A Deep Neural Network Optimization Method Via A Traffic Flow Model

Adeyemi Damilare Adeoye
African Master's in Machine Intelligence
African Institute for Mathematical Sciences (AIMS)
Kigali, Rwanda
`aadeoye@aimsammi.org`

Philipp Petersen
Faculty of Mathematics
University of Vienna
Vienna, Austria
`philipp.petersen@univie.ac.at`

Abstract

We present, via the solution of nonlinear parabolic partial differential equations (PDEs), a continuous-time formulation for stochastic optimization algorithms used for training deep neural networks. Using continuous-time formulation of stochastic differential equations (SDEs), relaxation approaches like the stochastic gradient descent (SGD) method are interpreted as the solution of nonlinear PDEs that arise from modeling physical problems. We reinterpret, through homogenization of SDEs, the modified SGD algorithm as the solution of the viscous Burgers' equation that models a highway traffic flow.

1 Introduction

Deep neural networks (DNNs) have achieved massive success in several areas including image classification, speech recognition, and natural language processing, particularly when there is a nonlinear relationship between the given data and the labels [1–4]. The depths of these networks allow complex data-label relationships to be expressed since each layer nonlinearly transforms the features and therefore effectively filters the information content. This makes them superior to the traditional techniques.

Neural networks (NNs) are considered a class of parametric functions each of which explains some data-label relation. In order to train a given network architecture, we need to find the parameters of these functions that generalize well to new unlabeled data. This leads to solving an inverse problem, also called the learning problem, which involves minimizing a non-convex function. With the presence of several hidden layers, the use of deeper network architectures introduces increased network capacity, and thus a high computational complexity, of the parameter estimation problem. Therefore, despite being used for decades, deep learning has only recently revolutionized many applications incited by advances in computational hardware and availability of large datasets.

When considering very large datasets, notable sources of limitations encountered in deep learning are the dimensionality and non-convexity of the relative optimization problem. The stochastic gradient descent (SGD) method [5, 6] and its variants [7–10] have been extensively used as a traditional learning algorithm for finding optimal solutions. In these SGD-based algorithms, iterative procedures for parameter estimation are performed using gradient information computed via backpropagation [11–13]. While they are efficient to implement and allow the estimation to scale to massive datasets, the incremental updates to the parameter tend to be slow, especially in the initial stages of the training.

Additionally, difficulties stemming from numerical instabilities of the neural network model, and statistical inefficiency (as the algorithms do not use all the information in the dataset) have been reported about these algorithms [14, 15]. A related problem is the observation of vanishing or exploding gradients [16]. Since the gradient represents the sensitivity of the output with respect to a perturbation in the input, a vanishing gradient implies that the output is insensitive with respect to the input, while an exploding gradient implies that the output is unstable with respect to the input.

These observations lead to the question of whether there are other methods for training deep neural networks. Attempts to answer this question result to the development of alternative training methods that outperform and eliminate some of the problems associated with the parameter estimation methods based on SGD. In recent times, researchers have developed frameworks inspired by differential equations for training algorithms. The continuous dynamical system approach to deep learning has been explored, where DNNs are being idealized as a discretization of an ordinary differential equation (ODE) [17–23], or as a discretization of a partial differential equation (PDE) [[24, 25]].

Motivated by previous work on local entropy that was introduced to replace the modified loss function of a non-convex optimization problem, [26] reinterpretes relaxation techniques arising in statistical physics as solutions of a visous Hamilton-Jacobi PDE through a stochastic differential equation (SDE) homogenization problem [27]. The authors proved, through a stochastic control interpretation, that the modified algorithm performs better than the SGD. In this work, we reinterprete the modified SGD as the solution of the viscous Burgers' equation that models a highway traffic flow.

The rest of this work is organized as follows: In section 2, we introduce the concept of DNNs, and study how optimization problems arise in machine learning. In particular, we present the use of DNNs in perceptual tasks such as speech or image recognition categorized as supervised learning tasks that involve large-scale, highly nonlinear, and non-convex optimization problems. We also provide a review of the SGD algorithm for optimizing DNNs, and a motivation for seeking modified SGD algorithms such as the one presented in this work. In section 3, we derive the viscous Burgers' equation as a model for a highway traffic flow, from which we obtain the stochastic differential equation (SDE) satisfied by the parameter updates of the neural network. In section 4, we solve the viscous Burgers' equation and show that the solution exactly gives the gradient of the local entropy function interpreted as the objective function of the optimization problem that results from training a DNN. In section 5, we introduce the concept of homogenization for SDEs [28], and obtain the continuous-time dynamics, which is then sampled numerically to obtain the discrete-time solution.

2 Continuous-time stochastic optimization in deep neural networks (DNNs)

Suppose we are given a set of data points $\{(\theta_1, y_1), \ldots, (\theta_n, y_n)\}$, where $\theta_i \in \mathbb{R}^{d_\theta}, y_i \in \mathbb{R}^{d_y}$, $i = 1, \ldots, n$ represent input-output pairs, and n is the number of samples. Here, the input instance θ_i may represent a word vector, or the feature vector of an object or image, etc, while the output instance y_i may represent a real-valued vector for a regression problem, or an integer-valued vector for a classification problem. A deep neural network (DNN) is a family $\mathcal{H}$ (called the hypothesis space) of prediction functions defined by

$$\mathcal{H} := \{y(\cdot; x) : x \in \mathbb{W}\}, \tag{2.1}$$

where $\mathbb{W}$ is a set of parameters specified below, for some given $y(\cdot;\cdot) : \mathbb{R}^{d_\theta} \times \mathbb{W} \to \mathbb{R}^{d_y}$ whose value, the predicted output vector $\hat{y} \in \mathbb{R}^{d_y}$, is computed by applying iterative transformations to a given input vector $\theta \in \mathbb{R}^{d_\theta}$ through the recursion formula

$$z_0 = \theta; \quad z_l = \sigma(W_l z_{l-1} + b_l), \quad l = 1, \ldots, L, \tag{2.2}$$

