{x \neq y} \frac{|\nabla f(y) - \nabla f(x)|}{|y - x|} \tag{13}$$

In practice it is always possible to find the actual Lipschitz value. So the Lipschitz value will be estimated locally at the starting point x_0 of the algorithm as defined in equation 14. With $\delta = max[\delta_l, 10^{-6} \cdot x_0]$ where δ_l is a small number chosen by the controller designer.

$$L = \sup \frac{|\nabla f(x + \delta) - \nabla f(x)|}{|\delta|} \tag{14}$$

The Lipschitz value is not explicitly saved but is used to estimate γ. γ is then used in the backtracking of the proximal gradient descent and saved in between iterations.

The Lipschitz value can be derived from γ if needed. This also means that as the algorithm progresses and γ improves, so does the estimation of the Lipschitz value.

3.1.2 Estimating gamma

From [3] we know that $\gamma < \frac{1}{L}$ in order to have convergence to a local minimum. As gamma needs to be smaller than $\frac{1}{L}$ an safety value is introduced. This idea was copied over from the kul-forbes/ForBES library by Lorenzo Stella and Panos Patrinos, which uses a β of 0.05. And leads to equation 15.

$$\gamma = \frac{1-\beta}{L} \tag{15}$$

3.2 Backtracking gamma

3.2.1 Backtracking algorithm

Line-search is based on "Armijo's sufficient decrease condition" written down as equation 16. The safety parameter θ multiplied with the step size t_k is $\gamma = \theta \cdot t_k$.

$$f(x_k + t_k p_k) \leq f(x_k) + \theta t_k \nabla f(x_k)^T p_k \tag{16}$$

Line-search more specific backtracking will reduce the value of γ until the condition of equation 16 fails.

3.2.2 Backtracking in proximal gradient descent used in FBS

The "Armijo's sufficient decrease condition" is adjusted to equation 17 Which is an quadratic bound, $-p_k = x - \bar{x}$ and an additional term $\frac{1-\beta}{2\gamma}||x - \bar{x}||^2$ is added.

$$f(\bar{x}) \leq f(x) - \nabla f(x)^T [x - \bar{x}] + \frac{1}{2\gamma}||x - \bar{x}||^2 \tag{17}$$

This all leads to algorithm 1 the proximal gradient method. Equation 17 can be seen as a quadratic model, as the Lipschitz value of the gradient is equal to $L = \frac{1}{\gamma}$.

3.3 Final algorthm

The final algorithm 2 delivers the upward direction. $x_{k+1} = x_k - direction$.

Algorithm 1 backtracking γ

```
1: procedure LINEASEARCH_GAMMA(x,$\gamma$)
2:     $\bar{x} = \text{prox}_g\big(x - \gamma\nabla f(x)\big)$
3:     while $f(\bar{x}) > f(x) - \nabla f(x)^T[x - \bar{x}] + \frac{1}{2\gamma}||x - \bar{x}||^2$ do
4:         $\gamma = \frac{\gamma}{2}$
5:         $\bar{x} = \text{prox}_g\big(x - \gamma\nabla f(x)\big)$
6:     end while
7:     return $\gamma$
8: end procedure
```

Algorithm 2 proximal gradient PANOC with backtracking

```
1: procedure GET_PROXIMAL_GRADIENT_STEP(x,$\gamma$)
2:     $\gamma$=LINESEARCH_GAMMA($\gamma$)
3:     $\bar{x} = \text{prox}_g\big(x - \gamma\nabla f(x)\big)$
4:     return direction=[$x - \bar{x}$], $\gamma$
5: end procedure
```

4 PANOC algorithm

This section is based on [3] and [1], the big difference is that this text is focused implementation. And so the formula's often look slightly different. The FBE part is based on [2].

4.1 Introduction

The PANOC algorithm is an accelerated version of the proximal gradient descent. The direction $x - \bar{x}$ calculated with the proximal gradient descent is combined with a second direction, who will hopefully accelerate the convergence of the algorithm.

$$x_{k+1} = x_k - (1 - \tau_k) \cdot (x - \bar{x}) + \tau_k d_k \tag{18}$$

The new term $\tau_k d_k$ should accelerate the convergence if $\tau_k \neq 0$. The step d_k is calculated using a quasi-newton algorithm. As a quasi-newton algorithm uses curvature information of the cost function, it uses information not available to a gradient descent based method.

