Analog Solutions of Systems of Linear Equations on a Configurable Platform

Aishwarya Natarajan and Jennifer Hasler
Georgia Institute of Technology, anatarajan35@gatech.edu

***Abstract*—Even though analog computation is better suited for differential equation solutions (ODE, PDE), sometimes it needs to solve systems of linear equations. This discussion focuses on analog solutions of linear equation systems, implemented on a configurable platform. Digital systems rely on solving linear equations as the fundamental numerical computation. Systems of linear equations are used to solve static circuits illustrating that at least a reduced class of analog physical linear system computation should be possible. The analog approaches utilize iterative techniques, setting up a set of ODEs to solve the system of linear equations, rather than relying on matrix decompositions (e.g. LU decomposition). The approach allows for multiple potential configurable circuit approaches. A set of amplifier networks has been designed to demonstrate the solutions for different matrices. These techniques provide energy-efficient continuous-time solutions. The resulting algorithm has been studied considering the analog numerical analysis for the solution and convergence time.**

***Index Terms*—Analog solutions of linear equations**

Solution of linear equations is the fundamental numerical computation for digital systems (Fig. 1), and simultaneously not well aligned with analog computation [1]. Solutions of linear systems through matrix decomposition is the most straight-forward method for digital computation, and yet is the most challenging option for analog approaches [1]. Solution of linear systems is ubiquitous for digital solutions (Fig. 1). Entire numerical software tools (e.g. MATLAB: Matrix solution Laboratory) are dedicated to these solutions, and benchmarks for computing are based on solving systems of linear equations (e.g. LINPACK [2]). Linear system solution requires the solution to the linear algebraic matrix equation

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{1}$$

where **A** is the input matrix, **b** is the input vector, and **x** is the solution vector. The classic digital solution method is a form of Gaussian elimination to decompose **A** into a product of a lower triangular matrix, **L**, and an upper triangular matrix, **U**. Once this solution is found, the solutions can be directly solved, first using **L** and **b**, and then using this result and **U**. This computational method shows all of the strength of symbolic digital processing, including the pivot stages, to get the maximum accuracy for the decomposition. Analog decomposition into **L** and **U** is extremely challenging, with numerous integer steps requiring intermediate value storage at high resolution, and memory manipulations (e.g. pivots) to retain as much final precision as possible. The lower analog precision, particularly through short-term sampled registers, makes this process incredibly challenging [1].

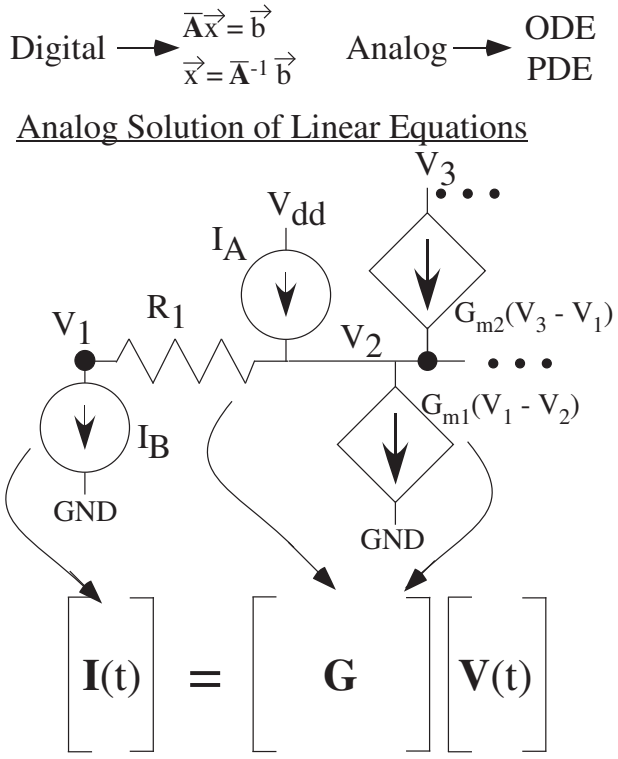

Fig. 1. Solutions of linear systems are the fundamental operations for digital computation, whereas solutions of Differential Equations (ODE, PDE) are the fundamental operations for analog computation. Analog solutions of linear systems would enable a wider range of analog capability. One approach is recognizing that the steady state solution of linear circuits with controlled sources can represent these networks; configurable analog systems can directly implement such models using voltage-controlled current sources.

