# Stochastic gradient algorithms in continuous time as ODE splitting

**Daniil Merkulov**
Skolkovo Institute of Science and Technology
daniil.merkulov@skolkovotech.ru

**Ivan Oseledets**
Skolkovo Institute of Science and Technology
i.oseledets@skoltech.ru

## Abstract

We present a different view on stochastic optimization, which goes back to the splitting schemes for approximate solutions of ODE. In this work, we provide a connection between stochastic gradient descent approach and first-order splitting scheme for ODE. We consider the special case of splitting, which is inspired by machine learning applications and derive a new upper bound on the global splitting error for it. We present, that the Kaczmarz method is the limit case of the splitting scheme for the unit batch SGD for linear least squares problem. We support our findings with systematic empirical studies, which demonstrates, that a more accurate solution of local problems leads to the stepsize robustness and provides better convergence in time and iterations on the softmax regression problem.

## 1 Introduction

A lot of practical problems arising in machine learning require minimization of a finite sample average which can be written in the form

$$f(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} f_i(\boldsymbol{\theta}) \to \min_{\boldsymbol{\theta}\in\mathbb{R}^p}, \tag{1}$$

where the sum goes over the *minibatches* of the original dataset. Vanilla stochastic gradient descent (SGD) method Robbins and Monro [1951] consists sequential steps in the direction of the gradient of $f_i(\boldsymbol{\theta})$, where $i$ is to be chosen randomly from $1$ to $n$ without replacement.

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - h_k \nabla f_i. \tag{2}$$

Gradient descent method Cauchy [1847] can be considered as an Euler discretization of the ordinary differential equation (ODE) of the form of the gradient flow

$$\frac{d\boldsymbol{\theta}}{dt} = -\nabla f(\boldsymbol{\theta}). \tag{3}$$

In continuous time, SGD if often analyzed by introducing noise into the right-hand side of (3). However, for a real dataset, the distribution of the noise obtained by replacing the full gradient by its minibatch variant is not known and can be different for different problems. Instead, we propose a new view on the SGD as a *first-order splitting scheme* for (3), thus shedding a new light on SGD-type algorithms. This representation allows using more efficient local problem solvers for the approximation of the full gradient flow.

**Contributions**

- We show, that vanilla SGD could be considered as a splitting scheme for a full gradient flow and highlight connection between learning rate, batch size and size of the approximation step of SGD in continuous time.

Preprint. Under review.

- We propose new optimization scheme, which uses numerical integration of simple ODE at each step instead of stochastic gradient calculation and show empirically, that such approach can be considered as a stepsize-robust alternative to SGD for some practical ML problems.
- We present, that the Kaczmarz method is the limit case of the splitting scheme for the unit batch SGD for linear least squares problem.

## 2 SGD as a splitting scheme

We firstly consider simple ODE, where we can apply splitting idea and corresponding minimization problem. The best example to start from is simple ODE with right-hand-side, consisting of two summands:

$$\frac{d\boldsymbol{\theta}}{dt} = -\frac{1}{2}\left(g_1(\boldsymbol{\theta}) + g_2(\boldsymbol{\theta})\right) \tag{4}$$

Suppose, we want to find the solution $\boldsymbol{\theta}(h)$ of (4) via integrating it on the small timestep $h$. The first order splitting scheme defined by solving first $\frac{d\boldsymbol{\theta}}{dt} = -\frac{1}{2}g_1(\boldsymbol{\theta}), \quad \boldsymbol{\theta}(0) = \boldsymbol{\theta}_0$ with exact solution $\boldsymbol{\theta}_1(h)$ at the moment $h$, followed by $\frac{d\boldsymbol{\theta}}{dt} = -\frac{1}{2}g_2(\boldsymbol{\theta}), \quad \boldsymbol{\theta}(0) = \boldsymbol{\theta}_1(h)$ with exact solution $\boldsymbol{\theta}_2(h)$ at the moment $h$. Thus, the first order approximation could be written as a combinations of both solutions $\boldsymbol{\theta}^I(h) = \boldsymbol{\theta}_2(h) \circ \boldsymbol{\theta}_1(h) \circ \boldsymbol{\theta}_0$.

It is interesting to study how the pure splitting scheme Marchuk [1968], Strang [1968] corresponds to the SGD approach. For this purpose, we consider an illustrative example of Gradient Flow equation 5, where the right-hand side of ODE is just the sum of operators acting on $\boldsymbol{\theta}$, which allows us to apply splitting scheme approximation directly.

$$\frac{d\boldsymbol{\theta}}{dt} = -\frac{1}{2}\sum_{i=1}^{2}\nabla f_i(\boldsymbol{\theta}) = -\frac{1}{2}\nabla f_1(\boldsymbol{\theta}) - \frac{1}{2}\nabla f_2(\boldsymbol{\theta}) \tag{5}$$

Table 1: The table describes the correspondence between splitting scheme for discretized Gradient Flow ODE and epoch of SGD

| Splitting step | Euler discretization | SGD Epoch | First-order splitting |
|---|---|---|---|
| $\frac{d\boldsymbol{\theta}}{dt} = -\frac{1}{2}\nabla f_1(\boldsymbol{\theta})$ | $\tilde{\boldsymbol{\theta}}_I = \boldsymbol{\theta}_0 - \frac{h}{2}\nabla f_1(\boldsymbol{\theta}_0)$ | $\tilde{\boldsymbol{\theta}}_{SGD} = \boldsymbol{\theta}_0 - h\nabla f_1(\boldsymbol{\theta}_0)$ | $\tilde{\boldsymbol{\theta}}_I = \boldsymbol{\theta}_0 - \frac{h}{2}\nabla f_1(\boldsymbol{\theta}_0)$ |
| $\frac{d\boldsymbol{\theta}}{dt} = -\frac{1}{2}\nabla f_2(\boldsymbol{\theta})$ | $\boldsymbol{\theta}_I = \tilde{\boldsymbol{\theta}}_I - \frac{h}{2}\nabla f_2(\tilde{\boldsymbol{\theta}}_I)$ | $\boldsymbol{\theta}_{SGD} = \tilde{\boldsymbol{\theta}}_{SGD} - h\nabla f_2(\tilde{\boldsymbol{\theta}}_{SGD})$ | $\boldsymbol{\theta}_I = \tilde{\boldsymbol{\theta}}_I - \frac{h}{2}\nabla f_2(\tilde{\boldsymbol{\theta}}_I)$ |

Thus, we can conclude, that *one epoch of SGD is just the splitting scheme for the discretized Gradient Flow ODE with $2 \cdot h$ step size ($m \cdot h$ in case of $m$ batches)*

Indeed, in SGD we go in the direction of the batch gradient, which stands for the Euler discretization of batch gradient flow ODE or *local ODE*. This idea gives additional intuition on the method. Given information about the Euler scheme limitation (first-order accuracy, stability issues), we propose to solve each local problem more precisely.

