FINAL PROJECT PRESENTATIONS
Spring 2004

Computing DC operating points of nonlinear circuit using homotopy methods

Renju S. Narayanan rsn@sfu.ca
Bozidar Vujicic bvujicic@cs.sfu.ca
Svetlana Vujicic svujicic@cs.sfu.ca

http://www.ensc.sfu.ca/research/cnl
School of Engineering Science
Simon Fraser University
Roadmap

- Introduction
- Implementation
- Results
- Conclusions
- References
Introduction

- Computing DC operating points is the first step in simulating nonlinear circuits.
- DC operating point computation involves solving a set of nonlinear equations that describe the circuit.
- Computational effort increases considerably with the complexity of the circuit.
- There are different methods to solve the nonlinear equations.
Different algorithms for solving nonlinear equations

- Newton’s method
- Powell’s algorithm
- Brown’s method
- Secant method
- Bisection
- Steepest Descent
- Trust Region
- Line search
- Continuation
- Homotopy
- Augmented Lagrange
- Bisection
- Reduced Gradient
- Tensor
# Available solvers 1

<table>
<thead>
<tr>
<th>Solver</th>
<th>Algorithm</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brown</td>
<td>Brown’s method</td>
<td>Lawrence Livermore National laboratory</td>
</tr>
<tr>
<td>QuasiA</td>
<td>Hybrid Powell (Newton/Trust Region)</td>
<td>Lawrence Livermore National laboratory</td>
</tr>
<tr>
<td>CHABIS</td>
<td>Characteristic Bisection</td>
<td>ACM TOMS</td>
</tr>
<tr>
<td>SOS</td>
<td>Brown’s method</td>
<td>NETLIB</td>
</tr>
<tr>
<td>HOMPACK</td>
<td>Homotopy</td>
<td>ACM TOMS/NETLIB</td>
</tr>
</tbody>
</table>
Available solvers 2

<table>
<thead>
<tr>
<th>Solver</th>
<th>Algorithm</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONTIN</td>
<td>Continuation</td>
<td>ACM TOMS/NETLIB</td>
</tr>
<tr>
<td>HYBRD</td>
<td>Hybrid Powell (Newton/Trust Region)</td>
<td>NETLIB</td>
</tr>
<tr>
<td>TENSOLVE</td>
<td>Tensor/Line Search</td>
<td>NETLIB</td>
</tr>
<tr>
<td>LANCELOT</td>
<td>Projected Gradient</td>
<td>NEOS</td>
</tr>
<tr>
<td>MINOS</td>
<td>Reduced Gradient</td>
<td>NEOS</td>
</tr>
</tbody>
</table>
Newton – Raphson method

- SPICE-like simulators compute DC operating points by using the Newton - Raphson method and its variants
  - robust
  - one solution to the problem
  - convergent when a good starting point is supplied
Newton-Raphson Method

\[ f(x^0) + f'(x^0)(x - x^0) \]

\[ f(x) \]

\[ x^* \quad x^2 \quad x^1 \quad x^0 \]
Ad hoc techniques

- Ad hoc techniques help DC convergence
- Natural parameter continuations
  - source-stepping algorithms
  - temperature-sweeping procedure
  - pseudo transient analysis
- Artificial parameter continuation
  - $G_{\text{min}}$ stepping
Traditional methods vs. homotopy methods

- Traditional methods for solving nonlinear algebraic equations
  - convergence difficulties
  - seldom find all the solutions
  - often fail
- Parameter embedding methods (homotopy or continuation methods)
  - resolve convergence difficulties
  - find multiple solutions
How to avoid these problems?
Continuously modulate problem whose answer is known into desired problem
Homotopy method

- Homotopy = paths leading to same place
- Nonlinear equations to be solved are of the form $F(x) = 0$, $F: \mathbb{R}^n \rightarrow \mathbb{R}^n, x \in \mathbb{R}^n \rightarrow \mathbb{R}^n$
- Embeds a parameter ($\lambda$) to $F(x)$
- $\lambda$ is the homotopy parameter
- $\lambda$ is varied from 0 to 1
- Construct homotopy equation $H$
  $$H(x, \lambda) = 0$$  $H: \mathbb{R}^{n+1} \rightarrow \mathbb{R}^n$
- Solving $H(x, 0) = 0$ is easy and it is the initial value of the homotopy path
- When $\lambda = 1$, $H(x, 1) = F(x)$
Homotopy method (2)

