الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي و البحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Mohamed Khider – Biskra
Faculté des Sciences et de la Technologie
Département : Génie Electrique
Ref :……

جامعة محمد خيضر بسكرة
كلية العلوم و التكنولوجيا
قسم: الهندسة الكهربائية
المرجع:.....

Thèse présentée en vue de l'obtention
du diplôme de

Doctorat en sciences en : Electronique

Option: ***Architecteur de système.***

Application of the Acceleration Method to Solve Systems of Nonlinear Equations in the Modeling of Analog Integrated Systems

Présenté par :
Mr. Dhiabi Fathi

Soutenue publiquement le 05/02/2018

Devant le jury composé de :

Pr. TOBBECHE Souad	**Professeur**	**Président**	**Université de Biskra**
Pr. BOUMEHRAZ Mohamed	**Professeur**	**Rapporteur**	**Université de Biskra**
Pr. DJEFFAL Fayçal	**Professeur**	**Examinateur**	**Université de Batna**
Pr. DEHIMI Lakhdar	**Professeur**	**Examinateur**	**Université de Batna**
Dr. RECHAM Djamil	**MCA**	**Examinateur**	**Université d'O. Bouaghi**
Dr. SAADOUNE Achour	**MCA**	**Examinateur**	**Université de Biskra**

Année universitaire : 2017-2018

Acknowledgements

First of all, I thank **ALLAH**, for giving me the strength during the course of this research work.

I convey my gratefulness to my teacher and supervisor **Prof. BOUMEHRAZ Mohamed** for his support, guidance and encouragement throughout the research period.

I extend my humble thanks to the faculty and staff members of Electrical Department of Mohamed kheider University – Biskra for accepting me as a part of this institution (as a teacher and a student).

I am really honored and proud that **Prof. TOBBECHE Souad** accepted to be the juries' president.

I would like to thank the members of the jury: **Prof. DJEFFAL Fayçal** from the university ELHADJ LAKHDHAR – BATNA, **Prof. DEHIMI Lakhdar** from the university ELHADJ LAKHDHAR – BATNA and **Dr. RECHAM Djamil** from the university LARBI BEN M'HIDI – Oum El Bouaghi.

Also, I want to express my sincere thanks to: **Dr. SAADOUNE Achour, Dr. BEN RAMACHE Said** and **Dr. HASSAN nasre** from the university Ghardaia.

I extend my humble thanks to my whole family for the constant moral support provided by them.I would like to thank all my colleagues and friends whose company and encouragement helped me a lot to work hard.

Dedication

I would like to dedicate my work to:

My Father (May Allah have mercy on him)

My Mother (May Allah bless her)

My wife and my sweet kids

My brothers and sisters and all my family members.

Also, to all my friends those who are close to me or far.

ملخص الأطروحة

إن معظم الوسائل التقنية تستعمل الدوائر التماثلية أو الدوائر المركبة , و تعتبر هذه الدوائر معقدة لاحتوائها على عناصر اللاخطية كأنصاف النواقل مثل الصمام و الترانزستور. قبل تصميم هذه الدوائر تستعمل عملية المحاكاة للتأكد من صحة عمل الدائرة و لتفادي الخسائر المحتملة بعد الانجاز. و لكن يبقى هناك مشاكل في عملية محاكاة الدوائر الالكترونية و خاصة المعقدة, إن المشكل الرئيسي في عملية المحاكاة هو إيجاد النقاط الفعالة بسبب الخوارزمية المختارة في حل أنظمة المعادلات وهو ما يخص بحثنا هذا. حيث سنقوم بتسريع حل أنظمة المعادلات اللاخطية وهذا باستعمال الطريقة المستمرة, بحيث نستعمل هذه الطريقة لتحويل العناصر اللاخطية إلى العناصر المستمرة مما ينتج عنها دائرة تحاكي الواقع.

إن الطريقة المستمرة تم الاعتماد عليها في العقد الأخير وهي على شكل خوارزمية تقوم بالانتقال من معادلة ذات حلول إلى معادلة أصلية مجهولة الحلول باستعمال المتغير المستمر ويطلق عليه متغير هموتوبي. و الخوارزمية تنقسم إلى ثلاث أقسام: القسم الأول تحويل المعادلة اللاخطية إلى معادلة هموتوبي و في القسم الثاني يتم استعمال معالج خاص بالتكهن حول مسار الحلول وذلك باستعمال طريقة الميل , وفي القسم الثالث يتم استعمال معالج خاص بتصحيح المسار نحو الحلول وذلك باستعمال طريقة نيوتن.

في بحثنا هذا نقوم بتسريع حل أنظمة المعادلات اللاخطية بالطريقة المستمرة حيث قمنا بإضافة القسم الرابع ليقوم بتحديث خطوات المتغير هموتوبي وذلك في حالة حدوث خطأ من طرف المعالج الخاص بالتصحيح. إن معالج التصحيح بواسطة نيوتن هي من الدرجة الثانية, وللتسريع نستعمل نيوتن من الدرجة الثالثة. فالمساهمة التي تم تقديمها في هذه الأطروحة ملخصة كما يلي:

1. إضافة الخطوة الرابعة وهي مراقبة وتحديث خطوات المتغير هموتوبي.
2. نستعمل نيوتن من الدرجة الثالثة بدلا من نيوتن رفسن في معالج التصحيح.
3. تكوين معادلة خاصة بالدارة تتم بطريقة حديثة حيث يتم تحويل العناصر الالكترونية على شكل نموذج يحتوي على منابع التغذية المتعلقة بالتوترات الموجودة في العقد و التيارات المارة عبر المولدات الخاصة بالجهد.و من ثم يمكن إيجاد الصيغة العامة للمعادلة المستمرة الخاصة بالدارة .
4. إنشاء تطبيق على نظام ويندوز يطلق عليه بياماس يسهل:
   - إنشاء العناصر على شكل مولدات.
   - إنشاء الدوائر الالكترونية بطريقتين الطريقة النصية أو المخطط الهندسي.
   - تحليل الدوائر الالكترونية.

إن تسريع أنظمة المعادلات اللاخطية الخاصة بالدوائر الالكترونية تم تلخيصه في إنشاء برنامج حاسوبي لتوضيح مدى ايجابيته وفعاليته مقارنة مع النتائج الأخرى. بالإضافة إلى عملية التسهيل في إنشاء نماذج أو عناصر المركبات الالكترونية مع محاكاتها في الدوائر الالكترونية مقارنة مع سبيس.

الكلمات المفتاحية : المعادلات اللاخطية, نيوتن رافسن, هموتوبي,نيوتن من الدرجة الثالثة, نيوتن كاريلوف, سبايس, الدوائر التماثلية, بياماس.

Abstract

All technologies used the electrical circuits with integrated components based on semiconductor devices such as Diodes, BJTs, MOSFETs…etc, it are nonlinear elements. Before construct any circuit, the simulation method should be used to verifying the function of circuit and eliminate of any problem. But there is a problem in the simulation of analog circuits or complex circuits in particular case. The problem is finding the operating point; this problem depends on the method used by the simulator to solve nonlinear equations. The *continuous method* was frequently used in the last ten years; it is an algorithm which deals with the problem of convergence of nonlinear equations using the *continuous* (*homotopy*) parameter. It is consists of three stages: the first stage was converts the nonlinear equation to the *homotopy* equation; the second uses the prediction process to obtain a good approximation of the solution. The last stage based on the correction process which is related on the *Newton Raphson* method.

In this work, we have proposed a new approach to accelerate the solving of nonlinear equations, which can be improve the *continuous method* by adding a new stage in the control of the step length of the continuous parameter.

The main objectives of this work are:

1. The adding a new stage (four stage) to the continuous method for controlling the step length of the continuous parameter.
2. We have used the *Newton order three* in the correction process rather than the *Newton order two* used in the Newton-Raphson method.
3. We have introduced a new idea to converting the analog elements of the circuit of their dependent of voltage or current sources for applying the KCL and the KVL laws to obtain the equation of the complex circuit.
4. We have created new software called PyAMS (**Py**thon for **A**nalog and **M**ixed **S**ignals) based on the proposed method in the windows system, which it is defined by the following software:
    - Modeling the analog element in the form of dependent source.
    - Create circuits by either the textual or the schema method (CAD).
    - Analysis of circuits.

The obtain results show that the using a new software based on the proposed method having a effectiveness in the our approach. Compared to other methods, this method is more efficient. Our new software is faster than the SPICE simulator in analyzing circuits, and offers the modeling of elements contrary to SPICE.

Keywords: Nonlinear equations; Newton Raphson; Homotopy, Newton order three; Newton-Karylov, Analog Circuits, SPICE; PyAMS.

Résumé

Toutes les technologies utilisent des circuits électriques avec des composants intégrés à base de dispositifs semi-conducteurs: Diodes, BJT, MOSFET,... etc., ces dispositifs sont des éléments non linéaires. Avant de construire un circuit, la méthode de simulation doit être utilisée pour vérifier le fonctionnement de ce circuit et éliminer tout problème. Mais il ya un problème dans la simulation des circuits analogiques et des circuits complexes en particulier. Le problème est de trouver le point de fonctionnement; Ce problème dépend de la méthode utilisée par le simulateur pour résoudre des équations non linéaires. La méthode continue a été fréquemment utilisée au cours des dix dernières années; Il s'agit d'un algorithme qui traite du problème de convergence des équations non linéaires à l'aide du paramètre continu (homotopie). Il se compose de trois étapes: la première étape convertit l'équation non linéaire en équation de homotopie; La seconde utilise le processus de prédiction pour obtenir une bonne approximation de la solution. La troisième étape est le processus de correction qui est basé sur la méthode de Newton Raphson.

Dans ce travail, nous proposons une nouvelle approche pour accélérer la résolution d'équations non linéaires basées sur l'amélioration de la méthode continue en ajoutant une nouvelle étape qui est le contrôle de la longueur d'étape du paramètre continu. Les principales contributions de notre travail sont:

1. Ajout de la quatrième étape à la méthode continue pour contrôler la longueur d'étape du paramètre continu.
2. Nous avons utilisé la méthode de Newton d'ordre trois dans le processus de correction plutôt que Newton d'ordre deux utilisée dans la méthode de Newton-Raphson.
3. Nous avons introduit une nouvelle idée qui consiste à convertir les éléments analogiques du circuit en leurs sources de tension dépendantes ou en sources de courant dépendantes et en appliquant les lois KCL et KVL pour obtenir l'équation du circuit.
4. Nous avons créé un nouveau logiciel appelé PyAMS (Python pour les signaux analogiques et mixtes) basé sur la méthode proposée pour le système Windows. Ce logiciel a les caractéristiques suivantes:
    - Modélisation de tout élément analogique sous forme de source dépendante.
    - Créer des circuits par la méthode du texte ou du schéma (CAD).
    - Analyse des circuits.

Les résultats obtenus en utilisant le nouveau logiciel basé sur la méthode proposée ont montré l'efficacité de notre approche. Par rapport à d'autres méthodes, notre méthode est plus efficace. Notre nouveau logiciel est plus rapide que le simulateur SPICE dans l'analyse des circuits, et offre la modélisation des éléments contraire à SPICE.

Mots-clés: Équations non linéaires, Newton Raphson, Homotopie, Newton d'ordre trois, Newton-Karylov, Circuits analogiques, SPICE, PyAMS.

Scientific Production.

Publications:

1. F. Dhiabi, M. Boumehraz, *"Accelerate the Solving of Nonlinear Equations Using Homotopy Method: Application on Finding the Operating Point of complex circuits"*, Turkish Journal of Electrical Engineering & Computer Sciences, Accept. Jul. 2016.

2. F. Dhiabi, M. Boumehraz, *"Application of Third-Order Convergence to Solve System Nonlinear of Equations for Analog Circuits"*, Turkish Journal of Electrical Engineering & Computer Sciences, Under Review.

3. F. Dhiabi, M. Boumehraz, *"PyAMS Python for Analog and Mixed Signals: New Software for Modeling and Analysis of Analog Circuits Based on the Continuous Method."* Elsevier: Advances in Engineering Software, Under Review.

International Conferences:

1. F. Dhiabi, M. Boumehraz, *"Solving Nonlinear Equations by Applying Homotopy Method to Find Operating Point in Analog Circuit",* Conference on Electrical Engineering, ICEEB'14, 7-8 Dec. 2014.

Contents

Chapter II: Accelerate the Solving of Nonlinear Equations using the Homotopy Method.

Chapter III: Application of Third-Order Convergence to Find Operating Point and Analyses Systems of Analog Circuits.

Chapter IV: New Software for Modeling and Analysis of Analog Circuits Based on the Continuous Method.

List of Figures

List of Tables

List of Algorithms

List of programs

List of abbreviations

SPICE	Simulation Program with Integrated Circuit Emphasis
Gmin	Minimum Conductance
Tnom	Nominal temperature.
ITL	Condition of limited iteration
MNA	Modified Nodel Analyses
GMRES	Generalized Minimal Residual
CG	Conjugate gradient method
NG	Newton-GMRES
NW	M. Waseem
MA	M. Darvishi and A. Barati method
NM	New method of family of Newton order three
NR	Newton Raphson.
CAD	Computer-aided design.
DC	Operating point analysis.
TR	Transient analysis.
AC	Frequency analysis.
OP	Operation point
KCL	Kirchhoff current law
KVL	Kirchhoff voltage law
NH	Newton Homotopy
VG	Variable Gain
VGNH	Variable Gain Newton Homotopy
FPNH	Fixed Point Newton Homotopy
PyAMS	Python Language for Analog and Mixed Signal.
GUI	Graphical user interface
LU	LU decomposition (a lower triangular matrix *L* and an upper triangular matrix *U)*

General Introduction

a. Introduction

With the rapid progress of the semiconductor technology, the integrate circuit designs become more and more computer-dependent. The practical circuit parameters can't be determined just by the hand calculation results since the minimizing feature size leads to the complex parasitic effects and the considerable large number of parameters of the device models. However, the increasing circuit scale and complexity also make the CAD (Computer-Aided Design) in the advanced techniques which it is play a crucial role in current IC designs. Nowadays SPICE [1-3] is used for the circuit simulator with most designers familiar.

Moreover, after more than three decades of running on all sorts of computers around the world, after solving critical problems in extremely complex circuits, SPICE can be regarded as the defect standard in circuit simulation. It is extensively used in current commercial EDA software, such as Virtuoso Spectre, HSPICE, PSpice, LTSpice, and so on [33-36]. The SPICE core, they all utilize the Newton-Raphson (NR) iterative algorithm to find the DC operating points of nonlinear circuits [6].

The method was used for solving systems of equations in complex analog circuits, i.e. finding the operating point in complex analog circuits has five main problems in circuit simulation by SPICE software [1,6]. The problems are slow convergence, the divergence, the cycle phenomena or oscillating, numerical overflew and nearly singular Jacobian. There are some techniques were applied in SPICE proposed to overcome the problem of convergence as [9-11]:

- Limiting method was used to solve problem nearly singular Jacobian.
- Source stepping method and *Gmin* method were used for good starting of the initial guess.
- Voltage limiting and diode damping method also were used to solve overflow problem.

These techniques: Limiting method, Source stepping method, *Gmin* method and Diode damping [9-11] are algorithms integrate with NR algorithm. But in this way, it is long

algorithm and generate slow of convergence for fined operating points. The *Continuous* or *Homotopy* method (see section b) is one algorithm replace of all those techniques, which is sampled for integrate in SPICE [16] fasted for fined operating points [12-20].

b. Motivations and objectives

In the recent years, new method is named method of *Continued* or *Homotopy* [12-20] can be suggested to improve the convergence problem or to accelerate the finding of the operating point, the *Continuous* method is based on the three following steps:

1. Transforming the nonlinear equation to the continuous or homotopy equation in NH (Newton Homotopy) form or FPNH (Fixed Point Newton Homotopy) [17,18] or VGNH (Variable Gain Newton Homotopy) [19,20].
2. The prediction process: estimating the direction of the solution by using one of those methods: Euler, Secant [27], Pseudo-Arclength [15] and ODE-Solver [28].
3. The correction process: is the iterative method based on Newton-Raphson [15,27].

The NH equation can be converted to FPNH or VGNH. The VGNH performance is better than the FPNH and NH [19,20], due to its simplicity of implementation in the SPICE simulator and its minimized iteration number to get convergence using the algorithm of the continuous method. VGNH it's good in the circuit with BJT elements [19] or MOSFET elements [20].

In the process of prediction by Euler or Secant [27] is very fast and stable for finding estimate the direction also it is better than of pseudo-arclength [15] and ODE-Solver [28-29], because those two methods using for bifurcation of solution by turning point method [28,29].

The NR method was used in the correction process is a second-order method [1].

The motivate by the previous works, the first objective is to boost the solving of non linear systems and to overcome the drawbacks of the previous works; this will be done by introducing new method based on improving the continuous method. The second is to build new software which can deal with the modeling and the simulation of integrate and circuits; this software was based on the proposed method.

c. Contributions

In the work of this thesis, we have proposed a new approach to accelerate the solving of nonlinear equations can be used to improve the *continuous method*. The main contributions of this work are:

1. Adding a new stage to the continuous method for control the step length of the continuous parameter, it is depends on the correction process; the objective is to obtain good direction (approximation) of solving.
2. We have used new method (NM) which is applied in the *Newton order three* in the correction process rather than the *Newton order two* used in the Newton-Raphson method. Our method was compared with similar iteration methods like the NR method, the MW method [63], the MA method [61] and the NG method [76].
3. We have introduced a new idea which was converted the analog elements of the circuit of their dependent voltage and current sources, and applying the Kirchhoff Current Law (KCL) and the Kirchhoff Voltage Law (KVL) to obtain the equation of the circuit.
4. We have created new software called PyAMS (**Py**thon for **A**nalog and **M**ixed **S**ignals) based on the proposed method in the windows system, which it is defined by the following software:
    - Modeling the analog element in the form of dependent source.
    - Create circuits by either the textual or the schema method (CAD).
    - Analysis of circuits.

İn order to demonstrate the effectiveness of the proposed method (the *continuous method*), a comparison was used to done with two other methods, there many types of integrated circuits are used in problems. Circuit analysis of many universal circuits has been carried out to verify the correct functioning of the circuit based on the proposed method. The obtain results were used in the new software based on the proposed method have shown the effectiveness of our approach where it is about three times faster than the VGNH method implemented in the SPICE software.

Compared to the SPICE software, our new software PyAMS has the following advantages:

1. It is faster than the SPICE simulator in analyzing circuits.
2. It offers to simplify the directly modeling of elements contrary to SPICE.
3. Create circuits by either the textual or the schema method (CAD).

4. We can create our own library of components.
5. We can modify the model and the design of the component.

e. Overview of thesis

This thesis is divided in four chapters:

The first chapter discusses the application of the NR algorithm for solving systems of nonlinear equations by the SPICE software for converting nonlinear elements (Diode, MOSFET, BJT) to linear elements and nonlinear circuits to linear circuits depending on the Newton iteration. The aim of converting nonlinear circuits to linear circuits is to simplify the solving by the linear method and find the operating point of the circuit. Two examples are presented to illustrate the use of the NR by SPICE to convert nonlinear elements (like diodes) in a circuit to linear elements (resistor and current source) depending on the Newton iteration. This chapter discusses to the structure of NR algorithm in three different algorithms of analysis:

- Quiescent domain (DC operating point analysis or DC simulation).
- Time domain (transient analysis or TR simulation).
- Frequency domain (harmonic analysis or AC simulation).

After that, the problem of convergence by NR on analyzing circuit (The problems are the slow convergence, the divergence, the cyclic phenomena or oscillating, numerical overflow and the nearly singular Jacobian) and the technical methods applied by SPICE to improve the convergence problem to accelerate the solving.

In the second chapter, we propose a method to accelerate the solving of nonlinear equations and to find the operating point in various integrated circuits by constructing the global homotopy equation of the analog circuit based on converting the elements in the circuit to their equivalent dependent sources. The proposed method was based on four steps. The first step is the use of the homotopy method to get a continuous function. The second step consists of finding the best direction of the solution by applying the production process; the third is the correct process. The fourth and the new added step is the control of the step size to accelerate the solution search. A comparison between the proposed method and other methods for five types of practical circuits widely used in analog LSI's are considered in this chapter. These circuits consist of Ebers-Moll BJT, which has a big problem of convergence in the SPICE

software, they are often used as a test circuit:

- The hybrid voltage reference circuit (HVRef).
- Six-stage limiting amplifier (6sLA).
- Operational amplifier (μA741).
- Wideband amplifier (RCA3040).
- Basic two-stage operational amplifier (2sOA).

In the third chapter, we present a new acceleration method for solving systems of nonlinear equations; it consists of the use of a Third-Order Newton family with cubic convergence. This method was applied to the correction process of the continuous method. This method too was employed in our new software PyAMS for analysis analog circuits. To compare the proposed method with some other methods: Globally convergent algorithm of NR**,** Third-order MW, third-order MA and Newton-Krylov, a general error of convergence analysis in analog circuits and numerical illustrations are given.

In the final chapter, new software called PyAMS (Python for Analog and Mixed Signals) for creating analog elements and designing circuits with analysis based on the Python language. The aim work is to simplify the modeling and the programming of analog elements and building circuits with analysis. The software depends on converting the analog elements (e.g. active elements as Diodes, BJTs, MOSFET, logical Gates… or passive elements as Resistors, Capacitors, Inductance…) to dependent sources which will be applied in the circuit, then; the circuit will be converted to the general equation which should be solved to be applied in the circuit analysis.

A general conclusion is given, which concludes the work and gives some future perspectives based upon the work in this thesis.

Chapter I

The Application of the Newton Raphson method by SPICE for Circuit Analysis.

I.1. Introduction

All technologies used the electrical circuits with integrated components based on semiconductor devices as Diode, BJT, and MOSFET…etc; these devices are nonlinear elements. The build of circuit can be tested by using simulation methods like SPICE (Simulation Program with Integrated Circuit Emphasis) to check circuit performance. The SPICE simulator was used the Newton Raphson (NR) for solving the nonlinear equations of analog circuits to obtain voltage values and currents in the circuit and simplify circuit analysis. The NR is the best performance method to find the operating point because it is simplified the nonlinear transform elements (Diode, BJT, MOSFET), directly to linear elements or nonlinear circuit to linear circuit, the subject of this convert is to simplify of circuit equation solve by linear method (LU decomposition), but there are problems in convergence. The problems are related to slow convergence, the divergence, the cycle phenomena or oscillating, numerical overflew and nearly singular Jacobian, so that, the simulator SPICE solve the problems by using a some techniques: Limiting method, Model Damping or G_{min} stepping, Source Stepping, Voltage Limiting, Diode damping and Continuation method.

The search of the operating point by NR was used for circuits' analysis in the three modes:

- Quiescent domain (DC operating point analysis or DC simulation).
- Time domain (transient analysis or TR simulation).
- Frequency domain (harmonic analysis or AC simulation).

In this chapter, the presentation of the NR application by SPICE was composed in the five principal parts: (i), the method of solving a system of nonlinear equations by NR algorithm. (ii), the system construct of equations for analog circuit. (iii), the application of NR by SPICE.

(iv), the position NR in algorithms of analyses. (v), the technical methods to eliminate the convergences problems.

I.2. History of SPICE

The full name of SPICE is “Simulator Program with Integrated Circuit Emphasis” [2,6]. It was at the Electronics Research Laboratory of the University of California, Berkeley by Larry Nagel with direction from his research advisor, Prof. Donald Pederson [2,32]. Today, SPICE is used worldwide as an essential computer-aid for circuit design, which can be simulated of prior circuits to their fabrication and predict detailed performance. For more than three decades of developing SPICE simulator, there are several widely used of versions SPICE: SPICE2, SPICE3, HSPICE, PSPICE and XSPICE [7].

The first SPICE version is SPICE1 which was largely a derivative of the CANCER program [7]. During the early 1970s, Larry Nagel had worked on under Prof. Ronald Rohrer. CANCER was an acronym for "Computer Analysis of Nonlinear Circuits, Excluding Radiation," a hint to Berkeley's liberalism of 1960s: at these times many circuit simulators were developed under the United States Department of Defense contracts that required the capability to evaluate the radiation hardness of a circuit. When Nagel's original advisor, Prof. Rohrer, left Berkeley, Prof. Pederson became his advisor. Pederson insisted that CANCER, a proprietary program, be rewritten enough that restrictions could be removed and the program could be put in the public domain.

The real popularity of SPICE started with SPICE2 in 1975 [7,6]. SPICE2, also coded in FORTRAN, it is improve the program with more circuit elements. The variable time step of transient analysis using either trapezoidal or Gear integration, the equation formulation can be modified by nodal analysis (avoiding the limitations of nodal analysis). An innovative FORTRAN-based memory allocation system was developed by another graduate student, Ellis Cohen [re]. The last FORTRAN version of SPICE was 2G.6 in 1983.

SPICE3 was developed by Thomas Quarles (with A. Richard Newton as advisor) in 1989. It was written in C with using same netlist syntax [7]. In the superset of SPICE2, including all of the analysis types and device models of SPICE2 as well as new features such as improved device models, voltage- and current-controlled switches, pole-zero analysis, and a graphical postprocessor for viewing simulation. Today, the latest version is SPICE3f5.

We have to say that the some features which have not been written into SPICE3, including the ability to run multiple circuit netlists from a single file, polynomial capacitors and inductors due to the hundreds of SPICE2 macromodels available today, because of the large

installed base of SPICE2-compatible simulators, the lack of backward compatibility is probably the overriding reason, SPICE3 has not replaced SPICE2 as the industry-standard circuit simulator.

However, although it may not have all the features of SPICE2, SPICE3 offers several technical advantages to SPICE2. SPICE3 was writing in modular C code which is easier to change than the FORTRAN of SPICE2. SPICE3 demonstrates superior convergence characteristics to the SPICE2 algorithms. In rewriting the SPICE3 device models, several SPICE2 errors discover and corrected. As Berkeley continues adding enhancements and SPICE2 compatibility to the program, the day will come when SPICE3 will replace SPICE2 [34].

As an early open source program, SPICE was widely distributed and used. It inspired and served as a basis for many other circuit simulation programs in academia, in industry, and in commercial products. Its ubiquity became such that "SPICE circuit" remains synonymous with circuit simulation. SPICE source code was from the beginning distributed by UC Berkeley for a nominal charge (to cover the cost of magnetic tape). The license includes an acknowledgement clause and distribution restrictions for countries not considered friendly to the USA. Nevertheless, Berkeley SPICE continues to influence both commercial and academic offshoots of the program. Early commercial versions of SPICE include HSPICE (now owned by Synopsys) and PSPICE (now owned by Cadence Design Systems) [35]. The academic spinoffs of SPICE include XSPICE [33], developed at Georgia Tech, which added mixed analog/digital "code models" for behavioral simulation, and Cider (previously CODECS, from UC Berkeley/Oregon State Univ.) which added semiconductor device simulation [2].

I.3. Newton-Raphson Method

The NR method is widely used for solving a system of nonlinear equations:

$$F(x) = 0 \tag{1.1}$$

where $F: \mathbb{R}^n \longrightarrow \mathbb{R}^n$ and $x \in \mathbb{R}^n$. For $F=(f_1,f_2,\ldots,f_n)$ it vector dependent by variable vector $x=(x_1,x_2,..,x_n)$. The NR method for (1.1) is a second-order method [4,5], based on this iterative expression:

$$x^{k+1} = x^k - \frac{F(x^k)}{J(x^k)} \tag{1.2}$$

where *k* is the iteration index, x_k is the current approximate solution and $J(x^k)$ is the Jacobian or derivation of $F(x^k)$.

$$J(x) = \frac{\partial F(x)}{\partial x} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix} \tag{1.3}$$

In Fig. I.1 sample demonstration to find the solution of nonlinear equation *F(x)* by NR, The function *F* is shown in on arc and the tangent line is in flash. The iterative process starts with an first guess of x^0, then the x^1 is obtained according to Eq. (1.1). After the three steps of iteration, the solution x^3 approximates to the real solution x^*.

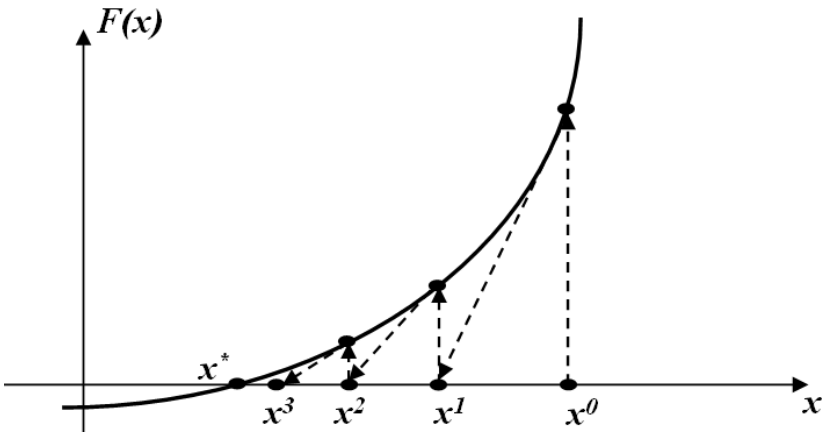


Fig. I.1. The successful case for Newton-Raphson method.

An algorithm of NR to solve the system nonlinear equation is given below:

Algorithm I.1: Algorithm NR

1. *Choose x_0 initial approximation and ε_F error of convergence.*
2. *Start with k= 0*
3. *Calculate x^{k+1}:*
   - $x^{k+1} = x^k - F(x^k)/J(x^k)$
   - $x^k = x^{k+1}$
4. *If there is no convergence ($\|F(x^k)\| > \varepsilon_F$) go to stage 3.*
5. *Return the final solution.*

I.4. The circuit equation

The system of equations of analog circuit is presented in the form (1.1), for variant vector *x* presents the voltage in the nods and the current across voltage sources, and for *F* presents the Kirchhoff current law and the Kirchhoff voltage law (KCL and KVL) of the circuit [1,2]. The system of equations of analog circuit is presented in this form:

$$F(v_1, v_2, \ldots\ldots, v_n, i_1, i_2, \ldots\ldots\ldots, i_m) = [f_1, f_2, f_3, \ldots\ldots, f_{n+m}] \tag{1.4}$$

Where $F: \mathbb{R}^{n+m} \longrightarrow \mathbb{R}^{n+m}$ For:

- *n*: Presents the number of the nodes in circuit.
- *m*: Presents the number of voltage sources in circuit.
- $v_1, v_2, \ldots\ldots, v_n$: Values of voltages in the *n* nodes.
- $i_1, i_2, \ldots\ldots\ldots, i_m$: Values of currents across the *m* voltage sources.
- $f_1, f_2, f_3, \ldots\ldots, f_n$: *n* equations extracted by application of KVL law in the *n* nodes.
- $f_{n+1}, f_{n+2}, f_{n+3}, \ldots\ldots, f_{n+m}$: *m* equations extracted by application of KCL law in the *m* voltage sources.

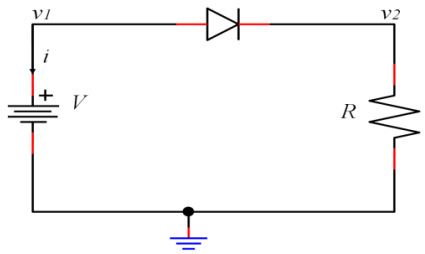


Fig. I.2. Example of nonlinear circuit.

For example, the circuit in Fig I.2 is nonlinear because it has a diode with exponential function, to create a system of equations with variants: v_1, v_2 and current *i* across the source *V*, It must apply KCL law in the two nodes which presented by f_1 and f_2 formulas shown in (1.5) and apply KVL in voltage source which presented by f_3.

$$F(v_1, v_2, i) = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} = \begin{cases} I_s\left(exp\left(\frac{v_1 - v_2}{V_T}\right) - 1\right) + i \\ \frac{v_2}{R} - I_s\left(exp\left(\frac{v_1 - v_2}{V_T}\right) - 1\right) \\ v_2 - V \end{cases} \tag{1.5}$$

Where *Vt* and *Is* are the thermal voltage and the saturation current of the diode, respectively, and *R* and *V are* constant values of resistors and voltage in a circuit. By applying the NR algorithm in the equation (1.5) we get the values of v_1 and v_2 voltages in the nodes with current *i* across the source *V1*.

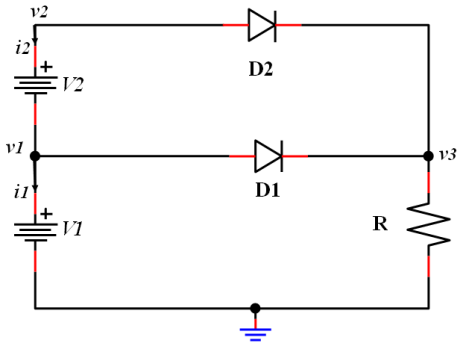


Fig. I.3. Example of nonlinear circuit with two diodes.

Second example, the circuit in Fig I.3 is nonlinear with three nodes (*n=3*) and two voltage sources (*m=2*), to create a system of equations depending on voltages in the nodes (v_1,v_2,v_3) and currents (i_1, i_2) across the sources *V1* and *V2*, it must use KCL law in the three nodes who presented by formulas f_1, f_2 and f_3 shown in equation (1.6) and KVL law in voltage sources *V1* and *V2* who presented by f_4 and f_5.

$$F(v_1,v_2,v_3,i_1,i_2)=\begin{bmatrix}f_1\\f_2\\f_3\\f_4\\f_5\end{bmatrix}=\begin{cases} I_s\left(exp\left(\frac{v_1-v_3}{V_T}\right)-1\right)+i_1-i_2 \\ I_s\left(exp\left(\frac{v_2-v_3}{V_T}\right)-1\right)+i_2 \\ \frac{v_2}{R}-I_s\left(exp\left(\frac{v_1-v_3}{V_T}\right)-1\right)-I_s\left(exp\left(\frac{v_2-v_3}{V_T}\right)-1\right) \\ v_1-V_1 \\ v_2-v_1-V_2 \end{cases} \quad (1.6)$$

I.5. The application of NR by SPICE

The simulator SPICE used NR to convert nonlinear elements (diode, BJT, MOSFET…etc) to linear elements, the subject of this concert is to simplify solving the equation of circuit by linear method (LU decomposition) [2,37].

I.5.1. Diode

As a two-terminal device with only one distinct operating region, the diode was a natural starting point for non-linear simulation. The current model does not include parasitic capacitance (not complete model). The $i - v$ relation describing a diode is the following:

$$I(v) = I_s\left(exp\left(\frac{v}{V_t}\right) - 1\right) \tag{1.7}$$

For $v = v_p - v_n$, where I_S is the reverse saturation current and V_t is the thermal voltage. From this, we derive the small-signal conductance:

$$g_d = \frac{\partial i}{\partial v} = \left(\frac{I_s}{V_t}\right)\ exp\left(\frac{v}{V_t}\right) \tag{1.8}$$

The error committed in the approximation is Is/Vt, which for common diodes is around 10^{-12}. This is negligible for ordinary applications and allows us to avoid computing the export function twice [37].

In simulation, we use the NR method to solve circuits with nonlinear elements. The method works by linearity the constitutive equations, solving for a new operating point, and iterating until convergence. The linear equation around operating point v^k is:

$$i(v^{k+1}) = i(v^k) + (v^{k+1} - v^k)\frac{\partial i}{\partial v}(v^k) \tag{1.9}$$

By rearranging this equation, we find the *i-v* relation:

$$i(v^{k+1}) = i(v^k) - g_d^k\ v^k + g_d^k\ v^{k+1} \tag{1.10}$$

Hence the appropriate companion model for a diode is a conductance g_d^k in parallel with a current source $I_{eq}^k = i(v^k) - g_d^k\ v^k$, not simply $i(v^k)$, as one might suspect. The result is shown in Fig. I.4.

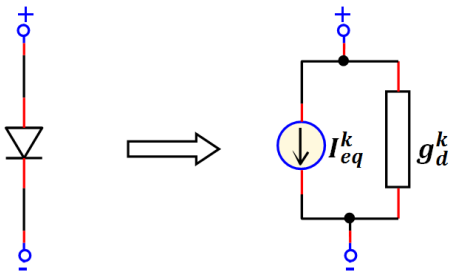


Fig. I.4. The linear of diode model by application of NR.

I.5.2. MOSFET

The next simplest non-linear devices are n-channel of MOSFETs, as no current flow into the gate and the non-linear source-to-drain port i_{ds} depends on only two voltages, v_{GS} and v_{DS}. The model was complicated slightly, however, by the fact that MOSFETs has three distinct operating regions. For brevity, we derive only the results for n-channel MOSFETs [3]. Similar results for p-channel MOSFETs.

$$i_{ds} = \begin{cases} 0 & v_{ds} \le V_t \\ K_P \frac{W}{L}\left(\left(v_{gs} - V_t\right)v_{ds} - \frac{{v_{ds}}^2}{2}\right)(1 + \lambda\, v_{ds}) & V_t \le v_{gd} \\ K_P \frac{W}{L}\left(v_{gs} - V_t\right)^2(1 + \ \lambda\ v_{ds}) & V_t > v_{gd} \end{cases} \qquad (1.11)$$

We seek to define a bias current from drain to source that will cause the node voltages to evolve by NR iteration, as we did for the diode in the previous subsection. We now use the multivariable form of the iteration equation:

$$i_{ds}\left(v_{ds}^{k+1}, v_{gs}^{k+1}\right) = i_{ds}\left(v_{ds}^{k}, v_{gs}^{k}\right) + \left(v_{ds}^{k+1} - v_{ds}^{k}\right)\frac{\partial i_{ds}}{\partial v_{ds}}\left(v_{ds}^{k}\right) + \left(v_{gs}^{k+1} - v_{gs}^{k}\right)\frac{\partial i_{ds}}{\partial v_{gs}}\left(v_{gs}^{k}\right) \qquad (1.12)$$

By rearranging this equation, we find the *i*-*v* relation:

$$i_{ds}\left(v_{ds}^{k+1}, v_{gs}^{k+1}\right) = i_{ds}\left(v_{ds}^{k}, v_{gs}^{k}\right) - v_{ds}^{k} - v_{gs}^{k} + g_{ds}^{k} v_{ds}^{k+1} + g_{m}^{k} v_{gs}^{k+1} \qquad (1.13)$$

For

$$g_m^k = \frac{\partial i_{ds}}{\partial v_{gs}}\left(v_{gs}^{k}\right) = \begin{cases} 0 & v_{ds}^{k} \le V_t \\ K_P \frac{W}{L} v_{ds}^{k}\left(1 + \lambda\, v_{ds}^{k}\right) & V_t \le v_{gs}^{k} \\ 2\, K_P \frac{W}{L} v_{gs}^{k}\left(1 + \ \lambda\ v_{ds}^{k}\right) & V_t > v_{gs}^{k} \end{cases} \qquad (1.14)$$

And

$$g_{ds}^k = \frac{\partial i_{ds}}{\partial v_{ds}}\left(v_{ds}^{k}\right) = \begin{cases} 0 & v_{ds} \le 0 \\ K_P \frac{W}{L}\left(\left(v_{gs} - V_t\right)v_{ds} - \frac{{v_{ds}}^2}{2}\right)(1 + \lambda\, v_{ds}) & V_t \le v_{gd} \\ K_P \frac{W}{L}\left(v_{gs} - V_t\right)^2(1 + \ \lambda\ v_{ds}) & V_t > v_{gd} \end{cases} \qquad (1.15)$$

Hence the appropriate companion model for a n-channel MOSFET is a conductance g_{ds}^k in parallel with a current source $I_{eq}^k = i_{ds}\left(v_{ds}^k, v_{gs}^k\right) - v_{ds}^k - v_{gs}^k$ and current source controlled by v_{gs}^{k+1} multiplied by g_m^k. The result is shown in Fig. I.5.