## 3 Optimization step with ODE solver

We propose to integrate local problem more precisely instead of Euler step in SGD. Solution of the local ODE problem involves replacing gradient in the right-hand side of gradient flow ODE 4 with batch gradient version. In our experiments the explicit Runge-Kutta method Dormand and Prince [1980], Shampine [1986] was used via scipy Virtanen et al. [2020] function odeint.

Typical machine learning problems involves dealing with mini-batch of size $b$, which is often less, than the number of trainable parameters $p$, which allows us to reduce dimensionality of the dynamic system via $QR$ decomposition of each batch data matrix $X_i^\top = Q_i R_i$ (see details in the Appendix) and substitution $\boldsymbol{\eta}_i = Q_i^\top \boldsymbol{\theta}$. Note, that $QR$ decomposition is only needed to be performed once before the training.

There is an analytical solution for each local ODE in linear least squares case:

Table 2: The table presents ODE, which we need to solve at each step of the algorithm. The last column shows the ODE, which is needed to be solved at each iteration of the algorithm for each given problem.

| Problem | Loss function | Batch gradient | Initial local ODE |
|---|---|---|---|
| Linear Least Squares | $f(\boldsymbol{\theta}) = \frac{1}{n}\sum\limits_{i=1}^{m} \|X_i\boldsymbol{\theta} - \mathbf{y_i}\|_2^2$ | $\frac{1}{b}X_i^\top(X_i\boldsymbol{\theta} - \mathbf{y_i})$ | $\frac{d\boldsymbol{\theta}}{dt} = -\frac{1}{n}X_i^\top(X_i\boldsymbol{\theta} - \mathbf{y_i})$ |
| Binary logistic regression | $f(\boldsymbol{\theta}) = -\frac{1}{n}\sum\limits_{i=1}^{n}\Big(y_i \ln\sigma(\boldsymbol{\theta}^\top \mathbf{x_i}) + +(1-y_i)\ln\Big(1-\sigma(\boldsymbol{\theta}^\top\mathbf{x_i})\Big)\Big)$ | $\frac{1}{b}X_i^\top\left(\sigma\left(X_i\boldsymbol{\theta}\right) - \mathbf{y_i}\right)$ | $\frac{d\boldsymbol{\theta}}{dt} = -\frac{1}{n}X_i^\top\left(\sigma\left(X_i\boldsymbol{\theta}\right) - \mathbf{y_i}\right)$ |
| One FC Layer + softmax | $f(\Theta) = -\frac{1}{n}\sum\limits_{i=1}^{n}\log\left(\frac{\mathbf{y_i}^\top e^{\Theta^\top \mathbf{x_i}}}{\mathbf{1}^\top e^{\Theta^\top \mathbf{x_i}}}\right)$ | $\frac{1}{b}X_i^\top\left(s(\Theta^\top X_i^\top) - Y_i\right)^\top$ | $\frac{d\Theta}{dt} = -\frac{1}{n}X_i^\top\left(s(\Theta^\top X_i^\top) - Y_i\right)^\top$ |

**Algorithm 1:** Splitting optimization

```
$\boldsymbol{\theta}_0$ - initial parameter; $b$ - batch size; $\alpha$ - learning rate; $m$- total number of batches
$h := \alpha m$
$t := 0$
for $k = 0, 1, \dots$ do
    for $i = 1, 2, \dots, m$ do
        Formulate local ODE problem $\mathcal{P}_i^k$
        $\boldsymbol{\theta}_{t+1}$ = integrate $\mathcal{P}_i^k$ given an initial value $\boldsymbol{\theta}(0) = \boldsymbol{\theta}_t$ to the step h
        $t := t + 1$
    end
end
```

Table 3: The table shows initial local ODE and paired $\mathcal{P}_i^k$. Note, that $\boldsymbol{\eta}_i \in \mathbb{R}^b$ , while $\boldsymbol{\theta} \in \mathbb{R}^p$

| Initial local ODE | $\mathcal{P}_i^k$ | Integration |
|---|---|---|
| $\frac{d\boldsymbol{\theta}}{dt} = -\frac{1}{n}X_i^\top(X_i\boldsymbol{\theta} - \mathbf{y_i})$ | $\frac{d\boldsymbol{\eta}_\mathbf{i}}{dt} = -\frac{1}{n}R_i\left(R_i^\top\boldsymbol{\eta}_\mathbf{i} - \mathbf{y_i}\right), \boldsymbol{\eta}_\mathbf{i} = Q_i^\top\boldsymbol{\theta}$ | analytical |
| $\frac{d\boldsymbol{\theta}}{dt} = -\frac{1}{n}X_i^\top\left(\sigma\left(X_i\boldsymbol{\theta}\right) - \mathbf{y_i}\right)$ | $\frac{d\boldsymbol{\eta}_\mathbf{i}}{dt} = -\frac{1}{n}R_i\left(\sigma\left(R_i^\top\boldsymbol{\eta}_\mathbf{i}\right) - \mathbf{y_i}\right), \boldsymbol{\eta}_\mathbf{i} = Q_i^\top\boldsymbol{\theta}$ | `odeint` |
| $\frac{d\Theta}{dt} = -\frac{1}{n}X_i^\top\left(s(\Theta^\top X_i^\top) - Y_i\right)^\top$ | $\frac{dH_i}{dt} = -\frac{1}{n}R_i(s(H_i^\top R) - Y_i)^\top, H_i = Q_i^\top\Theta$ | `odeint` |

