Continuous-Time, Configurable Analog Linear System Solutions With Transconductance Amplifiers

Jennifer Hasler, *Senior Member, IEEE*, and Aishwarya Natarajan

***Abstract*— This paper addresses and experimentally demonstrates a programmable linear equation solver by analog computation. A set of differential equations using transconductance devices directly translated from circuit theory converges to the linear equation solution. These energy-efficient analog techniques are experimentally demonstrated in a configurable analog platform. The resulting analog linear equation solution circuits are effectively analog filters. The paper analyzes the algorithmic issues and analog numerical analysis issues, including accuracy, convergence time, and the interpretation of condition number for analog solutions.**

***Index Terms*— Solving linear equations, transconductance amplifiers, FPAA.**

I. Framing Analog Solutions of Linear Equations

SOLUTION of linear equations is a fundamental digital computation technique (Fig. 1), whereas it is considered difficult to solve using analog computation [1]. Linear equations solve

$$\mathbf{Ax} = \mathbf{b}, \tag{1}$$

where $\mathbf{A}$ is the input matrix, $\mathbf{b}$ is the input vector, and $\mathbf{x}$ is the solution vector. The classic digital solution uses Gaussian elimination. This computation shows all of the strength of digital processing, including the pivots, to get the maximum accuracy for the decomposition [2]. Numerical tools (e.g. MATLAB: MATrix solution LABoratory) are dedicated to these ubiquitous operations. Computing benchmarks are based on solving linear equations (e.g. LINPACK [3]). As digital computing is well matched to solving (1), a problem is considered analytically solved when reduced to solving (1) [1].

If programmable analog techniques competitively solve (1) experimentally, analog techniques span the range of numerical analysis techniques [1], [4], enabled through analog algorithmic techniques [5]. Since a direct analog equivalent to Gaussian elimination involves numerous integer steps and memory manipulations (e.g. pivots) that require the storage of high-resolution intermediate values, this effort considers a different approach to solving (1). This Ordinary Differential Equations (ODE) converges for the solution of (1):

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

where τ is the network time-constant. The iterative method is

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

where ϵ is a function of τ and time-step. Sometimes, Iterative digital techniques in some cases require fewer operations than digital Gaussian elimination, particularly for sparse matrix solutions as well as for embedded computations (e.g. [6]). (2) and (3) converges for positive eigenvalues of $\mathbf{A}$; one can find related iterative equations for other $\mathbf{A}$ [2].

This work focuses on analog solutions to (1) using (2). Reconfigurable and programmable Transconductance Amplifiers (TA) implement (2), as well as provide a platform to analyze the analog numerics. An SoC Field-Programmable Analog Array (FPAA) [7] provides the experimental demonstration platform; these approaches could be implemented in earlier FPAA devices (e.g. [8]) or custom Silicon. Engineering students are taught that static resistive circuits *with independent and dependent voltage- and current-sources* (Fig. 1)

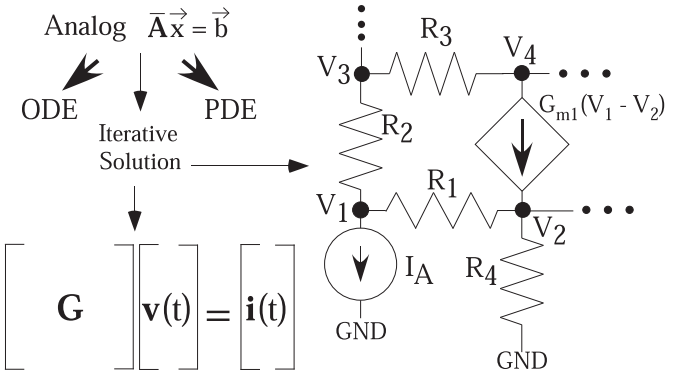

Fig. 1. Digital and analog linear equation solution techniques include both direct (e.g. Gaussian elimination, **L U** decomposition) and iterative techniques. Analog techniques can utilize iterative techniques by using differential equations that converge to the solution. Resistive circuits with linear dependent and independent sources can be directly transformed and then solved by a set of linear equations; this work reverses that perspective to solve linear equations by transforming these equations to a circuit that converges to the linear equation solution. This transformation requires using a set of Transconductance Amplifiers (TA) as voltage-controlled current sources, enabling direct implementation in a programmable and configurable platform (e.g. FPAA), thereby enabling the solution of a wide range of matrices.

Manuscript received May 1, 2020; revised September 7, 2020 and November 12, 2020; accepted November 16, 2020. Date of publication December 7, 2020; date of current version January 12, 2021. The implementation, simulation, and measurement work was supported in part by Sandia National Laboratories, Albuquerque, NM, USA. This article was recommended by Associate Editor E. Tlelo-Cuautle. *(Corresponding author: Jennifer Hasler.)*

The authors are with the School of Electrical and Computer Engineering (ECE), Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: jennifer.hasler@ece.gatech.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCSI.2020.3039763.

Digital Object Identifier 10.1109/TCSI.2020.3039763

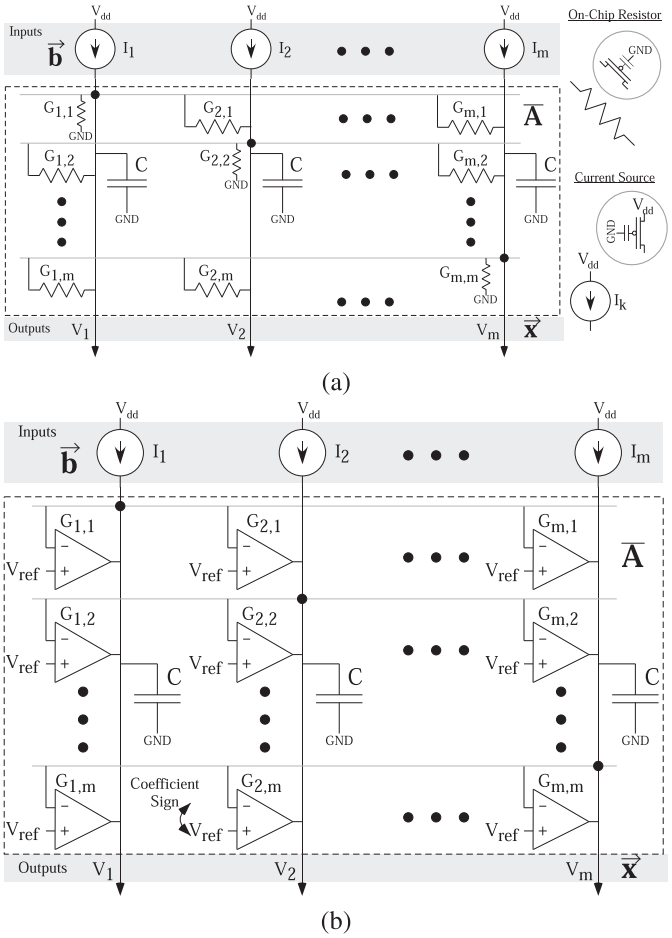

Fig. 2. Potential continuous-time circuit architectures to solve systems of linear equations. Shaded areas show the input **b** and output **x** regions. Connection dots show connections at connecting wire intersections. A *T* would be a connection. (a) Linear equation solution built from resistors and current sources. The programmable resistors and current sources can be implemented with programmable transistors. (b) Linear equation solution built from Transconductance Amplifiers (TA). The constraint matrix of voltage-controlled current sources is set by individual conductances and the corresponding bias currents.

are directly formulated and solved as a system of matrix equations like (1). This work experimentally demonstrates and characterizes the other side of this statement that configurable TA configurations solve (1) through (2) for any stable **A** (Fig. 1). A resistor-only network is limited in the possible **A** matrix, while resistors with op-amps may have stability, mismatch, or parasitic concerns for physical implementations [9]–[13]. These approaches encompass early theoretical discussions of analog computing solving (1) [14], [15], as well as theoretical discussions using multiple-output TA-based recurrent neural networks for solving (1) [16], [17]. Electronic circuits are used to illustrate solutions (e.g. Hopfield networks [18], [19]), simulating linear solutions for a reduced class of problems [20], [20], [21], and considering specialized cases (e.g. elliptic Partial Differential Equations (PDE) [23], [24].)

If an analog linear equation solver is built, one would want this computation to have at least a fraction of the typical 1000× improvement in computational energy efficiency (e.g. [25]) compared to digital techniques (e.g. [26]), as well as typical area efficiencies. Vector-Matrix Multiplication (VMM) shows these efficiencies compared with digital computation [27], [28], and has a related digital algorithmic complexity to solution of (1). This effort will discuss the potential energy efficiencies as well as algorithmic complexities.