- Starting from the solution to $H(x,0)=0$, one follows a connected set of points $(x, \lambda)$ such that $H(x, \lambda) = 0$ until $\lambda = 1$

- Arc length continuation
  - $X$ and $\lambda$ are functions of arc length $s$ along a connected set of points
  - Homotopy function : $H(x(s), \lambda(s))=0$
  - Value of $x$ at the point $\lambda(s)=1$ is the solution
Homotopy method (3)

\[ F(x) = 0 \quad \text{and} \quad H(x, \lambda) = 0 \]

Adding an additional parameter makes the original problem easier to solve.
Homotopy functions

- **Fixed-point homotopy**
  \[ H(x, \lambda) = (1 - \lambda)G(x - a) + \lambda F(x) \]

- **Variable-stimulus homotopy**
  \[ H(x, \lambda) = (1 - \lambda)G(x - a) + F(x, \lambda) \]

- **Variable gain homotopy**
  \[ H(x, \lambda) = (1 - \lambda)G(x - a) + F(x, \lambda \alpha) \]

- **Hybrid homotopy (modified variable-stimulus homotopy)**
  \[ H(x, \lambda) = (1 - \lambda)G(x - a) + F(\lambda x) \]
Value for nonlinear circuit

- Continuation parameter in circuit’s nonlinear equations
- Choosing a good starting point is essential to ensure fast convergence of algorithm
  - solution to linear circuit
  - nonlinear circuit which have unique solution
Roadmap

- Introduction
- Implementation
- Results
- Conclusions
- References
Implementation

- Formulating equations of form $F(x) = 0, \ x \in \mathbb{R}^n$
  - Parser
- Homotopy equation used
  $H(x,\lambda) = (1- \lambda) \text{Gleak} (x-a) + \lambda F(x)$
- Solving equations
- Circuits implemented
  - Schmitt trigger circuit
  - Chua’s circuit
  - Flip flop circuit
Parser

- Utility written in C++ by Edward Chan
- Input to the parser is a file containing circuit description in Spice format
- Parser creates nodal or modified nodal equations and their Jacobians
- Parser program has been modified to work smoothly
- The parser output will have to be modified slightly in order to use the equations in Matlab
- The equations were verified and found to be correct
Matlab Implementation

- Equations obtained from the Parser are solved in Matlab
- Homotopy used: \( H(x,\lambda) = (1- \lambda) \text{Gleak} (x-a) + \lambda F(X) \)
- Conditions
  \[
  H(x,0) = \text{Gleak} (x-a) \\
  H(x,1)= F(x)
  \]
- Trajectory \( y(s) = \lambda(s) x(s) \)
  - \( S \) is the arc length
- The solution is trajectory of the homotopy function through numerical integration
Matlab Code

- Matlab code consists of three functions
  - init
  - Pchomotopy
  - Deval
- The starting point of the trajectory is calculated in the function `init`
- The values of $\lambda$ and $s$ are calculated in the function `pchomotopy`
- Matlab code implements a variable-step predictor-corrector ODE solver
- Solver allows user to determine the order of the predictor-collector method
- The Jacobian of the homotopy function is the input for the solver
- Function `deval` evaluates derivative from homotopy Jacobian
PCHOMOTOPY

- Solves nonlinear system of equations using ODE based homotopy with variable-order variable-step predictor-corrector technique

- Input:
  - J-string containing name of homotopy Jacobian
  - x0-solution of homotopy when lambda is 0
  - sfinal-final value of arc length s
  - t 01 determines desired accuracy
  - k-order of predictor corrector method

- Output:
  - sout-returned integration arc length points (column vector)
  - xout-returned solutions along trajectory, one solution per tout value
  - NJaceval-number of evaluations of the homotopy Jacobian J
  - xsolout-returned solutions of f (x)=0, when lambda =1
  - lambdaout-returned values of embedded parameter along trajectory
DEVAL

- Evaluates derivative from homotopy Jacobian

**Inputs:**
- J-string containing name of user supplied homotopy Jacobian
- y-current value of homotopy trajectory
- ypold-previous calculated derivative from homotopy Jacobian

**Outputs:**
- yp-derivative calculated from homotopy Jacobian
- Bad-boolean, determines if there were problems during derivative calculations
Newton-Raphson (NR) Method