**Theorem 1.** *For any matrix $\mathbf{x_i} \in \mathbb{R}^{b\times p}, b \leq p, rank X_i = b$, any vector of right-hand side $\mathbf{y_i} \in \mathbb{R}^b$ and initial vector of parameters $\boldsymbol{\theta}_0$, there is a solution of the $\frac{d\boldsymbol{\theta}}{dt} = -\frac{1}{n}X_i^\top(X_i\boldsymbol{\theta} - \mathbf{y_i})$, given by formula:*

$$\boldsymbol{\theta}(h) = Q_i e^{-\frac{1}{n}R_i R_i^\top h}\left(Q_i^\top\boldsymbol{\theta}_0 - R_i^{-\top}\mathbf{y_i}\right) + Q_i R_i^{-\top}\mathbf{y_i} + (I - Q_i Q_i^\top)\boldsymbol{\theta}_0, \tag{6}$$

*where $Q_i \in \mathbb{R}^{p\times b}$ and $R_i \in \mathbb{R}^{b\times b}$ stands for the $QR$ decomposition of the matrix $\mathbf{X_i}^\top$, $\mathbf{X_i}^\top = Q_i R_i$.*

It is interesting to mention, that the splitting approach immediately leads to the Kaczmarz Kaczmarz. [1937], Strohmer and Vershynin [2009], Gower and Richtárik [2015] method for solving linear system in the same setting with unit batch size.

$$\lim_{h\to\infty}\boldsymbol{\theta}(h) = \frac{\left(y_i - \mathbf{x_i}^\top\boldsymbol{\theta}_0\right)}{\|\mathbf{x_i}\|^2}\mathbf{x_i} + \boldsymbol{\theta}_0, \tag{7}$$

which is exact formula for Kaczmarz method for solving linear system. This result correlates with the statements of Needell et al. [2014], but provides us with a new sense of similarity between SGD and Kaczmarz method.

## 4 Results

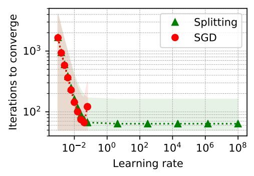
In this section, we describe the experimental setting. The majority of computations were performed on the NVIDIA DGX-2 cluster with 80 CPUs and 512 Gb RAM. We restricted the number of CPU usage per each experiment with an upper limit of 5 CPUs per experiment. All time measurements were done with the time library for Python. All experiments were done with the fixed random seed for reproducibility. For each experiment we performed 30 runs with random initialization and plotted trend line with the standard deviation.

**Linear Least Squares** Both random and the real linear systems were tested. For random linear system (`random lls`) we generated $10000 \times 500$ matrix with additive Gaussian noise of magnitude $0.01$. Presented figures correspond to the batch size equals to 20. The real linear system (`tom lls`) is the standard tomography data from AIRTools II Hansen and Jørgensen [2018]. Solution of the linear system is the $50 \times 50$ image reconstructed from solving $12780 \times 2500$ linear system. Presented figures correspond to the batch size equals to 60. Relative error $10^{-3}$ was used as the stopping criterion.

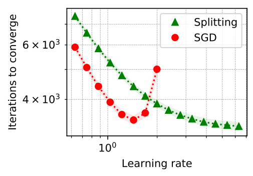
**Binary Logistic Regression** (`logreg`) In our experiments we used two classes from MNIST LeCun et al. [1998] dataset, which corresponds to the 0 and 1 digits. The size of the batch for presented figure is 50. Test error 0.001 was used as the stopping criterion.

**Softmax Logistic Regression** (`softmax`) We took Fashion MNIST Xiao et al. [2017] dataset with 60000 grayscale pictures from 10 classes. Each example is $28 \times 28$ image. The size of the batch for presented figure is 64. Test error 0.25 was used as the stopping criterion.

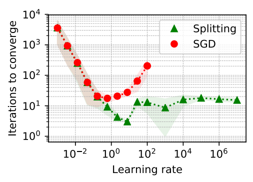
On the figures below we have two labels: `SGD` and `Splitting`, which stands for batch stochastic gradient descent and proposed algorithm. We use different constant learning rates to perform our experiments. All the learning rates tested for both algorithms. Lack of point of one algorithm on the graph means reaching the limit of iterations without achieving the termination rule.



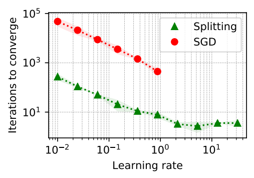
(a) `Random LLS`



(b) `Tom LLS`

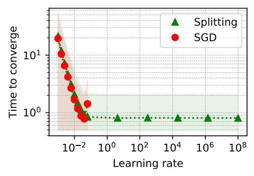


(c) `LogReg`

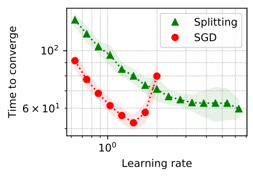


(d) `Softmax`

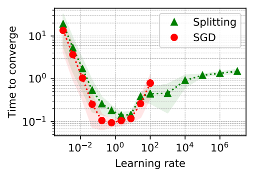
As it is expected, SGD diverges starting from some value of learning rate, which is specific for each problem. While we can see comparative robustness of the proposed splitting optimization approach.



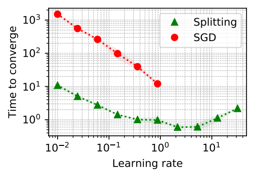
(a) `Random LLS`



(b) `Tom LLS`



(c) `LogReg`



(d) `Softmax`

## 5 Related work

In this work, we presented another point of view on the nature of stochasticity in the stochastic gradient algorithms. From this perspective, different splitting schemes yield different stochastic gradient algorithms. We focused on the first-order splitting scheme for ODE, which corresponds to the SGD with the constant learning rate. Given this tractable setting, we performed a systematic empirical study of the local problem integration influence on the quality of the approximation scheme in machine learning problems. While the question of using these ideas to make general-purpose optimizer remains open, splitting optimization approach showed itself quite robust to the hyperparameter tuning for particular practical problems. Appendix to the paper contains proofs of the theorems and a new global error upper bounds for the first-order splitting for the special case. In Su et al. [2014] authors introduced second order ODE, which is equivalent (in the limit sense) to the gradient descent with Nesterov momentum Nesterov [1983]. Generalization of these ideas was presented in Wibisono et al. [2016] with an arbitrary polynomial acceleration using the same parameter in ODE. General overview of the interplay between continuous-time and discrete-time points of view on dynamical systems and iterative optimization methods is covered in Helmke and Moore [2012], Evtushenko and Zhadan [1994]

## References

A. Cauchy. M'ethode g'en'erale pour la r'esolution des systemes d''equations simultan'ees. *Comp. Rend. Sci. Paris*, 25(1847):536–538, 1847.

