Gradient methods for minimizing composite functions
Nesterov's method

Jakub Bilski, Jakub Brojacz

April 28, 2022

1 Introduction

The goal is to minimize the following function:

$$\phi(x) = f(x) + \Psi(x) \tag{1}$$

where $x \in R^n$, $\Psi(x) = \lambda||x||_1$ and $f(x) = \frac{1}{2}||y - Xx||_2^2$ for some $\lambda \in R$, $y \in R^p$, $X \in R^{p \times n}, n \in N, p \in N$.

2 Basic symbols, equations

2.1 Composite gradient mapping

At any $x, y \in R^n$, define

$$\begin{aligned} m_L(y; x) &= f(y) + \langle \nabla f(y), x - y \rangle + \frac{L}{2}||x - y||^2 + \Psi(x) \\ T_L(y) &= \arg\min_{x \in R^n} m_L(y; x) \end{aligned} \tag{2}$$

where L is a positive constant. For function (1), we obtain

$$
\begin{aligned}
T_L(y) &= \arg\min_{x\in R^n} m_L(y,x) = \\
&= \arg\min_{x\in R^n} f(y) + \langle \nabla f(y), x-y\rangle + \frac{L}{2}||x-y||^2 + \Psi(x) = \\
&= \arg\min_{x\in R^n} \langle \nabla f(y), x\rangle - \langle \nabla f(y), y\rangle + \frac{L}{2}||x-y||^2 + \Psi(x) = \\
&= \arg\min_{x\in R^n} \langle \nabla f(y), x\rangle + \frac{L}{2}||x-y||^2 + \lambda||x||_1 = \\
&= \arg\min_{x\in R^n} \sum_{i=1}^{n} \left[(\nabla f(y))^{(i)} x^{(i)} + \frac{L}{2}(x^{(i)} - y^{(i)})^2 + \lambda|x^{(i)}| \right] = \\
&= \arg\min_{x\in R^n} \sum_{i=1}^{n} \left[\frac{L}{2}(x^{(i)})^2 + ((\nabla f(y))^{(i)} - Ly^{(i)})x^{(i)} + \lambda|x^{(i)}| \right]
\end{aligned}
\tag{3}
$$

Each term in the sum corresponds to only one element of the x vector. Thus, we transformed a single minimization problem with argument $x \in R^n$ into n independent problems with arguments $x^{(i)} \in R$.

$$
T_L(y)^{(i)} = \arg\min_{x^{(i)}\in R} \frac{L}{2}(x^{(i)})^2 + ((\nabla f(y))^{(i)} - Ly^{(i)})x^{(i)} + \lambda|x^{(i)}| \tag{4}
$$

Breaking down into cases and examining the quadratic functions, it's easy to obtain the following formula.

$$
T_L(y)^{(i)} = \begin{cases} \frac{Ly^{(i)} - \nabla f(y)^{(i)} - \lambda}{L} & \text{for } \nabla f(y)^{(i)} - Ly^{(i)} \le -\lambda \\ \frac{Ly^{(i)} - \nabla f(y)^{(i)} + \lambda}{L} & \text{for } \nabla f(y)^{(i)} - Ly^{(i)} \ge \lambda \\ 0 & \text{otherwise} \end{cases} \tag{5}
$$

2.2 Constrained analogue of the gradient direction

At any $y \in R^n$, define

$$
g_L(y) = L \cdot (y - T_L(y)) \tag{6}
$$

2.3 Estimate of a local variation of function ϕ

At any $y \in R^n$, define

$$
S_L(y) = \frac{||\nabla f(T_L(y)) - \nabla f(y)||_*}{||T_L(y) - y||} \le L_f \tag{7}
$$

where L_f is the Lipschitz constant of function ϕ.

2.4 Gradient Iteration

For input $x \in R^n$, $M \in R^+$ and parameter $\gamma_u > 1$, Gradient Iteration allows us to compute $G(x, M)$ using the following algorithm.
SET:

$$L := M$$

REPEAT:

$$\begin{aligned} &T := T_L(x) \\ &\text{if } \phi(T) > m_L(x; T) \text{ then } L := L \cdot \gamma_u \end{aligned} \tag{8}$$

UNTIL:

$$\phi(T) \leq m_L(x; T)$$

OUTPUT:

$$\begin{aligned} &G(x, M).T = T \\ &G(x, M).L = L \\ &G(x, M).S = S_L(x) \end{aligned} \tag{9}$$

3 Optimization methods

3.1 Basic Method

For parameter $\gamma_d \geq 1$, initial optimistic estimate L_0 of the Lipschitz constant L_f that satisfies $0 \leq L_0 \leq L_f$ and starting point $y_0 \in R^n$, Basic Method computes the optimization result y in an iterative fashion. The result after $k + 1$ steps is obtained using the following algorithm.

$$\begin{aligned} &y_{k+1} := G(y_k, L_k).T \\ &M_k := G(y_k, L_k).L \\ &L_{k+1} := max\{L_0, M_k/\gamma_d\} \end{aligned} \tag{10}$$

where y_{k+1} is the optimization result.

3.2 Dual Gradient Method

For parameter $\gamma_d \geq 1$, initial optimistic estimate L_0 of the Lipschitz constant L_f that satisfies $0 \leq L_0 \leq L_f$ and starting value $v_0 \in R^n$, Dual Gradient Method computes the optimization result y in an iterative fashion. The result after $k+1$ steps is obtained using the following algorithm.

$$\begin{aligned}
&y_k := G(v_k, L_k).T \\
&M_k := G(v_k, L_k).L \\
&L_{k+1} := max\{L_0, M_k/\gamma_d\} \\
&\psi_{k+1}(x) = \psi_k(x) + \frac{1}{M_k}[f(v_k) + \langle \nabla f(v_k), x - v_k\rangle + \Psi(x)] \\
&v_{k+1} := \arg\min_{x \in R^n} \psi_{k+1}(x)
\end{aligned} \tag{11}$$

$$\psi(x) = l_k(x) + A_k\phi(x) + \frac{1}{2}||x - x_0||^2$$

where $\psi_0(x) = \frac{1}{2}||x - v_0||^2$.

Here, y_k is the optimization result. For function (1), we get

$$\begin{aligned}
v_{k+1} &= \arg\min_{x \in R^n} \psi_{k+1}(x) = \\
&= \arg\min_{x \in R^n} \left[\psi_k(x) + \frac{f(v_k)}{M_k} + \frac{\langle \nabla f(v_k), x - y\rangle}{M_k} + \frac{\Psi(x)}{M_k}\right] = \\
&= \arg\min_{x \in R^n} \left[\psi_k(x) + \frac{\langle \nabla f(v_k), x\rangle}{M_k} - \frac{\langle \nabla f(v_k), y\rangle}{M_k} + \frac{\Psi(x)}{M_k}\right] = \\
&= \arg\min_{x \in R^n} \left[\psi_k(x) + \left\langle \frac{\nabla f(v_k)}{M_k}, x \right\rangle + \Psi(x)\frac{1}{M_k}\right] = \\
&= \arg\min_{x \in R^n} \left[\sum_{i=0}^{k} \left[\left\langle \frac{\nabla f(v_i)}{M_i}, x \right\rangle + \Psi(x)\frac{1}{M_i}\right] + \psi_0(x)\right] = \\
&= \arg\min_{x \in R^n} \left[\left\langle \sum_{i=0}^{k} \frac{\nabla f(v_i)}{M_i}, x \right\rangle + \Psi(x)\left[\sum_{i=0}^{k} \frac{1}{M_i}\right] + \frac{1}{2}||x - v_0||^2\right]
\end{aligned} \tag{12}$$

From now on, we denote $A_k = \sum_{i=0}^{k} \frac{\nabla f(v_i)}{M_i}$, $b_k = \sum_{i=0}^{k} \frac{1}{M_i}$.

$$
\begin{aligned}
v_{k+1} &= \arg\min_{x\in R^n}\left[\langle A_k, x\rangle + \lambda||x||_1 b_k + \frac{1}{2}||x-v_0||^2\right] \\
&= \arg\min_{x\in R^n}\sum_{i=1}^{n}\left[A_k^{(i)}x^{(i)} + \lambda|x^{(i)}|b_k^{(i)} + \frac{1}{2}(x^{(i)}-v_0^{(i)})^2\right] \\
&= \arg\min_{x\in R^n}\sum_{i=1}^{n}\left[\frac{1}{2}(x^{(i)})^2 + (A_k^{(i)}-v_0^{(i)})x^{(i)} + \lambda b_k^{(i)}|x^{(i)}|\right]
\end{aligned} \tag{13}
$$

Just like in (3), we obtained n independent minimization problems with arguments $x^{(i)} \in R$.

$$
v_{k+1}^{(i)} = \arg\min_{x^{(i)}\in R}\frac{1}{2}(x^{(i)})^2 + (A_k^{(i)}-v_0^{(i)})x^{(i)} + \lambda b_k^{(i)}|x^{(i)}| \tag{14}
$$

This expression is of the same type as (4). After performing the same transformations as in (4), we get the final result.

$$
v_{k+1}^{(i)} = \begin{cases} v_0^{(i)} - A_k^{(i)} - \lambda b_k & \text{for } A_k^{(i)} - v_0^{(i)} \le -\lambda b_k \\ v_0^{(i)} - A_k^{(i)} + \lambda b_k & \text{for } A_k^{(i)} - v_0^{(i)} \ge \lambda b_k \\ 0 & \text{otherwise} \end{cases} \tag{15}
$$

where

$$
\begin{aligned}
A_{k+1} &= A_k + \frac{\nabla f(v_{k+1})}{M_{k+1}} \\
b_{k+1} &= b_k + \frac{1}{M_{k+1}}
\end{aligned} \tag{16}
$$

and $A_0 = \frac{\nabla f(v_0)}{M_0}$, $b_0 = \frac{1}{M_0}$. A_k and b_k must have been expressed in an interative way (as opposed to $A_k = \sum_{i=0}^{k}\frac{\nabla f(v_i)}{M_i}$, $b_k = \sum_{i=0}^{k}\frac{1}{M_i}$) to sustain the time complexity of the algorithm.

