# Accelerating Nonlinear DC Circuit Simulation with Reinforcement Learning

Zhou Jin[1], Haojie Pei[1], Yichao Dong[2], Xiang Jin[3], Xiao Wu[4], Wei W. Xing[3] and Dan Niu[2]
1.Super Scientific Software Laboratory, China University of Petroleum-Beijing, Beijing, China
2. School of Automation, Southeast University, Nanjing, China
3. Beihang University, Beijing, China 4. Huada Empyrean Software Co. Ltd, Beijing, China
Emails:jinzhou@cup.edu.cn, haojiepei@student.cup.edu.cn, 220201739@seu.edu.cn, ZY2141113@buaa.edu.cn,
wuxiao@mail.empyrean.com.cn, wxing@buaa.edu.cn, 101011786@seu.edu.cn.

## ABSTRACT

DC analysis is the foundation for nonlinear electronic circuit simulation. Pseudo transient analysis (PTA) methods have gained great success among various continuation algorithms. However, PTA tends to be computationally intensive without careful tuning of parameters and proper stepping strategies. In this paper, we harness the latest advancing in machine learning to resolve these challenges simultaneously. Particularly, an active learning is leveraged to provide a fine initial solver environment, in which a TD3-based Reinforcement Learning (RL) is implemented to accelerate the simulation on the fly. The RL agent is strengthen with dual agents, priority sampling, and cooperative learning to enhance its robustness and convergence. The proposed algorithms are implemented in an out-of-the-box SPICE-like simulator, which demonstrated a significant speedup: up to 3.1X for the initial stage and 234X for the RL stage.

## CCS CONCEPTS

• **Hardware → Methodologies for EDA**; **Software tools for EDA**.

## KEYWORDS

Circuit simulation, DC analysis, nonlinear equations, pseudo transient analysis, reinforcement learning

## 1 INTRODUCTION

Direct current (DC) analysis, which locates DC operating points, is an important step to evaluate integrated circuits and is performed prior to any other analysis in SPICE-like transistor-level circuit simulators [1, 2]. It provides an initial solution for transient analysis and determines small signal model parameters of nonlinear devices in AC analysis. During the procedure, the challenge is to solve a set of nonlinear algebraic equations established from modified nodal analysis.

There are several numerical iterative algorithms for solving these systems of nonlinear algebraic equations, including basic Newton-Raphson (NR) method and continuation methods, e.g. Gmin stepping [3], source stepping[4], PTA[5], homotopy methods[6] and etc. However, the convergence of Gmin and source stepping are often inferior particularly when solving a highly nonlinear system. Homotopy, in contrast, is difficult to be implemented in actual simulations because it is highly dependent on the device model. As an alternative, PTA and its variants, such as Damped PTA (DPTA), Ramping PTA (RPTA), and Compound element PTA (CEPTA), have proven to be the most practical and principal solvers in the industry because they are easy to implement and have no discontinuity issues[7]. The PTA methods simply the complex problems of solving a set of nonlinear algebraic equations by inserting pseudo-elements into the circuit and turning it into solving the steady-state problem of a system of ordinary differential equations (ODE)[3]. The resulting ODE is highly dependent on the inserting pseudo-elements. For different circuits, the required inserting pseudo-elements can be totally different in order to achieve an easy-to-solve ODE system rather than an ill-defined system (e.g., containing bifurcation, fold, and ill-conditioned matrices). Unfortunately, there is no rule of thumb on the selections of inserting pseudo-elements for a given circuit. Various transistor models and circuit behaviors indicate that there is no single specific choice can be satisfactory under all circumstances, which hinders the application of PTA solvers. The most common solutions are ad-hoc selections based on the circuit types, expert experiences, and exhausting fine tuning. With this problem unsolved, the PTA solvers remain low-efficient.

Once an ODE system is obtained from the PTA solver, it is then solved iteratively through numerical integration stepping towards the steady state. The stepping strategy in PTA determines the number of nonlinear equations need to be solved at discrete time points, where time- and resource- consuming NR iterations are envolved. However, choosing the stepping scheme is nontrivial [8].

Despite the quick advancements in PTA methods for DC analysis, these two challenges remain the main obstacles for the applications of PTA methods. Some pioneer works have proposed heuristic approaches to accelerate PTA methods [7, 8]. However, as the device non-linearity continues to grow and the number of parasitics exponentially increases, the gains from many prior heuristics quickly fall behind the demands of simulations[9].

In this paper, we utilize cutting-edge machine learning techniques to resolve both challenges simultaneously. Particularly, we treat the DC analysis as a classic RL problem. Our proposed two stage acceleration framework is shown in Fig. 1. In the first stage, we implement an offline active learning with online prediction to provide a solver with fine initial parameters, which forms a fine environment for our RL setup. In the second stage, we propose an RL scheme to accelerate PTA iterative process with TD3 actor-critic agents. By considering the PTA iterative process as a RL setup and using residual, relative changes of solutions, the number of NR iterations, etc, to represent the states, we train the dual agents through interaction with the simulation process and dynamically adjust forward and rollback time-step size to accelerate reaching the final steady state.


*DAC '22, July 10–14, 2022, San Francisco, CA, USA*

ACM ISBN 978-1-4503-9142-9/22/07…$15.00
https://doi.org/10.1145/3489517.3530512

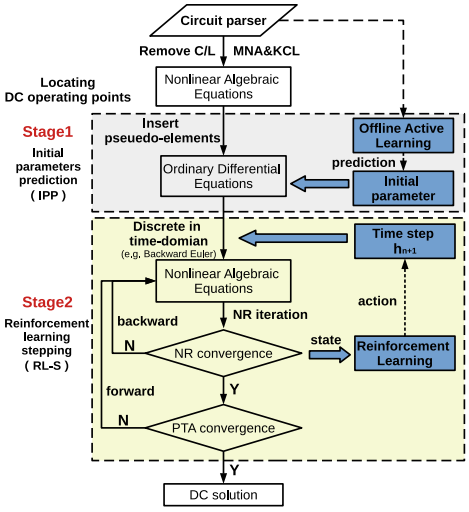


**Figure 1: Overall framework of proposed machine learning enhanced PTA.**

The novelty of this work is as follows,

(1) To the best of our knowledge, our work is the first RL enhanced PTA solver, enabling efficient time-stepping in numerical iterations and generating a substantial speedup over the SOTA PTA solvers.