This work moves to unify analog solutions of linear systems into a low-energy, compilable approach, including discussing its wider numerical analysis and resulting circuit techniques. The discussion moves towards transconductance-based (Fig. 2), as well as resistive-based, iterative methods for solving linear equations (Sec. II), describing the relevant FPAA implementation details (Sec. III), as well as demonstrating analog solutions of representative linear systems (Sec. IV). The discussion moves towards analyzing the algorithmic issues and analog numerical analysis issues, including accuracy and convergence time considerations. Finally, the resulting analog linear equation solution circuits are effectively analog filters or control systems, and we discuss the connection to these approaches (Sec. V). By directly considering the eigenvector-eigenvalue analysis of (2) (Sec. VI), we reinterpret the impact of condition number, the ratio of largest to smallest eigenvalue, as a question of signal gain as well as algorithm convergence.

II. Physical ODE Solutions of Linear Equations

Analog linear equation solutions could be implemented utilizing resistive coupling between nodes (Fig. 2a), and/or transconductance amplifier coupling between nodes (Fig. 2b). FPAAs efficiently implement resistive and transconductance networks, including utilizing routing fabric elements [7], [29].

Linear systems built of resistors create a diagonally dominant system with negative non-diagonal coefficients. Undergraduate engineering students would recognize that low-frequency circuits of resistors and supplies are modeled by linear system of equations. A generic resistor network for solving a system of linear equations (Fig. 2a) is modeled as

$$I_l = -\sum_{k \neq l}^{m} G_{l,k} V_k + V_l \sum_{k=1}^{m} G_{l,k} + C \frac{dV_l}{dt}, \tag{4}$$

where $G_{k,l}$ is the inter-node conductance, $G_{k,k}$ is the node conductance, and k, l represent the matrix indices of **A** that are of size m. This stable ODE converges to the solution.

Physical computation uses continuous variables as representations. These variables are often scaled, including the units, to abstract the computation from physical values. For example, users solving linear systems often prefer representing their problem as a normalized variable, x, that varies from 0 to 1, rather than keeping track of an arbitrary range of currents (e.g. 14.6nA and 31.7nA). Knowing the abstraction, tools should remove these low level details from the user. Normalizing these variables converts the physical equations to mathematical equations. The smallest of the diagonal sums, $\sum_{k=1}^{m} G_{l,k}$, typically normalizes the resulting equations. The time-constant τ is set by capacitance, C, over a value proportional to the

conductances. The relationships connect (4) to (2):

$$a_{l,k} \propto -G_{l,k} \quad k \neq l$$
$$a_{k,k} \propto \sum_{k=1}^{m} G_{l,k}, \quad b_k \propto I_k. \tag{5}$$

Input currents are signed, and the output voltages are signed values around some bias point. All diagonal values will be greater than 1; all off-diagonal values will be negative.

Transconductance amplifiers (TA) achieve a larger **A** range compared to resistive elements by using voltage-controlled current sources and not just resistor elements. A TA operating in its linearized region is expressed as

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

where G is the circuit's transconductance parameter. When the 9-transistor TA (as in [30]) in the SoC FPAA [7] is programmed in the typical case with a subthreshold bias current (I_{bias}), the resulting transconductance, and coupling between nodes (k,l), would be $G_{k,l} = \frac{\kappa I_{bias}}{U_T}$. where κ is the capacitive division between gate and surface potential for a MOSFET (e.g. [30]), and U_T is the thermal voltage, kT/q ($\approx$25mV at T = 300K). Similarly, an expression for G can be derived for different bias current regions (e.g. above-threshold bias currents) with smaller increases in G for further increases in I_{bias}. An on-chip amplifier, such as the TA in on-chip custom [30] and configurable [7], [31] designs, becomes a voltage or transconductance amplifier depending on the resulting circuit operation as they are often unbuffered designs. The SoC FPAA has 196 non-FG input TAs and 196 FG input TAs [7]. As the SoC FPAA TA bias current sources are set with an internal FG element, these TAs do not have a fourth terminal (e.g. [30]) to set the bias current.

A generic TA network for (2) (Fig. 2b) for the l-th row with subthreshold I_{bias} is modeled as

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

The normalizations connect (7) to (2):

$$a_{l,k} \propto G_{l,k}, \quad b_k \propto I_k \tag{8}$$

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. Positive current sources would go to V_{dd}, and negative current sources would go to GND. ODE solution for these networks requires positive eigenvalues for **A**; related techniques can transform the resulting matrix for the solution of general **A** (e.g. [21]). Often, quantities are built around a single bias current, I_{bias}, which abstracts the resulting currents (typically one of the programmed current values), resulting in $\tau = \frac{CU_T}{\kappa I_{ref}}$. The scaling of current is a question of speed of the computation. Each output row could be scaled accordingly, setting each row's resulting τ and normalizing the row values as required. Each row in (2) could have a different τ.

Physical linear equation solutions, transforming a linear system into an ODE problem, should involve problems that start as a series of linear equations. It is inefficient to take an ODE or PDE, transform it into a linear system, and then to transform it to a linear ODE circuit for the solution. As ODE / PDE applications are efficiently solved by direct implementation [1], [29], we focus on the physical solution of linear systems that originates from different applications.

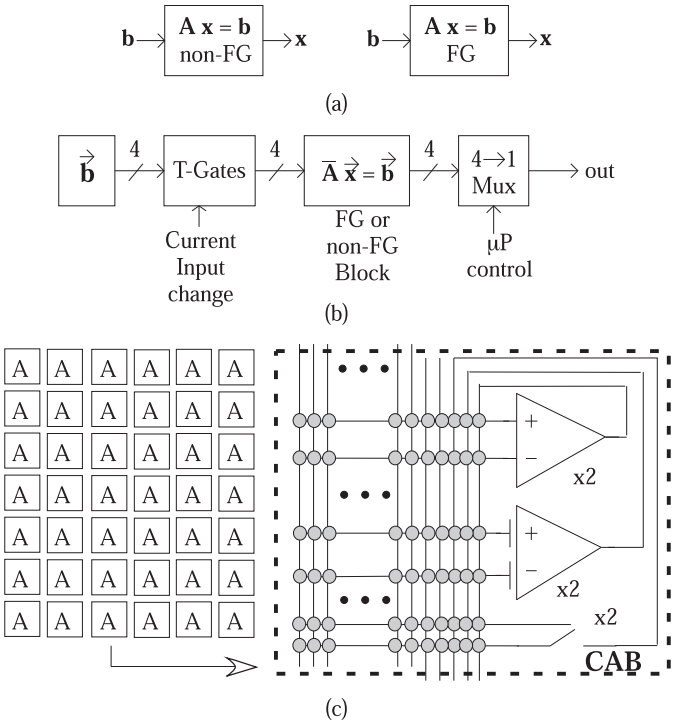

Fig. 3. Analog linear equation solution in the SoC FPAA [7]. (a) High-level tool (Scilab) blocks for the Floating-Gate (FG) and non-FG TA based linear equation solution. (b) A representative block diagram setup for computing and measuring a linear equation solution, where each line represents a parameterized bus of inputs or outputs. This diagram shows a 4×4 **A** as used in later experimental examples. (c) Arrays of FG and non-FG TAs in SoC FPAA Computational Analog Blocks (CAB) perform the linear equation solutions.

III. Configurable Analog Linear Equation Solver

The TA-based analog linear equation solution is experimentally demonstrated in an SoC FPAA [7]. The SoC FPAA is enabled by a significant infrastructure and tool base; an extensive overview of FPAA devices is written elsewhere [31]. The SoC FPAA operates with analog supplies at 2.5V and ground. This computation is encapsulated in a linear equation block (Fig. 3a), abstracting the analog TA computation [32] that can be targeted to an FPAA IC, as well as a macromodel simulated in a full system level simulation (level=1), in an open-source toolset in Scilab [33]. The block parametrizes the number of inputs (**b**) and outputs (**x**) and the resulting matrix (**A**).[1] This discussion presents experimental circuit measurements, and simulation computations where mentioned.

This linear block becomes part of the experimental on-chip test setup (Fig. 3b) that includes providing the **b** input as well as multiplexing each value of the output vector **x**.

[1]This block is implemented in the FPAA tool set, as well as a single example shown in this paper, is implemented as an example for these tools.