3.3 Accelerated Method

For initial optimistic estimate L_0 of the Lipschitz constant L_f that satisfies $0 \leq L_0 \leq L_f$, lower estimate $\mu \in [0, \mu_\Psi]$ of the convexity parameter of function Ψ, starting value $x_0 \in R^n$, initial values $\psi_0(x) = \frac{1}{2}||x - x_0||^2$ and $A_0 = 0$, Accelerated Method computes the optimization result y in an iterative fashion. The result after $k+1$ steps is obtained using the following algorithm.

SET:

$$L := L_k \tag{17}$$

REPEAT:

$$\begin{aligned} a &:= \frac{2 + 2\mu A_k \pm \sqrt{(2 + 2\mu A_k)^2 + 8LA_k(1 + \mu A_k)}}{2L} \\ y &:= \frac{A_k x_k + a v_k}{A_k + a} \\ &\text{if } \langle \phi'(T_L(y)), y - T_L(y) \rangle < \frac{1}{L}||\phi'(T_L(y))||_*^2 \text{ then } L := L \cdot \gamma_u \end{aligned} \tag{18}$$

UNTIL:

$$\langle \phi'(T_L(y)), y - T_L(y) \rangle \geq \frac{1}{L}||\phi'(T_L(y))||^2 \tag{19}$$

DEFINE:

$$\begin{aligned} &y_k := y \\ &M_k := L \\ &a_{k+1} := a \\ &L_{k+1} := \frac{M_k}{\gamma_d} \\ &x_{k+1} := T_{M_k}(y_k) \\ &\psi_{k+1}(x) := \psi_k(x) + a_{k+1}\left[f(x_{k+1}) + \langle \nabla f(x_{k+1}), x - x_{k+1} \rangle + \Psi(x)\right] \\ &A_{k+1} := A_k + a_{k+1} \\ &v_{k+1} := \arg\min_{x \in R^n} \psi_{k+1}(x) \end{aligned} \tag{20}$$

analogically to the Dual Gradient Method, we get

$$
\begin{aligned}
v_{k+1} &= \arg\min_{x\in R^n} \psi_{k+1}(x) = \\
&= \arg\min_{x\in R^n} \left[\psi_k(x) + a_{k+1}\cdot f(x_{k+1}) + a_{k+1}\cdot\langle \nabla f(x_{k+1}), x - x_{k+1}\rangle + a_{k+1}\cdot\Psi(x) \right] = \\
&= \arg\min_{x\in R^n} \left[\psi_k(x) + a_{k+1}\cdot\langle \nabla f(x_{k+1}), x\rangle - a_{k+1}\cdot\langle \nabla f(x_{k+1}), x_{k+1}\rangle + a_{k+1}\cdot\Psi(x) \right] = \\
&= \arg\min_{x\in R^n} \left[\psi_k(x) + \langle a_{k+1}\cdot\nabla f(x_{k+1}), x\rangle + a_{k+1}\cdot\Psi(x) \right] = \\
&= \arg\min_{x\in R^n} \left[\sum_{i=1}^{k+1} \left[\langle a_i\cdot\nabla f(x_i), x\rangle + a_i\cdot\Psi(x) \right] + \psi_0(x) \right] = \\
&= \arg\min_{x\in R^n} \left[\left\langle \sum_{i=1}^{k+1} a_i\cdot\nabla f(x_i), x \right\rangle + \Psi(x)\left[\sum_{i=1}^{k+1} a_i\right] + \frac{1}{2}||x - x_0||^2 \right]
\end{aligned}
\tag{21}
$$

We denote $C_{k+1} = \sum_{i=1}^{k+1} a_i \nabla f(x_i)$, $b_{k+1} = \sum_{i=1}^{k+1} a_i$.

$$
\begin{aligned}
v_{k+1} &= \arg\min_{x\in R^n} \left[\langle C_{k+1}, x\rangle + \lambda||x||_1 b_{k+1} + \frac{1}{2}||x - x_0||^2 \right] \\
&= \arg\min_{x\in R^n} \sum_{i=1}^{n} \left[C_{k+1}^{(i)} x^{(i)} + \lambda|x^{(i)}| b_{k+1}^{(i)} + \frac{1}{2}(x^{(i)} - x_0^{(i)})^2 \right] \\
&= \arg\min_{x\in R^n} \sum_{i=1}^{n} \left[\frac{1}{2}(x^{(i)})^2 + (C_{k+1}^{(i)} - x_0^{(i)})x^{(i)} + \lambda b_{k+1}^{(i)}|x^{(i)}| \right]
\end{aligned}
\tag{22}
$$

We obtain final result similar to Dual Gradient Method:

$$
v_{k+1}^{(i)} = \begin{cases} x_0^{(i)} - C_{k+1}^{(i)} - \lambda b_{k+1} & \text{for } C_{k+1}^{(i)} - x_0^{(i)} \le -\lambda b_{k+1} \\ x_0^{(i)} - C_{k+1}^{(i)} + \lambda b_{k+1} & \text{for } C_{k+1}^{(i)} - x_0^{(i)} \ge \lambda b_{k+1} \\ 0 & \text{otherwise} \end{cases}
\tag{23}
$$

where

$$
\begin{aligned}
C_{k+1} &= C_k + a_{k+1}\cdot\nabla f(x_{k+1}) \\
b_{k+1} &= b_k + a_{k+1}
\end{aligned}
\tag{24}
$$

and $C_0 = 0$, $b_0 = 0$.

Since we cannot explicitly compute $\phi'(T_L(y))$, in order to evaluate the conditional statement in (18) we first need to transform it.

From now on, we denote $T = T_L(y)$. Note that Ψ is closed and strongly convex on R, which enables us to write the first-order optimality condition for problem (2) in the following form:

$$\nabla f(y) + \xi_L(y) = L(y - T) \tag{25}$$

where $\xi_L(y) \in \partial\Psi(T)$. With this, we get:

$$\begin{aligned}
\phi'(T) &= \nabla f(T) + \xi_L(y) \\
&= L(y - T) + \nabla f(T) - \nabla f(y) \\
||\phi'(T)||^2 &= L^2||y - T||^2 + 2L\left\langle \nabla f(T) - \nabla f(y), y - T\right\rangle \\
&\quad + ||\nabla f(T) - \nabla f(y)||^2 \\
L||y - T||^2 &= \frac{1}{L}||\phi'(T)||^2 + 2\left\langle \nabla f(y) - \nabla f(T), y - T\right\rangle \\
&\quad - \frac{1}{L}||\nabla f(y) - \nabla f(T)||^2
\end{aligned} \tag{26}$$

We use this result to transform the term from (18):

$$\begin{aligned}
\langle \phi'(T), y - T\rangle &= \langle L(y - T) + \nabla f(y) - \nabla f(T), y - T\rangle \\
&= L||y - T||^2 - \langle \nabla f(y) - \nabla f(T), y - T\rangle \\
&= \frac{1}{L}||\phi'(T)||^2 + 2\left\langle \nabla f(y) - \nabla f(T), y - T\right\rangle \\
&\quad - \frac{1}{L}||\nabla f(y) - \nabla f(T)||^2 - \langle \nabla f(y) - \nabla f(T), y - T\rangle \\
&= \frac{1}{L}||\phi'(T)||^2 + \langle \nabla f(y) - \nabla f(T), y - T\rangle \\
&\quad - \frac{1}{L}||\nabla f(y) - \nabla f(T)||^2
\end{aligned} \tag{27}$$

Now, the conditional statement from (18) can now be expressed as:

$$\text{if } \langle \nabla f(T) - \nabla f(y), y - T\rangle < \frac{1}{L}||\nabla f(T) - \nabla f(y)||^2 \text{ then } L := L \cdot \gamma_u \tag{28}$$

Evaluation of the UNTIL statement from (19) is now also possible, as it is just a negation of this condition.

4 Implementation

All three methods were implemented using Python 3.9. and NumPy library.

5 Data

Generated

Independent values were generated as a normal distribution cloud with spherical covariance matrix with values on diagonal equal to 0.01. Mean point for the cloud was generated as a random point inside a p-dimensional unit cube, where p is the number of features, set to 30. Each predicted variable was computed as a linear composition of indepent variables summed with a random noise value from 0 to 1. Number of samples was set to 500.

Real

We used a well-known dataset for regression with one predicted value, the Boston House Prices Dataset. The dataset contains 506 samples with 13 features.

6 Experiments

Choice of λ parameter

Problem (1) can be fully described by a choice of dataset and λ parameter. We examine two different values of λ: $\{2, 10\}$ for the generated dataset and $\{50, 5000\}$ for the real one. These values were selected to yield substantially different results in terms of behaviour of the methods while ensuring that the nature of the problem (1) remains unchanged (i.e. some elements of x still approach zero).

Percentage of zeros

A way to measure characteristics of the problem can be calculating percentage of zeros in the solution vectors after all three methods converge. To justify the choice of λ parameter, we show that two problems in both datasets give significantly different results for different λ values, while the first problems and the second problems give similar results across the datasets. The reason for the difference in λ parameters and the number of iterations needed to obtain a stable solution originates from datasets' unique characteristics.

data	λ	steps	method	0s-share
generated	2	1000	basic	0.133
generated	2	1000	dual	0.133
generated	2	1000	accelerated	0.133
generated	10	20000	basic	0.700
generated	10	20000	dual	0.700
generated	10	20000	accelerated	0.700
real	50	1000000	basic	0.154
real	50	1000000	dual	0.154
real	50	1000000	accelerated	0.154
real	5000	20000	basic	0.615
real	5000	20000	dual	0.615
real	5000	20000	accelerated	0.615

Table 1: Share of zeros in solution after the last step

6.1 Comparison of methods

Author's of the original article chose $\gamma_u = 2$, $\gamma_d = 2$, $\mu = 0$ in all of their experiments. For this part, we followed their example. As for the starting values, we set $L_0 = 0.1$ and selected starting points (y_0, v_0 or x_0, depending on the algorithm) randomly from a p-dimensional unit cube.

