PEETER JOOT PEETERJOOT@PROTONMAIL.COM

MODELING OF MULTIPHYSICS SYSTEMS

MODELING OF MULTIPHYSICS SYSTEMS

PEETER JOOT PEETERJOOT@PROTONMAIL.COM

Notes and problems from UofT ECE1254H 2014

February 2019 – version V0.1.8

Peeter Joot peeterjoot@protonmail.com: *Modeling of Multiphysics Systems,* Notes and problems from UofT ECE1254H 2014, © February 2019

COPYRIGHT

DOCUMENT VERSION

Version V0.1.8

Sources for this notes compilation can be found in the github repository
git@github.com:peeterjoot/latex-notes-compilations.git
The last commit (Feb/11/2019), associated with this pdf was
dbb4592312d86a8f4e52ab46f48844c58dc6a89a

Dedicated to:
Aurora and Lance, my awesome kids, and
Sofia, who not only tolerates and encourages my studies, but is also awesome enough to think that math is sexy.

PREFACE

This document is based on my lecture notes for the Fall 2014, University of Toronto Modeling of Multiphysics course (ECE1254H), taught by Professor P. Triverio.

Official course description: "The course deals with the modeling and simulation of physical systems. It introduces the fundamental techniques to generate and solve the equations of a static or dynamic system. Special attention is devoted to complexity issues and to model order reduction methods, presented as a systematic way to simulate highly-complex systems with acceptable computational cost. Examples from multiple disciplines are considered, including electrical/electromagnetic engineering, structural mechanics, fluid-dynamics. Students are encouraged to work on a project related to their own research interests."

Topics:

- Automatic generation of system equations (Tableau method, modified nodal analysis).
- Solution of linear and nonlinear systems (LU decomposition, conjugate gradient method, sparse systems, Newton-Raphson method).
- Solution of dynamical systems (Euler and trapezoidal rule, accuracy, stability).
- Model order reduction of linear systems (proper orthogonal decomposition, Krylov methods, truncated balanced realization, stability/dissipativity enforcement).
- Modeling from experimental data (system identification, the Vector Fitting algorithm, enforcement of stability and dissipativity).
- If time permits, an overview of numerical methods to solve partial differential equations (Boundary element method, finite elements, FDTD).

Recommended texts include

- [8].

THIS DOCUMENT IS REDACTED. THE PROBLEM SET SOLUTIONS, MATLAB SOURCE CODE LINKS, AND FINAL PROJECT RELATED CONTENT IS NOT VISIBLE. PLEASE EMAIL ME FOR THE FULL VERSION IF YOU ARE NOT TAKING ECE1254.

This document contains:

- Plain old lecture notes. These mirror what was covered in class, possibly augmented with additional details.
- Personal notes exploring details that were not clear to me from the lectures, or from the texts associated with the lecture material.
- Assigned problems. Like anything else take these as is. I have attempted to either correct errors or mark them as such.
- Final report on an implementation of the Harmonic Balance method (a group project assignment). That work, including the code, the report content and presentation, was a collaborative effort between Mike Royle and myself.
- Links to Matlab function implementations associated with the problem sets.

My thanks go to Professor Triverio for teaching this course, to Ahmed Dorrah for supplying a copy of his (lecture 8) notes on the conjugate gradient method, and to Mike Royle for our Harmonic Balance collaboration.

Peeter Joot peeterjoot@protonmail.com

CONTENTS

LIST OF FIGURES

Part I

NOTES AND PROBLEMS

1 NODAL ANALYSIS

1.1 IN SLIDES

A review of systematic nodal analysis for a basic resistive circuit was outlined in slides, with a subsequent attempt to show how many similar linear systems can be modeled as circuits so that the same toolbox can be applied. This included blood flow through a body (and blood flow to the brain), a model of antenna interference in a portable phone, heat conduction in a one dimensional conductor under a heat lamp, and a few other systems.

This discussion reminded me of the joke where the farmer, the butcher and the physicist are all invited to talk at a beef convention. After meaningful and appropriate talks by the farmer and the butcher, the physicist gets his chance, and proceeds with "We begin by modeling the cow as a sphere, ...". The ECE equivalent of that appears to be a Kirchhoff circuit problem.

1.2 MECHANICAL STRUCTURES EXAMPLE

Continuing the application of circuits like linear systems to other systems, consider a truss system as illustrated in fig. 1.1, or in the simpler similar system of fig. 1.2.

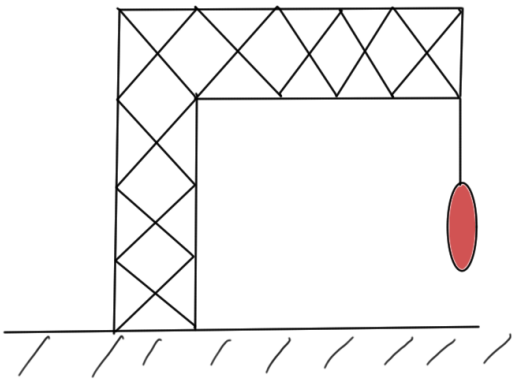


Figure 1.1: A static loaded truss configuration.

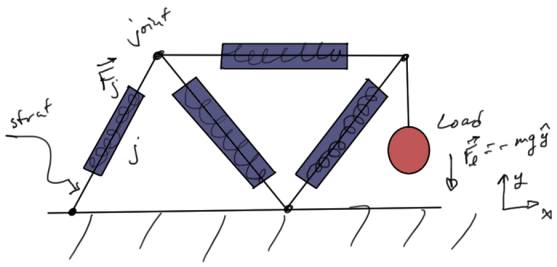


Figure 1.2: Simple static load.

The unknowns are

- positions of the joints after deformation (x_i, y_i).
- force acting on each strut $\mathbf{F}_j = (F_{j,x}, F_{j,y})$.

The constitutive equations, assuming static conditions (steady state, no transients)

- Load force. $\mathbf{F}_\mathrm{L} = (F_{\mathrm{L},x}, F_{\mathrm{L},y}) = (0, -mg)$.
- Strut forces. Under static conditions the total resulting force on the strut is zero, so $\mathbf{F}'_j = -\mathbf{F}_j$. For this problem it is redundant to label forces on both ends, so the labeled end of the object is marked with an asterisk as in fig. 1.3.

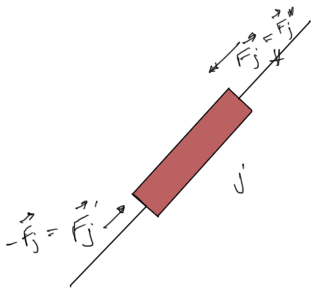


Figure 1.3: Strut model.

Consider a simple case One strut as in fig. 1.4.

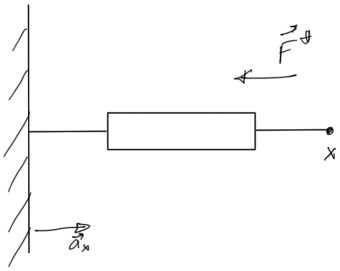


Figure 1.4: Very simple static load.

constant, describes the beam elasticity, given

$$\mathbf{F}^* = -\mathbf{a}_x \epsilon \left(L - L_0 \right) \tag{1.1}$$

unloaded length $L = |x^* - 0|$, given

The constitutive law for a general strut as in fig. 1.5 is

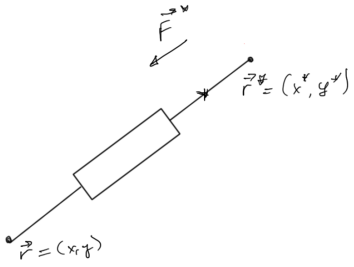


Figure 1.5: Strut force diagram.

The force is directed along the unit vector

$$\mathbf{e} = \frac{\mathbf{r}^* - \mathbf{r}}{|\mathbf{r}^* - \mathbf{r}|}, \tag{1.2}$$

and has the form

$$\mathbf{F}^* = -\mathbf{e}\epsilon \left(L - L_0 \right). \tag{1.3}$$

The value ϵ may be related to Hooks' constant, and L_0 is given by

$$L = |\mathbf{r}^* - \mathbf{r}| = \sqrt{(x^* - x)^2 + (y^* - y)^2}. \tag{1.4}$$

Observe that the relation between $\mathbf{F}^*$ and position is *nonlinear*!

Treatment of this system will be used as the prototype for the handling of other nonlinear systems.

Returning to the simple static system, and introducing force and joint labels as in fig. 1.6, the conservation law, a balance of forces, can be examined.

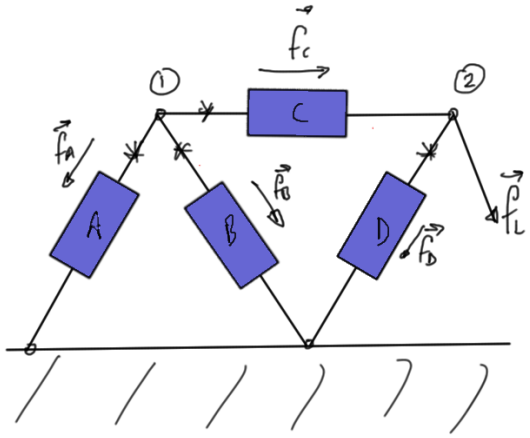


Figure 1.6: Strut system.

- At joint 1:

$$\mathbf{f}_A + \mathbf{f}_B + \mathbf{f}_C = 0 \tag{1.5}$$

or

$$\begin{aligned} \mathbf{f}_{A,x} + \mathbf{f}_{B,x} + \mathbf{f}_{C,x} &= 0 \\ \mathbf{f}_{A,y} + \mathbf{f}_{B,y} + \mathbf{f}_{C,y} &= 0 \end{aligned} \tag{1.6}$$

- At joint 2:

$$-\mathbf{f}_C + \mathbf{f}_D + \mathbf{f}_L = 0 \tag{1.7}$$

or

$$\begin{aligned} -\mathbf{f}_{C,x} + \mathbf{f}_{D,x} + \mathbf{f}_{L,x} &= 0 \\ -\mathbf{f}_{C,y} + \mathbf{f}_{D,y} + \mathbf{f}_{L,y} &= 0 \end{aligned} \tag{1.8}$$

There are two equivalences

- Force $\leftrightarrow$ Current.
- Force balance equation $\leftrightarrow$ KCL

1.3 ASSEMBLING SYSTEM EQUATIONS AUTOMATICALLY. NODE/BRANCH METHOD

Consider the sample circuit of fig. 1.7.

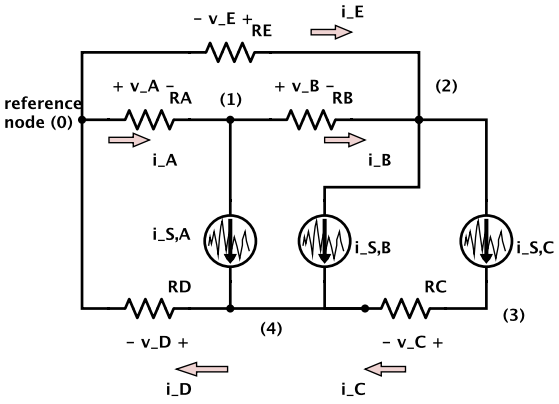


Figure 1.7: Sample resistive circuit.

Step 1. Choose unknowns: For this problem, take

- node voltages: V_1, V_2, V_3, V_4
- branch currents: i_A, i_B, i_C, i_D, i_E

No additional labels are required for the source current sources. A reference node is always introduced, given the node number zero.

For a circuit with N nodes, and B resistors, there will be $N-1$ unknown node voltages and B unknown branch currents , for a total number of $N-1+B$ unknowns.

Step 2. Conservation equations: KCL

- 0: $i_A + i_E - i_D = 0$
- 1: $-i_A + i_B + i_{S,A} = 0$
- 2: $-i_B + i_{S,B} - i_E + i_{S,C} = 0$
- 3: $i_C - i_{S,C} = 0$
- 4: $-i_{S,A} - i_{S,B} + i_D - i_C = 0$

Grouping unknown currents, this is

- 0: $i_A + i_E - i_D = 0$
- 1: $-i_A + i_B = -i_{S,A}$
- 2: $-i_B - i_E = -i_{S,B} - i_{S,C}$
- 3: $i_C = i_{S,C}$
- 4: $i_D - i_C = i_{S,A} + i_{S,B}$

Note that one of these equations is redundant (sum 1-4). In a circuit with N nodes, are are at most $N-1$ independent KCLs. In matrix form

$$\begin{bmatrix} -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 \end{bmatrix} \begin{bmatrix} i_A \\ i_B \\ i_C \\ i_D \\ i_E \end{bmatrix} = \begin{bmatrix} -i_{S,A} \\ -i_{S,B} - i_{S,C} \\ i_{S,C} \\ i_{S,A} + i_{S,B} \end{bmatrix} \tag{1.9}$$

This first matrix of ones and minus ones is called the incidence matrix A. This matrix has B columns and $N-1$ rows. The matrix of known currents is called $\mathbf{I}_S$, and the branch currents called $\mathbf{I}_B$. That is

$$A\mathbf{I}_B = \mathbf{I}_S. \tag{1.10}$$

Observe that there is a plus and minus one in all columns except for those columns impacted by the neglect of the reference node current conservation equation.

Algorithm for filling A In the input file, to describe a resistor of fig. 1.8, you'll have a line of the form

Rlabel n_1 n_2 **value**

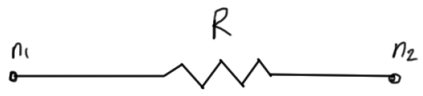


Figure 1.8: Resistor node convention.

The algorithm to process resistor lines is

Algorithm 1.1: Resistor line handling.

$A \leftarrow 0$
$ic \leftarrow 0$
for all resistor lines **do**
  $ic \leftarrow ic + 1$, adding one column to A
  if $n_1 != 0$ **then**
    $A(n_1, ic) \leftarrow +1$
  end if
  if $n_2 != 0$ **then**
    $A(n_2, ic) \leftarrow -1$
  end if
end for

Algorithm for filling $\mathbf{I}_S$ Current sources, as in fig. 1.9, a line will have the specification

Ilabel n_1 n_2 **value**

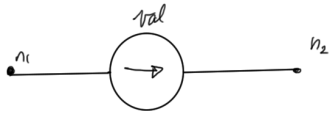


Figure 1.9: Current source conventions.

Algorithm 1.2: Current line handling.

$\mathbf{I}_S = \text{zeros}(N-1, 1)$
for all current lines **do**
  $\mathbf{I}_S(n_1) \leftarrow \mathbf{I}_S(n_1) - \text{value}$
  $\mathbf{I}_S(n_2) \leftarrow \mathbf{I}_S(n_2) + \text{value}$
end for

Step 3. Constitutive equations:

$$\begin{bmatrix} i_A \\ i_B \\ i_C \\ i_D \\ i_E \end{bmatrix} = \begin{bmatrix} 1/R_A & & & & \\ & 1/R_B & & & \\ & & 1/R_C & & \\ & & & 1/R_D & \\ & & & & 1/R_E \end{bmatrix} \begin{bmatrix} v_A \\ v_B \\ v_C \\ v_D \\ v_E \end{bmatrix} \tag{1.11}$$

Or

$$\mathbf{I}_B = \alpha \mathbf{V}_B, \tag{1.12}$$

where $\mathbf{V}_B$ are the branch voltages, not unknowns of interest directly. That can be written

$$\begin{bmatrix} v_A \\ v_B \\ v_C \\ v_D \\ v_E \end{bmatrix} = \begin{bmatrix} -1 & & & \\ 1 & -1 & & \\ & & 1 & -1 \\ & & & 1 \\ & -1 & & \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} \tag{1.13}$$

Observe that this is the transpose of A, so

$$\mathbf{V}_B = A^T \mathbf{V}_N. \tag{1.14}$$

Solving

- KCLs: $A\mathbf{I}_B = \mathbf{I}_S$

- constitutive: $\mathbf{I}_B = \alpha \mathbf{V}_B \implies \mathbf{I}_B = \alpha A^T \mathbf{V}_N$
- branch and node voltages: $\mathbf{V}_B = A^T \mathbf{V}_N$

In block matrix form, this is

$$\begin{bmatrix} A & 0 \\ I & -\alpha A^T \end{bmatrix} \begin{bmatrix} \mathbf{I}_B \\ \mathbf{V}_N \end{bmatrix} = \begin{bmatrix} \mathbf{I}_S \\ 0 \end{bmatrix}. \tag{1.15}$$

Is it square? This can be observing to be the case after noting that there are

- $N-1$ rows in A , and B columns.
- B rows in I.
- $N-1$ columns.