Furthermore it has a super linear convergence rate. Which is much faster then the sub-linear convergence of a typical gradient descent algorithm when it gets close to the solution. On top of that, the quasi-newton method will not require extra function evaluations. Which are the major contributors to the costs of the algorithm.

4.2 Quasi newton method

4.2.1 Problem definition

The iteration of equation 12 can indirectly be used to solve the optimization problem. By using the residue defined in equation 19 a fixed point can be found.

$$R_\gamma(x) = \frac{1}{\gamma}\left[x - \text{prox}_g(x - \nabla f(x)\gamma)\right] \tag{19}$$

The solution of equation 19 can be found trough the Newton iteration of equation 20. Where H_k satisfies the inverse secant condition of equation 21. As the implementation is aimed at embedded software a good option to solve this would be LBFGS.

$$x^{k+1} = x^k - H_k R_\gamma(x^k) \tag{20}$$

$$x^{k+1} - x^k = H_{K+1}\Big(R_\gamma(x^{k+1}) - R_\gamma(x^k)\Big) \tag{21}$$

4.2.2 LBFGS

The variation of LBFGS used to solve equation 22 is algorithm 3, the first iteration is not displayed.(the direction is equal to the gradient) And the current buffersize will increase every iteration from 1 to the limit specified by the controller designer.

$$R(x) = \frac{1}{\gamma}\left[x - \text{prox}_g(x - \nabla f(x)\gamma)\right] = 0 \tag{22}$$

Algorithm 3 LBFGS

```
1: procedure LBFGS($x^k$,M=current_buffersize)
2:     $q = R(x^k)$
3:     for i=M:1 do
4:         $\alpha(i) = \rho(i) \cdot s(:,i)^T q$
5:         $q = q - \alpha(i) \cdot y(:,i)$
6:     end for
7:     $H_k^0 = y(:,M) \cdot s(:,M)^T \cdot \frac{1}{y(:,M)^T \cdot y(:,M)}$
8:     $H_k^0 \cdot R(x^k)$
9:     for i=1:M do
10:        $\beta(i) = \rho(i) \cdot y(:,i)^T \cdot z$
11:        $z = z + s(:,i)[\alpha(i) - \beta(i)]$
12:    end for
13:    for i=1:M-1 do
14:        $s(:,i+1) = s(:,i)$
15:        $y(:,i+1) = y(:,i)$
16:    end for
17:
           $$\begin{cases} s(:,1) = x_{k+1} - x_k \\ y(:,1) = \nabla f(x_{k+1}) - \nabla f(x_k) \\ \rho_k(1) = \frac{1}{y(:,1)^T \cdot s(:,1)} \end{cases}$$
18:    return direction=$-z = -H_k \cdot R(x^k)$
19: end procedure
```

4.3 Forward backward envelop

Newton iterations only converge quickly when they are close to the solution. In order to get better global behavior a proper global strategy is required. The optimization problem is changed from $\varphi(x) = f(x) + g(x)$ to equation 24. This problem is smoother while it still has the same optimal solution.(proof see [3] and [2]) The same γ as with the proximal gradient should be used, notice how the FBE contains the line-search condition use in the proximal gradient. (more on the FBE in [2])

The Moreau envelope is de define as 23, this smooths the function, and has a close relationship with the proximal operator as illustrated earlier. Using simple algebra equation 24 can be transformed into equation 25.(proof in Appendix) The solution y of the infimum defined in equation 25 is $\bar{x}$. Considering the close relationship between the Moreau envelope and the proximal operator (see more in [2]) this is to be expected.