On the other hand, Engineering students (e.g. GT's ECE 2040) are taught that systems of linear equations, like nodal equations, are used to solve static circuits resistive circuits *with independent and dependent voltage- and current-sources* (e.g. Fig. 1). Therefore, the steady state of linear circuits could solve systems of linear equations (e.g. Fig. 1). Electronic circuits are used to illustrate solutions (e.g. Hopfield networks [3], [4]), simulating linear solutions for a reduced class of problems [5], [6], and considering specialized cases (e.g. elliptic PDEs [7], [8] or resistor-only concepts [9], [10], [11], [12]). This discussion focuses on a general algorithmic and numerical methods for analog computation of linear system of equations through the implementation and experimental measurements from a large-scale Field Programmable Analog Array (FPAA). [13].

978-1-7281-3320-1/20/$31.00 ©2020 IEEE

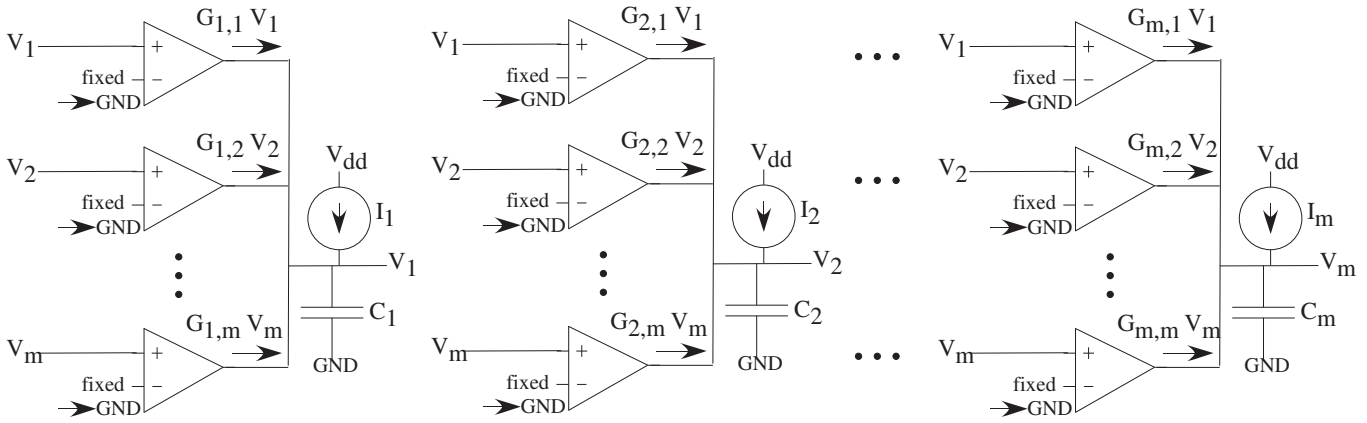

Fig. 2. A potential continuous-time circuit architecture to solve systems of linear equations is shown. Linear matrix solution is built from Transconductance Amplifiers (OTA), which are elements of the Computational Analog Block (CAB) on the SoC. OTAs with Floating-Gate (FG) bias currents make up the individual conductances allowing for a general constraint matrix to solve. The input vectors are fed into the gate input of the 9 transistor OTA, which acts as a current source, while the offset for the reference voltages can be controlled by DACs compiled on the Array.