where L is the number of hidden layers (or depth) of the neural network, z_l is the hidden state at layer l, the weight matrix $W_l \in \mathbb{R}^{d_l \times d_{l-1}}$ together with the bias vector $b_l \in \mathbb{R}^{d_l}$ constitute the trainable parameters in the l-th layer, and $x = (W_l, b_l) \in \mathbb{W}$ represents the collection of all the trainable parameters with $\mathbb{W} = \mathbb{R}^{d_l \times d_{l-1}} \times \mathbb{R}^{d_l}$. The function $\sigma : \mathbb{R} \to \mathbb{R}$ known as the neuron activation function introduces nonlinearity to the network, and is applied element-wise. Hence, the output z_L of the entire network is given by the function $y : \mathbb{R}^{d_\theta} \times \mathbb{W} \to \mathbb{R}^{d_y}$ defined by

$$y : \mathbb{R}^{d_\theta} \times \mathbb{W} \to \mathbb{R}^{d_y} \tag{2.3}$$

$$(\theta, x) \mapsto y(\theta; x) := T_L \sigma(T_L \cdots \sigma(T_2 \sigma(T_1(\theta)))), \tag{2.4}$$

where $T_l(x) = W_l \theta + b_l$, and $l = 1, \ldots, L$.

2.1 Non-convex optimization problems in DNNs

Let $\ell : \mathbb{R}^{d_y} \times \mathbb{R}^{d_y} \to \mathbb{R}$ be a given distance metric called the loss function. The optimization problem we want to solve involves choosing the parameter of the neural network that minimizes the distance $\ell(y_i, \hat{y}_i)$ between the predicted output $\hat{y}_i = y_x(\theta_i)$ and the true output y_i. Assume that the input-output space $\mathbb{R}^{d_\theta} \times \mathbb{R}^{d_y}$ is endowed with the unknown joint probability distribution $P : \mathbb{R}^{d_\theta} \times \mathbb{R}^{d_y} \to [0, 1]$, and consider the following optimization problem

$$\min_x \mathbb{E}[\ell(y, y_x(\theta))]; \quad \mathbb{E}[\ell(y, y_x(\theta))] \coloneqq \int_{\mathbb{R}^{d_\theta} \times \mathbb{R}^{d_y}} \ell(y, y_\theta(x)) dP(\theta, y). \tag{2.5}$$

Since P is unknown, and that in supervised learning we have access to a set of $N \in \mathbb{N}$ independently drawn input-output samples $\{(\theta_i, y_i)\}_{i=1}^N \subseteq \mathbb{R}^{d_\theta} \times \mathbb{R}^{d_y}$ from P, we rather attempt to minimize the *empirical risk* $F : \mathbb{W} \to \mathbb{R}$ given as

$$F(x) \coloneqq \frac{1}{N} \sum_{i=1}^{N} \ell(y_i, y_x(\theta_i)). \tag{2.6}$$

Therefore, the optimization problem we wish to solve is the empirical risk minimization problem

$$\min_x F(x), \tag{2.7}$$

and we write

$$x^* = \operatorname*{argmin}_x F(x), \tag{2.8}$$

as the value of x that minimizes $F(x)$.

The choice of the loss function $\ell(y, \hat{y})$ mostly depends on the type of problem we want to solve. For example, a popular choice of $\ell(y, \hat{y})$ for a regression problem is the quadratic loss function $\ell(y, \hat{y}) = \|y - \hat{y}\|^2$. For binary classification problems, $\ell(y, \hat{y})$ is usually chosen to be $\ell(y, \hat{y}) = \log(1 + \exp(-y\hat{y}))$.

The optimization problem (2.7) is highly nonlinear and nonconvex in general, making it difficult to solve to global optimality [29, 30]. Gradient-based methods have however been successfully used to compute approximate solutions, where the gradient of the objective in (2.7) is computed by the chain rule using back-propagation. A typical DNN contains very large numbers of parameters in their layers, each of which has to be optimized. This can take several hundreds of hours of training, and weeks of computation on a high-end computer. Popular gradient-based methods that have been used extensively to optimize DNNs are the first-order algorithms such as the gradient descent (GD), and the stochastic gradient descent (SGD) and its variants. Developments in this work are based on continuous-time SDE interpretation of the SGD.

2.2 Gradient-based optimization

The GD [31, 32] used for solving optimization problems of the form (2.8) performs iteration of the form

$$x_{k+1} = x_k - \delta^t \nabla F(x_k), \tag{2.9}$$

where $\delta^t > 0$ is the step-size (or *learning rate*) and $\nabla F(x_k)$ is the gradient of the loss function at the k-th iteration ($k \in \mathbb{N}$).

In practice, it is not possible to load all samples into the memory of a single GPU or CPU for computing the entire gradient $\frac{1}{N} \sum_{i=1}^{N} \nabla \ell(y_i, F_x(\theta_i))$. A more practical choice is SGD or any of its variants. Theoretically, SGD works as follows: at the k-th iteration, randomly select i and update the parameter by

$$x_{k+1} = x_k - \delta^t \nabla F_i(x_k), \tag{2.10}$$

where $F_i(x) \coloneqq \ell(y_i, F_x(\theta_i))$. The stochastic nature of SGD arises from the approximation of the gradient over a subset of the data points. Also common is "mini-batch" SGD, where we choose a random subset $I_k \subseteq 1, \dots, N, |I_k| = B \ll N$ to write (2.7) as a finite-sum optimization problem:

$$\min_x F_{\text{mb}}(x); \quad F_{\text{mb}}(x) \coloneqq \frac{1}{B} \sum_{i=1}^{B} F_i(x), \tag{2.11}$$

where each $F_i(x)$ represents the sum of training loss for a "mini-batch" of training samples, and B is the total number of mini-batches (smaller than the total number of training samples N) [5]. We then perform the iteration

$$x_{k+1} = x_k - \delta^t \nabla F_{\text{mb}}(x_k). \tag{2.12}$$

One reason for the use of SGD instead of GD is the faster convergence in a "memory-constraint" system where it is difficult to find a parallel way to load and process all samples in a single machine [33]. In such a system, batch GD has to scan through the entire training set during one iteration before taking the next iteration, whereas the SGD starts to converge right away from the first sample, hence a noticeable faster convergence speed in the early iterations.