1.4 NODAL ANALYSIS

Avoiding branch currents can reduce the scope of the computational problem. Consider the same circuit fig. 1.10, this time introducing only node voltages as unknowns

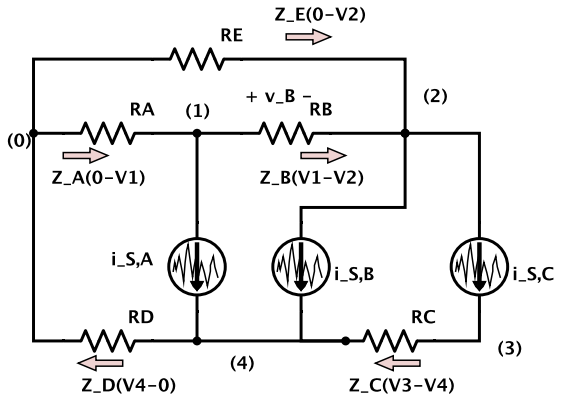


Figure 1.10: Resistive circuit with current sources.

Unknowns: node voltages: $V_1, V_2, \cdots V_4$
Equations are KCL at each node except 0.

1. $\frac{V_1-0}{R_A} + \frac{V_1-V_2}{R_B} + i_{S,A} = 0$

2. $\frac{V_2-0}{R_\mathrm{E}} + \frac{V_2-V_1}{R_\mathrm{B}} + i_{\mathrm{S,B}} + i_{\mathrm{S,C}} = 0$
3. $\frac{V_3-V_4}{R_\mathrm{C}} - i_{\mathrm{S,C}} = 0$
4. $\frac{V_4-0}{R_\mathrm{D}} + \frac{V_4-V_3}{R_\mathrm{C}} - i_{\mathrm{S,A}} - i_{\mathrm{S,B}} = 0$

In matrix form this is

$$\begin{bmatrix} \frac{1}{R_\mathrm{A}} + \frac{1}{R_\mathrm{B}} & -\frac{1}{R_\mathrm{B}} & 0 & 0 \\ -\frac{1}{R_\mathrm{B}} & \frac{1}{R_\mathrm{B}} + \frac{1}{R_\mathrm{E}} & 0 & 0 \\ 0 & 0 & \frac{1}{R_\mathrm{C}} & -\frac{1}{R_\mathrm{C}} \\ 0 & 0 & -\frac{1}{R_\mathrm{C}} & \frac{1}{R_\mathrm{C}} + \frac{1}{R_\mathrm{D}} \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \end{bmatrix} = \begin{bmatrix} -i_{\mathrm{S,A}} \\ -i_{\mathrm{S,B}} - i_{\mathrm{S,C}} \\ i_{\mathrm{S,C}} \\ i_{\mathrm{S,A}} + i_{\mathrm{S,B}} \end{bmatrix} \tag{1.16}$$

Introducing the nodal matrix G, this is written as

$$G\mathbf{V}_N = \mathbf{I}_\mathrm{S} \tag{1.17}$$

There is a recurring pattern in the nodal matrix, designated the stamp for the resistor of value R between nodes n_1 and n_2

$$\begin{array}{c} \\ n_1 \\ n_2 \end{array} \begin{array}{c} \begin{array}{cc} n_1 & n_2 \end{array} \\ \begin{bmatrix} \frac{1}{R} & -\frac{1}{R} \\ -\frac{1}{R} & \frac{1}{R} \end{bmatrix} \end{array}, \tag{1.18}$$

containing a set of rows and columns for each of the node voltages n_1, n_2.

Note that some care is required to use this nodal analysis method since the invertible relationship $i = V/R$ is required. Short circuits $V = 0$, and voltage sources such as $V = 5$ also cannot be handled directly. The mechanisms to deal with differential terms like inductors will be discussed later.

Recap of node branch equations The node branch equations were

- KCL: $A\mathbf{I}_\mathrm{B} = \mathbf{I}_\mathrm{S}$
- Constitutive: $\mathbf{I}_\mathrm{B} = \alpha A^\mathrm{T}\mathbf{V}_N$,
- Nodal equations: $\overbrace{A\alpha A^\mathrm{T}}^{G}\mathbf{V}_N = \mathbf{I}_\mathrm{S}$

where $\mathbf{I}_B$ was the branch currents, A was the incidence matrix, and $\alpha = \begin{bmatrix} \frac{1}{R_1} & & \\ & \frac{1}{R_2} & \\ & & \ddots \end{bmatrix}$.

The stamp can be observed in the multiplication of the contribution for a single resistor. The incidence matrix has the form $G = A\alpha A^T$

$$\begin{aligned} G &= \begin{matrix} \\ n_1 \\ n_2 \end{matrix} \overset{\downarrow}{\begin{bmatrix} +1 \\ -1 \end{bmatrix}} \begin{bmatrix} \frac{1}{R} \end{bmatrix} \overset{\begin{matrix} n_1 & n_2 \end{matrix}}{\begin{bmatrix} +1 & -1 \end{bmatrix}} \\ &= \begin{matrix} \\ n_1 \\ n_2 \end{matrix} \overset{\begin{matrix} n_1 & n_2 \end{matrix}}{\begin{bmatrix} \frac{1}{R} & -\frac{1}{R} \\ -\frac{1}{R} & \frac{1}{R} \end{bmatrix}} \end{aligned} \tag{1.19}$$

Theoretical facts Noting that $(AB)^T = B^T A^T$, it is clear that the nodal matrix $G = A\alpha A^T$ is symmetric

$$\begin{aligned} G^T &= \left(A\alpha A^T\right)^T \\ &= \left(A^T\right)^T \alpha^T A^T \\ &= A\alpha A^T \\ &= G \end{aligned} \tag{1.20}$$

1.5 MODIFIED NODAL ANALYSIS (MNA)

Modified nodal analysis (MNA), eliminates the branch currents for the resistive circuit elements, and is the method used to implement software such as spice. To illustrate the method, consider the same circuit, augmented with an additional voltage sources as in fig. 1.11. This method can also accomodate voltage sources, provided an unknown current source is also introduced for each voltage source circuit element. The unknowns are

- node voltages $(N-1)$: $V_1, V_2, \cdots V_5$
- branch currents for selected components (K): $i_{S,C}, i_{S,D}$

Compared to standard nodal analysis, two less unknowns for this system are required. The equations are

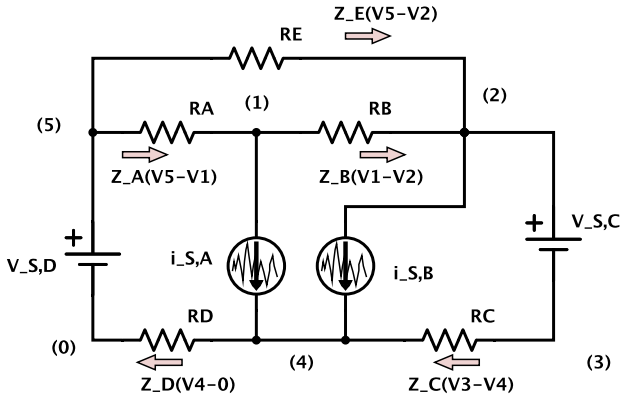


Figure 1.11: Resistive circuit with current and voltage sources.

1. $-\frac{V_5-V_1}{R_A}+\frac{V_1-V_2}{R_B}+i_{S,A}=0$
2. $\frac{V_2-V_5}{R_E}+\frac{V_2-V_1}{R_B}+i_{S,B}-i_{S,C}=0$
3. $i_{S,C}+\frac{V_3-V_4}{R_C}=0$
4. $\frac{V_4-0}{R_D}+\frac{V_4-V_3}{R_C}-i_{S,A}-i_{S,B}=0$
5. $\frac{V_5-V_2}{R_E}+\frac{V_5-V_1}{R_A}-i_{S,D}=0$

Put into giant matrix form, this is

$$
\begin{array}{c}
\overset{G}{} \\
-A_V^T
\end{array}
\left[\begin{array}{ccccc|c}
Z_A+Z_B & -Z_B & . & . & -Z_A & \\
-Z_B & Z_B-Z_E & . & . & -Z_E & -1 \\
. & . & Z_C & -Z_C & . & +1 \\
. & . & -Z_C & Z_C+Z_D & . & \\
-Z_A & -Z_E & & & Z_A+Z_E & -1 \\
\hline
 & +1 & -1 & & & \\
 & & & & 1 &
\end{array}\right]
\overset{A_V}{}
\left[\begin{array}{c} V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \\ i_{S,C} \\ i_{S,D} \end{array}\right]
=
\left[\begin{array}{c} -i_{S,A} \\ -i_{S,B} \\ 0 \\ i_{S,A}+i_{S,B} \\ 0 \\ V_{S,C} \\ V_{S,D} \end{array}\right]
\tag{1.21}
$$

Call the extension to the nodal matrix G, the voltage incidence matrix A_V.

Review Additional unknowns are added for

- branch currents for voltage sources
- all elements for which it is impossible or inconvenient to write $i = f(v_1, v_2)$. Imagine, for example, a component as illustrated in fig. 1.12.

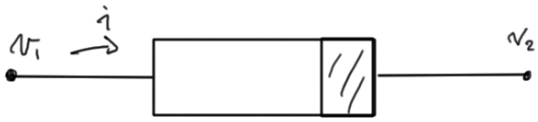


Figure 1.12: Variable voltage device.

$$v_1 - v_2 = 3i^2 \tag{1.22}$$

- any current which is controlling dependent sources, as in fig. 1.13.

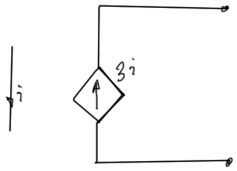


Figure 1.13: Current controlled device.

- Inductors

$$v_1 - v_2 = L\frac{di}{dt}. \tag{1.23}$$

2 SOLVING LARGE SYSTEMS

The goal is to solve linear systems of the form

$$M\mathbf{x} = \mathbf{b}, \tag{2.1}$$

possibly with thousands of elements.

2.1 GAUSSIAN ELIMINATION

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{bmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{bmatrix} \end{array} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \tag{2.2}$$

It's claimed for now, to be seen later, that back substitution is the fastest way to arrive at the solution, less computationally complex than completion the diagonalization.

Steps

$$(1) \cdot \frac{M_{21}}{M_{11}} \implies \begin{bmatrix} M_{21} & \frac{M_{21}}{M_{11}} M_{12} & \frac{M_{21}}{M_{11}} M_{13} \end{bmatrix} \tag{2.3}$$

$$(2) \cdot \frac{M_{31}}{M_{11}} \implies \begin{bmatrix} M_{31} & \frac{M_{31}}{M_{11}} M_{32} & \frac{M_{31}}{M_{11}} M_{33} \end{bmatrix} \tag{2.4}$$

This gives

$$\begin{bmatrix} M_{11} & M_{12} & M_{13} \\ 0 & M_{22} - \frac{M_{21}}{M_{11}} M_{12} & M_{23} - \frac{M_{21}}{M_{11}} M_{13} \\ 0 & M_{32} - \frac{M_{31}}{M_{11}} M_{32} & M_{33} - \frac{M_{31}}{M_{11}} M_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 - \frac{M_{21}}{M_{11}} b_1 \\ b_3 - \frac{M_{31}}{M_{11}} b_1 \end{bmatrix}. \tag{2.5}$$

Here the M_{11} element is called the pivot. Each of the M_{j1}/M_{11} elements is called a multiplier. This operation can be written as

$$\begin{bmatrix} M_{11} & M_{12} & M_{13} \\ 0 & M_{22}^{(2)} & M_{23}^{(3)} \\ 0 & M_{32}^{(2)} & M_{33}^{(3)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2^{(2)} \\ b_3^{(2)} \end{bmatrix}. \tag{2.6}$$

Using $M_{22}^{(2)}$ as the pivot this time, form

$$\begin{bmatrix} M_{11} & M_{12} & M_{13} \\ 0 & M_{22}^{(2)} & M_{23}^{(3)} \\ 0 & 0 & M_{33}^{(3)} - \frac{M_{32}^{(2)}}{M_{22}^{(2)}} M_{23}^{(2)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 - \frac{M_{21}}{M_{11}} b_1 \\ b_3 - \frac{M_{31}}{M_{11}} b_1 - \frac{M_{32}^{(2)}}{M_{22}^{(2)}} b_2^{(2)} \end{bmatrix}. \tag{2.7}$$

2.2 LU DECOMPOSITION

Through Gaussian elimination, the system has been transformed from

$$Mx = b \tag{2.8}$$

to

$$Ux = y. \tag{2.9}$$

The Gaussian transformation written out in the form $U\mathbf{x} = b$ is

$$U\mathbf{x} = \begin{bmatrix} 1 & 0 & 0 \\ -\frac{M_{21}}{M_{11}} & 1 & 0 \\ \frac{M_{32}^{(2)}}{M_{22}^{(2)}} \frac{M_{21}}{M_{11}} - \frac{M_{31}}{M_{11}} & -\frac{M_{32}^{(2)}}{M_{22}^{(2)}} & 1 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}. \tag{2.10}$$

As a verification observe that the operation matrix K^{-1}, where $K^{-1}U = M$ produces the original system

$$\begin{bmatrix} 1 & 0 & 0 \\ \frac{M_{21}}{M_{11}} & 1 & 0 \\ \frac{M_{31}}{M_{11}} & \frac{M_{32}^{(2)}}{M_{22}^{(2)}} & 1 \end{bmatrix} \begin{bmatrix} U_{11} & U_{12} & U_{13} \\ 0 & U_{22} & U_{23} \\ 0 & 0 & U_{33} \end{bmatrix} \mathbf{x} = \mathbf{b} \tag{2.11}$$

Using this LU decomposition is generally superior to standard Gaussian elimination, since it can be used for many different **b** vectors, and cost no additional work after the initial factorization.

The steps are

$$\begin{aligned} b &= Mx \\ &= L\,(Ux) \\ &\equiv Ly. \end{aligned} \tag{2.12}$$

The matrix equation $Ly = b$ can now be solved by substituting first y_1, then y_2, and finally y_3. This is called forward substitution.

The final solution is

$$Ux = y, \tag{2.13}$$

using back substitution.

Note that this produced the vector y as a side effect of performing the Gaussian elimination process.

Example 2.1: Numeric LU factorization.