(2) We provide an effective initial parameters prediction model for any unseen circuit based on a limited number of training circuits. This is done efficiently through a novel self-training active learning procedure by extending the classic Bayesian optimization.

(3) Cutting-edge machine learning techniques, e.g., dual agents, cooperative learning from public sample buffer, and TDError-based priority sampling are implemented to improve our TD3-based RL for insufficient and imbalanced data challenges.

(4) The proposed two-stage framework has been implemented in an out-of-the-box SPICE-like simulator and is verified by extensive circuit simulations. Significant acceleration is achieved, i.e., a maximum 3.1X for the initial stage and 234X for RL stage speedup is demonstrated on practical MOS and BJT circuits.

## 2 BACKGROUND

### 2.1 PTA and time-step control method

PTA is the currently most powerful and promising continuation approach to deal with the non-convergence in DC analysis caused by discontinuity and strong nonlinearity. It inserts specific pseudo elements, e.g. capacitors and inductors, into the circuit, so that the original hard-to-solve nonlinear algebraic equations $\boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{0}$ (where $\boldsymbol{F}(\cdot) : \mathbb{R}^m \rightarrow \mathbb{R}^m$, $\boldsymbol{x} = (\boldsymbol{v}, \boldsymbol{i})^T \in \mathbb{R}^m$, $m = N + M$, variable vector $\boldsymbol{v} \in \mathbb{R}^N$ denotes node voltage, and vector $\boldsymbol{i} \in \mathbb{R}^M$ represents internal branch current) are transferred to an initial value problem for ordinary differential equations $\boldsymbol{P}(\boldsymbol{x}(t), d\boldsymbol{x}(t)/dt, t) = \boldsymbol{F}(\boldsymbol{x}) + \boldsymbol{D} * \dot{\boldsymbol{x}}(t) = 0$, where $\dot{\boldsymbol{x}}(t) = (\dot{\boldsymbol{v}}(t), \dot{\boldsymbol{i}}(t))$, $and \boldsymbol{D}$ represents for the incidence matrix of inserted pseudo elements.

Use implicit numerical integration algorithms, e.g. (1), to discrete in time-domain and finally get the steady state through difference approximation of the differential term iteratively.

$$\dot{\boldsymbol{x}}(t)|_{t=t_{n+1}} = (\boldsymbol{x}_{n+1} - \boldsymbol{x}_n)/h_{n+1} \tag{1}$$

Conventional PTA methods use a simple iteration counting method [10] to determine time-step size. This strategy implements a stepping strategy through two options (IMAX and IMIN). It compares the number of NR iterations at each time-point with options to determine the next time-step size. The advantage of this stepping strategy is that the time step can be increased simply and quickly. However, it is a very difficult problem to select appropriate parameters for different circuits, including IMAX, IMIN, initial time step, and time step growth rate, etc. Another adaptive time-step control method was proposed in [8] based on Switched Evolution/Relaxation(SER). It is a heuristic method with domain experiences employed, demonstrated great potential in speedup that can be obtained through intelligent time step control.

### 2.2 Reinforcement learning

Reinforcement learning is a methodology of learning from an agent in the process of interacting with the environment to maximize rewards or achieve specific goals. The Markov decision process (MDP) is at the centrepiece of most RL models. A MDP can be expressed as a tuple containing 5 main components, $s$(state), $a$(action), $p$(policy), $r$(reward) and $\gamma$(gamma). A value function (e.g. Q-value function) is used to evaluate the value of a certain action $a$ in a Markov decision $\pi$ at the state of $s$. According to the Bellman expectation equation, the value of Q can be obtained as follow: $Q^{\pi}(s, a) = \mathbb{E}[r' + \gamma Q^{\pi}(s', a') | S_t = s, A_t = a]$, where $s', a', r'$ are the state, action and reward at the next moment. Thus, RL is ultimately to find the largest value function in the interaction between agent and environment as follow: $Q^*(s, a) = \max_{\pi} Q^{\pi}(s, a), \forall s \in \mathcal{S}, \forall a \in \mathcal{A}$.

The TD3(Twin Delayed Deep Deterministic policy gradient) [11] is a SOTA deterministic strategy RL algorithm based on the actor-critic [12] model, which is suitable for environment with high dimensional continuous action spaces. It uses double DQNs to fit optimal Q-value function and effectively overcomes the Q-value overestimation problem in DDPG [13]. Besides, it employs delayed policy updates and target policy smoothing regularization to deal with high variance estimate influence and improve the stability.

### 2.3 Gaussian process

Gaussian process (GP) is a common choice as a surrogate model for building the input-output mapping for complex computer code due to its model flexibility and uncertainty quantification. For the sake of clarity, let us consider a case where the circuit is fixed and its index $\boldsymbol{\xi}$ is thus omitted. Assume that we have observation $y_i = \eta(\mathbf{z}_i) + \varepsilon$ and design points $\mathbf{z}_i$, $i = 1, \ldots, N$, where $y$ is the (determined) iteration numbers needed for convergence. In a GP model we place a prior distribution over $\eta(\mathbf{z})$ indexed by $\mathbf{z}$: $\eta(\mathbf{z})|\boldsymbol{\theta} \sim GP(m(\mathbf{z}), k(\mathbf{z}, \mathbf{z}'|\boldsymbol{\theta}))$, where the mean function is normally assumed zero, i.e., $m_0(\mathbf{z}) \equiv 0$, by virtue of centering the data. The covariance function can take many forms, the most common being the automatic relevance determinant (ARD) kernel: $k(\mathbf{z}, \mathbf{z}'|\boldsymbol{\theta}) = \theta_0 \exp(-(\mathbf{z}-\mathbf{z}')^T diag(\theta_1, \ldots, \theta_l)(\mathbf{z}-\mathbf{z}'))$. The hyperparameters $\boldsymbol{\theta} = (\theta_0, \ldots, \theta_l)^T$. $\theta_1^{-1}, \ldots, \theta_l^{-1}$ in this case are called the square correlation lengths. For any fixed $\mathbf{z}$, $\eta(\mathbf{z})$ is a random variable. A collection of values $\eta(\mathbf{z}_i)$, $i = 1, \ldots, N$, on the other hand, is a partial realization of the GP. Realizations of the GP are deterministic functions of $\mathbf{z}$.