J. R. Dormand and P. J. Prince. A family of embedded runge-kutta formulae. *Journal of computational and applied mathematics*, 6(1):19–26, 1980.

Y. G. Evtushenko and V. G. Zhadan. Stable barrier-projection and barrier-newton methods in linear programming. *Computational Optimization and Applications*, 3(4):289–303, 1994.

R. M. Gower and P. Richtárik. Randomized iterative methods for linear systems. *SIAM Journal on Matrix Analysis and Applications*, 36(4):1660–1690, 2015.

P. C. Hansen and J. S. Jørgensen. Air tools ii: algebraic iterative reconstruction methods, improved implementation. *Numerical Algorithms*, 79(1):107–137, 2018.

U. Helmke and J. B. Moore. *Optimization and dynamical systems*. Springer Science & Business Media, 2012.

S. Kaczmarz. Angenäherte auflösung von systemen linearer gleichungen. *Bull. Internat. Acad. Polon.Sci. Lettres A*, pages 335–357, 1937.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

G. I. Marchuk. Some application of splitting-up methods to the solution of mathematical physics problems. *Aplikace matematiky*, 13(2):103–132, 1968.

D. Needell, R. Ward, and N. Srebro. Stochastic gradient descent, weighted sampling, and the randomized kaczmarz algorithm. In *Advances in neural information processing systems*, pages 1017–1025, 2014.

Y. E. Nesterov. A method of solving a convex programming problem with convergence rate $o(k^2)$. In *Doklady Akademii Nauk*, volume 269, pages 543–547. Russian Academy of Sciences, 1983.

H. Robbins and S. Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

L. F. Shampine. Some practical runge-kutta formulas. *Mathematics of computation*, 46(173):135–150, 1986.

Q. Sheng. Global error estimates for exponential splitting. *IMA Journal of Numerical Analysis*, 14(1):27–56, 1994.

G. Strang. On the construction and comparison of difference schemes. *SIAM Journal on Numerical Analysis*, 5 (3):506–517, 1968.

T. Strohmer and R. Vershynin. A randomized kaczmarz algorithm with exponential convergence. *Journal of Fourier Analysis and Applications*, 15(2):262, 2009.

W. Su, S. Boyd, and E. Candes. A differential equation for modeling nesterov's accelerated gradient method: Theory and insights. In *Advances in Neural Information Processing Systems*, pages 2510–2518, 2014.

P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. Jarrod Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. Carey, İ. Polat, Y. Feng, E. W. Moore, J. Vand erPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and S. . . Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: https://doi.org/10.1038/s41592-019-0686-2.

A. Wibisono, A. C. Wilson, and M. I. Jordan. A variational perspective on accelerated methods in optimization. *proceedings of the National Academy of Sciences*, 113(47):E7351–E7358, 2016.

H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

## A Upper bound on the global splitting error

Suppose, that we have only two batches, and the problem (17) is consistent, i.e. there exists an exact solution $\boldsymbol{\theta}_*$ such as $X\boldsymbol{\theta}_* = \mathbf{y}$. The GD flow has the form

$$\begin{aligned} \frac{d\boldsymbol{\theta}}{dt} &= -X^\top(X\boldsymbol{\theta} - \mathbf{y}) = -X^\top X(\boldsymbol{\theta} - \boldsymbol{\theta}_*) = \\ &= -(X_1^\top X_1 + X_2^\top X_2)(\boldsymbol{\theta} - \boldsymbol{\theta}_*), \end{aligned} \tag{8}$$

i.e. the splitting scheme corresponds to a linear operator splitting

$$A = A_1 + A_2,\ A = -X^\top X,\ A_i = -X_i^\top X_i,\ i = 1, 2.$$

Both $A_1$ and $A_2$ are symmetric non-negative definite matrices. Without loss of generality, we can assume that $\boldsymbol{\theta}_* = 0$,

Suppose that the rank of $A$ is $r_1$ and the rank of $A_2$ is $r_2$. Then, we can write them as

$$A_i = Q_i B_i Q_i^*,$$

where $Q_i$ is an $N \times r_i$ matrix with orthonormal columns. The following Lemma gives the representation of the matrix exponents of such matrices.

**Lemma 1.** *Let $A = QBQ^*$, where $Q$ is an $N \times r$ matrix with orthonormal columns, and $B$ is an $r \times r$ matrix. Then,*

$$e^{tA} = (I - QQ^*) + Qe^{tB}Q^*. \tag{9}$$

To prove (15) we note that

$$\begin{aligned} e^{tA} &= \sum_{k=0}^{\infty} \frac{t^k A^k}{k!} = \sum_{k=0}^{\infty} \frac{t^k Q B^k Q^*}{k!} = \\ &= I - QQ^* + QQ^* + Q\sum_{k=1}^{\infty} \frac{t^k B^k}{k!} Q^* = \\ &= (I - QQ^*) + Qe^{tB}Q^*. \end{aligned}$$

**Lemma 2.** *Let $A_1, A_2 \in \mathbb{S}_+^p$ be the square negative semidefinite matrices, that don't have full rank, i.e. $\operatorname{rank} A_1 \leq p$ and $\operatorname{rank} A_2 \leq p$. While the sum of those matrices has full rank, i.e. $A = A_1 + A_2, \operatorname{rank} A = p$. Then, the global upper bound error will be written as follows:*

$$\lim_{t\to\infty} \|e^{A_2 t} e^{A_1 t} - e^{At}\| = \|(I - Q_2 Q_2^*)(I - Q_1 Q_1^*)\| \tag{10}$$

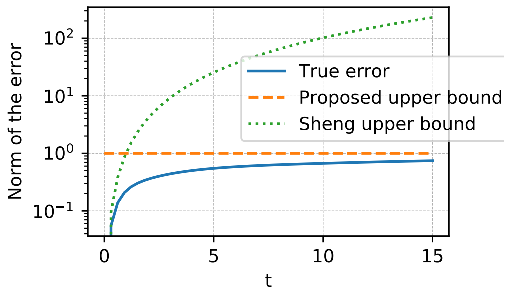
*Proof.* The proof is straightforward. We will use the low rank matrix exponential decomposition from the Lemma 3

$$e^{A_i t} = \Pi_i + Q_i e^{B_i t} Q_i^*, \text{where } \Pi_i = I - Q_i Q_i^*; i = 1, 2$$