Looking at a numeric example is helpful to get a better feel for LU factorization before attempting a Matlab implementation, as it strips some of the abstraction away.

$$M = \begin{bmatrix} 5 & 1 & 1 \\ 2 & 3 & 4 \\ 3 & 1 & 2 \end{bmatrix}. \tag{2.14}$$

This matrix was picked to avoid having to think of selecting the right pivot row when performing the LU factorization. The first two operations give

$$\begin{matrix} \\ (r_2 \to r_2 - \frac{2}{5} r_1) \\ (r_3 \to r_3 - \frac{3}{5} r_1) \end{matrix} \begin{bmatrix} 5 & 1 & 1 \\ 0 & 13/5 & 18/5 \\ 0 & 2/5 & 7/5 \end{bmatrix}. \tag{2.15}$$

The row operations (left multiplication) that produce this matrix are

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -3/5 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ -2/5 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ -2/5 & 1 & 0 \\ -3/5 & 0 & 1 \end{bmatrix}. \tag{2.16}$$

These operations happen to be commutative and also both invert simply. The inverse operations are

$$\begin{bmatrix} 1 & 0 & 0 \\ 2/5 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 3/5 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 2/5 & 1 & 0 \\ 3/5 & 0 & 1 \end{bmatrix}. \tag{2.17}$$

In matrix form the elementary matrix operations that produce the first stage of the Gaussian reduction are

$$\begin{bmatrix} 1 & 0 & 0 \\ -2/5 & 1 & 0 \\ -3/5 & 0 & 1 \end{bmatrix} \begin{bmatrix} 5 & 1 & 1 \\ 2 & 3 & 4 \\ 3 & 1 & 2 \end{bmatrix} = \begin{bmatrix} 5 & 1 & 1 \\ 0 & 13/5 & 18/5 \\ 0 & 2/5 & 7/5 \end{bmatrix}. \tag{2.18}$$

Inverted that is

$$\begin{bmatrix} 5 & 1 & 1 \\ 2 & 3 & 4 \\ 3 & 1 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 2/5 & 1 & 0 \\ 3/5 & 0 & 1 \end{bmatrix} \begin{bmatrix} 5 & 1 & 1 \\ 0 & 13/5 & 18/5 \\ 0 & 2/5 & 7/5 \end{bmatrix}. \tag{2.19}$$

This is the first stage of the LU decomposition, although the U matrix is not yet in upper triangular form. With the pivot row in the desired position already, the last row operation to perform is

$$r_3 \to r_3 - \frac{2/5}{5/13} r_2 = r_3 - \frac{2}{13} r_2. \tag{2.20}$$

The final stage of this Gaussian reduction is

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -2/13 & 1 \end{bmatrix} \begin{bmatrix} 5 & 1 & 1 \\ 0 & 13/5 & 18/5 \\ 0 & 2/5 & 7/5 \end{bmatrix} = \begin{bmatrix} 5 & 1 & 1 \\ 0 & 13/5 & 18/5 \\ 0 & 0 & 11/13 \end{bmatrix} = U, \tag{2.21}$$

so the desired lower triangular matrix factor is

$$\begin{bmatrix} 1 & 0 & 0 \\ 2/5 & 1 & 0 \\ 3/5 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 2/13 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 2/5 & 1 & 0 \\ 3/5 & 2/13 & 1 \end{bmatrix} = L. \tag{2.22}$$

A bit of Matlab code easily verifies that the above manual computation recovers $M = LU$

```
l = [ 1 0 0 ; 2/5 1 0 ; 3/5 2/13 1 ] ;
u = [ 5 1 1 ; 0 13/5 18/5 ; 0 0 11/13 ] ;
m = l * u
```

Example 2.2: Numeric LU factorization with pivots.

Proceeding with a factorization where pivots are required, does not produce an LU factorization that is the product of a lower triangular matrix and an upper triangular matrix. Instead what is found is what looks like a permutation of a lower triangular matrix with an upper triangular matrix. As an example, consider the LU reduction of

$$M\mathbf{x} = \begin{bmatrix} 0 & 0 & 1 \\ 2 & 0 & 4 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}. \tag{2.23}$$

Since r_2 has the biggest first column value, that is the row selected as the pivot

$$M\mathbf{x} \to M'\mathbf{x}' = \begin{bmatrix} 2 & 0 & 4 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_2 \\ x_1 \\ x_3 \end{bmatrix}. \tag{2.24}$$

This permutation can be expressed algebraically as a row permutation matrix operation

$$M' = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} M. \tag{2.25}$$

With the pivot permutations out of the way, the row operations remaining for the Gaussian reduction of this column are

$$\begin{aligned} r_2 &\to r_2 - \frac{0}{2} r_1 \\ r_3 &\to r_3 - \frac{1}{2} r_1 \end{aligned}, \tag{2.26}$$

which gives

$$\begin{aligned} M_1 &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1/2 & 0 & 1 \end{bmatrix} M' \\ &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1/2 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 & 4 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 2 & 0 & 4 \\ 0 & 0 & 1 \\ 0 & 1 & -1 \end{bmatrix}. \end{aligned} \tag{2.27}$$

Application of one more permutation operation gives the desired upper triangular matrix

$$\begin{aligned} U &= M_1' \\ &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 2 & 0 & 4 \\ 0 & 0 & 1 \\ 0 & 1 & -1 \end{bmatrix} \\ &= \begin{bmatrix} 2 & 0 & 4 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix}. \end{aligned} \tag{2.28}$$

This new matrix operator applies to the permuted vector

$$\mathbf{x}'' = \begin{bmatrix} x_2 \\ x_3 \\ x_1 \end{bmatrix}. \tag{2.29}$$

The matrix U has been constructed by the following row operations

$$U = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1/2 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} M. \tag{2.30}$$

$LU = M$ is sought, or

$$L \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1/2 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} M = M, \tag{2.31}$$

or

$$
\begin{aligned}
L &= \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1/2 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \\
&= \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1/2 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \\
&= \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 1/2 & 1 & 0 \end{bmatrix}.
\end{aligned} \tag{2.32}
$$

The LU factorization attempted does not appear to produce a lower triangular factor, but a permutation of a lower triangular factor?

When such pivoting is required it isn't obvious, at least to me, how to do the clever LU algorithm that outlined in class. How can the operations be packed into the lower triangle when there is a requirement to actually have to apply permutation matrices to the results of the last iteration?

It seems that a LU decomposition of M cannot be performed, but an LU factorization of PM, where P is the permutation matrix for the permutation $2, 3, 1$ that was applied to the rows during the Gaussian operations.

Checking that LU factorization:

$$
PM = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 2 & 0 & 4 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 4 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.33}
$$

The elementary row operation to be applied is

$$
r_2 \to r_2 - \frac{1}{2} r_1, \tag{2.34}
$$

for

$$(PM)_1 = \begin{bmatrix} 2 & 0 & 4 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.35}$$

The LU factorization is therefore

$$PM = LU = \begin{bmatrix} 1 & 0 & 0 \\ 1/2 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 & 4 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.36}$$

Observe that this can also be written as

$$M = \left(P^{-1}L\right) U. \tag{2.37}$$

The inverse permutation is a $3, 1, 2$ permutation matrix

$$P^{-1} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}. \tag{2.38}$$

It can be observed that the product $P^{-1}L$ produces the not-lower-triangular matrix factor found earlier in eq. (2.32).

Example 2.3: Final illustration of the LU algorithm with pivots by example.

Two previous examples of LU factorizations were given. I found one more to be the key to understanding how to implement this as a Matlab algorithm, required for exercise 2.2.

A matrix that contains both pivots and elementary matrix operations is

$$M = \begin{bmatrix} 0 & 0 & 2 & 1 \\ 0 & 0 & 1 & 1 \\ 2 & 0 & 2 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \tag{2.39}$$

The objective is to apply a sequence of row permutations or elementary row operations to M that put M into upper triangular form, while also tracking all the inverse operations. When no permutations were required to produce U, then a factorization $M = L'U$ is produced where L' is lower triangular.

The row operations to be applied to M are

$$U = L_k^{-1} L_{k-1}^{-1} \cdots L_2^{-1} L_1^{-1} M, \tag{2.40}$$

with

$$L' = L_0 L_1 L_2 \cdots L_{k-1} L_k \tag{2.41}$$

Here $L_0 = I$, the identity matrix, and L_i^{-1} is either a permutation matrix interchanging two rows of the identity matrix, or it is an elementary row operation encoding the operation $r_j \to r_j - M r_i$, where r_i is the pivot row, and $r_j, j > i$ are the rows that the Gaussian elimination operations are applied to.

For this example matrix, the M_{11} value cannot be used as the pivot element since it is zero. In general, the row with the biggest absolute value in the column should be used. In this case that is row 3. The first row operation is therefore a $1, 3$ permutation. For numeric stability, use

$$L_1^{-1} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2.42}$$

which gives

$$M \to L_1^{-1} M = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 2 & 1 \\ 0 & 0 & 1 & 1 \\ 2 & 0 & 2 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 2 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 2 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}. \tag{2.43}$$

Computationally, avoiding the matrix multiplication that achieve this permutation is desired. Instead just swap the two rows in question.

The inverse of this operation is the same permutation, so for L' the first stage computation is

$$L \sim L_0 L_1 = L_1. \tag{2.44}$$

As before, a matrix operation would be very expensive. When the application of the permutation matrix is from the right, it results in an exchange of columns $1, 3$ of the L_0 matrix (which happens to be identity at this point). So the matrix operation can be done as a column exchange directly using submatrix notation.

Now proceed down the column, doing all the non-zero row elimination operations required. In this case, there is only one operation todo

$$r_4 \to r_4 - \frac{1}{2} r_1. \tag{2.45}$$

This has the matrix form

$$L_2^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -1/2 & 0 & 0 & 1 \end{bmatrix}. \tag{2.46}$$

The next stage of the U computation is

$$M \to L_2^{-1} L_1^{-1} M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -1/2 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 & 2 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 2 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 2 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 2 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}. \quad (2.47)$$

Again, this is not an operation that should be done as a matrix operation. Instead act directly on the rows in question with eq. (2.45).

Note that the inverse of this matrix operation is very simple. An amount $r_1/2$ has been subtracted from r_4, so to invert this all that is required is adding back $r_1/2$. That is

$$L_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1/2 & 0 & 0 & 1 \end{bmatrix}. \quad (2.48)$$

Observe that when this is applied from the right to $L_0 L_1 \to L_0 L_1 L_2$, the interpretation is a column operation

$$c_1 \to c_1 + m c_4, \quad (2.49)$$

In general, if application of the row operation

$$r_j \to r_j - m r_i, \quad (2.50)$$

to the current state of the matrix U, requires application of the operation

$$r_i \to r_i + m r_j, \quad (2.51)$$

to the current state of the matrix L'.

The next step is to move on to reduction of column 2, and for that only a permutation operation is required

$$L_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \tag{2.52}$$

Application of a $2,4$ row interchange to U, and a $2,4$ column interchange to L' gives

$$M \rightarrow \begin{bmatrix} 2 & 0 & 2 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 2 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}. \tag{2.53}$$

The final operation is a regular row operation

$$r_4 \rightarrow r_4 - \frac{1}{2} r_3, \tag{2.54}$$

with matrix

$$L_4^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1/2 & 1 \end{bmatrix} \tag{2.55}$$

The composite permutation performed so far is

$$P = L_3 L_1 I. \tag{2.56}$$

This should also be computed by performing row interchanges, not matrix multiplication.

To solve the system

$$M\mathbf{x} = L'U\mathbf{x} = \mathbf{b}, \tag{2.57}$$

solve the equivalent problem

$$PL'U\mathbf{x} = P\mathbf{b}, \tag{2.58}$$

To do this let $\mathbf{y} = U\mathbf{x}$, for

$$PL'\mathbf{y} = P\mathbf{b}. \tag{2.59}$$

The matrix $L = PL'$ is lower triangular, as P contained all the permutations that applied along the way (FIXME: this is a statement, not a proof, and not obvious).
The system

$$L\mathbf{y} = P\mathbf{b}, \tag{2.60}$$

can now be solved using forward substitution, after which the upper triangular system

$$\mathbf{y} = U\mathbf{x}, \tag{2.61}$$

can be solved using only back substitution.

2.3 PROBLEMS

Exercise 2.1 Modified Nodal Analysis.

a. Write a Matlab routine `[G,b]=NodalAnalysis(filename)` that generates the modified nodal analysis (MNA) equations

$$\mathbf{G}\mathbf{x} = \mathbf{b} \tag{2.62}$$

from a text file (netlist) that describes an electrical circuit made of resistors, independent current sources, independent voltage sources, voltage-controlled

voltage sources. For the netlist, we use the widely-adopted SPICE syntax. For each resistor, the file will contain a line in the form

Rlabel node1 node2 value

where "value" is the resistance value. Current sources are specified with the line

Ilabel node1 node2 DC value

and current flows from node1 to node2. Note that DC is just a keyword. A voltage source connected between the nodes node+ and node- is specified by the line

Vlabel node+ node- DC value

where node+ and node- identify, respectively, the node where the "positive" and "negative" terminal is connected to. A voltage-controlled voltage source, connected between the nodes node+ and node-, is specified by the line

Elabel node+ node- nodectrl+ nodectrl- gain

The controlling voltage is between the nodes nodectrl+ and nodectrl-, and the last argument is the source gain.

b. Explain how did you include the controlled source into the modified nodal analysis formulation. Which general rule can be given to "stamp" a voltage-controlled voltage source into MNA?

c. Consider the circuit shown in the figure fig. 2.1. Write an input file for the netlist parser developed in the previous point, and use it to generate the matrices G and b for the circuit. The operational amplifiers have an input resistance of $1M\Omega$? and a gain of 10^6 . Model them with a resistor and a voltage-controlled voltage source. Use the Matlab command \ to solve the linear system eq. (2.62) and determine the voltage $V_\circ$ shown in the figure.

d. Implement your own LU factorization routine. Repeat the previous point using your own LU factorization and forward/backward substitution routines to solve the circuit equations. Report the computed $V_\circ$.

Answer for Exercise 2.1

PROBLEM SET RELATED MATERIAL REDACTED IN THIS DOCUMENT.PLEASE FEEL FREE TO EMAIL ME FOR THE FULL VERSION IF YOU AREN'T TAKING ECE1254.[redacted]

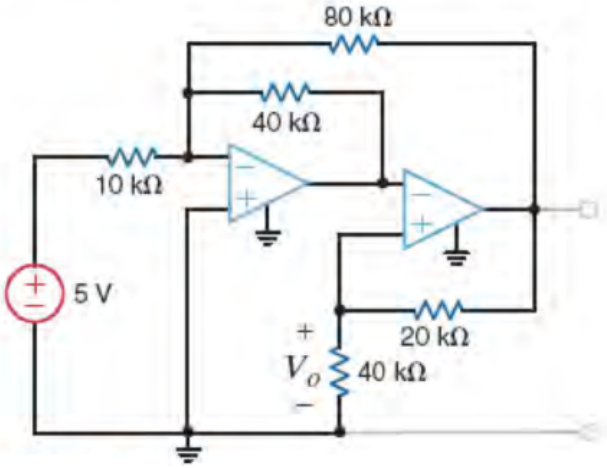


Figure 2.1: Circuit to solve.