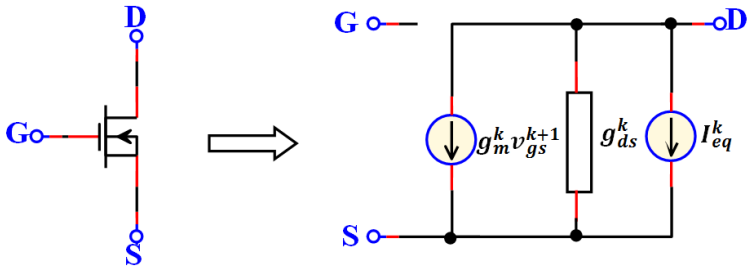


Fig. I.5. The linear of NMOS model by application of NR.

I.5.3. Bipolar Junction Transistor (BJT)

Companion models for BJTs are the most complex circuits presented here because current flows between all nodes. The model is not quite as unwieldy as it looks, however, because there is only one operating region. We start with a simplified, mid-band Gummel-Poon NPN model, as shown in Fig. I.6. The base currents are defined as follows:

$$i_{bc}(v_{bc}) = \frac{I_{sc}}{\beta_R}\left(exp\left(\frac{v_{bc}}{V_t}\right) - 1\right) \tag{1.16}$$

$$i_{be}(v_{be}) = \frac{I_{se}}{\beta_F}\left(exp\left(\frac{v_{be}}{V_t}\right) - 1\right) \tag{1.17}$$

$$i_{ce}(v_{be}, v_{bc}) = I_{se}\left(exp\left(\frac{v_{be}}{V_t}\right) - 1\right) - I_{sc}\left(exp\left(\frac{v_{bc}}{V_t}\right) - 1\right) \tag{1.18}$$

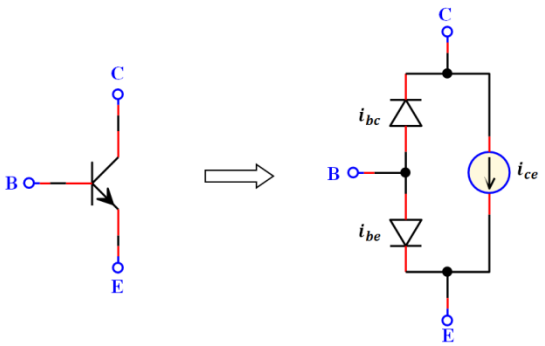


Fig. I.6. Mid-Band Gummel-Poon BJT Model [2].

Then, defining the small-signal conductance's as in Fig. I.7, we have:

$$g_{bc} = \frac{\partial i_{bc}}{\partial v_{bc}} = \left(\frac{I_{sc}}{\beta_R V_t}\right)\ exp\left(\frac{v_{bc}}{V_t}\right) \tag{1.19}$$

$$g_{be} = \frac{\partial i_{be}}{\partial v_{bc}} = \left(\frac{I_{se}}{\beta_C V_t}\right)\ exp\left(\frac{v_{be}}{V_t}\right) \tag{1.20}$$

$$g_{ce,bc} = \frac{\partial i_{ce}}{\partial v_{bc}} = \left(\frac{I_{sc}}{V_t}\right)\ exp\left(\frac{v_{bc}}{V_t}\right) \tag{1.21}$$

$$g_{ce,be} = \frac{\partial i_{ce}}{\partial v_{be}} = \left(\frac{I_{se}}{V_t}\right)\ exp\left(\frac{v_{be}}{V_t}\right) \tag{1.22}$$

In simulation, we use the NR method to solve circuits with nonlinear elements. The method works by linearity the constitutive equations, solving for a new operating point, and iterating until convergence. The linear equation around operating point v^k is:

$$i(v^{k+1}) = i(v^k) + (v^{k+1} - v^k)\frac{\partial i}{\partial v}(v^k) \tag{1.23}$$

By rearranging this equation, we find the *i-v* relation:

$$i_{bc}\left(v_{bc}^{k+1}\right) = i_{bc}\left(v_{bc}^{k}\right) - g_{bc}^{k}\ v_{bc}^{k} + g_{bc}^{k}\ v_{bc}^{k+1} \tag{1.24}$$

$$i_{be}\left(v_{be}^{k+1}\right) = i_{be}\left(v_{be}^{k}\right) - g_{be}^{k}\ v_{be}^{k} + g_{be}^{k}\ v_{be}^{k+1} \tag{1.25}$$

$$i_{c}\left(v_{be}^{k+1}, v_{bc}^{k}\right) = i_{c}\left(v_{be}^{k}, v_{bc}^{k}\right) - \beta_F g_{be}^{k}\ v_{be}^{k} + \beta_F g_{be}^{k}\ v_{be}^{k+1} \tag{1.26}$$

$$i_{c}\left(v_{be}^{k}, v_{bc}^{k+1}\right) = i_{c}\left(v_{be}^{k}, v_{bc}^{k}\right) + \beta_R g_{bc}^{k}\ v_{bc}^{k} - \beta_R g_{bc}^{k}\ v_{bc}^{k+1} \tag{1.27}$$

We obtained current sources and resistors (Fig. I.7):

$$I_{eq,c}^{k} = 2\ i_{c}\left(v_{be}^{k}, v_{bc}^{k}\right) - \beta_F g_{be}^{k}\ v_{be}^{k} + \beta_R g_{bc}^{k}\ v_{bc}^{k} \tag{1.28}$$

$$I_{eq,bc}^{k} = i_{bc}\left(v_{bc}^{k}\right) - g_{bc}^{k}\ v_{bc}^{k} \tag{1.29}$$

$$I_{eq,be}^{k} = i_{be}\left(v_{be}^{k}\right) - g_{be}^{k}\ v_{be}^{k} \tag{1.30}$$

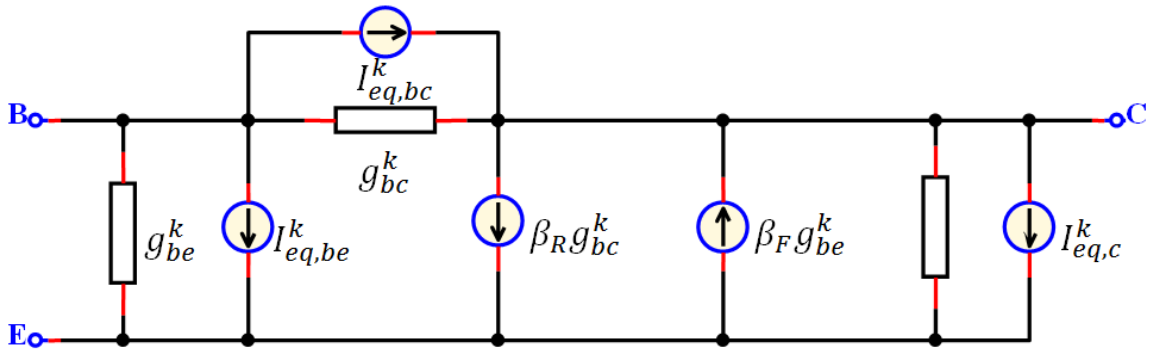


Fig. I.7. Linear companion model for NPN of BJT.

I.5.4 Element of storage of energie

Transient simulation of storage energy element requires companion models to be formed. For the element, it has derivative function by time *t* presented in this form:

$$\frac{\partial v}{\partial t} = f(v) \tag{1.31}$$

It used a numerical integration method to convert the nonlinear differential equation to nonlinear algebraic equation to simplify solving by NR in index of iteration "*k*". There are three simple numerical integration methods; it is used in the spice simulator [4]:

- Forward Euler:

$$f(v_{t-\Delta t}) = (v_t - v_{t-\Delta t})/\Delta t \tag{1.32}$$

- Backward Euler:

$$f(v_t) = (v_t - v_{t-\Delta t})/\Delta t \tag{1.33}$$

- Trapezoidal:

$$f(v_t) = -\frac{\Delta t}{2} f(v_{t-\Delta t}) + 2(v_t - v_{t-\Delta t})/\Delta t \tag{1.34}$$

Example of Capacitor, we start with the differential equation relating current to voltage in a capacitor:

$$I = C\ \frac{\partial v}{\partial t} \tag{1.35}$$

By Backward Euler (1.33):

$$I^{k+1} = C\left(v_t^{k+1} - v_{t-\Delta t}\right)/\Delta t \tag{1.36}$$

For *k* is an index of iteration in NR.

By equation (1.36), you can replace capacitor model by current source I_c and resistor g_c depends on step time Δt (Fig. I.8):

$$g_c = \frac{C}{\Delta t} \tag{1.37}$$

And

$$I_c = -\frac{C}{\Delta t}\, v_{t-\Delta t} \tag{1.38}$$

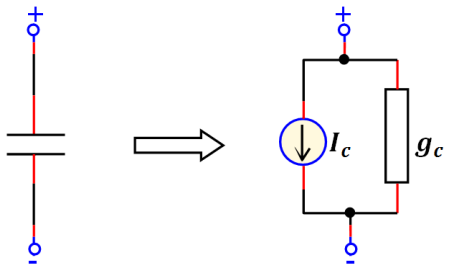


Fig. I.8. Linear companion model for capacitor.

I.5.5 Example of circuit

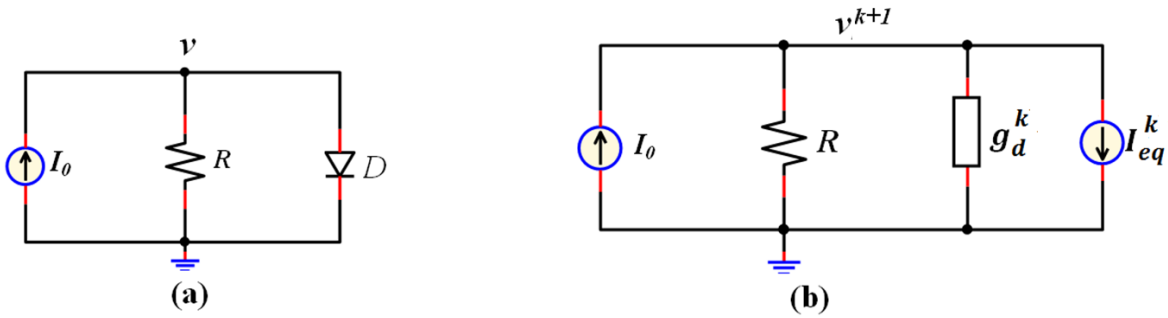


Fig. I.9. Example of circuit : (a)-Nonlinear circuit. (b)-The linear circuit of (a) by NR.

For a simple example, the circuit in Fig I.9.a has a diode which is a nonlinear element, and a single unknown value: the voltage "*v*" at the node. By using NR to replace diode by an equivalent resistor g_d^k and a current source I_{eq}^k (those equivalents based on voltage *v* in index *k* of iteration) which is shown in Fig I.9. By this method we get the linear circuit with new value in the node in circuit, v^{k+1} is related to *k+1* iterate.

The result v^{k+1}, obtained by applying the Kirchhoff's current law (KCL) in the circuit of Fig I.9.b:

$$\left(g_d^k + \frac{1}{R}\right) v^{k+1} - I_0 + I_{eq}^k = 0 \tag{1.39}$$

By (2.39) we get the value of v^{k+1}:

$$v^{k+1} = \frac{I_0 - I_{eq}^k}{g_d^k + \frac{1}{R}} \tag{1.40}$$

The equation (1.40) it simplified the solving of nonlinear equation presented by KCL for circuit in Fig I.9.a:

$$F(v) = \frac{v}{R} + I_s(\exp\left(\frac{v}{V_T}\right) - 1) - I_o \tag{1.41}$$

To stop iteration of equation (1.40) when $\|F\,(v^k)\| < \varepsilon_F$ or $\|v^{k+1} - v^k\| < \varepsilon_v$, when ε is an error of stop of convergence by NR algorithm [3].

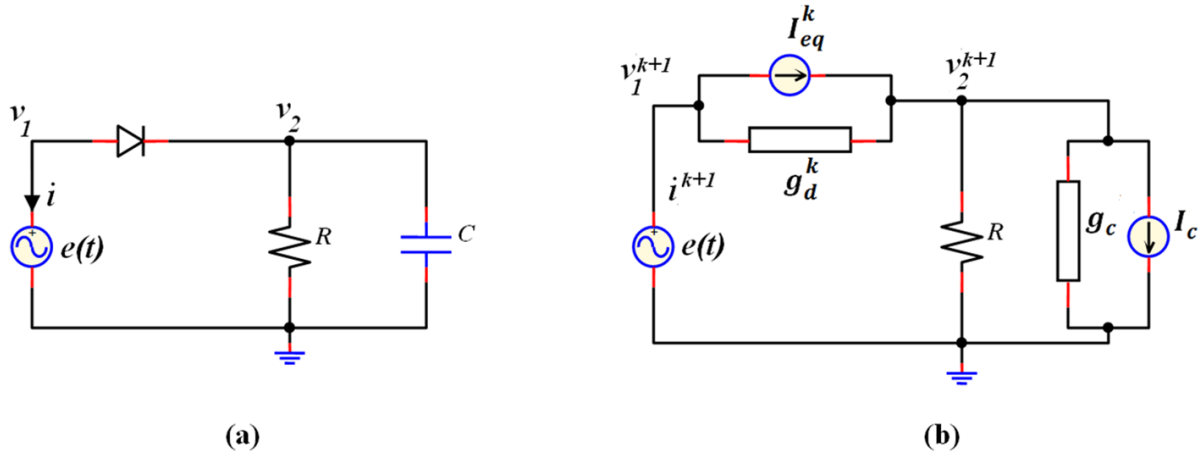


Fig. I.10. Example of circuit : (a)- circuit Nonlinear with dynamique element. (b)-The linear circuit of (a) by NR.

The circuit above includes dynamic elements (Inductance or Capacitor) and nonlinear elements represented in nonlinear system of differential equations, to simplify the solution, it must replace dynamic element to a voltage source or current and resistor depends on discrete time and Newton iteration index, this replacement was based on rewriting differential equation to Backward Euler or Trapezoidal or Gear methods [1,2].

The equivalents of capacitor are shown in Fig. I.10.a: are a current source I_c^k and a resistor g_c^k depends on setup time and Newton iteration index (Fig I.10.b), for the diode: is a resistor g_d^k and current I_{eq}^k .

The system of equation of the circuit is shown in Fig. II.10.b by using KCL law:

$$\begin{cases} g_d^k \left(v_1^{k+1}\text{-}v_2^{k+1}\right) + i_{k+1} - I_{eq}^k = 0 \\ g_d^k \left(v_2^{k+1} - v_1^{k+1}\right) + \left(\frac{1}{R} + g_c^k\right) v_2^{k+1} + I_{eq}^k - I_c^k \\ v_2^{k+1} - e(t) = 0 \end{cases} \quad (1.42)$$

We have obtained that the linear system of equations is presented in (1.43) , for solving we use LU decomposition method to get the value of voltage in the nodes v_1^{k+1} and v_2^{k+1} and the value of current i^{k+1} across the voltage source at the index of iteration *k+1* depends on g_d^k, I_{eq}^k , g_c^k and I_c^kin index *k*.

$$\begin{bmatrix} g_d^k & -g_d^k & 1 \\ -g_d^k & g_d^k + \frac{1}{R} + g_c^k & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} v_1^{k+1} \\ v_2^{k+1} \\ i^{k+1} \end{bmatrix} = \begin{bmatrix} I_{eq}^k \\ I_c^k - I_{eq}^k \\ e(t) \end{bmatrix} \quad (1.43)$$

The equation (1.43) can get directly by law of modified model analyses "MNA" [5], it simplified the solved of system nonlinear differential equations is presented by KCL law of the circuit in Fig. II.4.a:

$$F(v_1, v_{2,} i) = \begin{cases} I_s \left(exp\left(\frac{v_1 - v_2}{V_T}\right) - 1\right) + i \\ \frac{v_2}{R} + \frac{1}{C}\frac{\partial v_2}{\partial t} - I_s \left(exp\left(\frac{v_1 - v_2}{V_T}\right) - 1\right) \\ v_2 - e(t) \end{cases} \quad (1.44)$$

The iteration in (1.43) stopped by the Newton method when$\left\|v_1^{k+1} - v_1^k\right\| < \varepsilon_v$, $\left\|v_2^{k+1} - v_2^k\right\| < \varepsilon_v$ and $\left\|i^{k+1} - i^k\right\| < \varepsilon_i$ or $\left\|F(v_1^k, v_2^k, i^k)\right\| < \varepsilon_F$ for F is Function presented by equation (1.44) of the circuit in Fig I.4.a by KCL and KVL laws. And another method to stop iteration on converges (which simulator SPICE was based on [4-7]), it uses the convergence detection term for voltage and current.

For voltage:

$$\left\|v^{k+1} - v^k\right\| \leq reltol \times \max\left(1\left\|v^{k+1}\right\|, \left\|v^k\right\|\right) + vntol \quad (1.45)$$

And for current:

$$\left\|i^{k+1} - i^k\right\| \leq reltol \times \max\left(\left\|i^{k+1}\right\|, \left\|i^k\right\|\right) + abstol \quad (1.46)$$

For *vntol*, *abstol* and *reltol* are constant parameters: with the default values $abstol=10^{-12}$, $reltol= 10^{-3}$, and $vntol=10^{-6}$. The SPICE is not converted it stopped by condition of limited iteration "ITL"[1].

I.6. Modified Nodal Analysis

The system of linear equations of analog circuit was obtained directly by using modified nodal analysis method (MNA). The MNA often results in larger systems of equations were formulated by modified in KCL and KVL laws and applied to a circuit with only passive elements (resistors) and independent current and voltage sources results in a matrix equation of the form [1,2]:

$$\begin{bmatrix} G & F \\ B & R \end{bmatrix} \begin{bmatrix} v \\ i \end{bmatrix} = \begin{bmatrix} C \\ E \end{bmatrix} \tag{1.47}$$

where:

G: represent the admittance matrix of MNA.

B, F and R: represents various constitutive equations pertain to voltage sources.

C and E: are the vectors of the values of current and voltage sources, respectively.

For example the equation (1.43) of the circuit in Fig.I.10.b:

$$G = \begin{bmatrix} g_d^k & -g_d^k \\ -g_d^k & g_d^k + \frac{1}{R} + g_c^k \end{bmatrix} \tag{1.48}$$

$$F = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad B = [1\ 0] \quad \text{and} \quad R = [0] \tag{1.49}$$

$$\begin{bmatrix} v \\ i \end{bmatrix} = \begin{bmatrix} v_1^{k+1} \\ v_2^{k+1} \\ i^{k+1} \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} C \\ E \end{bmatrix} = \begin{bmatrix} I_{eq}^k \\ I_c^k - I_{eq}^k \\ e(t) \end{bmatrix} \tag{1.50}$$

Finally, a system of equations "MNA" (1.47) we can write it in form:

$$F(x) = Ax - b \tag{1.51}$$

For

$$A = \begin{bmatrix} G & F \\ B & R \end{bmatrix},\ x = \begin{bmatrix} v \\ i \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} C \\ E \end{bmatrix} \tag{1.52}$$

I.7. The structure of NR algorithm in algorithms of analyses circuits

The simulation of circuit by SPICE can be analyzed in three different domains [1,2]:

- Quiescent domain (DC operating point analysis or DC simulation)
- Time domain (transient analysis or TR simulation)
- Frequency domain (harmonic analysis or AC simulation)

In this part, we discuss the structure of the NR algorithm in algorithms of analyses by DC, TR and AC of circuit by SPICE.

The simulator SPICE modified algorithm NR to have good acceleration for solving nonlinear circuit, these modifications are presented in Fig. I.11: first, the best initial variable vector (v^0,i^0) or good approximation variable vector (initial Goss). Second, convert nonlinear element to linear element and after that construct modified nodal equation to simplify the solving by LU decomposition. Finally, we obtained new variable vector controlling by term of convergence (1.45) and (1.46), when converge; it stops iteration and the solution variable vector which presented the value of currents and voltages. This algorithm was applied for finding an operating point in the circuit (Fig I.11).

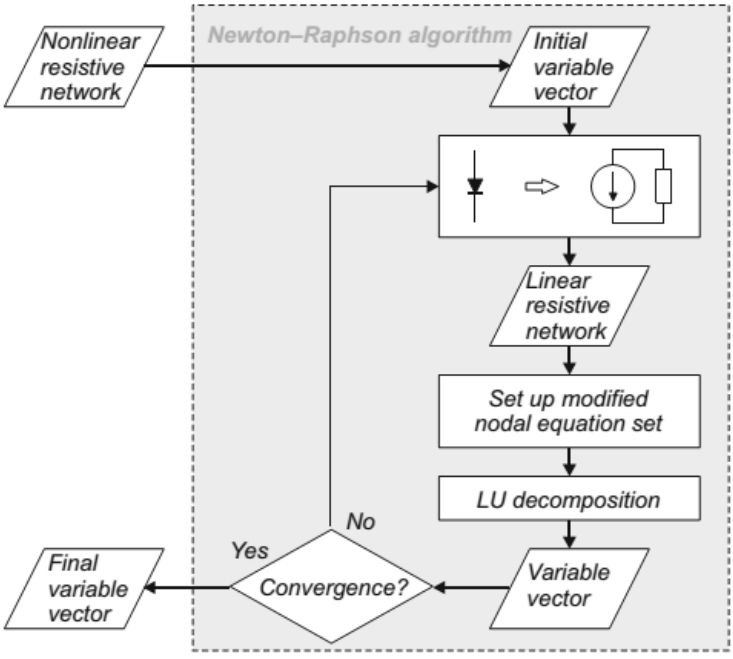


Fig. I.11. Algorithm of Newton-Raphson [1].

The operation point (OP) algorithm presented in Fig. I.12. The derivative equation in these algorithm is null [4], for that, the capacitor element is a current source with zero value and inductance is a voltage source with zero value, we have obtained that the nonlinear circuit without dynamic elements. After that, we use NR algorithm to obtain an operate point in

circuit. These modes of analyses were applied in TR and AC which to be explained after this section.

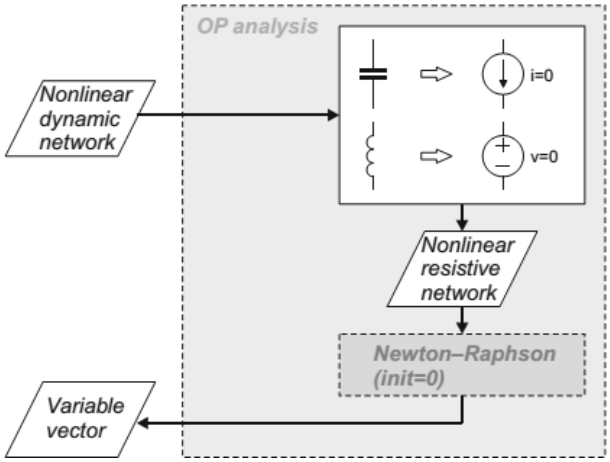


Fig. I.12. Algorithm of operation point [1] .

I.7.1. DC Simulation

The DC Simulator or analyses mode DC examine the work of circuit by steping one parameter in a circuit. The algorithm of DC analyses is presented in Fig I.13: first convert nonlinear dynamic circuit to nonlinear resistive circuit by replacing the capacitor to current source with zero value and inductance and by voltage source with zero value. Secondly, appling NR algorithm [2] to obtain variable vector. for new steping paramater the algorithm returned in nonlinear resistive circuit position [2].

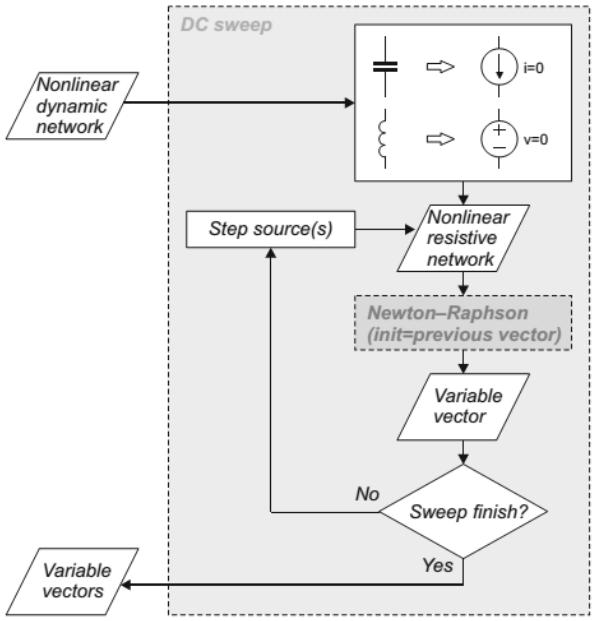


Fig. I.13. Algorithm of DC sweep analysis [1].

I.7.2. Transient Simulation

The TR Simulator or analyses mode TR can be used the nonlinear circuit simulation in the time domain, the algorithm of analyses by this mode is presented in Fig. II.14: first, finding operating point to have a good approximation or initial variable vector. Second, convert dynamic element to resistive and source elements. Finally, we have applied the NR algorithm for one setup time; if have not finished in interval time it returns to the second step of time.

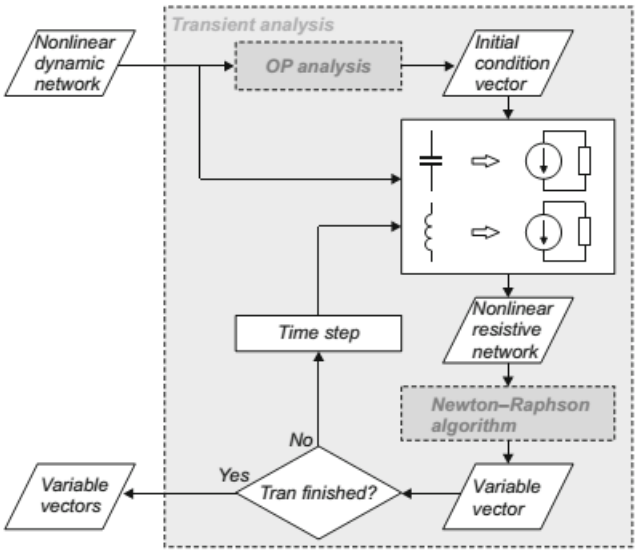


Fig. I.14. Time-domain transient analysis [1].

I.7.3. AC analyses

The AC Simulator was used to simulate circuit in the frequency domain, but the algorithm of analyses by this mode is presented in Fig II.15: First, it is get operating point in circuit and after that convert nonlinear element to resistive element to obtain a linear dynamic circuit. Second, the convert dynamic element to impedance element, finally the circuit was found a linear for that we use LU decomposition for solving; this is only for one step of frequency.

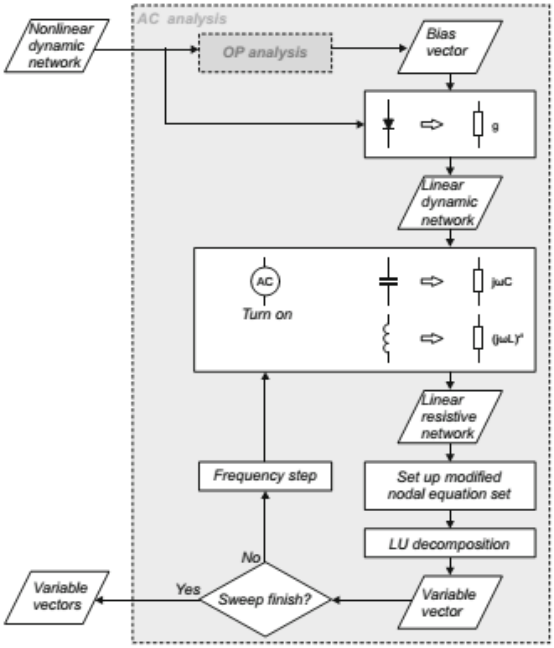


Fig. I.15. Small signal analysis [1].

I.8. Convergence Problem-Aiding Techniques

NR algorithm converges well if the initial guess for the operating point is good (Fig I.1). For a computer program, it is very difficult to make a good guess for the operating point and you can iteration by NR in cant to take one of these problems of convergence (Fig. I.16) [1]:

(a) The cycle phenomena or oscillating.

(b) Numerical overflew.

(c) Divergence.

(d) Nearly singular Jacobian.

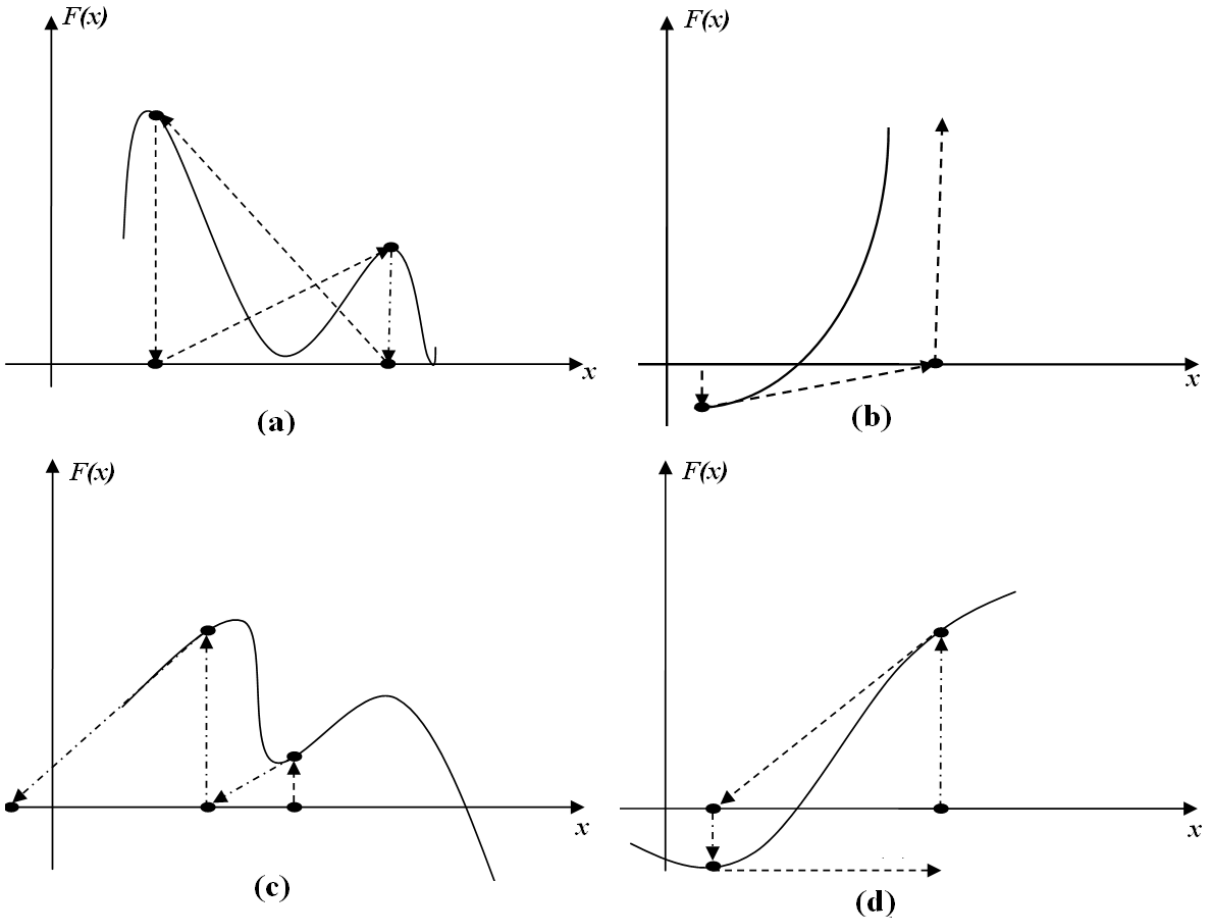


Fig. I.16. The problem of convergence by NR method.

These four problems can be set in DC analysis of circuits with nonlinear elements. For example, the diode has a very steep exponential characteristic function. It can be generated the problem of convergence, for example, the limitation problem or numerical overflow by diode and oscillation by tunnel diode.

For that, the convergence of the plain NR algorithm cannot be guaranteed for all practical circuits, a number of convergence-aiding methods have been developed. The model damping, source stepping, voltage limiting, diode damping, norm reduction, piecewise-linear analysis and dogleg method [9][10][11].

I.8.1. Limiting method

The modifications of the NR schemes are based on the limitation of the solution vector x^k for each iteration.

$$x^{k+1} = x^k + \alpha \ \ \Delta x^{k+1} \tag{2.53}$$

With

$$\Delta x^{k+1} = x^{k+1} - x^k \tag{2.54}$$

One possibility to choose a value for $\alpha \in]0,1]$ such that

$$\|F(x^{k+1})\| = \|F(x^k + \alpha \ \Delta x^{k+1})\| < \gamma \|F(x^k)\| \ \text{ for } \gamma < 1 \tag{2.55}$$

This is a heuristic and does not ensure global convergence, but it can help solving some of the discussed problems. Another possibility is to pick a value α^k which minimizes the *F* norm of the right hand side vector. This method performs a one-dimensional (line) search into Newton direction and guarantees global convergence [4].

$$x^{k+1} = x^k + \alpha^k \ \Delta x^{k+1} \tag{2.56}$$

With α^k which minimizes:

$$\|F(x^k + \alpha^k \ \Delta x^{k+1})\| \tag{2.57}$$

The one remaining problem about that line search method for convergence improvement the iteration into minimums local, where the Jacobean matrix is singular $J(x)$=0 (Fig. II.17.a). The damping NR method "pushes" the matrix into singularity as depicted in Fig. 1.17.b.

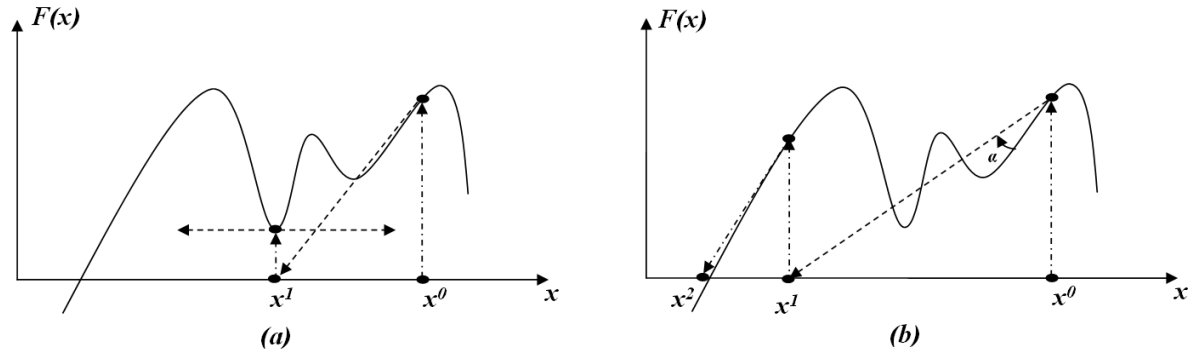


Fig. I.17. (a) Singular Jacobean problem (b) Using Limiting methods

I.8.2. Model Damping (G_{min} method)

In model damping, every nonlinear static source in the circuit is shunted with a conductance having a value:

$$G = 10^m G_{min} \tag{2.58}$$

where G_{min} is the minimum conductance in the circuit and *m* is an integer. Conductance *G* is also added between every node and ground (Fig. I.18) (A popular convergence aiding

technique which consists only of adding small conductance's between all nodes and ground was called the G_{min} stepping method [1][3]). The defaults values for G_{min} and m are 10^{-12} and 6, respectively. The user can alter both of the values. The iteration of the DC operating point was started with the m given, and after the iteration converges, m was reduced by one. This is repeated with until $m = 0$.

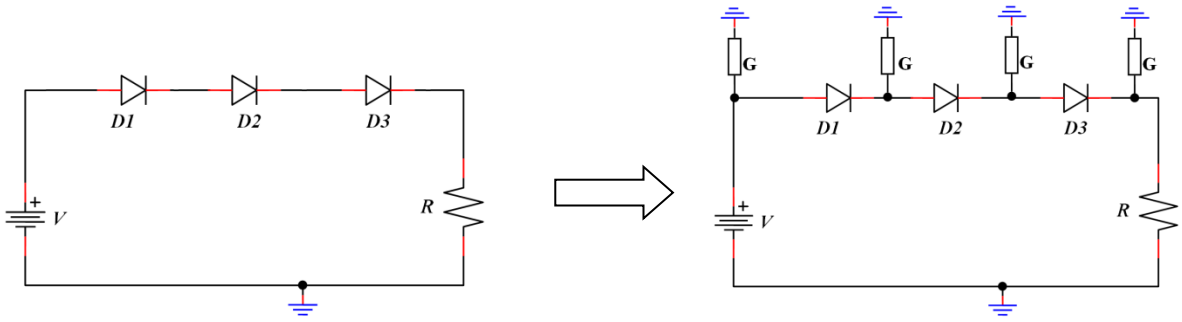


Fig. I.18. The position of G in circuit.

I.8.3. Source Stepping

In the source stepping method [9] as presented in Fig I.19, all the voltage sources in the circuit are multiplied by a constant $\lambda \in [0, 1]$. When making a DC analysis, the iteration was started with $\lambda=1$ (the original circuit). If convergence is not achieved, the constant λ was decreased to a new iterates was attempted.

When the iteration converges, λ is increased gradually, until $\lambda=1$. The operating point found with the previous convergent iteration was used as an initial guess for the next iteration. Source stepping method has an easy physical interpretation — the circuit is "powered up" gradually by increasing the source voltages. Source stepping is a very efficient method when dealing with strongly nonlinear circuits.

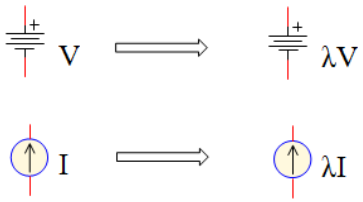


Fig. I.19. The equivalent of current source and voltage source when applying the steping in the source.