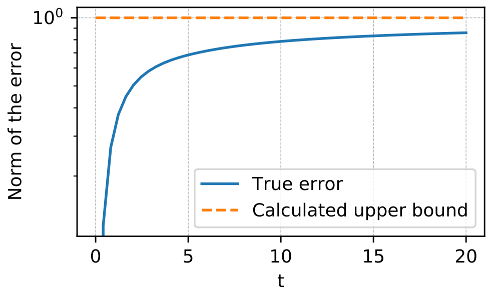
$$\begin{aligned} &\lim_{t\to\infty} \|e^{A_2 t} e^{A_1 t} - e^{At}\| = \\ &= \lim_{t\to\infty} \|(\Pi_2 + Q_2 e^{B_2 t} Q_2^*)(\Pi_1 + Q_1 e^{B_1 t} Q_1^*) - e^{At}\| = \\ &= \lim_{t\to\infty} \|\Pi_2\Pi_1 + Q_1 e^{B_1 t} Q_1^* \Pi_2 + \Pi_1 Q_2 e^{B_2 t} Q_2^* + \\ &+ Q_1 e^{B_1 t} Q_1^* Q_2 e^{B_2 t} Q_2^* - e^{At}\| = \\ &= \Pi_2\Pi_1 \end{aligned}$$

Since all matrices $B_1, B_2, A$ are negative all the matrix exponentials are decaying: $\|e^{At}\| \leq e^{t\mu(A)}\ \forall t \geq 0$, where $\mu(A) = \lambda_{max}\left(\frac{A + A^\top}{2}\right)$ - the logarithmic norm. □

The graph presented on the Figure 3a describes . One can easily see significant difference between existing global upper bounds for that case Sheng [1994] and derived upper bound.



(a) Global error of the splitting scheme. Initial random full rank matrix $X \in \mathbb{R}^{100\times100}$ was splitted by rows. $X_1, X_2 \in \mathbb{R}^{50\times100}$. Target matrices were obtained the following way: $A_1 = -X_1^*X_1, A_2 = -X_2^*X_2, A = -X^*X$. So $A_1, A_2$ are negative and lacking full rank, while $A = A_1 + A_2$ has full rank.



(b) Global upper bound on the splitting scheme in case of 40 summands in the right-hand side.

**Theorem 2.** *Let $A_1, A_2, \ldots, A_b \in \mathbb{S}_+^p$ be the square negative semidefinite matrices, that don't have full rank, i.e.* $\operatorname{rank} A_i \le p,\ \forall i = 1, \ldots, b$. *While the sum of those matrices has full rank, i.e.* $A = \sum\limits_{i=1}^{b} A_i, \operatorname{rank} A = p$. *Then, the global upper bound error will be written as follows:*

$$\lim_{t\to\infty} \|e^{A_b t} \cdot \ldots \cdot e^{A_1 t} - e^{At}\| = \left\| \prod_{i=1}^{b} \Pi_{b-i+1} \right\|, \tag{11}$$

*where $\Pi_i = I - Q_iQ_i^*$ and $A_i = Q_iB_iQ_i^*$ and $Q_i$ is a matrix with orthonormal columns.*

The graph on the Figure 3b shows empirical validity of the presented upper bound.

## B Proofs

**Theorem 1.** *For any matrix $\mathbf{x_i} \in \mathbb{R}^{b\times p}, b \le p, rank X_i = b$, any vector of right-hand side $\mathbf{y_i} \in \mathbb{R}^b$ and initial vector of parameters $\boldsymbol{\theta}_0$, there is a solution of the $\frac{d\boldsymbol{\theta}}{dt} = -\frac{1}{n}X_i^\top(X_i\boldsymbol{\theta} - \mathbf{y_i})$, given by formula:*

$$\boldsymbol{\theta}(h) = Q_i e^{-\frac{1}{n}R_iR_i^\top h}\left(Q_i^\top\boldsymbol{\theta}_0 - R_i^{-\top}\mathbf{y_i}\right) + Q_iR_i^{-\top}\mathbf{y_i} + (I - Q_iQ_i^\top)\boldsymbol{\theta}_0, \tag{6}$$

*where $Q_i \in \mathbb{R}^{p\times b}$ and $R_i \in \mathbb{R}^{b\times b}$ stands for the $QR$ decomposition of the matrix $\mathbf{X_i}^\top$, $\mathbf{X_i}^\top = Q_iR_i$.*

*Proof.* Given $X_i^\top = Q_iR_i$, we have $(I - Q_iQ_i^\top)X_i^\top = 0$. Note, that $Q_i$ is left unitary matrix, i.e. $Q_i^\top Q_i = I$.

$$\begin{aligned}
\frac{d\boldsymbol{\theta}}{dt} &= -\frac{1}{n}X_i^\top(X_i\boldsymbol{\theta} - \mathbf{y_i}) \\
(I - Q_iQ_i^\top)\frac{d\boldsymbol{\theta}}{dt} &= 0 \\
\frac{d\boldsymbol{\theta}}{dt} &= Q_i\frac{d(Q_i^\top\boldsymbol{\theta})}{dt} \qquad Q_i^\top\boldsymbol{\theta} = \boldsymbol{\eta}_\mathbf{i} \\
\frac{d\boldsymbol{\theta}}{dt} &= Q_i\frac{d\boldsymbol{\eta}_\mathbf{i}}{dt} \quad \text{integrate from 0 to } h \\
\boldsymbol{\theta}(h) &= Q_i\left(\boldsymbol{\eta}_\mathbf{i}(h) - \boldsymbol{\eta}_\mathbf{i}(0)\right) + \boldsymbol{\theta}_0
\end{aligned} \tag{12}$$

On the other hand:

$$\begin{aligned}\frac{d\boldsymbol{\eta}_\mathbf{i}}{dt} &= Q_i^\top \frac{d\boldsymbol{\theta}}{dt} = -\frac{1}{n} Q_i^\top X_i^\top (X_i \boldsymbol{\theta} - \mathbf{y}_\mathbf{i}) = \\ &= -\frac{1}{n} Q_i^\top Q_i R_i (R_i^\top Q_i^\top \boldsymbol{\theta} - \mathbf{y}_\mathbf{i}) = \\ &= -\frac{1}{n} \left( R_i R_i^\top \boldsymbol{\eta}_\mathbf{i} - R_i \mathbf{y}_\mathbf{i} \right) \end{aligned} \tag{13}$$