The main property of GPs is that the joint distribution of $\eta(\mathbf{z}_i)$, $i = 1, \ldots, N$, is multivariate Gaussian. Assuming the model inadequacy $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ is also a Gaussian, with the prior and available data $\mathbf{y} = (y_1, \ldots, y_N)^T$, we can derivative the model likelihood $(\mathbf{y}^T(\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{y} - \ln|\mathbf{K} + \sigma^2\mathbf{I}| - \log(2\pi))/2$ where the covariance matrix $\mathbf{K} = [K_{ij}]$, in which $K_{ij} = k(\mathbf{z}_i, \mathbf{z}_j)$, $i, j = 1, \ldots, N$. The hyperparameters $\boldsymbol{\theta}$ are normally obtained from point estimates [14] by

maximum likelihood estimate (MLE) of w.r.t. $\boldsymbol{\theta}$. The joint distribution of $\mathbf{y}$ and $\eta(\mathbf{z})$ also form a joint Gaussian distribution. Conditioning on $\mathbf{y}$ provides the conditional Gaussian distribution at $\mathbf{z}$ [14] with mean and variance.

$$\hat{\eta}(\mathbf{z})|\mathbf{y},\boldsymbol{\theta} \sim \mathcal{N}\left(\mu(\mathbf{z}|\boldsymbol{\theta}), v(\mathbf{z},\mathbf{z}'|\boldsymbol{\theta})\right), \mu(\mathbf{z}) = \mathbf{k}(\mathbf{z})^T \left(\mathbf{K}+\sigma^2\mathbf{I}\right)^{-1}\mathbf{y}$$
$$v(\mathbf{z}) = \sigma^2 + k(\mathbf{z},\mathbf{z}) - \mathbf{k}^T(\mathbf{z})\left(\mathbf{K}+\sigma^2\mathbf{I}\right)^{-1}\mathbf{k}(\mathbf{z}) \quad (2)$$

Based on the posterior in (2), we can optimize $\mathbf{z}$ by sequentially quarrying points such that each point shows an improvement $I(\mathbf{z}) = \max(\hat{\eta}(\mathbf{z}) - y^\dagger, 0)$, where $y^\dagger$ is the current optimal and $\hat{\eta}(\mathbf{z})$ is the predictive posterior in (2). Integrating out the posterior achieve the expected improvement(EI):$EI(\mathbf{z}) = \mathbb{E}_{\hat{\eta}(\mathbf{z})}[\max(\hat{\eta}(\mathbf{z}) - y^\dagger, 0)]$, which has a closed form solution $(\mu(\mathbf{z}) - y^\dagger)\psi(u(\mathbf{z})) + v(\mathbf{z})\phi(u(\mathbf{z}))$ in which $\psi(\cdot)$ and $\phi(\cdot)$ are the probabilistic density function (PDF) and cumulative density function (CDF) of a standard normal distribution, respectively. The candidates for next iteration is selected by $\text{argmax}_{\mathbf{z}\in\mathcal{X}} EI(\mathbf{z})$ with on-convex optimizations, e.g., L-BFGS-B.This is the basic procedure of a Bayesian optimization.

## 3 INITIAL PARAMETERS PREDICTION

### 3.1 Problem Formulation

Consider a PTA solver $g$ with initial parameters $\mathbf{z}$ (indicating the value of inserted pseudo-capacitor, pseudo-inductor and time-constant Tau) that operates on a netlist file denoted as $\boldsymbol{\xi}$ and generates the steady state $\boldsymbol{u} = g(\mathbf{z},\boldsymbol{\xi})$. We are interested in reducing the number of iterations, denoted as $\eta(\mathbf{z},\boldsymbol{\xi}) + \varepsilon$, for $g(\mathbf{z},\boldsymbol{\xi})$. Here $\varepsilon$ captures the model inadequacy and randomness that are not fully captured by $\mathbf{z}$ and $\boldsymbol{\xi}$. We aim to seek a function $\mathbf{z}^*(\boldsymbol{\xi}) = \text{argmin}_{\mathbf{z}\in\mathcal{X}}\,\eta(\mathbf{z},\boldsymbol{\xi})$, where $\mathbf{z}^*(\boldsymbol{\xi})$ is the optimal PTA solver parameters for any given netlist $\boldsymbol{\xi}$, and $\mathcal{X}$ is the feasible domain for $\mathbf{z}$. Note that this is not an optimization problem because we are not allowed to run $\eta(\mathbf{z},\boldsymbol{\xi})$ for an unseen circuit. Instead, our goal is to find the mapping $\mathbf{z}^*(\boldsymbol{\xi})$, which is a straightforward supervised learning problem given that we have sufficient data.