6.1.1 Generated data, $\lambda = 2$

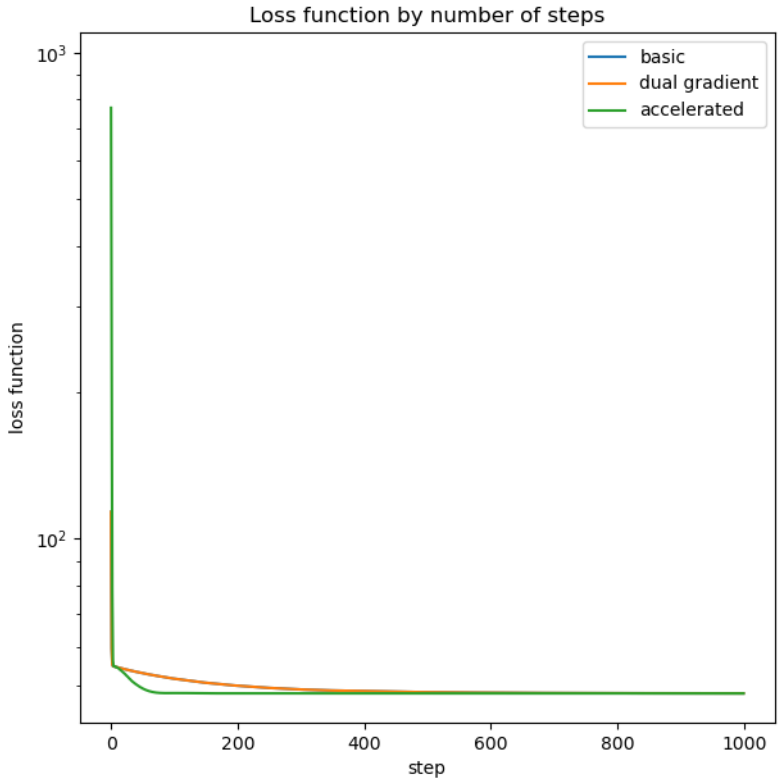

Figure 1: Change of the loss function values

As showed on the Figure 4, Accelerated Method outperformed other two methods during the first 300 iterations, while there was no noticeable difference between Dual Gradient Method and Basic Method. The final results of all three algorithms did not visibly differ.

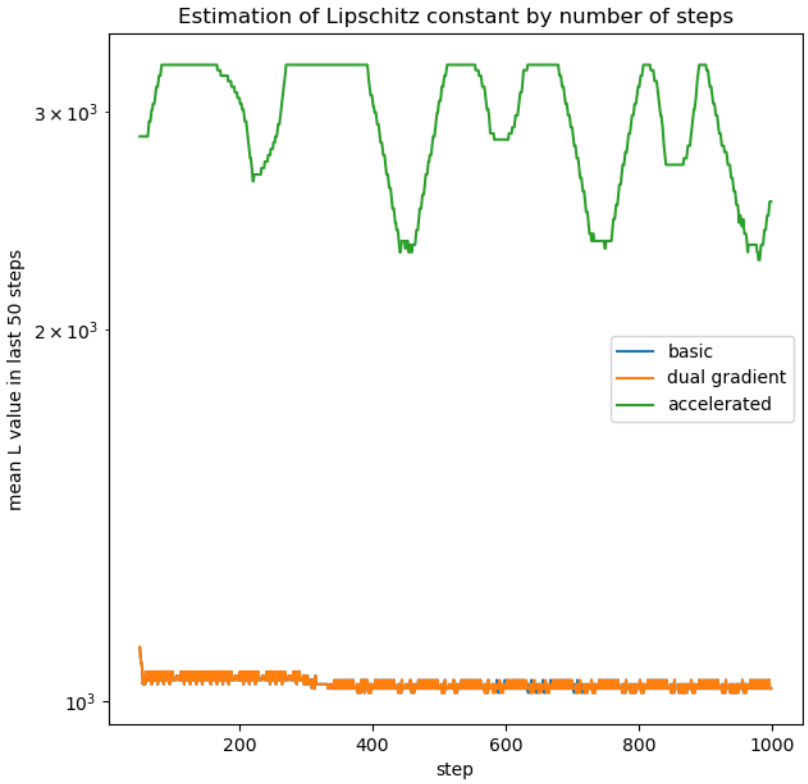

Figure 2: Estimations of Lipschitz constant

Estimating the Lipschitz constant plays an important role in all of the three algorithms. During this experiment, Dual Gradient Method's and Basic Method's estimations were on a very similar level, while values estimated by Accelerated Method were much higher and tended to fluctuate more. The figure shows mean of last 50 steps to make the plot more readable.

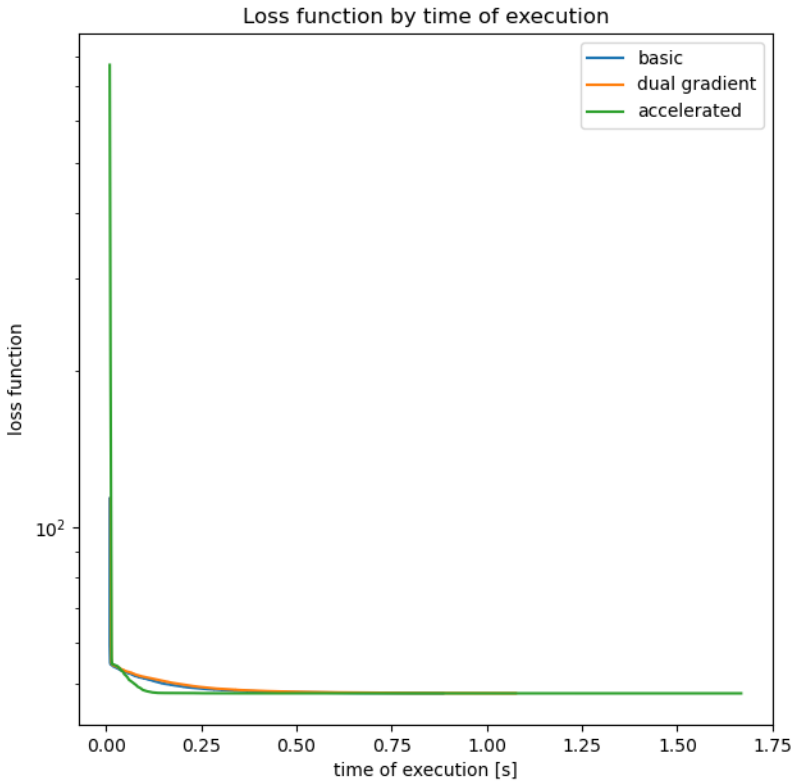

Figure 3: Performance in time

Figure 6 shows loss function values as a function of the time of execution. This allows us to see a very small advantage of Basic Method over Dual Gradient Method, which must have resulted from slower execution of a single iteration, as the convergence in number of steps did not differ, as shown in Figure 4.

6.1.2 Generated data, $\lambda = 10$

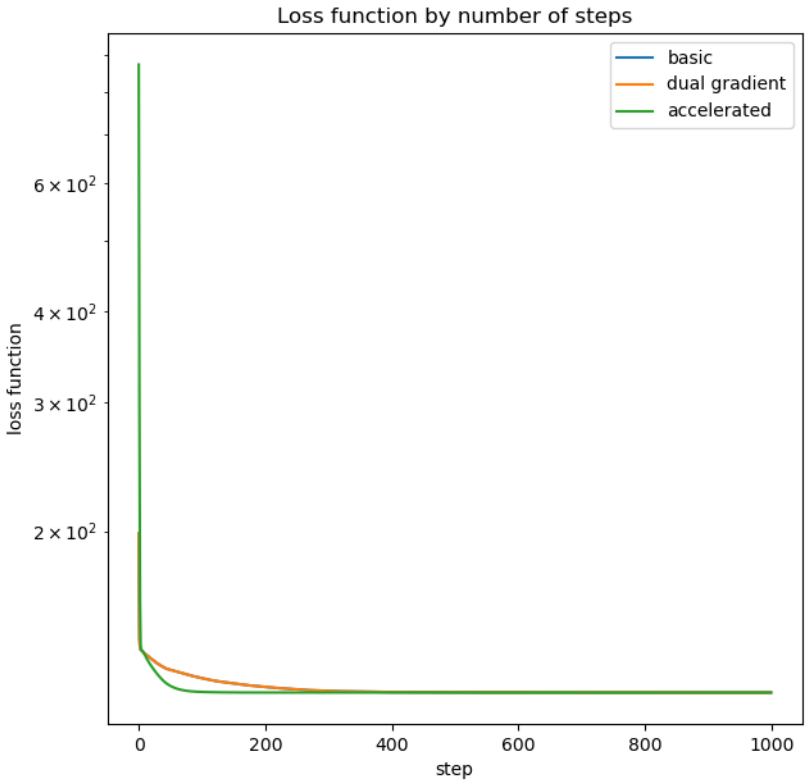

Figure 4: Change of the loss function values

Relations of methods for $\lambda = 10$ were similar to the results when $\lambda = 2$, as well as the number of steps performed between converging to a stable solution. Despite big differences in the number of zeros in the solution, these two problems turned out to be of similar difficulty.