I.8.4. Voltage Limiting

Voltage limiting is a simple method for preventing semiconductor components from switching on and off repeatedly during the iteration [2]. It can be defining a constant U_{max}, which describes the maximum allowed voltage change between iteration cycles. If the voltage change between two iteration cycles exceeds U_{max}, all the voltages are multiplied with a constant β so that the maximum voltage change is U_{max}, If none of the voltage changes exceeds U_{max}, $\beta = 1$:

$$\hat{v}_{k+1} = v_k + \beta \left(v_{k+1} - v_k\right) \tag{1.59}$$

Where:

$$\beta = \begin{cases} \frac{U_{max}}{\|v_{k+1} - v_k\|} & for \quad \|v_{k+1} - v_k\| > U_{max} \\ 1 & for \quad \|v_{k+1} - v_k\| < U_{max} \end{cases} \tag{1.60}$$

I.8.5. Diode damping

Diode damping [11] is a convergence improvement method for dealing with the steep exponential characteristic curve of diodes. It is closely related to the voltage limiting method. Every diode monitors its voltage, and the damping factor is calculated:

$$\beta = \begin{cases} 1 & if \quad \|v_{k+1} - v_k\| \leq V_T \, ln\left(\frac{V_T}{\sqrt{2}\, I_s}\right) \\ \frac{V_T \ln\left(\frac{v_{k+1} - v_k}{V_T} + 1\right)}{\left(v_{k+1} - v_k\right)} & if \quad \|v_{k+1} - v_k\| > V_T \, ln\left(\frac{V_T}{\sqrt{2}\, I_s}\right) \end{cases} \tag{1.61}$$

Where V_T and I_S are the thermal voltage and the saturation current of the diode, respectively. The new voltage was also calculated from the equation (1.61).

I.8.6. Continuation method

Continuation method or Homotopy method is effictivnesse method and replace all last techniques [12-31], is attractive for finding DC operating points of nonlinear circuits due to its global convergence in principle, if the NR method fails to converge to a solution in SPICE simulator [17-31].

I.9. Various Continuation (Homotopy) Methods

As an effective method to overcome the non-convergence or problem of convergence of NR method, homotopy methods were regarded as continuous deformation of functions and show the promising in resolving the computational difficulties often encountered in

transistor network simulations. Based on the understanding of this basic concept, we introduce homotopy methods as follow.

To employ a continuation method we embed a "continuation parameter" in the circuit nonlinear equations. By setting the parameter to zero, the system is reduced to the one that the equations can be solved easily or whose solution is trivial. The solution to this simple problem becomes the starting point of a continuation path. The augmented equations are then continuously deformed, as the parameter was varied, until they finally describe the originally posed difficult problem. In more detail, for example, the nonlinear equation to be solved as:

$$F(x) = 0 \tag{1.62}$$

where $F: \mathbb{R}^{n+1} \longrightarrow \mathbb{R}^n$. Let us create a homotopy mapping:

$$H: \mathbb{R}^{n+1} \longrightarrow \mathbb{R}^n \tag{1.63}$$

Then we solve the equation:

$$H(x, \lambda) = 0 \tag{1.64}$$

While varying the continuous homotopy parameter λ. As an example, a simple homotopy [16] is

$$H(x, \lambda) = \lambda F(x) + (1 - \lambda)\,(x - x_0) \tag{1.65}$$

where $\lambda \in [0,1]$is the continuation parameter, $x \in \mathbb{R}^n$, $x_0 \in \mathbb{R}^n$, is the starting vector for the homotopy paths. This homotopy mapping $H(x, \lambda)$ has the following properties: $H(x_0, 0) = 1$ starts the continuation and $H(x_1, 1) = F(x_1)$ ends the continuation.

The algorithm outline is as follows: First solve the problem. $H(x_0, 0)$, e.g. set $\lambda = \Delta\lambda = 0.01$ and try to solve. $H(x, \lambda)$. If NR converged then increase λ by $\Delta\lambda$ and double $\Delta\lambda = 2\,\Delta\lambda$, otherwise half $\Delta\lambda = 0.5\,\Delta\lambda$ and set $\lambda = \lambda_{prev} + \Delta\lambda$. Repeat this until $\lambda = 1$.

Algorithm I.3: Algorithm sample of continuation method.

1. *Choose $\Delta\lambda$=0.01 and x_0*
2. *Start with λ= 0 , $\lambda_{prev} = 0$ and solve H(x,λ) = 0*
3. *Step λ forward: $\lambda = \lambda_{prev} + \Delta\lambda$.*
4. *Calculate by NR algorithm: $x_1^{k+1} = x_1^k - H\left(x_1^k, \lambda_1\right)/J\left(x_1^k, \lambda_1\right)$*

5. *If there is no convergence, a half step of* $\Delta\lambda$, ($\Delta\lambda$=0.5 $\Delta\lambda$), *go to stage 3.*
6. $\Delta\lambda = 2\,\Delta\lambda$ *and* $\lambda_{prev} = \lambda$
7. $x_0 = x_1$
8. *If* ($\lambda < 1$) *go to stage 3.*

The several of continuation (Homotopy) Methods were based by homotopy type function and solving method.

I.10. Homotopy function

There are a four type of homotopy function: the Newton, fixed-point, nonlinear and variable gain homotopy functions described in e.g., [6,14,25], it are reviewed and tested with different solvers how described after these section.

I.10.1. Newton Homotopy

The Newton Homotopy (NH) is based on the equation:

$$H(x,\lambda) = F(x) + (\lambda - 1)\,F(x_0) \tag{1.66}$$

where an insertion of the parameter λ, and an initial value (random vector) x_0 are embedded. At λ=0, the starting point of the homotopy path, the added branches contain only a voltage source and force the nodal voltages to be equal to the elements x_1 of the initial value (random vector) x_0. When λ=1, the added branches get disconnected from the circuit and the augmented circuit reverts to the original circuit [6].

I.10.2. Fixed Point Homotopy

The fixed-point homotopy (FPH) was based on the equation [12,13]:

$$H(x,\lambda) = \lambda F(x) + (1-\lambda)\,A(x - x_0) \tag{1.67}$$

where, in addition to the parameter λ is a initial value (random vector) x_0 and a new parameter (a matrix) $A \in R^n \times R^n$ are embedded. A random choice of x_0 may result in a bifurcation-free homotopy path [16]. This homotopy has an interesting circuit interpretation [25].

I.10.3. Newton Fixed Point Homotopy

The Newton fixed-point homotopy (NFPH) is based on the equation of NH and FPH [17,18]:

$$H(x,\lambda) = F(x) + (\lambda - 1)\,F(x_0) + (1-\lambda)\,A(x - x_0) \tag{1.68}$$

According to the Newton fixed-point homotopy equation, we have gated the calculate method:

$$\begin{aligned} J_H &= J_0 + [(1-\lambda)A\ F(x_0) + A(x-x_0)] \\ RHS_H &= RHS_0 + F(x_0) - \lambda\, A\,(x-x_0) \\ x_0 &= 0 \\ F(x_0) &= \begin{bmatrix} I \\ -E \end{bmatrix} \end{aligned} \tag{1.69}$$

where J_H is the Jacobian matrix for homotopy equation, J_0 is the Jacobian matrix for circuit which have been load during homotopy implementation. x is a vector of $[v,i,\lambda]^T$, A is nonsingular matrix, I is the value of current source in the circuit, E is the value of voltage source in the circuit, and *RHS* is the "*Right Hand Side*".

I.10.4. Nonlinear Homotopy

The nonlinear homotopy was based on the following equation [14,21]:

$$H(x,\lambda) = F(x) + (\lambda - 1)\,[F(x_0) + \tilde{F}(x_0)] + (1-\lambda)\tilde{F}(x) \tag{1.70}$$

According to the nonlinear homotopy equation, we have gated the equations:

$$J_H = J_0 + [(1-\lambda_m)\tilde{F}_x(x_m)\ F(x_0) + F(x_0) - \tilde{F}(x_m)] \tag{1.71}$$

$$RHS_H = RHS_0 + [F(x_0) + \tilde{F}(x_0)] - \lambda_m \tilde{F}_x(x_m)x_m - [\tilde{F}(x_m) - \tilde{F}_x(x_m)x_m] \tag{1.72}$$

I.10.5. Variable Gain Homotopy

The variable gain homotopy function is [19]:

$$H(x,\lambda) = F(x,\lambda\alpha) + (1-\lambda)A\,(x-x_0) \tag{1.73}$$

where α is a vector consisting of forward and reverse current gains, it is used for reduce the high gain in the exponential function of transistors and diodes in circuit by the homotopy parameter λ. For A is a matrix and it the same form in FPH and FPNH.

I.10.6. Variable Gain Newton Homotopy

The VGH (1.73) it has good direction for solving by application continued method. By interaction with NH (1.66) we obtained VGNH equation [19,20]:

$$H(x,\lambda) = F(x,\lambda\alpha) + (1-\lambda)\,F(x_0, 0.\alpha) + (1-\lambda)A\,(x-x_0) \tag{1.74}$$

The VGNH is performance then of NH, FPH, FPNH and VGH, because [26]:

- The auxiliary equation at is closely related to the original nonlinear equation.
- Since this method is globally convergent for any initial point, we can choose a good initial point.
- Can be easily implemented on SPICE without programming.

The result of simulation is execute that the comparison between VGNH, FPH and FPNH in for circuit with BJT's elements as shown in Fig I.14 [19,20].

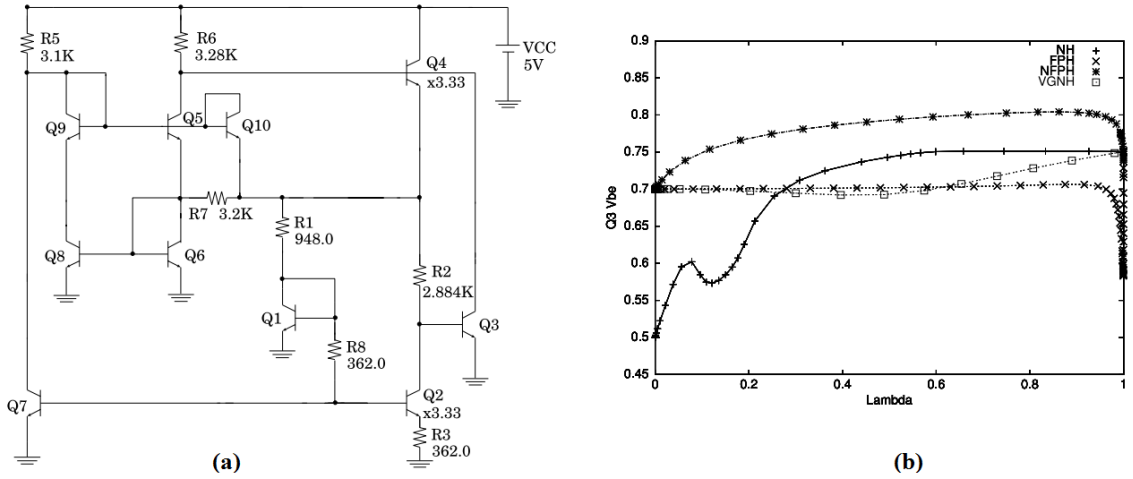


Fig. I.20. Solution curves for the hybrid voltage reference circuit (HVRef): (a) Circuit (b) Solution curves.

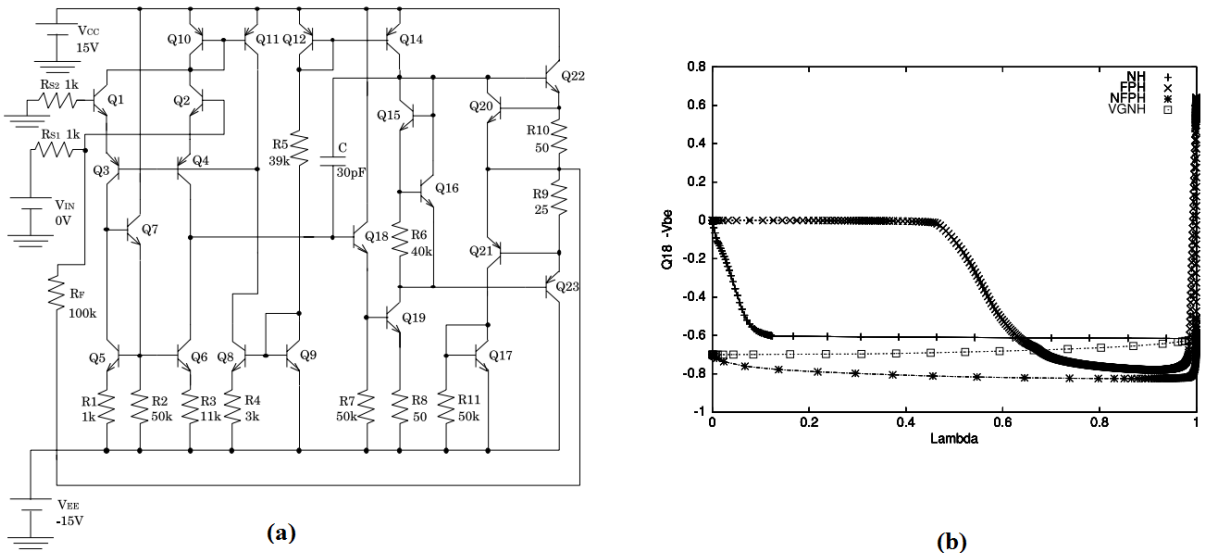


Fig. I.21. Solution curves for the high gain operational amplifier: (a) Circuit (b) Solution curves.

I.11. Solvers by Continous method

Other continuous algorithm can be used for solving of homotopy function.

I.11.1. Predictor corrector

It simplest way to solve the homotopy equations was used for a continuation solver by just stepping the homotopy parameter λ by choosing a setup size $\Delta\lambda$ ($0<\Delta\lambda<1$) and using predictor-corrector algorithms. The predictor function computes a point all the line tangent to the homotopy path at the point x_0, and then a corrector algorithm will be executed to get back on the solution [28] (see Fig.20.).

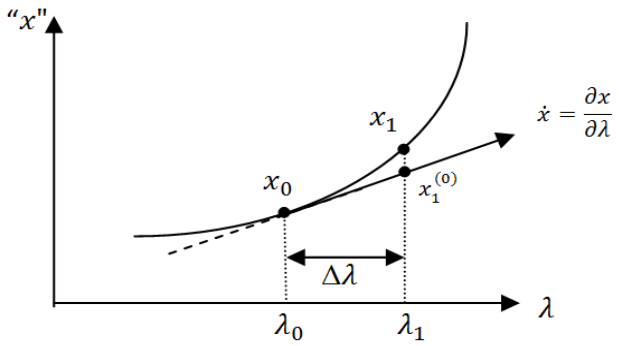


Fig. I.22. Predictor-corrector method

More specifically, if (x_0, λ_0) is the current point on the homotopy path, then giving the new coordinates based on the tangent vector which will be applied as initial approximation as follows:

$$x_1^0 = x_0 + \Delta\lambda\dot{x}_0 \tag{1.75}$$

$$\dot{x}_0 = \partial x/\partial\lambda \tag{1.76}$$

After that, the solution x_1 of $H(x_1,\lambda_1)$ at $\lambda_1=\lambda_0+\Delta\lambda$ can be computed by the corrector function using NR iteration:

$$\begin{cases} d_k = -H\left(x_1^k, \lambda_1\right)/J\left(x_1^k, \lambda_1\right) \\ x_1^{k+1} = x_1^k + d_k \end{cases} \tag{1.77}$$

where, $J = \partial H/\partial x$ is the Jacobien function of H and "k" is the iteration index. To ameliorate the continuation parameter, it must have a good predictor value or a good tangent value which has relationship with the performance step $\Delta\lambda$ (see chapter II).

I.11.2. Pseudo-arclength

The pseudo-arclength method described in [15] was based on tracing the zero curves; it is the turning points the values of λ and *x* decreases as the path progresses .by making λ and *x* a function of a new parameter: the arc length "*s*". How parameter *s* trace this method is known as the arc length continuation of the homotopy function *H(x, λ)*.

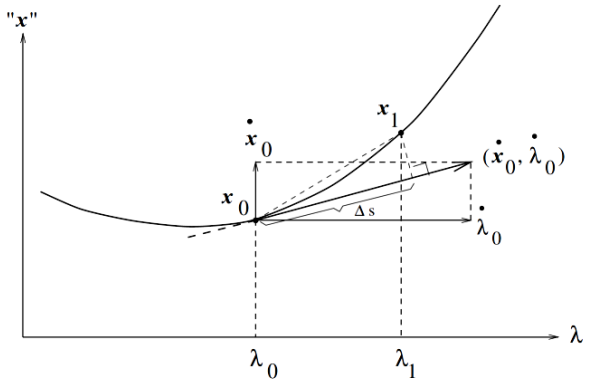


Fig. I.23. Graphical interpretation of pseudo-arclength continuation.

The basic idea of the pseudo-arc-length: on suppose we have a solution *(x_0, λ_0)* of *H(x, λ) =0*, as well as the direction vector $(\dot{x}_0, \dot{\lambda}_0)$ of the solution branch for $\dot{x}_0 = \partial x_0/\partial s$ and $\dot{\lambda}_0 = \partial\lambda_0/\partial s$. Pseudo-arclength continuation solves the following equations for *(x_1, λ_1)*:

$$\begin{cases} H(x_1, \lambda_1) = 0 \\ (x_1 - x_0)'\dot{x}_0 + (\lambda_1 - \lambda_0)\dot{\lambda}_0 - \Delta s = 0 \end{cases} \tag{1.78}$$

Fig. 4 shows a graphical interpretation of the continuation method. Newton's iteration method for pseudo-arc-length continuation becomes (for $J_x = \partial H/\partial x$ and $J_\lambda = \partial H/\partial\lambda$ is jacobien function of *H* in vector *x* and parameter *λ* and "*k*" is index of iteration)

$$\begin{pmatrix} J_x(x_1^k, \lambda_1^k) & J_\lambda(x_1^k, \lambda_1^k) \\ \dot{x}_0{}' & \dot{\lambda}_0 \end{pmatrix} \begin{pmatrix} \Delta x_1^k \\ \Delta\lambda_1^k \end{pmatrix} = \begin{pmatrix} H(x_1^k, \lambda_1^k) \\ (x_1^i - x_0)'\dot{x}_0 + (\lambda_1^i - \lambda_0)\dot{\lambda}_0 - \Delta s \end{pmatrix} \tag{1.79}$$

New iteration

$$\begin{cases} x_1^{k+1} = x_1^k + \Delta x_1^k \\ \lambda_1^{k+1} = \lambda_1^k + \Delta\lambda_1^k \end{cases} \tag{1.80}$$

with the new direction vector defined as:

$$\begin{pmatrix} J_x(x_1, \lambda_1) & J_\lambda(x_1, \lambda_1) \\ \dot{x}_1{}' & \dot{\lambda}_1 \end{pmatrix} \begin{pmatrix} \dot{x}_1 \\ \dot{\lambda}_1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{1.81}$$

As initial approximation, we have used the following formula:

$$\begin{cases} x_1^0 = x_0 + \Delta s\, \dot{x}_0 \\ \lambda_1^0 = \lambda_0 + \Delta s\, \dot{\lambda}_0 \end{cases} \tag{1.82}$$

Note that:

The Jacobian of the pseudo-arc-length system is nonsingular at a regular solution point.

• In practice $(\dot{x}_1, \dot{\lambda}_1)$ can be computed with one extra back-substitution.

• The orientation of the branch is preserved for if Δs is sufficiently small.

• The direction vector must be rescaled, so that indeed $\|\dot{x}_1\|^2 + \dot{\lambda}_1^2 = 1$.

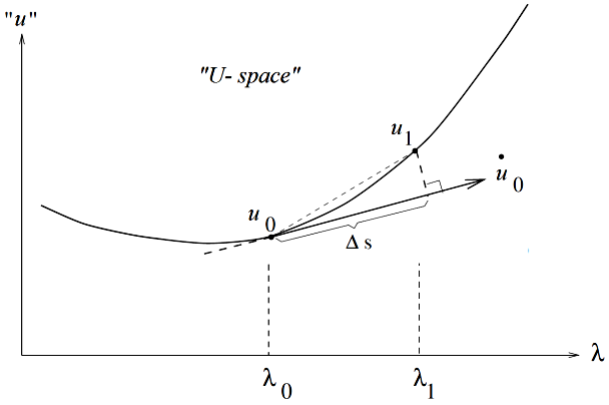


Fig. I.24. Fuses new space (Parameter-independent) pseudo-arclength continuation.

Let $u = (x, \lambda) \in R^{n+1}$. By $\lambda - x$ hyperplane in the spice u (Fig. I.17) and unit vector $\dot{u}_0$ for the direction vector $(\dot{x}_0, \dot{\lambda}_0)$.

Computation of $\dot{u}$

The good calculate of approximation $\dot{u}$, on obtain a good predictor, *i. e.* the construction of a good initial iterate for correction by Newton iteration at the next point on the path. For calculate $\dot{u}$ by three methods:

A. Analytic Computation of $\dot{u}$

Solving a system of equations:

$$\begin{cases} \frac{dH(x,\lambda)}{dx}\dot{x} + \frac{dH(x,\lambda)}{d\lambda}\dot{\lambda} = 0 \\ \|\dot{x}\| + \left\|\dot{\lambda}\right\| = 1 \end{cases} \tag{1.83}$$

The solution of equation (1.83) is

$$\begin{cases} \dot{\lambda} = \pm \frac{1}{\sqrt{1+\|v\|}} \\ \dot{x} = \dot{\lambda} v \end{cases} \tag{1.84}$$

Where

$$v = -\frac{\frac{dH(x,\lambda)}{d\lambda}}{\frac{dH(x,\lambda)}{dx}} \tag{1.85}$$

By equation (1.84) $\dot{\lambda}$ has two possible value (the curve has two tangent vectors pointing in opposite directions), a test should be made in order to prevent the algorithm reverse the direction on the curve [31].The only thing remaining is to compute the sign of $\dot{\lambda}$, this is an important thing to do correctly, for if we get the sign wrong, we could recomputed the path in the direction we came from, and keeping the sign of $\dot{\lambda}$ constant.

B. Secant Approximation of $\dot{u}$.

If we have computed two points on the path $u_{-1} = u(s_{-1})$ and $u_0 = u(s_0)$, we can use the approximation

$$\dot{u} = D_u \| D_u \| \tag{1.86}$$

Where

$$D_u = \frac{u_0 - u_{-1}}{s_0 - s_{-1}} \tag{1.87}$$

In (1.87). [31] If one does this, the one must initialize the continuation with two solutions *(x_i, λ_i), i=-1,0* of *H(x, λ)=0*, and then estimate $s_0 - s_{-1}$ by

$$s_0 - s_{-1} = \sqrt{\|x_0 - x_{-1}\|^2 + (\lambda_0 - \lambda_{-1})^2} \tag{1.88}$$

When the unit vectors $\dot{\boldsymbol{u}}$ have been calculated, the predictor step is taken:

$$u_{tan} = u_0 + \dot{u}\, \Delta s \tag{1.89}$$

If the path is smooth, then the trivial predictor is a first order (in $\Delta s = s - s_0$) correct approximation to the solution $u(s)$ and the tangent predictor is second order correct.

Changing Δs in response to the curvature of the path for the nonlinear solver performance, and effective use of the structure of the problem. We will discuss these issues in the sections that follow.

I.11.3. ODE-based method

The ODE-based method is an algorithm for tracing the zero curve of the homotopy function. In the method, the homotopy function was parameterized with respect to the arc length of the zero curves. Then, a differential equation generating that trajectory was derived

and solved. In the ODE-based approach, the zero curve of the homotopy function was treated for solution of an ordinary differential equation. The ODE was formed by parameterizing both x and λ by the arc length s [28-30].

$$H(x(s), \lambda(s)) \tag{1.90}$$

The problem can be treated as the solution of differential equation

$$\frac{\partial H(x(s),\lambda(s))}{\partial s} \tag{1.91}$$

From the chain rule,

$$DH(x(s), \lambda(s)) \ \frac{\partial \lambda / \partial s}{\partial x / \partial s} \tag{1.92}$$

Where

$$DH(x(s), \lambda(s)) \tag{1.93}$$

Is the Jacobian of the homotopy function; with initial conditions?

$$\lambda(0) = 0, \ x(0) = a, \text{ and } \ \left\| \left(\frac{\partial \lambda}{\partial s}, \frac{\partial x}{\partial s} \right) \right\| = 1 \tag{1.94}$$

The differential equation (19) was solved. The more detailed description is found in [28] and [29]. ODE solver used for finding critical points (limit points and bifurcation points) see the figure above, the application ODE solver in four-transistor circuit.

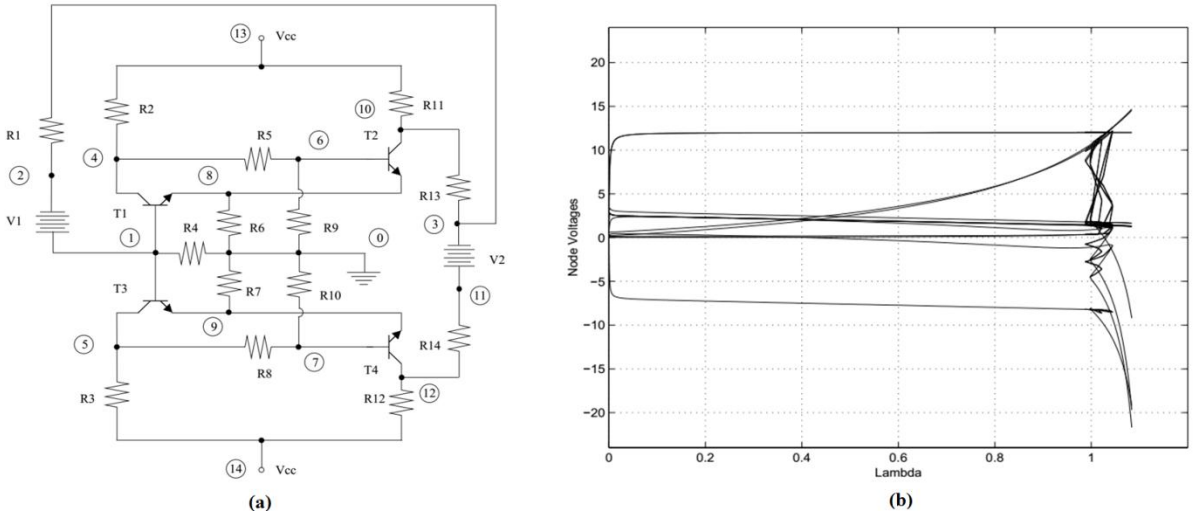


Fig. I.25. (a) Four-transistor circuit that has nine dc operating points. (b) Homotopy paths for the fourteen node voltages of the four-transistor circuit [28,29].

I.12. Conclusion

The NR iterative method used for solving system of nonlinear equations and applied in simulator SPICE to simulate circuit in one of three modes DC, TR and AC. The NR is the best numerical method for applying in analyses of circuit because it has an idea to convert nonlinear elements (Diode, BJT, MOSFET) and storage of energy elements (capacitor, inductor) to linear elements (resistor, voltage source, current source) which were depended on Newton iteration, this convert simplify to convert nonlinear circuit to linear circuit, the subject of converting for simplified to get system linear of circuit by applied KCL and KVL or directly by MNA and solving by the linear method "LU decomposition".

The NR has problems with convergence, the problems are the slow convergence, the divergence, the cyclic phenomena or oscillating, numerical overflew and the nearly singular Jacobian. There are some techniques applied in SPICE proposed to overcome the problem of convergence as:

- Limiting method used to solve problem nearly singular Jacobian.
- Source stepping method and *Gmin* method used to have good starting of the initial guess.
- Voltage limiting and diode damping method used to solve overflow problem.

These techniques: Limiting method, Source stepping method, *Gmin* method and diode damping are integrated with NR algorithm. But in this way, it is long algorithm and generate slow in convergence to find operating points. The *Continuous* or *Homotopy* method is one algorithm replace of all those techniques and integrated easily in SPICE and faster to find operating points. For that, other good performance methods of continues solver will be examined in the next chapter.

Chapter II

Accelerate the Solving of Systems Nonlinear Equations by using the Homotopy Method

II.1. Introduction

This chapter describes a method that can be used to accelerate the solving of nonlinear equations and to find the operating point in various integrated circuits by construction of the global homotopy equation of the analog circuit. This is done by converting the elements in the circuit to their equivalent dependent sources. The proposed method is based on four steps. The first step is the use of the homotopy method to get a continuous function. The second step consists of finding the best direction of the solution by applying the prediction process; the third is the correction process. The fourth and the new added step is the control of the step size to accelerate the solution search. In order to demonstrate the effectiveness of the proposed method, a comparison was done between the proposed method and other methods for five types of practical circuits widely used in analog LSI's are considered in this chapter. These circuits consist of Ebers-Moll BJT which has a big problem of convergence in the SPICE software, they are frequently used as a test circuits:

- The hybrid voltage reference circuit (HVRef).
- Six-stage limiting amplifier (6sLA).
- Operational amplifier (µA741).
- Wideband amplifier (RCA3040).
- Basic two-stage operational amplifier (2sOA).

II.2. The continuation method

The continuation method, proposed in [12-31] and [42-55], is an effective method to solve nonlinear circuits. The continuation method is based on three steps:

1. Construct a continuous function by using the homotopy method.

2. Find the best direction of the solution by using the Predictor-Corrector. The predictor computes the approximation of the solution of homotopy function. To calculate the predictor we can use Secant [54] [27] or Euler methods [55][27]. Then, the predicted approximate solution is corrected by applying the corrector, e.g., by Newton's method.
3. Accelerate the solution search by controlling the step size.

Based on these three steps, this paper investigates the development of an approach to accelerate the solving of nonlinear equations. It also attempts to locate the operating point in various analog and integrated circuits by either modeling the analog devices (BJT, diode, resistor, etc.) using the homotopy (continuation) parameters or by construct the global homotopy equation of the analog circuit. This is done converting the elements in the circuit to their equivalent dependent sources.

II.3. Homotopy function

Homotopy function is a common method used to solve nonlinear algebraic equations systems and can be applied to a large variety of problems [53]. We are most interested in solving the zero finding problems:

$$F(x) = 0 \tag{2.1}$$

Where, $x \in \mathbb{R}^n$ and $F: \mathbb{R}^n \longrightarrow \mathbb{R}^n$

The Homotopy function $H(x, \lambda)$is based on embedding a parameter λ into $F(x)$.As a result, an equation of new dimension:

$$H(x, \lambda) = 0 \tag{2.2}$$

Where, $\lambda \in \mathbb{R}$ and $H: \mathbb{R}^n \times \mathbb{R} \longrightarrow \mathbb{R}^n$. The parameter $\lambda$is called the continuation or homotopy parameter.

At present, the most widely used method is the Newton homotopy mapping (NH):

$$H(x, \lambda) = F(x) + (1 - \lambda) F(x_0) \tag{2.3}$$

$x_0 \in \mathbb{R}^n$ is the solution of the homotopy function when $\lambda = 0$.

According to the homotopy mapping, the solution of the equation $F(x) = 0$ can be adopted to solve the homotopy equation $H(\lambda, x) = 0$. For any λ range from 0 to 1, if the homotopy equation solution (x, λ)exists, the corresponding curve of (x, λ) starts from $(x_0, 0)$ and ends in the solution $(x^*, 1)$.

Other equation homotopy Variable-Gain Homotopy (VGH) [19][20], Method the equation it presents in these form:

$$H(x,\lambda) = F(x,\lambda\alpha) + (1-\lambda)A(x-x_0) \qquad (2.4)$$

Where α is a vector consisting of forward and reverse current gains, it using for reduced the high gain in the exponential function of transistors and diodes in circuit by the homotopy parameter (Fig. II.1.a). The values of the gain matrix *A* of VGH (2.4), must be the same as the values of G_{min} in the circuit structure of SPICE [2]. For that the *A* matrix have a resistive element the position value it parallel resistance of the diode element and the diode of the internal transistor (see Fig. II.2).

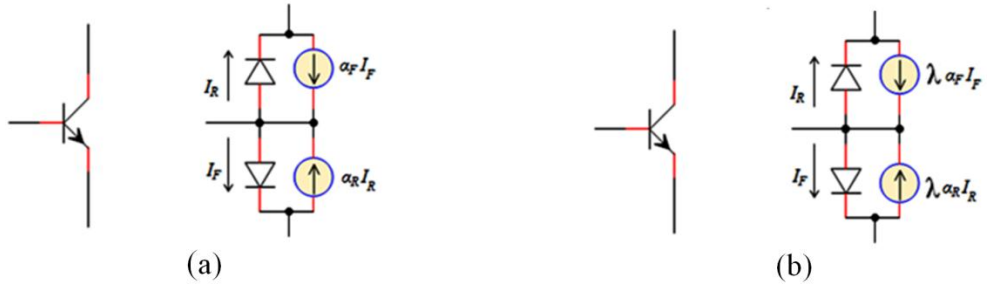


Fig. II.1. (a) The Ebers-Moll model for an npn BJT. (b) Reduced the high gain by the homotopy parameter.

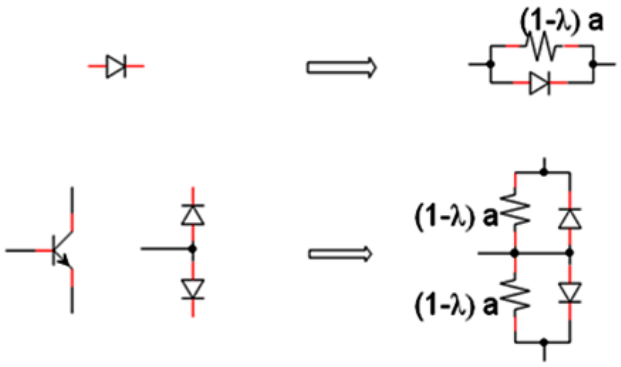


Fig. II.2. The position of *A* element in semi-conductor device when applying VGH equation.

The VGH (2.4) it has good direction for solving by application continued method. By integred with NH (2.3) we obtained VGNH equation [20]:

$$H(x,\lambda) = F(x,\lambda\alpha) + (1-\lambda)F(x_0,0.\alpha) + (1-\lambda)A(x-x_0) \qquad (2.5)$$

The VGNH is performance then of VGH and NH because [19]:

- The auxiliary equation at is closely related to the original nonlinear equation.
- Since this method is globally convergent for any initial point, we can choose a good initial point.
- Can be easily implemented on SPICE without programming.

II.4. Solving the homotopy equation

The continuation method has widely used in the literature to solve the homotopy eq. (2.2) [33-34]. Firstly, the homotopy parameter λ should be adjusted by choosing a step size $\Delta\lambda$ ($0<\Delta\lambda<1$), then, the predictor-corrector method (illustrated in Fig. II.3) is used. The predictor computes the approximation of the solution by computing a point along the tangent line to the homotopy path at the point (x_0, λ_0) using the following formula:

$$x_1^0 = x_0 + \Delta\lambda\dot{x}_0 \tag{2.6}$$

Where,

$$\dot{x}_0 = \partial x/\partial\lambda \tag{2.7}$$

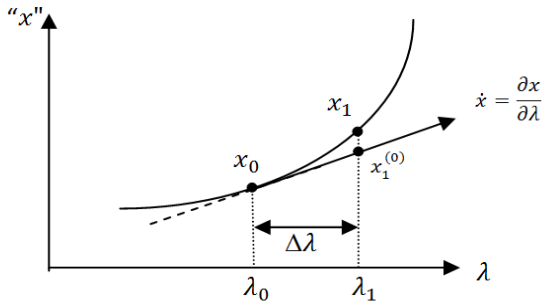


Fig. II.3. Predictor-corrector method

After that, the solution x_1 of $H(x_1,\lambda_1)$ at $\lambda_1=\lambda_0+\Delta\lambda$can be computed by the corrector function using NR iteration:

$$\begin{cases} J(x_1^i, \lambda_1)\Delta x_1^i = -H(x_1^i, \lambda_1) \\ x_1^{i+1} = x_1^i + \Delta x_1^i \end{cases} \tag{2.8}$$

Where, $J = \partial H/\partial x$ is the Jacobien function of H, *"i"* is the iteration index and x_1^0 is the initial approximation or the initial of gauss calculated by the predictor eq. (2.6).

II.4.1. Prediction

In this part, we discuss how to calculate a good prediction value to obtain a good direction of correction. There are two methods for calculating prediction: the secant method and the Euler method (Fig. II.4).

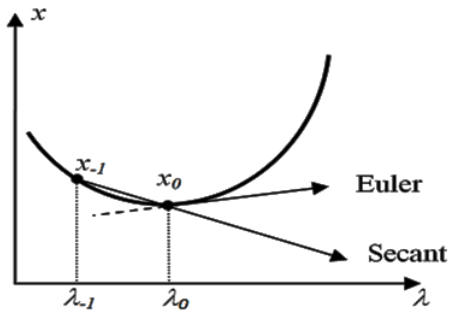


Fig. II.4. Calculating the tangent by the secant method and the Euler method.

The secant prediction is a very simple method based on two previous points of the vector *x*. The format of the tangent in the secant prediction is:

$$\dot{x}_0 = \frac{x_0 - x_{-1}}{\lambda_0 - \lambda_{-1}} \tag{2.9}$$

Euler prediction is a popular method for calculating the prediction tangent. It is based on *Davidenko Differential Equation* [27][55]:

$$\frac{\partial H}{\partial x}\dot{x} + \frac{\partial H}{\partial \lambda} = 0 \tag{2.10}$$

Using the equation (2.10), we obtain the tangent equation:

$$\dot{x} = -\left(\frac{\partial H}{\partial x}\right)^{-1}\frac{\partial H}{\partial \lambda} \tag{2.11}$$

Numerical analysis is an effective method to calculate the tangent $\dot{x}$ in Euler prediction:

- Replacing *(∂H/∂x)* by the Jacobian J,which is calculated by NR algorithm.
- The derivative *(∂H/∂λ)* can be presented by the backward difference methods [54].

Therefore, we obtain a new equation close to the numerical derivation of Euler prediction method:

$$\dot{x} = -J(x,\lambda)^{-1}\frac{H(x,\lambda) - H(x,\lambda - \partial\lambda)}{\partial\lambda} \tag{2.12}$$

II.4.2. Correction

To have good calculation in the correction process, first, we must find another performance direction in the fifth stage of the algorithm (01):

$$d_i = -H\left(x_1^i, \lambda_1\right)/J\left(x_1^i, \lambda_1\right) \tag{2.13}$$

For the Newton step we discuss the convergence using the Armijo rule [1], which is based on the Line Search Method for optimization or decreasing in $\|H\|$ along the line segment $[x_1^i, x_1^i + d_i]$, it is based on finding the smallest value σ such that:

$$\left\|H\left(x_1^i + \sigma d_i, \lambda_1\right)\right\| < (1 - \alpha d_i)\left\|H\left(x_1^i, \lambda_1\right)\right\| \tag{2.14}$$

The parameter $\alpha \in [0,1]$ is a small number intended to make (2.14) as easy as possible satisfied to get the new step:

$$x_1^{i+1} = x_1^i + \sigma d_i \tag{2.15}$$

Notes: 1- The stop iteration of the correction process in analog circuits is the same used in the SPICE simulator [4].

2- A good initial approximation x_0 is essential. This will be discussed below.