An alternative way to look at it is nicely illustrated in [1], where the problem can be seen as minimizing a quadratic approximation in point x. Remember that $L = \frac{1}{\gamma}$, and L is the Lipschitz constant of the gradient.

$$g^{\gamma} = \inf_{y} \{ f(y) + \frac{1}{2 \cdot \gamma} ||x - y||^2 \} \tag{23}$$

$$\varphi_{\gamma} = f(x) - \frac{\gamma}{2} ||\nabla f(x)||^2 + g^{\gamma}\big(x - \gamma \nabla f(x)\big) \tag{24}$$

$$\varphi_{\gamma} = f(x) + \inf_{y} \Big\{ \nabla f(x)^T (y - x) + g(y) + \frac{1}{2\gamma} ||x - y||^2 \Big\} \tag{25}$$

Proof. The solution to the infimum of equation 25 is y=$\bar{x} = \text{prox}_g(x - \gamma \nabla f(x))$

$$\begin{aligned}
\text{prox}_g(\bar{x}) &= \text{prox}_g(x - \gamma \nabla f(x)) \\
&= \arg\min_{y} \Big\{ g(y) + \frac{1}{2\gamma} ||(y - x) + \gamma \nabla f(x)||^2 \Big\} \\
&= \arg\min_{y} \Big\{ g(y) + \frac{1}{2\gamma} \big[||y - x||^2 + 2\gamma \nabla f(x)^T (y - x) + ||\nabla f(x)||^2 \gamma^2 \big] \Big\} \\
&= \arg\min_{y} \Big\{ g(y) + \frac{1}{2\gamma} \big[||y - x||^2 + 2\gamma \nabla f(x)^T (y - x) \big] \Big\} \\
&= \arg\min_{y} \Big\{ \nabla f(x)^T (y - x) + g(y) + \frac{1}{2\gamma} ||y - x||^2 \Big\}
\end{aligned}$$

□

Equation 26 is the practical implementation of the FBE. The parameter gamma is the line-search parameter used in the proximal gradient descent. The first 3 terms are the same as with the line-search on γ, the last term $g(\bar{x})$ is new and makes sure the solution complies with the constraint.

$$\begin{aligned} \varphi(\gamma, x) = \quad & f(x) - \nabla f(x)^T(x-\bar{x}) + \frac{1}{2\gamma}||x-\bar{x}||^2 + g(\bar{x}) \\ \textit{with} \qquad & \bar{x} = \operatorname{prox}_g(x - \gamma \nabla f(x)) \end{aligned} \tag{26}$$

4.4 Line-search with FBE

In [3] the line-search condition is specified as equation 28. The line-search parameter determines x^{k+1} by choosing the convex combination of the step from the proximal gradient and the LBGFS as defined in equation 27. It is also specifies in [3] that $\sigma \in (0, \gamma\frac{1-\gamma \cdot L}{2})$. As stated before it is assumed that $L = \frac{1-\beta}{\gamma}$, some simple algebra will lead to the condition $\sigma \in (0, \frac{\beta\gamma}{2})$.

Equation 30 is the practical implementation of equation 28. A new constant $\alpha \in (0,1)$ is introduced, an possible value for α would be 0.5 ($\alpha = 0.5$ is the choice used by the Matlab implementation of PANOC in ForBes).

$$x^{k+1} = u_k - (1-\tau_k) \cdot (x - \bar{x}) + \tau \cdot dir_{LBFGS} \tag{27}$$

$$\begin{aligned} \varphi_\gamma(x^{k+1}) &= \varphi_\gamma(x^k) - \sigma||\frac{x-\bar{x}}{\gamma}||^2 & (28) \\ &= \varphi_\gamma(x^k) - \frac{\sigma}{\gamma^2}||x-\bar{x}||^2 & (29) \end{aligned}$$

$$\varphi_\gamma(x^{k+1}) = \varphi_\gamma(x^k) - \alpha\frac{\beta}{\gamma \cdot 2}||x-\bar{x}||^2 \tag{30}$$

Algorithm 4 PANOC

```
 1: procedure PANOC_GET_NEW_LOCATION($x^k$,$\gamma$)
 2:     $[(x-\bar{x})\ ,\ \gamma]$ = GET_PROXIMAL_GRADIENT_STEP($\gamma$,$x^k$)
 3:     $dir_{LBFGS}$ = LBFGS($x^k$)
 4:     $\tau = 1$
 5:     $x^{k+1} = x_k - (1-\tau_k) \cdot (x-\bar{x}) + \tau \cdot dir_{LBFGS}$
 6:     while $\varphi_\gamma(x^{k+1}) = \varphi_\gamma(x^k) - \frac{\beta\gamma}{2}||(x-\bar{x})||^2$ do
 7:         $\tau = \tau/2$
 8:         $x^{k+1} = x_k - (1-\tau_k) \cdot (x-\bar{x}) + \tau \cdot dir_{LBFGS}$
 9:     end while
10: end procedure
```