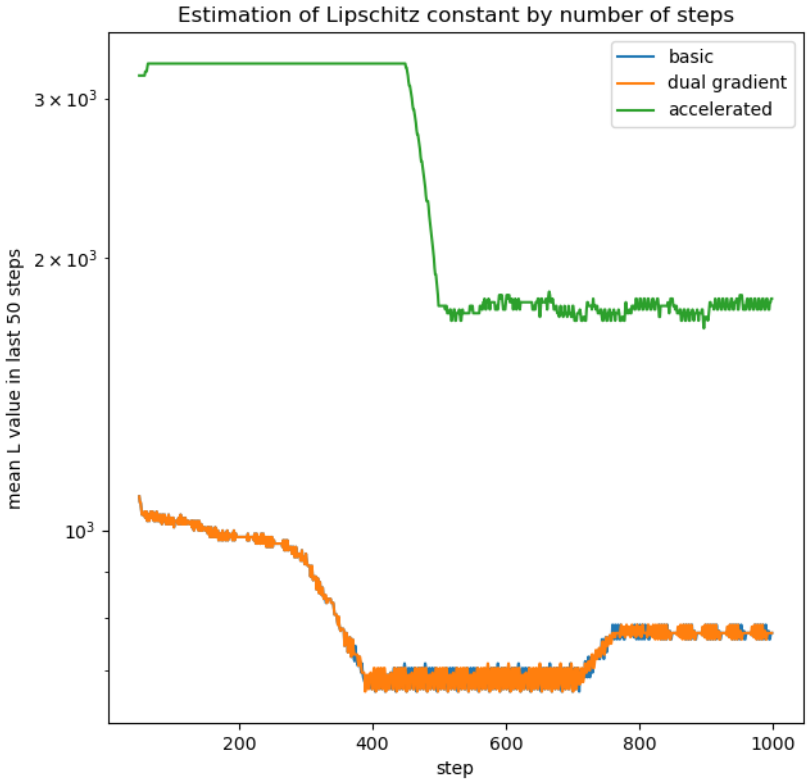

Figure 5: Estimations of Lipschitz constant

Once again, estimations of Lipschitz constant made by Accelerated Method remained on a higher level than those of other two methods. The seemingly big change in the Accelerated Method's prediction is not a big anomaly - this difference meant just one less multiplication by $\gamma_u = 2$. Lookng at Accelerated Method algorithm, it easy to see that with $\gamma_p = 2$, L can only take values in the form of $L_0 \cdot 2^n$ where $n \in N$. Hence, the drop around step 500 was just a permament change to the next smaller L.

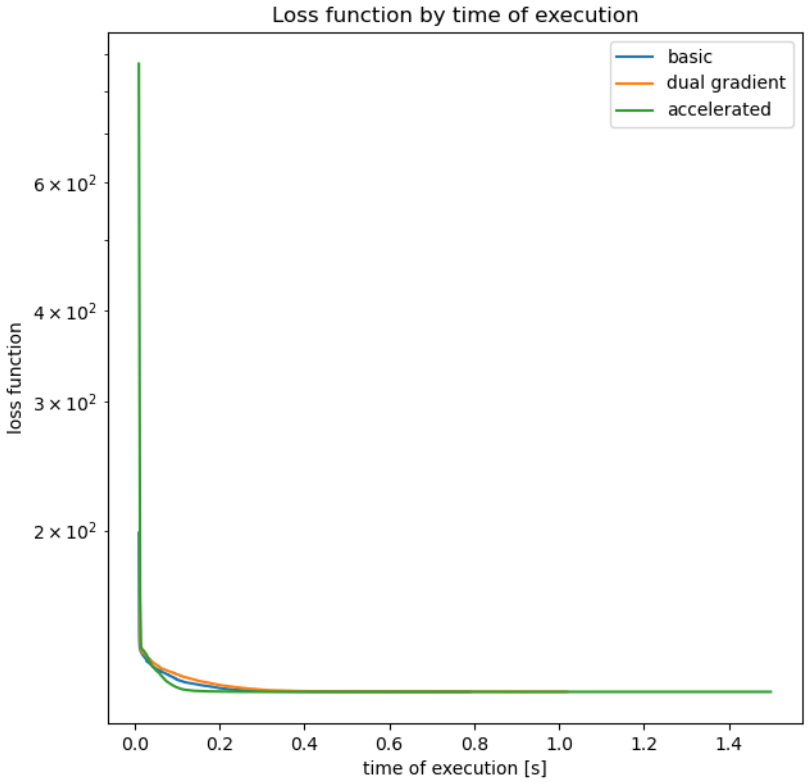

Figure 6: Performance in time

For the second time, the advantage of Basic Method over Dual Gradient Method was established when the time of execution was taken into accout. Nonetheless, Accelerated Method still had a strong lead.

6.1.3 Real data, $\lambda = 50$

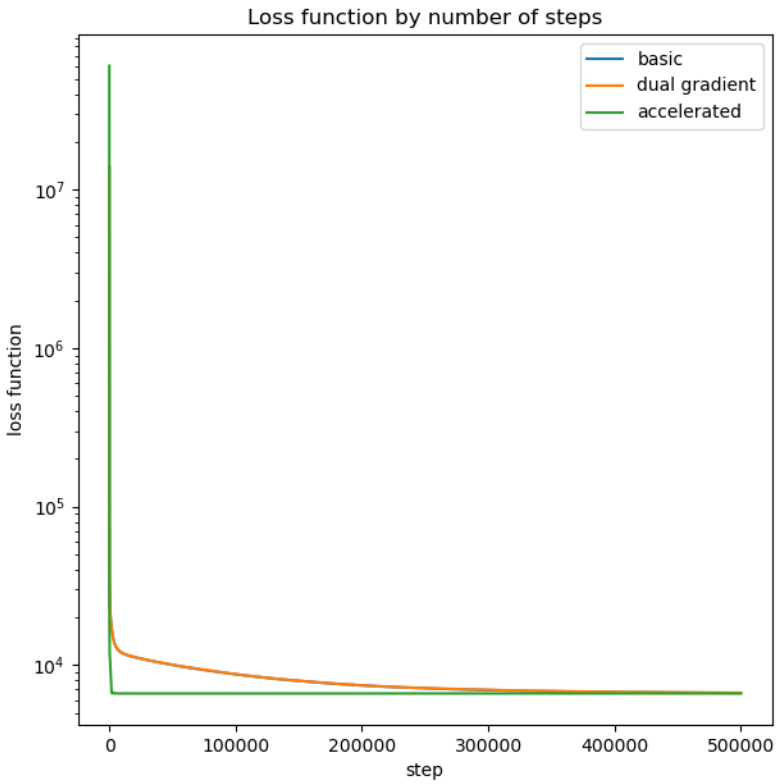

Figure 7: Change of the loss function values

Real data came with an even greater lead in performance of Accelerated Method, while results of Basic Method and Dual Gradient Method remained the same. Despite the fact that both datasets contained a similar number of samples, the real problem proved to be much harder for all three algorithms, especially for Basic and Dual Gradient Methods, hence the very big number of steps needed to converge.

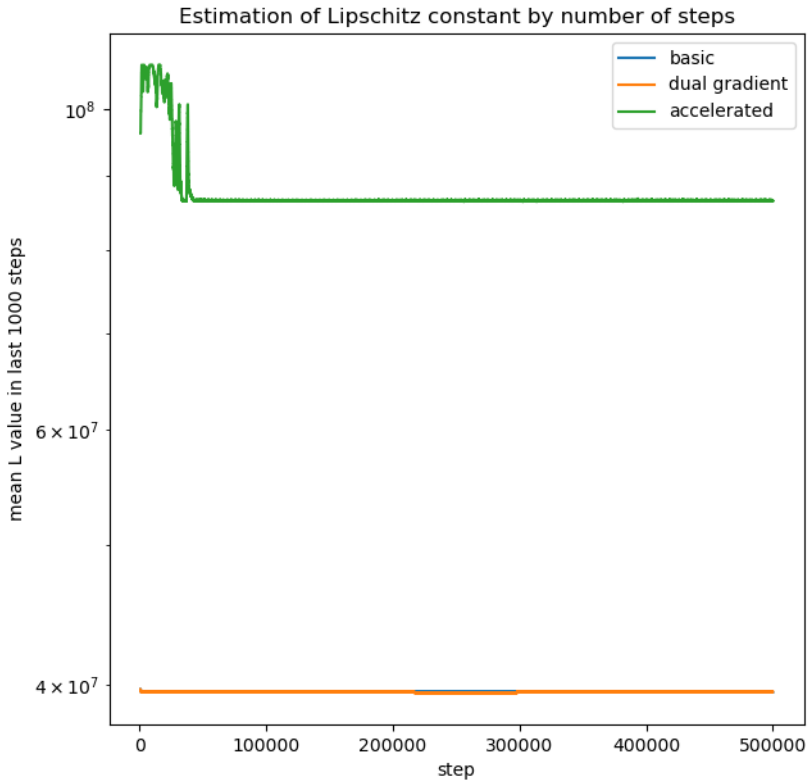

Figure 8: Estimations of Lipschitz constant

Estimation of Lipschitz constant was of a different nature than for the generated data, as Dual Gradient Method and Basic Method estimations did not visibly change over time. Moreover, the estimation made by Accelerated Method freezed around step 40000, probably after it converged to a stable solution.

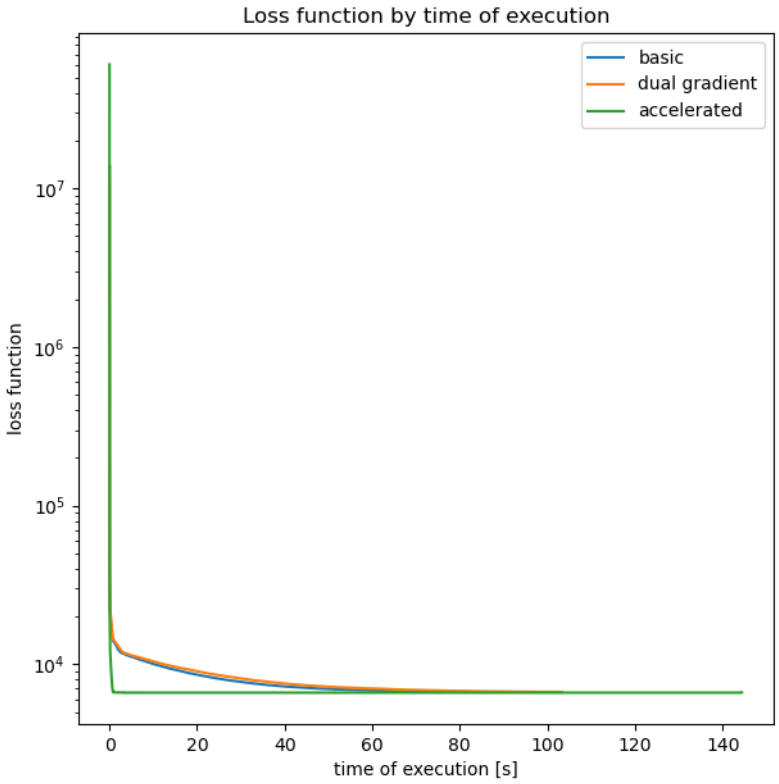

Figure 9: Performance in time

Performance in time does not yield any unexpected results when compared to performance in steps. There was a small difference between Basic Method and Dual Gradient Method in favor of the latter, but hardly noticeable.