Consider the moment of time $t = \infty$. $\frac{d\boldsymbol{\eta}_\mathbf{i}}{dt} = 0$, since $\exists \boldsymbol{\theta}^*, Q_i^\top \boldsymbol{\theta}^* = \boldsymbol{\eta}_\mathbf{i}{}^*$. Also consider (13):

$$\begin{aligned}\frac{d\boldsymbol{\eta}_\mathbf{i}}{dt} = 0 = -\frac{1}{n} \left( R_i R_i^\top \boldsymbol{\eta}_\mathbf{i}{}^* - R_i \mathbf{y}_\mathbf{i} \right) \\ R_i \mathbf{y}_\mathbf{i} = R_i R_i^\top \boldsymbol{\eta}_\mathbf{i}{}^* \end{aligned} \tag{14}$$

Now we look at the (13) with the replacement, given in (14):

$$\begin{aligned}\frac{d\boldsymbol{\eta}_\mathbf{i}}{dt} &= -\frac{1}{n} \left( R_i R_i^\top \boldsymbol{\eta}_\mathbf{i} - R_i R_i^\top \boldsymbol{\eta}_\mathbf{i}{}^* \right) \\ \frac{d\boldsymbol{\eta}_\mathbf{i}}{dt} &= -\frac{1}{n} R_i R_i^\top \left( \boldsymbol{\eta}_\mathbf{i} - \boldsymbol{\eta}_\mathbf{i}{}^* \right) \qquad \text{integrate from 0 to } h \\ \boldsymbol{\eta}_\mathbf{i}(h) - \boldsymbol{\eta}_\mathbf{i}{}^* &= e^{-\frac{1}{n} R_i R_i^\top h} (\boldsymbol{\eta}_\mathbf{i}(0) - \boldsymbol{\eta}_\mathbf{i}{}^*) \\ &\text{while } \boldsymbol{\eta}_\mathbf{i}{}^* = R_i^{-\top} \mathbf{y}_\mathbf{i}, \boldsymbol{\eta}_\mathbf{i}(0) = Q_i^\top \boldsymbol{\theta}_0 \\ \boldsymbol{\eta}_\mathbf{i}(h) &= e^{-\frac{1}{n} R_i R_i^\top h} (Q_i^\top \boldsymbol{\theta}_0 - R_i^{-\top} \mathbf{y}_\mathbf{i}) + R_i^{-\top} \mathbf{y}_\mathbf{i} \end{aligned}$$

Using (12) we obtain the target formula

$$\begin{aligned}\boldsymbol{\theta}(h) = Q_i e^{-\frac{1}{n} R_i R_i^\top h} \left( Q_i^\top \boldsymbol{\theta}_0 - R_i^{-\top} \mathbf{y}_\mathbf{i} \right) + \\ + Q_i R_i^{-\top} \mathbf{y}_\mathbf{i} + (I - Q_i Q_i^\top) \boldsymbol{\theta}_0, \end{aligned}$$

□

**Lemma 3.** *Let $A = QBQ^*$, where $Q$ is an $N \times r$ matrix with orthonormal columns, and $B$ is an $r \times r$ matrix. Then,*

$$e^{tA} = (I - QQ^*) + Qe^{tB}Q^*. \tag{15}$$

To prove (15) we note that

$$\begin{aligned} e^{tA} &= \sum_{k=0}^{\infty} \frac{t^k A^k}{k!} = \sum_{k=0}^{\infty} \frac{t^k Q B^k Q^*}{k!} = \\ &= I - QQ^* + QQ^* + Q \sum_{k=1}^{\infty} \frac{t^k B^k}{k!} Q^* = \\ &= (I - QQ^*) + Qe^{tB}Q^*. \end{aligned}$$

**Lemma 4.** *Let $A_1, A_2 \in \mathbb{S}_+^p$ be the square negative semidefinite matrices, that don't have full rank, i.e. $\operatorname{rank} A_1 \leq p$ and $\operatorname{rank} A_2 \leq p$. While the sum of those matrices has full rank, i.e. $A = A_1 + A_2, \operatorname{rank} A = p$. Then, the global upper bound error will be written as follows:*

$$\lim_{t \to \infty} \| e^{A_2 t} e^{A_1 t} - e^{At} \| = \| (I - Q_2 Q_2^*)(I - Q_1 Q_1^*) \| \tag{16}$$

*Proof.* The proof is straightforward. We will use the low rank matrix exponential decomposition from the Lemma 3

$$e^{A_i t} = \Pi_i + Q_i e^{B_i t} Q_i^*, \text{where } \Pi_i = I - Q_i Q_i^*; i = 1, 2$$

$$
\begin{aligned}
&\lim_{t\to\infty} \|e^{A_2 t} e^{A_1 t} - e^{At}\| = \\
&= \lim_{t\to\infty} \|(\Pi_2 + Q_2 e^{B_2 t} Q_2^*)(\Pi_1 + Q_1 e^{B_1 t} Q_1^*) - e^{At}\| = \\
&= \lim_{t\to\infty} \|\Pi_2 \Pi_1 + Q_1 e^{B_1 t} Q_1^* \Pi_2 + \Pi_1 Q_2 e^{B_2 t} Q_2^* + \\
&+ Q_1 e^{B_1 t} Q_1^* Q_2 e^{B_2 t} Q_2^* - e^{At}\| = \\
&= \Pi_2 \Pi_1
\end{aligned}
$$

Since all matrices $B_1, B_2, A$ are negative all the matrix exponentials are decaying: $\|e^{At}\| \leq e^{t\mu(A)}\ \forall t \geq 0$, where $\mu(A) = \lambda_{max}\left(\frac{A+A^\top}{2}\right)$ - the logarithmic norm. $\square$

## C Applications

### C.1 Linear least squares

#### C.1.1 Problem

Let $f_i(\boldsymbol{\theta}) = \|\mathbf{x_i}^\top \boldsymbol{\theta} - y_i\|^2$, then problem (1) is the linear least squares problem, which can be written as

$$
f(\boldsymbol{\theta}) = \frac{1}{n}\|X\boldsymbol{\theta} - \mathbf{y}\|_2^2 = \frac{1}{n}\sum_{i=1}^{s} \|X_i \boldsymbol{\theta} - \mathbf{y_i}\|_2^2 \to \min_{\boldsymbol{\theta}\in\mathbb{R}^p}, \tag{17}
$$

where $X \in \mathbb{R}^{n\times p}$ and $\mathbf{y} \in \mathbb{R}^p$ and the second part of the equation stands for $s$ mini-batches with size $b$ regrouping ($b \cdot s = n$): $X_i \in \mathbb{R}^{b\times p}, \mathbf{y_i} \in \mathbb{R}^b$

$$
\nabla_\theta f(\boldsymbol{\theta}) = \nabla f(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{s} X_i^\top (X_i \boldsymbol{\theta} - \mathbf{y_i}) \tag{18}
$$