The paper focuses on solving systems of linear equations using analog computation. The discussion moves towards iterative methods for solving linear equations, good methods for analog and digital computation, and its potential configurable circuit approaches. The discussion then moves towards analyzing the algorithmic issues and analog numerical analysis issues of this approach. These techniques provide energy-efficient continuous-time solutions. Even if an ideal solution exists, one needs confidence in the overall accuracy and convergence time.

I. Physical Computation Linear Equation Solutions → Iterative Matrix Techniques

Linear systems are also solved by iterative techniques, using coupled equations that converge $\mathbf{x}$ to the desired solution. Iterative digital techniques are sometimes faster and have fewer operations than digital Gaussian elimination, particularly for sparse matrix solutions. As one should simply use physical techniques to solve ODE / PDE applications [14], we focus on physical solution systems of linear systems from different applications.

A physical approach will be aligned with iterative matrix equations. Physical computing solutions of linear systems are iterative techniques, where the *iterations* result from coupled differential equations whose steady state represents the desired output ($\mathbf{x}$). One might modify (1) to build the differential equation

$$\tau \frac{d\mathbf{x}}{dt} + \mathbf{A}\mathbf{x} = \mathbf{b}, \tag{2}$$

where τ is the time-constant for the network. Each row could have a different τ, although we present the simpler case for clarity. Digital (sampled time) approaches would be written as

$$\mathbf{x}[n] = \mathbf{x}[n-1] + \epsilon\left(\mathbf{b} - \mathbf{A}\mathbf{x}[n]\right), \tag{3}$$

effectively approximating the time derivative of $\mathbf{x}$. These techniques are related to gradient-descent, Jacobi, and Gauss-Seidel iterative matrix solutions [15].

The dynamics of (2) are seen along the eigenspace corresponding to $\mathbf{A}$. We define the eigenvalue / eigenvector of $\mathbf{A}$ as

$$\mathbf{A} = \mathbf{E}\mathbf{\Lambda}\mathbf{E}^{-1} \tag{4}$$

where $\mathbf{\Lambda}$ is a diagonal matrix of eigenvalues, and $\mathbf{E}$ are the corresponding rows of normalized (power = 1) eigenvectors corresponding to the particular eigenvector. By projecting (1) and (2) on the eigenvector basis in (4), we get

$$\mathbf{x} = \mathbf{E}\mathbf{y}, \tau\frac{d\mathbf{y}}{dt} + \mathbf{\Lambda}\mathbf{y} = \mathbf{E}^{-1}\mathbf{b}, \tag{5}$$

This ODE requires positive $\mathbf{\Lambda}$ values requiring that $\mathbf{A}$ be positive definite. Non positive definite matrices can be solved through multiple transformations [6], [16], such as multiplying (2) by $\mathbf{A}^T$ resulting in a positive definite matrix [16]. Therefore, this method is general enough for these solution methods.

II. Linear Equation Solution using Transconductance Amplifiers

We solve these linear system of equations employing Transconductance Amplifiers (TA) in an architecture shown in Fig. 2. One can achieve a larger set of matrix equations utilizing TAs, enabling a richer set of potential matrices. The linear region of an OTA is expressed as

$$I_{out} = G(V^+ - V^-) \tag{6}$$

For an OTA programmed with subthreshold bias currents (I_{bias}), the resulting coupling between nodes (k,l) would be $G_{k,l} = \frac{\kappa I_{bias}}{U_T}$. The discussion holds when utilizing above threshold currents with a modification of the underlying function for G.