6.1.4 Real data, $\lambda = 5000$

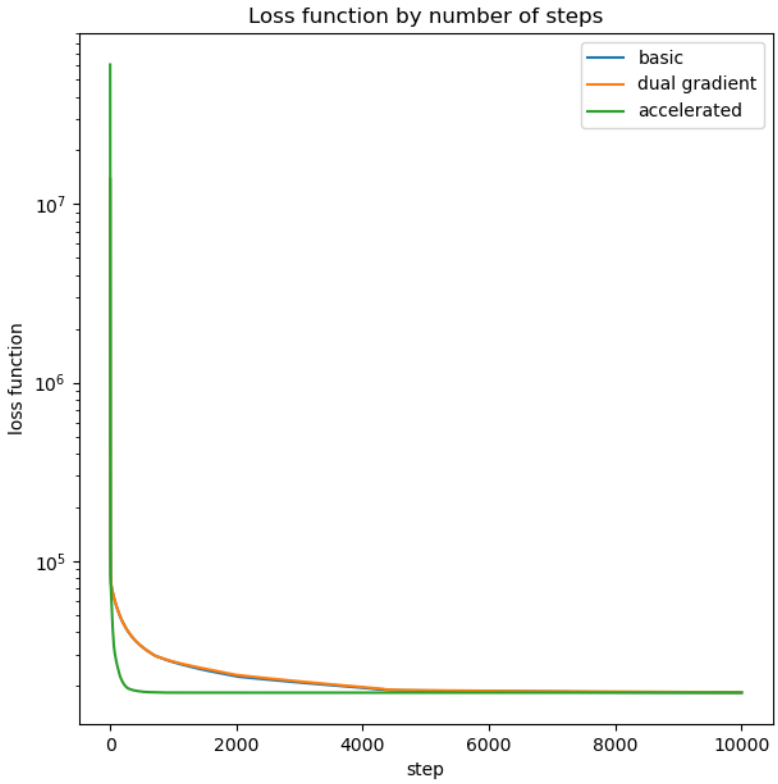

Figure 10: Change of the loss function values

This time, all three methods converged in 10000 steps, proving that changing λ to a higher value notably lowered the difficulty of the problem. Nonetheless, the relations between methods' performance remained almost the same.

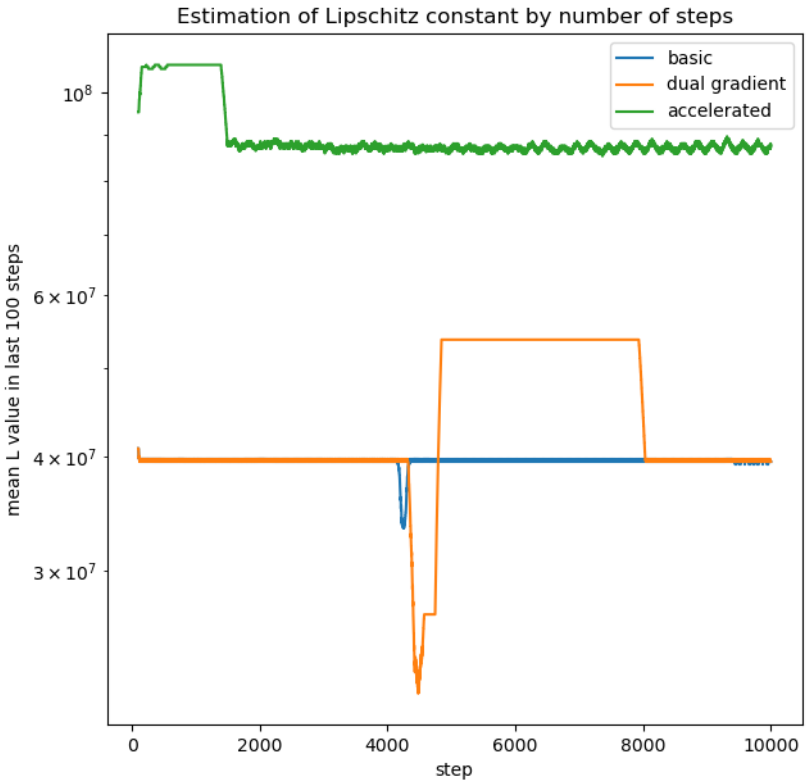

Figure 11: Estimations of Lipschitz constant

Figure 11 shows that despite their practically identical convergence ratio, runs of Basic Method and Dual Gradient Method differed in terms of estimated Lipschitz constant.

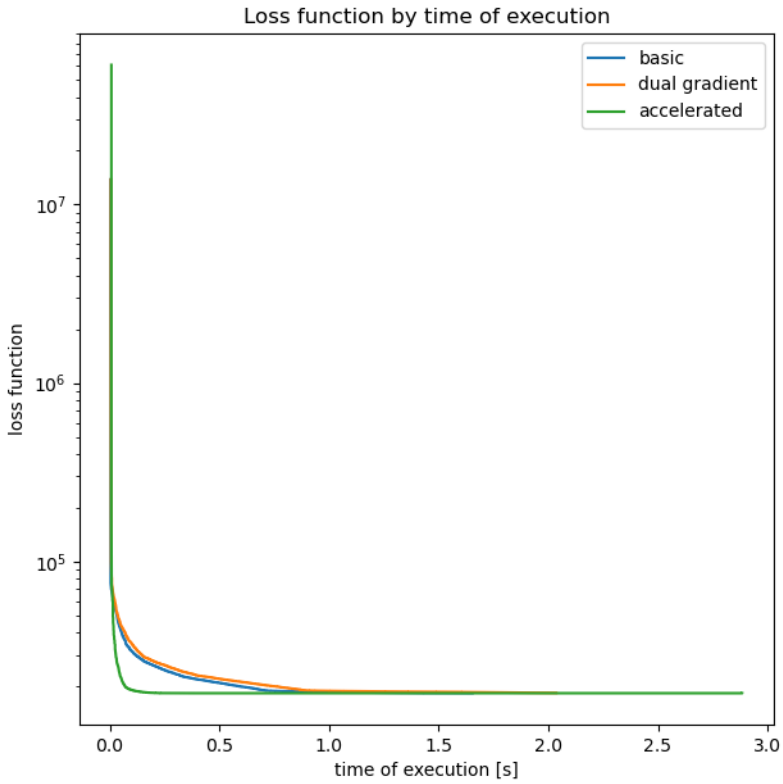

Figure 12: Performance in time

Although obtained final solutions were very differenct than for $\lambda = 50$, the characteristics of methods' execution in time remained almost the same.

6.2 Examination of γ_u parameter

For this part, we examined the impact of γ_u parameter value on the models' performance. All other parameters' values were left the same as in the previous section.