We can simply run a classic MC method to locate the best solver parameters to provide the training dataset. However, there are critical issues with this approach: (1) this approach is in general computationally expensive as we need to search the solver parameters space $X$ for all available circuits $\boldsymbol{\xi}$; (2) only the best parameters are included, whereas the majority of the training data are wasted, leading to an inferior model; (3) most critically, the best solver parameters are potentially multi-model, i.e., there is more than one best solver parameter that produces the best performance. To resolve these challenges simultaneously, we resolve $\mathbf{z}^*(\boldsymbol{\xi})$ approximately based on

$$\mathbf{z}^*(\boldsymbol{\xi}) = \underset{\mathbf{z}\in\mathcal{X}}{\text{argmin}}\,\tilde{\eta}(\mathbf{z},\boldsymbol{\xi}|\mathcal{D}), \quad (3)$$

where $\tilde{\eta}$ is a surrogate model approximating $\eta$ based on data $\mathcal{D}$. Our solution is a two-stage approach including (1) the online stage of solving Eq.(3) and the offline stage which build the training dataset $\mathcal{D}$ for the online stage.

### 3.2 Parameter transformation

Our model needs to handle two types of parameters: circuit characters and solver parameters, which have different correlations, scales, and ranges. To resolve this issue, we assume a separable kernel function and follow the work of [9] to introduce different transformations for them, $k([\mathbf{z},\boldsymbol{\xi}],[\mathbf{z}',\boldsymbol{\xi}']) = k_x(\Psi(\mathbf{z}),\Psi(\mathbf{z}'))\cdot k_{\tilde{\xi}}(\Phi(\boldsymbol{\xi}),\Phi(\boldsymbol{\xi}'))$, where $\Psi(\mathbf{z})$ and $\Phi(\boldsymbol{\xi})$ are functional transformation for $\mathbf{z}$ and $\boldsymbol{\xi}$, respectively. In this work, $\Phi(\boldsymbol{\xi})$ utilizes a deep feed forward fully connected network as an automatic feature extraction for $\boldsymbol{\xi}$ as in [9], whereas $\Psi(\mathbf{z})$ denotes a reparameterization for $\mathbf{z}$ such that the range and scale is handled properly. Specifically, we force $\log(z_d) = (7\cdot \text{sigmoid}(w_d))$, where $\text{sigmoid}(w_d) = 1/(1+\exp(w_d))$ is the sigmoid function. With this transformation, $\mathbf{z}$ is naturally constrained in the range of $[10^{-7}, 10^7]$. Also, the new parameters $w_d$ focuses on the scale of $\mathbf{z}$ rather than its particular value. This is particularly handy when we later optimize the number of iteration w.r.t. $\mathbf{w}$ instead of $\mathbf{z}$.

For the original circuit characters, we follow [15] and [9] and use seven key factors (the number of nodes, MNA equations, independent current sources, resistors, voltage sources, bipolar junction transistor, and MOS field-effect transistor) to characterize a netlist and a flag $\{0,1\}$ denoting whether this circuit is a BJT or MOS type circuit. We know that BJT and MOS type circuits have completely different *a-priory*. Thus the transformation should reflect this knowledge. To this end, we further modify the kernel structure as

$$k([\mathbf{z},\boldsymbol{\xi}],[\mathbf{z}',\boldsymbol{\xi}']) = \left(k_x^1(\Psi(\mathbf{z}),\Psi(\mathbf{z}'))\cdot k_{\tilde{\xi}}^1(\Phi(\boldsymbol{\xi}),\Phi(\boldsymbol{\xi}'))\right)^{\tau} \times \left(k_x^2(\Psi(\mathbf{z}),\Psi(\mathbf{z}'))\cdot k_{\tilde{\xi}}^2(\Phi(\boldsymbol{\xi}),\Phi(\boldsymbol{\xi}'))\right)^{1-\tau} \quad (4)$$

where $\tau = \{0,1\}$ indicates the BJT and MOS type circuit.

### 3.3 Offline training and online predictions

With the circuit characters and solver parameters being handled properly, we now discuss the offline stage of building the model and training data $\tilde{\eta}(\mathbf{z},\boldsymbol{\xi}|\mathcal{D})$ and the online stage, in which a new circuit is given and a set of predictive best solver parameters is suggested.

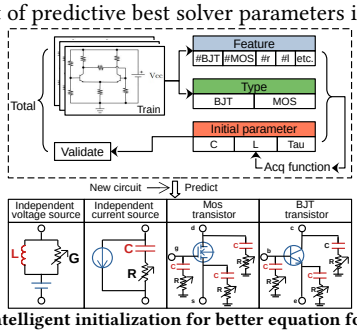


**Figure 2: Intelligent initialization for better equation formulation.**

Let $\boldsymbol{\xi}^\dagger$ denotes a validating data circuits unseen to our system and $\mathbf{z}_{\boldsymbol{\xi}^\dagger}$ the optimal solver parameters for it. Without loss of generality, assuming that $\mathcal{D}$ is constructed in a sequential manner, i.e., adding a point at a time. At t-th iteration, the regret for $\boldsymbol{\xi}^\dagger$ is $R_t^{\boldsymbol{\xi}^\dagger} = \eta(\hat{\mathbf{z}},\boldsymbol{\xi}^\dagger) - \eta(\mathbf{z}_{\boldsymbol{\xi}^\dagger},\boldsymbol{\xi}^\dagger)$, where $\hat{\mathbf{z}}$ is the proposition, $\mathbf{z}_{\boldsymbol{\xi}^\dagger}$ is the (unknown) best solver parameters. The proposed solver parameters is normally obtained based on Bayesian optimization. For instance, based on the EI approach, we can optimize,

$$\hat{\mathbf{z}} = \underset{\mathbf{z}\in\mathcal{X}}{\text{argmax}}\,\mathbb{E}_{\hat{\eta}(\mathbf{z})}\left[\max\left(y^\dagger - \hat{\eta}(\mathbf{z},\boldsymbol{\xi}^\dagger), 0\right)\right], \quad (5)$$

where $y^\dagger$ indicates the minimum number of iteration so far or simply the number of iteration for the default PTA solver parameters. For our problem, we need to modify the Bayesian optimization formulation because we neither know $\boldsymbol{\xi}^\dagger$ in advance nor are allowed to run simulation to approach the best solver parameter with a few iterations. Notice that any candidate among our training circuits $\boldsymbol{\xi} = [\boldsymbol{\xi}_1,\cdots,\boldsymbol{\xi}_N]^T$ can be used as $\boldsymbol{\xi}^\dagger$, if any data associated with it is excluded. For our training, we thus define a total regret as $R_T =$

