2018 IEEE Conference on Decision and Control (CDC)
Miami Beach, FL, USA, Dec. 17-19, 2018

Analog implementation of optimization algorithms: a distributed optimization view

Ion Matei, Alexander Feldman and Johan de Kleer

***Abstract*— Digital implementations of optimization algorithms for real time applications are often not fast enough for satisfying performance requirements. Analog computing has the potential to address this challenge, being able to provide an answer in milli- to nano-seconds. In this paper we show how we can leverage the distributed optimization framework to generate analog implementations of optimization algorithms for solving unconstrained optimization problems. We present a mapping of a distributed algorithm to the electrical domain, and discuss practical aspects of the physical realization using electronic circuits.**

I. Introduction

Distributed optimization algorithms are used in applications related to network resource allocation, collaborative control, estimation and identification. In a typical distributed optimization setup we assume that a group of N agents interact with each others through a communication topology modeled as an undirected communication graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Using the usual notations, $\mathcal{V} = \{1, 2, \ldots, N\}$ is the set of nodes and $\mathcal{E} = \{e_{ij} \mid i, j = 1, \ldots, N, i \neq j\}$ is the set of edges. An edge between two nodes i and j means that agents i and j can exchange information (or can cooperate). We denote by $\mathcal{N}_i \triangleq \{j \mid e_{ij} \in \mathcal{E}\}$ the set of neighbors of agent i. We assume a particular type of objective function $f : \mathbb{R}^n \to \mathbb{R}$ formulated as $f(x) = \sum_{i=1}^{N} f_i(x)$. The common goal of the agents is to solve the following optimization problem

$$(P_1) \quad \min_{x \in \mathbb{R}^n} \quad f(x),$$

where we assume that there exists at least one local minimizer. Distributed numerical algorithms are used to solve (P_1) whose convergence rate depends in part on the communication topology. The more connected the agents are, the faster the algorithms converge. There is a plethora of algorithms for setups ranging from unconstrained [2], [5], [13], to constrained [11], [12], [16] or stochastic [7], [9] cases.

Real time applications require results in a short amount of time. This is not always possible using digital devices. Results can obtained significantly faster using analog computers that can potentially generate results in micro- to nano-seconds. Electrical domain is our best bet for implementing an analog device. Electronic circuits are fast and cheap to design and implement. In [6] a nonlinear circuit is designed to solve a constrained non-linear program, but implementing a neural network like architectures. In [15] the authors introduced an analog computer based on electronic circuits for solving constrained quadratic optimization problems. They used it within the model predictive control paradigm. The analog computer was implemented on a PCB using programmable potentiometers and switches.

Our goal is to show how we can build an analog computer for minimizing arbitrary objective functions, by leveraging the distributed optimization setup. It is not trivial to implement an arbitrary function with analog devices. We can however give reasonable approximations, where the objective function is approximated as a sum of other functions. A first idea is to use piecewise linear approximations. For example, in the scalar case $f(x)$ can be approximated as: $f(x) \approx \sum_{i=1}^{N} f_i(x)$, where $f_i(x) = a_i x + b_i$ for $x \in [z_i, z_{i+1}]$, where z_i are points on the domain of f. Alternatively, we can use polynomial interpolations, resulting in the approximation $f(x) \approx \sum_{i=0}^{N} f_i(x)$, with $f_i(x) = a_i x^i$. For such simpler representations, we have a better chance of finding physical realizations of an optimization algorithm. The function approximation fits exactly the distributed optimization setup. Realizing distributed optimization algorithms via analog circuits as compared to centralized versions has the following advantages: makes the PCB less cluttered, more robust to potential faults, (we can add copies of the functions f_i for redundancy), deals better with the uncertainty in components due to the averaging nature of the algorithms, and fits more natural into the networked structure of field-programmable analog arrays (FPAA) [14]. The latter are integrated circuits based on a set of analog building blocks connected through a network of programmable switches similar to the FPGAs of the digital domain. Their main advantage is reconfigurability: by controlling the switches we can generate new electrical circuits. This is particular advantageous in the context of our problem, since we can program the FPAA to work with different cost functions. In this paper, we show how distributed optimization algorithms can be realized with electronic components. In addition, we discuss what building blocks would be necessary in a FPAA to minimize a large class of cost functions.

Paper structure: Section II introduces the distributed optimization framework, the algorithm and a convergence result. In Section III we demonstrate how the distributed algorithm map to realizations in the electric and the mechanical domains. Section IV discusses the realization of the distributed optimization algorithm by electronic components/circuits.