2.3 Continuous-time stochastic optimization

While it converges very fast in the early iterations, the SGD algorithm slows down in later iterations and thus struggles to reach an accurate solution. This is because although the stochastic gradient is an unbiased estimator of the gradient:

$$\mathbb{E}[\nabla F_{\text{mb}}(x)] = \nabla F(x), \tag{2.13}$$

it may end up with a high variance. In order to get the best of both the fast initial convergence of SGD and the steady linear convergence of GD to a great extent, we further assume a bounded variance for the stochastic gradient, that is, for all iterates x_k of the SGD, there exists $\nu_{\text{mb}} > 0$ such that [34]

$$\mathbb{E}[\|\nabla F_{\text{mb}}(x) - \nabla F(x)\|^2] \leq 2\nu_{\text{mb}}. \tag{2.14}$$

Here, the "mb" in the notation ν_{mb} *only* indicates the noise is coming from the mini-batch gradients. Henceforth, we will simply write ν.

We start with the standard observation that the discrete-time dynamics in (2.10) can be viewed as a discretization of the continuous-time Langevin dynamics described by the Itô stochastic differential equation (SDE) [35, 36]

$$\mathrm{d}x(t) = -\nabla F(x(t))\,\mathrm{d}t + \sqrt{2\nu}\,\mathrm{d}W(t), \quad t \geq 0, \quad x(0) = x_0, \tag{2.15}$$

where $W(t) \in \mathbb{R}^m$ is the standard m-dimensional Wiener process.

Under suitable assumptions on F, it can be shown that the unique invariant distribution of (2.15) is the Gibbs distribution

$$\rho^{\infty}(\nu; x) \propto \exp\left(-\frac{1}{2\nu} F(x)\right), \tag{2.16}$$

and that using the idea of simulated annealing to evade metastability[1], the distributions of $x(t)$ are made to converge rapidly to ρ^{∞} as $t \to \infty$ [37–39]. Moreover from (2.16), ρ^{∞} concentrates around the minimizers of $F(x)$ for small enough values of ν [40, 41]. Consequently, as the Gibbs distribution turns out to be an approximate minimizer for the empirical risk (2.6), we will observe that the resulting modified SGD algorithm tracks the Langevin dynamics in a suitable sense.

2.4 Regularization of the loss function

As discussed, training DNNs involves modeling a high-dimensional nonconvex loss function. In order to improve the computational efficiency with a large number of training data, the stochastic estimation of the gradients of these loss functions are carried out using the SGD algorithm or any of its variants. However, to improve performance of the optimization algorithm and the generalization of the model, it is generally required to introduce a regularization term in the loss function. Several regularization techniques that show outstanding results have been developed for optimizing nonconvex empirical risks of deep networks. These regularization techniques are categorized as either *explicit*, e.g. weight decay, sparsity constraint, and entropy minimization or *implicit*, e.g. noise injection, dropout, learning rate decay, model ensemble, and batch normalization [42].

[1]**Metastability** is a phenomenon that describes a set of states of a dynamic system that can persist for a long time. In other words, a dynamic system is said to exhibit metastability if the convergence of the probability density of its states takes an exponentially long time.

Our focus in this work is to derive, through homogenization of SDEs, a form of the regularized SGD presented by [26], from the solution of the viscous Burgers' equation that models a highway traffic flow. [26] models the loss function of a class of deep networks as a spin glass Hamiltonian system, and with results that show good empirical performance, they introduce a smoothing technique that stems from imposing PDE regularity which contributes to the analysis of the geometric property of the energy landscape of the networks. By combining tools from the theory of PDEs and tools from modeling stochastic first-order algorithms as continuous-time stochastic processes, we are able to provide insights on how new approaches for solving optimization problems that arise in DNNs could be developed starting with mathematical models that describe real life situations. As shown, this process can often lead to improvements in state-of-the-art DNN algorithms.

3 The viscous Burgers' equation as the model for a highway traffic flow

Consider the viscous Burgers' equation

$$u_t + u \cdot \nabla u = \nu \Delta u \quad \text{for } (x,t) \in \mathbb{R}^d \times [0,\infty), \tag{3.1}$$

where $\nu \in \mathbb{R}^+$ is a positive coefficient, and $\nabla = \sum_{i=1}^{n} \frac{\partial}{\partial x_i}$ and $\Delta = \sum_{i=1}^{n} \frac{\partial^2}{\partial x_i^2}$ are respectively the gradient and Laplacian in the space variables.

We will express the model for a highway traffic flow as the Burgers' equation (3.1). The traffic flow model will be derived using a continuous, deterministic approach. By continuous, we mean that the traffic will be observed from a distance far enough that the size and behaviour of a car becomes negligible, and will rather deal with average concepts, such as the number of cars per mile and the average speed of the car. By deterministic, we mean to consider the situation where the change of the traffic with time is exclusively specified by the model. Hence, in this setting, each car entering and exiting a highway will be replaced by a continuous density function

$$\begin{aligned} \rho\colon \mathbb{R}^d \times (0,\infty) &\to \mathbb{R} \\ (x,t) &\mapsto \rho(x,t). \end{aligned} \tag{3.2}$$

With the additional constraint that follows from the realization that the typical car speed depends on the level of traffic congestion (the car density), that is $q = Q(\rho)$, the differential form of the equation for car (nonlinear) conservation is given by the closed system

$$\rho_t + \nabla q = 0 \tag{3.3}$$

$$q = Q(\rho) \tag{3.4}$$

where the traffic density $\rho(x,t) \in \mathbb{R}^d$ measures the number of cars per unit length of the road at position x and time t, and the traffic flow $Q(x,t) \in \mathbb{R}^d$, also known as the *flux*, represents the number of cars crossing position x per unit time at time t [43]. A third fundamental (dependent) quantity that helps to describe the traffic flow is the velocity $u \in \mathbb{R}^d$ of a car at any point on the highway, and is assumed to depend only on the traffic density, that is, $u = U(\rho)$. These three fundamental variables are related by the identity

$$Q = \rho U. \tag{3.5}$$

The prescribed function $Q(\rho)$ depends on the characteristics of the road, cars and drivers, such as the car speed, speed limits, weather conditions, and the road conditions.