The gradient flow equation will be written as follows:

$$
\frac{d\boldsymbol{\theta}}{dt} = -\frac{1}{n}\sum_{i=1}^{s} X_i^\top (X_i \boldsymbol{\theta} - \mathbf{y_i}) \tag{19}
$$

#### C.1.2 Exact solution of the local problem

Theorem 1 gives us explicit formula for the local solution:

$$
\boldsymbol{\theta}(h) = Q_i e^{-\frac{1}{n} R_i R_i^\top h}\left(Q_i^\top \boldsymbol{\theta}_0 - R_i^{-\top} \mathbf{y_i}\right) + Q_i R_i^{-\top} \mathbf{y_i} + (I - Q_i Q_i^\top)\boldsymbol{\theta}_0
$$

#### C.1.3 Kaczmarz as the limit case of splitting

Kaczmarz method Kaczmarz. [1937], Strohmer and Vershynin [2009], Gower and Richtárik [2015] is a well-known iterative algorithm for solving linear systems It is interesting to mention, that splitting approach immediately leads to the Kaczmarz method for solving linear system in the same setting with unit batch size.

When the batch size is equal to one, we need to do $n$ QR decompositions for each transposed batch matrix, which is just column vector $\mathbf{x_i}$ in our case:

$$
\mathbf{x_i} = \mathbf{q_i r_i} = \underbrace{\frac{\mathbf{x_i}}{\|\mathbf{x_i}\|}}_{\mathbf{q_i}} \underbrace{\|\mathbf{x_i}\|}_{\mathbf{r_i}} \tag{20}
$$

Now, we need to use (6) to derive analytic local solution in that case:

$$\begin{aligned}\boldsymbol{\theta}(h) &= \frac{\mathbf{x_i}}{\|\mathbf{x_i}\|} e^{-\frac{\|\mathbf{x_i}\|^2 h}{n}} \left( \frac{\mathbf{x_i}^\top}{\|\mathbf{x_i}\|}\boldsymbol{\theta}_0 - \frac{y_i}{\|\mathbf{x_i}\|} \right) + \\ &+ \frac{\mathbf{x_i}}{\|\mathbf{x_i}\|^2} y_i + \left( I - \frac{\mathbf{x_i}\mathbf{x_i}^\top}{\|\mathbf{x_i}\|^2} \right) \boldsymbol{\theta}_0 = \\ &= \frac{\left(y_i - \mathbf{x_i}^\top \boldsymbol{\theta}_0\right)}{\|\mathbf{x_i}\|^2} \left(1 - e^{-\frac{\|\mathbf{x_i}\|^2 h}{n}}\right) \mathbf{x_i} + \boldsymbol{\theta}_0\end{aligned}$$

It can be easily seen, that:

$$\lim_{h \to \infty} \boldsymbol{\theta}(h) = \frac{\left(y_i - \mathbf{x_i}^\top \boldsymbol{\theta}_0\right)}{\|\mathbf{x_i}\|^2} \mathbf{x_i} + \boldsymbol{\theta}_0, \tag{21}$$

which is exact formula for Kaczmarz method for solving linear system. This result correlates with the statements of Needell et al. [2014], but provides us with a new sense of similarity between SGD and Kaczmarz method.

## C.2 Binary logistic regression

### C.2.1 Problem

In this classification task then problem (1) takes the following form:

$$-\frac{1}{n} \sum_{i=1}^{n} \Big( y_i \ln \sigma(\boldsymbol{\theta}^\top \mathbf{x_i}) + (1 - y_i) \ln(1 - \sigma(\boldsymbol{\theta}^\top \mathbf{x_i})) \Big) \to \min_{\boldsymbol{\theta} \in \mathbb{R}^p}, \tag{22}$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function, while $y_i \in \{0, 1\}$ stands for the label of the object class.

$$\nabla_\theta f(\boldsymbol{\theta}) = \nabla f(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x_i}(\sigma(\boldsymbol{\theta}^\top \mathbf{x_i}) - y_i) \tag{23}$$

The gradient flow equation will be written as follows:

$$\frac{d\boldsymbol{\theta}}{dt} = -\frac{1}{n} \sum_{i=1}^{n} \mathbf{x_i}(\sigma(\boldsymbol{\theta}^\top \mathbf{x_i}) - y_i) \tag{24}$$

Our particular interest lies in mini-batch reformulation of the given problem. We consider $s$ mini-batches with size $b$ regrouping ($b \cdot s = n$): $X_i \in \mathbb{R}^{b \times p}, \mathbf{y_i} \in \mathbb{R}^b$ and $\sigma(\mathbf{x})$ stands for the element-wise sigmoid function.

$$\frac{d\boldsymbol{\theta}}{dt} = -\frac{1}{n} \sum_{i=1}^{s} X_i^\top \left( \sigma\left(X_i \boldsymbol{\theta}\right) - \mathbf{y_i} \right) \tag{25}$$

### C.2.2 Splitting scheme and local problem

Since we are applying splitting scheme to find the approximate solution of the (25), each local problem should be written as follows:

$$\frac{d\boldsymbol{\theta}}{dt} = -\frac{1}{n} X_i^\top \left( \sigma\left(X_i \boldsymbol{\theta}\right) - \mathbf{y_i} \right) \tag{26}$$

Note, that this is not linear equation and cannot be solved as easy as in Theorem 1. However, we can apply the same technique to reduce the dimension of ODE, which is needed to be solved numerically.