Notations: We denote by $\nabla f(x)$ and $\nabla^2 f(x)$ the gradient and Hessian of $f(x)$ with respect to x, respectively. The notation $\mathbf{x}' = [x_1', \ldots, x_N']$ represents a vector with N vector

Ion Matei, Alexander Feldman and Johan de Kleer are with the Palo Alto Research Center (PARC), Palo Alto, CA (emails: ion.matei@parc.com, afeldman@parc.com, dekleer@parc.com)

components. The i^{th} vector component of $\mathbf{x}$ is x_i. The j^{th} entry of x_i is denoted by x_{ij}. For a matrix $A = (a_{ij})$, a_{ij} represents the entry at row i and column j. Symbol $\otimes$ denotes the Kronecker product.

II. Distributed optimization algorithm

To derive a distributed optimization algorithm we follow the steps we described in [8], [10], [11]. We define the function $\mathbf{F} : \mathbb{R}^{nN} \to \mathbb{R}$ given by $\mathbf{F}(\mathbf{x}) = \sum_{i=1}^{N} f_i(x_i)$, where $\mathbf{x}' = (x_1', x_2', \ldots, x_N')$, with $x_i \in \mathbb{R}^n$. In addition we introduce the vector-valued function $\mathbf{g}(\mathbf{x}) = \left(g_{ij}(\mathbf{x})\right), i \in \{1, \ldots, N\}, j \in \mathcal{N}_i$, with $g_{ij} : \mathbb{R}^{nN} \to \mathbb{R}$ given by $g_{ij}(\mathbf{x}) = d_{ij}(x_i - x_j)$, where d_{ij} are positive scalars. The vector-valued function $\mathbf{g}(\mathbf{x})$ can be compactly expressed as $g(\mathbf{x}) = \mathbf{D}\mathbf{x}$, where $D = (d_{ij})$, $\mathbf{D} = D \otimes I$, with I the n-dimensional identity matrix. We can now define the optimization problem

$$(P_2) \quad \min_{\mathbf{x} \in \mathbb{R}^{nN}} \quad \mathbf{F}(\mathbf{x}), \tag{1}$$

$$\mathbf{g}(\mathbf{x}) = \mathbf{D}\mathbf{x} = 0. \tag{2}$$

We make the following assumptions on the functions $f_i(x)$ and on the communication model.

Assumption 2.1:
1) Functions $f_i(x)$, $i = 1, \ldots, N$ are twice continuously differentiable;
2) Agent i has knowledge of only function $f_i(x)$ and scalars d_{ij}, for $j \in \mathcal{N}_i$;
3) Agent i can exchange information only with agents belonging to the set of its neighbors $\mathcal{N}_i$;
4) The communication graph $\mathcal{G}$ is connected.

We present a set of results leading to the formulation of the first order necessary conditions for (P_2). They are simplified versions of a set of results introduced by us in [8], [10], [11]. The following proposition states that by solving (P_2) we solve in fact (P_1) as well, and vice-versa.

Proposition 2.1 ([10]): Let Assumptions 2.1 hold. The vector x^* is a local minimizer of (P_1) if and only if $\mathbf{x}^* = \mathbb{1} \otimes x^*$ is a local minimizer of (P_2).

Let $\mathbf{x}^* = \mathbb{1} \otimes x^*$ denote a local minimizer of (P_2). From the theory concerning optimization problems with equality constraints, the first order necessary conditions for (P_2) ensure the existence of the scalar λ_0 and vector $\boldsymbol{\lambda}^*$ so that $\lambda_0^* \nabla \mathbf{F}(\mathbf{x}^*) + \mathbf{D}'\boldsymbol{\lambda}^* = 0$.

Note that since $\mathbf{D}$ is not full rank, the uniqueness of $\boldsymbol{\lambda}^*$ cannot be guaranteed. The following result characterizes the set of Lagrange multipliers verifying the first order necessary conditions of (P_2).

Proposition 2.2 (first order necessary conditions for (P_2)): [10], [11] Let Assumptions 2.1 hold and let $\mathbf{x}^* = \mathbb{1} \otimes x^*$ be a local minimizer for problem (P_2). There exists a unique vector $\boldsymbol{\lambda}^* \in \text{Range}(\mathbf{D})$ so that

$$\nabla \mathbf{F}(\mathbf{x}^*) + \mathbf{D}'\boldsymbol{\lambda} = 0,$$

for all $\boldsymbol{\lambda} \in \{\boldsymbol{\lambda}^* + \boldsymbol{\lambda}_\perp \mid \boldsymbol{\lambda}_\perp \in \text{Null}(\mathbf{D}')\}$.