6.2.1 Generated data, $\lambda = 2$

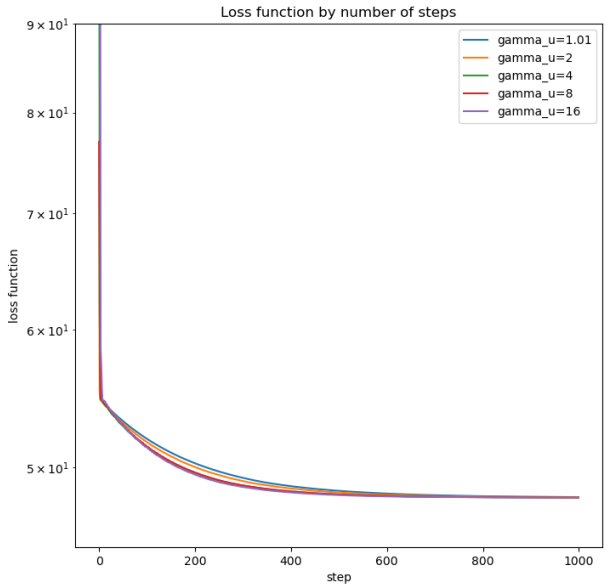

Figure 13: Impact of γ_u on Basic Method performance

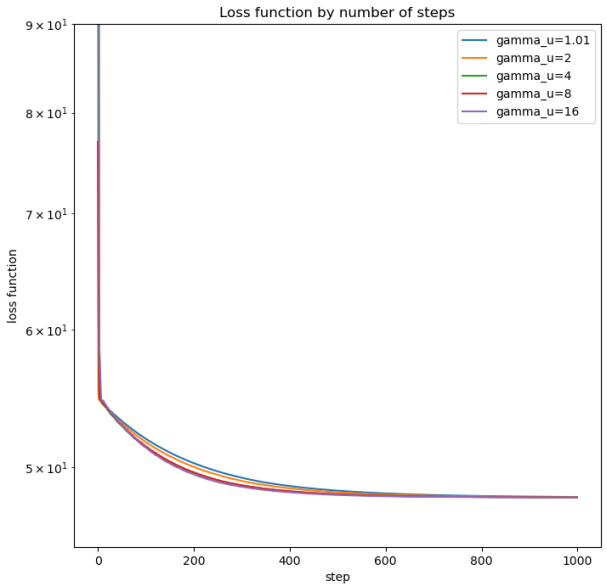

Figure 14: Impact of γ_u on Dual Gradient Method performance

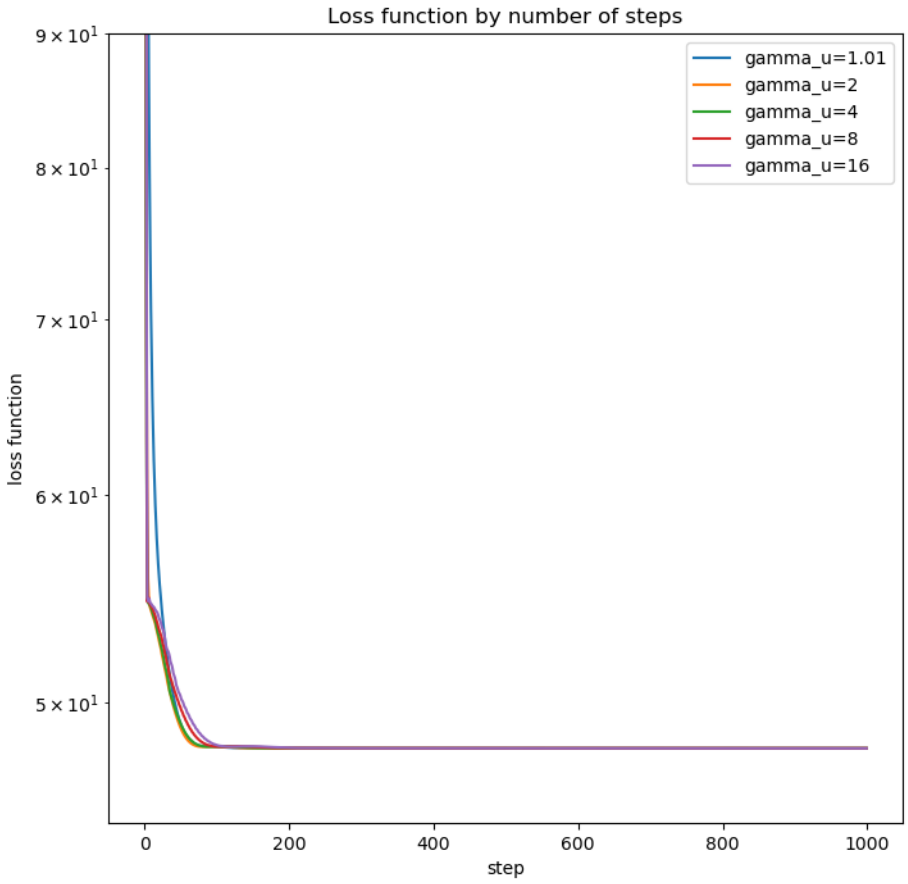

Figure 15: Impact of γ_u on Accelerated Method performance

Although it may be hard to see, the order of lines for Accelerated Method is inversed compared to the other two methods (except for the line for $\gamma_u = 1.01$ at its very beginning). However, changes in γ_u values did not yield any significant differences in any method's performance. Overall, the different raise speed of Lipschitz constant estimate was successfully compensated by the changing number of iterations in the main loop of each method (Gradient Iteration for Basic Method and Dual Gradient Method, and unnamed REPEAT in Accelerated Method). Figures below illustrate how the numbers of iterations in these loops depended on the γ_u parameter.

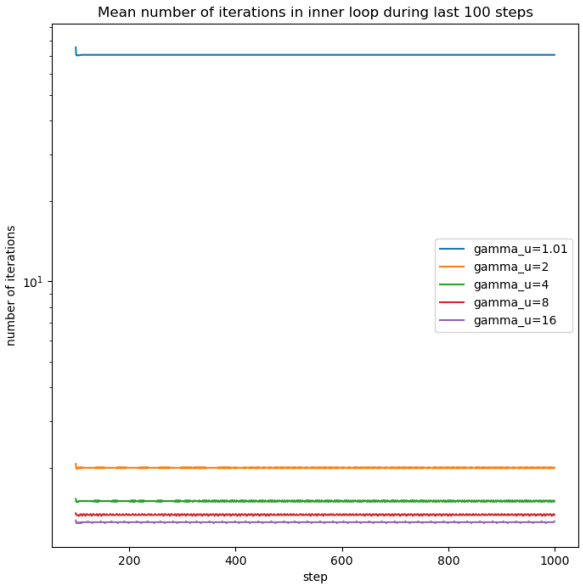

Figure 16: Impact of γ_u on number of loop iterations in Basic Method

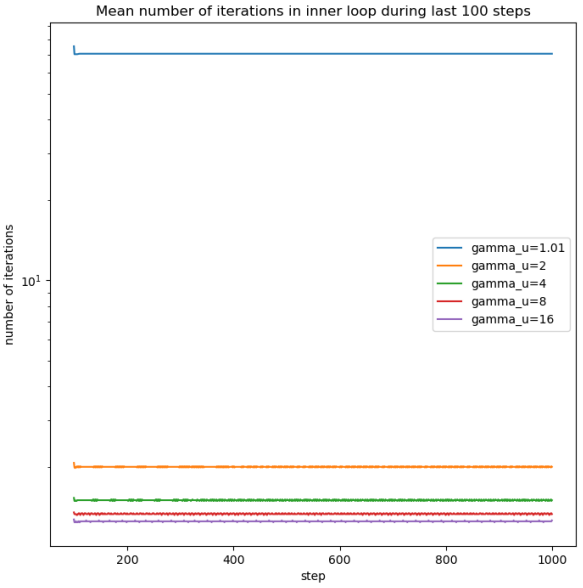

Figure 17: Impact of γ_u on number of loop iterations in Dual Gradient Method

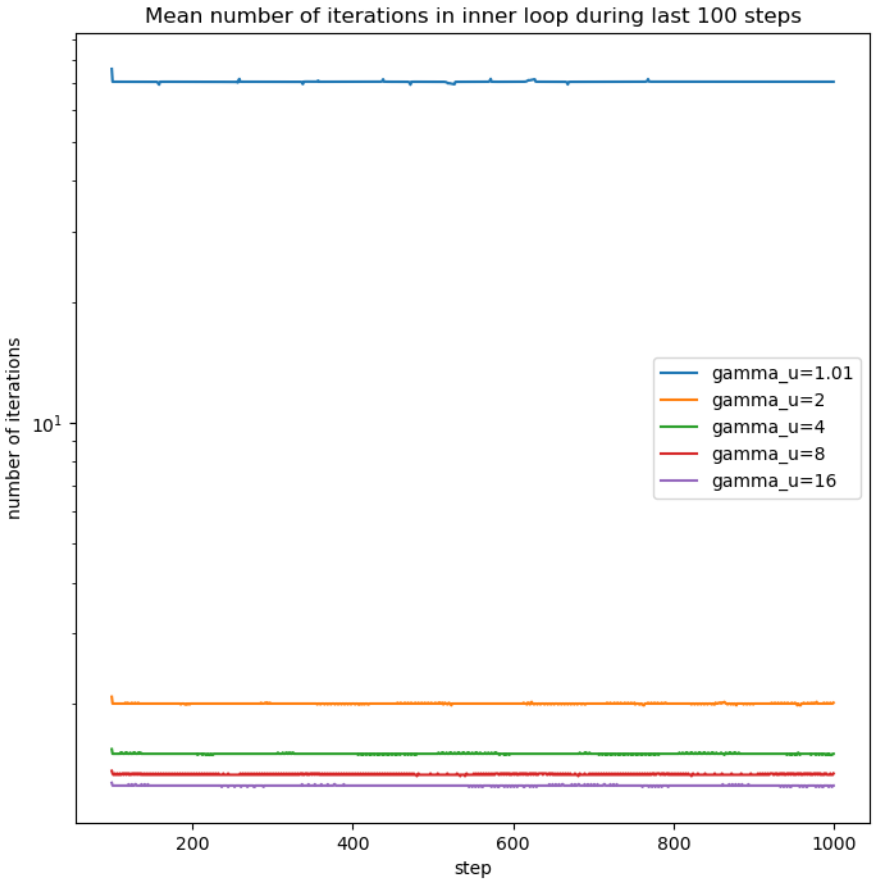

Figure 18: Impact of γ_u on number of loop iterations in Accelerated Method

The number of iterations of the main loop depended on γ_u and did not significantly change over time.