[redacted]

[redacted] END-REDACTION

Exercise 2.2 Resistor Mesh.

a. Write a small Matlab function that generates a netlist for a network made by:· · ·
   1. $N \times N$ square grid of resistors of value R, where N is the number of resistors per edge. The grid nodes are numbered from 1 to $(N+1)^2$
   2. voltage source $V = 1V$ connected between node 1 and ground
   3. three current sources, each one connected between a randomly selected node of the grid and the reference node. The source current flows from the grid node to the reference node. Choose their value randomly between 10 mA and 100 mA;

   Generate the modified nodal analysis equations (1) for a grid with $N = 50, R = 0.2\Omega$ and solve them with your LU routine to find the node voltages. Plot the result with the Matlab command `surf()`.
b. Plot the CPU time taken by your system solver (LU factorization + forward/backward substitution) as a function of the size n of the 3 modified nodal analysis matrix **G**. Note: do not exploit the sparsity of the matrix.
c. Determine experimentally how the CPU time scales for large n. Fit the CPU times you observe with a power law like $t_{\text{cpu}}(n) \simeq Kn^{\alpha}$ where K is a constant in order to determine the exponent α.
d. Comment on the result. Discuss your findings in light of what we discussed in class

Answer for Exercise 2.2

PROBLEM SET RELATED MATERIAL REDACTED IN THIS DOCUMENT.PLEASE FEEL FREE TO EMAIL ME FOR THE FULL VERSION IF YOU AREN'T TAKING ECE1254.[redacted]

[redacted] END-REDACTION

Exercise 2.3 Heat conduction.

a. In this problem we will examine the heat conducting bar basic example, but will consider the case of a "leaky" bar to give you practice developing a numerical technique for a new physical problem. With an appropriate input file, the simulator you developed in problem 1 can be used to solve numerically the one-dimensional Poisson equation with arbitrary boundary conditions. The Poisson equation can be used to determine steady-state temperature distribution in a heat-conducting bar, as in

$$\frac{\partial^2 T(x)}{\partial x^2} = \frac{\kappa_a}{\kappa_m}\left(T(x) - T_0\right) - \frac{H(x)}{\kappa_m} \tag{2.63}$$

where $T(x)$ is the temperature at a point in space x, $H(x)$ is the heat generated at x, κ_m is the thermal conductivity along the metal bar, and κ_a is the thermal conductivity from the bar to the surrounding air. The temperature T_0 is the surrounding air temperature. The ratio κ_a/κ_m will be small as heat moves much more easily along the bar than dissipates from the bar into the surrounding air.

Use your Matlab simulator to numerically solve the above Poisson equation for $T(x), x \in [0,1]$, given $H(x) = 50\sin^2(2\pi x)$ for $x \in [0,1], \kappa_a = 0.001, and \kappa_m = 0.1$. In addition, assume the ambient air temperature is $T_0 = 400$, and $T(0) = 250$ and $T(1) = 250$. The boundary conditions at $x = 0$ and $x = 1$ model heat sink connections to a cool metal cabinet at both ends of the package. That is, it is assumed that the heat sink connections will insure both ends of the package are fixed at near room temperature.
To represent equation eq. (2.63) as a circuit, you must first discretize your bar along the spatial variable x in small sections of length Δx, and approximate the derivatives using a finite difference formula, e.g.,

$$\frac{\partial^2 T(x)}{\partial x^2} \approx \frac{1}{\Delta x}\left(\frac{T(x+\Delta x) - T(x)}{\Delta x} - \frac{T(x) - T(x-\Delta x)}{\Delta x}\right) \tag{2.64}$$

Then, interpret the discretized equation as a KCL using the electrothermal analogy where temperature corresponds to node voltage, and heat flow to current. Draw the equivalent circuit you obtained.

b. Plot $T(x)$ in $x \in [0,1]$.

c. In your numerical calculation, how did you choose Δx? Justify the choice of Δx.

d. Now use your simulator to numerically solve the above equation for $T(x), x \in [0,1]$, given $H(x) = 50$ for $x \in [0,1], \kappa_a = 0.001$, and $\kappa_m = 0.1$. In addition, assume the ambient air temperature is $T_0 = 400$, and there is not heat flow at both ends of the bar. The zero heat flow boundary condition at $x = 0$ and $x = 1$ implies that there are no heat sinks at the ends of the package. Since heat flow is given by

$$\text{heatflow} = \kappa\frac{\partial T}{\partial x}. \tag{2.65}$$

zero heat flow at the boundaries means that $T(0)$ and $T(1)$ are unknown, but $\partial T/\partial x(0) = 0$, and $\partial T/\partial x(1) = 0$.
Given the zero-heat-flow boundary condition, what is the new equivalent circuit? How the different boundary condition maps into the equivalent circuit?

e. Plot the new temperature profile.

f. Explain the temperature distributions that you obtained from a physical standpoint.

Answer for Exercise 2.3

PROBLEM SET RELATED MATERIAL REDACTED IN THIS DOCUMENT.PLEASE FEEL FREE TO EMAIL ME FOR THE FULL VERSION IF YOU AREN'T TAKING ECE1254.[redacted]. END-REDACTION

3 NUMERICAL ERRORS AND CONDITIONING

3.1 STRICT DIAGONAL DOMINANCE

Related to a theorem on one of the slides:

Definition 3.1: Strictly diagonally dominant.

A matrix $[M_{ij}]$ is strictly diagonally dominant if

$$|M_{ii}| > \sum_{j \neq i} |M_{ij}| \quad \forall i \tag{3.1}$$

For example, the stamp matrix

$$\begin{array}{c} \\ i \\ j \end{array} \begin{array}{c} \begin{array}{cc} i & j \end{array} \\ \begin{bmatrix} \frac{1}{R} & -\frac{1}{R} \\ -\frac{1}{R} & \frac{1}{R} \end{bmatrix} \end{array} \tag{3.2}$$

is not strictly diagonally dominant. For row i this strict dominance can be achieved by adding a reference resistor

$$\begin{array}{c} \\ i \\ j \end{array} \begin{array}{c} \begin{array}{cc} i & j \end{array} \\ \begin{bmatrix} \frac{1}{R_0} + \frac{1}{R} & -\frac{1}{R} \\ -\frac{1}{R} & \frac{1}{R} \end{bmatrix} \end{array} \tag{3.3}$$

However, even with strict dominance, there will be trouble with ill posed (perturbative) systems.

Round off error examples with double precision

$$(1 - 1) + \pi 10^{-17} = \pi 10^{-17}, \tag{3.4}$$

vs.

$$\left(1 + \pi 10^{-17}\right) - 1 = 0. \tag{3.5}$$

This is demonstrated by

```
#include <stdio.h>
#include <math.h>

// produces:
// 0 3.14159e-17
int main()
{
   double d1 = (1 + M_PI * 1e-17) - 1 ;
   double d2 = M_PI * 1e-17 ;

   printf( "%g %g\n", d1, d2 ) ;

   return 0 ;
}
```

Note that a union and bitfield [5] can be useful for exploring double precision representation.

3.2 EXPLORING UNIQUENESS AND EXISTENCE

For a matrix system $\mathbf{M}x = \mathbf{b}$ in column format, with

$$\begin{bmatrix} \mathbf{M}_1 & \mathbf{M}_2 & \cdots & \mathbf{M}_N \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} = \mathbf{b}. \tag{3.6}$$

This can be written as

$$\overset{\text{weight}}{\boxed{x_1}}\mathbf{M}_1 + \overset{\text{weight}}{\boxed{x_2}}\mathbf{M}_2 + \cdots x_N\mathbf{M}_N = \mathbf{b}. \tag{3.7}$$

Linear dependence means

$$y_1\mathbf{M}_1 + y_2\mathbf{M}_2 + \cdots y_N\mathbf{M}_N = 0, \tag{3.8}$$

or $M\mathbf{y} = 0$.

Table 3.1: Solution space.

	$\mathbf{b} \in \text{span}\{M\_i\}$	$\mathbf{b} \notin \text{span}\{M\_i\}$
columns of M linearly independent	$\mathbf{x}$ exists and is unique	No solution
columns of M linearly dependent	$\mathbf{x}$ exists. Infinitely many solutions	No solution

With a linear dependency an additional solution, given solution $\mathbf{x}$ is $\mathbf{x}^1 = \mathbf{x} + \alpha y$. This becomes relevant for numerical processing since for a system $M\mathbf{x}^1 = \mathbf{b}$ a $\alpha M\mathbf{y}$ can often be found, for which

$$M\mathbf{x} + \alpha M\mathbf{y} = \mathbf{b}, \tag{3.9}$$

where $\alpha M\mathbf{y}$ is of order 10^{-20}.

3.3 PERTURBATION AND NORMS

Consider a perturbation to the system $M\mathbf{x} = \mathbf{b}$

$$(M + \delta M)(\mathbf{x} + \delta\mathbf{x}) = \mathbf{b}. \tag{3.10}$$

Some vector norms

- L_1 norm

$$\|\mathbf{x}\|_1 = \sum_i |x_i| \tag{3.11}$$

- L_2 norm

$$\|\mathbf{x}\|_2 = \sqrt{\sum_i |x_i|^2} \tag{3.12}$$

- L_∞ norm

$$\|\mathbf{x}\|_\infty = \max_i |x_i|. \tag{3.13}$$

These are illustrated for $\mathbf{x} = (x_1, x_2)$ in fig. 3.1.

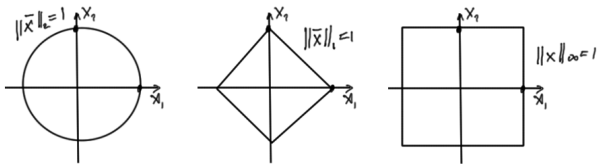


Figure 3.1: Some vector norms.

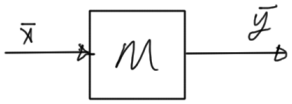


Figure 3.2: Matrix as a transformation.

3.4 MATRIX NORM

A matrix operation $\mathbf{y} = M\mathbf{x}$ can be thought of as a transformation as in fig. 3.2.

The 1-norm for a Matrix is defined as

$$\|M\| = \max_{\|\mathbf{x}\|_1=1} \|M\mathbf{x}\|\,, \tag{3.14}$$

and the matrix 2-norm is defined as

$$\|M\|_2 = \max_{\|\mathbf{x}\|_2=1} \|M\mathbf{x}\|_2\,. \tag{3.15}$$

4 SINGULAR VALUE DECOMPOSITION, AND CONDITIONING NUMBER

4.1 SINGULAR VALUE DECOMPOSITION

Recall that the matrix norm of M, for the system $\mathbf{y} = M\mathbf{x}$ was defined as

$$\|M\| = \max_{\|\mathbf{x}\|=1} \|M\mathbf{x}\| . \tag{4.1}$$

The L_2 norm will typically be used, resulting in a matrix norm of

$$\|M\|_2 = \max_{\|\mathbf{x}\|_2=1} \|M\mathbf{x}\|_2 . \tag{4.2}$$

It can be shown that

$$\|M\|_2 = \max_i \sigma_i(M), \tag{4.3}$$

where $\sigma_i(M)$ are the (SVD) singular values.

Definition 4.1: Singular value decomposition (SVD).

Given $M \in \mathbb{R}^{m \times n}$, a factoring of M can be found with the form

$$M = U\Sigma V^{\mathrm{T}}, \tag{4.4}$$

where U and V are orthogonal matrices such that $U^{\mathrm{T}}U = 1$, and $V^{\mathrm{T}}V = 1$, and

$$\Sigma = \begin{bmatrix} \sigma_1 & & & & & & \\ & \sigma_2 & & & & & \\ & & \ddots & & & & \\ & & & \sigma_r & & & \\ & & & & 0 & & \\ & & & & & \ddots & \\ & & & & & & 0 \end{bmatrix} \tag{4.5}$$

The values $\sigma_i, i \leq \min(n, m)$ are called the singular values of M. The singular values are subject to the ordering

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq 0 \tag{4.6}$$

If r is the rank of M, then the σ_r above is the minimum non-zero singular value (but the zeros are also called singular values).

Observe that the condition $U^{\mathrm{T}}U = 1$ is a statement of orthonormality. In terms of column vectors $\mathbf{u}_i$, such a product written out explicitly is

$$\begin{bmatrix} \mathbf{u}_1^{\mathrm{T}} \\ \mathbf{u}_2^{\mathrm{T}} \\ \vdots \\ \mathbf{u}_m^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_m \end{bmatrix} = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix}. \tag{4.7}$$

This is both normality $\mathbf{u}_i^{\mathrm{T}}\mathbf{u}_i = 1$, and orthonormality $\mathbf{u}_i^{\mathrm{T}}\mathbf{u}_j = 1, i \neq j$.

Example 4.1: A 2×2 case.

(for column vectors $\mathbf{u}_i, \mathbf{v}_j$).

$$M = \begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 \end{bmatrix} \begin{bmatrix} \sigma_1 & \\ & \sigma_2 \end{bmatrix} \begin{bmatrix} \mathbf{v}_1^{\mathrm{T}} \\ \mathbf{v}_2^{\mathrm{T}} \end{bmatrix} \tag{4.8}$$

Consider $\mathbf{y} = M\mathbf{x}$, and take an $\mathbf{x}$ with $\|\mathbf{x}\|_2 = 1$

Note: I've chosen not to sketch what was drawn on the board in class. See instead the animated gif of the same in [15], or the Wolfram online SVD demo in [6].

A very nice video treatment of SVD by Prof Gilbert Strang can be found in [12].

4.2 CONDITIONING NUMBER

Given a perturbation of $M\mathbf{x} = \mathbf{b}$ to

$$(M + \delta M)(\mathbf{x} + \delta\mathbf{x}) = \mathbf{b}, \tag{4.9}$$

or

$$\cancel{M\mathbf{x} = \mathbf{b}} + \delta M\mathbf{x} + M\delta\mathbf{x} + \delta M\delta\mathbf{x} = 0. \tag{4.10}$$

This gives

$$M\delta\mathbf{x} = -\delta M\mathbf{x} - \delta M\delta\mathbf{x}, \tag{4.11}$$

or

$$\delta\mathbf{x} = -M^{-1}\delta M\left(\mathbf{x} + \delta\mathbf{x}\right). \tag{4.12}$$

Taking norms

$$\begin{aligned} \|\delta\mathbf{x}\|_2 &= \left\|M^{-1}\delta M\left(\mathbf{x} + \delta\mathbf{x}\right)\right\|_2 \\ &\leq \left\|M^{-1}\right\|_2 \|\delta M\|_2 \|\mathbf{x} + \delta\mathbf{x}\|_2, \end{aligned} \tag{4.13}$$

or

relative error of solution

relative perturbation of M

$$\frac{\|\delta\mathbf{x}\|_2}{\|\mathbf{x} + \delta\mathbf{x}\|_2} \leq \|M\|_2 \left\|M^{-1}\right\|_2 \frac{\|\delta M\|_2}{\|M\|_2}. \tag{4.14}$$

conditioning number of M

The conditioning number can be shown to be

$$\text{cond}(M) = \frac{\sigma_{\max}}{\sigma_{\min}} \geq 1 \tag{4.15}$$

FIXME: justify.

sensitivity to conditioning number Double precision relative rounding errors can be of the order $10^{-16} \sim 2^{-54}$, which allows the relative error of the solution to be gauged

relative error of solution	$\leq$	cond(M)	$\frac{\|\delta M\|}{\|M\|}$
10^{-15}	$\leq$	10	$\sim 10^{-16}$
10^{-2}	$\leq$	10^{14}	10^{-16}

5 SPARSE FACTORIZATION

5.1 FILL INS

The problem of fill ins in LU computations arise in locations where rows and columns cross over zero positions.