Equations (3.3) and (3.4) together gives the first-order, non-linear, partial differential equation for traffic flow in the form of a closed equation for the single variable ρ:

$$\rho_t + Q'(\rho)\nabla\rho = 0 \quad \text{for } (x,t) \in \mathbb{R}^d \times (0,\infty). \tag{3.6}$$

As the traffic flow model is based on the first-order approximation, we could define a better approximation for q by assuming it is a function of the density ρ as well as its gradient ρ_x, and define

$$q = Q(\rho) - \nu\nabla\rho. \tag{3.7}$$

Consider a linear relationship between velocity and traffic density [43]:

$$u(\rho) = u_m \left(1 - \frac{\rho}{\rho_m}\right), \tag{3.8}$$

where u_m is the maximum velocity, and ρ_m is the maximum density. We have (from (3.5))

$$\begin{aligned} Q'(\rho) &= \frac{d}{d\rho}(\rho u) \\ &= \frac{d}{d\rho}\left[\rho u_m\left(1-\frac{\rho}{\rho_m}\right)\right] \\ &= u_m\left(1-\frac{2\rho}{\rho_m}\right). \end{aligned} \tag{3.9}$$

Also, from (3.7)

$$\nabla q = Q'(\rho)\nabla\rho - \nu\Delta\rho,$$

or (using (3.3))

$$\rho_t + Q'(\rho)\nabla\rho - \nu\Delta\rho = 0. \tag{3.10}$$

Assume $w, \rho \in C^\infty(\mathbb{R}^d \times (0,\infty))$, and define

$$w(\rho) = Q'(\rho) = u_m\left(1-\frac{2\rho}{\rho_m}\right), \tag{3.11}$$

then

$$\rho = \frac{\rho_m}{2u_m}(1-w), \quad \rho_t = -\frac{\rho_m w_t}{2u_m}, \quad \nabla\rho = -\frac{\rho_m\nabla w}{2u_m}, \quad \text{and} \quad \Delta\rho = \frac{\rho_m\Delta w}{2u_m}. \tag{3.12}$$

Substituting (3.12) into (3.10) and multiplying the resulting equation by $w'(\rho) = -\frac{2u_m}{\rho_m}$, we get the viscous Burgers' equation for the highway traffic flow:

$$w_t + w\nabla w = \nu\Delta w \quad \text{for } (x,t) \in \mathbb{R}^d \times (0,\infty). \tag{3.13}$$

4 Nonlinear parabolic PDEs and the local entropy

Introduced for studying the energy landscape of neural network learning algorithms, the local entropy is a modified form of the objective function $F(x)$ of the neural network parameters [see for example, 44]. While extending the notion of local entropy to continuous variables, [27] replaced the objective function $F(x)$, defined for $x \in \mathbb{R}^d$, with the function $F_\gamma(x)$ given by

$$F_\gamma(x) = 2\nu\log\left(G_{2\nu\gamma} * \exp\left(-\frac{1}{2v}F(x)\right)\right) \quad (x \in \mathbb{R}^d), \tag{4.1}$$

where $G_\gamma(x) = (2\pi\gamma)^{d/2}\exp\left(-\frac{|x|^2}{2\gamma}\right)$ is the Green's function (or heat kernel), and $f * g$ defines the convolution of f and g:

$$(f * g)(x) := \int_{-\infty}^{\infty} f(t)g(x-t)dt. \tag{4.2}$$

Assuming $F \in C^\infty(\mathbb{R}^d)$, we will derive a closed form solution $w(x,t)$ of the initial-value problem

$$\begin{cases} w_t + w\nabla w = \nu\Delta w & \text{in } (x,t) \in \mathbb{R}^d \times (0,\infty), \\ w(x,0) = F(x) & \text{on } (x,t) \in \mathbb{R}^d \times \{t=0\}, \end{cases} \tag{4.3}$$

for the viscous Burgers' equation (3.13), by a non-linear change of variables, due to the transformation of Cole [45] and Hopf [46], which turns it into an initial-value problem for a linear diffusion equation (the heat equation). We will then show that this solution $w(x,t)$ recovers the gradient of $F_\gamma(x)$.

4.1 Proposition. The local entropy function $F_\gamma(x)$ defined by (4.1) is the solution of the initial-value problem for the viscous Hamilton-Jacobi equation

$$\phi_t + \frac{(\nabla\phi)^2}{2} = \nu\Delta\phi \quad \text{for } (x,t) \in \mathbb{R}^d \times (0,\infty), \tag{4.4}$$

with initial values $\nabla\phi(x,0) = F(x)$ [26]. Moreover, the gradient $\nabla F_\gamma(x)$ solves the viscous Burgers' equation (4.3).

Proof. We start by writing the viscous Burgers' equation (3.13) as

$$w_t + \nabla \left(\frac{w^2}{2} - \nu \nabla w \right) = 0. \tag{4.5}$$

As it is not yet specified, let us assume for the moment that w is a smooth solution of (4.5), we set $\psi := \phi(w)$, where $\phi : \mathbb{R}^d \times (0, \infty) \to \mathbb{R}$ is a smooth function. We want to choose ϕ so that ψ solves a linear equation. Let w satisfy the irrotational condition

$$\nabla \times W = \sum_{i,j=1}^{d} \left(\frac{\partial w_j}{\partial x_i} - \frac{\partial w_i}{\partial x_j} \right) \left(e_i \wedge e_j \right) = 0, \tag{4.6}$$

where $e_i, i \leq 1 \leq d$ are basis of $\mathbb{R}^d$, and for vectors $u, v \in \mathbb{R}^d$, $u \wedge v$ defines the wedge/exterior product given by

$$u \wedge v = u \otimes v - v \otimes u. \tag{4.7}$$

We define

$$\phi(x,t) := \int_{-\infty}^{x} w(z,t) dz, \implies w(x,t) = \nabla \phi(x,t), \tag{4.8}$$

and write the Hopf-Cole transformation [46, 45]

$$\phi = -2\nu \log(\psi), \tag{4.9}$$

where $\log(v) = (\log v_i)_{1 \leq i \leq d}$ for a vector $v \in \mathbb{R}^d$.