To find a solution of problem (P_2) we solve the set of necessary conditions:

$$\nabla \mathbf{F}(\mathbf{x}) + \mathbf{D}'\boldsymbol{\lambda} = 0, \tag{3}$$

$$\mathbf{D}\mathbf{x} = 0. \tag{4}$$

Solving (3)-(4) does not guarantee finding a local minimizer, but at least the local minimizers are among the solutions of the above nonlinear system of equations. A first order method for solving (3)-(4) is:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha\left[\nabla \mathbf{F}(\mathbf{x}_k) + \mathbf{D}'\boldsymbol{\lambda}_k\right], \tag{5}$$

$$\boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k + \alpha \mathbf{D}\mathbf{x}_k, \tag{6}$$

where $\alpha > 0$ is chosen small enough to ensure the stability of the algorithm.

Local convergence of the difference equations (5)-(6) is addressed in the next theorem. It states that under some assumptions on the functions $f_i(x)$, provided the initial values are close enough to a solution of the first order necessary conditions of (P_2), and a small enough step-size α is used, the sequence $\{\mathbf{x}_k, \boldsymbol{\lambda}_k\}$ converges to this solution.

Theorem 2.1 ([10], [11]): Let Assumptions 2.1 hold and let $(\mathbf{x}^*, \boldsymbol{\lambda}^*)$ with $\boldsymbol{\lambda}^* \in \text{Range}(\mathbf{D})$, be a local minimizer-Lagrange multipliers pair of (P_2). Assume also that $\nabla^2 \mathbf{F}(\mathbf{x}^*)$ is positive definite. Then there exists $\bar{\alpha}$, such that for all $\alpha \in (0, \bar{\alpha}]$, the set $(\mathbf{x}^*, \boldsymbol{\lambda}^* + \text{Null}(\mathbf{D}'))$ is an attractor of iteration (5)-(6) and if the sequence $\{\mathbf{x}_k, \boldsymbol{\lambda}_k\}$ converges to the set $(\mathbf{x}^*, \boldsymbol{\lambda}^* + \text{Null}(\mathbf{D}'))$, the rate of convergence of $\|\mathbf{x}_k - \mathbf{x}^*\|$, and $\|\boldsymbol{\lambda}_k - [\boldsymbol{\lambda}^* + \text{Null}(\mathbf{D}')]\|$ is linear.

III. Physical domain mappings

In this section we discuss physical implementations for finding the solution of (3)-(4). The continuous time version of (5)-(6) is

$$\dot{\mathbf{x}} = -\left[\nabla \mathbf{F}(\mathbf{x}) + \mathbf{D}'\boldsymbol{\lambda}\right], \tag{7}$$

$$\dot{\boldsymbol{\lambda}} = \mathbf{D}\mathbf{x}. \tag{8}$$

We are interested in analog devices whose stationary regimes represent the solution of the optimization problem. With the right choice of components, they have the potential to converge to a solution faster than digital implementations. We leverage the distributed structure of the optimization problem to come up with analog implementations based on relative simple components. We give two examples of analog implementations: in the electrical and mechanical domains. The latter is for theoretical purposes mainly, since electrical circuits are cheaper to implement and can reach steady state much faster.

A. Electrical domain

To demonstrate how we can implement the distributed optimization algorithm in the electrical domain, consider the circuit in Figure 1. It has two inductors, three capacitors

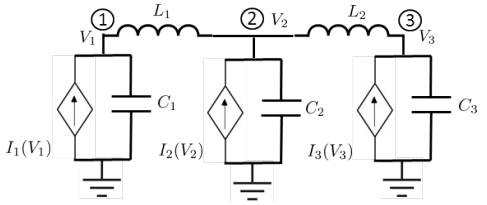

Fig. 1: Electric circuit composed of a network of inductors together with capacitors and voltage controlled current sources

and three voltage controlled current sources (VCCSs). The currents through the inductors are denoted by I_{L_1} and I_{L_2}, while the currents through the capacitors are denoted by I_{C_1}, I_{C_2} and I_{C_3}, respectively. The behavior of the circuit (with some implicit current directions) is described by the following equations:

$$\begin{aligned} C_1\frac{dV_1}{dt} &= -I_1(V_1) - I_{L_1}, \\ C_2\frac{dV_2}{dt} &= -I_2(V_2) - I_{L_2} + I_{L_1}, \\ C_3\frac{dV_3}{dt} &= -I_3(V_3) + I_{L_2}, \\ L_1\frac{dI_{L_1}}{dt} &= V_1 - V2, \\ L_2\frac{dI_{L_2}}{dt} &= V_2 - V3. \end{aligned}$$