$\sum_{t=1}^{T}\sum_{n=1}^{N} r_t^{\xi_n}$. For each iteration when circuit $\xi_n$ is selected, its associated data is excluded from the GP surrogate. More specifically, the optimization of (5) is augmented as

$$\hat{\mathbf{z}} = \underset{\mathbf{z}\in\mathcal{X}}{\operatorname{argmax}}\, \mathbb{E}_{\hat{\eta}(\mathbf{z})}\left[\max\left(y^{\dagger} - \tilde{\eta}\left(\mathbf{z}, \xi^{\dagger} | \mathcal{D}^{t}_{\setminus\xi^{\dagger}}\right), 0\right)\right], \tag{6}$$

where $\min \tilde{\eta}\left(\mathbf{z}, \xi^{\dagger} | \mathcal{D}^{t}_{\setminus\xi^{\dagger}}\right)$ is the optimal solver iteration based on the solver parameters for the current $\mathcal{D}^{r}_{\setminus\xi^{\dagger}}$, which indicates that any data corresponding to $\xi^{\dagger}$ circuit is excluded. We can now achieve the minimum total regret by iteratively solving Eq. (6) with enumerations of all candidate circuits. The detailed offline training is shown in Algorithm 1.

**Algorithm 1** Bayesian Active Learning for Offline Stage

**Input:** Netlists $\Xi$, number of iteration $M$, PTA solver $\eta(\mathbf{z}, \xi)$, initial data $\mathcal{D}$
**Output:** Surrogate model $\tilde{\eta}(\mathbf{z}, \xi)$ and data $\mathcal{D}$

```
for i = 1 to M do
  for n = 1 to N do
    Update GP η̃(z, ξ) model based on data D
    Find z* for ξ_n by solving Eq. (6) based on D_\ξ_n
    Execute y = η(z*, ξ_n)
    Update data D = D ∩ {z*, ξ_n, y}
  end for
end for
```

At the online stage,with the model after offline training and given a new circuit $\xi^{*}$, we can simply optimize $\tilde{\eta}(\mathbf{z}, \xi^{*}|\mathcal{D})$ w.r.t. $\mathbf{z}$, i.e., solving Eq.(3) to propose the best predictive solver parameters.

## 4 REINFORCEMENT LEARNING STEPPING

In this section, we propose a reinforcement learning stepping (RL-S) technique for PTA iteration, in which we can improve simulation efficiency based on how the simulator works internally.

### 4.1 TD3-based RL-S algorithm

Here we present key concepts of the proposed RL-S, mapping RL process to the PTA simulation problem. Considering the whole simulation as the environment, aiming at fast convergence, RL-S explores a fast stepping strategy so that the ODE system reaches the steady state sufficiently fast. Regarding the iterative solution trace process of PTA as MDPs, at each time-point $t_n$, we can get a simulation state $S_n$, defined by the number of NR iterations, residual of equation $\boldsymbol{F}(\boldsymbol{x})$, relative change of solution compared with previous step and two flags to demonstrate the convergence performance of NR iterations and PTA iterations, respectively, as shown in Table. 1. Such simulation state could help evaluate wheather the simulation trends to convergence gradually or not, representing a "convergence distance". Then an action $a$ is determined by an actor network to give a time-step size $h_{n+1}$ for next time-point iteration as shown in Fig.3.

Considering sufficiently fast convergence is our final goal, any action leads to better simulation state, defined as better trends to final solution, deserves higher reward. The most powerful indicator of circuit simulation strategy is the time spent in simulation. Besides, the gradual learning of optimal strategy should be fast and stable. We choose Actor-Critic based TD3 algorithm as reinforcement learning model to achieve this goal. Among the numerous reinforcement learning algorithms, TD3 demonstrates a more stable and smooth performance for training. Details are shown in Algorithm2. TD3 algorithm utilizes two sets of Q networks to fit the value function on the basis of DDPG and selects the minimum value each time as

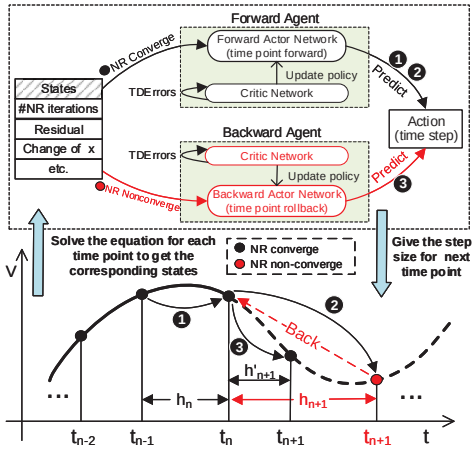


**Figure 3: Reinforcement learning workflow based on TD3 in simulation iterations. ❶ ❷ predicted through the forward agent, ❸ predicted through the backward agent.**

the target of learning, which greatly reduce the overestimation compared with DDPG Q value network, as shown in Algorithm2(line 11). TD3 also introduces soft update, actor-critic asynchronous update, and the idea of adding estimated noise, which further improved the stability of the algorithm.