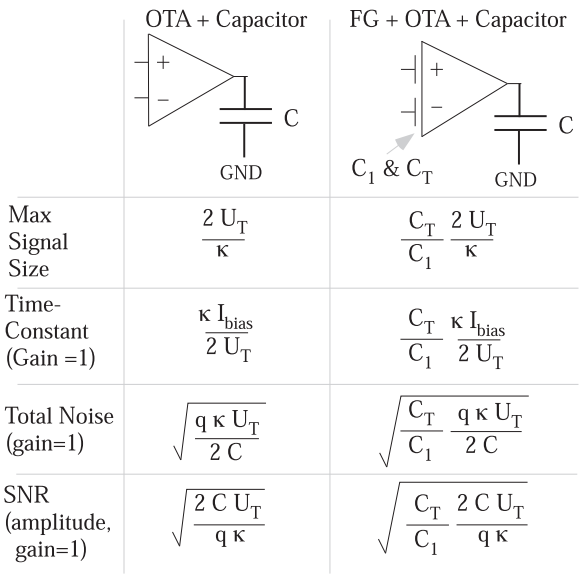

Fig. 4. Comparison of non-FG TA and FG TA for solving $\mathbf{Ax} = \mathbf{b}$. The FG TA results in higher voltage signals and SNR, as well as larger τ by the ratio of total capacitance (C_T) to the input coupling capacitance (C_1).

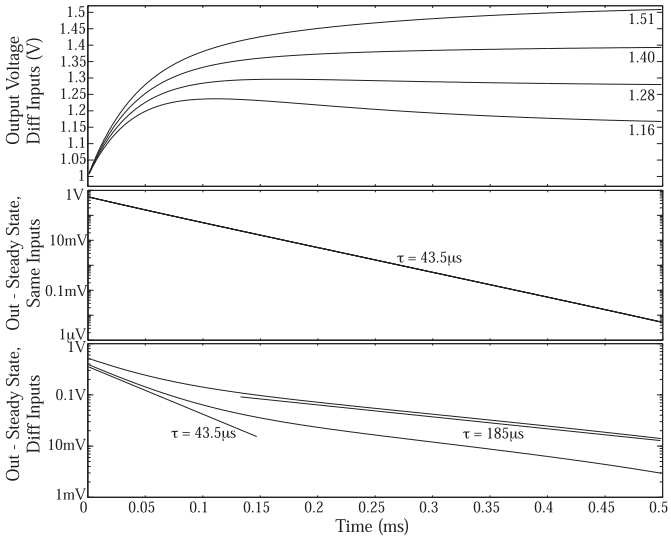

Fig. 5. Measured system level (level=1) simulation results for a 4×4 FG TA for a matrix programmed with 100nA on the diagonals, 50nA for the off diagonal elements, and an input (**b**) that switches between no current and its particular current level. The graph shows two cases of the input vector (**b**), one case (*same inputs*) for $\mathbf{b} = [$ 300nA 300nA 300nA 300nA], and a second case (*different inputs*) for $\mathbf{b} = [$50nA 75nA 100nA 125nA]. These results are compared for an equivalent, more detailed level=2 [39] simulation model, and the results compare closely with experimental measurements.

TA in Computational Analog Blocks (CAB) is a 9 transistor circuit topology (Fig. 3c) [7], [30]. The **A** matrix is set by the conductances of the individual TAs, that in turn are set by Floating-Gate (FG) bias currents set by the pFET bias current. The TA has nearly rail-to-rail output range, although the upper range is limited by the pFET current source remaining saturated. Typical operation occurs for signals around a 1V to 1.25V reference (V_{ref}). A TA is used to convert the input voltage (**b**) signal to a current. The reference voltage(s) can be controlled by Digital-to-Analog Converters (DAC) compiled on the FPAA [7]. 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 measured outputs, **x**(t), are scanned and buffered out.

The choice of FG or non-FG TA depends on the required linearity and convergence (Fig. 4), where a FG TA has higher linearity with a slower time-constant for a given bias current, while a non-FG TA has a lower linear range and provides a faster convergence for a given bias current. Other papers have discussed a range of programmable current bias TA [34], the effect of FG capacitance coupling on core circuit parameters [35], built-in self-test algorithms [36] and application of these structures in nonlinear dynamics [37] and education [38]. We summarize these core results to facilitate our linear-equation solutions (Fig. 4). The non-FG TA is more energy efficient for a given bias current. The FG TA is easier to instrument using 1V signals rather than 50-100mV signals. The key parameter that scales these results is the ratio of total capacitance (C_T) to the input coupling capacitance (C_1). The experimental measurements will illustrate the properties of these two TA approaches (Fig. 4) corresponding to these two blocks (Fig. 3a). Using FPAA routing fabric results in efficient resistive networks [7], [29], although they typically result in lower SNR and signal amplitudes than TAs.

The system model (level=1) for this TA based equation solver where voltages are referenced to V_{ref}, and where the inputs utilize a similar FG-based TA would be

$$\frac{dV_l}{dt} = \frac{I_{b,l}}{C} \tanh\left(V_{x,l}/V_L\right) - \frac{1}{C}\sum_{k=1}^{m} I_{A,l,k} \tanh\left(V_k/V_L\right) \tag{9}$$

where $I_{b,l}$ are the bias currents for the input FG TAs (**b**), $V_{x,l}$ are the input voltages (**b**), $I_{A,l,k}$ are the bias currents for the matrix TAs (**A**), V_L is the linear range of the TA, and we assume a nominal value of C (e.g. 1pF) until compilation provides better data that gives a better estimate, including the circuit place and route. This model is implemented for the FG TA block, and can be directly modified for the non-FG block. One can show results from this abstracted simulation model (from (9)), and compare it to a more detailed transistor level simulation [39] and experimental measurements (Fig. 5).

IV. Linear Equation Experimental Dynamics

Experimental measurements from a 4×4 **A** matrix illustrate the linear equation solver (Fig. 3b) dynamics. The compiled 4×4 linear solver requires 16 TAs to implement the 4×4 **A** matrix, and 4 TAs to implement the **b** matrix. The **b** vectors are inputs fed to the gate input of the TA structure, step functions from a *zero* point (= fixed potential or current switched off) to a desired input for **b**. The matrix solution outputs, *x(t)* , show the dynamics when new inputs are applied.

An identity matrix for **A** illustrates the solution circuit dynamics (Fig. 6). The circuit model (Fig. 6, non-FG TAs) simplifies to a group of two-TA components for a diagonal **A** (Fig. 6). The time-constant is directly related to the diagonal

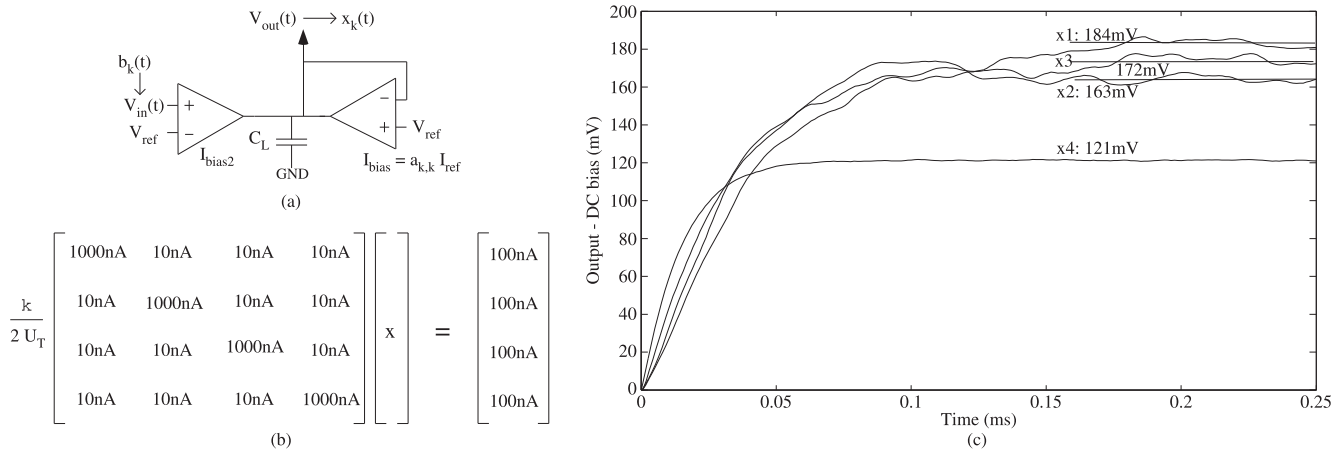

Fig. 6. Setting up $\mathbf{Ax} = \mathbf{b}$ using TA elements. (a) Effective circuit for solving $\mathbf{Ax} = \mathbf{b}$ for a diagonal matrix $\mathbf{A}$, that corresponds to a first-order TA circuit with tunable gain based on the ratio of the bias currents. (b) Effective matrix programmed for this diagonal computation (1μA as the *1* elements and 10nA as the *0* elements), where the off diagonal elements programmed to 1% or less of the diagonal elements (1μA compared to 10nA). Effectively, the off diagonal elements can be ignored in these measurements. (c) Analysis for the convergence of $\mathbf{A}$ matrix after applying a 40mV $\mathbf{b}$ input through the TA current sources. The time-constant was extracted to be roughly 47μs for each component. It is nearly identical for all the four curves, thereby getting similar convergences.