Assuming zero initial conditions for the inductor currents and equal inductance and capacitance values, the differential equations for the capacitor voltages can be written as

$$\begin{aligned} \frac{dV_1}{dt} &= -\left[\frac{1}{C}I_1(V_1) + \frac{1}{LC}\int_0^t (V_1 - V_2)d\tau\right] \\ \frac{dV_2}{dt} &= -\left[\frac{1}{C}I_2(V_2) + \frac{1}{LC}\int_0^t (2V_2 - V_3 - V1)d\tau\right] \\ \frac{dV_3}{dt} &= -\left[\frac{1}{C}I_3(V_3) + \frac{1}{LC}\int_0^t (V_3 - V2)d\tau\right], \end{aligned}$$

or in matrix form

$$\frac{d\boldsymbol{V}}{dt} = -\frac{1}{C}\boldsymbol{I}_v(\boldsymbol{V}) - \frac{1}{LC}\mathcal{L}\int_0^t \boldsymbol{V}d\tau, \tag{9}$$

where $\boldsymbol{V}' = [V_1, V_2, V_3]$, $\boldsymbol{I}_v(\boldsymbol{V})' = [I_1(V_1), I_2(V_2), I_3(V_3)]$, and $\mathcal{L}$ is a Laplacian like matrix

$$\mathcal{L} = \begin{bmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix},$$

The Laplacian can be decomposed as $\mathcal{L} = \frac{1}{2}\mathcal{D}'\mathcal{D}$, with

$$\mathcal{D} = \begin{bmatrix} 1 & -1 & 0 \\ -1 & 1 & 0 \\ 0 & 1 & -1 \\ 0 & -1 & 1 \end{bmatrix}.$$

Introducing $\boldsymbol{\lambda} = \frac{1}{\sqrt{2LC}}\mathcal{D}\int_0^t \boldsymbol{V}(\tau)d\tau$, we have

$$\frac{d\boldsymbol{V}}{dt} = -\frac{1}{C}\boldsymbol{I}_v(\boldsymbol{V}) - \frac{1}{\sqrt{2LC}}\mathcal{D}'\boldsymbol{\lambda}, \tag{10}$$

$$\frac{d\boldsymbol{\lambda}}{dt} = \frac{1}{\sqrt{2LC}}\mathcal{D}\boldsymbol{V}, \tag{11}$$

By making the additional notations $\mathbf{x} = \boldsymbol{V}$, $\mathbf{D} = \frac{1}{\sqrt{2LC}}\mathcal{D}$ and $\frac{1}{C}\boldsymbol{I}_v(\boldsymbol{V}) = \nabla\mathbf{F}(\mathbf{x})$, we recover the continuous dynamics (7)-(8), for the scalar case. We recall that we can scale the cost function without affecting the minimum: if $\mathbf{x}^*$ minimizes $\mathbf{F}(\mathbf{x})$ then it minimizes $c\mathbf{F}(\mathbf{x})$ as well, for any positive c. In other words, with $\boldsymbol{I}_v(\boldsymbol{V}) = C\nabla\mathbf{F}(\mathbf{x})$ we do not affect the steady state, which protects us from having to generate unreasonable current values. Hence we can choose C and L such that the circuit reaches steady state fast. Therefore we have that the optimization variables correspond to capacitor voltages, the Lagrangian variables $\boldsymbol{\lambda}$ correspond to (copies) of inductor currents, and the gradients correspond to scaled VCCSs. This idea can be extended to the multivariable case. Figure 2 builds on the previous example and depicts the electric circuit for solving an optimization variable with a cost function with two variables. Basically, we have two circuits with the same structure coupled through the capacitor potentials that control the current sources.

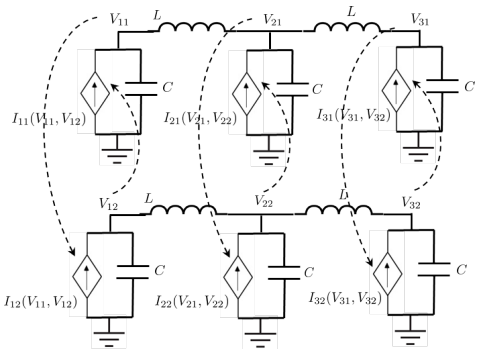

Fig. 2: Electric circuit for solving a distributed optimization problem with a cost function with two variables

For n optimization variables, the structural map from the communication graph in the distributed optimization setup to the circuit that solves the optimization problem is shown in Figure 3. The electric circuit has n topological identical