5 Implementation

5.1 the dynamic C code

5.2 The static C lib

The static c code is structured in an layered architecture. The PANOC algorithm is the first layer. PANOC calls the second layer containing the LBFGS and proximal gradient descent. And finally the lowest layer containing the Casadi functions, a buffer and the Lipschitz estimator.

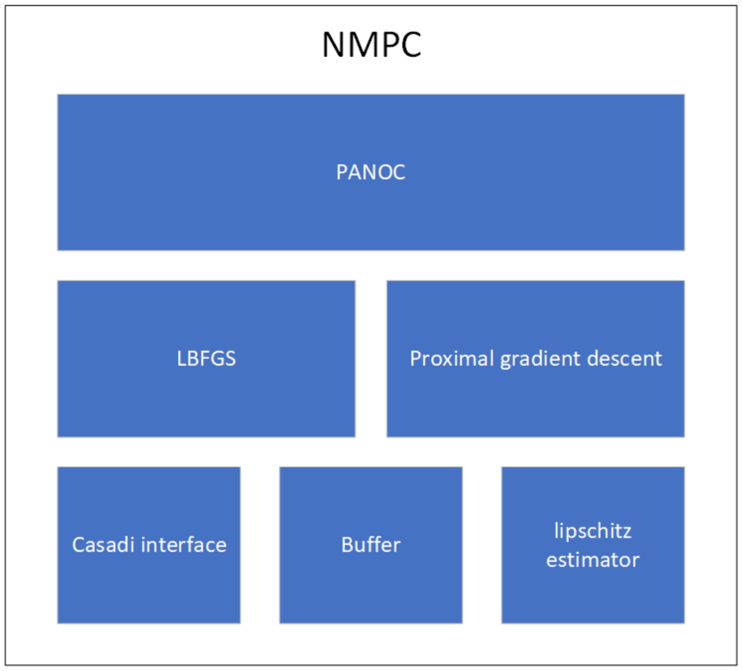

Figure 1: software architecture

A Function Definitions

A.1 box function

1 dimension:

$$box_{1D}(u) = \begin{matrix} 1 & u \in [-U_b : U_b] \\ 0 & otherwise \end{matrix} \tag{31}$$

N dimensions:

$$box(u) = min \left[\sum_{k=1}^{N} box_{1D}(u) \right] \tag{32}$$

A.2 Indicator Box function

$$I[box] = \begin{matrix} 0 & u \in [-U_b : U_b] \\ \inf & otherwise \end{matrix} \tag{33}$$

$$prox[I[box]] = \begin{matrix} u & u \in [-U_b : U_b] \\ -U_b & u \in [-\inf : -U_b] \\ -U_b & u \in [U_b : \inf] \end{matrix} \tag{34}$$

B functions

B.1 Conjugate of strongly convex function

Following lemma's are useful from

$$f^*(x) = \underset{u \in dom(f)}{< y, x >} - f(y) \tag{35}$$

If ∇f^* is lipschitz and obeys equation 37, then f^* is well defined and differentiable. (assume dom(f) is convex and closed)

$$\nabla f^*(x) = y^* = \text{argmax} < y, x > - f(y) \tag{36}$$

$$||\nabla f^*(x) - \nabla f^*(y)||_2 \leq \mu^{-1} ||x - y||_2 \tag{37}$$

C Proof FBE alternate equation

Proof. $\varphi_\gamma = f(x) + \inf_y \Big\{ \nabla f(x)^T(y-x) + g(y) + \frac{1}{2\gamma}||x-y||^2 \Big\}$