**Algorithm 2** TD3-based Reinforcement Learning Stepping

1: Initialize critic networks $Q_{\theta_1}$,$Q_{\theta_2}$,and actor network $\pi_{\phi}$
2: Initialize target networks $\theta_1' \leftarrow \theta_1, \theta_2' \leftarrow \theta_2, \phi' \leftarrow \phi$
3: Initialize private sample buffer$\mathcal{B}_{\alpha}$ ,$\mathcal{B}_{\beta}$ and public buffer $\mathcal{B}$
**Input:** Netlist $\tau$
**Output:** The optimal time stepping strategy $\pi^{*}$
4: Get initial state $s \leftarrow s_0$
5: **while** PTA iteration does not converge **do**
6: Select agent from dual-agents according to NR flag
7: Determine action with exploration noise $a(s) + \epsilon$ , $\epsilon \in \mathcal{N}(0, \sigma)$ observe reward $r$ and new state $s'$ by selected agent
8: Store transition tuple $(s, a, r, s')$ using **Collaborative leaning**
9: Sample mini-batch size of $N$ transitions $(s, a, r, s')$ from buffer $\mathcal{B}$ and $B_{\alpha}$ or $\mathcal{B}$ and $B_{\beta}$ based on **Priority sampling**
10: $\tilde{a} \leftarrow \pi_{\phi'}(s') + \epsilon, \quad \epsilon \sim \mathrm{clip}(\mathcal{N}(0, \tilde{\sigma}), -c, c)$
11: $y \leftarrow r + \gamma \min_{i=1,2} Q_{\theta_i'}(s', \tilde{a})$
12: Update critics $\theta_i \leftarrow \operatorname{argmin}_{\theta_i} N^{-1} \sum \left(y - Q_{\theta_i}(s, a)\right)^2$
13: **if** *step mod d* **then**
14: Update $\phi$ by the deterministic policy gradient
15: $\nabla_{\phi} J(\phi) = N^{-1} \sum \nabla_a Q_{\theta_1}(s, a)\big|_{a=\pi_{\phi}(s)} \nabla_{\phi} \pi_{\phi}(s)$
16: Update target networks
17: $\theta_i' \leftarrow \tau\theta_i + (1-\tau)\theta_i'$ ; $\phi' \leftarrow \tau\phi + (1-\tau)\phi'$
18: **end if**
19: **end while**

The training algorithm in RL-S belongs to the category of online learning. Though we could leverage some offline learning by pre-training the parameters of RL network with several typical circuits to accelerate the convergence, directly using the fixed pre-trained model for any new circuit prediction can not guarantee the optimal step size in the majority circumstances due to the huge differences among circuit behaviors. The combination of offline pre-training and online learning for each circuit during its simulation, especially from those just experienced simulation states, help improve the strategy with better self adaptive ability for any new circuit.

**Table 1: Observable states from simulation environment**

| States | Brief Descriptions |
|---|---|
| Iters[1] | Evaluate the difficulties of NR convergence |
| Res[2] | Evaluate whether equation is close to final solution |
| $\Gamma$[3] | Evaluate whether solution trends to reach the steady state or still varies heavily |
| $NR^4_{flag}$ | Evaluate whether NR converges or not |
| $PTA^5_{flag}$ | Evaluate whether the PTA steady state is successfully reached |

[1]The number of NR iterations. [2]The residual of the equation.
[3]The relative change of the solution.
[4]The flag of NR convergence. [5]The flag of PTA convergence.

## 4.2 Dual agents for forward/backward stepping

During simulation, most training data are collected from the converged phase with larger time steps. However, in a few cases, small step size is required to ensure its convergence, i.e. circuits with high loop gain. Unfortunately, it happens to be very limited, which leads to very difficult model training. So we set up two agents to deal with these two situations, respectively, as shown in Fig. 3. The first one is suitable for the forward prediction with longer time-step, while the other one is suitable for rollbacking to the previous time point and giving the prediction with smaller time-step.

The action $a$ obtained directly from the agents is normalized between (-1,1) through tanh function. Action $a$ determines time-step size $h$ by multiplying an coefficient for the previous time-step size. During simulation, in order to ensure a fast step-by-step process and enable the forward agent to always predict an increased time step, we select a exponential function to standardize the range of $h$: $h_n = \frac{m}{1-e^{-a+n}} \cdot h_{n-1}$, where $n$ and $m$ are constant parameters and satisfy $m > 1 - e^{-a+n}$, $n > 1$. Similarly, backward agent uses $h_n = \frac{c}{1+e^{-a+b}} \cdot h_{n-1}$ and sets $c < 1 + e^{-a+b}$.

Furthermore, according to the relationship between simulation state and convergence trends, we set the reward function as $r_t = c_1\Gamma + c_2 Iters + c_3 Res + c_4 NR_{flag} + c_5 PTA_{flag}$ to evaluate the reward that agent can get by taking action $a$ under the strategy $\pi$ at state $s$. Rewards are also obtained from forward and backward agents, respectively.

## 4.3 Collaborative learning via public sample buffer

An important indicator of online learning is the speed at which the model converges to the optimal strategy under each environment. Obviously, in the early stage of online learning, there are very few experiences available, which will affect the convergence speed of the model. Furthermore, if the time span of online learning is extended, the time consumed in this process can not be ignored during the simulation. To resolve this problem, we propose a public sample buffer to achieve higher utilization of experiences.

The dual agents mentioned in the previous section can improve the professionalism of handling different circumstances, but also dilute the sample buffer since it divided the past experiences into two parts. If the output action given by forward agent causes the convergence failure of next NR iteration, it's experience will be stored in the forward agent's sample buffer. However, we found that this experience could also benefit the backward agent, which makes it more radically shorten the original step size to avoid such kind of low reward action. Public sample buffer includes such samples that from convergence state to nonconvergence state, similar for the backward agent that samples from nonvergence state to convergence state, as shown in Fig.4. This scheme greatly improves the utilization of samples, thereby accelerating the convergence speed of the agent in the online learning phase, and greatly reducing the time cost of simulation.

## 4.4 Priority sampling for fast convergence

As mentioned earlier, the convergence speed of the model during online training is critical, thus we propose another effective solution: priority sampling. For the gradient update of the Critic network and the Actor network in the TD3 algorithm, larger TD error (Temporal-Difference error) will contribute more for the Critic network updation. Therefore, we can use the priority sampling method to learn from more samples that make great contributions to the network convergence. In this implementation, the absolute value of TD error and the key value of each sample are stored in SumTree (a binary sum tree), where the value of each parent node is equal to the sum of the values of its child nodes. We first sample randomly and uniformly from 0 to the value of the root node, then in the tree with this sample as the root node, find the corresponding leaf node by in order-traversing method, and finally get the priority of the sample we want to collect. During the random sample collection process, the probability of the leaf node being searched will be proportional to the stored value, so a sample with a larger TD error is more likely to be collected. It is worth noting that this method will not completely eliminate those samples that may be beneficial to network learning but have small update weights, which continues the richness of the sample buffer.