Figure 2 shows the resulting general network to solve (1), where the l row is written as

$$C\frac{dV_l}{dt} = I_l - \sum_{k=1}^{m} G_{l,k}V_k \tag{7}$$

and by comparing with (2), $a_{l,k} \propto G_{l,k}$ and $b_k \propto I_k$. The timeconstant, τ is the load capacitance divided by a

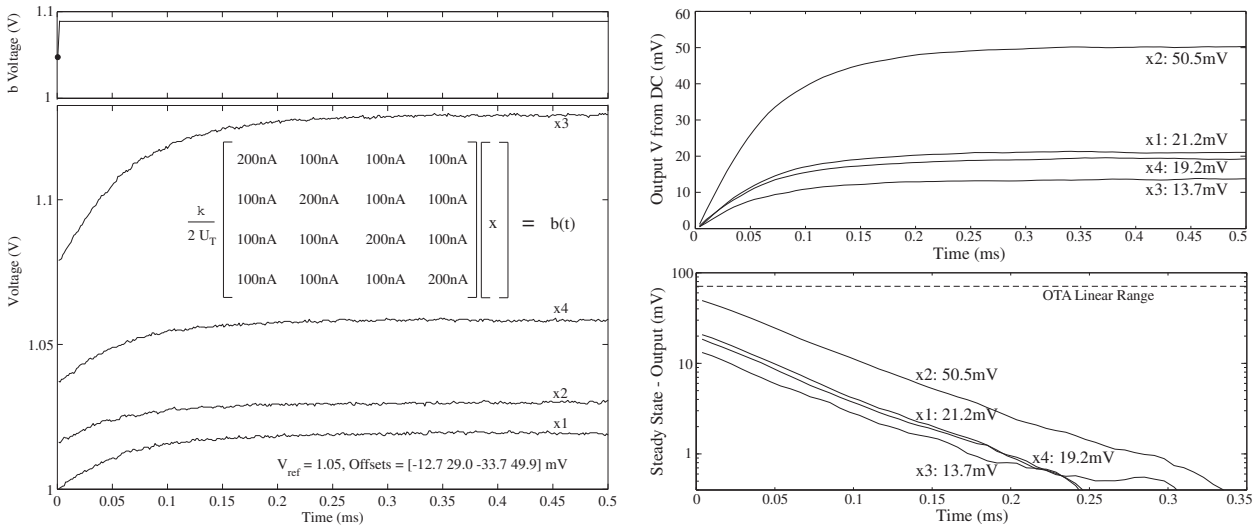

Fig. 3. Analysis for the convergence of $\mathbf{A}$ matrix with 200nA and 100nA as the diagonal and off-diagonal elements respectively, is shown. For a 40mV $\mathbf{b}$ input applied as current sources, the x(t) solution is plotted and is within the linear range of the OTAs. The time constant is 61μs curve-fitting the exponential curves from the step responses. One of the components along the eigenvalue that is larger by a factor of 5, converges 5 times faster in 12μs.

normalizing conductance, often related to a larger diagonal element of $\mathbf{A}$. For a particular normalizing subthreshold bias current, I_{bias}, $\tau = \frac{CU_T}{\kappa I_{ref}}$. One could have a different τ per row due to different conductance modeling and capacitances, and potentially tuned to optimize convergence. Current sources set **b** inputs where positive current sources would go to V_{dd}, and negative current sources would go to GND. These current sources could also be implemented as additional TAs with controlling input voltages.

III. Measured Dynamics on the SoC FPAA

This approach is implemented in an SoC FPAA device [13]. The linear equations are solved by 9-transistor TA elements in Computational Analog Blocks (CAB) [13]. The **A** matrix is set by the conductances of the individual OTAs, that in turn are set by Floating-Gate (FG) bias currents. A TA is used to convert the input voltage (**b**) signal to a current. The reference voltage(s) can be controlled by DACs compiled on the FPAA [13]. A 4x4 linear system solution is compiled and programmed on the FPAA using 16 TAs to implement the 4x4 **A** matrix, and 4 TAs to implement the **b** matrix. The sign of each **A** matrix element determines the TA input sign, where positive values are input in the - input and the reference to the + input, while negative values are input in the + input and the reference to the - input. The outputs, **x**(t) can be scanned and buffered out to be measured.