6.2.2 Generated data, $\lambda = 10$

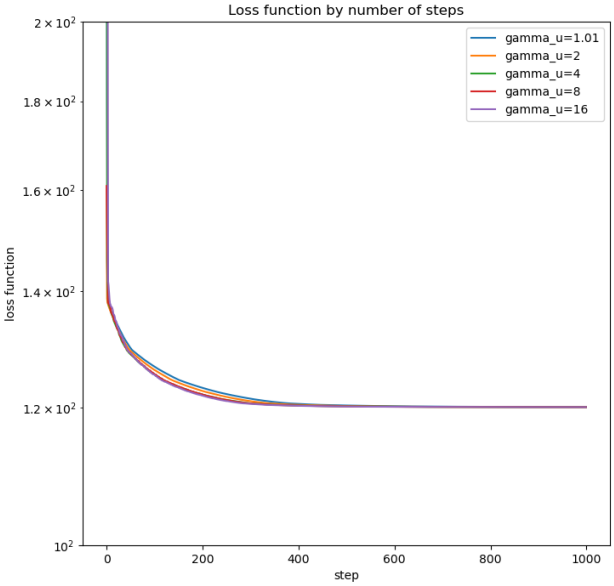

Figure 19: Impact of γ_u on Basic Method performance

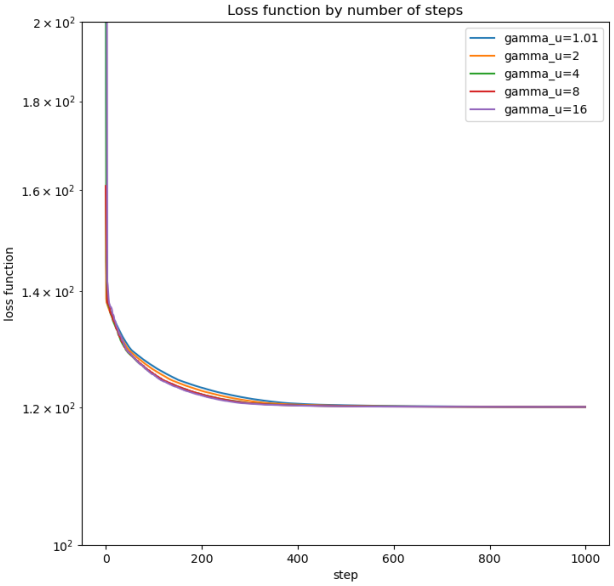

Figure 20: Impact of γ_u on Dual Gradient Method performance

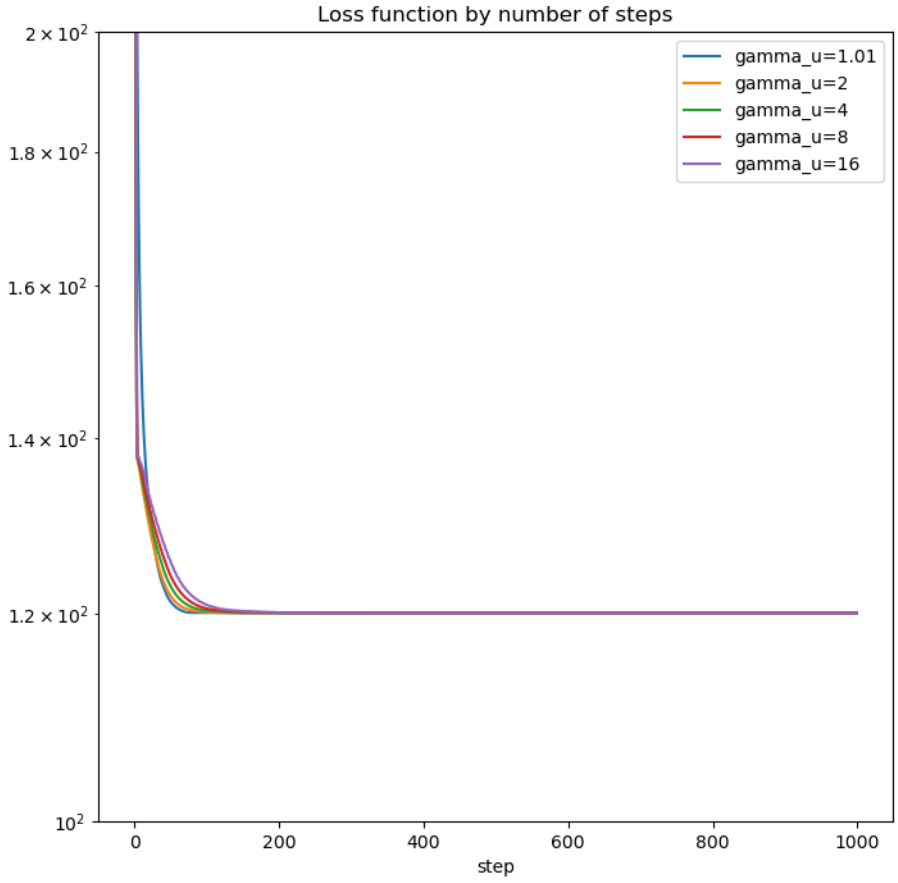

Figure 21: Impact of γ_u on Accelerated Method performance

The results from this section were quite similar to the results when $\lambda = 2$. Again, please note the inversed order of lines in Accelerated Method compared to other two methods. Numbers of main loop iterations for $\lambda = 10$ with the generated dataset looked almost exactly the same both in terms of shape and absolute values. Because of that, we decided not to show them here. Instead, in the next paragraph, we present results of iteration measurements for the real dataset.

6.2.3 Real data

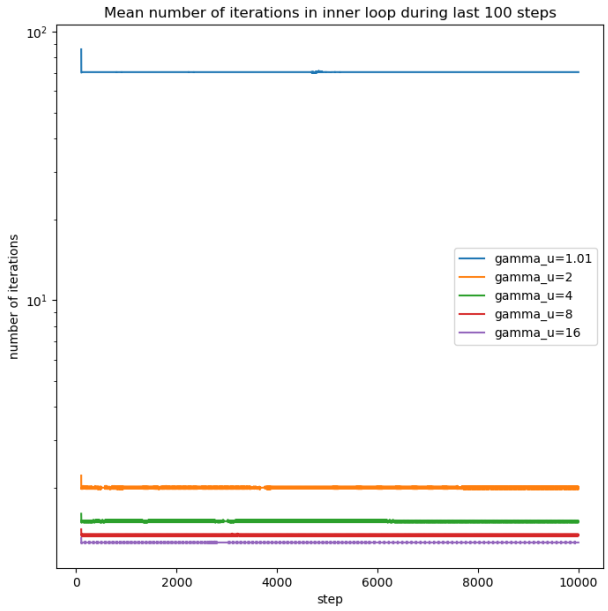

Figure 22: Impact of γ_u on number of loops in Basic Method, $\lambda = 5000$

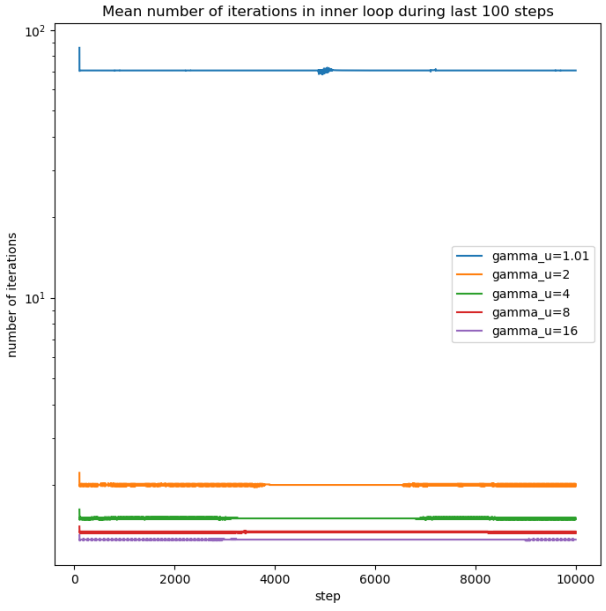

Figure 23: Impact of γ_u on number of loops in Dual Gradient Method, $\lambda = 5000$

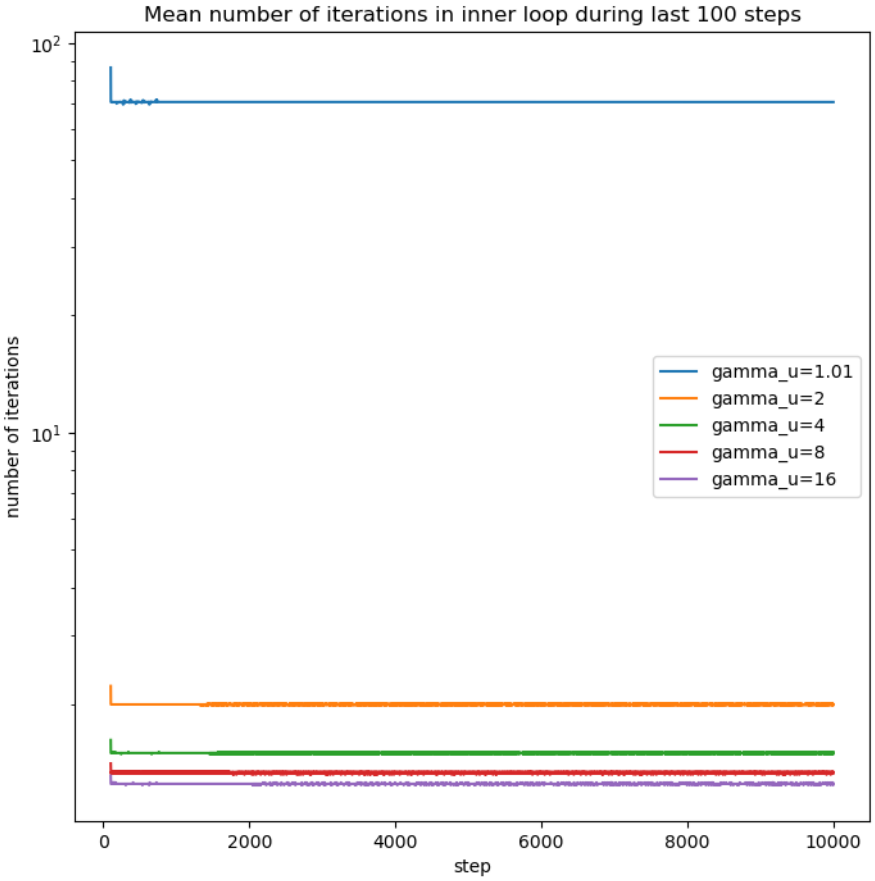

Figure 24: Impact of γ_u on number of loops in Accelerated Method, $\lambda = 5000$

While the number of iterations in main loops for different values of γ_u were different than with the generated data, their proportional relationships remained the same. Plots for $\lambda = 50$ were very similar (except for the number of steps) and we decided not to present them here.

For both values of λ, the convergence performance results were very similar to those presented for the generated data, perhaps with one notable exception: $\gamma_u = 1.01$ hidnered convergence of Basic Method and Dual Gradient Method to a slightly greater extent.