Rows and columns can be permuted to deal with these. Here is an ad-hoc permutation of rows and columns that will result in less fill ins.

$$
\begin{aligned}
&\begin{bmatrix} a & b & c & 0 \\ d & e & 0 & 0 \\ 0 & f & g & 0 \\ 0 & h & 0 & i \end{bmatrix}
\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} =
\begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} \\
\Longrightarrow &\begin{bmatrix} a & c & 0 & b \\ d & 0 & 0 & e \\ 0 & g & 0 & f \\ 0 & 0 & i & h \end{bmatrix}
\begin{bmatrix} x_1 \\ x_4 \\ x_3 \\ x_2 \end{bmatrix} =
\begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} \\
\Longrightarrow &\begin{bmatrix} 0 & a & c & b \\ 0 & d & 0 & e \\ 0 & 0 & g & f \\ i & 0 & 0 & h \end{bmatrix}
\begin{bmatrix} x_3 \\ x_4 \\ x_1 \\ x_2 \end{bmatrix} =
\begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} \\
\Longrightarrow &\begin{bmatrix} i & 0 & 0 & h \\ 0 & a & c & b \\ 0 & d & 0 & e \\ 0 & 0 & g & f \end{bmatrix}
\begin{bmatrix} x_3 \\ x_4 \\ x_1 \\ x_2 \end{bmatrix} =
\begin{bmatrix} b_4 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} \\
\Longrightarrow &\begin{bmatrix} i & 0 & 0 & h \\ 0 & c & a & b \\ 0 & 0 & d & e \\ 0 & g & 0 & f \end{bmatrix}
\begin{bmatrix} x_3 \\ x_1 \\ x_4 \\ x_2 \end{bmatrix} =
\begin{bmatrix} b_4 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}
\end{aligned}
\tag{5.1}
$$

5.2 MARKOWITZ PRODUCT

To facilitate such permutations the Markowitz product that estimates the amount of fill in required.

Definition 5.1: Markowitz product.

$$\begin{aligned}\text{Markowitz product} = &(\text{Non zeros in unfactored part of Row -1}) \times \\ &(\text{Non zeros in unfactored part of Col -1})\end{aligned}$$

In [7] it is stated "A still simpler alternative, which seems adequate generally, is to choose the pivot which minimizes the number of coefficients modified at each step (excluding those which are eliminated at the particular step). This is equivalent to choosing the non-zero element with minimum $(\rho_i - 1)(\sigma_j - 1)$."

Note that this product is applied only to ij positions that are non-zero, something not explicitly mentioned in the slides, nor in other locations like [13].

Example 5.1: Markowitz product.

For this matrix

$$\begin{bmatrix} a & b & c & 0 \\ d & e & 0 & 0 \\ 0 & f & g & 0 \\ 0 & h & 0 & i \end{bmatrix}, \tag{5.2}$$

the Markowitz products are

$$\begin{bmatrix} 1 & 6 & 2 & \\ 1 & 3 & & \\ & 3 & 1 & \\ & 3 & & 0 \end{bmatrix}. \tag{5.3}$$

5.3 MARKOWITZ REORDERING

The Markowitz Reordering procedure (copied directly from the slides) is

- For i = 1 to n
- Find diagonal $j >= i$ with min Markowitz product
- Swap rows $j \leftrightarrow i$ and columns $j \leftrightarrow i$
- Factor the new row i and update Markowitz products

Example 5.2: Markowitz reordering.

Looking at the Markowitz products eq. (5.3) a swap of rows and columns $1, 4$ gives the modified matrix

$$\begin{bmatrix} i & 0 & h & 0 \\ 0 & d & e & 0 \\ 0 & 0 & f & g \\ 0 & a & b & c \end{bmatrix} \tag{5.4}$$

In this case, this reordering has completely avoided any requirement to do any actual Gaussian operations for this first stage reduction.

Presuming that the Markowitz products for the remaining 3x3 submatrix are only computed from that submatrix, the new products are

$$\begin{bmatrix} & & \\ 1 & 2 & \\ & 2 & 1 \\ 2 & 4 & 2 \end{bmatrix}. \tag{5.5}$$

The pivot position contains a minimal product, and happens to lie on the diagonal. Note that it is not necessarily the best for numerical stability. It appears the off diagonal Markowitz products are not really of interest since the reordering algorithm swaps both rows and columns.

5.4 GRAPH REPRESENTATION

It is possible to interpret the Markowitz products on the diagonal as connectivity of a graph that represents the interconnections of the nodes. Consider the circuit of fig. 5.1 as an example

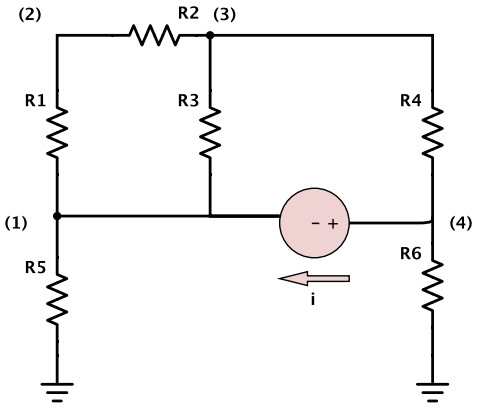


Figure 5.1: Simple circuit.

The system equations for this circuit is of the form

$$\begin{bmatrix} x & x & x & 0 & 1 \\ x & x & x & 0 & 0 \\ x & x & x & x & 0 \\ 0 & 0 & x & x & -1 \\ -1 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \\ i \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ x \end{bmatrix}. \tag{5.6}$$

The Markowitz products along the diagonal are

$$\begin{aligned} M_{11} &= 9 \\ M_{22} &= 4 \\ M_{33} &= 9 \\ M_{44} &= 4 \\ M_{55} &= 4 \end{aligned} \tag{5.7}$$

Compare these to the number of interconnections of the graph fig. 5.2 of the nodes in this circuit. These are the squares of the number of the node interconnects in each case.

Here a 5th node was introduced for the current i between nodes 4 and 1. Observe that the Markowitz product of this node was counted as the number of non-zero values excluding the $5,5$ matrix position. However, that doesn't matter too much since a Markowitz swap of row/column 1 with row/column 5 would put a zero in the

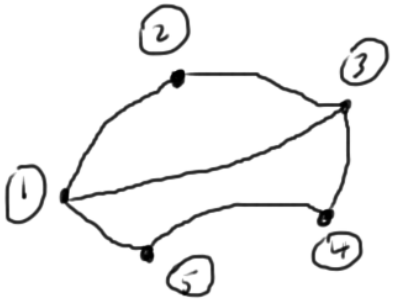


Figure 5.2: Graph representation.

1, 1 position of the matrix, which is not desirable. Permutations of zero diagonal positions will have to be restricted to pivots required for numerical stability, or taken into account with a a more advanced zero fill avoidance algorithm.

The minimum diagonal Markowitz products are in positions 2 or 4, with respective Markowitz reorderings of the form

$$\begin{bmatrix} x & x & x & 0 & 0 \\ x & x & x & 0 & 1 \\ x & x & x & x & 0 \\ 0 & 0 & x & x & -1 \\ 0 & -1 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} V_2 \\ V_1 \\ V_3 \\ V_4 \\ i \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ x \end{bmatrix}, \tag{5.8}$$

and

$$\begin{bmatrix} x & 0 & 0 & x & -1 \\ 0 & x & x & x & 1 \\ 0 & x & x & x & 0 \\ x & x & x & x & 0 \\ 1 & -1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} V_4 \\ V_1 \\ V_2 \\ V_3 \\ i \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ x \end{bmatrix}. \tag{5.9}$$

The original system had 7 zeros that could potentially be filled in the remaining 4×4 submatrix. After a first round of Gaussian elimination, the system matrices have the respective forms

$$\begin{bmatrix} x & x & x & 0 & 0 \\ 0 & x & x & 0 & 1 \\ 0 & x & x & x & 0 \\ 0 & 0 & x & x & -1 \\ 0 & -1 & 0 & 1 & 0 \end{bmatrix} \tag{5.10a}$$

$$\begin{bmatrix} x & 0 & 0 & x & -1 \\ 0 & x & x & x & 1 \\ 0 & x & x & x & 0 \\ 0 & x & x & x & 0 \\ 0 & -1 & 0 & x & x \end{bmatrix} \tag{5.10b}$$

The remaining 4×4 submatrices have interconnect graphs sketched in fig. 5.3.

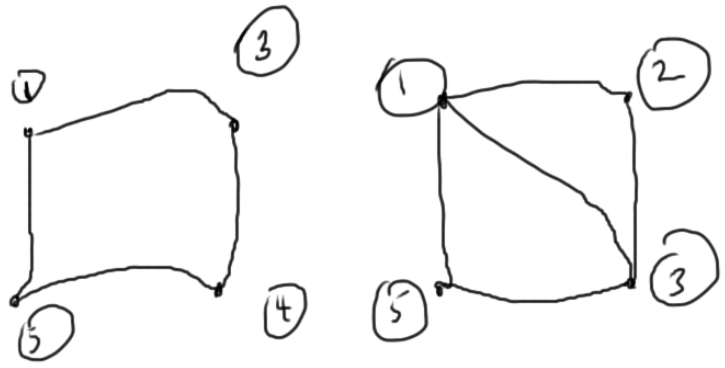


Figure 5.3: Graphs after one round of Gaussian elimination.

From a graph point of view, the objective is to delete the most connected nodes. This can be driven by the Markowitz products along the diagonal or directly with graph methods.

6 GRADIENT METHODS

6.1 SUMMARY OF FACTORIZATION COSTS

LU (dense)

- cost: $O(n^3)$
- cost depends only on size

LU (sparse)

- cost: Diagonal and tridiagonal are $O(n)$, but can be up to $O(n^3)$ depending on sparsity and the method of dealing with the sparsity.
- cost depends on size and sparsity

Computation can be affordable up to a few million elements.

Iterative methods Can be cheap if done right. Convergence requires careful preconditioning.

6.2 ITERATIVE METHODS

Given an initial guess of $\mathbf{x}_0$, any iterative methods are generally of the form

Algorithm 6.1: Iterative methods.

repeat
  $\mathbf{r} = \mathbf{b} - M\mathbf{x}_i$
until $\|\mathbf{r}\| < \epsilon$.

The difference $\mathbf{r}$ is called the residual. For as long as it is bigger than desired, continue improving the estimate $\mathbf{x}_i$.

The matrix vector product $M\mathbf{x}_i$, if dense, is of $O(n^2)$. Suppose, for example, that the solution can be found in ten iterations. If the matrix is dense, performance can be of $10\,O(n^2)$. If sparse, this can be worse than just direct computation.

6.3 GRADIENT METHOD

This is a method for iterative solution of the equation $M\mathbf{x} = \mathbf{b}$.

This requires symmetric positive definite matrix $M = M^{\mathrm{T}}$, with $M > 0$, for which an energy function is defined

$$\Psi(\mathbf{y}) \equiv \frac{1}{2}\mathbf{y}^{\mathrm{T}} M\mathbf{y} - \mathbf{y}^{\mathrm{T}}\mathbf{b} \tag{6.1}$$

For a two variable system this is illustrated in fig. 6.1.

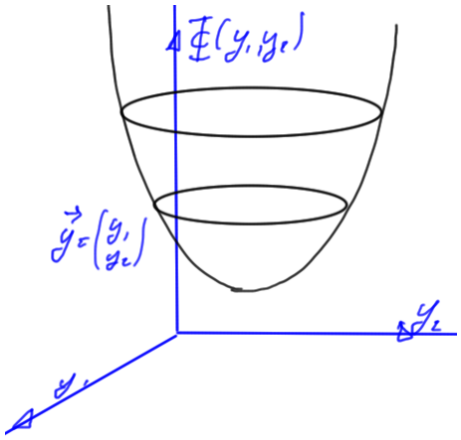


Figure 6.1: Positive definite energy function.

Theorem 6.1: Energy function minimum.

The energy function eq. (6.1) has a minimum at

$$\mathbf{y} = M^{-1}\mathbf{b} = \mathbf{x}. \tag{6.2}$$

To prove this, consider the coordinate representation

$$\Psi = \frac{1}{2} y_a M_{ab} y_b - y_b b_b, \tag{6.3}$$

for which the derivatives are

$$\begin{aligned}\frac{\partial \Psi}{\partial y_i} &= \frac{1}{2} M_{ib} y_b + \frac{1}{2} y_a M_{ai} - b_i \\ &= (M\mathbf{y} - \mathbf{b})_i \,.\end{aligned} \tag{6.4}$$

The last operation above was possible because $M = M^{\mathrm{T}}$. Setting all of these equal to zero, and rewriting this as a matrix relation gives

$$M\mathbf{y} = \mathbf{b}, \tag{6.5}$$

as asserted.

This is called the gradient method because the gradient moves a point along the path of steepest descent towards the minimum if it exists.

The method is

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \underbrace{\alpha_k}_{\text{step size}} \underbrace{\mathbf{d}^{(k)}}_{\text{direction}}, \tag{6.6}$$

where the direction is

$$\begin{aligned}\mathbf{d}^{(k)} &= -\boldsymbol{\nabla}\Phi \\ &= \mathbf{b} - M\mathbf{x}^{(k)} \\ &= r^{(k)}.\end{aligned} \tag{6.7}$$

Optimal step size The next iteration expansion of the energy function $\Phi\left(\mathbf{x}^{(k+1)}\right)$ is

$$\begin{aligned}\Phi\left(\mathbf{x}^{(k+1)}\right) &= \Phi\left(\mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}\right) \\ &= \frac{1}{2}\left(\mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}\right)^{\mathrm{T}} M \left(\mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}\right) - \left(\mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}\right)^{\mathrm{T}} \mathbf{b}.\end{aligned} \tag{6.8}$$

To find the minimum, the derivative of both sides with respect to α_k is required

$$0 = \frac{1}{2}\left(\mathbf{d}^{(k)}\right)^{\mathrm{T}} M\mathbf{x}^{(k)} + \frac{1}{2}\left(\mathbf{x}^{(k)}\right)^{\mathrm{T}} M\mathbf{d}^{(k)} + \alpha_k \left(\mathbf{d}^{(k)}\right)^{\mathrm{T}} M\mathbf{d}^{(k)} - \left(\mathbf{d}^{(k)}\right)^{\mathrm{T}} \mathbf{b}. \tag{6.9}$$

Because M is symmetric, this is

$$\alpha_k \left(\mathbf{d}^{(k)}\right)^{\mathrm{T}} M\mathbf{d}^{(k)} = \left(\mathbf{d}^{(k)}\right)^{\mathrm{T}} \left(\mathbf{b} - M\mathbf{x}^{(k)}\right) = \left(\mathbf{d}^{(k)}\right)^{\mathrm{T}} r^{(k)}, \tag{6.10}$$

or

$$\alpha_k = \frac{\left(\mathbf{r}^{(k)}\right)^{\mathrm{T}} \mathbf{r}^{(k)}}{\left(\mathbf{r}^{(k)}\right)^{\mathrm{T}} M\mathbf{r}^{(k)}} \tag{6.11}$$

This this method is not optimal when a new direction is picked each step of the iteration.

6.4 RECAP: SUMMARY OF GRADIENT METHOD

The gradient method allowed for a low cost iterative solution of a linear system

$$M\mathbf{x} = \mathbf{b}, \tag{6.12}$$

without a requirement for complete factorization. The residual was defined as the difference between the application of any trial solution $\mathbf{y}$ to M and $\mathbf{b}$

$$\mathbf{r} = \mathbf{b} - M\mathbf{y}. \tag{6.13}$$

An energy function was introduced

$$\Psi(\mathbf{y}) = \frac{1}{2}\mathbf{y}^{\mathrm{T}} M\mathbf{y} - \mathbf{y}^{\mathrm{T}}\mathbf{b}, \tag{6.14}$$

which has an extremum at the point of solution. The goal was to attempt to following the direction of steepest decent $\mathbf{r}^{(k)} = -\boldsymbol{\nabla}\Psi$ with the hope of finding the minimum of the energy function. That iteration is described by

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{r}^{(k)}, \tag{6.15}$$

and illustrated in fig. 6.2.