Improving the continuation parameter requires either a good predictor value or a good tangent value that has a relationship with the performance step $\Delta\lambda$. If there is a problem in the convergence in the correction process, a half of the previous step ($\Delta\lambda/2$) should be used to find a new prediction. An algorithm proposed in [55] to solve the homotopy equation is given below:

Algorithm II.1: Algorithm of predictor and corrector (Alg. 01):

6. *Choose $\Delta\lambda$ and x_0*
7. *Start with $\lambda = 0$ and solve $H(x,\lambda) = 0$*
8. *Step λforward*:$\lambda = \lambda + \Delta\lambda$.
9. *Calculate the predictor*:$x_1^0 = x_0 + \Delta\lambda\dot{x}_0$
10. *Calculate the corrector by NR algorithm:*
    - $x_1^{i+1} = x_1^i - H\left(x_1^i, \lambda_1\right)/J\left(x_1^i, \lambda_1\right)$
    - *Use the Line Search Method for optimization.*
11. *If there is no convergence, a half step of $\Delta\lambda$, ($\Delta\lambda=\Delta\lambda/2$), go to stage 3.*
12. *Choose a good step $\Delta\lambda$*
13. $x_0 = x_1$

14. If $\lambda < 1$, go to stage 3.

15. Return the final solution.

II.5. New method to control the step size

In Alg. (01) proposed in [55], the prediction-correction of the homotopy method uses half-step control strategy. More specifically, the step length is increased by the user if it converges rapidly. If not, it will be decreased by a half-step. Other methods such as Asymptotic Expansion and Den Heijer& Reinbolt step length adaptationare are discussed for Newton-Gauss iterations in [53]. Unfortunately, although a half-step is used, the convergence remains a problem. To overcome this problem, a new method of choosing a new step size $\Delta\lambda$ based on an adaptive step length is proposed. The subject of step length adaptation is to eliminate the problem of the slow convergence or even the divergence in the correction process. This method is mainly based on the performance of the Newton iterations. A simple scheme may be that we increase the step length where a few Newton iterations are needed to compute the last point; conversely, we decrease the step length when the previous point needs many Newton iterations.

To choose the best adaptive step length, we must know the relationship between the step length and the convergence problem in Alg. (01). To do that, we choose the value of $\Delta\lambda$ which can give the convergence in the process of correction (i.e., the NR iteration $\|\Delta x_i\| > \varepsilon_xor\|H(x_i, \lambda_1)\| > \varepsilon_H$, where,ε_x and $\varepsilon_H$are the errors of the stop iteration in NR method [1]), as result of the correction process we get this relation:

$$1 \geq 1/\left(1 + \frac{\|H\|}{\varepsilon_H}\right) > 0 \qquad (2.16)$$

This value $1/\left(1 + \frac{\|H\|}{\varepsilon_H}\right)$is close to 0when we have divergence and takes the value 1 in the case of the convergence. For that, the adaptive step length must be directly proportional to $\Delta\lambda_0$ and inversely proportional to $1 + \frac{\|H\|}{\varepsilon_H}$. It can be given by the following equation:

$$\Delta\lambda_1 = \left(\frac{1}{1+G\|H\|}\right)\Delta\lambda_0 \qquad (2.17)$$

1. Where, G is a gain in the interval $1 < G < 1/\varepsilon_H$. By using the eq. (2.17) we can automatically increase or decrease the step length if there is convergence or divergence, respectively. However, there is another problem, when we have slowly

convergence; the eq. (2.17) will not increase the step length $(1 + G\|H\| \approx 1)$. In order to correct this, we must use an n-step length. The eq. (2.17) can be then improved to:

$$\Delta\lambda_1 = \left(\frac{n}{1+G\|H\|}\right)\Delta\lambda_0 \qquad for \;\; n > 1 \tag{2.18}$$

The new proposed algorithm for this new adaptive step is:

Algorithm II.2: Algorithm of update of predictor and corrector (Alg. 02):

1. *Chose* $\Delta\lambda_0$ *,n, G and* x_o
2. *Start with* $\lambda = 0$, $\Delta\lambda_1 = \Delta\lambda_0$ *and solve* $H(x,\lambda) = 0$
3. *Step* λ *forward* $\lambda=\lambda+\Delta\lambda_1$.
4. *Calculate the predictor by Euler method:*
   $x_1 = x_0 + \Delta\lambda_1\dot{x}_0$
5. *Calculate the corrector by NR algorithm, and find the smallest value* σ *by the line search method:*
   $x_1^{i+1} = x_1^i + \sigma d_i$
6. *If there is Convergence by NR:* $x_0=x_1$.
7. $\Delta\lambda_1 = \Delta\lambda_0 \frac{n}{1+G\|H\|}$.
8. *If* $\lambda < 1$, *go to step 3.*
9. *Return the final solution.*

The difference between Alg.01 and Alg.02 in the new step-length adaptation is seen on the circuits modeling using the homotopy method in the simulation section.

II.6. Modeling of analog elements using the continuous parameter

In this section, we will look at how to change the structure of an analog element by embedding the homotopy parameter to acquire good acceleration in the operating point with use the continuation solver (Alg.01, Alg.02).The continuation solver starts from a linear element ($\lambda = 0$), and then reverts to the original problem ($\lambda = 1$). To do so, we need to create a global homotopy equation of analog circuit. It consists of three steps:

1. Convert any element in the circuit to its equivalent in either a dependent voltage sources or a dependent current source in the circuit. .
2. Create the global equation of the circuit that it has a current source presented by the KCL (Kirchhoff's Current Law) method and a voltage source presented by the MNA (Modified Nodal Analysis) method.

3. Transform the global equation to the homotopy equation.

II.6.1. Convert any element in the circuit to its equivalent

In this step, any element should be converted to its equivalent of voltage source or current source in the circuit in which their values depend on the voltage in the nodes of the element or the current across the voltage sources.

- **Example 1:**

A resistor can be represented by a current source I (Fig. II.5-a), that is dependent on the voltage between the node v_1 and v_2 and expressed by:

$$I(v_1, v_2) = \frac{v_1 - v_2}{R} \tag{2.19}$$

Where, R is the resistance value.

The resistor can also be represented by a voltage source V (Fig. II.3-b), dependent on the voltage between the nodes v_1 and v_2 and the current i across the source V, it is expressed by:

$$V(i) = Ri \tag{2.20}$$

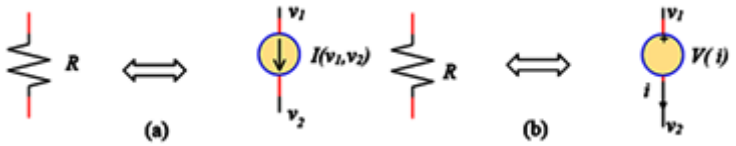


Fig. II.5. The equivalent of resisstor: (a) dependent current source. (b) dependent voltage source.

- **Example 2:**

The diode can be represented by its equivalent nonlinear current source I, dependent on the voltage in the nodes v_1 and v_2 Fig. II.6. It is given by:

$$I(v_1, v_2) = Iss\,(exp\left(\frac{v_1 - v_2}{Vt}\right) - 1) \tag{2.21}$$

Fig. II.6. The equivalent of diode by dependent voltage source.

- **Example 3:**

In this example, we deal with the BJT, which is represented by The Ebers-Moll model [4], shown in Fig. II 7-a. For current I_R and I_F, we have:

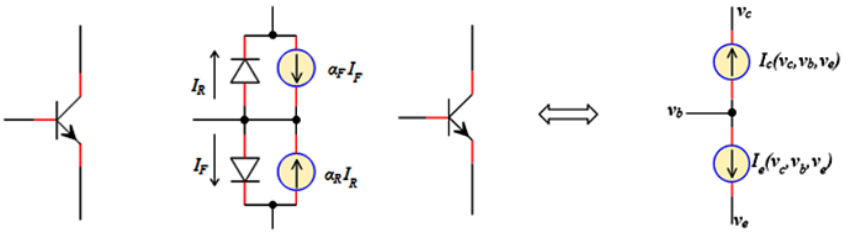
$$I_R = I_{se}\left(exp\left(\frac{V_{be}}{V_{te}}\right) - 1\right) \quad (2.22)$$

$$I_F = I_{sc}\left(exp\left(\frac{V_{bc}}{V_{tc}}\right) - 1\right) \quad (2.23)$$

The equivalent of the transistor can be represented by two current sources (Fig. II.5-b), namely I_c and I_e and is dependent on the voltage in the three nodes v_c,v_b and v_e:

$$I_c(v_c, v_b, v_e) = I_{sc}\left(exp\left(\frac{v_b - v_c}{V_{tc}}\right) - 1\right) - \alpha_R I_{se}\left(exp\left(\frac{v_b - v_e}{V_{te}}\right) - 1\right) \quad (2.24)$$

$$I_e(v_c, v_b, v_e) = -\alpha_F I_{sc}\left(exp\left(\frac{v_b - v_c}{V_{tc}}\right) - 1\right) - I_{se}\left(exp\left(\frac{v_b - v_e}{V_{te}}\right) - 1\right) \quad (2.25)$$



(a) (b)

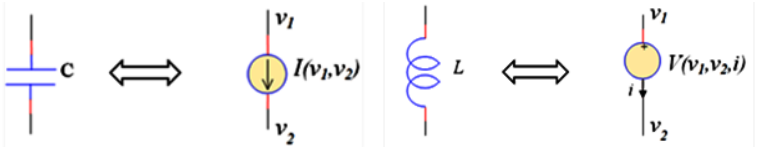
Fig. II.7. (a) The Ebers-Moll model for an npn BJT. (b)The equivalent of BJT by dependent current source.

- **Example 4:**

The capacitor and the inductance are elements of storage of energy, and they are dependent on time. The capacitor is equivalent to a current source *I*, which is dependent on the voltage in the nodes v_1 and v_2 (Eq. 24 and Fig. II.8-a). The inductance is represented by a source voltage dependent on the current across the source (Eq. 25 and Fig. II.8-b).

$$I(v_1, v_2) = C\frac{\partial(v_1 - v_2)}{\partial t} \quad (2.26)$$

$$V(i) = L\frac{\partial i}{\partial t} \quad (2.27)$$



(a) (b)

Fig. II.8. (a) The equivalent of capacitor by dependent current source. (b) The equivalent of inductance by dependent voltage source.

- **Example 5:**

The equivalent of the transistor MOSFET (N-Channel), can be represented by current sources I_d (Fig. II.9) and is dependent on the voltage in the three nodes v_g,v_s and v_d:

$$I_d(v_d, v_s, v_g) =$$

$$\begin{cases} 0 & v_g - v_s \le V_T \\ K_P \frac{W}{L}\left((v_g - v_s - V_T)(v_d - v_s) - \frac{(v_d - v_s)^2}{2}\right)\left(1 + Lambda\ (v_d - v_s)\right) & v_d \le v_g - V_T \\ K_P \frac{W}{L}(v_g - v_s - V_T)^2\left(1 + \ Lambda\ (v_d - v_s)\right) & v_d \ge v_g - V_T \end{cases} \quad (2.28)$$

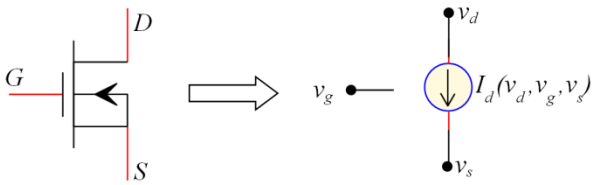


Fig. II.9. The equivalent of NMOS by dependent current source.

II.6.2. Create the global equation of the circuit

The subject of converting the circuit elements to their dependent sources is to simplify the generation of the global equation of the circuit by application two laws: the KCL law for the dependent current source and MNA law for the dependent voltage source [4]. The result is the global equation of the circuit described by the following formula:

$$F(v,i) = [P_i \quad P_v]\begin{bmatrix} i \\ v \end{bmatrix} + [P_I \quad P_V]\begin{bmatrix} I(v,i) \\ V(v,i) \end{bmatrix} \quad (2.29)$$

Where,

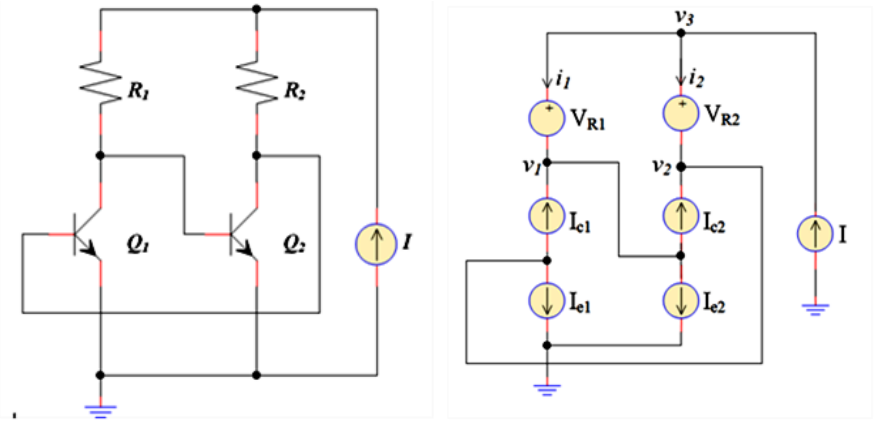
- $I(v,i)$: The dependent current source.
- $V(v,i)$: The dependent voltage source.

- v: The voltage in the node.
- i: The current across the voltage source.
- P_i: Position and direction of *i* in the circuit by 0 and ±1.
- P_v: Position and direction of *v* in the circuit by 0 and ±1.
- P_I: Position and direction of *I* in the circuit by 0 and ±1.
- P_V: Position and direction of *V* in the circuit by 0 and ±1.
- $(v, i) = (v_1, v_2, \ldots . v_n, i_1, i_2, \ldots, i_m)$.

Note: *F* is the global function where, $F: \mathbb{R}^{n+m} \longrightarrow \mathbb{R}^{n+m}$, *n* represent the number of nodes in the circuit and *m* the number of voltage sources in the circuit.

Example of circuit:

In the circuit illustrated in Fig. II.10-a, by replacing the equivalent of BJTs by their dependent current sources and the resistor by the dependent voltage source, the equivalent circuit shown in Fig. II.10-b is obtained, the equation system of this circuit uses the KCL and MNA laws in the three nodes.



(a) (b)

Fig. II.10. Example of a circuit: a) Simple osillator circuit with BJTs, b) The equivalent of the circuit using dependent source

The expression of the equivalent circuit can be obtained applying the KCL law in the three nodes and the MNA laws which is given by:

$$F(v,i) = \begin{cases} f_1: i_1 + I_{c1}(v) - I_{c2}(v) - I_{e2}(v) \\ f_2: i_2 + I_{c2}(v) - I_{c1}(v) - I_{e1}(v) \\ f_3: i_1 + i_2 - I(v) \\ f_4: v1 - v3 + V_{R1}(i) \\ f_5: v2 - v3 + V_{R2}(i) \end{cases} \quad (2.30)$$

Where,$(v,i) = (v_1, v_2, v_3, i_1, i_2)$, because the number of nodes in the circuit of Fig. II.10-b is three, and i_1 *and* i_2 are across the two dependent voltage sources V_{R1} and V_{R2} respectively. So that equation (2.30) can be written as:

$$F(v_1, v_2, v_3, i_1, i_2) = (f_1, f_2, f_3, f_4, f_5) \tag{2.31}$$

And can be expressed directly using Eq. (2.29) as:

$$F(v,i) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} i_1 \\ i_2 \\ v_1 \\ v_2 \\ v_3 \end{bmatrix} + \begin{bmatrix} 1 & -1 & 0 & -1 & 0 & 0 & 0 \\ -1 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} I_{c1}(v) \\ I_{c2}(v) \\ I_{e1}(v) \\ I_{e2}(v) \\ I(v) \\ V_{R1}(i) \\ V_{R2}(i) \end{bmatrix} \tag{2.32}$$

II.6.3. Transform the global equation to the homotopy equation

Constructing the global equation Eq. (2.27) simplifies the construction of the homotopy equation which has the general form of Eq. (2.3), thus we obtain the global homotopy equation given by the following formula:

$$H(v,i,\lambda) = F(v,i) + (1-\lambda)\,F(v_0,i_0) \tag{2.33}$$

To obtain the same step source used in SPICE, we have to add $(v_0, i_0) = (0,0)$, the final result is:

$$H(v,i,\lambda) = [P_i \quad P_v]\begin{bmatrix} i \\ v \end{bmatrix} + [P_I \quad P_V]\left(\begin{bmatrix} I(v,i) \\ V(v,i) \end{bmatrix} + (1-\lambda)\begin{bmatrix} I(0,0) \\ V(0,0) \end{bmatrix}\right) \tag{2.34}$$

In this homotopy equation we obtain the step in the source:

$$\begin{bmatrix} I(v,i) \\ V(v,i) \end{bmatrix} + (1-\lambda)\begin{bmatrix} I(0,0) \\ V(0,0) \end{bmatrix} \tag{2.35}$$

Eq. (2.35) eliminates the problem of convergence of the circuit applying the method of continuous solver.

Finally, we can compare between Alg.01 and Alg.02 for the acceleration of the solving in the circuit represented by Eq. (2.34), which will be detailed in the next section.

II.7. Simulation results

Simulation results were done using a type of new software created by the author called PyAMS (Python Language for Analog and Mixed Signal). The software is written in Python language version 2.7, and the graphical user interface is programmed with Delphi XE6. The software simplifies the modeling of the elements using their dependent sources and solves the homotopy equation of the circuit using Alg2. It can be applied to analyze the circuits. The user interface is illustrated in chapter IV .

To demonstrate the effectiveness of the proposed method:

Comparison of the proposed algorithm and other methods. Five types of practical circuits widely used in analog LSI's are considered in this comparison. These circuits consist of Ebers-Moll BJT and have a convergence issues in the SPICE software, but are frequently used as a test circuits [25][54][55]:

- The hybrid voltage reference circuit (HVRef).
- Six-stage limiting amplifier (6sLA).
- Operational amplifier (μA741).
- Wideband amplifier (RCA3040).
- Basic two-stage operational amplifier (2sOA).

The equations of these circuits are realized by the global homotopy equation Eq. (2.34).

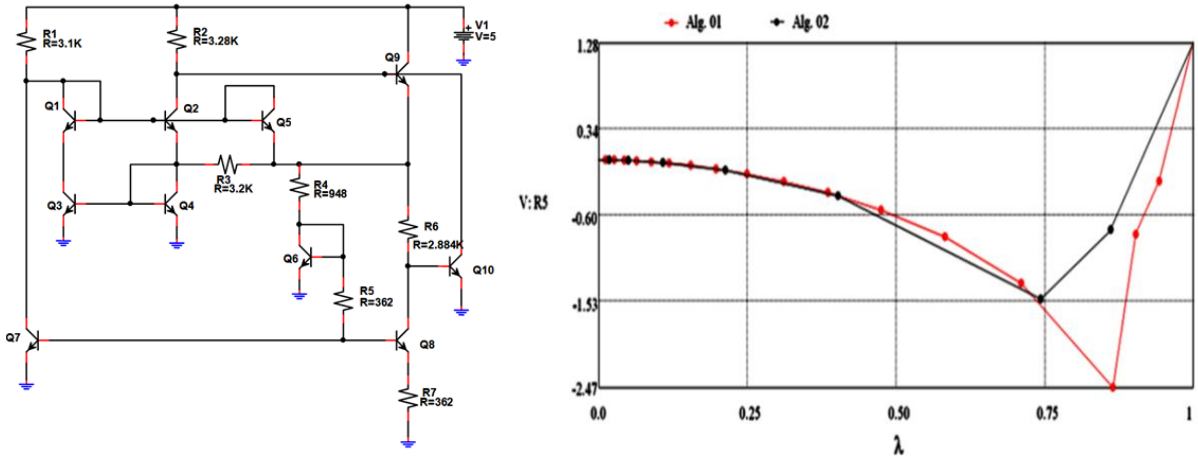


Fig. II.11. The HVRef circuit: left) the schematic diagram of the circuit, right) the comparison between Alg. 01 and Alg. 02

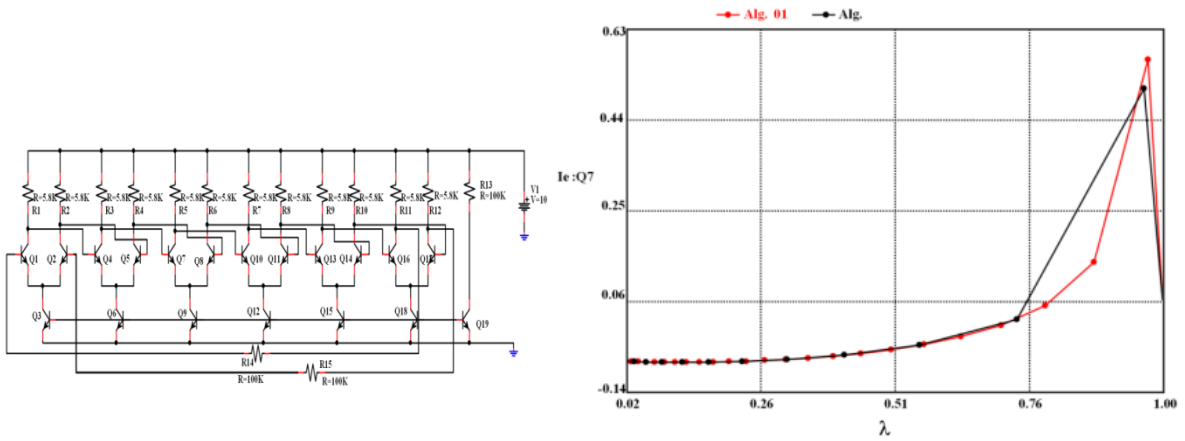


Fig. II.12. The 6sLA circuit: left) the schematic diagram of the circuit, right) the comparison between Alg. 01 and Alg. 02

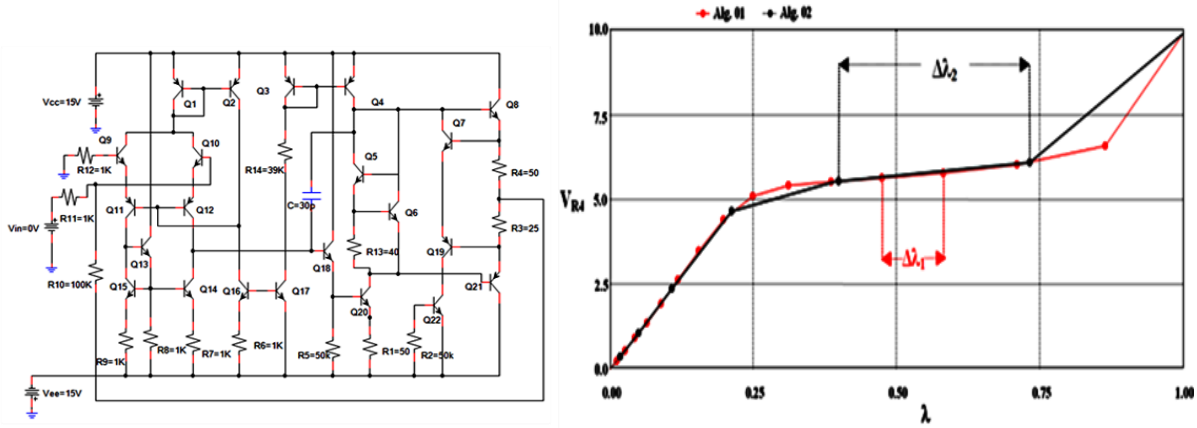


Fig. II.13. The µA741 circuit: left) the schematic diagram of the circuit, right) the comparison between Alg. 01 and Alg. 02

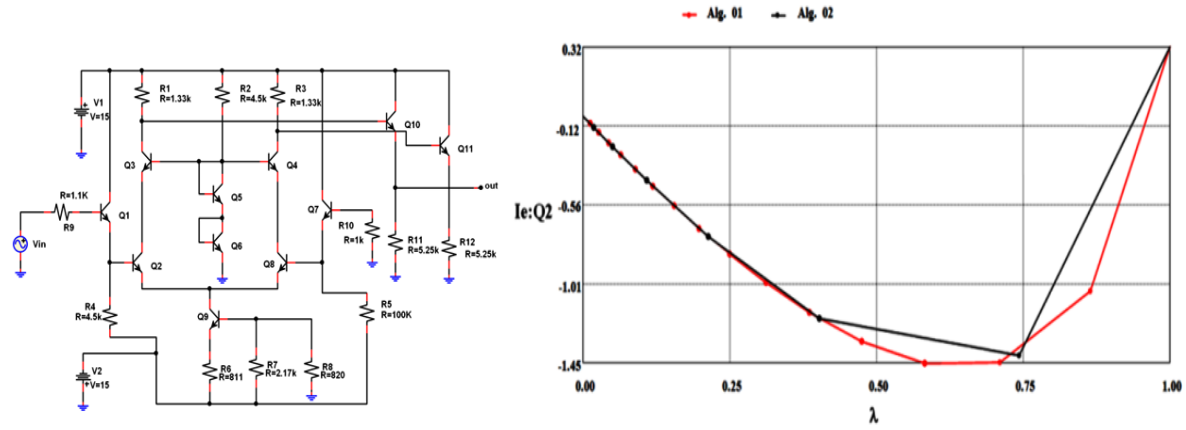


Fig. II.14. The RCA3040 circuit: left) the circuit, right) the comparison between Alg. 01 and Alg.02

Fig. II.11- II.14 compare the proposed algorithm Alg. 02, and the Alg. 01 proposed in [22].

The figure on the left gives the schematic diagram of the circuit realized in PyAMS while the figure on the right compares both Alg. 02 is represented with a black line and Alg. 01 with the red line. We can count the number of steps λ in the right figure using the points on the lines, and it can be observed that the number of iterations used in Alg. 02 is lower than the number used in Alg. 01; therefore, Alg. 02, the proposed algorithm, is faster than Alg. (01).

It can also be noted that the curves in the figure on the right can be obtained using any point in the circuit. In the first circuit (Figure II.11), we used the voltage between the terminals of resistor R5. In second circuit of Figure II.12, we used the emitter current of transistor Q7. In the circuit shown in Figure II.13, the voltage between the terminals of resistor R4 is used. Finally, in the fourth circuit of Figure II.14, the emitter current of transistor Q2 is used.

The comparison of the computational efficiency is summarized in the Table II.1. The Table gives a comparison between the proposed method (Alg. 02) and two other methods; the first method is the VGNH (Variable Gain Newton Homotopy) (2.4) implemented in the SPICE software [22] while the second is the method (Alg. 01) proposed in [50].The parameters used in Alg. 2 are: $G=10^6$, $n=1.6$ and $\Delta\lambda_0 = 0.08$.

Table. II.1 Comparison of computational efficiency

Circuit	**Number of steps λ in:**		
	VGNH	**Alg.01**	**Alg.02**
HVRef	21	15	07
6sLA	20	26	12
µA741	28	16	07
RCA3040	21	15	08
2sOA	20	15	05
Average	22	17.4	7.8

Note that for the fifth circuit 2sOA in the Table II.1, the textual description of the circuit was used rather than the schematic diagram.

We can see in the table that the proposed algorithm Alg. 02 is faster than the two methods; the method used in SPICE and the one used in [19][20]. From the average line, the proposed algorithm Alg. 02 is about three times faster than the VGNH method of SPICE.

II.8. Conclusion

A method to accelerate the solving of nonlinear equations has been presented. The proposed method is based on four steps. The first step is the use of the homotopy method to get a continuous function. The second step is applying the continuation parameter solver with a modification in the predictor (initial approximation) by using Euler method. The third step is predicted approximate solution is corrected by applying the corrector. The last step is the control of the new step size to accelerate the solution search. In order to demonstrate the effectiveness of the proposed method, new software called PyAMS (Python Language for Analog and Mixed Signal) based on the proposed method (Chap. IV). A comparison was done between the proposed method and two other methods, and many types of integrated circuits were used in the process. Circuit analysis of many universal circuits was carried out to verify the correct functioning of the circuit based on the proposed method. Simulation results showed the effectiveness of the proposed method. It is approximatly three times faster than the VGNH method implemented in the SPICE software.

Chapter III

Application of Third-Order Convergence in Continues Method

III.1. Introduction

In the last years use of the continuation method has attracted a big attention for its effectiveness to solve nonlinear circuits.

Continues method based by predictor corrector methods, in corrector it used Newton order two. We present in this chapter a new acceleration of solving system nonlinear equation, is a Third-Order family of Newton with cubic convergence, which includes; to find operate point by application in process of correction in continues method and analyses analog circuit with the aid of PyAMS software (Python for analog and mixed signal Ch. IV). A general errors analysis of convergence in analog circuits is given, and numerical illustrations are given to compare the proposed methods with some other methods: Globally convergent algorithm of NR, Third-order and Newton-Krylov.

III.2. Globally convergent algorithm of NR

The NR (1.2) is a second-order method [4-5], you can presented in the form:

$$x_{k+1} = x_k + d_k \tag{3.1}$$

Where k is the iteration index, x_k the current approximate solution and d_k is direction of NR:

$$d_k = \frac{F(x_k)}{J(x_k)} \tag{3.2}$$

Globally convergent algorithm of NR is based by line search method [4]. The line search is used to find good direction of solving by minimization of direction d_k from interval $[d_k, d_k/2^j]$ for some $j \geq 0$. The algorithm of NR with line search direction to solve the system

nonlinear equation is given below.

Algorithm III.1: Algorithm NR (update 1)

1. *Choose x_0 initial approximation and ε_F error of convergence.*
2. *Start with k= 0*
3. *Search direction* $d_k = -F(x_k)/J(x_k)$
4. *Calculate trial point* x_{k+1}*:*
   - $x_{k+1} = x_k + d_k$
   - If $\|\mathrm{F}(x_{k+1})\| < \|\mathrm{F}(x_k)\|$ then
     $x_k = x_{k+1}$ or $k = k + 1$ (accept the step)
     else $d_k = d_k/2$ goto 4 (reject the step)
5. *If there is no convergence* $(\|F(x_k)\| > \varepsilon_F)$ *go to stage 3.*
6. *Return the final solution.*

The algorithm is to compute a search direction d_k which for us will be the Newton direction and then test steps of the form $\lambda\, d_k$, with $\lambda = 2^{-j}$ for some $j \geq 0$. Until $\mathrm{F}(x_k + \lambda\, d_k)$ satisfies:

$$\|\mathrm{F}(x_k + \lambda\, d_k)\| < (1 - \alpha\lambda)\|\mathrm{F}(x_k)\| \qquad (3.3)$$

The condition in (3.3) is called sufficient decrease of $\|\mathrm{F}\|$. The parameter $\alpha \in (0,1)$ is a small number and used to make (3.3) as possible to satisfy. We follow the recent optimization literature and set $\alpha = 10^{-4}$ [4]. Once sufficient decrease has been obtained we accept the step $\lambda\, d_k$. This strategy, from [4] is called the Armijo rule. The algorithm of NR with line search direction by Armijo rule to solve the system nonlinear equation is given below.

Algorithm III.2: Algorithm NR (update 2)

1. *Choose x_0 initial approximation and ε_F error of convergence.*
2. *Start with k= 0 and* $\alpha = 10^{-4}$
3. $\lambda = 1$
4. Search direction $d_k = -F(x_k)/J(x_k)$
5. *Calculate* trial point x_{k+1}*:*
   - $x_{k+1} = x_k + \lambda\, d_k$
   - if $\|\mathrm{F}(x_{k+1})\| < (1 - \alpha\lambda)\|\mathrm{F}(x_k)\|$*then*

$$x_k = x_{k+1} \ or \quad k = k + 1 \text{ (accept the step)}$$
$$\text{else } \lambda = \lambda/2 \text{ goto 5 (reject the step)}$$

6. *If there is no convergence* ($\|F(x_k)\| > \varepsilon_F$) *go to stage 3.*
7. *Return the final solution.*

III.3. New iterative method and convergence analysis

The third Newton order is used in solving of system nonlinear equation and redacts it in many forms [61-77] based on M. Darvishi and A. Barati method or MA method [61] which is given by this equation:

$$\begin{cases} y_k = x_k - \frac{F(x_k)}{J(x_k)} \\ x_{k+1} = x_k - \frac{F(x_k)}{J(x_k)} - \frac{F(y_k)}{J(x_k)} \end{cases} \tag{3.4}$$

Each iteration in MA needs two evaluations of the vector functions and one evaluation of the Jacobian matrix while the order is three. Through the MA we can reduce the computational cost of Jacobian matrix, in some cases; the sequences produced by the MA converge rapidly than of NR.

This paper suggests a new global formula constructed from (3.4) which improves and accelerate the solving of nonlinear systems of equations and application in analog circuit.

III.3.1. New family based by MA

We obtain a new family of modified Newton method by observing formula (3.4), you can generate global formula:

$$\begin{cases} y_k = x_k - m\frac{F(x_k)}{J(x_k)} & m \neq 0 \\ x_{k+1} = x_k - A\frac{F(x_k)}{J(x_k)} - B\frac{F(y_k)}{J(x_k)} \end{cases} \tag{3.5}$$

Where *A* and *B* are two parameters to be determined such that the iterative method defined by (3.6) and (3.7) have a three order convergence. In the following, sufficient conditions of these parameters have presented by (3.6) and (3.7).

Theorem. *Let $x^* \in D$ be a simple zero of a function $F: D \subset \mathbb{R}^n \to \mathbb{R}^n$ for an open interval D. Assume that F has derivatives up to third order in D and x_0 is sufficiently close to x^*. Then the iterative (3.5) is cubically convergent to x^* if and only if the parameters satisfy.*

$$A = \frac{1}{m^2}(m-1) + 1 \tag{3.6}$$

And

$$B = \frac{1}{m^2} \tag{3.7}$$

Where $m \neq 0$.

Proof. If $x^* \in D$ be the root and e_k be the error at k^{th} iteration, then $e_k = x_k - x^*$. Using Taylor's expansion, we have:

$$F(x_k) = F'(x^*)\left(e_k + L_2 e_k^2 + L_3 e_k^3 + O(e_k^4)\right) \tag{3.8}$$

And

$$F'(x_k) = F'(x^*)\left(1 + 2L_2 e_k + 3L_3 e_k^2 + O(e_k^3)\right) \tag{3.9}$$

Where

$$L_n = \frac{F^{(n)}(x^*)}{n!F'(x^*)}, \qquad n = 2,3 \ldots \tag{3.10}$$

Using (3.8), (3.9) and apply them in y_k (3.5), we have:

$$y_k = x^* + (1-m)e_k + mL_2 e_k^2 + 2m(L_3 - 2L_1^2)e_k^3 + O(e_k^4) \tag{3.11}$$

Now again by Taylor's series, we have:-

$$F(y) = F'(x^*)[(1-m)e_k + (m^2 - m + 1)L_2 e_k^2 - (2m^2 L_2^2 + (m^3 - 3m^2 + m - 1)L_3)e_k^3 + O(e_n^4)] \tag{3.12}$$

Finally, using (3.8), (3.9),(3.12) and (3.5), we get:

$$x_{k+1} = x_k + V_1\, e_k + V_2 e_k^2 + O(e_k^3) \tag{3.13}$$

Where

$$V_1 = -A - B(1-m) \tag{3.14}$$

And

$$V_2 = B(-m^2 - m + 1) + A \tag{3.15}$$

To make Eq. (3.13) third order convergence formula it is sufficient to choose $K_1 = -1$ and $K_2 = 0$, therefore,

$$A = \frac{1}{m^2}(m-1) + 1 \tag{3.16}$$

And

$$B = \frac{1}{m^2} \tag{3.17}$$

Thus

$$e_{k+1} = O(e_k^3) \tag{3.18}$$

Finally, the order of e_k is:

$$e_{k+1} = (2L_2^2 + (m-1)L_3)e_k^3 + O(e_k^4) \tag{3.19}$$

The proof is complete.

III.3.2. New family based by MW and MA

In [63], MW adopted the MA method which is given by:

$$\begin{cases} y_k = x_k - \frac{F(x_k)}{J(x_k)} \\ z_k = x_k - \frac{F(x_k)}{J(x_k)} - \frac{F(y_k)}{J(x_k)} \\ x_{k+1} = x_k - \frac{F(x_k)}{J(x_k)} - \frac{F(y_k)}{J(x_k)} - \frac{F(z_k)}{J(x_k)} \end{cases} \tag{3.20}$$

They proved that it also converges cubically.

New global formula constructed from (3.20) and (3.5) which improve and accelerate the solving of nonlinear systems of equations.

$$\begin{cases} y_k = x_k - m\frac{F(x_k)}{J(x_k)} \quad m \neq 0 \\ z_k = x_k - A\frac{F(x_k)}{J(x_k)} - B\frac{F(y_k)}{J(x_k)} \\ x_{k+1} = x_k - A\frac{F(x_k)}{J(x_k)} - B\frac{F(y_k)}{J(x_k)} - C\frac{F(z_k)}{J(x_k)} \end{cases} \tag{3.21}$$

Where *A*, *B* and *C* are three parameters to be determined such that the iterative method defined by (3.21) have a three order convergence. In the following, sufficient conditions of these parameters have presented by (3.22).

Theorem. *Let $x^* \in D$ be a simple zero of a function $F: D \subset \mathbb{R}^n \to \mathbb{R}^n$ for an open interval D. Assume that F has derivatives up to third order in D and x_0 is sufficiently close to x^*. Then the iterative (3.21) is cubically convergent to x^* if and only if the parameters satisfy.*

$$\begin{cases} A = \frac{1}{m^2} \\ B = \frac{1}{m^2}(m-1)+1 \\ C \in \mathbb{R} - \{1\} \end{cases} \tag{3.22}$$

Where $m \neq 0$.

Proof. If $x^* \in D$ be the root and e_k be the error at k^{th} iteration, then $e_k = x_k - x^*$. We have:

$$z_k = x_k - A\frac{F(x_k)}{J(x_k)} - B\frac{F(y_k)}{J(x_k)} \tag{3.23}$$

Using (3.8), (3.9),(3.12) and (3.23), we have:

$$z_k = x^* + (2L_2^2 + (m-1)L_3)e_k^3 + O(e_k^4) \tag{3.24}$$

Now again by Taylor's series, we have:

$$F(z) = F'(x^*)[(2L_2^2 + (m-1)L_3)e_k^3 + O(e_k^4)] \tag{3.25}$$

We have:

$$x_{k+1} = x_k - A\frac{F(x_k)}{J(x_k)} - B\frac{F(y_k)}{J(x_k)} - C\frac{F(z_k)}{J(x_k)} \tag{3.26}$$

And you can simplify to:

$$x_{k+1} = z_k - C\frac{F(z_k)}{J(x_k)} \tag{3.27}$$

Using (3.24), (3.25), (3.26) and (3.27), we have:

$$x_{k+1} = x^* + (2L_2^2 + (m-1)L_3) - C(2L_2^2 + (m-1)L_3)e_k^3 + O(e_k^4) \tag{3.28}$$

Finally, the order of e_k is three for $\in \mathbb{R} - \{1\}$:

$$e_{k+1} = (2L_2^2 + (m-1)L_3)e_k^3 - C(2\,L_2^2 + (m-1)L_3)e_k^3 + O(e_k^4) \tag{3.29}$$

The proof is complete.

Remark:

1. For *m*=1 and *C=0* in (3.21) we obtain the *MA* formula [61].
2. For *m*=1 and *C=1* in (3.21) we obtain the *MW* formula [63].
3. For *C=0* we obtain the global formula (3.5).
4. For best results of NM (3.21) we use *m*=0.5 or *m*=0.75 or *m*=0.9 for *C=1.75*.