We have

$$\begin{cases} \phi_t = -\dfrac{2\nu}{\psi} \psi_t, \\ \nabla \phi = -\dfrac{2\nu}{\psi} \nabla \psi, \\ \Delta \phi = -\dfrac{2\nu}{\psi} \Delta \psi + 2\nu \left(\dfrac{\nabla \psi}{\psi} \right)^2. \end{cases} \tag{4.10}$$

Putting (4.10) in (4.8) and (4.5), we get the linear d-dimensional heat equation

$$\psi_t = \nu \Delta \psi \quad (x \in \mathbb{R}^d, t > 0), \tag{4.11}$$

with the initial condition $\psi(x, 0) = g_1(x)$. In terms of the initial value $w(x, 0) = F(x)$, and from (4.10), the initial data for ψ becomes

$$w(x,0) = F(x) = -2\nu \frac{\nabla \psi(x,0)}{\psi(x,0)}. \tag{4.12}$$

Integrating, we get

$$\psi(x,0) = g_1(x) = \exp \left(-\frac{1}{2\nu} \int_0^x F(z) dz \right). \tag{4.13}$$

The general solution to the resulting initial-value problem for ψ is given as convolution of the initial-value with the heat kernel $H(x,t)$:

$$\begin{cases} \psi(x,t) = \displaystyle\int_{-\infty}^{\infty} H(x-y,t) \psi(y,0) dy \qquad \text{for } (x,t) \in \mathbb{R}^d \times (0, \infty), \\ H(x,t) = \dfrac{1}{\sqrt{4\pi\nu t}} \exp \left(-\dfrac{x^2}{4\nu t} \right). \end{cases} \tag{4.14}$$

By normalizing $\phi(x,t)$ by $\phi(0,t) = 0$, we obtain from $\psi(x,t)$, the solution for the initial-value problem for the viscous Hamilton-Jacobi equation (4.4):

$$\phi(x,t) = -2\nu \log \left\{ \int_{-\infty}^{\infty} \frac{1}{\sqrt{4\pi\nu t}} \exp \left[-\frac{(x-y)^2}{4\nu t} - \frac{1}{2\nu} \int_0^y F(z) dz \right] dy \right\} =: F_t(x). \tag{4.15}$$

Finally, the solution of the viscous Burgers' equation (3.13) is

$$w(x,t) = \nabla\phi(x,t) = \frac{\int_{-\infty}^{\infty} \frac{x-y}{t} \exp\left(-\frac{(x-y)^2}{4\nu t} - \frac{1}{2\nu}\int_0^y F(z)dz\right) dy}{\int_{-\infty}^{\infty} \exp\left(-\frac{(x-y)^2}{4\nu t} - \frac{1}{2\nu}\int_0^y F(z)dz\right) dy}, \tag{4.16}$$

where $x \in \mathbb{R}^d, t > 0$. ■

4.2 Remark. The solution $\psi(x,t)$ of the heat equation (4.11), given by (4.14), can be written in a more convenient form as

$$\psi(x,t) = \frac{1}{2\sqrt{\pi\nu t}} \int_{-\infty}^{\infty} \exp\left(-\frac{G}{2\nu}\right) dy, \tag{4.17}$$

where

$$G(x,y;t) = \int_0^y F(z)dz + \frac{(x-y)^2}{2t} \quad (x, y \in \mathbb{R}^d, t > 0), \tag{4.18}$$

and the gradient is given by

$$\nabla\psi(x,t) = -\frac{1}{4\nu\sqrt{\pi\nu t}} \int_{-\infty}^{\infty} \frac{x-y}{t} \exp\left(-\frac{G}{2\nu}\right) dy. \tag{4.19}$$

4.3 Remark. We can equivalently write (4.16) as

$$w(x,t) = \nabla\phi(x,t) = \int_{-\infty}^{\infty} \frac{y-x}{t} \rho_1^{\infty}(dy;x) = \int_{-\infty}^{\infty} \nabla F(x-y) \rho_2^{\infty}(dy;x), \tag{4.20}$$

where $x \in \mathbb{R}^d, t > 0$. The functions $\rho_1^{\infty}(y;x), \rho_2^{\infty}(y;x)$ are invariant distributions given by

$$\begin{cases} \rho_1^{\infty}(y;x) = \dfrac{1}{Z_1} \exp\left(-\dfrac{1}{2\nu} F(y) - \dfrac{|x-y|^2}{4\nu t}\right), \\ \rho_2^{\infty}(y;x) = \dfrac{1}{Z_2} \exp\left(-\dfrac{1}{2\nu} F(x-y) - \dfrac{|y|^2}{4\nu t}\right), \end{cases} \tag{4.21}$$

and

$$\begin{cases} Z_1(x) = -\displaystyle\int_{-\infty}^{\infty} \exp\left(-\frac{1}{2\nu} F(y) - \frac{|x-y|^2}{4\nu t}\right) dy, \\ Z_2(x) = -\displaystyle\int_{-\infty}^{\infty} \exp\left(-\frac{1}{2\nu} F(x-y) - \frac{|y|^2}{4\nu t}\right) dy, \end{cases} \tag{4.22}$$

are normalizing constants.

5 The local entropy via homogenization for SDEs

We assume a smooth, steady, periodic, and incompressible (or divergence-free) flow velocity, that is, $w(x,t) = -h(x)$, where h is a smooth periodic vector field with $\nabla \cdot h(x) = 0$. Consider $F(x) = g^{\epsilon} := g(\epsilon x)$, with $\epsilon > 0$, so that initially the velocity is slowly varying in space. We expect that the velocity will not vary significantly on small length and time scales, hence to see the effective behaviour of (3.13), we need to look at large length and time scales. Furthermore, if h averages over the unit cube (not necessarily to zero), and if we consider the rescaling $x \to x/\epsilon$ and $t \to t/\epsilon^p$ $(p > 0)$, so as to bring out order-one variations in velocity on large length and time scales, homogenization techniques enable us to show that the rescaled velocity field $w^{\epsilon}(x,t) := w(x/\epsilon, t/\epsilon^p)$ converges, for $\epsilon \ll 1$, to the solution $w(x,t)$ of the viscous Burgers' equation (4.3).