Figure 3 shows the dynamics of the matrix solutions, demonstrated through a 4x4 network. The **b** vectors are inputs fed to the gate input of the TA structure. In particular, the inputs are step functions. They are step inputs, starting from *zero* (= fixed potential) and moving to a single input value for **b**. The matrix solution outputs, *x(t)* , show the dynamics when new inputs are applied to different conductance matrices. The time constant is roughly the diagonal TA bias current and the FPAA routing capacitive load, and is related to any mismatch in the resulting components, both for small and large signal sizes. Additional **A** matrices can be programmed and dynamics measured.

Figure 3 analyzes the convergence and resulting eigenvalue timeconstants from the step responses. The dynamics are studied by plotting the matrix solution outputs, **x**(t) which are the steady-state responses. In case of the matrix with 200nA diagonal elements, the effective time constant is 61μs from the three eigenvalues, obtained through curve-fitting the exponential curves from the step responses. Since one of the eigenvalues is larger by a factor of 5, it converges 5 times faster, in 12μs. The time constant is 47μs for the diagonal matrix with 1μA as the *1* element and 10nA as the *0* elements.

The effective time constant is related to the minimum eigenvalues, resulting in the largest value for settling to a solution. Other terms will settle faster, and depending on the particular problem and method, the τ for each row could be adapted to optimize convergence. Larger eigenvalues converge faster. Smaller eigenvalues that correspond to the smaller changes in the solution (signal power), have little changes overall, and therefore, not in the required precision. Hence, it is not important to wait for them. One could visualize the adaptation of τ to accommodate for λ, because, if (2) is stable for one set of τ, it is stable for all positive τ.

Another way to understand the ODE of the linear equation solver as a dynamical system or fully coupled filters, is typically as a set of low-pass filters. Figure 4 shows the response of the network to a chirp input, sweeping between frequencies from 1Hz to 20kHz over a time duration of 1ms. These approaches give a new way of approaching the solution of linear equations as effectively filter design, consistent with an approach typically seen in control system implementations. The dynamics of the output at one tap show the higher frequency components being attenuated, which is consistent with low pass behavior.

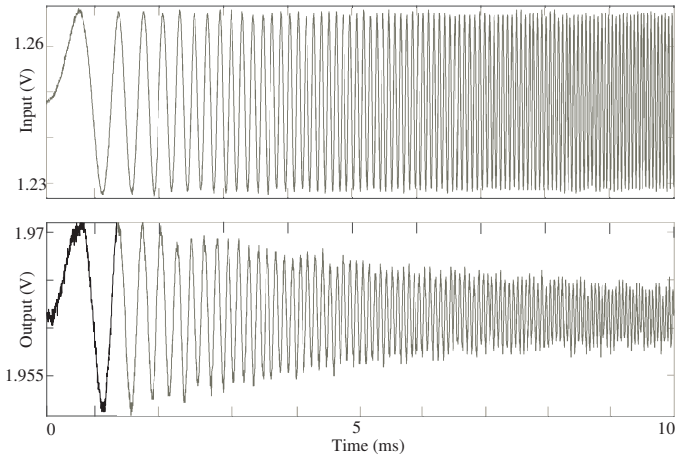

Fig. 4. A chirp input of 20mV offset about 1.25V is applied as the b(t) vector for a set of currents to the 4x4 OTA network structure. The output response is shown, where one could observe the higher frequencies being attenuated, thereby effectively behaving like a Low-pass filter. The linear equation solver can be described as a filter response, thereby connecting to control theory and filter design concepts.

IV. Linear Equation Solver Architecture Complexity

Digital systems utilize a number of techniques for solving linear systems based on a range of potential applications. The complexity for solving diagonal matrices, whether by analog or digital techniques is similar. Upper or Lower diagonal matrices (e.g. **L** or **U**) can easily be solved by analog techniques, linearly propagating each of the results in a similar fashion for a digital solution for **L** and **U**, retaining the typical 1000× improvement for an analog system over a digital system (e.g. [17]), which turns out to be a similar complexity for a Vector-Matrix Multiplication (VMM). These operations (analog or digital) are O(m^2) in area and total operations for mxm matrices when only considering computational elements, and for full architecture analysis [18] with communication / memory access, requiring (at least) O(m^3) Area–Delay and Power–Delay products.