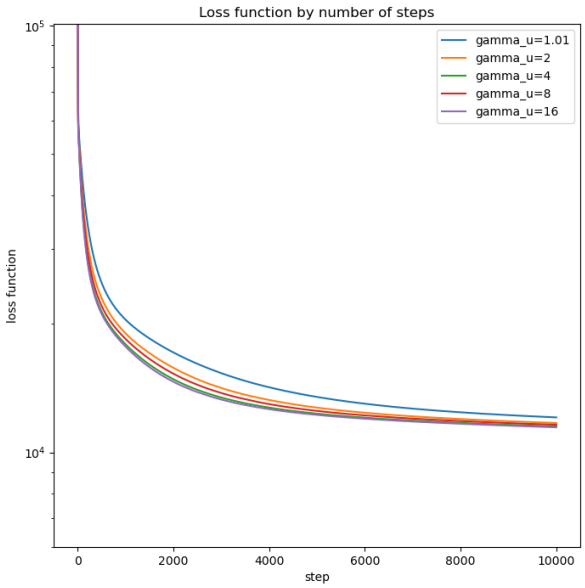

Figure 25: Impact of γ_u on Basic Method performance, $\lambda = 50$

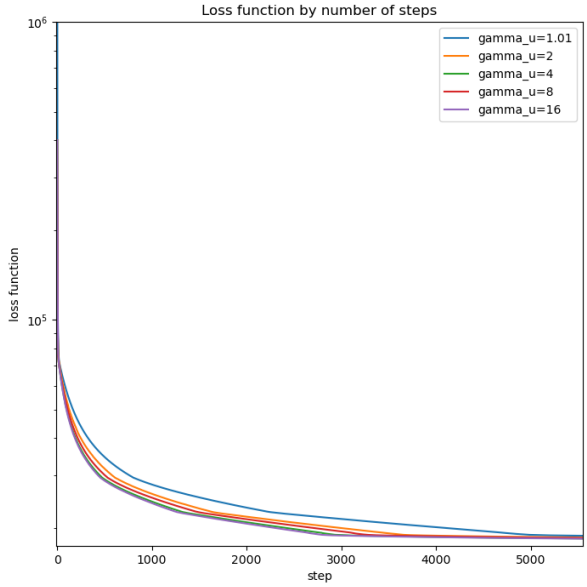

Figure 26: Impact of γ_u on Basic Method performance, $\lambda = 5000$

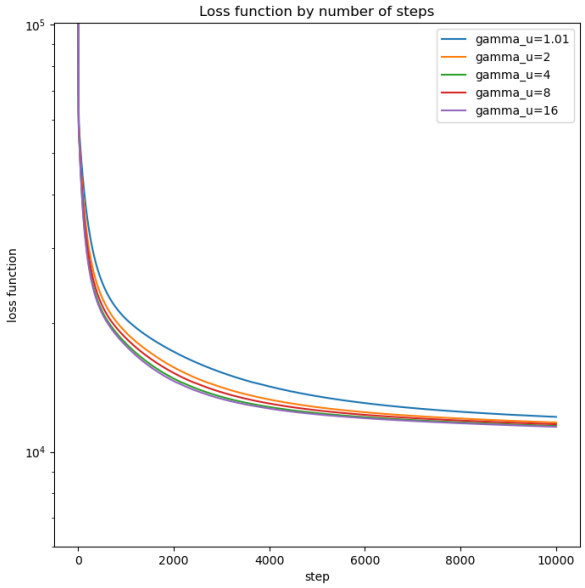

Figure 27: Impact of γ_u on Dual Gradient Method performance, $\lambda = 50$

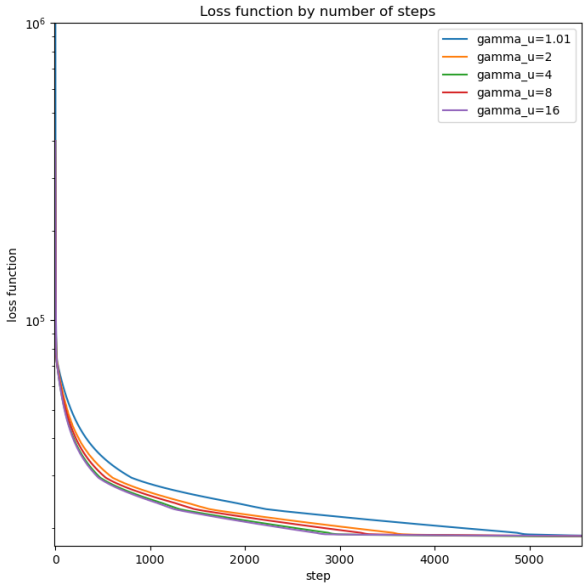

Figure 28: Impact of γ_u on Dual Gradient Method performance, $\lambda = 5000$

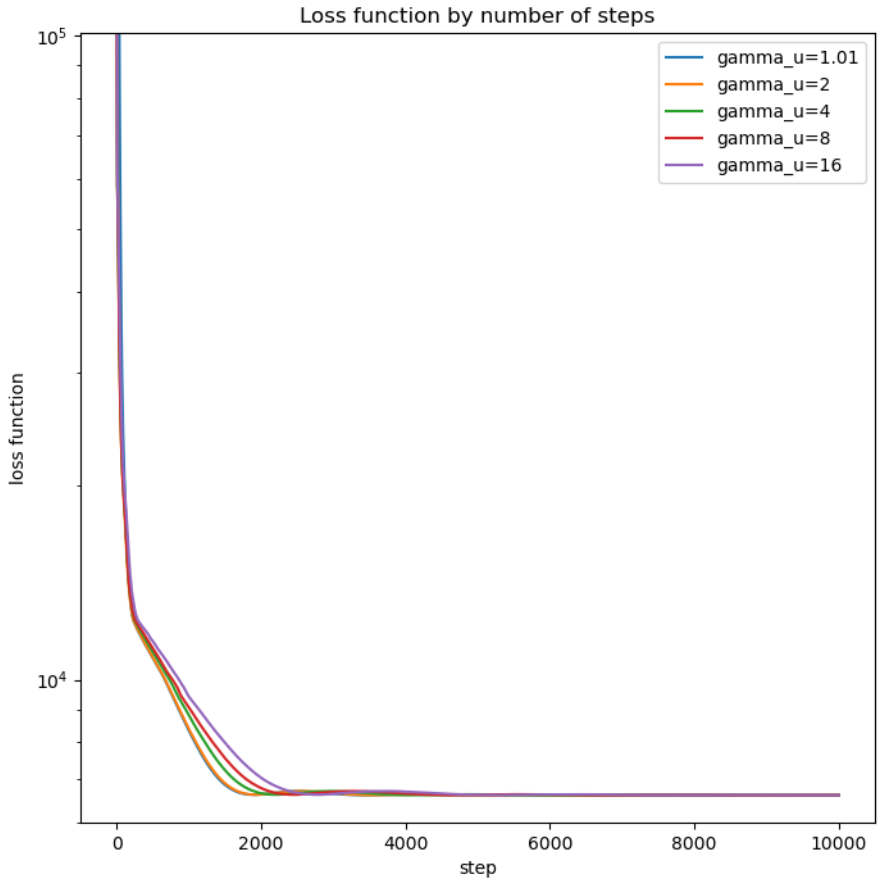

Figure 29: Impact of γ_u on Accelerated Method performance, $\lambda = 50$

The best convergence ratio for Accelerated Method was once again observed for $\gamma_u = 1.01$, but its advantage over $\gamma_u = 2$ was hardly noticeable.

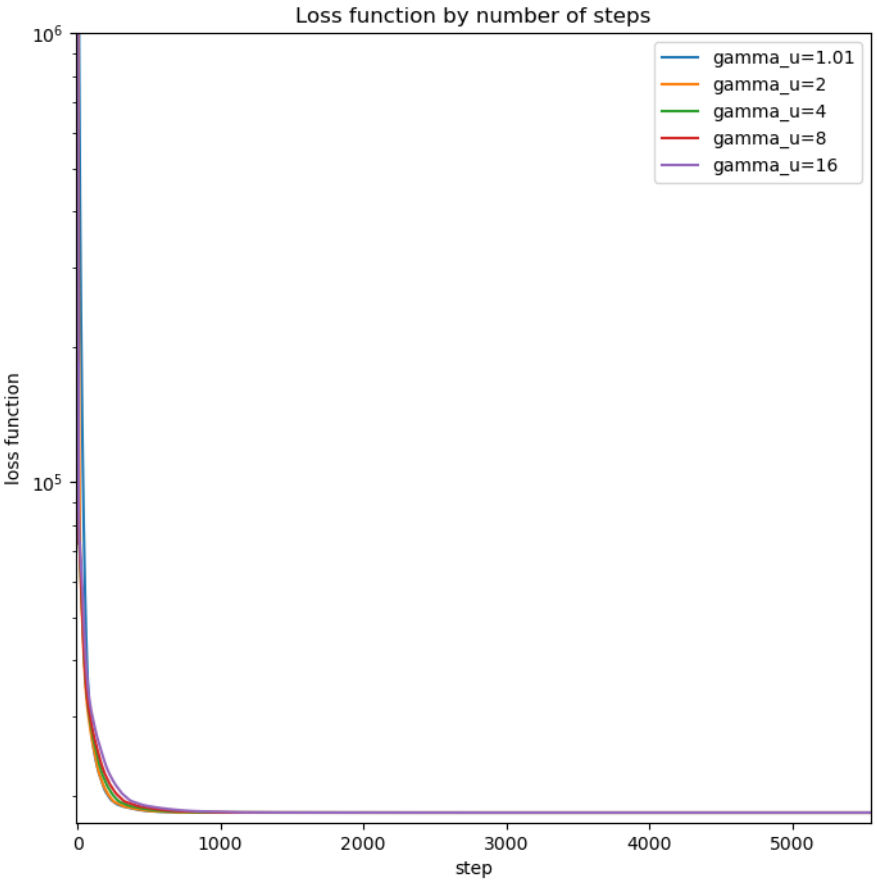

Figure 30: Impact of γ_u on Accelerated Method performance, $\lambda = 5000$

It seems like Accelerated Method benefited from lower γ_u values, while Basic Method and Dual Gradient Method seemed to perform better when γ_u was higher. Note, that this is only in terms of convergence in number of steps; to minimize the time of execution, one must take into consideration the number of iterations in the main loop, and this decreases when γ_u gets higher. In fact, Accelerated Method with $\gamma_u = 1.01$ was disastrously slow because of the number of main loop iterations, and thus it could not be applied in practice.

7 Conclusions

1. Accelerated Method were always the fastest to converge, both in terms of number of steps and computation time.
2. Basic Method and Dual Gradient Method always converged in the same pace when it comes to the number of steps.
3. Basic Method execution was slightly faster, causing a very small lead over Dual Gradient Method in convergence time.
4. In all tests, all three methods converged to the same final results.
5. The way methods estimated the Lipschitz constant differed throughout the problems. Accelerated Method tended to use higher estimations than the other two methods.
6. Values of γ_u did not have a significant impact on the models' performance.
7. Values of γ_u had a huge impact on the models' execution time. Raising γ_u was causing the execution time to drop significantly.