In this setting, equation (4.3) becomes

$$\begin{cases} \dfrac{1}{\epsilon^{2-p}} w_t^{\epsilon} - \dfrac{1}{\epsilon} h^{\epsilon} w_x^{\epsilon} = \nu w_{xx}^{\epsilon} & \text{in } (x,t) \in \mathbb{R}^d \times (0,\infty), \\ w^{\epsilon}(x,0) = g(\epsilon x) & \text{on } (x,t) \in \mathbb{R}^d \times \{t=0\}, \end{cases} \tag{5.1}$$

where $h^\epsilon = h(x/\epsilon)$. As to what time scale to consider (choice of p) in order to observe interesting dynamical behaviour, it is intuitive to study (5.1) for different values of p in cases when h averages to zero and when it does not average to zero. For the purpose of this work, with $v > 0$, we are considering the case when h averages to zero, hence the choice $p = 2$, and equation (5.1) becomes

$$\begin{cases} w_t^\epsilon - \dfrac{1}{\epsilon} h^\epsilon w_x^\epsilon = \nu w_{xx}^\epsilon & \text{in } (x,t) \in \mathbb{R}^d \times (0,\infty), \\ w^\epsilon(x,0) = g(\epsilon x) & \text{on } (x,t) \in \mathbb{R}^d \times \{t=0\}, \end{cases} \tag{5.2}$$

Let $W(t) \in \mathbb{R}^m$ denote an independent standard m-dimensional Wiener process, $h : \mathcal{Z} = \mathbb{R}^n \to \mathbb{R}^n$ a smooth periodic vector-valued function, and $\beta : \mathcal{Z} = \mathbb{R}^n \to \mathbb{R}^{n\times m}$ a smooth matrix-valued or scalar-valued function. Consider the Itô SDE

$$\frac{dz}{dt} = h(z) + \beta(z)\frac{dW}{dt}, \quad z(0) = z_0, \tag{5.3}$$

which is interpreted as an integral equation for $z(t) \in C(\mathbb{R}^+, \mathcal{Z})$:

$$z(t) = z_0 + \int_0^t h(z(s))ds + \int_0^t \beta(z(s))dW(s). \tag{5.4}$$

We introduce the auxiliary dynamics $y = x/\epsilon$ and consider the case where $z = (x^T, y^T)^T$, with $x \in \mathcal{X} = \mathbb{R}^d$, $y \in \mathcal{Y} = \mathbb{R}^{n-d}$. We then write the system of SDEs

$$\begin{cases} dx(s) = h(x,y)ds, & x(0) = x_0, \\ dy(s) = \dfrac{1}{\epsilon} g(x,y)ds + \dfrac{1}{\sqrt{\epsilon}}\beta(x,y)dW(s), & y(0) = y_0, \end{cases} \tag{5.5}$$

where we have scaled time to $s = \epsilon t$, and g is a sufficiently smooth function. Both $x(s)$ and $y(s)$ contain fast dynamics, but with the homogenization parameter $\epsilon > 0$, the dynamics in $y(s)$ is provided with a faster time-scale than that in $x(s)$.

The system (5.5) corresponds to the backward Kolmogorov equation

$$\begin{cases} \dfrac{\partial w}{\partial t} = \dfrac{1}{\epsilon}\mathcal{L}_0 w & \text{in } (x,y,t) \in \mathcal{X} \times \mathcal{Y} \times (0,\infty), \\ w(x,0) = g(\epsilon x) & \text{on } (x,y,t) \in \mathcal{X} \times \mathcal{Y} \times \{t=0\}, \end{cases} \tag{5.6}$$

where the generator $\mathcal{L}_0$ is defined as

$$\mathcal{L}_0 w = g(x,y) \cdot \nabla_y w + \frac{1}{2} B \Delta_y w, \tag{5.7}$$

with $B(x,y) = \beta(x,y)\beta(x,y)^T$.

The corresponding Fokker-Planck (forward Kolmogorov) equation is

$$\begin{cases} \dfrac{\partial w}{\partial t} = \dfrac{1}{\epsilon}\mathcal{L}_0^* w & \text{in } (x,y,t) \in \mathcal{X} \times \mathcal{Y} \times (0,\infty), \\ w(x,0) = g(\epsilon x) & \text{on } (x,y,t) \in \mathcal{X} \times \mathcal{Y} \times \{t=0\}, \end{cases} \tag{5.8}$$

where the $L^2(\mathbb{R}^d)$-adjoint operator $\mathcal{L}_0^*$ is defined as

$$\mathcal{L}_0^* w = -\nabla_y (g(x,y)w) + \frac{1}{2}\Delta_y (B(x,y)w). \tag{5.9}$$

Next, we eliminate the $y(s)$ dependence in the Kolmogorov equation (5.6) to identify only a simplified equation for the dynamics of $x(s)$. To do this, we make an ergodicity assumption: we asume that for each fixed x, $\mathcal{L}_0$ has a one-dimensional null space, and $\mathcal{L}_0^*$ has a null space spanned by the invariant distribution $\rho_1^\infty(y;x)$. That is, $\mathcal{L}_0$ satisfies

$$\begin{cases} \mathcal{L}_0 \mathbf{1}(y) = 0, \\ \mathcal{L}_0^* \rho_1^\infty(y;x) = 0, \end{cases} \tag{5.10}$$

where $\mathbf{1}(y)$ denotes constants (equal to 1 a.e.) in $y(s)$ and $\rho_1^\infty(y;x)$ is the density of an ergodic measure $\mu_x(dy) = \rho_1^\infty(y;x)dy$.