The problem of the gradient method is that it introduces multiple paths as sketched in fig. 6.3.

which lengthens the total distance that has to be traversed in the iteration.

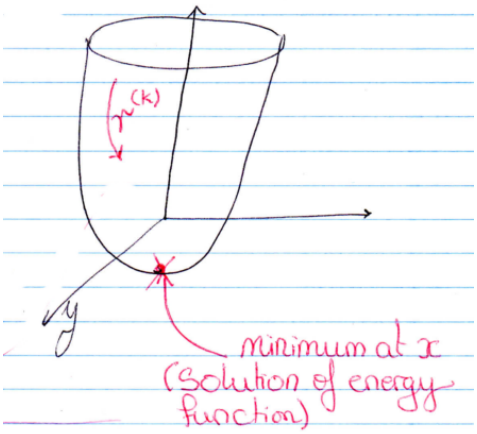


Figure 6.2: Gradient descent.

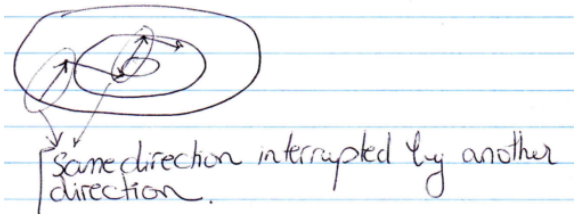


Figure 6.3: Gradient descent iteration.

6.5 CONJUGATE GRADIENT METHOD

The Conjugate gradient method makes the residual at step k orthogonal to all previous search directions

1. $\mathbf{r}^{(1)} \perp \mathbf{d}^{(0)}$
2. $\mathbf{r}^{(1)} \perp \mathbf{d}^{(0)}, \mathbf{d}^{(1)}$
3. $\cdots$

After n iterations, the residual will be zero.

First iteration Given an initial guess $\mathbf{x}^{(0)}$, proceed as in the gradient method

$$\mathbf{d}^{(0)} = -\nabla\Psi(\mathbf{x}^{(0)}) = \mathbf{r}^{(0)}, \tag{6.16}$$

$$\mathbf{x}^{(1)} = \mathbf{x}^{(0)} + \alpha_0 \mathbf{r}^{(0)}, \tag{6.17}$$

with

$$\alpha_0 = \frac{\left(\mathbf{r}^{(0)}\right)^{\mathrm{T}} \mathbf{r}^{(0)}}{\left(\mathbf{r}^{(0)}\right)^{\mathrm{T}} M\mathbf{r}^{(0)}}, \tag{6.18}$$

so that the residual is

$$\begin{aligned} \mathbf{r}^{(1)} &= \mathbf{b} - M\mathbf{x}^{(1)} \\ &= \mathbf{b} - M\mathbf{x}^{(0)} - \alpha_0 M\mathbf{r}^{(0)} \\ &= \mathbf{b} - \alpha M\mathbf{r}^{(0)}. \end{aligned} \tag{6.19}$$

An orthogonality condition $\mathbf{r}^{(1)} \perp \mathbf{d}^{(0)}$ is desired.

Proof:

$$\begin{aligned} \langle \mathbf{d}^{(0)}, \mathbf{r}^{(1)} \rangle &= \langle \mathbf{d}^{(0)}, \mathbf{r}^{(0)} \rangle - \alpha_0 \langle \mathbf{d}^{(0)}, M\mathbf{r}^{(0)} \rangle \\ &= \langle \mathbf{d}^{(0)}, \mathbf{r}^{(0)} \rangle - \alpha_0 \langle \mathbf{r}^{(0)}, M\mathbf{r}^{(0)} \rangle. \end{aligned} \tag{6.20}$$

Second iteration

$$\mathbf{x}^{(2)} = \mathbf{x}^{(1)} + \alpha_1 \mathbf{d}^{(1)}. \tag{6.21}$$

The conditions to satisfy are

$$\mathbf{d}^{(0)} \perp \mathbf{r}^{(2)} \tag{6.22a}$$

$$\mathbf{d}^{(1)} \perp \mathbf{r}^{(2)} \tag{6.22b}$$

Observe that the orthogonality condition of eq. (6.22a) is satisfied

$$\begin{aligned} \langle \mathbf{d}^{(0)}, \mathbf{r}^{(2)} \rangle &= \langle \mathbf{d}^{(0)}, \mathbf{b} - M\mathbf{x}^{(2)} \rangle \\ &= \langle \mathbf{d}^{(0)}, \mathbf{b} - M\mathbf{x}^{(1)} - \alpha_1 M\mathbf{d}^{(1)} \rangle \\ &\qquad = 0 \text{ because } \mathbf{d}^{(0)} \perp \mathbf{r}^{(1)} \\ &= \boxed{\langle \mathbf{d}^{(0)}, \mathbf{b} - M\mathbf{x}^{(1)} \rangle} - \alpha_1 \langle \mathbf{d}^{(0)}, \alpha_1 M\mathbf{d}^{(1)} \rangle \end{aligned} \tag{6.23}$$

This will be zero if an M orthogonality or conjugate zero condition can be imposed

$$\langle \mathbf{d}^{(0)}, M\mathbf{d}^{(1)} \rangle = 0. \tag{6.24}$$

To find a new search direction $\mathbf{d}^{(1)} = \mathbf{r}^{(1)} - \beta_0 \mathbf{d}^{(0)}$, eq. (6.24) is now imposed. The $\mathbf{r}^{(1)}$ is the term from the standard gradient method, and the $\beta_0 \mathbf{d}^{(0)}$ term is the conjugate gradient correction. This gives β_0

$$\beta_0 = \frac{\langle \mathbf{d}^{(0)}, M\mathbf{r}^{(1)} \rangle}{\langle \mathbf{d}^{(0)}, M\mathbf{d}^{(0)} \rangle} \tag{6.25}$$

Imposing eq. (6.22b) gives

$$\alpha_1 = \frac{\langle \mathbf{d}^{(0)}, \mathbf{r}^{(1)} \rangle}{\langle \mathbf{d}^{(1)}, M\mathbf{d}^{(1)} \rangle}. \tag{6.26}$$

Next iteration

$$\begin{aligned} \mathbf{x}^{(k+1)} &= \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)} \\ \mathbf{d}^{(k)} &= \mathbf{r}^{(k)} - \beta_{k-1} \mathbf{d}^{(k-1)} \end{aligned} \tag{6.27}$$

The conditions to impose are

$$\begin{aligned} &\mathbf{d}^{(0)} \perp \mathbf{r}^{(k+1)} \\ &\vdots \\ &\mathbf{d}^{(k-1)} \perp \mathbf{r}^{(k+1)} \end{aligned} \tag{6.28}$$

However, there are only 2 degrees of freedom α, β, despite having many conditions to impose.

Impose the following to find β_{k-1}

$$\mathbf{d}^{(k-1)} \perp \mathbf{r}^{(k+1)} \tag{6.29}$$

(See slides for more)

6.6 FULL ALGORITHM

CG without preconditioning The conjugate gradient algorithm presented in the slides (without preconditioning) was

Algorithm 6.2: Conjugate gradient.

$\mathbf{d}^{(0)} = \mathbf{r}^{(0)}$
repeat
  $\alpha_k = \dfrac{\left(\mathbf{d}^{(k)}\right)^{\mathrm{T}} \mathbf{r}^{(k)}}{\left(\mathbf{d}^{(k)}\right)^{\mathrm{T}} M \mathbf{d}^{(k)}}$
  $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}$
  $\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} - \alpha_k M \mathbf{d}^{(k)}$
  $\beta_k = \dfrac{\left(M\mathbf{d}^{(k)}\right)^{\mathrm{T}} \mathbf{r}^{(k+1)}}{\left(M\mathbf{d}^{(k)}\right)^{\mathrm{T}} \mathbf{d}^{(k)}}$
  $\mathbf{d}^{(k+1)} = \mathbf{r}^{(k+1)} - \beta_k \mathbf{d}^{(k)}$
until converged

The repeated calculations are undesirable for actually coding this algorithm. First introduce a temporary for the matrix product

Algorithm 6.3: Conjugate gradient with temporaries.

$\mathbf{d}^{(0)} = \mathbf{r}^{(0)}$
repeat
  $\mathbf{q} = M\mathbf{d}^{(k)}$
  $\alpha_k = \frac{\left(\mathbf{d}^{(k)}\right)^{\mathrm{T}} \mathbf{r}^{(k)}}{\left(\mathbf{d}^{(k)}\right)^{\mathrm{T}} \mathbf{q}}$
  $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}$
  $\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} - \alpha_k \mathbf{q}$
  $\beta_k = \frac{\mathbf{q}^{\mathrm{T}} \mathbf{r}^{(k+1)}}{\mathbf{q}^{\mathrm{T}} \mathbf{d}^{(k)}}$
  $\mathbf{d}^{(k+1)} = \mathbf{r}^{(k+1)} - \beta_k \mathbf{d}^{(k)}$
until converged

This has a lot more computation than the algorithm specified in [11] §B.2. It looks like the orthogonality properties can be used to recast the $\mathbf{d}^{(k)} \cdot \mathbf{r}^{(k)}$ products in terms of $\mathbf{r}^{(k)}$

$$\left(\mathbf{d}^{(k)}\right)^{\mathrm{T}} \mathbf{r}^{(k)} = \mathbf{r}^{(k)} \cdot \left(\mathbf{r}^{(k)} + \beta_{k-1} \mathbf{d}^{(k-1)}\right). \tag{6.30}$$

Since the new residual is orthogonal to all the previous search directions $\mathbf{r}^{(k)} \cdot \mathbf{d}^{(k-1)} = 0$, the β_{k-1} term is killed leaving just $\mathbf{r}^{(k)} \cdot \mathbf{r}^{(k)}$.

The numerator of β_k can be tackled by noting that the transformed direction vector $\mathbf{q}$ is a scaled difference of residuals. Taking dot products

$$\begin{aligned} \mathbf{q} \cdot \mathbf{r}^{(k+1)} &= \frac{1}{\alpha_k} \left(\mathbf{r}^{(k)} - \mathbf{r}^{(k+1)}\right) \cdot \mathbf{r}^{(k+1)} \\ &= \frac{1}{\alpha_k} \left(\cancel{\mathbf{d}^{(k)}} - \beta_{k-1} \cancel{\mathbf{d}^{(k-1)}} - \mathbf{r}^{(k+1)}\right) \cdot \mathbf{r}^{(k+1)} \\ &= -\frac{1}{\alpha_k} \mathbf{r}^{(k+1)} \cdot \mathbf{r}^{(k+1)}. \end{aligned} \tag{6.31}$$

This gives

$$\alpha_k = \frac{\left(\mathbf{r}^{(k)}\right)^{\mathrm{T}} \mathbf{r}^{(k)}}{\left(\mathbf{d}^{(k)}\right)^{\mathrm{T}} \mathbf{q}} \tag{6.32a}$$

$$\beta_k = -\frac{\left(\mathbf{r}^{(k+1)}\right)^{\mathrm{T}} \mathbf{r}^{(k+1)}}{\alpha_k \mathbf{q}^{\mathrm{T}} \mathbf{d}^{(k)}}, \tag{6.32b}$$

A final elimination of α_k from eq. (6.32b) gives

$$\beta_k = -\frac{\left(\mathbf{r}^{(k+1)}\right)^{\mathrm{T}} \mathbf{r}^{(k+1)}}{\left(\mathbf{r}^{(k)}\right)^{\mathrm{T}} \mathbf{r}^{(k)}}. \tag{6.33}$$

All the pieces put together yield

Algorithm 6.4: Optimized conjugate gradient.

$\mathbf{d}^{(0)} = \mathbf{r}^{(0)}$
repeat
  $\mathbf{q} = M\mathbf{d}^{(k)}$
  $\alpha_k = \frac{\left(\mathbf{r}^{(k)}\right)^{\mathrm{T}} \mathbf{r}^{(k)}}{\left(\mathbf{d}^{(k)}\right)^{\mathrm{T}} \mathbf{q}}$
  $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}$
  $\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} - \alpha_k \mathbf{q}$
  $\beta_k = -\frac{\left(\mathbf{r}^{(k+1)}\right)^{\mathrm{T}} \mathbf{r}^{(k+1)}}{\left(\mathbf{r}^{(k)}\right)^{\mathrm{T}} \mathbf{r}^{(k)}}$
  $\mathbf{d}^{(k+1)} = \mathbf{r}^{(k+1)} - \beta_k \mathbf{d}^{(k)}$
until converged

This is now consistent with eqns 45-49 of [11], with the exception that the sign of the β_k term has been flipped.

Since all the previous state does not have to be tracked the indexes can be dropped after introducing a couple temporary variables for the squared residuals

Algorithm 6.5: Optimized conjugate gradient, more temporaries.

$\mathbf{x} = \mathbf{x}^{(0)}$
$\mathbf{q} = M\mathbf{x}$
$\mathbf{r} = \mathbf{b} - \mathbf{q}$
$\mathbf{d} = \mathbf{r}$
repeat
  $\mathbf{q} = M\mathbf{d}$
  $\delta = \mathbf{r}^{\mathrm{T}}\mathbf{r}$
  $\alpha = \frac{\delta}{\mathbf{d}^{\mathrm{T}}\mathbf{q}}$
  $\mathbf{x} = \mathbf{x} + \alpha\mathbf{d}$
  $\mathbf{r} = \mathbf{r} - \alpha\mathbf{q}$

$\delta' = \mathbf{r}^T\mathbf{r}$
$\beta = \frac{\delta'}{\delta}$
$\delta = \delta'$
$\mathbf{d} = \mathbf{r} + \beta\mathbf{d}$
until converged

This is coded in ps2a/conjugateGradientPainlessB2.m. That implementation allows for a preconditioner, but applies the preconditioner in a dump and inefficient way.

CG with preconditioning An optimized preconditioned CG algorithm can be found in [10] §4.3.5. Given $\mathbf{x}^{(0)}$, that algorithm is

Algorithm 6.6: Conjugate gradient with preconditioning.

$\mathbf{r}^{(0)} = \mathbf{b} - A\mathbf{x}^{(0)}$
$\mathbf{z}^{(0)} = P^{-1}\mathbf{r}^{(0)}$
$\mathbf{p}^{(0)} = \mathbf{z}^{(0)}$
repeat
$\alpha_k = \frac{\left(\mathbf{z}^{(k)}\right)^T\mathbf{r}^{(k)}}{\left(\mathbf{p}^{(k)}\right)^T A\mathbf{p}^{(k)}}$
$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k\mathbf{p}^{(k)}$
$\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} - \alpha_k A\mathbf{p}^{(k)}$
$P\mathbf{z}^{(k+1)} = \mathbf{r}^{(k+1)}$
$\beta_k = \frac{\left(\mathbf{z}^{(k+1)}\right)^T\mathbf{r}^{(k+1)}}{\left(\mathbf{z}^{(k)}\right)^T\mathbf{r}^{(k)}}$
$\mathbf{p}^{(k+1)} = \mathbf{z}^{(k+1)} + \beta_k\mathbf{p}^{(k)}$
until converged

To adapt this to code, drop the indexes and introduce some temporaries

Algorithm 6.7: Conjugate gradient with preconditioning and temporaries.