III.4. Globally convergent algorithm of NM

The algorithm of NM or MA or MW with line search direction $d_k$to solve the system nonlinear equation is given below:

Algorithm III.3: Algorithm of NM

1. *Choose x_0 initial approximation, ε_F error of convergence and parameters C and m.*
2. *Get A and B :* $\begin{cases} A = \frac{1}{m^2} \\ B = \frac{1}{m^2}(m-1)+1 \end{cases}$
3. *Start with k= 0*
4. $y_k = x_k - m\frac{F(x_k)}{J(x_k)}$
5. $z_k = x_k - A\frac{F(x_k)}{J(x_k)} - B\frac{F(y_k)}{J(x_k)}$
6. *Search direction* $d_k = -A\frac{F(x_k)}{J(x_k)} - B\frac{F(y_k)}{J(x_k)} - C\frac{F(z_k)}{J(x_k)}$
7. *Calculate* trial point x_{k+1}*:*
   - $x_{k+1} = x_k + d_k$
   - if $\|F(x_{k+1})\| < \|F(x_k)\|$ then

     $x_k = x_{k+1}$ or k = k + 1 (Accept the step)

     else $d_k = d_k/2$ goto 4 (Reject the step)
8. *If there is no convergence* $(\|F(x_k)\| > \varepsilon_F)$ *go to stage 4.*
9. *Return the final solution.*

III.5. Application

In this section, simulation results shows the difference between NR (3.1), MA (3.4), MW (3.20) and new method NM (3.21), for acceleration of solving systems of equations for analog circuits by calculating error of convergence dependent by number of iteration. Also, The effectiveness of NM in analyzing circuit.

The software is used for calculating errors of convergence and analyzing circuit by proposing method which is PyAMS software (Python for Analog and Mixed Signal Chap. IV). PyAMS is created by the editor to simplify: modeling analog and mixed element, design circuit and analyzing circuit.

PyAMS is programmed by Python 2.7 and Interface graphic by Delphi XE8.

The simulation result is composed in two parts: the first part error of convergence and second part comparison NM with Newton-Krylov [4].

III.5.1. Error of convergence

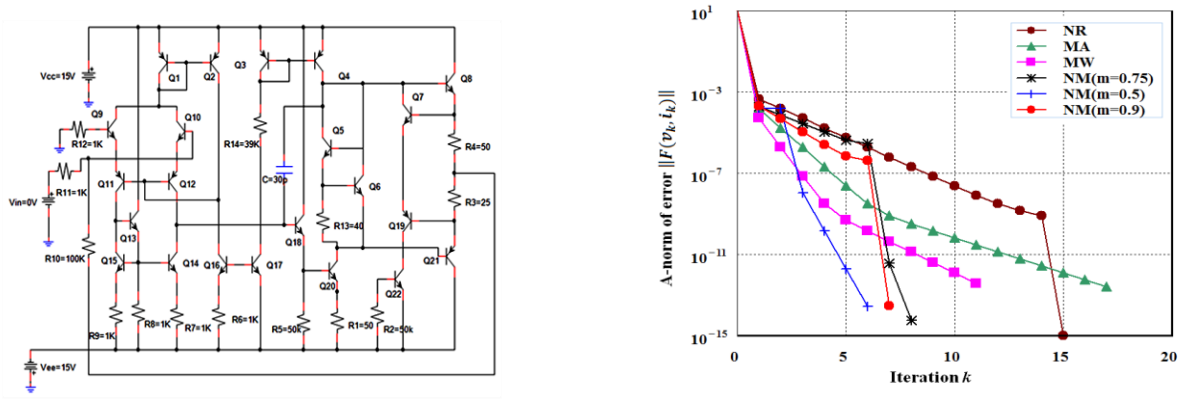
In this section, simulation results show the difference between NR, MA, MW and new method NM (with *m=0.5*, *m=0.75* and *m=0.9* for *C=1.75)* by calculated the error of convergence into six types of practical circuits widely used in analog LSI's:

- Operational amplifier (µA741).
- Wideband amplifier (RCA3040).
- The hybrid voltage reference circuit (HVRef).
- Schmitt trigger circuit.
- Operational transconductance amplifier (OTA).
- Six-stage limiting amplifier (6sLA).

The comparison of computational efficiency is presented by convergence or a-norm of error by number of iteration ($k \rightarrow \|F(v_k, i_k)\|$ for k is index of iteration):

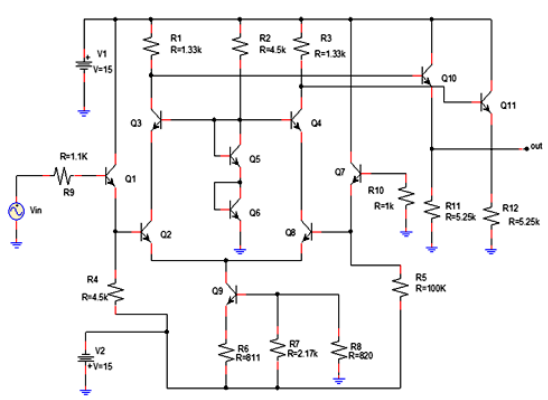
- The circuit of µA741 is given in Fig. III.1.a, while the comparison is shown in Fig. III.1.b.
- The circuit of RCA3040 is given in Fig. III.2.a, while the comparison is shown in Fig. III.2.b.
- The circuit of HVRef is given in Fig. III.3.a, while the comparison is shown in Fig. III.3.b.
- The circuit of Schmitt trigger is given in Fig.III.4.a, while the comparison is shown in Fig. III.4.b.
- The circuit of OTA is given in Fig. III.5.a, while the comparison is shown in Fig.III.5.b.
- The circuit of 6sLA given in Fig. III.6.a, while the comparison is shown in Fig. III.6.b.

For initial approximation it using continues method (chap. II) and error of stop of convergence is 10^{-14}. For the point in the curves of simulation it position of iteration.

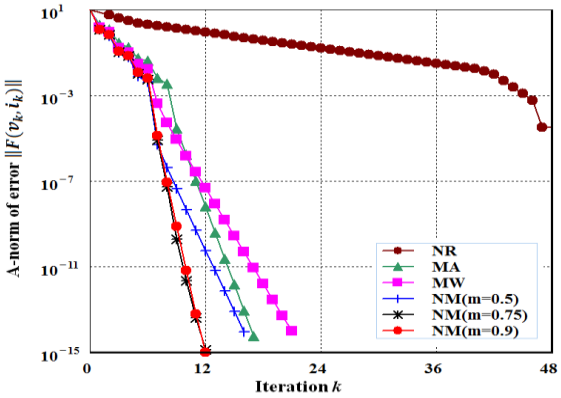


(a) **(b)**

Fig. III.1. Operational amplifier µA741: (a) The circuit (b) Convergence the circuit when applying NR ,MA, MW and NM.

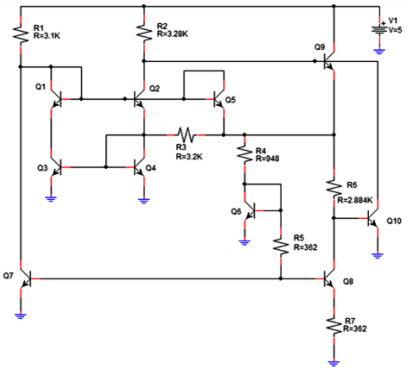


(a)

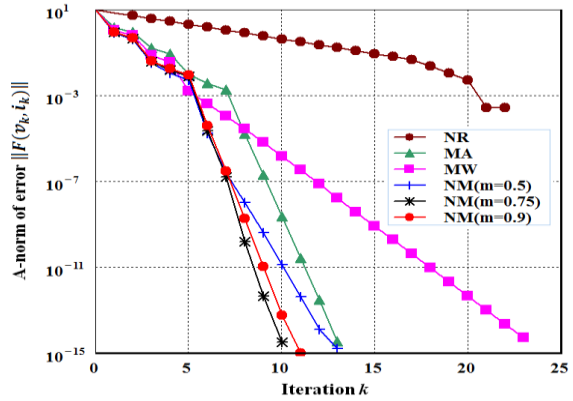


(b)

Fig. III.2. Wideband Amplifier (RCA3040) : (a) the circuit (b) Convergence the circuit when applying NR ,MA, MW and NM.

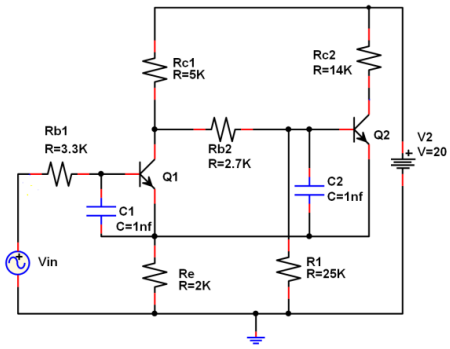


(a)

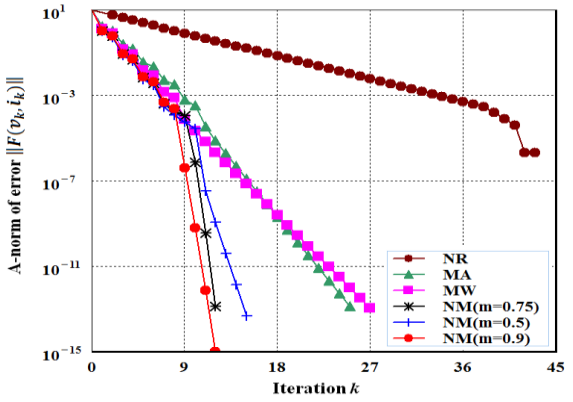


(b)

Fig. III.3. The hybrid voltage reference circuit (HVRef): (a) The circuit (b) Convergence the circuit when applying NR ,MA, MW and NM.

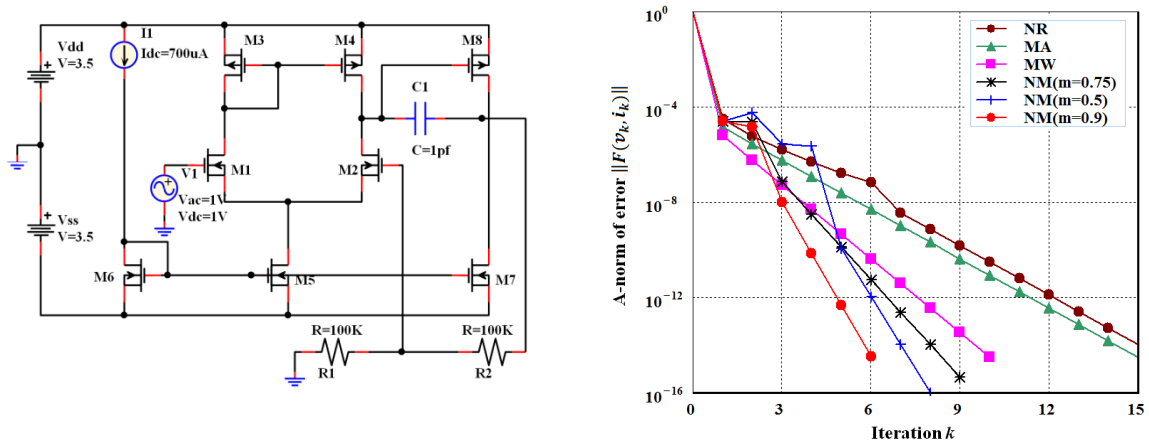


(a)



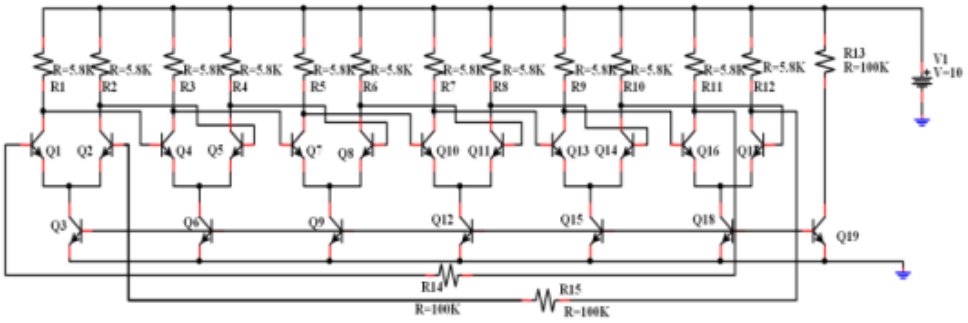
(b)

Fig. III.4. Schmitt trigger circuit: (a) the circuit (b) convergence the circuit when applying NR ,MA, MW and NM.

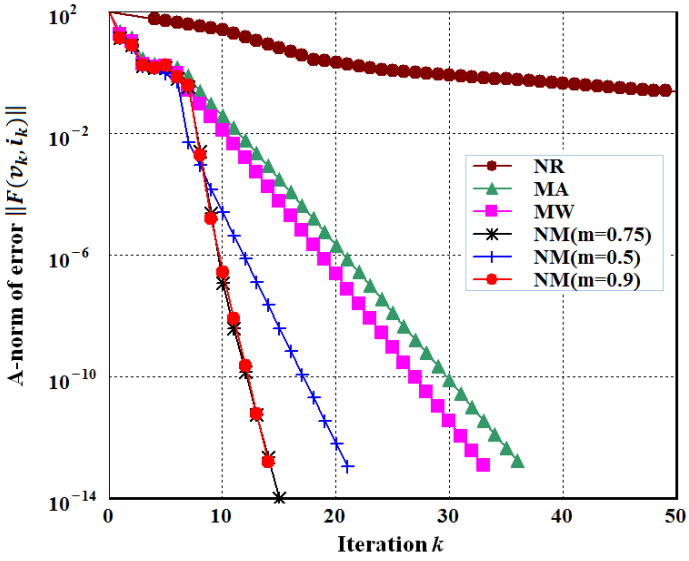


(a) **(b)**

Fig. III.5. Operational transconductance amplifier (OTA) with outstage : (a) the circuit (b) convergence the circuit when applying NR ,MA, MW and NM.



(a)



(b)

Fig. III.6. Six-stage limiting amplifier (6sLA): (a) the circuit (b) convergence the circuit when applying NR ,MA, MW and NM.

Simulation results (Fig. III.1, Fig. III.2, Fig. III.3, Fig. III.4, Fig. III.5 and Fig. III.6) show that the use of *NM* (with *m=0.5, m=0.75* and *m=0.9* for *C=1.75*) for solving system of equation of circuit, it have number of iteration minimum with good convergence then of NR, MA and MW. The result for *m=0.75* it performance then of *m=0.5* and *m=0.9* you can set in Fig. III.4.b, Fig. III.3.b, Fig. III.7.b and Fig. III.6.b.

The NR in simulation it ameliorated by line search direction or Armijo rule [4] to hove good direction of solving, for MA, MW and NW without line search algorithm.

III.5.2. Comparison NM with Newton-Krylov

Krylov Subspace is iterative methods for sparse linear systems [4] or for solving very large system of linear equation [78-112]. The best known Krylov subspace methods are the Arnoldi, Lanczos, Conjugate gradient , IDR(s) (Induced dimension reduction), GMRES (generalized minimum residual), BiCGSTAB (Biconjugate Gradient Stabilized), QMR (quasi minimal residual), TFQMR (transpose-free QMR) and MINRES (Minimal Residual) methods [4][93][94].

In System of equation nonlinear Krylov Space is method to use to approximate the Jacobian matrix by function implements (3.30) a Newton-Krylov solver. The basic idea is to compute the inverse of the Jacobian with an iterative Krylov method. These methods require only evaluating the Jacobian-vector products, which are conveniently approximated by numerical differentiation:

$$J(x)d = \frac{F(x+\varepsilon d)-F(x)}{\varepsilon} \tag{3.30}$$

Due to the use of iterative matrix inverses, these methods can deal with large nonlinear problems. The perfrmance method or the bast method for solving in Krylov subspace is Newton-GMRES (NG) [4]. NG it applicated in SPICE by Sandia National Laboratories [72-77]to create new software for analyzing large circuit by name Xyce « Parallel Electronic Simulator », Xyce is designed to run on large-scale parallel computing platforms, though it also executes efficiently on a variety of architectures, including single processor workstations. As a mature platform for large-scale parallel circuit simulation, Xyce supports standard capabilities available from commercial simulators, in addition to a variety of devices and models specific to Sandia's needs [79-85].

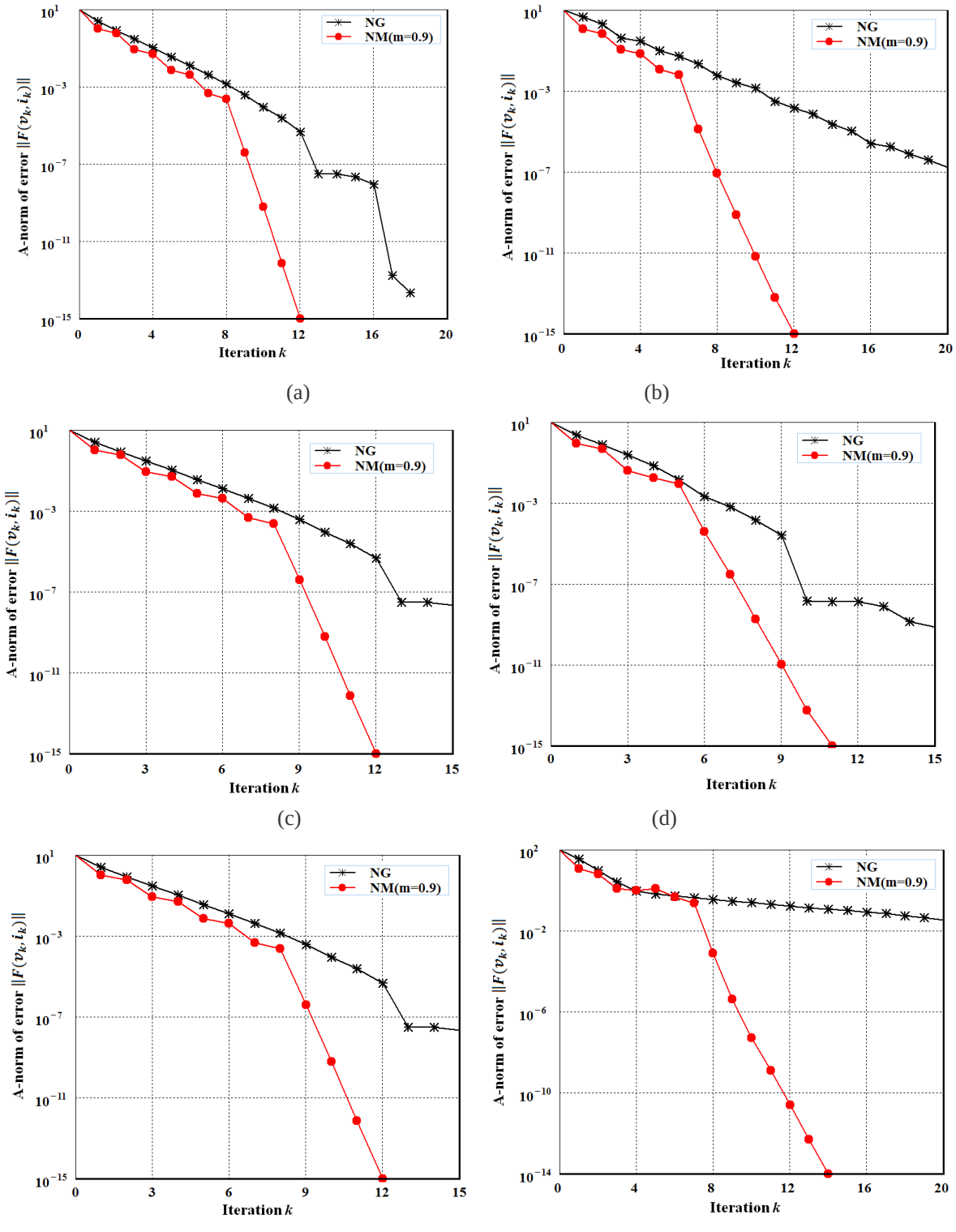


Fig. III.7. Error of convergence the circuits when applying NM (*m=0.9*)and NG.

The NM method for (*c=1.75* and *m=0.9*) it performance of NG you can set the comparison batwing NM and NG is presented by convergence or a-norm of error by number of iteration ($k \rightarrow \|F(v_k, i_k)\|$ for *k* is index of iteration):

- The circuit of µA741 is given in Fig. III.1.a, while the comparison is shown in Fig. III.7.a.
- The circuit of RCA3040 is given in Fig. III.2.a, while the comparison is shown in Fig. III.7.b.
- The circuit of HVRef is given in Fig. III.3.a, while the comparison is shown in Fig. III.7.c.
- The circuit of Schmitt trigger is given in Fig.3.4.a, while the comparison is shown in Fig. 3.7.d.
- The circuit of OTA is given in Fig. III.5.a, while the comparison is shown in Fig. III.7.e.
- The circuit of 6sLA given in Fig. III.6.a, while the comparison is shown in Fig.III.7.f.

III.7. Conclusions

We obtained a new and global modification of Newton's method presented in (3.21) for solving system nonlinear equations for analog circuit with parameter (*m*) and parameter (*C*), and then the convergence orders of the iterative methods are always three. We can obtain efficient iterative methods different from any known schemes by chosen excellent parameter (*m*=0.5 or *m*=0.75 or *m*=0.9 for *C=1.75*) which is better performance than MW. In addition, MA performance is better than NR which is used in simulator SPICE. This method applied in circuits which have nonlinear elements (BJT, MOSFET, Diode…etc) to accelerate the solving and analyzing the circuits. All what was described previously was demonstrated and applied in PyAMS software in the next chapter.

Chapter IV

New Software for Modeling and Analysis of Analog Circuits Based on the Continuous Method

IV.1. Introduction

In this chapter, new software called *Python for Analog and Mixed Signals* (PyAMS) for creating analog elements and constructing circuits with analysis based on Python language. The objective is simplifying the modeling and the programming of analog elements and constructing circuits with analysis. The software depends on converting the analog elements (e.g. active elements as Diodes, BJTs, MOSFET, logical Gates… or passive elements as Resistors, Capacitors, Inductance…) to dependent sources which will be applied in the circuit, then; the circuit will be converted to the general equation which should be solved to be applied in the circuit analysis. The method of solve for analysis is based by continues method with Newton order three.

IV.2. What difference between SPICE and PyAMS

In the last years, a lot of software were created for analysis of analog circuits, all this software is based on the SPICE method (Simulation Program with Integrated Circuit Emphasis) [1][2], for example PSpice, LTSpice, Multisim, Tina…etc, and all these software are based on the original Spice3f5 created by the California University, Berkeley. The structure of the circuit with analysis in the simulator SPICE is written by the Netlist form (its textual description of the names of the analog elements with the position of attachment in the circuit and the analysis commands). The types of analysis used by SPICE to verify the correct circuit function are [2]:

- AC analysis (small-signal or frequency domain analysis)

- DC analysis (a sequence of nonlinear operating points calculated while sweeping an input voltage or current, or a circuit parameter)
- TR analysis (Transient analysis: time-domain large-signal solution)

The system of equations applied for analysis that SPICE adopted is written by the MNA (Modified Nodal Analysis) method. The objective of the MNA method is simplifying the structure of passive elements, current and voltage sources and active elements in a matrix equation which can be solved using linear methods as LU decomposition method [3]. The nonlinear elements are converted to linear elements using the Newton Raphson (NR) method. Although using the MNA method the convergence remains a problem, for that the SPICE adopts some techniques to avoid this problem as the *Stepping Source* method and the *Gmin* method [2]. VHDL-AMS and Varilog-AMS are languages used to modeling analog, digital and Mixed elements, in the last years are integrated with SPICE simulator to simply the building of new elements for application in circuits simulation [58][59][60].

Although the SPICE has a great utility, it has some drawbacks:

(i) The elements models are unchangeable and the user can not add a new element model, but it modeling by standard models [1].

(ii) The model who created by VHDL-AMS is not used directly in SPICE it compiling first for find errors of description then converting to Netlist form (example: Vhdl2Spice[58]) and finaly do SPICE analyzing, the same method for Verilog-AMS [59]. Sametimes, the model in form nonlinear element it has a problem in convergance [60].

(iii) The convergence remains a problem in the solution of the matrix equation representing the circuit, especially in large complex circuits.

(iv) The slowness of the circuit analysis due to the slowness on the equation solving [6].

This paper has the objective to overcome the mentioned drawbacks and brings some new features:

1. Proposal of new software called *Python for Analog and Mixed Signals* (PyAMS) for creating analog elements and constructing circuits with analysis based on Python language.
2. The user can create new models applicable directly in the circuits.
3. The software uses the *Continuous* method based on the *homotopy* equation to accelerate the solution finding.

4. The fastness in the circuit analysis.
5. The simplicity in creating sub-models for any compound element.
6. The choice in constructing circuits using either the textual mode (modeling language) or the graphical mode (CAD) description by schematic diagram.
7. Description of elements based on dependent sources.

The work is divided into two main parts: the first part is the theoretical background, which discusses how to convert an analog element to a dependent source and apply it in the circuit. The circuit will be converted to the general equation which should be solved to be applied in the circuit analysis. The second part is the programming of the theory part by creating new software for circuit analysis (circuit analysis in the one of the three modes DC, TR and AC). The software is based on solving the nonlinear circuit using our improved continuous method [54]. The method can be summarized as follows:

- Converting the elements of the circuit to dependent sources to simplify the generation of the *Global Equation* (2.34) of the circuit.
- Converting the *Global Equation* to the *Homotopy Equation* (the continuous equation).
- Finding the effective starting point in the *Homotopy Equation*.
- Using the *Predictor Corrector* to solve the *Homotopy Equation*.

IV.3. Theoretical background

This section puts the background to creating global system of equation of circuit based by dependent source and performance method for finding operating point and application in analyses, for that this part it composed in three parts: Converting analog element to dependent source. Generating global equation. A better method of solving to application in analyzing.

IV.3.1. The structure of analog element in PyAMS

Any analog element in PyAMS is presented as a voltage source *V* (Fig. IV.1.a) or current source *I* (Fig. IV.1.b) depending on the voltage in the nodes *v* and the current *i* across the voltage sources (Fig. IV.2) [55], where, $v=v_1,v_2,\ldots,v_n$ and $i= i_1,i_2,\ldots,i_m$. *V* and *I* are functions of the variables *v* and *I*, there are divided by direction:

- *V*: The output voltage signal.
- *I*: The output current signal.
- *v*: The input voltage signal.

- *i*: The input current signal.

(a) (b)

Fig. IV.1. Dependent sources: (a) voltage source, (b) current source.

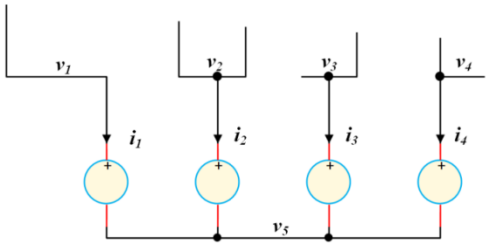


Fig. IV.2. Voltage v_1, v_2, v_3 and v_4 in the nodes and current i_1, i_2, i_3 and i_4 across of voltages source.

IV.3.2. The equation of the circuit in PyAMS

The aim of converting the circuit elements to their dependent sources is to simplify the generation of the global equation of the circuit by applying the two Kirchhoff's Laws:

1. The Kirchhoff Current Law (KCL) law for current [1]: the assumed of dependent current source and current across of dependent voltage source in the node equal zero (see example of Fig. II.8) and presented by flowing formula:

$$\sum_k d_I \; I_k(v,i) + \sum_p d_i \; i_p = 0 \qquad (4.1)$$

For d_I is sign (+1 or -1) of direction of dependent current I and d_i is sign of direction of current across in the dependent voltage source. k and p is number of courant in node.

2. The Kirchhoff Voltage Law (KVL) law for voltage [1]: dependent voltage source equal the difference of voltage between and presented by flowing formula:

$$v_a - v_b - V(v,i) = 0 \qquad (4.2)$$

By eq. (4.1) and (4.2) you can generate equation global of circuit how presented by this formula:

$$F(v,i) = \begin{cases} \sum_p d_i \; i_p + \sum_k d_I \; I_k(v,i) \\ v_a - v_b - V(v,i) \end{cases} \quad (4.3)$$

For

$$F(v,i) = 0 \quad (4.4)$$

Note: $(v,i) = (v_1, v_2, \ldots . v_n, i_1, i_2, \ldots, i_m)$.

F is the global function where, $F: \mathbb{R}^{n+m} \longrightarrow \mathbb{R}^{n+m}$, n represent the number of nodes in the circuit and m the number of voltage sources in the circuit, and we can represent it directly in form continues (4.6).

IV.3.3. Method of finding operating point

The best method of acceleration in solving of equation system (4.4), it used continuous method (Chap. II).

The continuous method it based on converting equation (4.4) to homotopy equation (4.5) and using predictor corrector solver.

The homotopy function $H(x,\lambda)$ is based on embedding a parameter λ into $F(v,i)$. As a result, an equation of new dimension:

$$H(v,i,\lambda) = 0 \quad (4.5)$$

Where, $\lambda \in \mathbb{R}$ and $H: \mathbb{R}^{n+m} \times \mathbb{R} \longrightarrow \mathbb{R}^{n+m}$. The parameter λ is called the continuation or homotopy parameter. At present, the most widely used method is the Newton homotopy mapping (NH):

$$H(\, v,i,\lambda) \; = \; F(v,i) \; + \; (1 \; - \lambda) \, F(0,0) \quad (4.6)$$

According to the homotopy mapping, the solution of the equation $F(0,0) = 0$ can be adopted to solve the homotopy equation $H(v,i,\lambda) = 0$. For any λ range from 0 to 1, if the homotopy equation solution (v,i,λ) exists, the corresponding curve of (v,i,λ) starts from $(0,0,0\,)$ and ends in the solution $(v^*, i^*, 1\,)$.

The final form of homotopy equation is:

$$H(v,i,\lambda) = \begin{cases} \sum_n d_n i + \sum_n d_n \big(I(v,i) + (\lambda - 1) I(0,0)\big) \\ v_a - v_b - \big(V(v,i) + (\lambda - 1) V(0,0)\big) \end{cases} \quad (4.7)$$

We obtain the step in the source of current:

$$I(v,i) + (\lambda - 1)I(0,0) \tag{4.8}$$

And step in the source of voltage:

$$V(v,i) + (\lambda - 1)V(0,0) \tag{4.9}$$

We apply continues solver of equation (4.8) and (4.9) by stepping λ from 0 to 1, we obtain the global step of out signal.

To solve the homotopy equation (4.5), a continuation solver has been used [55]. First, we adjust the homotopy parameter λ by choosing a step size $\Delta\lambda$ ($0<\Delta\lambda<1$), then, we use the predictor-corrector (Fig. IV.3). The predictor computes the approximation of solution by computing a point along the tangent line to the homotopy path at the point (v_0, i_0, λ_0) using the following formula:

$$\begin{pmatrix} v_1^0 \\ i_1^0 \end{pmatrix} = \begin{pmatrix} v_0 \\ i_0 \end{pmatrix} + \Delta\lambda \begin{pmatrix} \dot{v_0} \\ \dot{i_0} \end{pmatrix} \tag{4.10}$$

Where,

$$\begin{pmatrix} \dot{v_0} \\ \dot{i_0} \end{pmatrix} = \begin{pmatrix} \frac{\partial v_0}{\partial \lambda} \\ \frac{\partial i_0}{\partial \lambda} \end{pmatrix} \tag{4.11}$$

After that, the solution (v_1, i_1) of $H(v_1, i_1, \lambda_1)$ at $\lambda_1=\lambda_0+\Delta\lambda$ can be computed by the corrector function using NR iteration:

$$\begin{cases} J\left(v_1^k, i_1^k, \lambda_1\right)\begin{pmatrix} \Delta v_1^k \\ \Delta i_1^k \end{pmatrix} = -H\left(v_1^k, i_1^k, \lambda_1\right) \\ \begin{pmatrix} v_1^{k+1} \\ i_1^{k+1} \end{pmatrix} = \begin{pmatrix} v_1^k \\ i_1^k \end{pmatrix} + \begin{pmatrix} \Delta v_1^k \\ \Delta i_1^k \end{pmatrix} \end{cases} \tag{4.12}$$

Where, $J = \frac{\partial H}{\partial v\, \partial i}$ is the Jacobien function of *H*, *"k"* is the iteration index and (v_1^0, i_1^0) is the initial approximation or the initial of gauss calculated by the predictor Eq. (4.10).

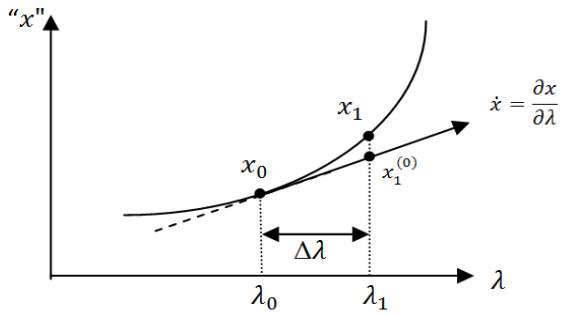


Fig. IV.3. Predictor-corrector method ($x = v \cup i$)

The equation (4.12) it's Newton order two to acceleration on using new families of Newton order three (3.21) depending on the parameters *m* and *C*.

$$\begin{cases} \begin{pmatrix} v_1^a \\ i_1^a \end{pmatrix} = \begin{pmatrix} v_1^k \\ i_1^k \end{pmatrix} - m\frac{H(v_1^k, i_1^k, \lambda_1)}{J(v_1^k, i_1^k, \lambda_1)} \quad m \neq 0 \\ \begin{pmatrix} v_1^b \\ i_1^b \end{pmatrix} = \begin{pmatrix} v_1^k \\ i_1^k \end{pmatrix} - A\frac{H(v_1^k, i_1^k, \lambda_1)}{J(v_1^k, i_1^k, \lambda_1)} - B\frac{H(v_1^a, i_1^a, \lambda_1)}{J(v_1^k, i_1^k, \lambda_1)} \\ \begin{pmatrix} v_1^{k+1} \\ i_1^{k+1} \end{pmatrix} = \begin{pmatrix} v_1^k \\ i_1^k \end{pmatrix} - A\frac{H(v_1^k, i_1^k, \lambda_1)}{J(v_1^k, i_1^k, \lambda_1)} - B\frac{H(v_1^a, i_1^a, \lambda_1)}{J(v_1^k, i_1^k, \lambda_1)} - C\frac{H(v_1^b, i_1^b, \lambda_1)}{J(v_1^k, i_1^k, \lambda_1)} \end{cases} \tag{4.13}$$

For

$$\begin{cases} A = \frac{1}{m^2} \\ B = \frac{1}{m^2}(m-1) + 1 \\ C \in \mathbb{R} - \{1\} \end{cases} \tag{4.14}$$

Remark

5. For *m*=1 and *C=0* represent the *MA* formula [61].
6. For *m*=1 and *C=1* represent the *MW* formula [63].
7. For best method we use *m*=0.5 or *m*=0.75 or *m*=0.9 for *C=1.75*.

IV.3.4. The algorithms of Analyses of circuit by PyAMS

In this part we will discuss the application of continues solver with corrector based on the Newton order three (4.13) to analyze circuits in four modes: OP, DC, TR and AC analyses.

a. The OP analyses

The algorithm of finding operating point by PyAMS for circuit with element presented by dependent source is presented in Fig. IV.4: first start by construct homotopy equation and after that initial voltage in the node, current across in the source and homotopy parameter by zero value and after that using continues algorithm, finally, we get operating point in circuit presented the value of voltages in the nodes and currents across in the sources. Note the derivation by time in this mode is zero. This algorithm is using in these analyses: DC, TR and AC.

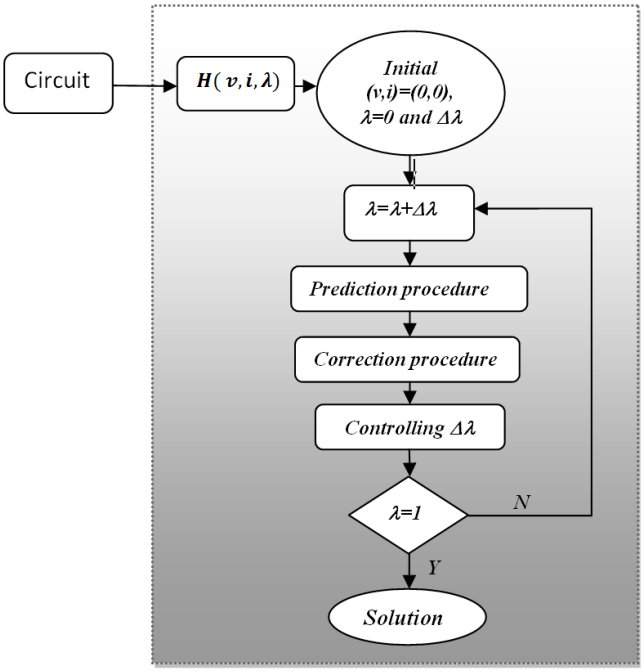


Fig. IV.4. Algorithm OP analyses.

b. DC analysis

The DC analysis is based on finding operating point algorithm (Fig. IV.5), this analysis used to get the function of circuit by variation one of parameters in circuit (ex. resistor value, current value, voltage value, temperature value…).

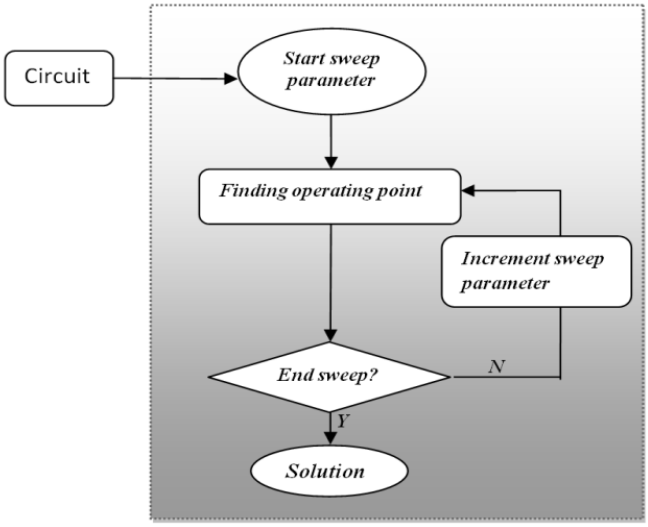


Fig. IV.5. Algorithme DC analyses.

c. Transient Analysis

The TR analysis is to study the function circuit by time. The algorithm of analysis TR by PyAMS is presented in Fig. IV.6:

- Start to find operating point for time $t_0=0$.
- Stepping in the time $t_j=t_{j-1}+\Delta t$, (the Δt is called the step size of time).
- Using the Newton order three for solving (4.13).

For the element, it has derivative function presented in this form:

$$\frac{\partial v}{\partial t} = f(v) \tag{4.15}$$

It used numerical integration method to convert nonlinear differential equations to nonlinear algebraic equation to simplify solving by Newton order three. There are three simple numerical integration methods; it the same in the spice simulator [3]:

- Forwared Euler:

$$v_j = v_{j-1} + \Delta t\, f(v_{j-1}) \tag{4.16}$$

- Back ward Euler:

$$v_j = v_{j-1} + \Delta t\, f(v_j) \tag{4.17}$$

- Trapezoidal:

$$v_j = v_{j-1} + \frac{\Delta t}{2}\left(f\left(v_{j-1}\right) + f\left(v_j\right)\right) \quad (4.18)$$

For Δt is step time, it is minimized when have problem of convergence by using controlling the step time method, it is the same algorithm used in the SPICE method [2].