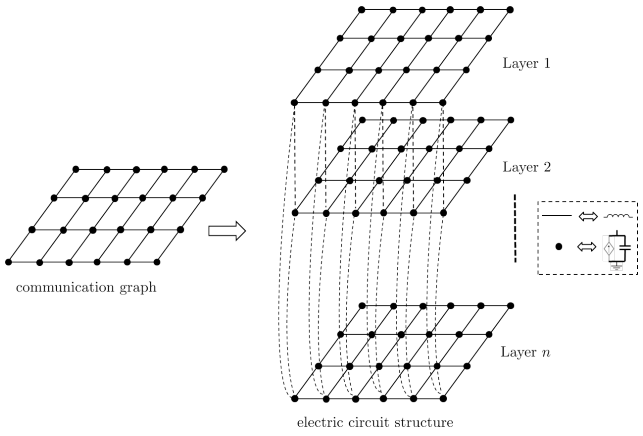

Fig. 3: Mapping of the communication graph to the electric circuit structure: a node in the communication graph maps to a capacitor and a VCCS in parallel; a link in the communication graph maps to an inductor.

layers. For each vertex in the communication graph we have a pin in the electric circuit at each layer. For each edge e_{ij} we have an inductor between pins i and j at every layer. Finally, each pin is connected to a capacitor and a current source in parallel. The current source is controlled by the capacitors' potentials at similar positions in the layer. This is depicted with dotted connections in Figure 3, where for reducing the clutter in the figure we showed the dotted

connections for bottom, border nodes only. More formally, let $\mathbf{x}' = [x_1', \ldots, x_N']$, with $x_i \in \mathbb{R}^n$ and x_{ij} the j^{th} entry of x_i. In the electrical circuit, j corresponds to a layer j, and i corresponds to pin i. The variable x_{ij} corresponds to the potential of pin i at layer j. We recall that the gradient of the cost function in (P_2) can be written as $\nabla \mathbf{F}(\mathbf{x})' = [\nabla f_1(\mathbf{x})', \ldots, \nabla f_N(\mathbf{x})'] = [\nabla f_1(x_1)', \ldots, \nabla f_N(x_N)']$. The j^{th} entry of $\nabla f_i(x_i)$ is denoted by $\nabla f_{ij}(x_i)$ and it corresponds to the voltage controlled current source at pin i and layer j. Note that $\nabla f_{ij}(x_i)$ is a function of x_i and hence of all potentials at pin i across all n layers.

1) Example: We introduce an example depicting the electric circuit for minimizing a 4^{th} order degree polynomial function $f(x) = a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$, with $a_4 = 0.2$, $a_3 = 0.05$, $a_2 = 1.5$, $a_1 = -1$ and $a_0 = 0.7$. We can write the cost function of the augmented optimization problem as $\mathbf{F}(\mathbf{x}) = f_4(\mathbf{x}) + f_3(\mathbf{x}) + f_2(\mathbf{x}) + f_1(\mathbf{x})$ with $f_4(\mathbf{x}) = a_4x_4^4$, $f_3(\mathbf{x}) = a_3x_3^3$, $f_2(\mathbf{x}) = a_2x_1^2$, and $f_1(\mathbf{x}) = a_1x_1 + a_0$. The electric circuit used to calculate the minimizer of $f(x)$ is shown in Figure 4, where we chose a grid like topology. The more connected the nodes are, the fastest convergence in reached. When choosing the topology for the electrical circuit, manufacturing constraints must be taken into account though. The current sources must implement the gradients

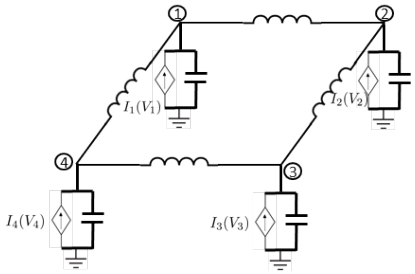

Fig. 4: Electric circuit for minimizing $f(x) = a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$

for the functions f_i: $I_1(V_1) = ca_1$, $I_2 = c2a_2V_2$, $I_3 = c3a_3V_3^2$, $I_4 = c4a_4V_4^3$, where c is a scaling parameter that ensures the currents are within realizable physical limits. The technical challenge is the implementation of the nonlinear VCCSs. We will discuss how they can be physically implemented in Section IV. Figures 5, 6 and 7 show the capacitor potentials, the inductor currents and the currents from the VCCSs respectively, for $C = 10\mu F$, $L = 1mH$, $\alpha = 0.1$, and initial potential conditions $V_{10} = 1V$, $V_{20} = -1V$, $V_{30} = 2V$ and $V_{40} = -0.2V$. By choosing appropriate values for L, C and c we can achieve a reasonable trade-off between transient time of the circuit and the physical realizability of the circuit.