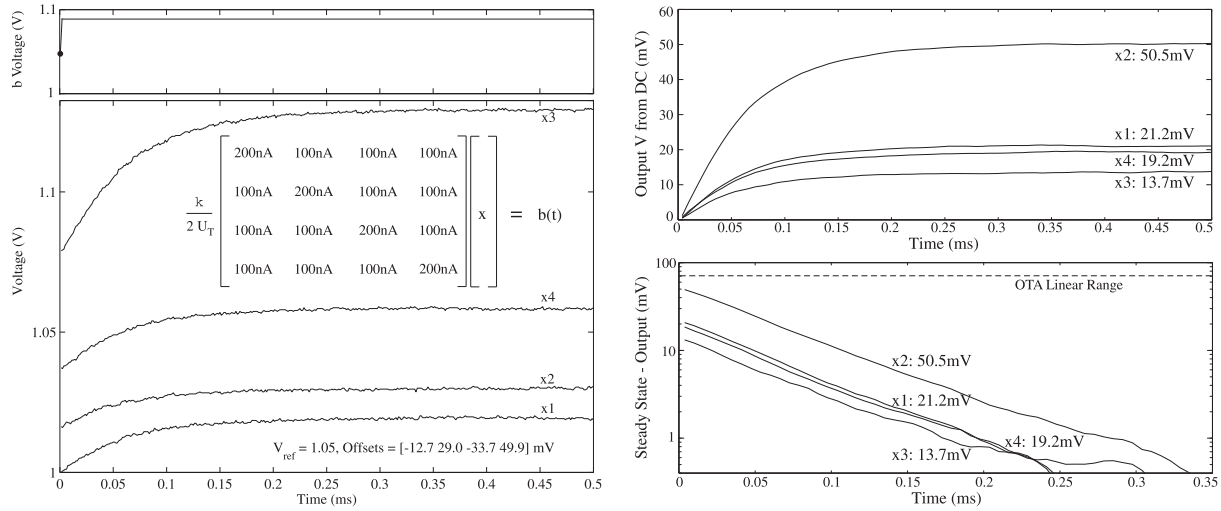

Fig. 7. Measured convergence $\mathbf{x}$ for $\mathbf{A}$ matrix with 200nA and 100nA as the diagonal and off-diagonal elements, respectively. For a 40mV $\mathbf{b}$ input applied as current sources, the x(t) solution is plotted and is within the linear range of the TAs. The time-constant (61μs) from curve-fitting the $\log(\cdot)$ of the step responses; as expected, there are three identical eigenvalues for this matrix. One of the components along the larger eigenvalue (by a factor of 5) converges 5 times faster in 12μs.

TA bias current and the FPAA routing capacitive load, showing un-programmed mismatches. In general, we can normalize each row, effectively changing the time-constant, but not affecting the final steady-state solution. The voltage offsets could be tuned out by using FG-TA elements. If the $\mathbf{A}$ elements are operating in their linear region, one can define *zero* at any particular offset because if the outputs ($\mathbf{x}$) are measured around an offset vector ($\mathbf{x_0}$) due to an offset ($\mathbf{b_0}$) in the input ($\mathbf{b}$), then one can simply normalize the output around the starting zero point because we are solving a linear system, $\mathbf{A}(\mathbf{x} - \mathbf{x_0}) = \mathbf{b} - \mathbf{b_0}$.

Analyzing (2) illustrates the circuit dynamics that can be experimentally verified. $\mathbf{A}$ can be written as

$$\mathbf{A} = \mathbf{E}\Lambda\mathbf{E}^{-1} \tag{10}$$

where Λ is a diagonal matrix of eigenvalues, and $\mathbf{E}$ are the corresponding rows of normalized (power = 1) eigenvectors corresponding to the particular eigenvector. This relationship simplifies to $\mathbf{A} = \mathbf{E}\Lambda\mathbf{E}^{T}$ for symmetric $\mathbf{A}$. Transforming the solution $\mathbf{x}$ into a projection along the eigenvector basis, $\mathbf{x} = \mathbf{Ey}$ for (1), we project along the eigenvector basis to get

$$\tau\frac{d\mathbf{y}}{dt} + \Lambda\mathbf{y} = \mathbf{E}^{-1}\mathbf{b} = \hat{\mathbf{b}},$$
$$\frac{\tau}{\lambda_k}\frac{dy_k}{dt} + y_k = \frac{\hat{b}_k}{\lambda_k}, \tag{11}$$

where y_k is the kth component of $\mathbf{y}$, and λ_k is the kth eigenvalue of $\mathbf{A}$. The matrix requires positive λ_k values, although through transformations in $\mathbf{A}$ and $\mathbf{b}$, one could achieve positive values ($\mathbf{A}$ is positive definite). Depending on the projection of $\mathbf{b}$ on the eigenvector basis, the solution could effect one or all

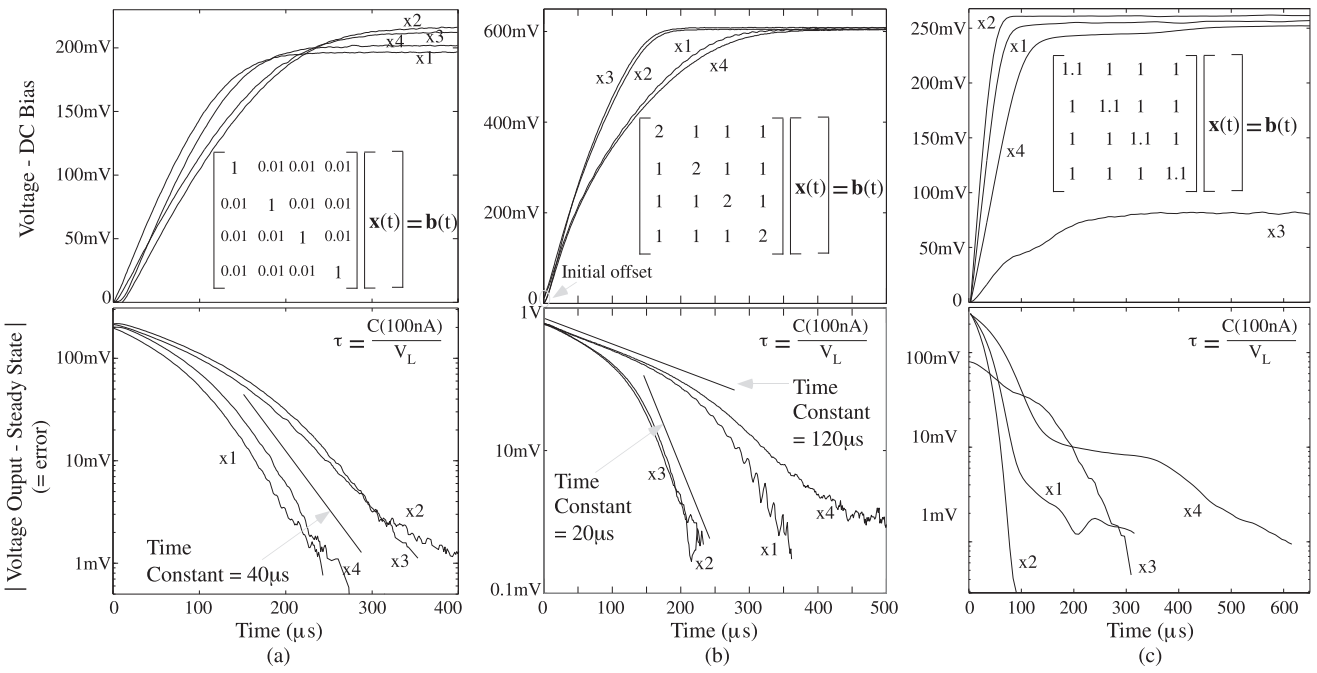

Fig. 8. Measured solutions for a FG TA based linear equation solution circuit. Each case shows the measured results and the log trajectory (or error) towards the steady state. (a) Solution of diagonal matrix with 100nA diagonal elements. (b) Solution of 200nA diagonal elements and 100nA off-diagonal elements (c) Solution of 110nA programmed diagonal elements and programmed 100nA elements. Some mismatch resulting from indirect programming was not compensated, so the elements had some random variation.