5.1 Result. For $\epsilon \ll 1$ and times t up to $\mathcal{O}(1)$, $w^\epsilon(x,t)$ solving (5.2) is approximated by $w(x,t)$ solving the homogenized equation

$$\begin{cases} w_t = \nabla F_\gamma(x) w_x & \text{in } (x,t) \in \mathcal{X} \times (0,\infty), \\ w(x,0) = g, & \text{on } (x,t) \in \mathcal{X} \times \{t=0\}. \end{cases} \tag{5.11}$$

We equivalently state the result as follows: For $\epsilon \ll 1$ and times t up to $\mathcal{O}(1)$, $x(s)$ solving (5.5) is approximated by X solving

$$\frac{dX}{ds} = \nabla F_\gamma(X), \quad X(0) = x_0, \tag{5.12}$$

where the homogenized vector field ∇F for X is defined as the average against the ergodic measure μ_x, and by ergodicity, the spatial average $\int_{\mathcal{Y}} \frac{y(s)-X}{\gamma} d\mu_x$, is equal to the temporal average $\frac{1}{T}\int_0^T \frac{y(s)-x}{t} ds$ for $T \gg 1$:

$$\nabla F_\gamma(X) = \int_{\mathcal{Y}} \frac{y(s)-X}{\gamma} \mu_x(dy) = \lim_{T\to\infty} \frac{1}{T}\int_0^T \frac{y(s)-x}{t} ds. \tag{5.13}$$

Put $\beta = (2\nu)^{1/2}$, and define $g(x,y) \equiv -\nabla_y G(x,y;\gamma) = -\nabla F(y) + \frac{x-y}{\gamma}$ in (5.5). We obtain the system of SDEs

$$\begin{cases} dx(s) = -\gamma^{-1}(x-y)ds, & x(0) = x_0, \\ dy(s) = -\frac{1}{\epsilon}\left[\nabla F(y) + \frac{y-x}{\gamma}\right] ds + \sqrt{\frac{2}{\epsilon}}\nu^{1/2} dW(s), & y(0) = y_0. \end{cases} \tag{5.14}$$

Let $\rho(x,t) \in C^{2,1}\left(\mathcal{X}\times(0,\infty),\mathbb{R}\right) \cap C\left(\mathcal{X}\times(0,\infty),\mathbb{R}\right)$ denote the probability density of the dynamic $y(s)$, then the Fokker-Planck equation for $\rho(x,t)$ is given by

$$\begin{cases} \frac{\partial\rho}{\partial t} = \mathcal{L}_0^*\rho = \nabla_y(\nabla_y G\rho) + \frac{1}{2}\sqrt{2\nu}\Delta_y\rho & \text{in } (x,y,t) \in \mathcal{X}\times\mathcal{Y}\times(0,\infty), \\ \rho(x,0) = \rho_0 & \text{on } (x,y,t) \in \mathcal{X}\times\mathcal{Y}\times\{t=0\}. \end{cases} \tag{5.15}$$

The invariant measure ρ_1^∞ given by

$$\rho_1^\infty(y;x) = \frac{1}{Z}\exp\left(-\beta G(x,y;\gamma)\right), \tag{5.16}$$

satisfies the Fokker-Planck equation (5.15) [28, 47], which agrees with equation (4.21) of Remark 4.3.

5.2 Sampling for a discrete-time solution

Using the Milstein method [48], the system (5.14) is sampled for $y(s)$ and $x(s)$ on the time interval $[0,T]$ with step $\delta^t > 0$, known as the learning rate. Suppose that we partition the interval $[0,T]$ into N equal subintervals

$$0 = \tau_0 < \tau_1 < \cdots < \tau_N = T, \tag{5.17}$$

with $\tau_k = k\delta^t$ and $\delta^t = T/N$, then for $k = 0,\ldots,N$, we obtain the discrete-time system

$$y_{k+1} = y_k - \delta^t\left[\nabla F_{\text{mb}}(y_k) + \frac{y_k - x_k}{\gamma}\right] + \sqrt{2\delta^t\nu}\delta_k^w, \tag{5.18}$$

$$x_{k+1} = \begin{cases} x_k - \delta^t\gamma^{-1}(x_k - y_k) & \text{if } (k \bmod N) = 0, \\ x_k & \text{otherwise,} \end{cases} \tag{5.19}$$

where again, the "mb" notation indicates we *only* have the mini-batch gradients, and $\delta_k^w = W_{\tau_{k+1}} - W_{\tau_k}$ are normal random variables with mean zero and scale $\sqrt{\delta^t}$.

6 Conclusion

We presented an optimization method which is based on the SGD in continuous-time. Like any other gradient-based algorithm developed through a continuous-time model, the algorithm allow parameters to be updated on-line in continuous-time with the parameter updates x_t satisfying an SDE. While establishing a connection, via homogenization techniques, between solutions of nonlinear PDEs (like the viscous Hamilton-Jacobi equation) and stochastic optimization algorithms used for training DNNs, it was studied in [26] that one could develop modified SGD algorithms that scale better in practice than the SGD method. Following this result, we were able to show in this work that the resulting modified SGD algorithm which is interpreted as the gradient of the solution of a viscous Hamilton-Jacobi equation indeed solves a viscous Burgers' equation that models a highway traffic flow.

Acknowledgements

The authors are grateful to Google and Facebook for sponsoring the Master's program (African Master's in Machine Intelligence) which birthed this work, and to the African Institute for Mathematical Sciences, Rwanda for providing the necessary platform.

References

[1] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," *Neural Information Processing Systems*, vol. 25, 01 2012.

[2] Y. Bengio *et al.*, "Foundations and trends in machine learning," *Foundations and Trends in Signal Processing*, vol. 7, no. 3-4, 2009.

[3] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal processing magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[4] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[5] H. Robbins and S. Monro, "A stochastic approximation method," *The annals of mathematical statistics*, pp. 400–407, 1951.

[6] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*, pp. 177–186, Springer, 2010.

[7] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization.," *Journal of machine learning research*, vol. 12, no. 7, 2011.

[8] M. D. Zeiler, "Adadelta: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

[9] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[10] R. Johnson and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction," in *Advances in neural information processing systems*, pp. 315–323, 2013.

[11] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," tech. rep., California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[12] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[13] Y. LeCun, D. Touresky, G. Hinton, and T. Sejnowski, "A theoretical framework for back-propagation," in *Proceedings of the 1988 connectionist models summer school*, vol. 1, pp. 21–28, CMU, Pittsburgh, 1988.

[14] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *SIAM Review*, vol. 60, no. 2, pp. 223–311, 2018.