$$\begin{aligned}
g^\gamma &= \inf_y \{ f(y) + \frac{1}{2\cdot\gamma}||x-y||^2 \} \\
\varphi_\gamma &= f(x) - \frac{\gamma}{2}||\nabla f(x)||^2 + g^\gamma\big(x - \gamma\nabla f(x)\big) \\
&= f(x) - \frac{\gamma}{2}||\nabla f(x)||^2 + g^\gamma\big(\bar{x}\big) \\
g^\gamma(\bar{x}) &= \inf_y \Big\{ g(y) + \frac{1}{2\gamma}||\bar{x}-y||^2 \Big\} \\
\frac{1}{2\gamma}||\bar{x}-y||^2 &= \frac{1}{2\gamma}\Big[(\bar{x}-y)^T(\bar{x}-y)\Big] \\
&= \frac{1}{2\gamma}\Big[x^Tx - 2x^Ty + y^Ty\Big] \\
\bar{x}^T\bar{x} &= [x-\gamma\nabla f(x)]^T[x-\gamma\nabla f(x)] \\
&= x^Tx - 2x^T\nabla f(x)\gamma + \gamma^2\nabla f(x)^T\nabla f(x) - 2x^Ty \\
&= -2(x-\gamma f\nabla(x))^Ty \\
&= -2x^Ty + 2\gamma\nabla f(x)^Ty \\
\frac{1}{2\gamma}||\bar{x}-y||^2 &= \frac{1}{2\gamma}[x^Tx - 2x^T\nabla f(x)\gamma + \gamma^2\nabla f(x)^T\nabla f(x) - 2x^Ty + 2\gamma\nabla f(x)^Ty + y^Ty] \\
&= \frac{1}{2\gamma}[-2x^T\nabla f(x)\gamma + 2\gamma\nabla f(x)^Ty + \gamma^2\nabla f(x)^T\nabla f(x) + x^Tx - 2x^Ty + y^Ty] \\
&= \frac{1}{2\gamma}[2\gamma\nabla f(x)^T(y-x) + \gamma^2||\nabla f(x)||^2 + (x-y)^T(x-y)] \\
&= \frac{1}{2\gamma}[2\gamma\nabla f(x)^T(y-x) + \gamma^2||\nabla f(x)||^2 + ||x-y||^2] \\
&= \nabla f(x)^T(y-x) + \frac{\gamma}{2}||\nabla f(x)||^2 + \frac{1}{2\gamma}||x-y||^2 \\
g^\gamma(\bar{x}) &= \inf_y \Big\{ g(y) + \nabla f(x)^T(y-x) + \frac{\gamma}{2}||\nabla f(x)||^2 + \frac{1}{2\gamma}||x-y||^2 \Big\} \\
&= \frac{\gamma}{2}||\nabla f(x)||^2 + \inf_y \Big\{ g(y) + \nabla f(x)^T(y-x) + \frac{1}{2\gamma}||x-y||^2 \Big\} \\
\varphi_\gamma &= f(x) - \frac{\gamma}{2}||\nabla f(x)||^2 + g^\gamma\big(\bar{x}\big) \\
&= f(x) - \frac{\gamma}{2}||\nabla f(x)||^2 + \frac{\gamma}{2}||\nabla f(x)||^2 + \inf_y \Big\{ g(y) + \nabla f(x)^T(y-x) + \frac{1}{2\gamma}||x-y||^2 \Big\} \\
&= f(x) + \inf_y \Big\{ \nabla f(x)^T(y-x) + g(y) + \frac{1}{2\gamma}||x-y||^2 \Big\}
\end{aligned}$$

□

References

[1] Ruben Van Parys Andreas Themelis Goele Pipeleers Ajay Sathya, Pantelis Sopaskis and Panagiotis Patrinos. Embedded nonliear model predicitve control for obstacle avoidance using panoc. 2017.

[2] Panagiotis Patrinos Andreas Themelis, Lorenzo Stella. Forward-backward envelope for the sum of two nonconvex functions: further properties and nonmonotone line-search algorithms.

[3] Pantelis Sopasakis Lorenzo Stella, Andreas Themelis and Panagiotis Patrinos. A simple and efficient algoirthm for nonlinear model predictive control. 2017.

[4] Stephen Boyd Neal Parikh. Proximal algorithms.

[5] Emmanuel Candes Qian Yang. Math 301: Advanced topics in convex optimization, lecture 22.