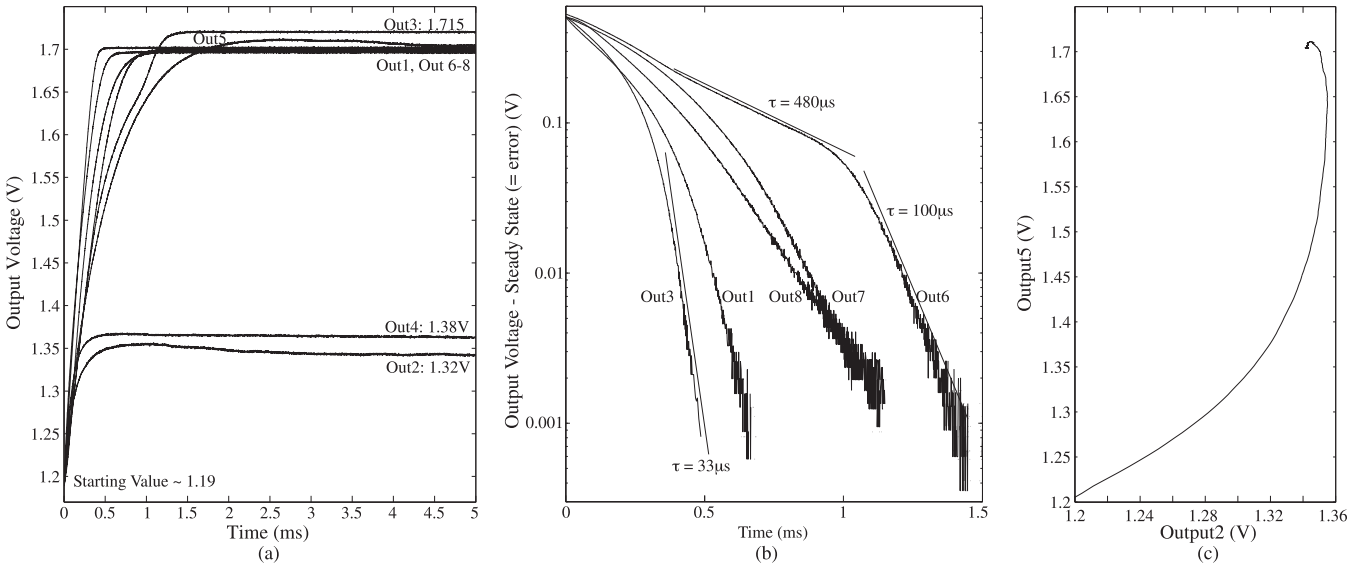

Fig. 9. Dynamics for a compiled 8×8 **A** matrix with the diagonal elements programmed to 200nA and the off-diagonal elements programmed to 100nA. (a) Starting from an initial value, each of the eight nodes converges to their final value. (b) Several of the outputs directly converge to their steady state solution (Out1, 3, 6, 7, 8), although they might have different time-constants depending on the convergence of other nodes. (c) Some of the outputs may overshoot or have a damped oscillation into their steady-state solution (e.g. Out2 vs. Out5).

of the eigenvectors. The time-constants are scaled by λ_k, so the largest eigenvalue component will converge first, the smallest eigenvalue component converges last and often is the component most noticed in the system dynamics. The value of τ could be modified along each row to compensate for the slower response of the smaller λ_k values, where these changes are projected onto the eigenvector basis.

Programming different **A** matrices can illustrate these dynamics (Fig. 7). The dynamics are studied by plotting the matrix solution outputs, **x**(t) which are the steady-state responses. 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 FG TA implementation enables further investigation of the linear solution dynamics given the larger signal amplitudes and SNR. The larger signal amplitudes from the FG TA devices enable observation of the multiple time-constants (Fig. 8), particularly when we choose matrices with less symmetry and identical eigenvalues (Fig. 8c). Scaling the currents of the **A** matrix also scales the convergence time (Fig. 8a versus Fig. 8b). The larger eigenvalue spread results in a larger spread in the resulting time-constants (Fig. 8d). This case is an example in our FPAA graphical tools.[2]

The core block can compile solutions to larger **A** matrices, including 8×8 (e.g. Fig. 9), where the largest square matrix on an SoC FPAA would be 14 x 14 resulting from the 196 FG TAs and/or 196 non-FG TAs. A trajectory may converge to the solution through an oscillatory path (e.g. imaginary roots, elliptic paths), depending on the rows of **A**, where a particular variable might overshoot or spiral into the solution (e.g. Fig. 9). One could imagine building TA with other elements, including routing elements, to increase the matrix size. Fewer elements are needed for a sparse matrix, where one needs one particular element per value in the **A** matrix.

V. Analog Linear Solutions as Linear Filtering

A different look at (2) through the TA circuit configurations (Fig. 2b) illustrates that this linear equation solver sets up a linear filter between the inputs **b** and outputs **x**. The 4×4 equation solver in Fig. 8b) shows a low-pass frequency response when applying a sinusoid to one of the **b** inputs (Fig. 10a), as well as a low-pass chirp signal response (Fig. 10b). This second-order low-pass response from the higher-frequency attenuation is consistent with the step-response dynamics (Fig. 8).

The solution of linear equations transforms to filter design as well as approaches used in control system implementations. One might be able to utilize this transformation to optimize for a filter transfer function using a full **A** relaxing other parameter constraints, often expressed as requirements for high resonance and/or minimizing energy requirements. G_m-C filter cascades of first and second-order filter components (Fig. 11) match directly to the linear equation solver with TA components (Fig. 2b). Second-order filter sections, which are cascaded for many filters, are equivalent to a linear equation solver (Fig. 12). These circuits illustrate wave-propagating behavior where the delay would linearly scale with the number of components, consistent with cascades of TA elements including unity gain devices or cochlear models (e.g. [30]).

VI. Analog Linear Equation Solver Numerics

Given the opportunity of an analog linear equation solver (Fig. 2b), and using the equivalency of linear equation solution and filter design, we can start addressing the computational algorithmic questions. The discussion focuses on understanding the architecture aspects of linear equation solutions (Sec. VI-A) as well as discussing the accuracy of linear equation solutions and matrix condition number (Sec. VI-B).

[2]We will post this item in our FPAA toolkit example elements by the time this paper would be first made available.

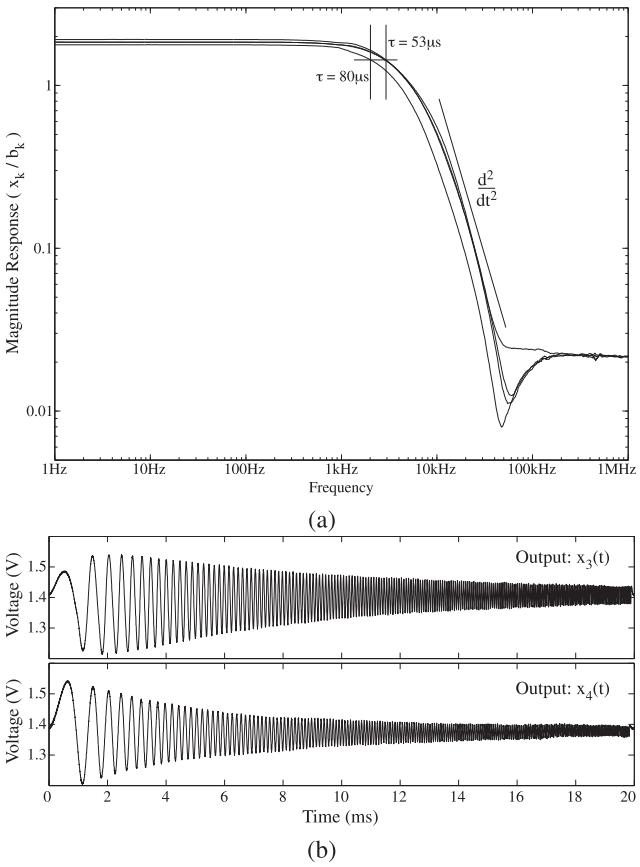

Fig. 10. Linear equation solver characterized as a linear filter for a 4×4 FG TA configuration. The **A** matrix was the same as in Fig. 8b. (a) Frequency response from input (**b**) to the output (**x**) vectors. (b) The output response of a linear equation solver for a chirp input (**b**), shows an expected response of a second-order low-pass filter as in (a). The chirp input sweeps from 1Hz to 20kHz over a time duration of 10ms as a 20mV sinusoidal offset around 1.25V. The output response shows the attenuation of the higher frequencies corresponding to a first-order low-pass filter.