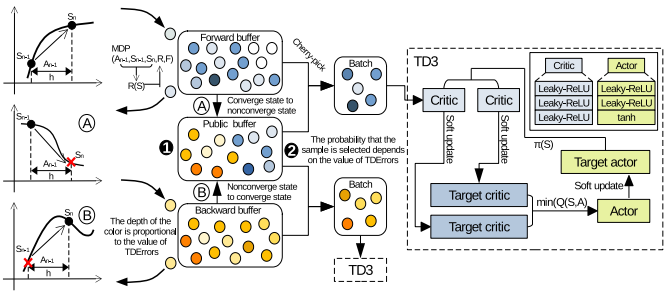


**Figure 4: Updating strategy of reinforcement learning for predicting step size. ❶ The public sample buffer is added to realize the sharing learning of good samples. The sample selection method is that the XOR value marking the convergence of iteration in continuous two states is 1. ❷ Cherry-pick of the samples are used for training. The probability of a sample being selected is directly proportional to its TDErrors value.**

## 5 NUMERICAL EXAMPLES

We have implemented the proposed two-stage acceleration framework in a SPICE-like circuit simulator and evaluated their performance with dozens of commonly used transistor circuits.

We demonstrate the speedup over default CEPTA[7] setting based on our IPP in Table 2. Here, 43 canonical benchmark circuits were used as training set whereas 7 practical circuits (including two MOS circuits and five bipolar junction transistor circuits) are tested. We can see a steady 2X-3X NR iterations speedup without performance degradation. More importantly, IPP can make non-convergence case

converge, which is highly desirable in our applications by providing a feasible working space for the follow-up RL procedure.

**Table 2: Simulation efficiency of initial parameters prediction.(# of NR)**

| Circuits | Type | #Nodes | #Elem | CEPTA | IPP | Speedup |
|---|---|---|---|---|---|---|
| Adding | MOS | 15 | 18 | 272 | 95 | 2.86 |
| MOSBandgap | MOS | 34 | 16 | 153 | 93 | 1.65 |
| 6stageLimAmp | BJT | 80 | 35 | 72 | 34 | 2.12 |
| TRCKTorig | BJT | 15 | 20 | 53 | 34 | 1.56 |
| UA709 | BJT | 42 | 39 | 407 | 223 | 1.83 |
| UA733 | BJT | 22 | 31 | 121 | 39 | 3.10 |
| D22 | BJT | 6 | 8 | N/A | 111 | - |

In order to assess our RL-S scheme, we compared it with adaptive stepping (widely used SOTA PTA stepping)[8] and the simple stepping. The speedup on 27 circuits are shown in Fig. 5. It is clear that RL-S scheme consistently outperforms adaptive stepping up to 3.77X and simple stepping 2.71X in terms of NR iterations in CEPTA. Similar speedups are observed for the numbers of PTA steps, indicating that our RL-S method can achieve both inter- and outer-NR loops speedup and will not cause degeneration.

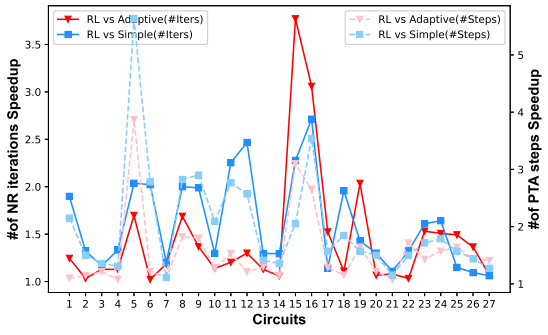


**Figure 5: Speed-up of RL-S over conventional stepping strategies for CEPTA.**

Our RL-S is compatible to all kinds of PTA solver. As a demonstrate, we assessed it in DPTA[10] and showed the result in Table 3. Significant improvement in RL-DPTA over the RL-CEPTA is observed. Particularly, The RL-DPTA always achieves the largest percentage of the time point reduction both in two comparisons (up to **99.79%**), whose average PTA steps are 60.57% over adaptive method, showing an excellent efficiency of our RL-S. The main differences between them is the ability of dealing with oscillation. CEPTA uses time-variant pseudo-resistors and conductances to eliminate oscillation caused by pseudo-capacitors and inductors. However, additional inserted resistor and conductance may bring discontinuity, where though time-step size is sufficiently small, convergence can not be obtained. DPTA resolves this problem through artificially enlarged damping effect in numerical integration algorithm, which ensures continuity and shows better applicability with RL-S. Through RL-S, the efficiency has been substantially improved for DPTA method.

## 6 CONCLUSIONS

In this paper, we proposed a fast DC analysis method. Cutedge active learning and RL are used to accelerate the currently most powerful and promising continuation approach-PTA in a two-stage acceleration framework. Proposed RL-S is compared with adaptive and conventional simple iteration time-step control methods, demonstrated an acceleration up to 234.23X in DPTA.

## ACKNOWLEDGEMENT

We deeply appreciate the invaluable comments from the reviewers. Wei W.Xing and Dan Niu are the corresponding authors of this paper.