$\mathbf{r} = \mathbf{b} - A\mathbf{x}$
$P\mathbf{z} = \mathbf{r}$
$\mathbf{p} = \mathbf{z}$
$\delta = \mathbf{z}^T\mathbf{r}$
repeat
$\mathbf{q} = A\mathbf{p}$
$\alpha = \delta/\left(\mathbf{p}^T\mathbf{q}\right)$

$\mathbf{x} = \mathbf{x} + \alpha\mathbf{p}$
$\mathbf{r} = \mathbf{r} - \alpha\mathbf{q}$
$P\mathbf{z} = \mathbf{r}$
$\delta' = \mathbf{z}^{\mathrm{T}}\mathbf{r}$
$\beta = \delta'/\delta$
$\delta = \delta'$
$\mathbf{p} = \mathbf{z} + \beta\mathbf{p}$
until converged

This is coded in ps2a/conjugateGradientQuarteroniPrecond.m

Table 6.1: LU vs Conjugate gradient order.

	Full	Sparse
LU	$O(n^3)$	$O(n^{1.2-1.8})$
Conjugate gradient	$O(kn^2)$	5.4

Table 6.2: Convergence.

	Full
Direct	$O(n^3)$
C.G.	$O(kn^2)$

6.7 ORDER ANALYSIS

Note that for $\mathbf{x}, \mathbf{y}$ in $\mathbb{R}^n$, $\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^T\mathbf{y}$ is $O(n)$, and $M\mathbf{x}$ is $O(n^2)$.

A conjugate gradient and LU comparision is given in table 6.1, where $k < n$

Final comments

1. How to select $\mathbf{x}^{(0)}$? An initial rough estimate can often be found by solving a simplified version of the problem.
2. Is it neccessary to reserve memory space to store M? No. If $\mathbf{y} = M\mathbf{z}$, the product can be calculated without physically storing the full matrix M.

6.8 CONJUGATE GRADIENT CONVERGENCE

For $k \ll n$, convergence orders are given in table 6.2. A matrix norm, similar to $\|\mathbf{x}\| = \sqrt{\mathbf{x}^T\mathbf{x}}$, is defined

$$\|\mathbf{x}\|_M \equiv \sqrt{\mathbf{x}^T M\mathbf{x}}. \tag{6.34}$$

Note that this norm is real valued for CG which only applies to positive definite matrices (or it will not converge), so this norm is real valued.

... lots on slides...

$$K(M) = \frac{\sigma_{\max}}{\sigma_{\min}} \tag{6.35}$$

...
Some fast ways to estimate the conditioning number are required.

6.9 GERSHGORIN CIRCLE THEOREM

Theorem 6.2: Gershgorin circle theorem.

Given M, for any eigenvalue of M there is an $i \in [1, n]$ such that

$$|\lambda - M_{ii}| \leq \sum_{j \neq i} |M_{ij}|$$

Consider this in the complex plane for row i

$$\begin{bmatrix} M_{i1} & M_{i2} & \cdots M_{ii} & \cdots M_{in} \end{bmatrix} \tag{6.36}$$

This inequality covers a circular region in the complex plane as illustrated in fig. 6.4 for a two eigenvalue system.

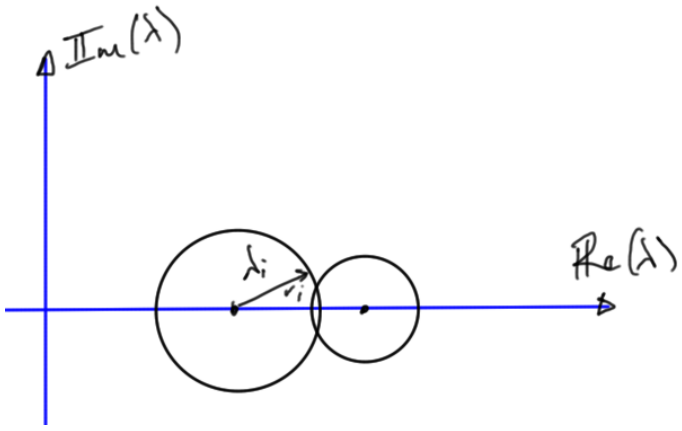


Figure 6.4: Gershgorin circles.

These are called Gershgorin circles.

Example 6.1: Leaky bar.

For the leaky bar of fig. 6.5, the matrix is

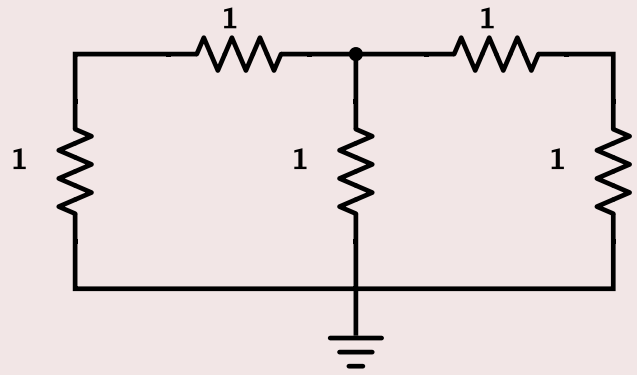


Figure 6.5: Leaky bar.

$$M = \begin{bmatrix} 2 & -1 & & & \\ -1 & 3 & -1 & & \\ & -1 & 3 & -1 & \\ & & & \ddots & \\ & & & & -1 \\ & & & -1 & 2 \end{bmatrix} \tag{6.37}$$

The Gershgorin circles are fig. 6.6.

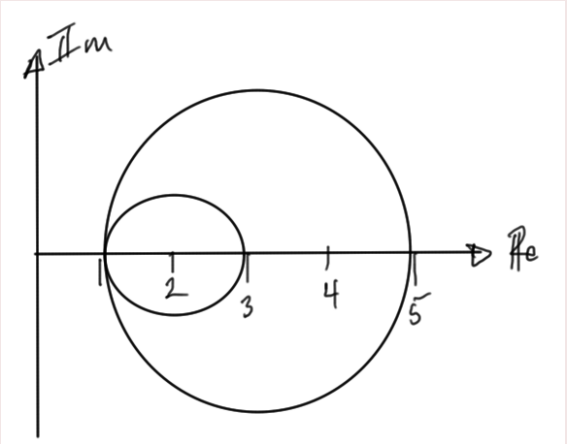


Figure 6.6: Gershgorin circles for leaky bar.

This puts a bound on the eigenvalues

$$1 \leq \lambda(M) \leq 5, \tag{6.38}$$

so that

$$K(M) = \frac{\lambda_{\max}}{\lambda_{\max}} \leq 5. \tag{6.39}$$

On slides: example with smaller leakage to ground. On slides: example with no leakage to ground.

These had, progressively larger and larger (possibly indefinite for the latter) conditioning number estimates.

The latter had the form of

$$M = \begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & 1 \\ 0 & 0 & -1 & 1 \end{bmatrix} \tag{6.40}$$

The exact eigenvalues for this system happens to be

$$\lambda \in \{3.10690.2833, 1.3049 \pm 0.7545i\} \tag{6.41}$$

so the exact conditioning number is $3.1/0.28 \approx 11$.

Compare this to the estimates, which are

$$\begin{aligned} |\lambda_1 - 1| &\leq 1 \\ |\lambda_2 - 2| &\leq 2 \\ |\lambda_3 - 2| &\leq 2 \\ |\lambda_4 - 1| &\leq 1 \end{aligned} \tag{6.42}$$

These are two circles at $z = 1$ of radius 1, and two circles at $z = 2$ of radius 2, as plotted in fig. 6.7.

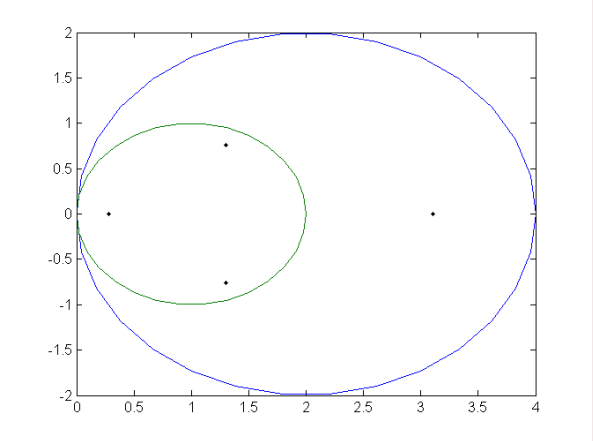


Figure 6.7: Gershgorin circles for 4 eigenvalue system.

6.10 PRECONDITIONING

Goal is to take

$$M\mathbf{x} = \mathbf{b} \tag{6.43}$$

and introduce an easy to invert matrix P to change the problem to

$$P^{-1}M\mathbf{x} = P^{-1}\mathbf{b}. \tag{6.44}$$

This system has the same solution, but allows for choosing P to maximize the convergence speed.

6.11 SYMMETRIC PRECONDITIONING

Because the conjugate gradient methods requires a symmetric matrix, it is desirable to pick a preconditioning method that preserves the symmetric (and positive definite) nature of the matrix. This is possible by splitting P into square root factors

$$P = P^{1/2}P^{1/2}, \tag{6.45}$$

and apply to $M\mathbf{x} = \mathbf{b}$ as

$$P^{-1/2}M\underbrace{P^{-1/2}P^{1/2}}_{=I}\mathbf{x} = P^{-1/2}\mathbf{b}. \tag{6.46}$$

Now introduce a change of variables $\mathbf{y} = P^{1/2}\mathbf{x}$, transforming the system to solve into

$$P^{-1/2}MP^{-1/2}\mathbf{y} = \mathbf{b}'. \tag{6.47}$$

Some options

- Diagonal preconditioner: $\mathbf{P} = \text{diag}\{M\}$
- Incomplete LU or Cholesky factorization. Cheap, approximate decomposition where a preconditioner $M \simeq LU = P$ is picked. An incomplete LU factorization would be easy to invert since lower or upper triangular matrices are easy to invert. In Matlab the ilu() function can be used to do an incomplete LU factorization.
- ... (many preconditioner are available).

For a symmetric positive definite matrix M, an LU decomposition of the form $M = LL^{\mathrm{T}}$, is called the Cholesky factorization.

As an example consider the matrix

$$M = \begin{bmatrix} 2 & & & & & & & & \\ & 2 & & & & \text{(small)} & & & \\ & & \ddots & & & & & & \\ & & & 6 & & & & & \\ & & & & 6 & & & & \\ & & & & & \ddots & & & \\ & & & & & & 50 & & \\ & & \text{(small)} & & & & & 50 & \end{bmatrix}, \tag{6.48}$$

for which a diagonal preconditioner can be used

$$P = \begin{bmatrix} 1/2 & & & & & & & & \\ & 1/2 & & & & & & & \\ & & \ddots & & & & & & \\ & & & 1/6 & & & & & \\ & & & & 1/6 & & & & \\ & & & & & \ddots & & & \\ & & & & & & 1/50 & & \\ & & & & & & & 1/50 & \end{bmatrix}. \tag{6.49}$$

The preconditioned matrix will now have the form

$$M' = \begin{bmatrix} 1 & & & & & & & & \\ & 1 & & & \text{(small)} & & & & \\ & & \ddots & & & & & & \\ & & & 1 & & & & & \\ & & & & 1 & & & & \\ & & & & & \ddots & & & \\ & & & & & & 1 & & \\ & & \text{(small)} & & & & & 1 & \end{bmatrix}, \tag{6.50}$$

so that the Gershgorin circles can all be found within a small radius of unity as sketched in fig. 6.8.

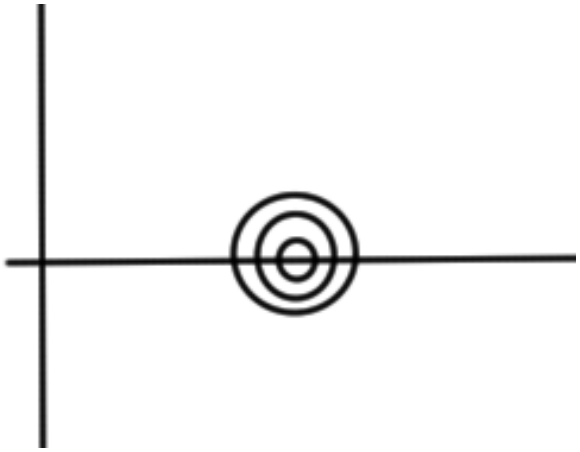


Figure 6.8: Gershgorin circles after preconditioning.

6.12 PRECONDITIONED CONJUGATE GRADIENT

It is possible to avoid inverting the preconditioner by requiring that the LU decomposition of P be easily computed. Then

$$P\mathbf{z}^k = \mathbf{r}^k \tag{6.51}$$

can be solved by successive forward and back substitution.
More on slides...

6.13 PROBLEMS

Exercise 6.1 Conjugate gradient and Gershgorin circles.

a. Consider an arbitrary circuit made by positive resistors and independent DC current sources. Prove, mathematically, that its nodal matrix is always symmetric and positive semi-definite.

b. Consider an electrical network made by:

- $N \times N$ square grid of resistors of value R, where N is the number of resistors per edge. The grid nodes are numbered from 1 to $(N+1)^2$. Node 1 is a corner node
- resistor R_g between each node of the grid and the reference node.

- a non-ideal voltage source connected between node 1 and ground. The voltage source has value V_s and internal (series) resistance R_s
- three current sources between three randomly-selected nodes of the grid and ground. The source current must flow from the grid to the reference node as shown in fig. 6.9 (and not vice versa!) Choose the current source values randomly between 10 mA and 100 mA.

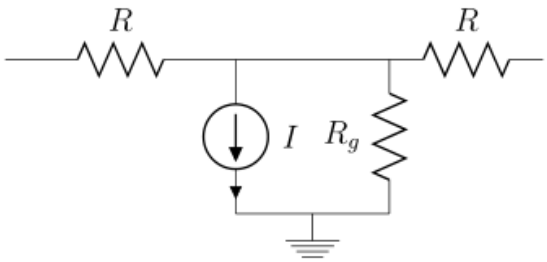


Figure 6.9: Subcircuit.

Write a Matlab routine that generates a SPICE compatible netlist for this system (you can reuse the code you developed for the first problem set). Generate the modified nodal analysis equations

$$\mathbf{G}\mathbf{x} = \mathbf{b} \tag{6.52}$$

for a grid with $N = 40$, $R = 0.1\Omega$, $R_g = 1M\Omega$, $V_s = 2V$, $R_s = 0.1\Omega$. Then, write in Matlab your own routine for the conjugate gradient method. Give to the user the possibility of specifying a preconditioning matrix P. The routine shall stop iterations when the residual norm satisfies

$$\frac{\|\mathbf{G}\mathbf{x} - \mathbf{b}\|_2}{\|\mathbf{b}\|_2} < \epsilon \tag{6.53}$$

where ϵ is a threshold specified by the user. Use the conjugate gradient method to solve modified nodal analysis eq. (6.52) of the grid.
Does the conjugate gradient method converge?

c. Discuss if the conjugate gradient method can be applied to the modified nodal analysis equations of the grid. Support your answer with some numerical results.

d. Suggest a transformation of the grid that will lead to a circuit equivalent to the original one, but for which conjugate gradient can be used. We call this new circuit "2D grid". You will have to use it for all the questions that follow.

e. Solve the "2D grid" circuit using three methods:
   - your own LU decomposition
   - the conjugate gradient method with $\epsilon = 10^{-3}$
   - the conjugate gradient method with $\epsilon = 10^{-3}$ and a tri-diagonal preconditioner $\mathbf{P}$

   for increasing N. Use $R = 0.1\Omega$. Plot the CPU time taken by the three methods vs N, and the number of iterations taken by the two iterative methods. Explore a range of N compatible with your PC speed and memory, but make sure to reach fairly large values of N.

f. Does preconditioning reduce the number of iterations required by conjugate gradient? Does preconditioning reduce CPU time?

g. Try to make your code for the conjugate gradient method as efficient as possible. Show the improvements that you have obtained.

h. Generate nodal analysis eq. (6.52) for "2D grid" with $N = 20$. Plot the eigenvalues of $\mathbf{G}$ and the Gershgorin circles in three cases: $R = 0.1\Omega$, $R = 1\Omega$, $R = 10\Omega$. Verify Gershgorin Circle theorem in the three cases.

i. Repeat the previous point for the preconditioned nodal matrix, and compare the circles obtained in the two cases.

j. Let $N = 20$, and solve "2D grid" with conjugate gradient with and without preconditioning (use the tri-diagonal preconditioner). Plot the normalized norm of the residual

$$\frac{\|\mathbf{G}\mathbf{x} - \mathbf{b}\|_2}{\|\mathbf{b}\|_2} \tag{6.54}$$

   versus the iteration counter for three cases: $R = 0.1\Omega$, $R = 1\Omega$, $R = 10\Omega$. Discuss your findings in the light of Gershgorin circles.