[15] P. Toulis, D. Tran, and E. Airoldi, "Towards stability and optimality in stochastic gradient descent," in *Artificial Intelligence and Statistics*, pp. 1290–1298, 2016.

[16] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[17] Q. Li, T. Lin, and Z. Shen, "Deep learning via dynamical systems: An approximation perspective," *arXiv preprint arXiv:1912.10382*, 2019.

[18] W. E, "A proposal on machine learning via dynamical systems," *Communications in Mathematics and Statistics*, vol. 5, no. 1, pp. 1–11, 2017.

[19] P. Parpas and C. Muir, "Predict globally, correct locally: Parallel-in-time optimal control of neural networks," *arXiv preprint arXiv:1902.02542*, 2019.

[20] D. Zhang, T. Zhang, Y. Lu, Z. Zhu, and B. Dong, "You only propagate once: Painless adversarial training using maximal principle," *arXiv preprint arXiv:1905.00877*, 2019.

[21] E. Haber and L. Ruthotto, "Stable architectures for deep neural networks," *Inverse Problems*, vol. 34, no. 1, p. 014004, 2017.

[22] B. Wang, B. Yuan, Z. Shi, and S. J. Osher, "Enresnet: Resnet ensemble via the Feynman-Kac formalism," *arXiv preprint arXiv:1811.10745*, 2018.

[23] L. Zhang, W. E, and L. Wang, "Monge-ampère flow for generative modeling," 2018.

[24] L. Ruthotto and E. Haber, "Deep neural networks motivated by partial differential equations," *Journal of Mathematical Imaging and Vision*, pp. 1–13, 2019.

[25] E. Haber, L. Ruthotto, E. Holtham, and S.-H. Jun, "Learning across scales-a multiscale method for convolution neural networks," *arXiv preprint arXiv:1703.02009*, 2017.

[26] P. Chaudhari, A. Oberman, S. Osher, S. Soatto, and G. Carlier, "Deep relaxation: partial differential equations for optimizing deep neural networks," *Research in the Mathematical Sciences*, vol. 5, no. 3, p. 30, 2018.

[27] P. Chaudhari, A. Choromanska, S. Soatto, Y. LeCun, C. Baldassi, C. Borgs, J. Chayes, L. Sagun, and R. Zecchina, "Entropy-sgd: Biasing gradient descent into wide valleys," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2019, no. 12, p. 124018, 2019.

[28] G. Pavliotis and A. Stuart, *Multiscale methods: averaging and homogenization*. Springer Science & Business Media, 2008.

[29] A. L. Blum and R. L. Rivest, "Training a 3-node neural network is np-complete," *Neural Networks*, vol. 5, no. 1, pp. 117–127, 1992.

[30] I. Safran and O. Shamir, "Spurious local minima are common in two-layer relu neural networks," in *International Conference on Machine Learning*, pp. 4433–4441, PMLR, 2018.

[31] C. Lemaréchal, "Cauchy and the gradient method," *Doc Math Extra*, vol. 251, p. 254, 2012.

[32] H. B. Curry, "The method of steepest descent for non-linear minimization problems," *Quarterly of Applied Mathematics*, vol. 2, no. 3, pp. 258–261, 1944.

[33] R. Sun, "Optimization for deep learning: theory and algorithms," *arXiv preprint arXiv:1912.08957*, 2019.

[34] S. Ghadimi and G. Lan, "Stochastic first-and zeroth-order methods for nonconvex stochastic programming," *SIAM Journal on Optimization*, vol. 23, no. 4, pp. 2341–2368, 2013.

[35] V. S. Borkar and S. K. Mitter, "A strong approximation theorem for stochastic recursive algorithms," *Journal of optimization theory and applications*, vol. 100, no. 3, pp. 499–513, 1999.

[36] M. Welling and Y. W. Teh, "Bayesian learning via stochastic gradient Langevin dynamics," in *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 681–688, 2011.

[37] B. Gidas, "Global optimization via the Langevin equation," in *1985 24th IEEE Conference on Decision and Control*, pp. 774–778, IEEE, 1985.

[38] H. J. Kushner, "Asymptotic global behavior for stochastic approximation and diffusions with slowly decreasing noise effects: global minimization via Monte Carlo," *SIAM Journal on Applied Mathematics*, vol. 47, no. 1, pp. 169–185, 1987.

[39] T.-S. Chiang, C.-R. Hwang, and S. J. Sheu, “Diffusion for global optimization in Rˆn,” *SIAM Journal on Control and Optimization*, vol. 25, no. 3, pp. 737–753, 1987.

[40] C.-R. Hwang, “Laplace’s method revisited: weak convergence of probability measures,” *The Annals of Probability*, pp. 1177–1182, 1980.

[41] S. Geman and C.-R. Hwang, “Diffusions for global optimization,” *SIAM Journal on Control and Optimization*, vol. 24, no. 5, pp. 1031–1043, 1986.

[42] J. Cho, J. Kwon, and B.-W. Hong, “Adaptive regularization via residual smoothing in deep learning optimization,” *IEEE Access*, vol. 7, pp. 122889–122899, 2019.

[43] L. Debnath, *Nonlinear partial differential equations for scientists and engineers*. Springer Science & Business Media, 2011.

[44] C. Baldassi, A. Ingrosso, C. Lucibello, L. Saglietti, and R. Zecchina, “Local entropy as a measure for sampling solutions in constraint satisfaction problems,” *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2016, p. 023301, Feb 2016.

[45] J. D. Cole, “On a quasi-linear parabolic equation occurring in aerodynamics,” *Quarterly of Applied Mathematics*, vol. 9, no. 3, pp. 225–236, 1951.

[46] E. Hopf, “The partial differential equation ut + uux = uxx,” *Communications on Pure and Applied Mathematics*, vol. 3, no. 3, pp. 201–230, 1950.

[47] G. A. Pavliotis, *Stochastic processes and applications: diffusion processes, the Fokker-Planck and Langevin equations*, vol. 60. Springer, 2014.

[48] G. Mil’shtejn, “Approximate integration of stochastic differential equations,” *Theory of Probability & Its Applications*, vol. 19, no. 3, pp. 557–562, 1975.