**Table 3: Simulation efficiency comparisons between proposed RL-S and adaptive stepping methods for DPTA.**

| Circuits | Adaptive | | RL-S | | Speedup(RL-S vs.) | |
|---|---|---|---|---|---|---|
| | #Ite | #Ste | #Ite | #Ste | #Ite | #Ste |
| astabl | 71 | 17 | 57 | 17 | 1.25X | 0.00% |
| bias | 740 | 112 | 300 | 71 | 2.47X | 36.61% |
| latch | 100 | 25 | 77 | 18 | 1.3X | 28.00% |
| nagle | 1948 | 891 | 364 | 48 | 5.35X | 94.61% |
| rca | 119 | 28 | 69 | 18 | 1.72X | 35.71% |
| ab_ac | 3947 | 1959 | 126 | 31 | 31.33X | 98.42% |
| ab_integ | 4406 | 2167 | 203 | 37 | 21.7X | 98.29% |
| ab_opamp | 2536 | 1210 | 225 | 38 | 11.27X | 96.86% |
| cram | 87 | 36 | 58 | 21 | 1.5X | 41.67% |
| e1480 | 5514 | 2741 | 203 | 37 | 27.16X | 98.65% |
| gm6 | 93 | 39 | 73 | 24 | 1.27X | 38.46% |
| mosrect | 826 | 407 | 85 | 26 | 9.72X | 93.61% |
| schmitfast | 5691 | 2820 | 123 | 32 | 46.27X | 98.87% |
| slowlatch | 9353 | 4649 | 146 | 34 | 64.06X | 99.27% |
| fadd32 | 1859 | 917 | 152 | 50 | 12.23X | 94.55% |
| voter25 | 124 | 45 | 97 | 30 | 1.28X | 33.33% |
| gm1 | 47 | 21 | 40 | 19 | 1.18X | 9.52% |
| gm17 | 152 | 36 | 86 | 22 | 1.77X | 38.89% |
| todd3 | 9341 | 4645 | 838 | 121 | 11.15X | 97.40% |
| 6stageLimAmp | 116 | 42 | 69 | 21 | 1.68X | 50.00% |
| D10 | 126 | 29 | 102 | 22 | 1.24X | 24.14% |
| D11 | 205 | 44 | 115 | 26 | 1.78X | 40.91% |
| DCOSC | 130 | 34 | 88 | 20 | 1.48X | 41.18% |
| mosamp | 239 | 97 | 144 | 27 | 1.66X | 72.16% |
| MOSBandgap | 303 | 135 | 159 | 32 | 1.91X | 76.30% |
| RCA3040 | 119 | 28 | 69 | 18 | 1.72X | 35.71% |
| SCHMITT | 76 | 20 | 47 | 15 | 1.62X | 25.00% |
| TADEGLOW | 143 | 30 | 72 | 18 | 1.99X | 40.00% |
| THM5 | 5324 | 2660 | 142 | 42 | 37.49X | 98.42% |
| TRISTABLE | 77 | 31 | 58 | 21 | 1.33X | 32.26% |
| UA727 | 806 | 294 | 275 | 38 | 2.93X | 87.07% |
| UA733 | 152 | 37 | 95 | 21 | 1.6X | 43.24% |
| MOSMEM | 26000 | 12977 | 111 | 27 | 234.23X | 99.79% |
| Average | - | - | - | - | 16.56X | 60.57% |

#Ite: number of NR iterations #Ste: number of pseudo steps

This work was supported by State Key Laboratory of Computer Architecture (ICT,CAS) under Grant No.CARCHA202115 and Science Foundation of China University of Petroleum, Beijing under Grant No.2462020YXZZ024.

## REFERENCES

[1] J. Deng, K. Batselier, Y. Zhang, and N. Wong, "An efficient two-level dc operating points finder for transistor circuits," in *DAC*, pp. 1–6, 2014.
[2] Z. Jin, T. Feng, Y. Duan, X. Wu, M. Cheng, Z. Zhou, and W. Liu, "PALBBD: A parallel arclength method using bordered block diagonal form for DC analysis," in *GLSVLSI*, pp. 327–332, 2021.
[3] K. Kundert, *The Designer's Guide to SPICE and SPECTRE®*. Springer Science & Business Media, 2006.
[4] T. Najibi, "Continuation methods as applied to circuit simulation," *IEEE Circuits and Devices Magazine*, pp. 48–49, 1989.
[5] C. T. Kelley and D. E. Keyes, "Convergence analysis of pseudo-transient continuation," *SIAM Journal on Numerical Analysis*, pp. 508–523, 1998.
[6] C. Lemke, "Pathways to solutions, fixed points, and equilibria (cb garcia and wj zangwill)," *SIAM Review*, pp. 445–446, 1984.
[7] J. Zhou, L. Meiping, and W. Xiao, "An adaptive dynamic-element pta method for solving nonlinear dc operating point of transistor circuits," in *MWSCAS*, pp. 37–40, 2018.
[8] X. Wu, Z. Jin, D. Niu, and Y. Inoue, "An adaptive time-step control method in damped pseudo-transient analysis for solving nonlinear DC circuit equations," *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, pp. 619–628, 2017.
[9] W. W. Xing, X. Jin, Y. Liu, D. Niu, W. Zhao, and Z. Jin, "Boa-pta, a bayesian optimization accelerated error-free spice solver," *arXiv preprint arXiv:2108.00257*, 2021.
[10] X. Wu, Z. Jin, D. Niu, and Y. Inoue, "A pta method using numerical integration algorithms with artificial damping for solving nonlinear dc circuits," *Nonlinear Theory and Its Applications, IEICE*, vol. 5, pp. 512–522, 2014.
[11] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International Conference on Machine Learning*, pp. 1587–1596, 2018.
[12] C. Millán-Arias, B. J. T. Fernandes, F. Cruz, R. Dazeley, and S. Fernandes, "A robust approach for continuous interactive actor-critic algorithms," *IEEE Access*, pp. 104242–104260, 2021.
[13] F. Rezazadeh, H. Chergui, L. Alonso, and C. V. Verikoukis, "Continuous multi-objective zero-touch network slicing via twin delayed DDPG and openai gym," *CoRR*, 2021.
[14] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. MIT Press, 2006.
[15] S. Zhang, W. Lyu, F. Yang, C. Yan, D. Zhou, and X. Zeng, "Bayesian optimization approach for analog circuit synthesis using neural network," in *DATE*, pp. 1463–1468, 2019.