A. Methods of Linear-Equation Solutions

This discussion focuses on the architectural tradeoffs for analog solutions of (2) as well as some digital alternatives. If the application comes from a directly solvable physical system (e.g. PDE computations), one would utilize those more natural techniques for that application. Both analog and digital techniques have similar tradeoffs for sparse computation, particularly with configurable analog capabilities [31].

Gaussian elimination can be represented as decomposing **A** into a Lower diagonal matrix (**L**), and an Upper diagonal matrix (**U**), a technique used when solutions for multiple inputs (**b**) are required. Analog techniques directly solve these two matrices (Fig. 14), linearly propagating each of the results in a similar fashion to a digital solution for **L** and **U**, retaining the typical improvement for an analog system over a digital system (e.g. [28]). This analog operation could be transformed to a Vector-Matrix Multiplication (VMM). All of these computations are $O(m^2)$ in area and total operations for mxm matrices when only considering computational elements, while for full architecture analysis [32] with communication / memory access, the computations require (at least) $O(m^3)$

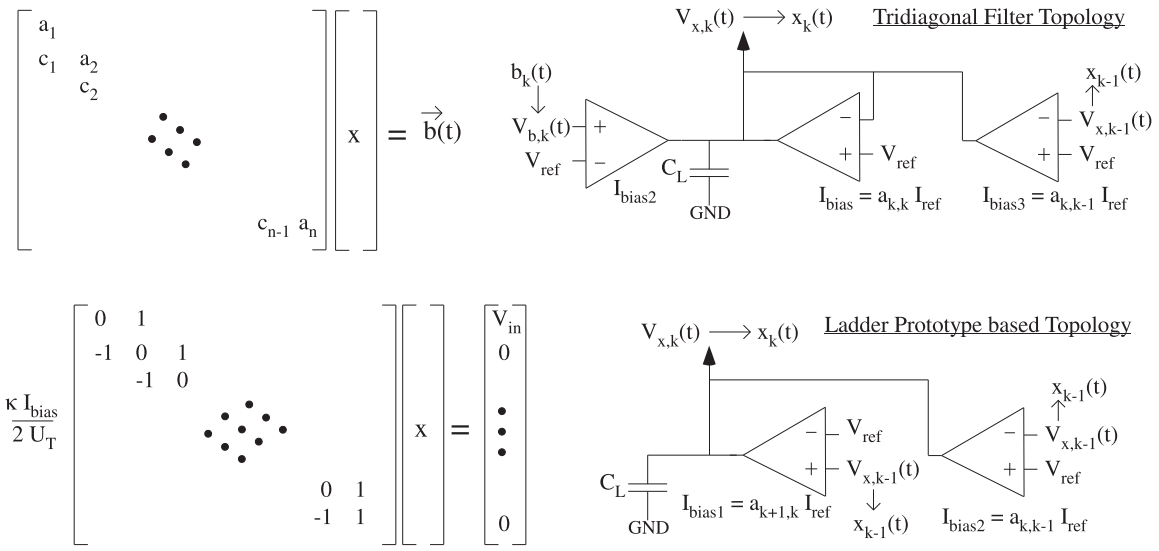

Fig. 11. Visualizing linear equation solution as active linear filtering. (a) The effective circuit for solving $\mathbf{Ax} = \mathbf{b}$ for a tridiagonal matrix $\mathbf{A}$ results in the propagation of a sequence of linear, first-order filters. (b) A ladder filter topology based on second-order LC prototypes results in a different tridiagonal $\mathbf{A}$ matrix, resulting in the propagation of coupled filter elements. The last diagonal element is *1* for typical matched resistive termination.

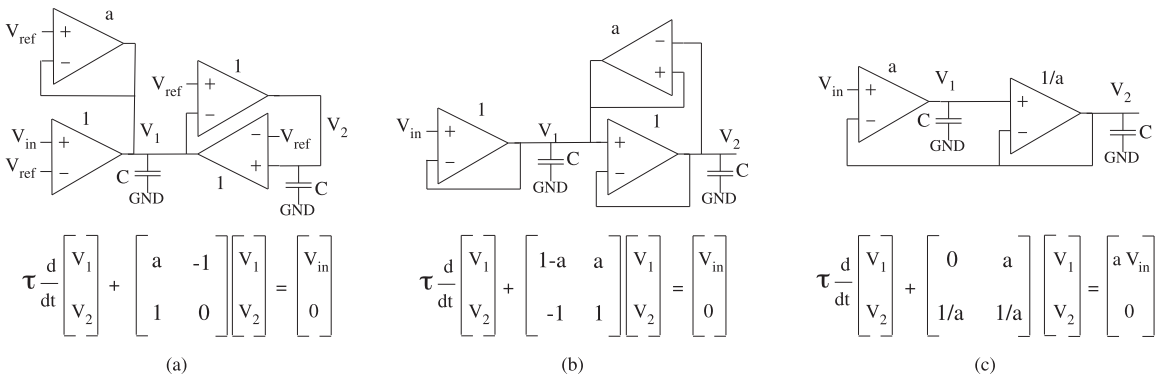

Fig. 12. Second-order filter prototypes result in 2×2 $\mathbf{A}$ matrices that are often cascaded for a typical linear system. The TAs are normalized around a baseline transconductance (G), resulting in a baseline time-constant of C/G. A TA operating in its linear region that uses both + and - terminals can be split into two different TAs, and therefore fits the form in Fig. 2b. (a) Bandpass filter based on a gyrator topology. (b) Lowpass filter utilizing positive feedback (a). (c) Lowpass filter based on Tow-Thomas configurations.

Area–Delay and Power–Delay products. One might digitally decompose a single $\mathbf{A}$ into $\mathbf{L}$ and $\mathbf{U}$ and further download them into the analog solver for continuous analog processing.

Since the solution of several matrices has a similar architectural scaling between digital and analog approaches, comparing the computational efficiencies between digital and analog solvers shows the comparison between the two methods, in a similar way to VMM comparisons [27]. The operating frequency, f, which is inversely proportional to the τ of the system, is proportional to the bias current (sub-threshold operation) and inversely proportional to the load capacitance and linear range. Each TA effectively computes a Multiply-ACcumulate (MAC) operation in addition to ODE integration; we will start by comparing MAC operations at a given frequency. The proportionality depends on the eigenvalues of $\mathbf{A}$, typical of other iterative matrix solutions. The required computational power (P) is proportional to the bias current per TA ($2\times$) and V_{dd}. The total (thermal) noise and SNR per node will be dependent on kT/C noise, given the higher linear range (V_L). Table I shows this model assuming an average I_{bias} over the m values on a row or column, and C is the single element total capacitance (C = 200fF).

TABLE I
COMPUTATIONAL EFFICIENCY FOR M = 16, V_L = 1V, AND V_{dd} = 2.5V

	Freq (f)	Power / node	Comp Eff MMAC(/s)/μW	SNR (power)
I_{bias}	$\propto \frac{I_{bias}}{2\pi C V_L}$	$2m^2 V_{dd} I_{bias}$	$\frac{1}{2\pi C V_L V_{dd}}$	$\frac{2V_L Cm}{q}$
1nA	800Hz	1.28μW	0.32	78dB
100nA	80kHz	128μW	0.32	78dB

Analog techniques to solving linear equations do have computational energy efficiency improvements compared with digital techniques, although not quite the $1000\times$ advantages over digital computation in this configurable platform. The TAs in this configurable framework have higher capacitance than other algorithms, such as VMM computations [28]. A scaled down FPAA device, an optimized FPAA device, or

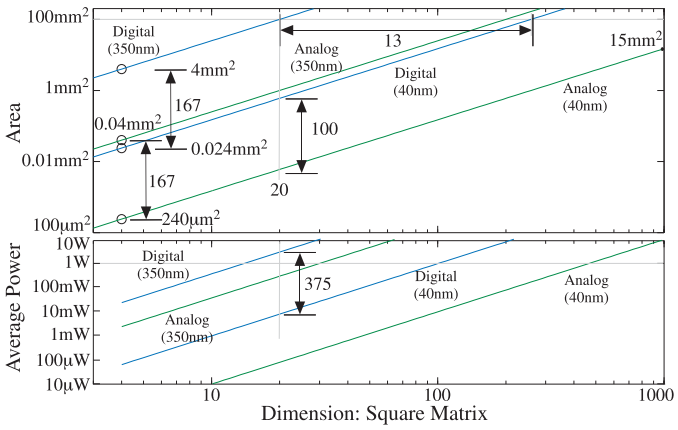

Fig. 13. Linear equation solution comparison ($n \times n$ matrix) between analog versus digital approaches at 350nm CMOS and projected 40nm CMOS both in area and in average power consumption. *Area Comparison:* An FG TA is roughly $1500\mu m^2$ in 350nm CMOS, and a 16×16 multiplier and MAC unit is roughly $0.25mm^2$ in 350nm CMOS (optional block in MSP430 SoC FPAA processor). The same blocks in 40nm CMOS would roughly scale quadratically ($100\times$) in area for both digital (e.g. $2500\mu m^2$) and analog (e.g. $15\mu m^2$) blocks. The area improvement between analog approaches at 350nm and at projected 40nm CMOS is $167\times$ the digital approach in the same process. The energy comparisons *Power / Energy Comparison:* Power comparison assuming a 100kHz output bandwidth for linear equation solutions, assuming the iterative solution requires O(n) convergence time (eigenvalue spacing). At 350nm CMOS, a custom analog implementation would require $16\mu W$ per stage ($\approx 10\times$ lower capacitance than this FPAA case), and a custom digital implementation would require 6mW per stage (1MMAC(/s)/mW). For 40nm CMOS, the analog implementation scales quadratically because of the decreased capacitance, and the digital implementation optimistically moves to a factor of the energy efficiency wall (10MMAC(/s)/mW).