Suppose, we have $QR$ decomposition of each batch data matrix $X_i^\top = Q_i R_i$, then we can multiply both sides of (26) on the $(I - Q_i Q_i^\top)$ on the left.

$$\begin{aligned}
(I - Q_i Q_i^\top) \frac{d\boldsymbol{\theta}}{dt} &= (I - Q_i Q_i^\top) \frac{1}{n} X_i^\top (\mathbf{y_i} - \sigma(X_i \boldsymbol{\theta})) \\
\frac{d\boldsymbol{\theta}}{dt} &= Q_i \frac{d(Q_i^\top \boldsymbol{\theta})}{dt} \quad Q_i^\top \boldsymbol{\theta} = \boldsymbol{\eta}_\mathbf{i} \\
\frac{d\boldsymbol{\theta}}{dt} &= Q_i \frac{d\boldsymbol{\eta}_\mathbf{i}}{dt} \quad \text{integrate from 0 to } h \\
\boldsymbol{\theta}(h) &= Q_i (\boldsymbol{\eta}_\mathbf{i}(h) - \boldsymbol{\eta}_\mathbf{i}(0)) + \boldsymbol{\theta}_0
\end{aligned} \tag{27}$$

On the other hand:

$$\begin{aligned}
\frac{d\boldsymbol{\eta}_\mathbf{i}}{dt} &= Q_i^\top \frac{d\boldsymbol{\theta}}{dt} = -\frac{1}{n} Q_i^\top X_i^\top (\sigma(X_i \boldsymbol{\theta}) - \mathbf{y_i}) = \\
&= -\frac{1}{n} Q_i^\top Q_i R_i (\sigma(X_i \boldsymbol{\theta}) - \mathbf{y_i}) = \\
&= -\frac{1}{n} R_i (\sigma(X_i \boldsymbol{\theta}) - \mathbf{y_i})
\end{aligned}$$

Recall, that each hypothesis function depends on linear function $\mathbf{x_i}^\top \boldsymbol{\theta}$, which means, that in batch reformulation it is just entries of the vector $X_i \boldsymbol{\theta}$. Since we have $QR$ decomposition of $X_i^\top$, we can write: $X_i \boldsymbol{\theta} = R_i^\top Q_i^\top \boldsymbol{\theta} = R_i^\top \boldsymbol{\eta}_\mathbf{i}$. In other words:

$$\frac{d\boldsymbol{\eta}_\mathbf{i}}{dt} = -\frac{1}{n} R_i \left( \sigma \left( R_i^\top \boldsymbol{\eta}_\mathbf{i} \right) - \mathbf{y_i} \right), \tag{28}$$

To sum it up, we need to solve (28) (which is much simpler, than original differential equation (26)), than substitute it to the (27) with $\boldsymbol{\eta}_\mathbf{i}(0) = Q_i^\top \boldsymbol{\theta}_0$. Note, that matrices $Q_i$ and $R_i$ can be computed only once before the training.

## C.3 Softmax Regression

### C.3.1 Problem

In this classification task then problem (1) takes the following form:

$$-\frac{1}{n} \sum_{i=1}^{n} \log \left( \frac{\mathbf{y_i}^\top e^{\Theta^\top \mathbf{x_i}}}{\mathbf{1}^\top e^{\Theta^\top \mathbf{x_i}}} \right) \to \min_{\Theta \in \mathbb{R}^{p \times K}}, \tag{29}$$

where $e^{\mathbf{x}}$ is element-wise exponential function, while $\mathbf{y_i} \in \mathbb{R}^K$ stands for the one-hot encoding of the $i$-th object label.

$$\nabla_\Theta f(\Theta) = -\frac{1}{n} \sum_{i=1}^{n} \mathbf{x_i} \left( \mathbf{y_i} - \frac{e^{\Theta^\top \mathbf{x_i}}}{\mathbf{1}^\top e^{\Theta^\top \mathbf{x_i}}} \right)^\top \tag{30}$$

$$\nabla_\Theta f(\Theta) = -\frac{1}{n} \sum_{i=1}^{n} \mathbf{x_i} \left( \mathbf{y_i} - s\left( \Theta^\top \mathbf{x_i} \right) \right)^\top \tag{31}$$

Here we use $s(\mathbf{x})$ as a softmax function of a vector $\mathbf{x}$, i.e. $s(\mathbf{x}) = \frac{e^{\mathbf{x}}}{\mathbf{1}^\top e^{\mathbf{x}}}$ .While mini-batch reformulation will take the following form:

$$\nabla_\Theta f(\Theta) = -\frac{1}{n} \sum_{i=1}^{s} X_i^\top \left( Y_i - s(\Theta^\top X_i^\top) \right)^\top, \tag{32}$$

where $s(X) = \begin{bmatrix} | & | & | & | \\ s(\mathbf{x}_{(1)}) & s(\mathbf{x}_{(2)}) & \cdots & s(\mathbf{x}_{(b)}) \\ | & | & | & | \end{bmatrix}$ is a column-wise softmax function. Indeed, in a very similar manner to the binary logistic regression we can write down gradientflow ODE for softmax regression in a mini-batch form:

$$\frac{d\Theta}{dt} = -\frac{1}{n}\sum_{i=1}^{s} X_i^\top \left(s(\Theta^\top X_i^\top) - Y_i\right)^\top \tag{33}$$

Splitting method requires the local problem, which is focused on a single minibatch:

$$\frac{d\Theta}{dt} = -\frac{1}{n} X_i^\top \left(s(\Theta^\top X_i^\top) - Y_i\right)^\top \tag{34}$$

$$\begin{aligned}
(I - Q_iQ_i^\top)\frac{d\Theta}{dt} &= (I - Q_iQ_i^\top)\frac{1}{n}X_i^\top (Y_i - s(\Theta^\top X_i^\top))^\top \\
\frac{d\Theta}{dt} &= Q_i\frac{d(Q_i^\top \Theta)}{dt} \quad Q_i^\top \Theta = H_i \\
\frac{d\Theta}{dt} &= Q_i\frac{dH_i}{dt} \quad \text{integrate from 0 to } h \\
\Theta(h) &= Q_i\left(H_i(h) - H_i(0)\right) + \Theta_0
\end{aligned} \tag{35}$$

On the other hand:

$$\begin{aligned}
\frac{dH_i}{dt} = Q_i^\top \frac{d\Theta}{dt} &= -\frac{1}{n}Q_i^\top X_i^\top (s(\Theta^\top X_i^\top) - Y_i)^\top = \\
&= -\frac{1}{n}Q_i^\top Q_i R_i (s(\Theta^\top X_i^\top) - Y_i)^\top = \\
&= -\frac{1}{n}R_i (s(\Theta^\top X_i^\top) - Y_i)^\top = \\
&= -\frac{1}{n}R_i (s(H_i^\top R) - Y_i)^\top
\end{aligned}$$

Now we need to solve ODE of variable of the size $b \times k$, rather, than $p \times k$.