IV. Physical implementation: electrical domain

For non-convex cases, iterations (5)-(6) converge to one of the possibly multiple local minimizers based on the initial values. An avenue for finding the best such local minimizer is to try several initial conditions and evaluate the objective function. This procedure can be numerically expensive. We

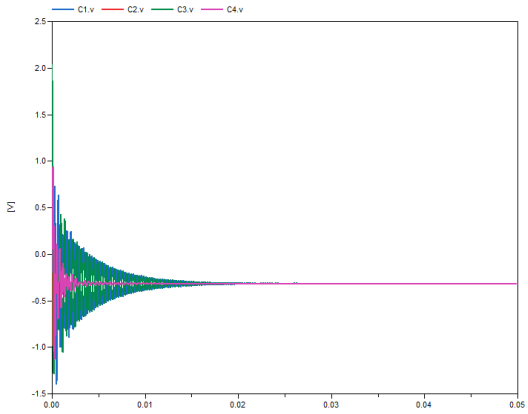

Fig. 5: Capacitor potentials

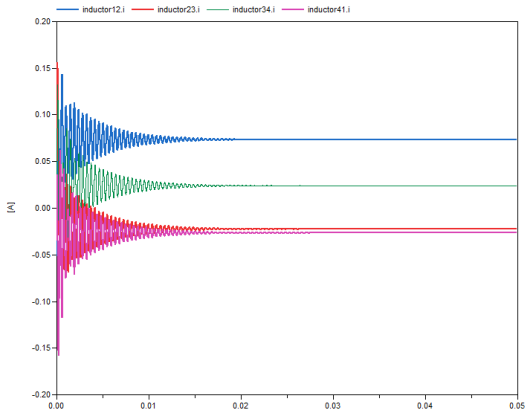

Fig. 6: Inductor currents

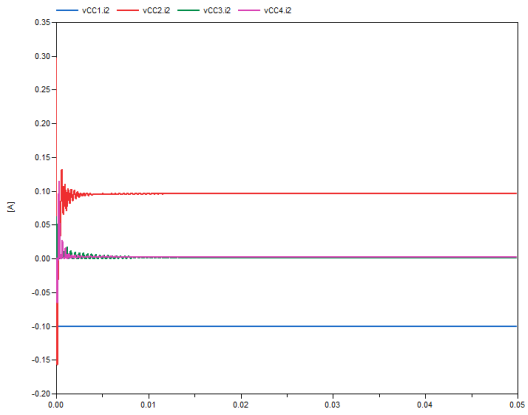

Fig. 7: Voltage controlled sources

can overcome this challenge by building a physical device capable of finding local minimizers very fast. In the context of the electrical circuit discussed in Section III-A, changing the initial conditions amounts to setting different values for the initial potentials of the capacitors.

This section discusses a practical realization of an electrical circuit that can be used to solve distributed optimization problems. The circuit consists of an interconnected topology

of analog cells, where the topology is envisioned to be generated through a programmable FPAA. The FPAA will contain a set of building blocks. One such building block is the voltage controlled current source cell. Figure 8 shows a single cell implementing a linear function $g(x) = ax$, where $g(x)$ is a current quantity and x is a voltage quantity. The function $g(x)$ can be thought as the gradient of a quadratic function that is a term in the cost function. The implementation is based on the well-known Howland current pump [4].

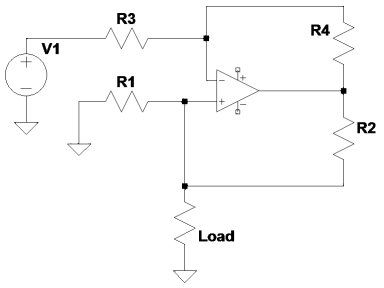

Fig. 8: Howland current pump

The current through the load is $I_{load} = \frac{1}{R_1} V_1$. Constant current source can be implemented by using the same Howland current pump, connected to a constant voltage source and choosing appropriate values for the resistors to realize the desired current output. To achieve the linear relation between the output current and the input voltage, the current pump design requires that $\frac{R_1}{R2} = \frac{R_3}{R_4}$).

Figure 9 shows three cells connected to a bus. It is the implementation of the ideal model of the circuit shown in Figure 1, when the terms in the cost function are quadratic. The steady state voltage measured at the negative inputs of the op-amps is the root of the equation $g(x) = (a_1x + b_1) + (a_2x + b_2) + (a_3x + b_3)$.