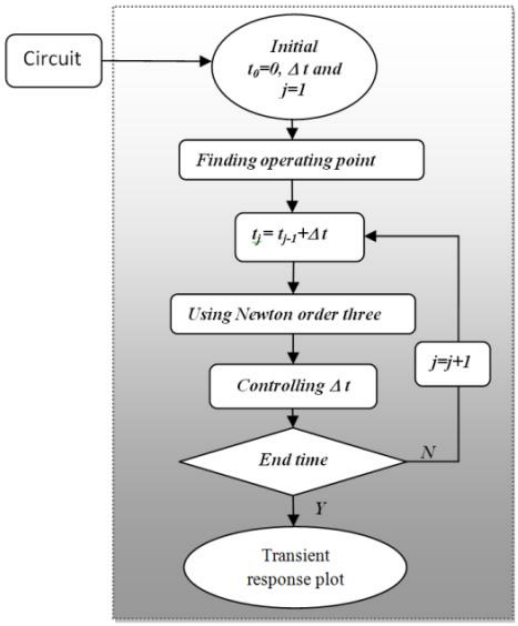


Fig. IV.6. Algorithm nonlinear TR transeit analysis.

d. Small Signal AC Analysis

The AC analysis is the work of circuit in frequency domain, the algorithm of AC analysis is presented in flowchart of Fig. IV.7: First, finding operating point. Second, convert the dependent source to resistive element and convert derivative of voltage (4.19) or current (4.20) by time to voltage or current multiplied by frequencies depending on ω:

$$\frac{\partial v}{\partial t} \equiv j\omega v \quad (4.19)$$

$$\frac{\partial i}{\partial t} \equiv j\omega i \quad (4.20)$$

For j is a complex number equal to $\sqrt{-1}$. The system equation of circuit is linear, for solving it, we apply linear method (LU decomposition).

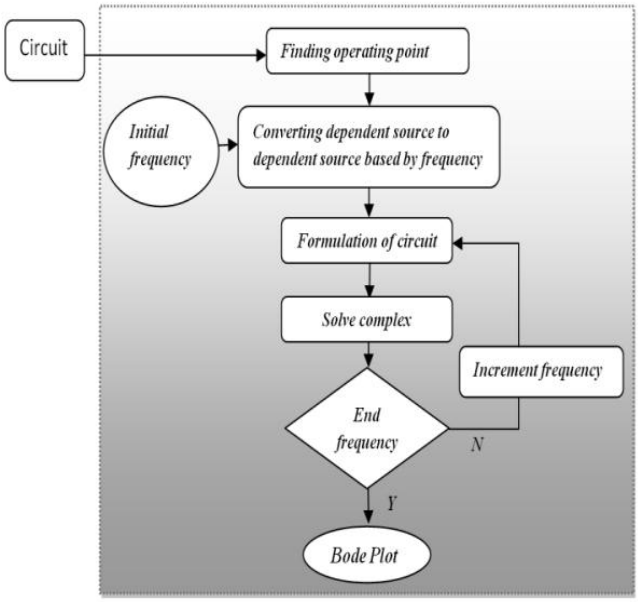


Fig. IV.7. Algorithm small signall AC analysis.

IV.4. The modeling of analog elements and simulation circuit by PyAMS

PyAMS is a software programmed using Python language and Delphi XE8 IDE. PyAMS used to modeling of analog element, construct and analyzing circuits in the following modes: DC, TR and AC.

IV.4.1. Modeling of analog elements by PyAMS

The modeling of analog elements by PyAMS software is based on: the name of model, the ports, the signals (voltage or current) and parameters.

The form of any model by PyAMS is presented in the program as follow (program IV.1). First, declaration of library. After that, declaration name of element by using class by python language. The class decomposed in two function (by using instruction def): __init__ and analog.

Program IV.1: Structure of modeling any analog element by PyAMS

```
1 #  Declaration of  library
2 from PyAMS import Signal
3 #  Name of new element by using class
4 class Name_of_element:
5       #  Declared the names of ports
6       def __init__(self, Port1, Port2,.....,Portn):
7       # Declaration of signals(voltages or currents)
8       # Declaration and initialization parameters.
9       def  analog (self):
10      # Relation between signals and parameters.
```

Initial function "__init__" used to declared name of ports in element between brackets and after that in this function we declare and initialize the signal and parameter.

For signal declaration, it is presented by this instruction: *Signal (Direction, Type, $Port_a$,$Port_b$)*

There are four type of signal decomposed by direction and type:

- The output current signal (dependent current source Fig. IV.8.a) it is presented by this method of declaration: *I= Signal ('current', 'out', $Port_a$,$Port_b$)*
- The output voltage signal (dependent voltage source Fig. IV.8.b) it is presented by this method of declaration: *V= Signal ('voltage', 'out', $Port_a$,$Port_b$)*
- The input current signal (current across voltage source Fig. IV.9.a) it is presented by this method of declaration: *i= Signal ('current', 'in', $Port_a$,$Port_b$)*
- The input voltage signal (voltage in the node Fig. IV.9.b) it is presented by this method of declaration: *v= Signal ('voltage', 'out', $Port_a$)*
- The input voltage signal (voltage in the difference between two nodes Fig. IV.9.c) it is presented by this method of declaration: *vd= Signal ('voltage', 'out', $Port_a$,$Port_b$)*

Fig. IV.8. Direction of out signal for voltage and current in PyAMS Language.

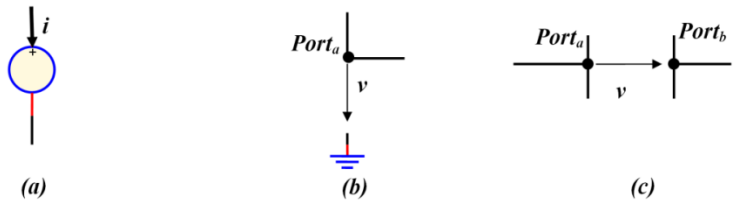


Fig. IV.9. Direction of in signal for voltage and current in PyAMS Language.

Analog function is used for giving relation between output of signals (dependent source *V* and *I)* with input of signals (voltage node and courante across of voltage source *v* and *i*) by using:

- Operation math (+,-,/) or logic(==,!=,>=,<=).
- Statement condition (if, ifel, else) or boucle (while, for).
- Function example math function (sin, cos, exp,….).

- Standard function

IV.4.2. Examples of models by PyAMS

In this part we discuss the creation of model passive elements (Resistor and Capacitor), active elements (Diode, BJT, MOSFET and OpAmp), transformer nonlinear, port logic (AND, OR, NAND, NOR and NOT) and source.

IV.4.2.1. Resistor

A resistor can be represented by a current source *I* (Fig. IV.10), which it is dependent on the voltage between the nodes *p* and *n* and resistor value, is expressed by law ohm:

$$I(v_p, v_n) = \frac{v}{R} \tag{4.21}$$

For

$$v = v_p - v_n \tag{4.22}$$

Where, *R* is the resistance value.

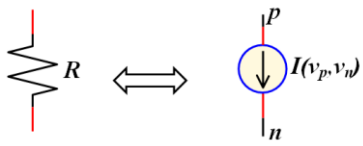


Fig. IV.10. The equivalent of resisstor by dependent current source.

The model of resistor is presented in program IV.2. it initialize by two type of signal: the first signal is presented by current *I* across at port *p* to port *n* and the second signal is presented by difference of tension *v* between ports *p* and *n*.

The analog function is operation between output current signal *I* and input voltage signal *v* divide by parameter *R* to construct law ohm. Note the "self" in program is the name of model element and "+=" it operation for signal by PyAMS languge it equivalent to equal operator "=".

Program IV.2: Modeling of Resistor by PyAMS.

```
1 from PyAMS import Signal
2
3 class Resistor:
4   def __init__(self, p, n):
5     #Signals------------------------------------------------
6     self.v = Signal('in','voltage',p,n)
7     self.I = Signal('out','current',p,n)
8     #Parameters---------------------------------------------
9     self.R=100.0
10
11  def analog(self):
12    self.I+=self.v/self.R        # I(v)=v/R
```

The resistor can be represented too by a voltage source V (Fig. IV.11) dependent on the voltage between the node p and n and the current i across the source V, it is expressed by:

$$V(i) = Ri \tag{4.23}$$

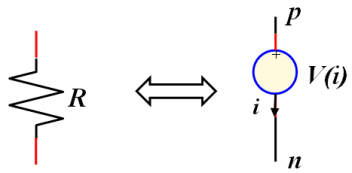


Fig. IV.11. The equivalent of resisstor by dependent voltage source.

The model of resistor by dependent voltage source is presented in program IV.3. It initialize by two type of signal: the first signal is presented by current i across at port p to port n and the second signal is presented by source tension or signal type out V.

Program IV.3: Modeling of Resistor by PyAMS (equivalent to voltage source).

```
1 from PyAMS import Signal
2
3 class Resistor:
4     def __init__(self, p, n):
5       #Signals------------------------------------------------
6       self.i = Signal('in','current',p,n)
7       self.V = Signal('out ','voltage',p,n)
8       #Parameters---------------------------------------------
9       self.R=100.0
10
11    def analog(self):
12      self.V+=self.i*self.R        # V(i)=i*R
```

IV.4.2.2. Capacitor

The capacitor is elements of storage of energy, and they are dependent on time. The capacitor is equivalent to a current source I dependent on the voltage in the nodes p and n

(Fig. IV.12), and it is expressed by:

$$I(v_p, v_n) = C \frac{\partial v}{\partial t} \tag{4.24}$$

For

$$v = v_p - v_n \tag{4.25}$$

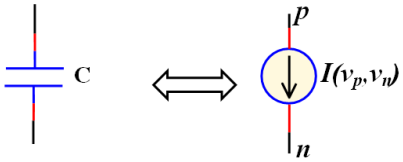


Fig. IV.12. The equivalent of capacitor by dependent current source.

Capacitor model by PyAMS is presented in program IV.4 is implemented in a similar way to resistors model (program IV.2). However, these devices have dependencies on time. In this case, the right hand side includes a derivative with respect to time in analog function. This is implemented with the *ddt* operator from PyAMS library.

Program IV.4: Modeling of Capacitor by PyAMS.

```
1 from PyAMS import Signal
2 from PyAMS import ddt
3
4 class Capacitor:
5       def __init__(self, p, n):
6            #Signals---------------------------------------
7            self.v = Signal('in',' voltage',p,n)
8            self.I = Signal('out','current',p,n)
9           #Parameters-------------------------------------
10           self.C= 1.0e-3
11
12       def  analog(self):
13           self.I+=self.C*ddt(self.v)
```

IV.4.2.3. Diode

The diode is nonlinear element and semiconductor device, can be represented by its equivalent nonlinear current source *I* dependent by the voltage in the nodes *p* and *n* Fig. IV.13, and it is given by:

$$I(v_p, v_n) = Is\ (exp\left(\frac{v}{Vt}\right) - 1) \tag{4.26}$$

For

$$v = v_p - v_n \tag{4.27}$$

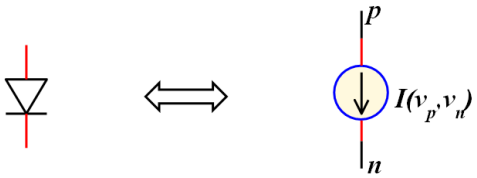


Fig. IV.13. The equivalent of diode by dependent voltage source.

The model of diode by PyAMS is presented in program 4.5, it has two type of signal: the first signal is current *I* across at *p* to *n* and the second signal is difference of voltage *v* between of ports *n* and *p*, those two type of signal declared in initial function.

The mathematical equation between the *v* voltage and the current *I* which passes through it is as follows in analog function by using equation (4.26). With using by *Vt* parameter is the thermal voltage and *Is* is the reverse bias saturation current (or scale current).

Program IV.5: Modeling of Diode by PyAMS.

```
1 from PyAMS import Signal
2 from math import exp
3
4 class Diode:
5   def __init__(self, p, n):
6     #Signals-------------------------------------
7     self.v = Signal('in',' voltage ',p,n)
8     self.I = Signal('out', 'current',p,n)
9     #Parameters----------------------------------
10    self.Is=1.0e-12
11    self.Vt=0.025
12
13  def analog(self):
14    self.I+=self.Is*(exp(self.v/self.Vt)-1)
```

IV.4.2.4. Bipolar junction transistor BJT

In this example we deal with the BJT which is represented by The Ebers-Moll model [2] Fig. IV.14-a, for current I_R and I_F we have:

$$I_R = I_{se}\left(exp\left(\frac{V_{be}}{V_{te}}\right) - 1\right) \tag{4.28}$$

$$I_F = I_{sc}\left(exp\left(\frac{V_{bc}}{V_{tc}}\right) - 1\right) \tag{4.29}$$

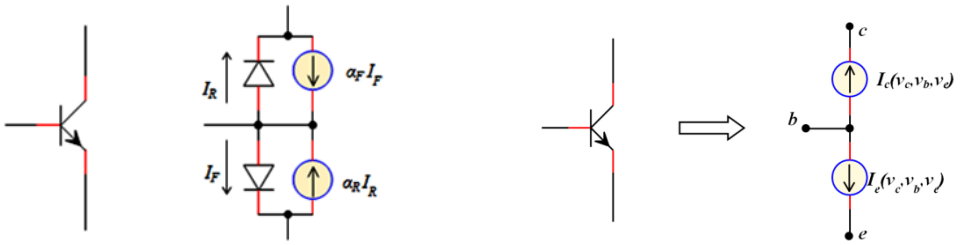
The equivalent of the transistor can be represented by two current sources (Fig. 14-b) I_c and I_e and it is dependent on the voltage in the three nodes *c*, *b* and *e*:

$$I_c(v_c, v_b, v_e) = I_{sc}\left(exp\left(\frac{v_{bc}}{V_{tc}}\right) - 1\right) - \alpha_R\, I_{se}\left(exp\left(\frac{v_{be}}{V_{te}}\right) - 1\right) \quad (4.30)$$

$$I_e(v_c, v_b, v_e) = -\alpha_F I_{sc}\left(exp\left(\frac{v_{bc}}{V_{tc}}\right) - 1\right) - I_{se}\left(exp\left(\frac{v_{be}}{V_{te}}\right) - 1\right) \quad (4.31)$$

For

$$\begin{cases} v_{bc} = v_b - v_c \\ v_{be} = v_b - v_e \end{cases} \quad (4.32)$$



(a) (b)

Fig. IV.14.(a)The Ebers-Moll model for an npn BJT. (b)The equivalent of BJT by dependent current source.

Program IV.6: Modeling of Ebers-Moll transistor by PyAMS.

```
1 from PyAMS import Signal
2 from math import exp
3
4 class NPN:
5       def __init__(self,c,b,e):
6            #Signals----------------------------------------
7            self.vbe=Signal('in','voltage',b,e)
8            self.vbc=Signal('in','voltage',b,c)
9            self.Ic=Signal('out', 'current',b,c)
10           self.Ie=Signal('out', 'current',b,e)
11           #parameters-------------------------------------
12           self.af=0.2
13           self.ar=0.9
14           self.Isc=1.0e-16
15           self.Ise=1.0e-16
16           self.Vte=0.025
17           self.Vtc=0.025
18
19      def analog(self):
20            Icc=self.Ise*(exp(self.vbe/self.Vte)-1)
21            Ice=self.Isc*(exp(self.vbc/self.Vtc)-1)
22            af=self.af
23            ar=self.ar
24            self.Ic+= Icc-ar*Ice
25            self.Ie+= -af*Icc-Ice
```

BJT of type npn Ebers-Moll transistor can be represented by two current sources (Fig. IV.14) I_c and I_e and it is dependent on the voltage in the three nodes or ports *c*, *b* and *e*, see the equation (4.30) and (4.31). The model of npn by PyAMS is presented in program IV.6, it has

two out current signal *Ic* and *Ie* and two in voltage signal *vbc* and *vbe*. In analog function is presented of operation between signals by using equation (4.30) and (4.31).

IV.4.2.5. Complete model of BJT

This type of model in Fig. IV.15 is named by Gummel-Poon without substrate [115] the program IV.7, it is presented by relation between signal of type currents *Ibe*, *Ice*, and *Ict* dependent with the signal type voltages *Vbe* and *Vbc*. For the parameters and the signals is presented in initial function and for expression is presented in analog function.

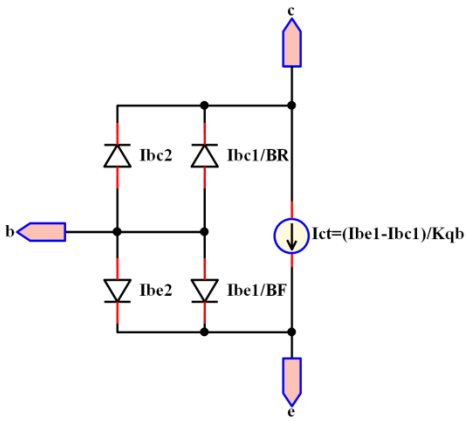


Fig. IV.15. Bipolar Transistor npn Model type Gummel-Poon without substrate.

Program IV.7: Modeling of Gummel-Poon transistor by PyAMS.

```
from PyAMS import Signal
from PyAMS import limexp

class nBJT:
      def __init__(self,c,b,e):
           #Signals----------------------------------------
           self.vbe=Signal('in',' voltage',b,e)
           self.vbc=Signal('in','voltage',b,c)
           self.Ice=Signal('out','current ',c,e)
           self.Ibe=Signal('out','current',b,e)
          self.Ict=Signal('out','current',c,e)
           #Parameters---------------------------------
           self.NF=1.0
           self.NE=1.5
           self.NR=1.3
           self.NC=2.0
           self.Iss=1.0e-16
           self.Ise=1.0e-16
           self.Isc=1.0e-15
           self.area=1.0
           self.BR=3.5
           self.BF=100.0
           self.VAR=30
           self.VAF=40
           self.IKF=0.04
           self.IKR=0.001
```

```
28      def analog(self):
29            Vt=0.025
30            Ibe1=self.Iss*(explim(self.vbe/(self.NF*Vt))-1)
31            Ibe2=self.Ise*(explim(self.vbe/(self.NE*Vt))-1)
32            Ibc1=self.Iss*(explim(self.vbc/(self.NR*Vt))-1)
33            Ibc2=self.Isc*(explim(self.vbc/(self.NC*Vt))-1)
34            Kq1= 1/(1-(self.vbc/self.VAF) - (self.vbe/self.VAR))
35            Kq2= (Ibe1/self.IKF) + (Ibc1/self.IKR)
36            Kqb=Kq1*(1+sqrt(1+4*Kq2))/2
37            self.Ibe+=self.area*((Ibe1/self.BF) + Ibe2 + (Ibc1/self.BR) + Ibc2)
38            self.Ice+=self.area*((Ibe1/Kqb) -(Ibc1/Kqb)-(Ibc1/self.BR)- Ibc2)
39            self.Ict+=(Ibe1-Ibc1)/Kqb
```

In Fig. IV.16. is Gummel-Poon with substrate [1], To program this model (program IV.8) based by constructing new nodes *cn*,*bn* and *en* by using instruction *NewNode* and attached with sub model of resistors (*Ra*, *Rb*, *Rc*), variable capacitors (*Cjs*, *Cjc*, *Cje*) and sample Gummel-Poon (nBJT). In the node *s* and node *cn* substrate current based on out signal dependent by voltage *Vcs* in the nodes *cn* and *s*.

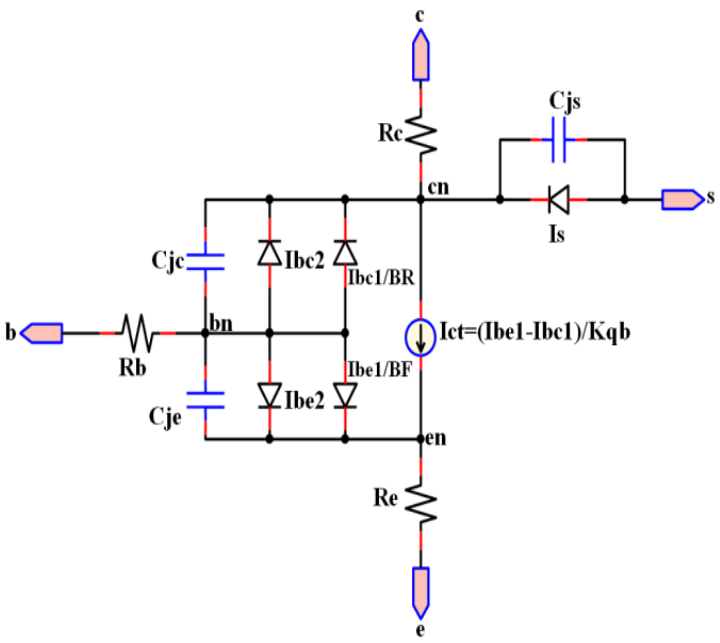


Fig. IV.16. Model complete of Gummel-Poon BJT.

Program IV.8: Modeling of complete Gummel-Poon transistor by PyAMS.

```
from PyAMS import Signal
from PyAMS import limexp

# import Model-------------------------------------
from nBJT import nBJT
from Resistor import Resistor
from Capacitor import Capacitor

class nBJTS:
      def __init__(self,c,b,e):
           #New Nodes---------------------------------
           cn=NewNode()
           bn=NewNode()
           en=NewNode()
           #Add Models--------------------------------
           self.Rc=Resistor(c,cn)
           self.Rb=Resistor(b,bn)
           self.Re=Resistor(e,en)
           self.Cjc=Capacitor(bn,cn)
           self.Cje=Capacitor(bn,en)
           self.Cjs=Capacitor(s,cn)
           self.NPN=nBJT(cn,bn,en)
           #Signals-----------------------------------------
           self.Vsc=Signal('in','voltage',s,c)
           self.Is=Signal('out','current',s,c)
           #Parameters--------------------------------
           self.Ns=1.0
           self.Iss=1.0e-16
           self.area=1.0

      def analog(self):
            Vt=0.025
            Is=self.Iss*(explim(self.Vsc/(self.Ns*Vt))-1)

      def SubBlock(self):
           #Return by sub circuit----------------------
           return [self.Rc,self.Re,self.Rb,self.Cjs,self.Cje,self.Cjc,self.NPN]
```

Note: The function SubBlock can be used to confirm the list names of elements linked to new models.

IV.4.2.6. MOSFET

The equivalent of the transistor MOSFET (N-Chanal), can be represented by current source (Fig. VI.17) I_d and it is dependent on the voltage in the three nodes v_g,v_s and v_d:

$$I_d(v_d, v_s, v_g)$$

$$= \begin{cases} 0 & v_{gs} \le V_T \\ K_P \frac{W}{L}\left((v_{gs}-V_T)v_{ds} - \frac{{v_{ds}}^2}{2}\right)(1 + Lambda\ v_{ds}) & v_{ds} \le v_{gs} - V_T \\ K_P \frac{W}{L}(v_{gs}-V_T)^2(1 + Lambda\ v_{ds}) & v_{ds} \ge v_{gs} - V_T \end{cases} \quad (4.33)$$

For

$$\begin{cases} v_{ds} = v_d - v_s \\ v_{gs} = v_g - v_s \end{cases} \quad (4.34)$$

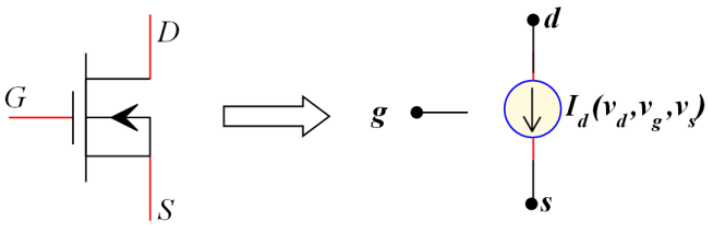


Fig. IV.17. The equivalent of NMOS by dependent current source

The program of this model is presented in program IV.9; the model is based by out current signal *Id* and two in voltage signal *vgs* and *vds*. In analog function is presented of operation between signals by using equation (4.33).

Program IV.9: Modeling of NMOS by PyAMS.

```
from PyAMS import Signal

class NMOS:
      def __init__(self,d,g,s):

            #Signals-----------------------------------
            self.vgs=Signal('in','voltage',g,s)
            self.vds=Signal('in','voltage',d,s)
            self.Id=Signal('out','current',d,s)
            #Parameters--------------------------------
            self.Kp=200e-6
            self.W=100.0e-6
            self.L=100.0e-6
            self.Vt=0.5
            self.lambd=0.000

      def analog(self):
          K=self.Kp*self.W/self.L
          if self.vgs <= self.Vt:
             Id+=0.0
          elif (self.vgs-Vt)<self.vds:
                self.Id+= K*(self.vgs-self.vt)*(self.vgs-self.vt)*(1+(self.lambd*self.vds))/2
          else:
                self.Id+= K*((self.vgs-self.vt)-(self.vds/2))*(1+(self.lambd*self.Vds))
                        *self.vds
```

IV.4.2.7. Voltage source

The program IV.10 presented of voltage source generated sinusoidal function between the nodes *a* and *b* with three parameters: *Va* the amplitude of the generated voltage in volts (V), *Ph* the phase in radine. *Fr* is the frequency of source in hertz (Hz). The function *RealTime* is used for return the current simulation time.

Program IV.10: Modeling of source voltage sinusoidal by PyAMS.

```
from PyAMS import Signal
from PyAMS import RealTime
from math import pi, sin

class SourceVsin:
   def __init__(self, a, b):
      # Signals------------------------------------------------------
      self.Vsin = Signal('in','voltage',a,b)
      #Parameters---------------------------------------------------
      self.Fr=100.0
      self.Va=15.0
      self.Ph=0.0

   def analog(self):
      self.Vsin+=self.Va*sin(pi*2.0*self.Fr*RealTime()+self.Ph)
```

IV.4.2.8. Logic gates

There are five types of logic gates (AND, OR, NOT, NAND, NOR). In program IV.11 simple example of gate logic type NAND of CMOS family with two types of in voltage signal *vina* and *vinb*, and one out voltage signal *Vout*, with parameters *(Vhl*, *Vlh*, *Vh* and *Vl)* presented the interval of work of NAND gate.

Program IV.11: Modeling of NAND Gate by PyAMS.

```
from PyAMS import Signal
class  NAND:
  def __init__(self,a,b,c):
        # Signals ----------------------------------
         self.vina=Signal('in','voltage',a)
         self.vinb=Signal('in','voltage',b)
         self.Vout=Signal('out','voltage',c)
        #Parameters-----------------------------
         self.Vhl=2.5
         self.Vlh=0.8
         self.Vh=5.0
         self.Vl=0.0

  def analog(self):
      if (self.vina>self.Vhl) and (self.vinb>self.Vhl):
               self.Vout+=Vl
     elif (self.vina<self.Vlh) and (self.vinb<self.Vlh):
               self.Vout+=Vh
```

IV.4.3. The functions used by PyAMS language.

The class is an instruction used for creating object in python language and used for creating new model of analog element organized by functions or events: function for declaration of signals, parameters and elements, function for operation, function for display or command or aiding in analyses. The subject of organized model by functions is to have good

description. The key of functions in class of element by PyAMS is described in the table below:

Table IV.1 The functions in instruction of class with application.

Function	Application
__init__(self, ports)	Is used for declaring initial parameters and signals.
analog(self)	Is used for operating between signals and parameters by using operation logic or math or using statement condition or statement loop.
Start(self)	Is used for modifying parameters or initializing signal value when starting analysis.
Stop(self)	Is used to get finale result.
Memory(self)	Is used to memory the result by using list to stock result and when stop analyzing, we notice by Stop (self) event.
Output(self)	Is used for visualization value or messages in analysis, for example use of print instruction or display in animation elements.
Input(self)	It is for command or variant in parameters or signal value in analysis, for example key and switch.
SubBlock(self)	Is used for return elements in circuit which added to model (example BJT)
Temperature(self)	Is used for analysis in mode DC for temperature parameter of one element in circuit.

For functions used to operate or return value or display, it is described in the table below:

Table IV.2 The functions for operation.

Function	Application
Signal()	It is signal function used for constructing dependent source or voltage in the node or current across of source.
RealTime()	The simulation time function provides an access to current simulation time or system function returns a value of time as a real number.
Temperature()	Returns the current simulation temperature in Kelvin. It is presented the nominal temperature *Tnom* in option of simulation.
Vt()	Vt[(*temperature_expression*)] Returns the thermal voltage at *temperature expression*. If *temperature expression* is not supplied, the value at the

	current simulation temperature will be returned.The thermal voltage is defined as: *K.T/q* Where, *K* is poltzmann's constant, *T* is temperature (defined by *temperature expression*) and *q* is the charge on an electron. K=1.3806226e-23 q =1.6021918e-19
ACSim()	ACSim[(*mag, phase*)]: The AC stimulus function returns 0 during large-signal analyses (such as DC and transient) as well as on all small-signal analyses the source becomes active and models a source with magnitude *mag* and phase *phase*: $Mag\ e^{-j\ phase}$ The default magnitude is 1 and the default phase is 0. Phase is given in radians.
ddt()	ddt[(*Signal*)]: it is used for calculating derivative variant of signal by time.
idt()	idt[(*Signal*)] : is used for calculating integration of signal by time.
limexp()	limexp([*Expression*]): Limitation in exponential function.
NewNode()	Construct new node in circuit.
ItDC()	Its function is to return the true value when analysis by mode DC.
ItAC()	Its function is to return the true value when analysis by mode AC.
ItTR()	Its function is to return the true value when analysis by mode TR.
Display()	Display [(*String*)]: it is used to display message in output of software.
DescParam()	DescParam[(*Name of parameter, Description of parameter*)]: is used for description of parameter.
HelpFile()	HelpFile[(*File Name*)]: is used for attaching the description file with model of element.

IV.4.4. Create circuit by PyAMS

The creation of circuit in PyAMS is based on attachment between models of analog element in the ports. This is by two method: textual method or by schema method.

IV.4.4.1 Create circuit by method of textual

The creation of circuit by method textual is an attachment between models of analog elements by python language. First, importing the names of models using in circuit, second, declare name or reference of model and change name of port in model by name of position of attachment in circuit, finally, change parameter.

For example the program IV.12 describe RLC circuit by textual method, the first four lines are the importing name of models used in circuits, after that, the four lines are references of model presented the element in circuit (Fig. IV.18): *V1*, *R1*, *L1* and *C1* presented source Vdc, Resistor, Inductance and Capacitor. Those models are attached by serial in nodes (‘*n1*’, ‘*n2*’, ‘*n3*’ and *gnd*), for *gnd* or ‘0’ represent the reference of circuit. After that, the changing parameter of element which declared in function “__init__” for model.

Other method, by creating model of circuit (program IV.13), it construct circuit with same method of subblock.

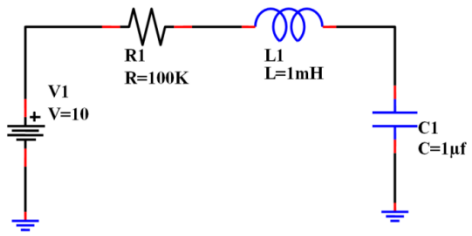


Fig. IV.18. Circuit RLC.

Program IV.12: Description textual of circuit RLC.

```
# Import models---------------------------------------
from Resistor import Resistor
from SourceVdc import SourceVdc
from Inductance import Inductance
from Capacitor import Capacitor
from PyAMS  import gnd

# Attachment between elements-----------------------
V1=SourceVdc('n1',gnd)
R1=Resistor('n1', 'n2')
L1=Inductance('n2', n3)
C1=Capacitor('n3',gnd)

# modified parameters-------------------------------
R1.R=100e3
L1.L=1e-3
C1.C=1e-6

# construit circuit by list of ‘cir’.---------------
cir=[V1,R1,L1,C1]
```

Program IV.13: Create circuit RLC by model method.

```
# Import models----------------------------------------
from Resistor import Resistor
from SourceVdc import SourceVdc
from Inductance import Inductance
from Capacitor import Capacitor
from  PyAMS  import gnd

#Create model of circuit RLC---------------------------
class Circuit_RLC():
    def  __init__(self):
       # attachment between elements-------------------
       self.V1=SourceVdc('n1',gnd)
       self.R1=Resistor('n1', ' n2')
       self.L1=Inductance('n2', ' n3')
       self.C1=Capacitor('n3',gnd)
       # modified parameters---------------------------
       self.R1.R=4
       self.L1.L=2e-6
       self.C1.C=2e-9
```

The interface of description textual is shown in Fig. IV.19. It is programmed by Delphi XE8, and it is used to compile the structure textual by finding the errors in description of model, also find errors of attachment between elements by python language, after that, the interface analyze and execute the structure of circuit.

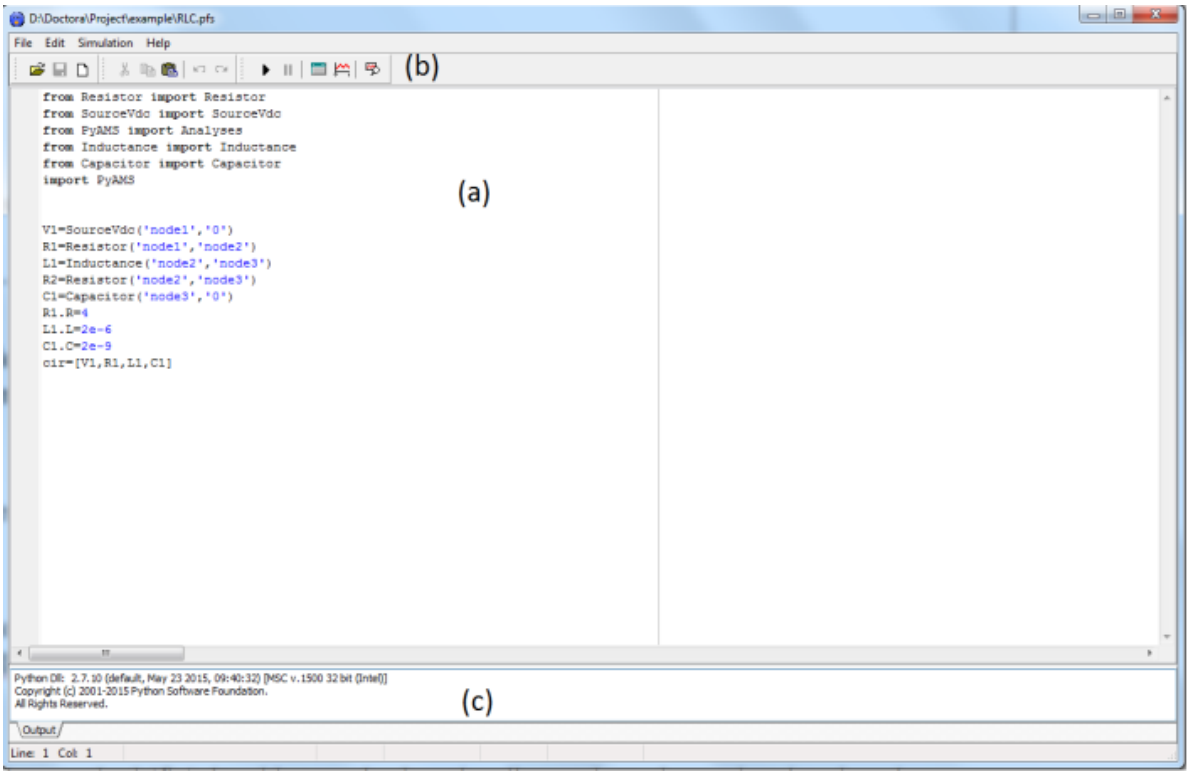


Fig. IV.19. The interface for design of circuit by textual method: (a) Editeur of texte. (b) Command of : editing, compilation and analyses. (c) Output messages.

IV.4.4.2 Create circuit by method of schema

The interface is used to design circuit by method of schema it is presented in Fig. VI.20. it is programmed by Delphi XE8. It is used for drawing circuit by adding parts in schematic editor and attachment between parts by wires.

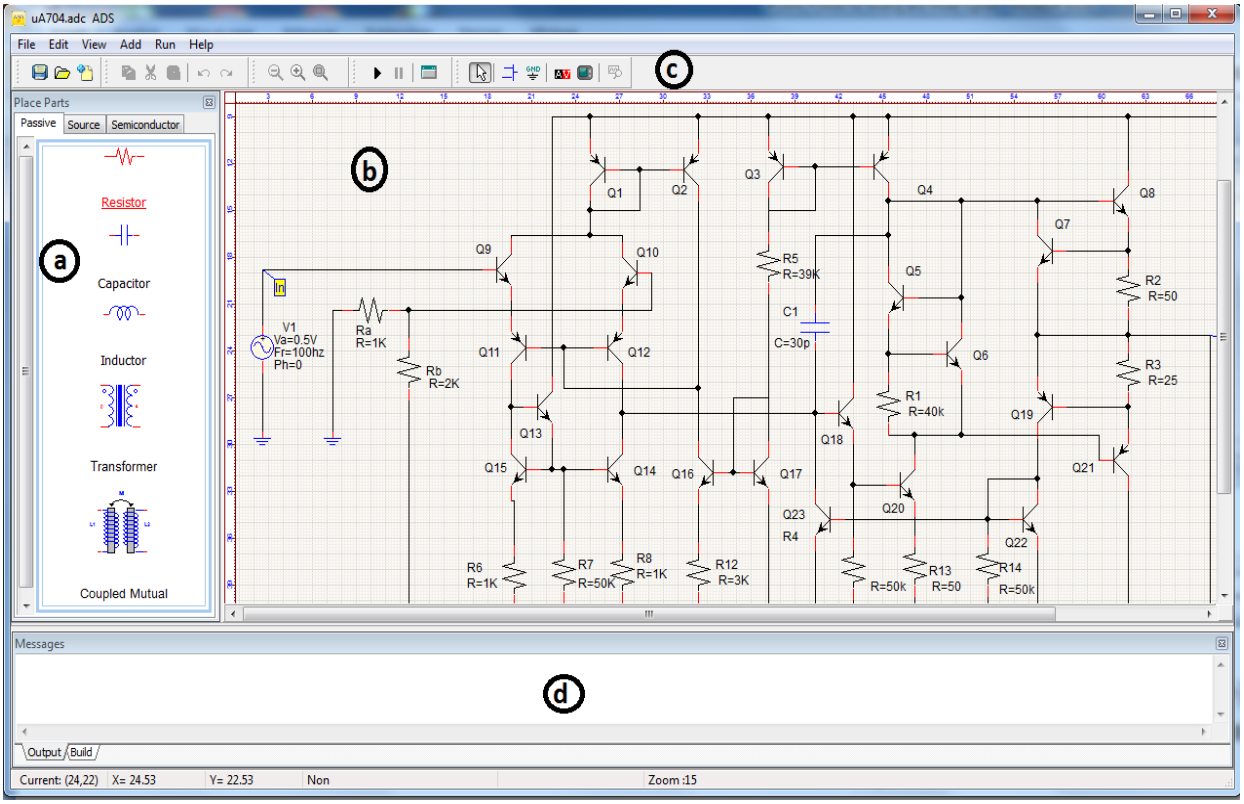
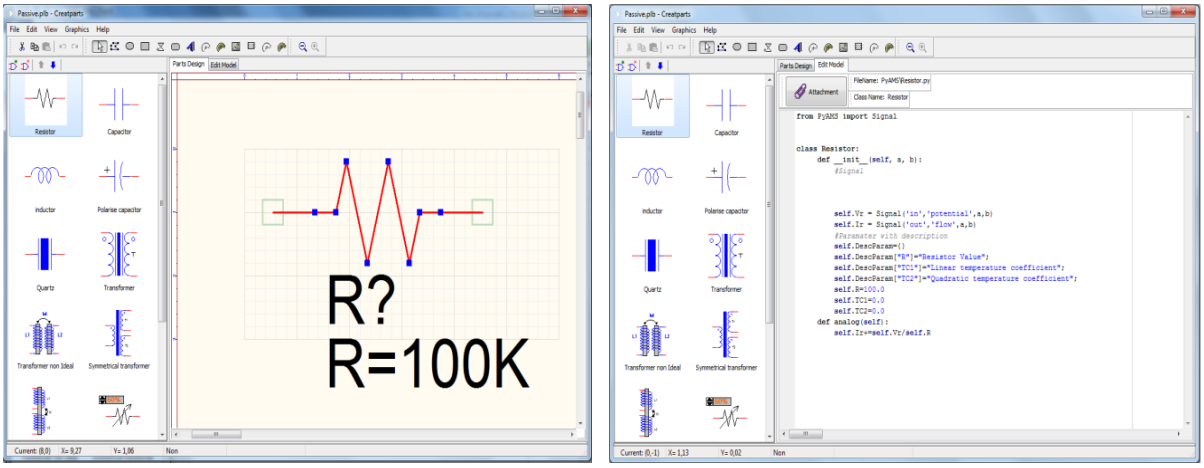


Fig. IV.20. The Interface for design circuit by schema method: (a) List of Parts. (b) Schematic editor. (c) Command of: add wires and parts, compilation and analyzing. (d) Out messages.