The remaining question is comparing the analog computation to the linear solution using Gaussian decomposition of **L** and **U** or finding $\mathbf{A}^{-1}$. These digital computations require O(N^3) operations. The analog techniques closely resemble iterative matrix solutions [15]. The relative scaling question for analog iterative techniques is eigenvalue spread with increasing number of nodes, which in turn, is related to the propagation of the resulting iterative outputs. The particular method highly depends on the application, as the computational complexities are similar in each case.

These computational comparisons make some particular assumptions. If application comes from a directly solvable physical system (e.g. PDE computations), one would utilize those more natural techniques for that application as they already benefit the analog techniques. Both analog and digital have similar tradeoffs for sparse computation, particularly with configurable analog capabilities [19]; therefore sparse computation has similar effects on both approaches.

V. Analog Numerics of Iterative Linear Equation Solutions

For digital numerical analysis of linear systems, the condition number of **A** determines the discussion of numerical accuracy. The condition number of **A** is related to the ratio between the largest magnitude (λ_{max}) eigenvalue and smallest (magnitude) (λ_{min}) eigenvalue of **A**, demonstrating the eigenvalues spread. The larger the condition number, the higher required starting precision is expected to counteract the amplification of numerical errors in **b**. The metric is a loose bounds on the error propagation for numerical approaches.

One might imagine that **A** with a moderate or high condition number would be unusable for analog techniques. Typical practice assumes that one loses the number of bits related to the $\log_2$ of the condition number of **A**; such loss of precision makes analog techniques nearly infeasible. This assumption needs to be revisited both for the actual nature of the computation, as well as within the framework of analog numerical analysis techniques [1].

The maximum gain from **b** to **x** is $\lambda_{max}/\lambda_{min}$. Higher gain is one reason for considering a decrease in precision for the operation, particularly when using floating-point arithmetic. Gain of errors in the input (b) or the eigenvector projected input (E^{-1} b) would have a similar issue for the worst case, and therefore either formulation will help. Gain, like all analog gains, will raise up signal and noise.

VI. Summary and Discussion

The paper discussed solving systems of linear equations using analog computation transforming the linear system solution method to a set of ODEs. The technique is related to iterative digital methods for solving linear equations. These approaches extend the energy efficient properties of analog computing initially shown for vector-matrix multiplication to solutions of linear systems, where the vector-matrix multiplication happens through arrays of TAs.

The paper also starts analyzing the algorithmic issues and analog numerical analysis issues of this approach. The dynamics and convergences are studied for different matrices, through experimental measurements of the matrix output solutions from hardware. Analog solutions of linear systems typically is the most challenging algorithm for analog computation [1]. Hence, finding analog algorithmic solutions for linear systems opens the entire range of analog computing towards high-performance computing. This approach allows for solution of any positive definite **A** matrix through the use of OTA devices, and not limited as in resistive coupling networks, originally proposed in Hopfield networks [3], [4] that could be built using two-terminal nano devices like memristors [12], [20], [21], with simple transformations on the set of linear equations (e.g. multiplying by $\mathbf{A}^T$).

VII. Acknowledgements

The authors wish to thank Sandia Laboratories, NM, for partial funding that enabled the circuit characterization.

REFERENCES

[1] J. Hasler, "Starting Framework for Analog Numerical Analysis for Energy Efficient Computing," *Journal of Low Power Electronics Applications*, vol. 7, no. 17, June 2017. pp. 1-22.

[2] J.J. Dongarra, P.Luszczek, and A. Petitet, "The LINPACK Benchmark: past, present and future," *Concurrency and Computation: Practice and Experience*, Wiley, 2003, pp. 803-820.

[3] J.J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of National Academy of Science*, vol. 79, 1982, pp. 2554.