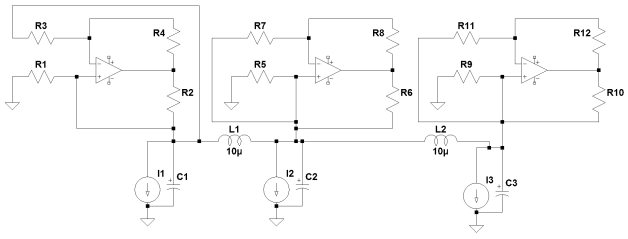

Fig. 9: Three linear function cells connected to a bus

In a practical implementation, we would replace the current source I_1 with another "constant" cell which is also a Howland current pump with a constant input voltage. A physical implementation of the FPAA will use non-ideal operational amplifiers (OP AMP). For the discrete component proof-of-concept we use common dual precision amplifiers such as OPA207. The amplifiers will be powered from a clean voltage source and we cannot use them in saturation mode. This gives us ranges for the optimization problems. These ranges can be pre-computed when the design of the FPAA chip is finished. Another consideration is that we want to avoid too large or too small currents through the feedback of the op-amps. This restricts us in the choice of the resistor gains.

The Howland current pump implements linear relations between the input voltage and the output current. If we have higher order terms in the cost function, additional components are required as part of the FPAA. Figure 10 shows a circuit that implements a square function for an electric potential [3]. Therefore, by preceding the Howland current pump with this circuit we can implement gradients that contain second order polynomials. We can go even further, and use circuits that implement cube functions [3] as shown in Figure 11. For example a gradient of the form $g(x) = a_3x^3 + a_2x^2 + a_1x$ can be implemented using the circuit shown in Figure 12. To implement cells that compute second

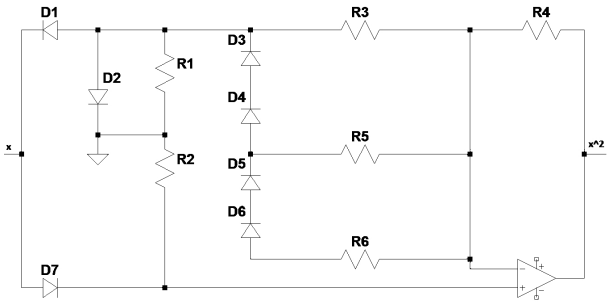

Fig. 10: Implementation of a square function

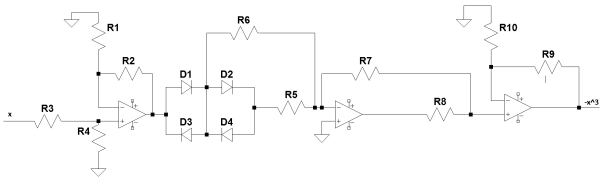

Fig. 11: Implementation of the cube function

and higher powers another option is to use integrated circuits such as the analog computational unit AD538 [1]. This integrated circuit provides precision analog multiplication, division, and exponentiation. The functional block diagram is shown in Figure 13. The transfer function realized by the circuit is $V_O = V_Y(V_Z/V_X)^m$, where m can be set to take values between 0.2 and 5. More details on how this circuit can be configured can be found in [1].

We showed how we can implement various gradient functions using Howland current pumps when combined with circuits implementing multiplication operations. To allow for programmable cost functions, the FPAA will be configured digitally. It will include precision Digital-to-Analog Converters, JFET-based analog digipots and other established electronics design patterns. This will enable programming different resistance values, hence enabling the FPAA to implement different cost functions. We also envision replacing

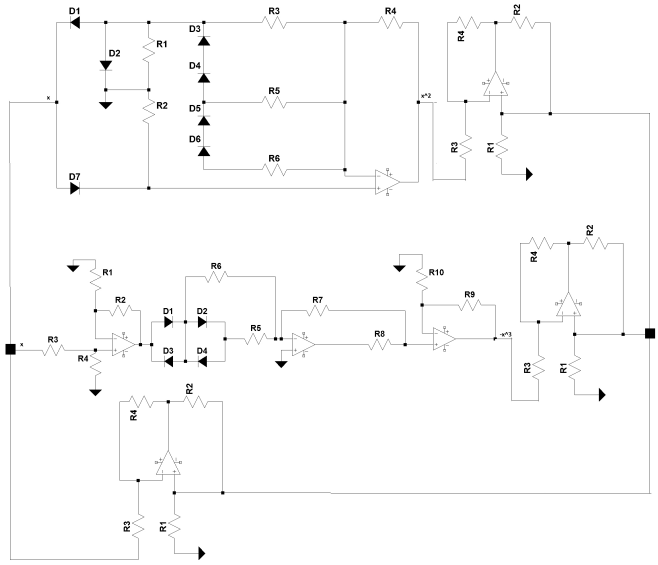

Fig. 12: Implementation of a current source whose dependence of the input voltage is $i(v) = a_3v^3 + a_2v^2 + a_1v$

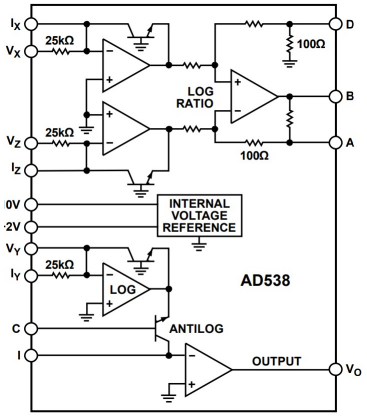

Fig. 13: Functional block diagram of AD538 computational unit [1]

the inductors with equivalent OP AMP based implementations to decrease the size of the circuit and improve accuracy. The topologies constructed using the FPAA are realized by acting of a set of programmable switches. The same switches will be used to charge the capacitors associated to each term in the cost function to allow for exploration of multiple initial conditions.