- Consists of linearizing the system.
- Want to solve \( f(x) = 0 \) → Replace \( f(x) \) with its linearized version and solve.

\[
f(x) = f(x^*) + \frac{df}{dx}(x^*)(x - x^*) \quad \text{Taylor Series}
\]

\[
f(x^{k+1}) = f(x^k) + \frac{df}{dx}(x^k)(x^{k+1} - x^k)
\]

\[
\Rightarrow x^{k+1} = x^k - \left[ \frac{df}{dx}(x^k) \right]^{-1} f(x^k) \quad \text{Iteration function}
\]

Note: at each step need to evaluate \( f \) and \( f' \)
Predictor corrector methods

- Predictor-corrector method proceed by extrapolating a polynomial fit to the derivative from previous points to the new points then using this interpolate the derivative.
- Combination of explicit and implicit technique.
- Forward Euler methods as predictor and Adams Moulton as corrector equation.

$$y_{n+1}^p = y_n + hf(y_n, t_n)$$  \hspace{1cm} \text{predictor}  \\

$$y_{n+1} = y_n + \frac{h}{2} \left[ f(y_{n+1}^p, t_{n+1}) + f(y_n, t_n) \right]$$  \hspace{1cm} \text{corrector}  \\

Implicit term for AM is replaced with value of f evaluated at predictor, hence explicit method.
Forward Euler methods

- \( t_n \) the time at nth time-step
- \( y_n \) computed solution at the nth time-step
- \( h \) step-size
- \( y_{n+1} = y_n + hf(y_n, t_n) \) Explicit forward Euler method
- Based on truncated Taylor series expansion

\[
y(t_n + h) \equiv y_{n+1} = y(t_n) + h \frac{dy}{dt} \bigg|_{t_n} + O(h^2) = y_n + hf(y_n, t_n) + O(h^2)
\]

- Local Truncation Error induced due to truncation of the Taylor series
- First order technique, higher LTE than higher order tech.
Adams methods

- Adams method is based on approximating the integrand with a polynomial within the interval \((t_n, t_{n+1})\)

\[
y_{n+1} = y_n + \int_{t_n}^{t_{n+1}} \frac{dy}{dt} dt = y_n + \int_{t_n}^{t_{n+1}} f(y, t) dt
\]

- Adams-Bashforth, explicit type, only conditionally stable

\[
y_{n+1} = y_n + \frac{h}{2}(3f(y_n, t_n) - f(y_{n-1}, t_{n-1}))
\]

- Adams-Moulton, implicit type, every time step have to solve equ.

\[
y_{n+1} = y_n + \frac{h}{2}(f(y_{n+1}, t_{n+1}) + f(y_n, t_n))
\]

- First order of AB and AM are simply FE and BE
P-th order use
P-1 order polynomial results

Figure 7: t-points at which order p Adams-Bashforth and Adams-Moulton formulas interpolate f.
Runge-Kutta Methods

- Class of methods which uses information on slope at more than one point to extrapolate the solution to the future time step
- Using trial step at the midpoint of interval to cancel out lower-order error terms
Roadmap

- Introduction
- Implementation
- Results
- Conclusions
- References
Schmitt Trigger circuit
Schmitt trigger output (Matlab)
### Schmitt trigger output voltage values

#### Results from Matlab

<table>
<thead>
<tr>
<th>voltages</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>V(1)</td>
<td>0.71026</td>
<td>1.7659</td>
<td>8.6308</td>
</tr>
<tr>
<td>V(2)</td>
<td>0.67246</td>
<td>0.69174</td>
<td>0.96455</td>
</tr>
<tr>
<td>V(3)</td>
<td>10</td>
<td>7.2396</td>
<td>1.0382</td>
</tr>
<tr>
<td>V(4)</td>
<td>0.71024</td>
<td>1.4899</td>
<td>1.7962</td>
</tr>
<tr>
<td>V(5)</td>
<td>1.5</td>
<td>1.5</td>
<td>1.5</td>
</tr>
<tr>
<td>V(6)</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