The part is symbol present the analog elements drawing by Editor Parts which shown in Fig. IV.21.a. The design of part is by add element of drawing: line, arc, rectangle … etc, attaching by description model Fig. IV.21.b.

When compile the schema of circuit, it converted automatically the structure graphic of circuit to mode textual in python language to simplify of compiling and analyzing by python language. The interface of analyzing is described after this section.



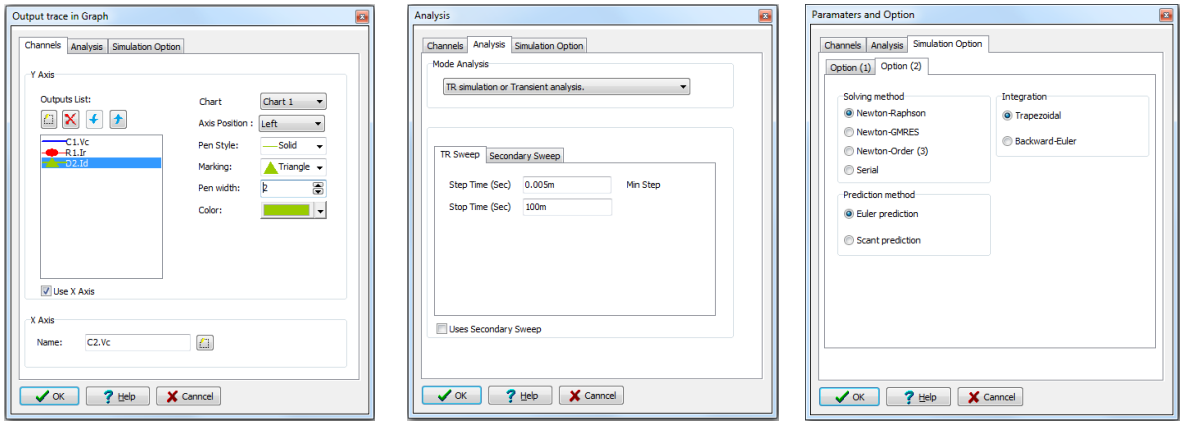
(a) **(b)**

Fig. IV.21. The interface of Editor Part: (a) Drawing part. (b) Sheet for attachment model by part description by PyAMS languge.

IV.4.4.3 Method of analyzing and interface using

After construct the circuit by one of method textual or schema using command of analyzes which is presented by dialog box shown in Fig. IV.22, it has three pages:

- Page 1: (Fig. IV.22.a) is used to choose the signals or nodes in circuit, to execute in output simulation to show or display in wave editor Fig. IV.23.
- Page 2: (Fig. IV.22.b) is used to choose mode of analysis (mode DC or TR or AC) with start point and end point of analysis. The option of analysis in PyAMS is the same method of analysis in SPICE.
- Page 3: (Fig. IV.22.c) is an option of analyzing; it has to choose method of solving, method of integration, condition of stop iteration...etc.



(a) **(b)** **(c)**

Fig. IV.22. Dialogue of analysis. (a) Page output signals or nodes. (b) Page of analysis. (c) Page of Option.

After choose the output signals or nodes and option with method of analysis, we execute the circuit. The result of output is automatically displayed in wave editor (Fig. IV.23), the wave editor simplifies the presenting of result of simulation in schematic editor (Fig. IV.24).

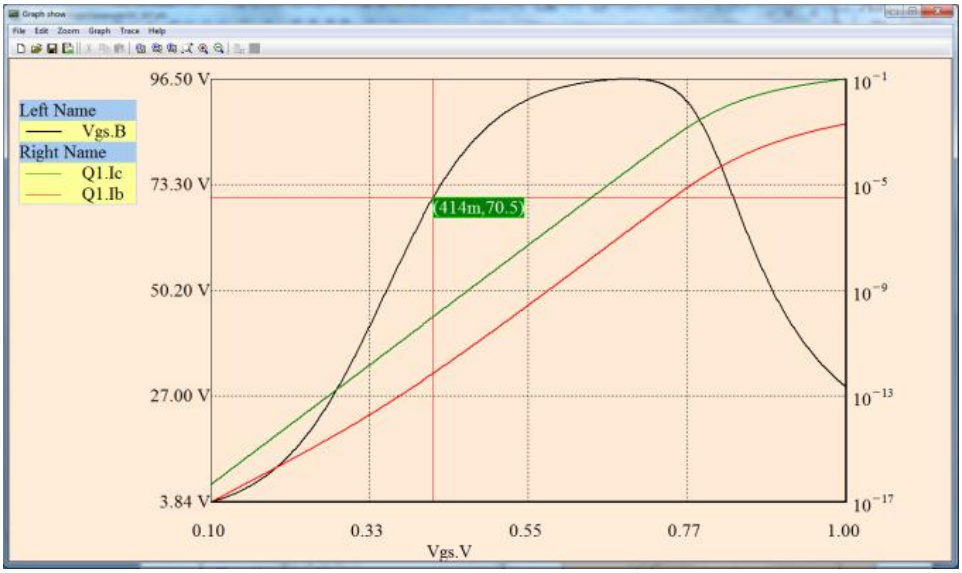


Fig. IV.23. Wave Editor.

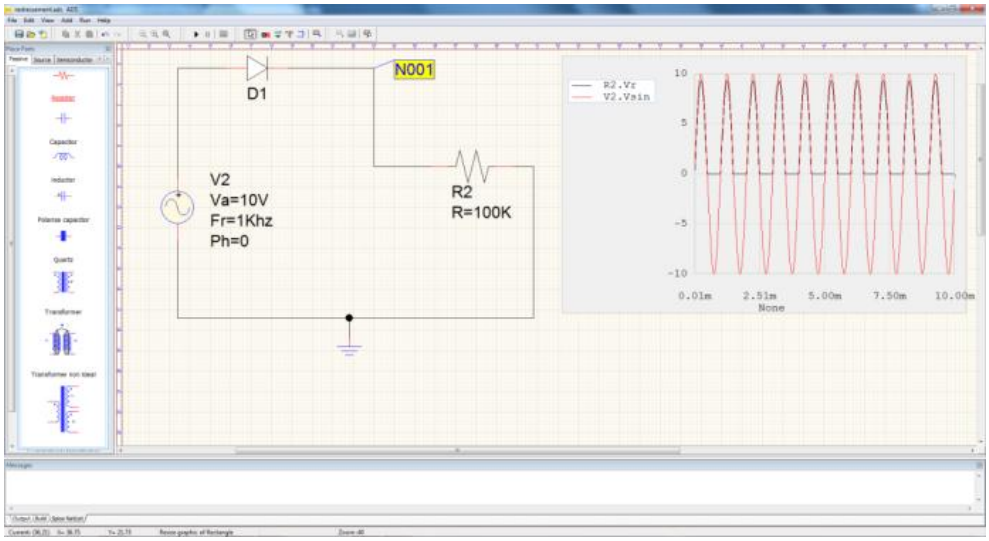


Fig. IV.24. The result of simulation presented in schematic editor.

IV.5. Example of analysis circuits

In this part, the application of PyAMS software to analyze some examples of circuit, by constructing circuit in mode graphic or mode textual, based on model of library constructed before (Resistor, Capacitor, Diode, BJT, NMOS, PMOS, and Source).

IV.5.1 Simple RC circuit

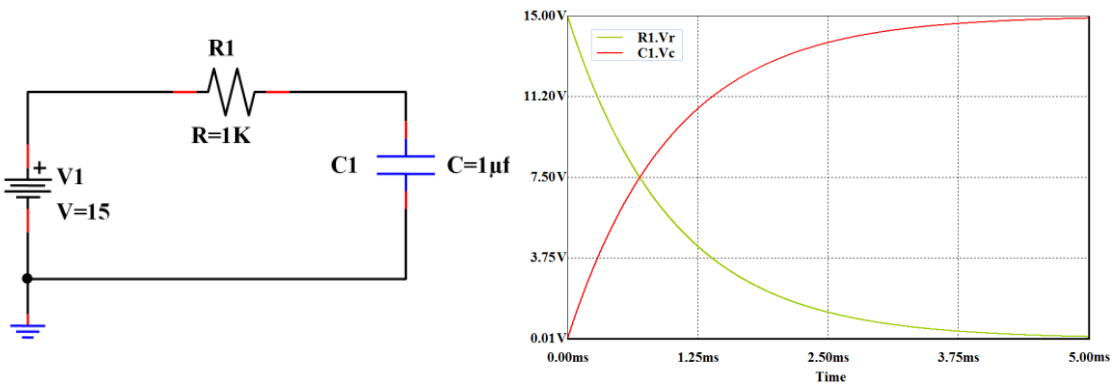
The simple RC (resistor-capacitor) circuit is usually the beginner model for evaluating electrical analog simulation systems. The following PyAMS text (Program IV.14) presented the circuit of Fig. VI.26.a The simulation result (Fig. IV.26.b) is the behavior of in voltage signal *Vc* of a capacitor *C1 (C1.Vc)* which is charged and in voltage signal *Vr* of a resistor *R1* (*R1.Vr*) which is discharge, dependent by time.

Program IV.14: Create circuit RC by textual method.

```
# Import models--------------------------------
from Resistor import Resistor
from SourceVdc import SourceVdc
from Capacitor import Capacitor
form PyAMS import gnd

class Circuit_RC():
    def  __init__(self):
       # attachment btuine elements-------------
       self.V1=SourceVdc('node1',gnd)
       self.R1=Resistor('node1', 'node2')
       self.C1=Capacitor('node3',gnd)

      # modefied parameters---------------------
       self.V1.Vdc=15
       self.R1.R=1000.0
       self.C1.C=1.0e-6
```



(a) (b)

Fig. IV.26. (a) Simple RC circuit. (b) Simulation results of RC circuit.

IV.5.2 MOS Oscillator

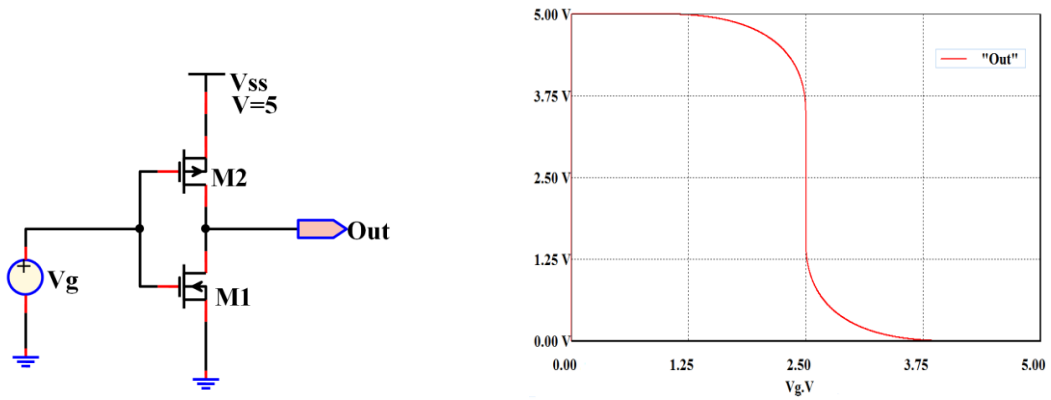
Obviously, an odd number of inverters based by MOS transistor [116](the Fig. IV.27 presents the inverting value by MOSFET simulation by mode DC) with resistors and capacitors with in a chain tend to oscillate. A single inverter is composed by transistor (PMOS and NMOS), resistor *R1* and capacitor *C1*. The PyAMS description textually of the inverter is presented in following:

Program IV.15: Create circuit inverter by textual method.

```
# Import models-----------------------------------------
from PMOS import PMOS
from NMOS import NMOS
from SourceVdc import  SourceVdc
from Capacitor  import  Capacitor
from PyAMS import gnd

class Circuit_ Inverter():
   def  __init__(In,Out):
        # attachment between elements----------------------
        self.V1=SourceVdc('node1',gnd)
        self.V2=SourceVdc(gnd,'node2')
        self.C1=Capacitor('node3',gnd)
        self.M1=NMOS(Out,In,'node3')
        self.M2=PMOS(Out,In,'node3')

       # modified parameters-------------------------------
        self.V1.Vdc=5
        self.R1.R=4
        self.C1.C=2e-9
  def SubBlock(self):
     # Return by Sub circuit-------------------------------
     return [self.V1,self.V2,self.R1,self.C1,self.M1,self.M2]
```



(a) (b)

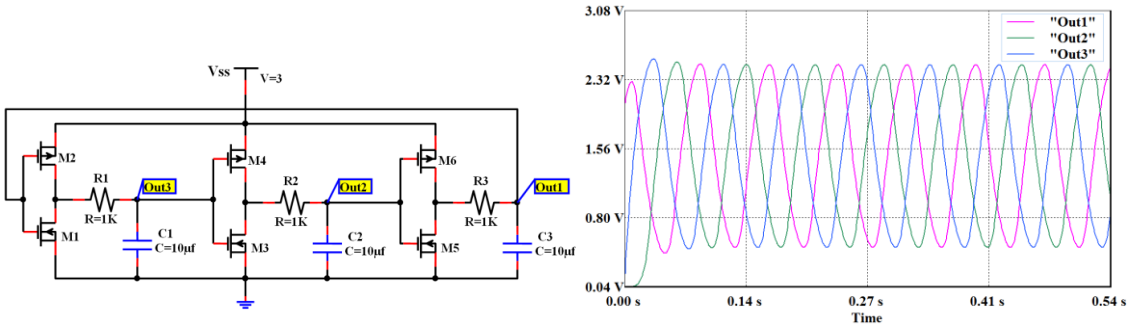
Fig. IV.27. Inverting by NMOS and PMOS: (a) Circuit of inverter (b) Result of analysis in mode DC.

The program IV.16 construct MOS Oscillator based on three inverter Inv1, Inv2 and Inv3 connected in loop, the Fig. VI.22 presents out result of simulation in the three nodes Out1, Out2 and Out3.

Program IV.16: Textual method of oscillator circuit.

```
# Import models-----------------------------------------
from Circuit_ inverter import Circuit_ inverter
from PyAMS import gnd

# Attachment between elements----------------------------
self.Inv1=Circuit_ inverter ('out1','out2')
self.Inv2= Circuit_ inverter ('out2','out3')
self.Inv3= Circuit_ inverter ('out3','out1')
```



(a) (b)

Fig. IV.28. Oscillator circuit: (a) the schema of circuit (b) the result of simulation in mode TR.

IV.5.3 Chua circuit

The only nonlinear component in the circuit which causes the chaos is Chua's Diode [117]. It is a one-port. Its characteristic can be seen in the following. It generating resistor negative with two values.

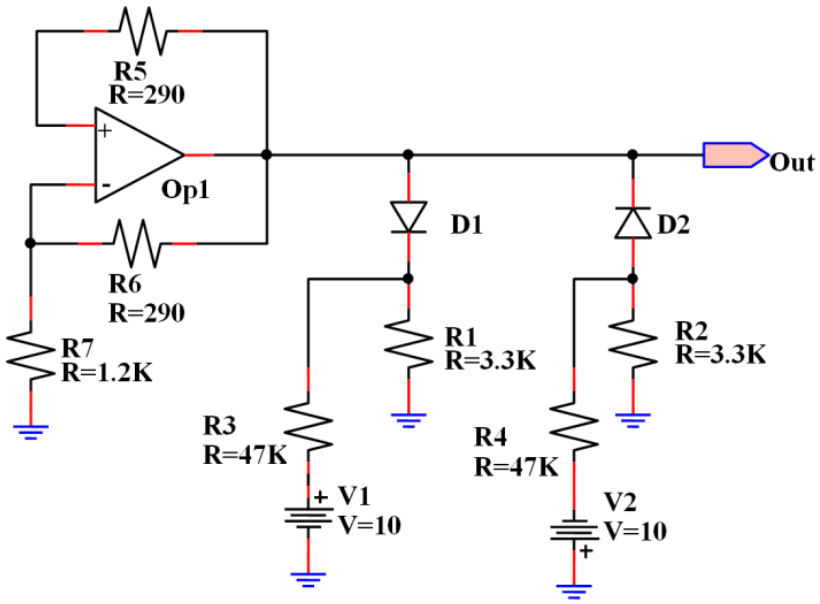


Fig. IV.29. Diode Chua circuit.

The following PyAMS text shows the circuit description, and the Schematic description is presented in Fig. IV.29.

Program IV.15: Create circuit of Diode Chua by mode textual.

```
#Import models------------------------------
from Resistor import Resistor
from Diode import Diode
from SourceVdc import SourceVdc
from Opa import Opa
from PyAMS import gnd

class Diode_Chuas:
def  __init__(Out):
          # attachment between elements---------
          self.Vp=SourceVdc('N6',gnd)
          self.Vn=SourceVdc(gnd,'N7')
          self.R1=Resistor('N1',Out)
          self.R2=Resistor('N3','N1')
          self.R3=Resistor('N4',gnd)
          self.R4=Resistor('N4','N6')
          self.R5=Resistor('N5',gnd)
          self.R6=Resistor('N5','N7')
          self.VR1=Resistor('N3',gnd)
          self.P1=Opa(Out,'N3','N1')
          self.D1=Diode(Out,'N4')
          self.D2=Diode('N5',Out)
          # modified parameters-----------------
          self.P1.Vmax=10
          self.P1.G=3e+3
          self.Vp.V=10
          self.Vn.V=10
          self.R1.R=290
          self.R2.R=290
          self.R3.R=3300
          self.R5.R=3300
          self.R4.R=47000
          self.R6.R=47000
          self.VR1.R=1200
          self.Rt=Resistor('N8','N2')
          self.C1=InitCap('N2',gnd)
          self.C2=Capacitor('N8',gnd)
          self.L1=Inductor('N8',gnd)
          self.Rt.R=1600
          self.C1.C=4.6e-9
          self.C1.In=0.2
          self.C2.C=47e-9
          self.L1.L=8.5e-3
          self.D1.Iss=4.5e-9
          self.D2.Iss=4.5e-9

def SubBlock(self):
     # Return by Sub circuit--------------------
     return [self.Vp,self.Vn,self.R1,self.R2,self.R3,self.R4,self.R5,self.R6,
             self.VR1,self.P1,self.D1,self.D2]
```

To verify the function Diode Chua's we use to analyze in mode DC but in attachment to resistor *Rd* and source *Vd*.

We changed the voltage parameter *V* of source *Vd* from -10 to 10 we notify the direction of out current signal *Ir* from element Resistor *Rd* and it is the opposite direction of *Vd*, or generate negative resistor.

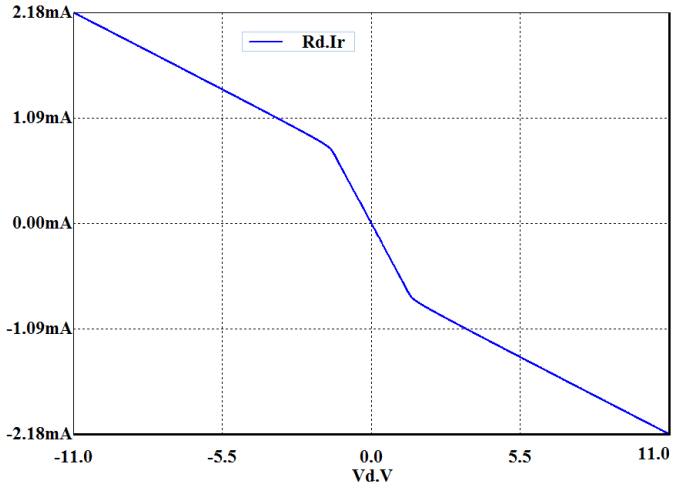


Fig. IV.30. Characteristic of Chua's Diode

Program IV.17 represent the work Chua's circuit in mode TR analyses , it is connected the elements resistors (*R1*, *R2*), capacitor (*C1*, *C2*) and inducatance *L1* with diode chua *Nr* presented in circuit Fig. IV.31 with the result of simulation.

Program IV.17: Create Chua circuit.

```
#Import model-------------------------------
from  Diode_Chuas import  Diode_Chuas

class Circuit_Chuas:
 def __init__(self):
          self.Dh=Diode_Chuas('N1')
          self.Id=Resistor('N1','N2')
          self.Vd=SourceVdc('N2','0')
```

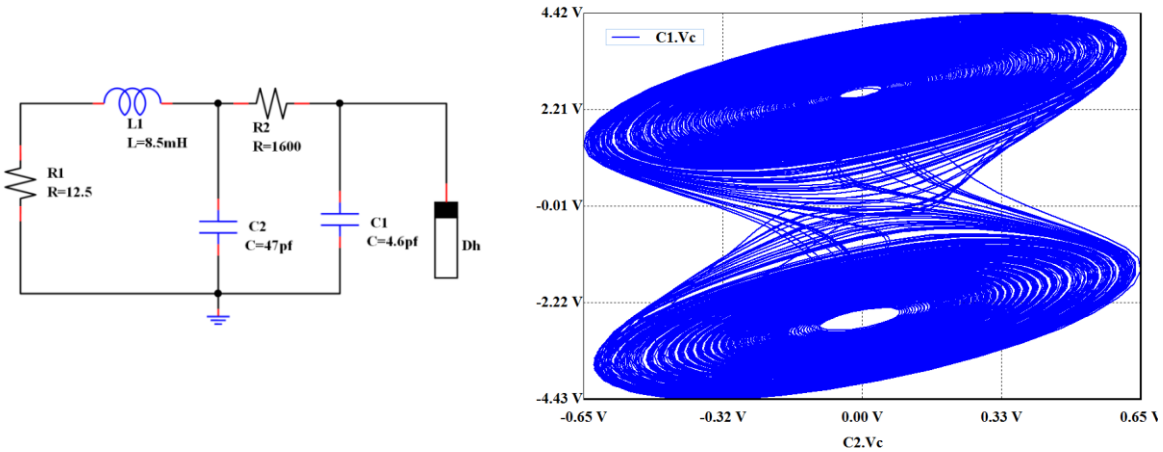


Fig. IV.31. Resulte of simulation of Chua's Diode.

IV.5.4 Modeling a nonlinear transformer

In this sub section we make study on how to convert equation and formula of nonlinear transformer model to model based on PyAMS language. Suppose you want to model the

effects of nonlinearity in a transformer core. A transformer is schematically depicted in Fig. IV.32 [1]. The cross section of the core is denoted by *A*. *L* denotes the mean length of the magnetic force lines inside the core (dashed line in Fig. IV.32). N_1 and N_2 denote the number of windings in the primary and secondary coils. R_1 and R_2 are the resistances of the primary and secondary windings.

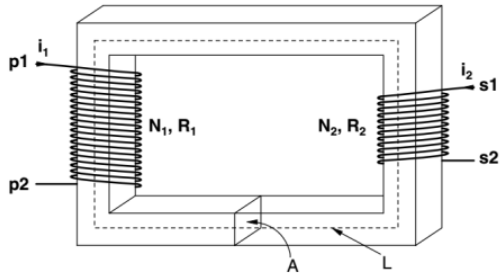


Fig. IV.32. A transformer with two windings [1].

The currents in the primary i_1 and the secondary winding i_2 are sources of magnetization resulting in a magnetic field strength in the core given by:

$$H(t) = g(i_1, i_2) = \frac{N_1\, i_1(t) + N_2\, i_2(t)}{L} \tag{4.35}$$

The corresponding magnetic flux density depends on the material and is generally related to *H(t)* in a nonlinear manner:

$$B(t) = f(H(t)) \tag{4.36}$$

The change in magnetic flux density induces a voltage in the primary (e_1) and secondary (e_2) windings:

$$\begin{cases} e_1(t) = \frac{\partial}{\partial t}(N_1\, A\, B(t)) \\ e_2(t) = \frac{\partial}{\partial t}(N_2\, A\, B(t) \end{cases} \tag{4.37}$$

The function *f (H)* depends on the type of material the core is made of. In this example the following characteristic will be used:

$$f(H) = \arctan\left(\frac{H}{40}\right) + 4\pi\, 10^{-7}\, 110\, H. \tag{4.38}$$

The units of *f(H)* are Vs/m^2 and the units for H are *A/m*. Note that this function does not take into account hysteresis.

The above equations can be described by a model circuit with two electric inputs and an internal magnetic circuit, as depicted by Fig. 33. It is electrical model of nonlinear transformer [4].

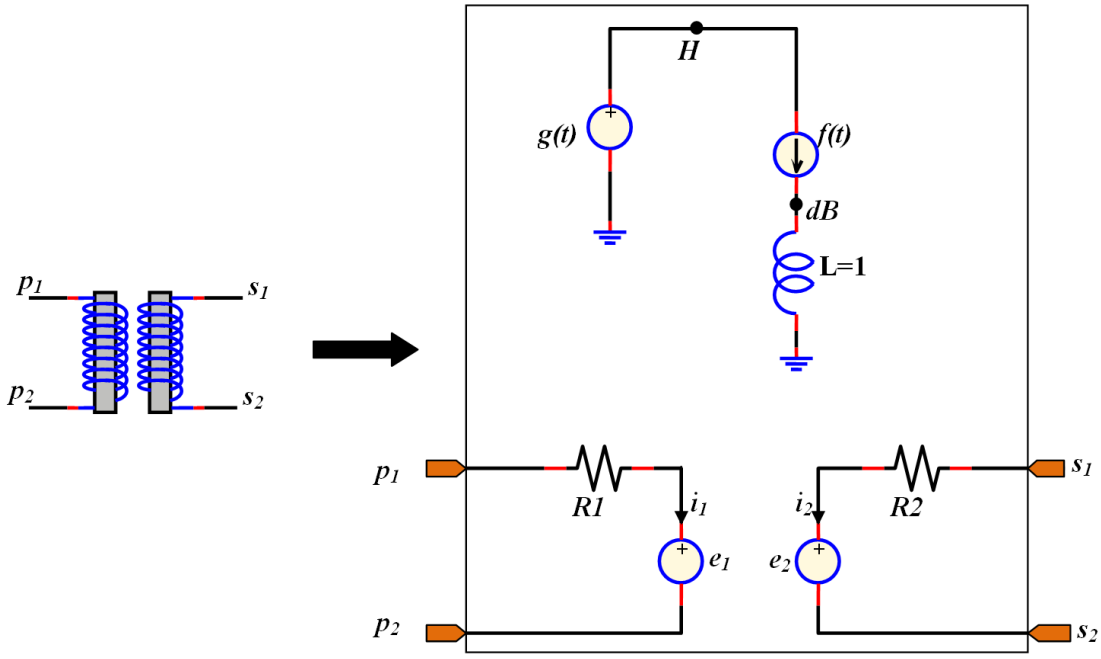


Fig. IV.33. A nonlinear transformer symbol (*left*) and the corresponding model (*right*).

The fonction *H* you can presented by new node in the circuit. The value of voltag in the node *H* it genreted by source *g* dependent by currents acrosse of source e_1 and source e_2 with paramater N_1, N_2 and *L*:

*self.g+=(self.N1*self.e1.i+ self.N2*self.e2.i)/self.L*

the modefied of unit of *g*:

self.g.unit="A/m"

the function *B* you can presanted by source cournent attachment with *H* and new node *dB*, the source current is f dependent by voltage in the node *h*, you can presented by these expresion:

*self.f+=atan(self.H/40)+110 *4e-7*pi*self.h*

for the unit of *f*:

self.f.unit="Vs/m^2"

by inductance in *gnd* and *dB* you can genret derivativ of *B* on *L=1*, for that you must add element how created inductance. The coplete modele of nonlinear transformer it presented by thise programe:

Program IV.18: Modeling of nonlinear transformer by PyAMS.

```
#Import model-------------------------------
from Resistor import Resistor
from PyAMS import Signal
from math  import atan,pi
from Inductance import Inductance
from NewNode import NewNode

class Nonlinear_Transformer:
      def __init__(self,p1,p2,s1,s2):
           # creat new nodes-----------------------------------------
           [H,dB,n1,n2]=[NewNode(),NewNode(),NewNode(),NewNode()]
          #creat Signals-----------------------------------------------
           self.e1=Signal('out','potential',n1,p2)
           self.e2=Signal('out','potential',n2,s2)
           self.g=Signal('out','potential',H)
           self.f=Signal('out','flow',H,dB)
           self.dB=Signal('in','potential',dB)
           self.H=Signal('in','potential',H)
           self.g.unit="A/m"
           self.f.unit="Vs/m^2"
          #Creat elements-----------------------------------------------
           self.r1=Resistor(p1,n1)
           self.r2=Resistor(s1,n2)
           self.l1=Inductance(dB,'0')
          #Paramaters---------------------------------------------------
           self.N1=100.0
           self.N2=1.0
           self.a=1.0
           self.L=1.0
           self.R1=100.0
           self.R2=100.0

      def  SubBlock(self):
           self.r1.R=self.R1
           self.r2.R=self.R2
           self.l1.L= 1
           return [self.r1,self.r2,self.l1]

      def analog(self):
            self. g+= (self.N1*self.e1.i+self.N2*self.e2.i)/self.L
            self. f+=atan(self.H/40)+110*V*4e-7*pi
            self.e1+= self.n1*self.a*self.dB
            self.e2+= self.n2*self.a*self.dB
```

Now let us look at the response of the transformer when a sinusoidal voltage is applied at the primary. Let us take a look at the response when the secondary is open-circuited. Initially the correct net class is selected and the load resistance is set to a high value (*100 M*). This is followed by a simulation and plotting of two graphs (*B* and transformer currents). The resulting graphs are depicted in Fig. IV.34.

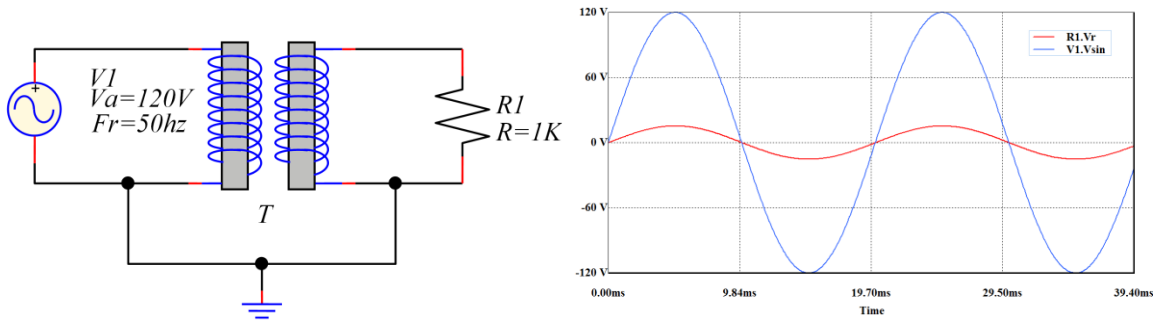
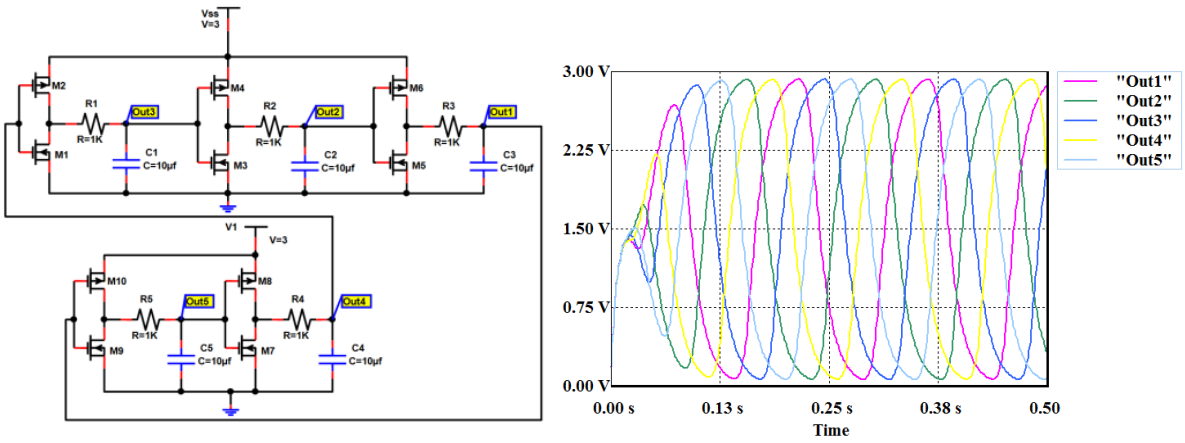


Fig. IV.34. Response to sinusoidal voltage at the primary coil of nonlinear transformer.

IV.6. Circuit analysis by PyAMS

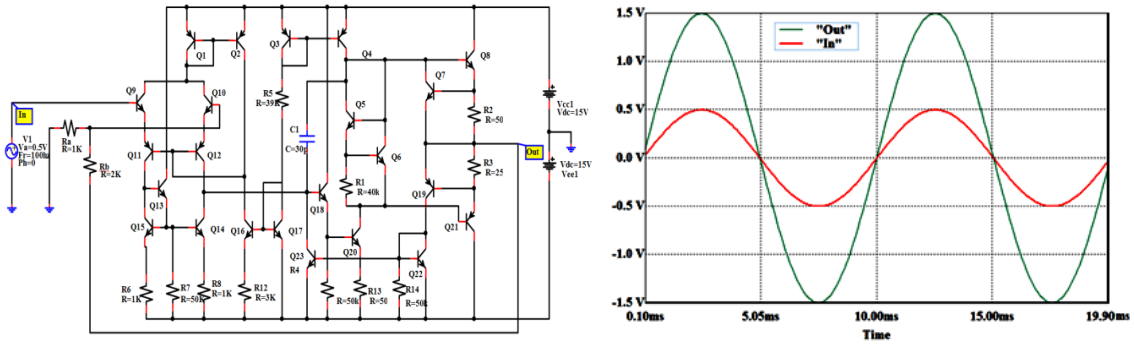
In this subsection the PyAMS softwar used to analyze some universal circuits as: the five-stage ring oscillator circuit (Fig. IV.35), the LM741 non-inverting amplifier (Fig. IV.36), Phase-shift oscillator circuit (Fig. IV.37), Schmitt trigger circuit (Fig. IV.38), The Peltz oscillator circuit (Fig. IV.39), Operational transconductance amplifier (Fig. IV.40), Pulse generator circuit (Fig. IV.41) and the stable multivibrator (Fig. IV.42).

The left figures depict the schematic of the circuit realized by the developed software PyAMS while the right figures illustrate the circuit analysis.



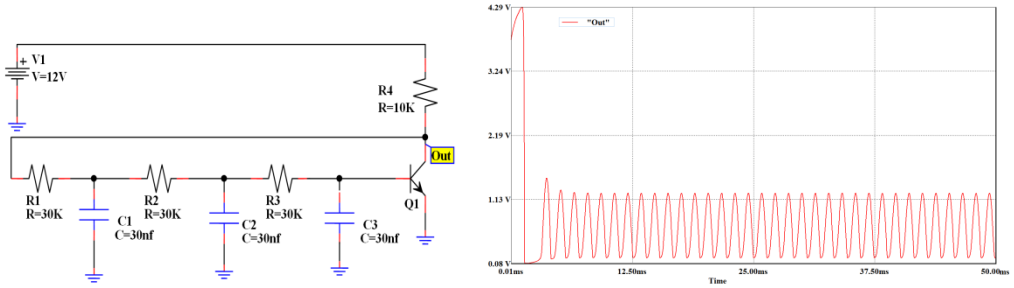
(a) **(b)**

Fig. IV.35. Five-stage ring oscillator circuit: (a)- the circuit (b)- simulation results of genration of osillations in five position of circuit.



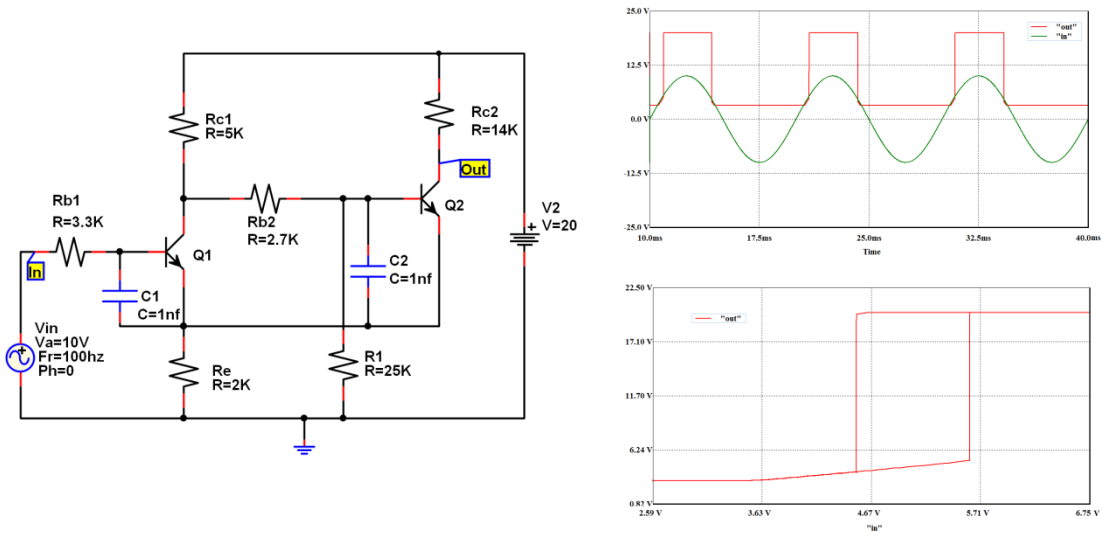
(a) **(b)**

Fig. IV.36. The LM741 non-inverting Amplifier circuit: (a)- the circuit, (b)- the waveforms for LM741 input and output



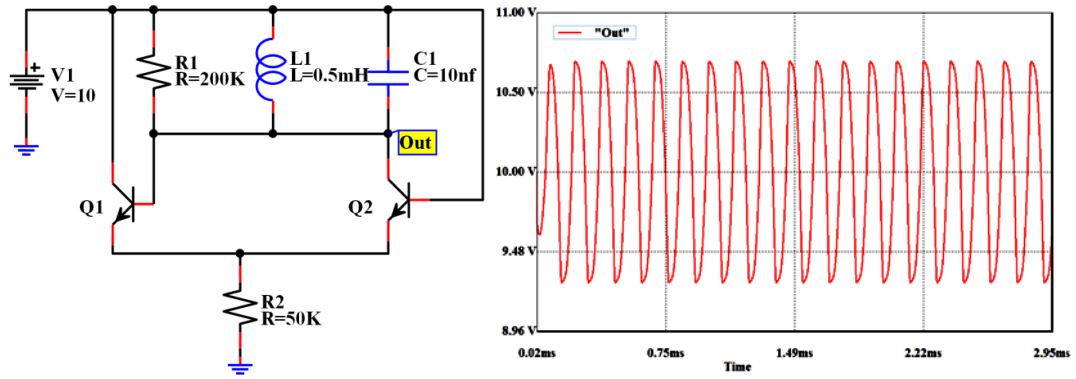
(a) **(b)**

Fig. IV.37. Phase-shift oscillator circuit: (a)- the circuit (b)-simulation results of osillation in "out" of circuit.