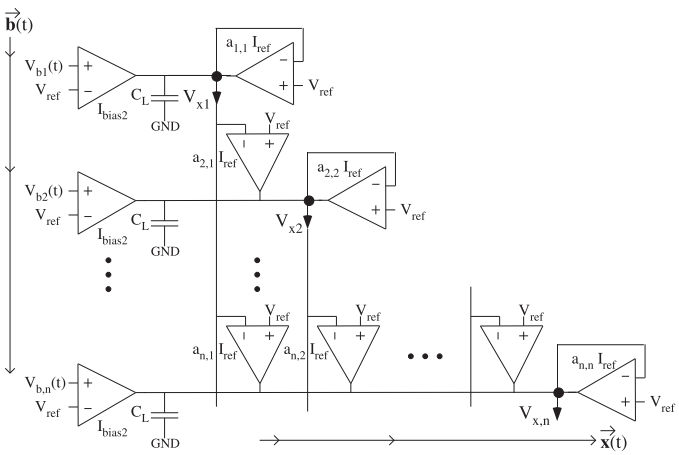

Fig. 14. Linear equation solver / analog filter direct implementation for solving a lower-triangular (**L**) or upper-triangular(**U**) linear equation.

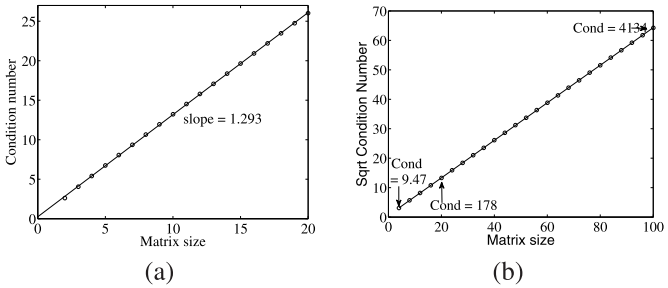

Fig. 15. Condition number relates to the propagation time of the resulting network. (a) Condition number of the Ladder Filter Topology scales linearly with node size, consistent with the propagation time of wave-propagating systems. Tridiagonal systems have similar wave-propagating properties, and similar changes in condition number for normalized **A**. (b) Condition number of the Resistive Diffusion Problem scales quadratically with node size, consistent with the propagation time of diffusive systems.

a custom IC implementation would result in substantially smaller capacitances and similar VMM efficiencies. In a custom solution, a TA could be built using a single FG device. One would have similar VMM crossbar computational efficiencies, preserving the $1000\times$ factor for analog computation compared to the digital efficiency wall of 40MMAC(/s)/mW (16bit registers) [26], to obtain similar numerical results for analog computations [1]. These analyses allow a comparison between a custom analog and a custom digital solution at 350nm CMOS, as we have measured components in this process as well as extrapolate for 40nm CMOS [40], showing roughly a $100\times$ area improvement and $350\times$ energy efficiency improvement (Fig. 13).

B. Eigenvalue Analysis of Matrix Solutions

The numerical accuracy of analog linear equation solutions is a bigger issue for analog solutions than digital solutions, given the lower starting precision of analog computation. 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 metric is a loose bound on the error propagation for digital linear solvers, grouping all errors in the starting values and the numerical computing errors together. The condition number measures the output value sensitivity for a small change or errors in the input argument. The larger the condition number, the higher expected starting precision is required to counteract the amplification of numerical errors in **b**.

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 could make analog techniques nearly infeasible in several cases. The analog steady-state solutions are derived from (11),

$$y_k = \frac{\mathbf{E}^{-1}\mathbf{b}}{\lambda_k} \tag{12}$$

The output, **x**, is the eigenvector projection of **y**, $\mathbf{x} = \mathbf{E}\mathbf{y}$, so the difference in gain of one component would be the ratios of their eigenvalues. Gain of errors in the input (b) or the eigenvector projected input (E^{-1} b) would have a similar issue for the worst case. The maximum gain, the ratio between the maximum and minimum eigenvalues, is the condition number of **A**.

Does condition number of **A** for analog linear equation solvers relate to the loss of accuracy between the input **b** and the output **x**? Condition number does not directly translate to numerical errors in the analog solver, because if the solution exists and is bounded in the dynamic range of the analog

solver, then one has the correct solution within the output measurement precision. The system converges to the exact steady state of the programmed physical system. Linear filters, solvers of linear equations, are built without catastrophic accuracy losses. The numerical issues are primarily questions of having sufficient dynamic range for the particular solution, relating to issues of solving **Ax = b** using fixed-point digital computation. If a component has such a small gain that it is small compared to the output accuracy, then its required accuracy is already small.

1) Large Signal Gain: The gain for particular solutions could saturate the available dynamic range. For the FG TA case, that dynamic range would be nearly the entire power supply range, and yet the range is still finite. One interpretation of condition number gives an upper bounds of the gain between input (**b**) and output (**x**), although the ratio of two large eigenvalues is considerably different than the ratio of two small eigenvalues. Normalization of the initial matrix (**A**) values as well as having bounded input (**b**) values in implementing the analog solver partially addresses these issues; additional normalization (e.g. along a row of **A x = b**) could further improve these systems. Filter cascades do consider balancing the noise and distortion power accumulation per stage although these issues typically grow linearly with node size as well as are handled in multiple situations. One such cascade would be a ladder-filter delay line implementation, a second-order system where one might expect instabilities, and yet, the condition number scales linearly for this system (Fig. 15a). Circuit implementations often have these normalization aspects. For example, for a one-dimensional PDE resistor solution one rarely expects having large solutions for this diffusive system, even though the condition number increases quadratically with node size. Many linear equation formulations are transformations of linear sets of ODEs or PDEs, transforming the two dimensional space into a one-dimensional vector, enabling these solutions for digital computation. Solving this linear set of equations, either in whole or in blocks, by analog circuits seems highly inefficient, although such viewpoints are sometimes considered (e.g. [23], [24]). The physics behind solving linear systems may be useful for other ODE solutions, and these techniques should be used for those ODE applications where applicable.

2) Preconditioning: Multiple preconditioning steps can be applied to improve the convergence and eigenvalue spread for a particular linear system, techniques often used in digital solvers [2]. Several techniques attempt for simple modifications, such as normalizing to optimize **A** diagonal components, and these can be directly utilized, as well as adapting τ values as a part of that preconditioning. Preconditioning along the diagonal and related elements can also quickly translate an **A** matrix with negative eigenvalues to a stable ODE system.

VII. 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. 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 TA devices, and not limited as in resistive coupling networks.

These techniques could be extended towards building a canonical nonlinear function solver by mixing different types of transconductance amplifiers (e.g. built on an FPAA). Different TA circuits result in different even and odd nonlinearities, allowing the direct implementation of second and third-order normal forms within a similar architectural framework. Considering such nonlinearities expands solution spaces to include oscillatory systems. This capability further enables compilation and synthesis of nonlinear ODEs in experimental hardware, as well as a framework for further theoretical development of applications utilizing nonlinear functions.

References

[1] J. Hasler, "Starting framework for analog numerical analysis for energy-efficient computing," *J. Low Power Electron. Appl.*, vol. 7, no. 17, pp. 1–22, 2017.

[2] G. E. Golub and C. F. Van Loan, *Matrix Computation*, 2nd ed. Baltimore, MD, USA: John Hopkins Press, 1989.

[3] J. J. Dongarra, P. Luszczek, and A. Petitet, "The LINPACK benchmark: Past, present and future," *Concurrency Comput., Pract. Exper.*, vol. 15, no. 9, pp. 803–820, 2003.