#### Results from Hspice

<table>
<thead>
<tr>
<th>voltages</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>V(1)</td>
<td>0.7128394</td>
<td>1.7694</td>
<td>8.632</td>
</tr>
<tr>
<td>V(2)</td>
<td>0.6753985</td>
<td>0.694778</td>
<td>0.96460</td>
</tr>
<tr>
<td>V(3)</td>
<td>10</td>
<td>7.2084</td>
<td>1.038</td>
</tr>
<tr>
<td>V(4)</td>
<td>0.7128395</td>
<td>1.4902</td>
<td>1.7931</td>
</tr>
<tr>
<td>V(5)</td>
<td>1.5</td>
<td>1.5</td>
<td>1.5</td>
</tr>
<tr>
<td>V(6)</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>
Chua’s circuit

R1 = 1K
R2 = R3 = 4K
R4 = 5K
R5 = R8 = 30K
R6 = R7 = 0.5K
R9 = R10 = 10.1K
R11 = R12 = 4K
R13 = R14 = 30K
V1 = 10v  V2 = 2v
Vcc = 12v
Output voltages and currents of Chua’s circuit
Voltage at node 8
Chua’s voltage values

<table>
<thead>
<tr>
<th>Node voltages</th>
<th>Matlab values</th>
<th>Hspice values</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>V(1)</td>
<td>1.7718</td>
<td>1.7823</td>
</tr>
<tr>
<td>V(2)</td>
<td>-8.2282</td>
<td>-8.2177</td>
</tr>
<tr>
<td>V(3)</td>
<td>-1.1066</td>
<td>-2.5572</td>
</tr>
<tr>
<td>V(4)</td>
<td>1.4645</td>
<td>7.7505</td>
</tr>
<tr>
<td>V(5)</td>
<td>1.4645</td>
<td>1.4715</td>
</tr>
<tr>
<td>V(6)</td>
<td>0.36887</td>
<td>1.8035</td>
</tr>
<tr>
<td>V(7)</td>
<td>0.36887</td>
<td>0.37062</td>
</tr>
<tr>
<td>V(8)</td>
<td>1.3881</td>
<td>1.4298</td>
</tr>
<tr>
<td>V(9)</td>
<td>1.3881</td>
<td>1.3982</td>
</tr>
<tr>
<td>V(10)</td>
<td>10.458</td>
<td>3.3447</td>
</tr>
<tr>
<td>V(11)</td>
<td>0.89338</td>
<td>-0.55715</td>
</tr>
<tr>
<td>V(12)</td>
<td>10.693</td>
<td>10.523</td>
</tr>
<tr>
<td>V(13)</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>V(14)</td>
<td>12</td>
<td>12</td>
</tr>
</tbody>
</table>
Flip Flop circuit

[Image of a Flip Flop circuit diagram]

May 15, 2001
Flip flop circuit output
## Flip flop values

### From Matlab

<table>
<thead>
<tr>
<th>Value 1</th>
<th>Value 2</th>
<th>Value 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.52757</td>
<td>0.47974</td>
<td>0.006819</td>
</tr>
<tr>
<td>0.0084418</td>
<td>0.47325</td>
<td>0.5324</td>
</tr>
<tr>
<td>0.0084414</td>
<td>0.47325</td>
<td>0.5324</td>
</tr>
<tr>
<td>0.52757</td>
<td>0.47974</td>
<td>0.006819</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td>12</td>
</tr>
</tbody>
</table>

### Results from Hspice

<table>
<thead>
<tr>
<th>Value 1</th>
<th>Value 2</th>
<th>Value 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.522017</td>
<td>0.46887</td>
<td>0.0075409</td>
</tr>
<tr>
<td>0.0075409</td>
<td>0.46887</td>
<td>0.522018</td>
</tr>
<tr>
<td>0.0075409</td>
<td>0.46887</td>
<td>0.522017</td>
</tr>
<tr>
<td>0.52201</td>
<td>0.46887</td>
<td>0.0075409</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td>12</td>
</tr>
</tbody>
</table>
Roadmap

- Introduction
- Implementation
- Results
- Conclusions
- References
Conclusions

- The existing code for Parser was modified to give correct nodal equations for the nonlinear circuit.
- The Matlab implementation of homotopy methods was verified using Schmitt trigger, Chua’s and Flip-Flop circuits. The output values from Matlab was compared to the output from Hspice. The two outputs were similar.
- Solving homotopy methods using Matlab takes a lot of computing time.
References


- E. Chan, Lj. Trajkovic, "Documentation on the Parser program" Fall 1996

- Heath Hofmann, “EECS 290N:Final Project Report”


- W. Horia, Lj. Trajkovic“ New numerical algorithms for simulating electronic circuits” Spring 1997
References