There are physical limitations to what voltage and current values can be achieved in an analog circuit. Components are usually rated for bounded voltage and currents intervals. Hence the optimization problem may need to be scaled to fit such restrictions. Detailed analysis of the FPAA circuit is beyond the scope of this paper but the basics for building the elementary computational elements were discussed above. The actual implementation of the FPAA chip will depend on additional considerations such as feature (transistor) size, number of layers, materials, etc. A grid like topology seems the most appropriate choice.

V. Conclusions

In this paper we argued that distributed optimization algorithms are a feasible avenue for creating analog devices that solve optimization problems. Analog devices have the advantage of being much faster compared to digital circuits, when appropriate component parameter values are chosen. We showed how we can realize a distributed optimization algorithm with an analog circuit for single variable cost functions. In addition, we discussed how the analog circuit architecture would look like for multivariable cost functions. We discussed how we can implement the gradients of the functions in the cost function using Howland current pumps combined with circuits executing multiplication operations. Our next step is to design and implement an FPAA architecture that would enable analog circuit designs aimed at solving optimization problems with arbitrarily cost functions.

References

[1] Analog Devices. *Real-Time Analog Computational Unit (ACU) - AD538*, 2017. Rev. E.

[2] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.*, 3(1):1–122, January 2011.

[3] A. Buscarino, L. Fortuna, M. Frasca, and G. Sciuto. *A Concise Guide to Chaotic Electronic Circuits*. SpringerBriefs in Applied Sciences and Technology. Springer International Publishing, 2014.

[4] Sergio Franco. *Design with operational amplifiers and analog integrated circuits*. McGraw-Hill series in electrical and computer engineering, 2015.

[5] B. Johansson, T. Keviczky, M. Johansson, and K.H. Johansson. Subgradient methods and consensus algorithms for solving convex optimization problems. *Proceedings of the 47th IEEE Conference on Decision and Control*, pages 4185–4190, Dec 2008.

[6] M. P. Kennedy and L. O. Chua. Neural networks for nonlinear programming. *IEEE Transactions on Circuits and Systems*, 35(5):554–562, May 1988.

[7] I. Lobel and A. Ozdaglar. Distributed subgradient methods for convex optimization over random networks. *Automatic Control, IEEE Transactions on*, 56(6):1291–1306, june 2011.

[8] I. Matei and J. S. Baras. Distributed nonlinear programming methods for optimization problems with inequality constraints. In *CDC*, pages 2649–2654. IEEE, 2015.

[9] I. Matei and J.S. Baras. Performance evaluation of the consensus-based distributed subgradient method under random communication topologies. *Selected Topics in Signal Processing, IEEE Journal of*, 5(4):754–771, aug. 2011.

[10] I. Matei and J.S. Baras. Distributed algorithms for optimization problems with equality constraints. In *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, pages 2352–2357, Dec 2013.

[11] I. Matei, J.S. Baras, M. Nabi, and T. Kurtoglu. An extension of the method of multipliers for distributed nonlinear programming. In *Decision and Control (CDC), 2014 IEEE 53nd Annual Conference on*, pages 6951–6956, Dec 2014.

[12] A. Nedic, A. Ozdaglar, and P.A. Parrilo. Constrained consensus and optimization in multi-agent networks. *IEEE Trans. Autom. Control*, 55(4):922–938, Apr 2010.

[13] A. Nedic and A. Ozdalgar. Distributed subgradient methods for multi-agent optimization. *IEEE Trans. Autom. Control*, 54(1):48–61, Jan 2009.

[14] E. Pierzchala, G. Gulak, L. Chua, and A. Rodríguez-Vázquez. *Field-Programmable Analog Arrays*. Springer US, 1998.

[15] S. Vichik. *Quadratic and linear optimization with analog circuits*. PhD thesis, University of California, Berkely, 2015.

[16] M. Zhu and S. Martinez. An approximate dual subgradient algorithm for multi-agent non-convex optimization. *Automatic Control, IEEE Transactions on*, 58(6):1534–1539, June 2013.