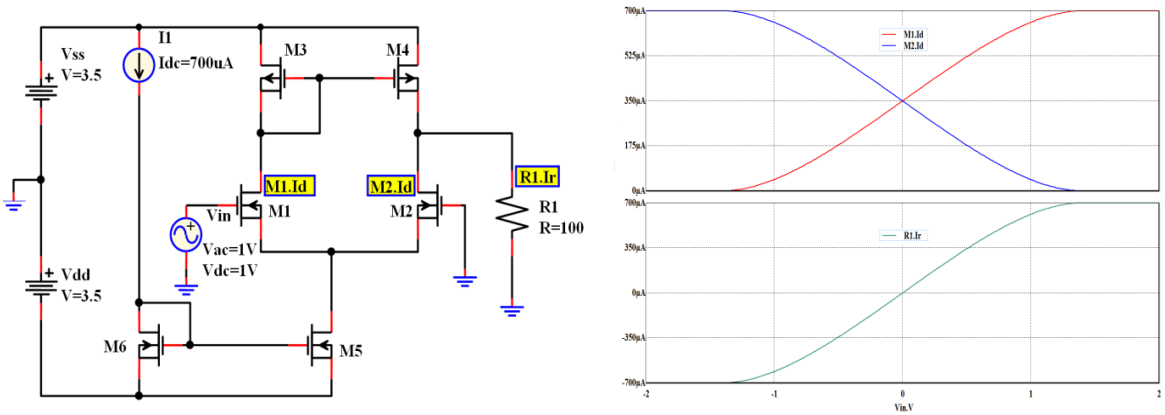
(a) **(b)**

Fig. IV.38. Schmitt trigger circuit: (a)- the circuit (b)-simulation results of comparator input signal presented by output and curve non-inverting of schmitt trigger.



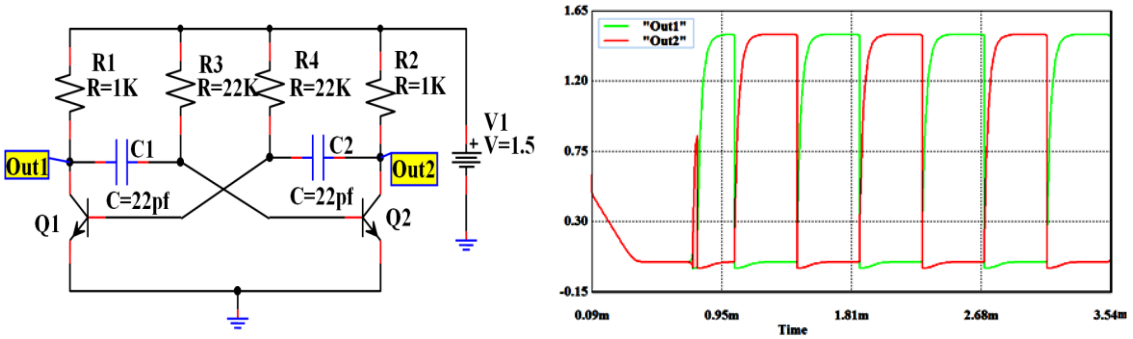
(a) **(b)**

Fig. IV.39. The Peltz oscillator Circuit: (a)- the circuit, (b)- simulation results.



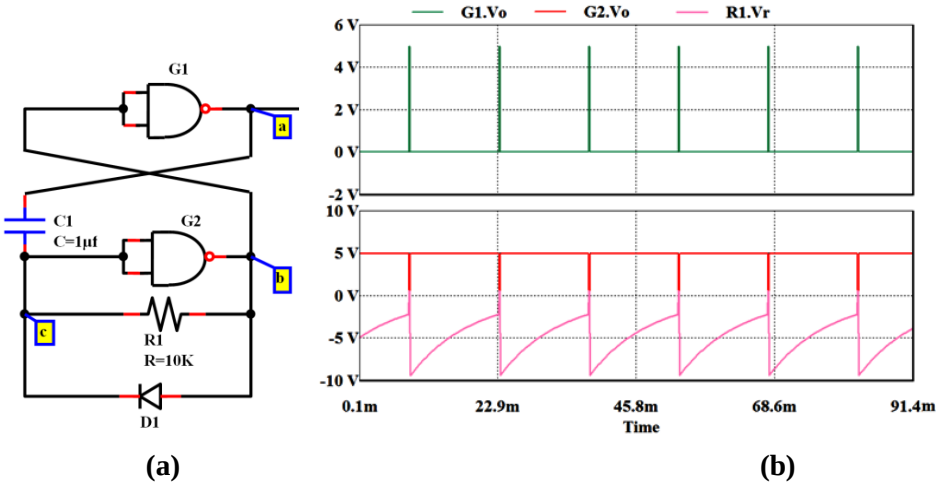
(a) **(b)**

Fig. IV.40. Operational transconductance amplifier (OTA): (a)- the circuit (b)-simulation results of output current in drin of M1 with M2 and resistor dependent by input voltage.



(a) **(b)**

Fig. IV.41. Astable multivibrator circuit: (a)-the circuit (b)-simulation results of astable multivibrator circuit presented in two output.



(a) **(b)**

Fig. IV.42. Pulse generator circuit: (a)-the circuit, (b)-simulation results.

IV.7. Conclusion

PyAMS is new software for modeling of analog and mixed elements with the help of Python language and construct circuits and simulating them by analyzing using one of these three modes: AC, DC and TR analysis. These methods of analysis used continuous method to accelerate the finding of operating point.

The PyAMS simplify modeling element to voltage or current source dependent by voltage in the nodes and current across the source, the modeling of element in PyAMS with python language is based on two functions principal: *_initi_ function* for declaration of signals and parameters and *analog function* for operation between signals.

The software of PyAMS simplify for the user to construct circuit by using two modes:

- Textual based on python language
- Graphic based on design schema of circuit linking wires and parts, (the parts are drawing by editor of parts software).

The effectiveness of working PyAMS software is presented: Modeling analog elements (Resistor, Capacitor, Inductor, Diode, BJT, MOSFET, Logic gates and Nonlinear Transformer) and analyzing or simulating universal and complex circuits.

In the future:

- Amelioration of solving a system of nonlinear differential equations for analyzing circuit in real time.

- Construct animation elements to visualize value of signal type out direction and for command the signal type in direction.

In PyAMS there are two types of signal voltage and current, in the future we will add three types of signals:

1. Signal type digital to construct digital element (ex: gates, flipflop, counter,…etc).
2. Signal type data digital to construct element based on data (ex: RAM, Controller,…etc).
3. Signal type flow and potential to construct mechanical element (ex: Motor,…etc).

Conclusion and Perspectives

Finding the operating points of nonlinear circuits is a key problem in circuit simulation, till now, Spice simulator is used to formulate the circuit's equation by using the MNA method and employs the Newton-Raphson (NR) to solve it. The NR is the best numerical method applied in circuit analysis due to its principle of converting nonlinear elements to linear elements depending on Newton iteration, but the NR method does not always converge to a solution, for that, some conditions and techniques developed in SPICE should be used to help the convergence of the NR method. These techniques are: Source stepping, G_{min} stepping, and voltage limiting.

In our doctorate thesis, we have proposed a new approach to accelerate the solving of nonlinear equations based on improving the *continuous method*. The major contributions of our work are:

1. Adding the fourth stage to the continuous method to control the step length of the continuous parameter, it depends on the correction process; the objective is to obtain good direction (approximation) of solving.
2. We have used new method (NM) which applies the *Newton order three* in the correction process rather than the *Newton order two* used in the Newton-Raphson method. Our method employs two parameters (m) and (C). By selection convenient parameters (m=0.5 or m=0.75 or m=0.9 for $C=1.75$). the performance of our method was compared with similar iteration methods which demonstrated that our method is better than:
    - Newton order three created by M. Waseem (MW) [63].
    - Newton order three created by M. Darvishi and A. Barati (MA) [61].
    - Newton Raphson (NR) which is used in simulator SPICE [1].
    - Newton Karylov (NG) which used in simulator Xyce [76].

3. We have introduced a new idea which is converting the analog elements of the circuit to their dependent voltage sources or dependent current sources, and applying the Kirchhoff Current Law (KCL) and the Kirchhoff Voltage Law (KVL) to obtain the equation of the circuit.
4. We have created new software called PyAMS (**Py**thon for **A**nalog and **M**ixed **S**ignals) for windows system programmed using Delphi XE8 and Python 2.7. It employs the proposed method. This software has the following features:
    - Modeling any analog element in the form of dependent source.
    - Create circuits by either the textual or the schema method (CAD).
    - Analysis of circuits in three modes DC, TR and AC.

İn order to demonstrate the effectiveness of the proposed method, a comparison was done with two other methods, to do so, many types of integrated circuits are used. Circuit analysis of many universal circuits has been carried out to verify the correct functioning of the circuit based on the proposed method. The obtained results using the new software based on the proposed method have shown the effectiveness of our approach where it is about three times faster than the VGNH method implemented in the SPICE software.

Compared to the SPICE software, our new software PyAMS has the following advantages:

1. It is faster than the SPICE simulator in analyzing circuits, where it is about three times faster than the VGNH method implemented in the SPICE software.
2. It offers the modeling of elements contrary to SPICE,
3. Create circuits by either the textual or the schema method (CAD),
4. We can create our own library of components,
5. We can modify the model and the design of the component.

Extensions and future work

Potential future work will focus on:

- Explore and extend our method to artificial intelligence methods like Neural Networks, Fuzzy Logic and Genetic Algorithms, and optimization methods as Ant Colony Optimization (ACO), Particle Swarm Optimization and Honey Bee Mating Optimization (HBMO).
- Construct animation elements to visualize the output signal values and to command the input signal.

- In PyAMS there are two types of signal voltage and current, we will add three types of signals:
    1. Digital Signal to build digital elements (e.g. gates, flip-flops, counters,…etc).
    2. Digital data signal to build elements based on data (e.g. RAM, Controller,…etc).
    3. Flow and potential Signals to build mechanical and hydraulic elements (e.g. Motors, jacks,…etc).

References

[1]. T. Tuma, and A. Burmen, **"Circuit Simulation with SPICE OPUS Theory and Practice"**, *Birkhäuser, Boston*, 2009.

[2]. A. Vladimirescu, **"The SPICE Book"**, *John Wiley & Sons, New York*, 1994.

[3]. F. Najm, **"Circuit Simulationa: Hoboken"**, *John Wiley & Sons, Jersey*, 2010.

[4]. C.T. Kelley, ***"Solving Nonlinear Equations with Newton's Method",*** *Algorithms for Numerical Calculations. SIAM, Philadelphia*, 2003.

[5]. P. Deuflhard, **"Newton Methods for Nonlinear Problems Affine Invariance and Adaptive Algorithms"**, *Springer, Berlin*, 2011.

[6]. Yu Hong, **"Study on the pseudo-transient analysis algorithm for nonlinear circuit DC analysis"**, *Thesis of Waseda University*, 2008.

[7]. D.O. Pederson, **"A Historical Review of Circuit Simulation"**, *Circuits and Systems, IEEE Transactions*, Vol.31, pp. 103 - 111 Jan 1984.

[8]. T.L. Quarles, **"Analysis of Performance and Convergence Issues for Circuit Simulation"**, *Univ. of California, Berkeley,* 1989.

[9]. L.W. Nagel, **"SPICE2: A Computer Program to Simulate Semiconductor Circuits"**, *Univ. of California, Berkeley*, 1975.

[10]. T. L. Quarles, **"Analysis of performance and convergence issues for circuit simulation",** Univ. of California, Berkeley, CA, ERL-M89/42, April 1989.

[11]. R. Wilton, "Supplementary algorithms for DC convergence", IEE Colloquium, SPICE: Surviving Problems in Circuit Evaluation, pp.3/1-3/19, June 1993.

[12]. Y. Inoue, S. Kusanobu, K. Yamamura and T. Takahashi, **"Newton-fixed point homotopy method for finding DC operating-points of nonlinear circuits",** Proc. 2001 Int. Tech. Conf. Circuits/Syst., Computer & Commun., vol.1, pp.370–373, July 2001.

[13]. Y. Inoue, S. Kusanobu and K. Yamamura, **"A practical approach for the fixed-point homotopy method using a solution-tracing circuit",** IEICE Trans., vol.E85-A, no.1, pp.287–298, Jan. 2002.

[14]. Y. Inoue, Y. Imai, M. Ando, K. Yamamura, **"An efficient homotopy method for solving transistor circuits"** *In: IEEE 2004 International Symposium on Circuits and Systems, 47th Midwest: IEEE*. 25-28 July 2004.

[15]. K. Yamamura, W. Kuroki, **"An Efficient and Globally Convergent Homotopy Method for Finding DC Operating Points of Nonlinear Circuits"**, *In: IEEE 2006 Asia and South Pacific Conference on Design Automation*, IEEE. pp. 408-415, 24-27 Jan. 2006.

[16]. K. Yamamura, W. Kuroki, **"An Efficient Homotopy Method That Can Be Easily Implemented on SPICE"**, *In: IEEE 2006 International Symposium on Circuits and Systems, Island of Kos: IEEE*. pp. 4911-4914, 21-24 May 2006.

[17]. D. Niu, X. Wu, Z. Jin, **"Effective and Globally Convergent Newton Fixed-Point Homotopy Method for MOS Transistor Circuits"**, *Transactions on Fundamentals of Electronics, Communications and Computer Sciences,* Vol. E96-A, no.9, pp. 1848-1856, Sept. 2013.

[18]. D. Niu, Z. Zhao, Y. Inoue, **"An Effective and Globally Convergent Newton Fixed-Point Homotopy Method for MOS Transistor Circuits"**, *IEICE Technical Report on Nonlinear Problems,* Vol. 112, no. 117, pp. 85-90, July 2012.

[19]. W. Kuroki, K. Yamamura, S. Furuki, **"An efficient variable gain homotopy method using the SPICE-oriented approach"**, *IEEE Transactions on Circuits and Systems II,* Vol. 54, pp. 621-625, July. 2007.

[20]. K. Yamamura, T. Miyamoto,**"DC Operating Point Analysis of Transistor Circuits Using the Variable-Gain Homotopy Method"**, *IEICE Transactions* Vol. 97, pp. 1042-1050, May 2014.

[21]. D. Niu, G. Hu, Y. Inoue, **"Theorems on the Global Convergence of the Nonlinear Homotopy Method for MOS Circuits"**, *In: IEEE 2011 Asia Pacific Conference on Postgraduate Research in Microelectronics and Electronics (Prime Asia), Macau: IEEE,* pp. 41 - 44, 6-7 Oct. 2011.

[22]. M. Jeeradit, J. Kim, M. Horowitz, **"Intent-leveraged Optimization of Analog Circuits via Homotopy"**, *In: IEEE 2010 Europe Conference & Exhibition in Design, Automation & Test (DATE), Dresden: IEEE, pp. 1614-1619, 8-12* Mars *2010.*

[23]. Y. Inoue, S. Kusanobu, K. Yamamura, M. Ando, **"An Effective Initial Solution Algorithm for Globally Convergent Homotopy Methods"**, *In: IEEE 2003 Proceedings of the 2003 International Symposium on Circuits and Systems, 2003. ISCAS '03: IEEE,* pp. 196-199, 25-28 May 2003.

[24]. D. Niu, Z. Jin, X. Wu, Y. Inoue, **"A Globally Convergent and Highly Efficient Homotopy Method for MOS Transistor Circuits"**, *IEEE Computing and Convergence Technology,* pp. 1349-1352, Dec. 2012.

[25]. Y. Imai, K. Yamamura**,** Y. Inoue, **"An Efficient Homotopy Method for Finding DC Operating Points of Nonlinear Circuits"**, *IEICE Trans. Fundamentals,* Vol. 88, pp. 2554-2561, Oct. 2005.

[26]. M. Tadeusiewicz, S. Hałgas, **"Multiple Soft Fault Diagnosis of Nonlinear Circuits Using the Continuation Method"**, *Springer Journal of Electronic Testing,* Vol. 28, pp 487-493, 2012.

[27]. W. Gu, W. Liu, R. Wang, **"Study on Prediction-Correction Homotopy Method of Tracking Homotopy Method of Tracking Hopf Bifurcation Point"**, *In: IEEE 2010 Conference of Transmission, Distribution and Exposition (T&D),* pp. 1-7, April 2010.

[28]. Lj. Trajkovic, **"DC Operating points of transistor circuits"**, *Nonlinear Theory and Its Applications, IEICE,* Vol. 3, pp. 287-300, July 2012.

[29]. A. Dyess, E. Chan, H. Hofmann, W. Horia, and Lj. Trajkovic, **"Simple implementations of homotopy algorithms for finding dc solutions of nonlinear circuits"**, *In Proc. IEEE Int. Symp. Circuits and Systems, Orlando, FL,* Vol. 4, pp. 290-293, May 1999.

[30]. J. Eric Melo and L. Trajkovic, **"Improving an electronic circuit simulator based on homotopy methods"**, *Technical report, SFU Burnaby*, 2014.

[31]. K. Dickson, C. Kelley, I. Ipsen, and I. Kevrekidis, **"Condition estimates for pseudo-arclength continuation"**,*SIAM Society for Industrial and Applied Mathematics* Vol. 45, No. 1 pp. 263–276, 2007

[32]. J. Vlach and K. Singhal, **"Computer Methods for Circuit Analysis and Design"**, *John Wiley & Sons, New York*, 1983.

[33]. F. Cox, W. Kuhn, J. P. Murray, S. Tynor, **"Code-level Modeling in XSPICE",** *In: IEEE 1992 International Conference on Circuits and Systems (ISCAS '92),* Vol.2, pp. 871- 874, 10-13 May 1992.

[34]. **Ngspice:** Mixed-level/Mixed-signal Circuit Simulator Based on Berkeley's Spice3f5. Ngspice project team, [online] Available: https://www.ngspice.org/.

[35]. **Orcad/PSpice:** OrCAD PSpice Designer Advanced Circuit Simulation and Analysis for Analog and Mixed-Signal Circuits, [online] Available: https://www.Orcad.com/.

[36]. **LTSpice:** Linear Technology Design Simulation and Device Models,[online] Available: http://www.linear.com/designtools/software/#LTspice.

[37]. K. Gajab and A. Chitturi, **"A Functional Analysis of SPICE Simulations and Parameters",** *International Journal of Science and Engineering Investigations,* Vol. 2, pp. 126-131, July 2013.

[38]. H. Park, H. Shim, **"What Is The Homotopy Method for A System of Nonlinear Equations (Survey)?"**, *J Appl Math & Computing,* Vol. 17, pp. 689-700, 2005.

[39]. D. Niu, K. Sako, G. Hu, Y. Inoue, **"A Globally Convergent Nonlinear Homotopy Method for MOS Transistor Circuits"**, *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, Vol. *E95-A*, pp. 2251-2260, Dec 2012.

[40]. D. Niu, Z. Jin, X. Wu, Y. Inoue, **"A Globally Convergent and Highly Efficient Homotopy Method for MOS Transistor Circuits"**, *In: IEEE 2012 International Conference on Computing and Convergence Technology (ICCCT),* pp. 1349-1352, Seoul, Korea, 3-5 Dec. 2012.

[41]. Z. JIN, X. WU, D. NIU ,Y. INOUE , **"Effective Implementation And Embedding Algorithm of CEPTA Method for Finding DC Operating Points"**, *IEEE Japan Technical Meeting on Electronic Circuits* , Vol. ECT-12, pp. 87-91, Oct. 2012.

[42]. Z. Jin, X. Wu, X. Guan, D. Niu, Y. Inoue, **"An Effective Ramping PTA Method For The DC Analysis Of Nonlinear Circuits"**, *The 28th International Technical Conference on Circuits/Systems, Computers and Communications, Yeosu, Korea,* 30-June, 3-july 2013.

[43]. X. Wu, Z. Jin, X. Lian, D. Niu, Y. Inoue, **"An Effective Switching Algorithm for the Damped Pseudo-Transient Analysis"**, *The 28th International Technical Conference on Circuits/Systems, Computers and Communications,* Yeosu, Korea, 30-June, 3-july 2013.

[44]. L.T. Watson, **"Globally Convergent Homotopy Algorithms for Nonlinear Systems of Equations",** *Springer, Nonlinear Dynamics,* Vol. 1, pp. 143–191, Mar. 1990.

[45]. H. Brachtendorf, R. Melville, P. Feldmann, S. Lampe, R. Laur: "Homotopy Method for Finding the Steady States of Oscillators", *IEEE Trans. on CAD of Integrated Circuits and Systems,* Vol. 33, pp. 867-878, Jun 2014.

[46]. T. Wang, H. D. Chiang, **"On the Global Convergence of a Class of Homotopy Methods for Nonlinear Circuits and Systems"**, *IEEE Trans. on Circuits and Systems,* Vol. 61-II, pp. 900-904, Nov. 2014.

[47]. H. Vázquez, L. Hernández, A. Sarmiento, R. Castañeda, A. Gallardo, **"Homotopy method with a formal stop criterion applied to circuit simulation",** *IEICE Electronic Express,* Vol. 21, pp. 1808-1815 , Aout 2011.

[48]. T. Conaghy, P. Palmers, M. Steyaert, E. Gielen, **"Variation-Aware Structural Synthesis of Analog Circuits via Hierarchical Building Blocks and Structural Homotopy"**. *IEEE Trans. on CAD of Integrated Circuits and Systems,* Vol. 28, pp. 1281-1294, Sept. 2009.

[49]. Y. Inoue, S. Kusanobu, K. Yamamura and M. Ando, **"An initial solution algorithm for globally convergent homotopy methods"**, *IEICE Trans.,* Vol. E87-A, no.4, pp.780-786, April 2004.

[50]. D. Martin, P. Wilsey, R. Hoekstra, E. Keiter, S. Hutchinson, T.V. Russo, L.J. Waters, **"Integrating multiple parallel simulation engines for mixed-technology parallel simulation",** *In: IEEE 2002 Proceedings. 35th Annual on Simulation Conference (SS'02) California: IEEE*, 14-18 April 2002.

[51]. J. Roy, R. Melville, **"Delivering Global DC Convergence for Large Mixed-Signal Circuits via Homotopy/Continuation Methods"**, *IEEE Transactions on Computer-Aided Design of Integrated Circuits,* Vol. 25, pp. 66-78, Nov. 2006.

[52]. T. Conaghy, E. Gielen, **"Globally Reliable Variation-Aware Sizing of Analog Integrated Circuits via Response Surfaces and Structural Homotopy"**, *IEEE Trans. on CAD of Integrated Circuits and Systems,* Vol. *28, pp. 1627-1640, Nov 2009.*

[53]. L. Allgower, K. Georg, **"Introduction to Numerical Continuation Methods"**, *SIAM, Philadelphia,* 2003.

[54]. F. Dhiabi, M. Boumehraz, **"Solving Nonlinear Equations by Applying Homotopy Method to Find Operating Point in Analog Circuit"**, *Conference on Electrical*

Engineering, ICEEB'14, 7-8 Dec. 2014.

[55]. F. Dhiabi, M. Boumehraz, **"Accelerate the Solving of Nonlinear Equations Using Homotopy Method: Application on Finding the Operating Point of complex circuits"**, *Turkish Journal of Electrical Engineering & Computer Sciences, Accept*. Jul. 2016.

[56]. M .Tadeusiewicz, S. Hałgas, **"Multiple Soft Fault Diagnosis of Nonlinear Circuits Using the Continuation Method"**, *Springer Journal of Electronic Testing*, Vol. 28, pp. 487-493. Apr. 2012.

[57]. C.P. Reames and A.N. Willson, **"Global Convergence of Newton's Method in the DC analysis of Single-Transistor Networks"**, *Electronics Letters,* Vol. 18, pp. 519-520, June 1982.

[58]. Shenggao Li, Brian Okoon, M.M. Hella, Maya Rubeiz, **"The implementation of a VHDL-AMS to SPICE converter"**, *Journal of VLSI Signal Processing, vol. 22, pp.113-121,* Mars 1999.

[59]. M. Brinson, and V. Kuznetsov, **"A new approach to compact semiconductor device modelling with Qucs Verilog-A analogue module synthesis"**, *in International Journal of Numerical Modelling: Electronic Networks, Devices and Fields,* 2016.

[60]. M. Brinson and V. Kuznetsov, **"Qucs equation-defined and Verilog-A RF device models for harmonic balance circuit simulation"**, *In Mixed Design of Integrated Circuits Systems (MIXDES), 22nd International Conference,* pp. 192-197, June 2015.

[61]. M. Darvishi, A. Barati, **"A third-order Newton-type Method to Solve Systems of Nonlinear Equations"**, *Applied Mathematics and Computation.* Vol. *187*, pp. 630–635, Apr. 2007

[62]. M. Noor, M. Waseem, **"Some Iterative methods for Solving a System of Nonlinear Equations"**, *Computers and Mathematics with Applications,* Vol. 57, pp. 101–106. Jan. 2009.

[63]. M. Waseem, M. Noor, K. Inayat, **"Efficient Method for Solving a System of Nonlinear Equations"**, *Applied Mathematics and Computation,* Vol. 275, pp. 134–146, 2016.

[64]. S. Abbasbandy, **"Improving Newton–Raphson Method for Nonlinear Equations by Modified Adomian Decomposition Method"**, *Appl. Math. Comput*, Vol. 145, pp. 887–893, 2003.

[65]. C. Chun, **"A New Iterative Method for Solving Nonlinear equations"**, *Appl. Math. Comput.* Vol. 178 (2), pp. 415–422, 2006.

[66]. C. Chun, **"Iterative Methods Improving Newton's Method by the Decomposition method"**, *Comput. Math. Appl,* Vol. 50, pp. 1559–1568, 2005.

[67]. M. Frontini, E. Sormani, **"Third-order Methods from Quadrature Formulae for Solving Systems of Nonlinear Equations"**, *Appl. Math. Comput,* Vol. 149, pp. 771–782, 2004.

[68]. M. Frontini, E. Sormani, **"Some Variants of Newton's Method with Third-order Convergence"**, *Appl. Math. Comput,* Vol. 140, pp. 419–426, 2003.

[69]. H.H.H. Homeier, **"On Newton-type Methods with Cubic Convergence"**, *J. Comput. Appl. Math,* Vol. 176, pp. 425–432, 2005.

[70]. S. Weerakoon, and G. I. Fernando, **"A variant of Newton's method with Accelerated Third-Order Convergence"**, *Applied Mathematics Letters,* Vol. 17, pp. 87-93, 2000.

[71]. M. A. Noor, and M. Waseem, **"Some Iterative Methods for Solving a System of Nonlinear Equations"**, *Computers and Mathematics with Applications,* Vol. 57, pp. 101-106, 2009.

[72]. J. Kou, Y. Li, and X. Wang, **"A Modification of Newton Method with Third-order Convergence"**, *Applied Mathematics and Computation,* Vol. 181 , pp. 1106-1111, 2006.

[73]. H. H. H. Homeier, **"A Modified Newton Method with Cubic Convergence: the Multivariate case"**, *Journal of Computational and Applied Mathematics,* Vol. 169, pp. 161-169, 2004.

[74]. M. Frontini, and E. Sormani, **"Third-order methods from quadrature formulae for solving systems of nonlinear equations"**, *Applied Mathematics and Computation* Vol. 149, pp. 771-782, 2004.

[75]. C. Chun, **"Construction of Newton-like Iteration Methods for Solving Nonlinear Equations"**, *Numerical Mathematics,* Vol. 104, pp. 297-315, 2006.

[76]. S. Abbasbandy, **"Improving Newton-Raphson Method for Nonlinear Equations by Modified Adomian Decomposition Method",** *Applied Mathematics and Comp,* Vol. 145, pp. 887-893, 2003.

[77]. W. Haijun, **"On New Third-order Convergent Iterative Formulas"**, *springer, Numer Algor*, Vol. 48, pp. 317–325, 2008.

[78]. Y. Saad and M. H. Schultz, **"GMRES: a Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems"** *SIAM Journal on Scientific and Statistical Computing*, Vol. 7, pp. 856–869, 1986.

[79]. Xyce Parallel Electronic Simulator: version 6.2. Sandia National Laboratories, [online] Available: https://xyce.sandia.gov/.

[80]. G. Baker, G. Boman, A. Heroux, R. Keiter, S. Rajamanickam, R. Schiek, **"Thornquist: Enabling Next-Generation Parallel Circuit Simulation with Trilinos"**, *Euro-Par Workshops*, pp. 315-323, Juni 2011.

[81]. A. Heroux, I. Rouson, **"Special Issue on the Trilinos Project"**, *Part 1 of 2. Scientific Programming*, Vol. 20, pp. 81-81, Fev. 2012.

[82]. A. Heroux, M. Willenbring, **"A New Overview of the Trilinos Project"**, *Scientific Programming*, Vol. 20, pp. 83-88, Fev. 2012.

[83]. A. Heroux, W. Rouson, **"Special issue on the Trilinos project"**, *Part 2 of 2. Scientific Programming*, Vol. 20, pp. 221-221, Mar. 2012.

[84]. F. Spotz **"PyTrilinos: Recent advances in the Python interface to Trilinos"**, *Scientific Programming*, Vol. 20, pp. 311-325, Mar. 2012.

[85]. A. Heroux, A. Bartlett, E. Howle, J. Hoekstra, J. Hu, G. Kolda, B. Lehoucq, R. Long, P. Pawlowski, T. Phipps, G. Salinger, H. Thornquist, S. Tuminaro, M. Willenbring, B. Williams, S. Stanley, **"An overview of the Trilinos project."**, *ACM Trans. Math. Softw*,Vol 31, pp. 397-423, Mar. 2005.

[86]. A. Henk, V. Vorst, **"Bi-CGSTAB: A Fast and Smoothly Converging Variant of Bi-CG for the Solution of Nonsymmetric Linear Systems"**, *SIAM J. Scientific Computing*, Vol. 13, pp. 631-644 , Feb. 1992.

[87]. S. Rui, C. Xu, **"A globally and locally superlinearly convergent inexact Newton-GMRES method for large-scale variational inequality problem"**, *Int. J. Comput. Math.* Vol. 91, pp. 578-587, Mar. 2001.

[88]. S. Abhyankar, J. Flueck, **"Real-Time Power System Dynamics Simulation Using a Parallel Block-Jacobi Preconditioned Newton-GMRES Scheme"**, *SC Companion*, Vol. 3, pp. 299-305, 2012

[89]. A. Nejat, C. Gooch, **"Effect of discretization order on preconditioning and convergence of a high-order unstructured Newton-GMRES solver for the Euler equations"** *J. Comput. Physics*, Vol. 227, pp. 2366-2386, Apr. 2008.

[90]. B. Dossou, R. Pierre, **"A Newton-GMRES Approach for the Analysis of the Postbuckling Behavior of the Solutions of the von Kármán Equations"**, *SIAM J. Scientific Computing*, Vol. 24, pp. 1994-2012, Jun 2003.

[91]. S. Bellavia, B. Morini, **"A Globally Convergent Newton-GMRES Subspace Method for Systems of Nonlinear Equations"**, *SIAM J. Scientific Computing*, Vol 23, pp 940-960, Mars 2001.

[92]. S. Bellavia, M. Macconi, B. Morini, **"A Hybrid Newton-GMRES Method for Solving Nonlinear Equations"**, *Second International Conference, NAA 2000, Numerical Analysis and Its Applications*, Rousse, Bulgaria, pp. 68-75, June 11-15, 2000.

[93]. P. Orkwis, H. George, **"A comparison of CGS preconditioning methods for Newton's method solvers"**, *Neural Parallel & Scientific Comp*, Vol 2, 177-194, Feb. 1994.

[94]. W. Schönauer, H. Häfner, R. Weiss, **"LINSOL, a Parallel Iterative Linear Solver Package of Generalized CG-Type for Sparse Matrices"**, *Proceedings of the Eighth SIAM Conference on Parallel Processing for Scientific Computing*, PPSC 1997, March 14-17, 1997.

[95]. R. Pethiyagoda, W.M. Cue, J. Moroney, M. Back, **"Jacobian-free Newton-Krylov Methods with GPU Acceleration for Computing Nonlinear Ship Wave patterns"**, *J. Comput. Physics*, Vol 269, pp. 297-313, 2014.

[96]. G. Chen, L. Chacón, A. Leibs, A. Knoll, T. Taitano, **"Fluid preconditioning for Newton-Krylov-based, fully implicit, electrostatic particle-in-cell simulations"**, *J. Comput. Physics,* Vol. 258, pp. 555-567, 2014.

[97]. R. Pethiyagoda, W. McCue, J. Moroney, M. Back, **"Jacobian-free Newton-Krylov methods with GPU acceleration for computing nonlinear ship wave patterns"**, *J. Comput. Physics* Vol. 269, pp. 297-313, 2014.

[98]. G. Fasano, M. Roma, **"Preconditioning Newton-Krylov Methods in Nonconvex Large Scale Optimization"**, *Comp. Opt. and Appl*, Vol. 56, pp. 253-290, Vol. 2013.

[99]. H. An, J. Wen, T. Feng, **"On finite difference approximation of a matrix-vector product in the Jacobian-free Newton-Krylov method"**, *J. Computational Applied Mathematics*, Vol. 236, 1399-1409, June 2011.

[100]. C. Shin, T. Darvishi, H. Kim, **"A comparison of the Newton-Krylov method with high order Newton-like methods to solve nonlinear systems",** *Applied Mathematics and Computation*, Vol. 217, pp. 3190-3198, July 2010.

[101]. S. Veldhuizen, C. Vuik, R. Kleijn, **"On projected Newton-Krylov solvers for instationary laminar reacting gas flows"**, *J. Comput. Physics* Vol. 229, pp. 1724-1738, May 2010.

[102]. J. Sánchez, M. Net, **"On the Multiple Shooting Continuation of Periodic orbits by Newton-Krylov Methods."**, *I. J. Bifurcation and Chaos*, Vol. 20, pp. 43-61, Jan 2010.

[103]. P. Lucas, A. Zuijlen, H. Bijl, **"Fast Unsteady Flow Computations with a Jacobian-Free Newton-Krylov Algorithm."**, *J. Comput. Physics*, Vol. 229, pp. 9201-9215, May. 2010.

[104]. J. Chen, L. Qi, **"Globally and Superlinearly Convergent Inexact Newton-Krylov Algorithms for Aolving Nonsmooth equations"**, *Numerical Lin. Alg. with Applic.*Vol. 17, pp. 155-174, Jan 2010.

[105]. V. Nguyen, C. Cheng, E. Hammack, S. Maier, **"Parallel Newton-Krylov solvers for modeling of a navigation lock filling system."**, *Proceedings of the International Conference on Computational Science, ICCS 2010, University of Amsterdam, The Netherlands,* , pp. 699- 707, 31-May ,2- June 2010.

[106]. V. Smirnov, V. Sá, **"On the linear convergence of Newton-Krylov methods.",** *Optimization Methods and Software,* Vol. 24, pp. 271-283, Fab 2009.

[107]. N. Brown, F. Walker, R. Wasyk, S. Woodward, **"On Using Approximate Finite Differences in Matrix-Free Newton-Krylov Methods."**, *SIAM J. Numerical Analysis,* Vol. 46, pp. 1892-1911, Apr. 2008.

[108]. Z. Li, C. Shi, **"A Quasi-Newton Preconditioned Newton-Krylov Method for Robust and Efficient Time-Domain Simulation of Integrated Circuits With Strong Parasitic Couplings."**, *IEEE Trans. on CAD of Integrated Circuits and Systems*, Vol. 25, pp. 2868-2881, Dec 2006.

[109]. Z. Li, C. Shi, **"A Quasi-Newton Preconditioned Newton-Krylov Method for**

Robust and Efficient Time-Domain Simulation of Integrated Circuits with Strong Parasitic Couplings." , *IEEE Proceedings of the 2006 Conference on Asia South Pacific Design Automation: ASP-DAC 2006,* Yokohama, Japan, pp. 402-407, 24-27 Jan. 2006.

[110]. J. Harrison, **"Krylov Subspace Accelerated Inexact Newton Method for Linear and Nonlinear Equations"**, *Journal of Computational Chemistry,* Vol. 25, pp. 328-334, Mar. 2004.

[111]. S. Sundar, K. Bhagavan, S. Prasad, **"Newton-Preconditioned Krylov Subspace Solvers for System of Nonlinear Equations: A Numerical Experiment."** *Appl. Math. Lett.* Vol. 14, pp. 195-200, Feb. 2001.

[112]. N. Brown, Y. Saad, **"Convergence Theory of Nonlinear Newton-Krylov Algorithms."**, *SIAM Journal on Optimization,* Vol. 4, pp. 297-330, Feb. 1994.

[113]. **"Python Programming Language"**, [online] Available: www.python.org

[114]. **"Pascal Programming Language by Using GUI Delphi XE8"**, [online] Available: https://www.embarcadero.com/fr/products/delphi

[115]. H.K. Gummel and H.C. Poon, **"An Integral Charge Control Model of Bipolar Transistors"**, *Bell Syst. Tech. J.*, Vol. 49, pp. 827–852, May–June 1970

[116]. M.K Mandal, B. Sarkar, **"Ring Oscillators: Caracteristics and Applications"**, *Indian Journal of Pure & Applied Physics* Vol. 48, pp. 136-145, Feb. 2010.

[117]. M. Halimi, K. Kemih, M. Ghanes, **"Circuit Simulation of an Analog Secure Communication based on Synchronized Chaotic Chua's System"**, *Appl. Math. Inf. Sci.* Vol. 4, pp.1509-1516 Jul. 2014.