[4] S. D. Conte and C. de Boor, *Elementary Numerical Analysis: An Algorithmic Approach*. New York, NY, USA: McGraw-Hill, 1980.

[5] J. Hasler, "Analog architecture complexity theory empowering ultra-low power configurable analog and mixed mode SoC systems," *J. Low Power Electron. Appl.*, vol. 9, no. 4, pp. 1–37, 2019.

[6] J. Liu, Y. V. Zakharov, and B. Weaver, "Architecture and FPGA design of dichotomous coordinate descent algorithms," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 56, no. 11, pp. 2425–2438, Nov. 2009.

[7] S. George *et al.*, "A programmable and configurable mixed-mode FPAA SoC," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 24, no. 6, pp. 2253–2261, Jun. 2016.

[8] S. Brink, J. Hasler, and R. Wunderlich, "Adaptive floating-gate circuit enabled large-scale FPAA," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 11, pp. 2307–2315, Nov. 2014.

[9] R. M. Walker, "An analogue computer for the solution of linear simultaneous equations," *Proc. IRE*, vol. 37, no. 12, pp. 1467–1473, Dec. 1949.

[10] S. K. Mitra, "Electrical analog computing machine for solving linear equations and related problems," *Rev. Sci. Instrum.*, vol. 26, no. 5, pp. 453–457, May 1955.

[11] K. P. Lanneau and I. L. Griffin, "Analogue computer for solving simultaneous equations," U.S. Patent 2 911 146, Nov. 3, 1959.

[12] Z. Sun, G. Pedretti, E. Ambrosi, A. Bricalli, W. Wang, and D. Ielmini, "Solving matrix equations in one step with cross-point resistive arrays," *Proc. Nat. Acad. Sci. USA*, vol. 116, no. 10, pp. 4123–4128, Mar. 2019.

[13] Z. Sun, G. Pedretti, and D. Ielmini, "Fast solution of linear systems with analog resistive switching memory (RRAM)," in *Proc. IEEE Int. Conf. Rebooting Comput. (ICRC)*, Nov. 2019, pp. 1–5.

[14] Y. Xia, J. Wang, and D. L. Hung, "Recurrent neural networks for solving linear inequalities and equations," *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, vol. 46, no. 4, pp. 452–462, Apr. 1999.

[15] B. Ulmann and D. Killat, "Solving systems of linear equations on analog computers," in *Proc. Kleinheubach Conf.*, Miltenberg, Germany, Sep. 2019, pp. 1–4.
[16] M. S. Ansari and S. A. Rahman, "A non-linear neural circuit for solving system of simultaneous linear equations," in *Proc. Int. Multimedia, Signal Process. Commun. Technol.*, Mar. 2009, pp. 120–123.
[17] M. S. Ansari and S. A. Rahman, "MO-OTA based recurrent neural network for solving simultaneous linear equations," in *Proc. Int. Conf. Multimedia, Signal Process. Commun. Technol.*, Dec. 2011, pp. 192–195.
[18] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proc. Nat. Acad. Sci. USA*, vol. 79, no. 8, pp. 2554–2558, 1982.
[19] J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons," *Proc. Nat. Acad. Sci. USA*, vol. 81, no. 10, pp. 3088–3092, May 1984.
[20] R. Umbehauen and A. Cichocki, *MOS Switched-Capacitor and Continuous-Time Integrated Circuits and Systems: Analysis and Design*. New York, NY, USA: Springer-Verlag, 1989.
[21] A. Cichocki and R. Unbehauen, "Neural networks for solving systems of linear equations and related problems," *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, vol. 39, no. 2, pp. 124–138, Mar. 1992.
[22] A. Cichocki and R. Unbehauen, "Neural networks for solving systems of linear equations. II. Minimax and least absolute value problems," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 39, no. 9, pp. 619–633, Sep. 1992.
[23] N. Guo *et al.*, "Energy-efficient hybrid analog/digital approximate computation in continuous time," *IEEE J. Solid-State Circuits*, vol. 51, no. 7, pp. 1514–1524, Jul. 2016.
[24] Y. Huang, N. Guo, M. Seok, Y. Tsividis, and S. Sethumadhavan, "An analog accelerator for linear algebra," in *Proc. 43rd Int. Symp. Comput. Archit.*, Seoul, South Korea, Jun. 2016, pp. 570–582.
[25] C. Mead, "Neuromorphic electronic systems," *Proc. IEEE*, vol. 78, no. 10, pp. 1629–1636, Oct. 1990.
[26] H. B. Marr, B. Degnan, P. Hasler, and D. Anderson, "Minimization of energy per Op in an asynchronous pipeline above and below threshold," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 21, no. 1, pp. 147–151, 2013.
[27] R. Chawla, A. Bandyopadhyay, V. Srinivasan, and P. Hasler, "A 531 nW/MHz, 128×32 current-mode programmable analog vector-matrix multiplier with over two decades of linearity," in *Proc. IEEE Custom Integr. Circuits Conf.*, Oct. 2004, pp. 651–654.
[28] C. R. Schlottmann and P. E. Hasler, "A highly dense, low power, programmable analog vector-matrix multiplier: The FPAA implementation," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 1, no. 3, pp. 403–411, Sep. 2011.
[29] S. Koziol, R. Wunderlich, J. Hasler, and M. Stilman, "Single-objective path planning for autonomous robots using reconfigurable analog VLSI," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 47, no. 7, pp. 1301–1314, Jul. 2017.
[30] C. Mead, *Analog VLSI and Neural Systems*. Reading, MA, USA: Addison-Wesley, 1989.
[31] J. Hasler, "Large-scale field programmable analog arrays," *Proc. IEEE*, vol. 108, no. 8, pp. 1283–1302, Aug. 2020.
[32] J. Hasler, A. Natarajan, and S. Kim, "Enabling energy-efficient physical computing through analog abstraction and IP reuse," *J. Low Power Electron. Appl.*, vol. 8, no. 4, pp. 1–23, Dec. 2018.
[33] M. Collins, J. Hasler, and S. George, "An open-source toolset enabling analog–digital–software codesign," *J. Low Power Electron. Appl.*, vol. 6, no. 1, Feb. 2016, pp. 1–15.
[34] R. Chawla, F. Adil, G. Serrano, and P. E. Hasler, "Programmable G_m–C filters using floating-gate operational transconductance amplifiers," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 54, no. 3, pp. 481–491, Mar. 2007.
[35] P. Hasler, B. Minch, and C. Diorio, "An autozeroing floating-gate amplifier," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 48, no. 1, pp. 74–82, Jan. 2001.
[36] S. Shah and J. Hasler, "Tuning of multiple parameters with a BIST system," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 64, no. 7, pp. 1772–1780, Jul. 2017.
[37] K. M. Odame and P. Hasler, "Theory and design of OTA-C oscillators with native amplitude limiting," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 56, no. 1, pp. 40–50, Jan. 2009.
[38] J. Hasler, "Circuit implementations teaching a junior level circuits course utilizing the SoC FPAA," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Florence, Italy, May 2018, pp. 1–5.
[39] A. Natarajan and J. Hasler, "Modeling, simulation and implementation of circuit elements in an open-source tool set on the FPAA," *Anal. Integr. Circuits Signal Process.*, vol. 91, no. 1, pp. 119–130, Apr. 2017.
[40] J. Hasler, S. Kim, and F. Adil, "Scaling floating-gate devices predicting behavior for programmable and configurable circuits and systems," *J. Low Power Electron. Appl.*, vol. 6, no. 13, 2016, pp. 1–19.

Jennifer Hasler (Senior Member, IEEE) received the M.S. and B.S.E. degrees in electrical engineering from Arizona State University in 1991, the Ph.D. degree in computation and neural systems from the California Institute of Technology, in 1997, and the Master of Divinity from Emory University in 2020. She is currently a Full Professor with the School of Electrical and Computer Engineering, Georgia Institute of Technology. Dr. Hasler received the Paul Raphorst Best Paper Award from the IEEE Electron Devices Society in 1997, the Best Paper Award from SCI in 2001, the NSF CAREER Award in 2001, the ONR YIP Award in 2002, the Best Paper from CICC in 2005, the Best Sensor Track Paper from ISCAS in 2005, the Best Paper Award from Ultrasound Symposium in 2006, and the Best Demonstration Paper Award from ISCAS in 2010.

Aishwarya Natarajan received the B.S. degree from University of Mumbai, India, and the M.S. degree from the Georgia Institute of Technology, Atlanta, GA, USA, where she is currently pursuing the Ph.D. degree in electrical and computer engineering. Her research interests include analog and mixed signal integrated circuits and systems design, neuromorphic computing, and low-power bio-inspired circuits and systems.