[4] J.J. Hopfield, "Neurons with graded responses have collective computational properties like those of two- state neurons," *Proceedings of National Academy of Science*, vol. 81, 1984, pp. 3088-3092.

[5] R. Umbehauen and A. Cichocki, *MOS Switched-Capacitor and Continuous-Time Integrated Circuits and Systems*, Springer-Verlag, 1989.

[6] A. Cichocki and R. Umbehauen, "Neural networks for solving systems of linear equations and related problems," IEEE March 1992 *CAS I*, vol. 39, no. 2,

[7] N. Guo, Y. Huang, T. Mai, S. Patil, C. Cao, M Seok, S. Sethumadhavan, and Y. Tsividis, "Energy-Efficient Hybrid Analog/Digital Approximate Computation in Continuous Time," *IEEE Journal of Solid State Circuits*, 2016.

[8] Y. Huang, N. Guo, M. Seok, Y. Tsividis, and S. Sethumadhavan, "An Analog Accelerator for Linear Algebra," *Proceedings of the 43rd International Symposium on Computer Architecture*, Seoul, June 18, 2016. pp. 570-582.

[9] R. M. Walker, "An Analogue Computer for the Solution of Linear Simulaneous Equations," *Proceedings of the IRE – Waves and Electrons Section*, 1949, pp. 1467-1473.

[10] S. K. Mitra, "Electrical Analog Computing Machine for Solving Linear Equations and Related Problems," Review of Scientific Instruments vol. 26, 1955. pp.453-457.

[11] K. P. Lanneau and Lindsay I. Griffin, "Analogue Computer for Solving Simultaneous Equations," US Patent 2,911,146 Filed May 27, 2953, issued, Nov. 3, 1959.

[12] Sun, Z., Pedretti, G., Ambrosi, E., Bricalli, A., Wang, W., and Ielmini, D. , "Solving matrix equations in one step with cross-point resistive arrays", *Proceedings of the National Academy of Sciences*, 2019, 116(10), 4123-4128.

[13] S. George, S. Kim, S. Shah, J. Hasler, M. Collins, F. Adil, R, Wunderlich, S. Nease, and S. Ramakrishnan, "A Programmable and Configurable Mixed-Mode FPAA SoC," *IEEE Transactions on VLSI*, vol. 24, no. 6, 2016, pp. 2253-2261.

[14] J. Hasler, "Opportunities in Physical Computing driven by Analog Realization," *IEEE ICRC*, San Diego, 2016.

[15] G. E. Golub and C.F. Van Loan, *Matrix Computation*, 2^{nd} ed, John Hopkins Press, 1989.

[16] B. Ulmann and D. Killat, "Solving systems of linear equations on analog computers," *IEEE Kleinheubach Conference*, 2019.

[17] C. Schlottmann, and P. Hasler, "A highly dense, low power, programmable analog vector-matrix multiplier: the FPAA implementation," *IEEE Journal of Emerging CAS*, vol. 1, 2012, pp. 403-411.

[18] J. Hasler, "Analog Architecture Complexity Theory Empowering Ultra-Low Power Configurable Analog and Mixed Mode SoC Systems," *Journal of Low Power Electronics Applications*, Jan. 2019. pp. 1-37.

[19] J. Hasler, "Large-Scale Field Programmable Analog Arrays," *IEEE Proceedings*, February 2020.

[20] K.-H. Kim, S. Gaba, D. Wheeler, J. M. Cruz-Albrecht, T. Hussain, N. Srinivasa, and W. Lu, "A Functional Hybrid Memristor Crossbar-Array/CMOS System for Data Storage and Neuromorphic Applications", Nano Lett., vol. 12, 2012, pp. 389-395.

[21] S. H. Jo, and W. Lu, "Programmable Resistance Switching in Nanoscale Two-Terminal Devices", *Nano Letters* vol. 9, 2009, pp. 496-500.

[22] S.Y. Kung, *VLSI Array Processors*, Prentice Hall, 1988.