Answer for Exercise 6.1

PROBLEM SET RELATED MATERIAL REDACTED IN THIS DOCUMENT.PLEASE FEEL FREE TO EMAIL ME FOR THE FULL VERSION IF YOU AREN'T TAKING ECE1254.

[redacted] END-REDACTION

7 SOLUTION OF NONLINEAR SYSTEMS

7.1 NONLINEAR SYSTEMS

On slides, some examples to motivate:

- struts
- fluids
- diode (exponential)
  Example in fig. 7.1.

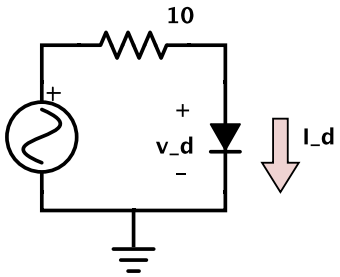


Figure 7.1: Diode circuit.

$$I_\mathrm{d} = I_\mathrm{s}\left(e^{V_\mathrm{d}/V_\mathrm{T}} - 1\right) = \frac{10 - V_\mathrm{d}}{10}. \tag{7.1}$$

7.2 RICHARDSON AND LINEAR CONVERGENCE

Seeking the exact solution x^* for

$$f(x^*) = 0, \tag{7.2}$$

Suppose that

$$x^{k+1} = x^k + f(x^k) \tag{7.3}$$

If $f(x^k) = 0$ then the iteration has converged at $x^k = x^*$.

Convergence analysis Write the iteration equations at a sample point and the solution as

$$x^{k+1} = x^k + f(x^k) \tag{7.4a}$$

$$x^* = x^* + \underbrace{f(x^*)}_{=0} \tag{7.4b}$$

The difference is

$$x^{k+1} - x^* = x^k - x^* + \left(f(x^k) - f(x^*)\right). \tag{7.5}$$

The last term can be quantified using the mean value theorem B.2, giving

$$\begin{aligned} x^{k+1} - x^* &= x^k - x^* + \left.\frac{\partial f}{\partial x}\right|_{\tilde{x}} \left(x^k - x^*\right) \\ &= \left(x^k - x^*\right)\left(1 + \left.\frac{\partial f}{\partial x}\right|_{\tilde{x}}\right). \end{aligned} \tag{7.6}$$

The absolute value is thus

$$\left|x^{k+1} - x^*\right| = \left|x^k - x^*\right|\left|1 + \left.\frac{\partial f}{\partial x}\right|_{\tilde{x}}\right|. \tag{7.7}$$

Convergence is obtained when $\left|1 + \left.\frac{\partial f}{\partial x}\right|_{\tilde{x}}\right| < 1$ in the iteration region. This could easily be highly dependent on the initial guess.

A more accurate statement is that convergence is obtained provided

$$\left|1 + \frac{\partial f}{\partial x}\right| \leq \gamma < 1 \qquad \forall \tilde{x} \in [x^* - \delta, x^* + \delta], \tag{7.8}$$

and $\left|x^0 - x^*\right| < \delta$. This is illustrated in fig. 7.2.

It could very easily be difficult to determine the convergence regions.

There are some problems

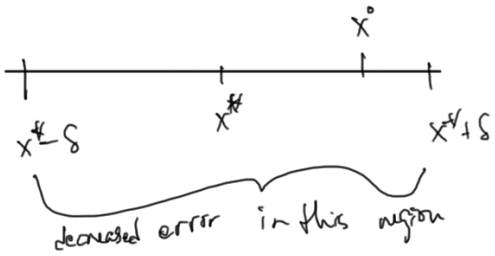


Figure 7.2: Convergence region.

- Convergence is only linear
- $x, f(x)$ are not in the same units (and potentially of different orders). For example, x could be a voltage and $f(x)$ could be a circuit current.
- (more on slides)

Examples where this may be desirable include

- Spice Gummal Poon transistor model. Lots of diodes, ...
- Mosfet model (30 page spec, lots of parameters).

7.3 NEWTON'S METHOD

The core idea of this method is sketched in fig. 7.3. To find the intersection with the x-axis, follow the slope closer to the intersection.

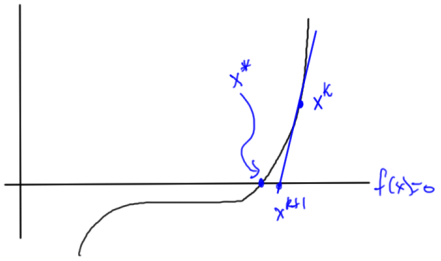


Figure 7.3: Newton's method.

To do this, expand $f(x)$ in Taylor series to first order around x^k, and then solve for $f(x) = 0$ in that approximation

$$f(x^{k+1}) \approx f(x^k) + \left.\frac{\partial f}{\partial x}\right|_{x^k} \left(x^{k+1} - x^k\right) = 0. \tag{7.9}$$

This gives

$$\boxed{x^{k+1} = x^k - \frac{f(x^k)}{\left.\frac{\partial f}{\partial x}\right|_{x^k}}.} \tag{7.10}$$

Example 7.1: Newton's method.

For the solution of

$$f(x) = x^3 - 2, \tag{7.11}$$

it was found

Table 7.1: Numerical Newton's method example.

k	x^k	$\lvert x^k - x^* \rvert$
0	10	8.74
1	·	5.4
⋮	·	⋮
8	·	1.7×10^{-3}
9	·	2.4×10^{-6}
10	·	6.6×10^{-12}

The error tails off fast as illustrated roughly in fig. 7.4.

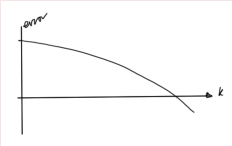


Figure 7.4: Error by iteration.

Convergence analysis The convergence condition is

$$0 = f(x^k) + \left.\frac{\partial f}{\partial x}\right|_{x^k} \left(x^{k+1} - x^k\right). \tag{7.12}$$

The Taylor series for f around x^k, using a mean value formulation is

$$f(x) = f(x^k) + \left.\frac{\partial f}{\partial x}\right|_{x^k} \left(x - x^k\right). + \frac{1}{2} \left.\frac{\partial^2 f}{\partial x^2}\right|_{\tilde{x} \in [x^*, x^k]} \left(x - x^k\right)^2. \tag{7.13}$$

Evaluating at x^* gives

$$0 = f(x^k) + \left.\frac{\partial f}{\partial x}\right|_{x^k} \left(x^* - x^k\right). + \frac{1}{2} \left.\frac{\partial^2 f}{\partial x^2}\right|_{\tilde{x} \in [x^*, x^k]} \left(x^* - x^k\right)^2, \tag{7.14}$$

and subtracting this from eq. (7.12) leaves

$$0 = \left.\frac{\partial f}{\partial x}\right|_{x^k} \left(x^{k+1} - \cancel{x^k} - x^* + \cancel{x^k}\right) - \frac{1}{2} \left.\frac{\partial^2 f}{\partial x^2}\right|_{\tilde{x}} \left(x^* - x^k\right)^2. \tag{7.15}$$

Solving for the difference from the solution, the error is

$$x^{k+1} - x^* = \frac{1}{2} \left(\frac{\partial f}{\partial x}\right)^{-1} \left.\frac{\partial^2 f}{\partial x^2}\right|_{\tilde{x}} \left(x^k - x^*\right)^2, \tag{7.16}$$

or in absolute value

$$\left|x^{k+1} - x^*\right| = \frac{1}{2} \left|\frac{\partial f}{\partial x}\right|^{-1} \left|\frac{\partial^2 f}{\partial x^2}\right| \left|x^k - x^*\right|^2. \tag{7.17}$$

Convergence is quadratic in the error from the previous iteration. There will be trouble if the derivative goes small at any point in the iteration region. For example in fig. 7.5, the iteration could easily end up in the zero derivative region.

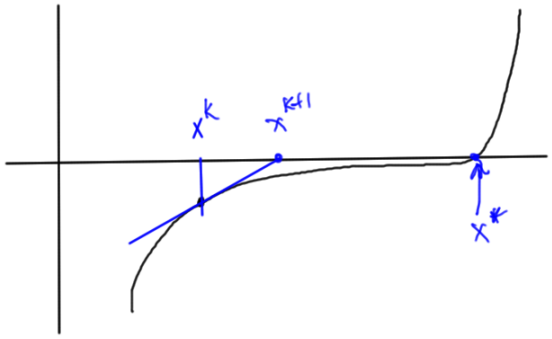


Figure 7.5: Newton's method with small derivative region.

When to stop iteration One way to check is to look to see if the difference

$$\left\| x^{k+1} - x^k \right\| < \epsilon_{\Delta x}, \tag{7.18}$$

however, when the function has a very step slope this may not be sufficient unless the trial solution is also substituted to see if the desired match has been achieved.

Alternatively, if the slope is shallow as in fig. 7.5, then checking for just $\left| f(x^{k+1} \right| < \epsilon_f$ may also put the iteration off target.

Finally, a relative error check to avoid false convergence may also be required. Figure 7.6 shows both absolute convergence criteria

$$\left| x^{k+1} - x^k \right| < \epsilon_{\Delta x} \tag{7.19a}$$

$$\left| f(x^{k+1}) \right| < \epsilon_f. \tag{7.19b}$$

A small relative difference may also be required

$$\frac{\left| x^{k+1} - x^k \right|}{\left| x^k \right|} < \epsilon_{f,r}. \tag{7.20}$$

This can become problematic in real world engineering examples such as to diode of fig. 7.7, containing both shallow regions and fast growing or dropping regions.

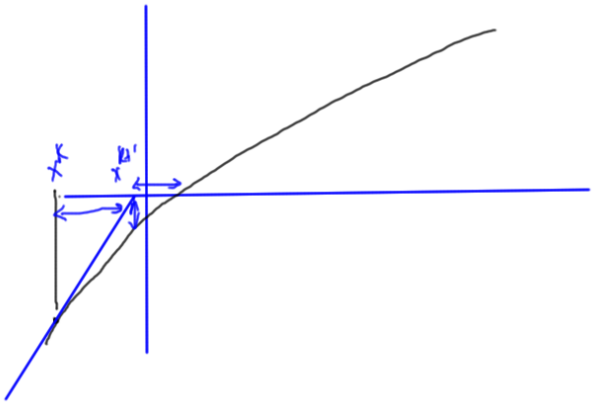


Figure 7.6: Possible relative error difference required.

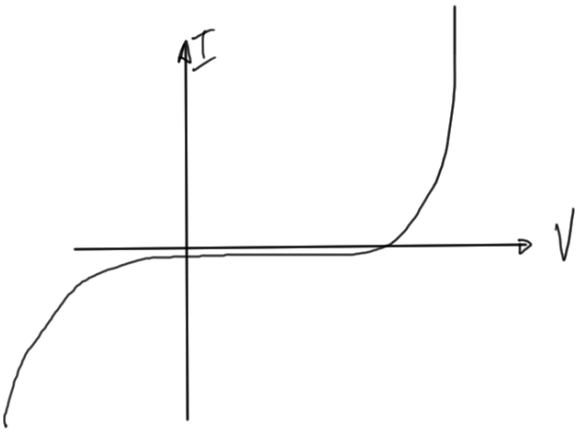


Figure 7.7: Diode current curve.

7.4 SOLUTION OF N NONLINEAR EQUATIONS IN N UNKNOWNS

It is now time to move from solutions of nonlinear functions in one variable:

$$f(x^*) = 0, \tag{7.21}$$

to multivariable systems of the form

$$\begin{aligned} f_1(x_1, x_2, \cdots, x_N) &= 0 \\ &\vdots \quad , \\ f_N(x_1, x_2, \cdots, x_N) &= 0 \end{aligned} \tag{7.22}$$

where the unknowns are

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}. \tag{7.23}$$

Form the vector F

$$F(\mathbf{x}) = \begin{bmatrix} f_1(x_1, x_2, \cdots, x_N) \\ \vdots \\ f_N(x_1, x_2, \cdots, x_N) \end{bmatrix}, \tag{7.24}$$

so that the equation to solve is

$$\boxed{F(\mathbf{x}) = 0.} \tag{7.25}$$

The Taylor expansion of F around point $\mathbf{x}_0$ is

$$F(\mathbf{x}) = F(\mathbf{x}_0) + \boxed{J_F(\mathbf{x}_0)}\,(\mathbf{x} - \mathbf{x}_0)\,, \tag{7.26}$$

Jacobian

where the Jacobian is

$$J_F(\mathbf{x}_0) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_N} \\ & \ddots & \\ \frac{\partial f_N}{\partial x_1} & \cdots & \frac{\partial f_N}{\partial x_N} \end{bmatrix} \tag{7.27}$$

7.5 MULTIVARIABLE NEWTON'S ITERATION

Given $\mathbf{x}^k$, expand $F(\mathbf{x})$ around $\mathbf{x}^k$

$$F(\mathbf{x}) \approx F(\mathbf{x}^k) + J_F(\mathbf{x}^k)\left(\mathbf{x} - \mathbf{x}^k\right) \tag{7.28}$$

Applying the approximation

$$0 = F(\mathbf{x}^k) + J_F(\mathbf{x}^k)\left(\mathbf{x}^{k+1} - \mathbf{x}^k\right), \tag{7.29}$$

multiplying by the inverse Jacobian, and some rearranging gives

$$\boxed{\mathbf{x}^{k+1} = \mathbf{x}^k - J_F^{-1}(\mathbf{x}^k)F(\mathbf{x}^k).} \tag{7.30}$$

The algorithm is

Algorithm 7.1: Newton's method.

Guess $\mathbf{x}^0, k = 0$.
repeat
  Compute F and J_F at $\mathbf{x}^k$
  Solve linear system $J_F(\mathbf{x}^k)\Delta\mathbf{x}^k = -F(\mathbf{x}^k)$
  $\mathbf{x}^{k+1} = \mathbf{x}^k + \Delta\mathbf{x}^k$
  $k = k + 1$
until converged

As with one variable, there are a number of convergence conditions to check for

$$\begin{aligned} \left\| \Delta \mathbf{x}^k \right\| &< \epsilon_1 \\ \left\| F(\mathbf{x}^{k+1}) \right\| &< \epsilon_2 \\ \frac{\left\| \Delta \mathbf{x}^k \right\|}{\left\| \mathbf{x}^{k+1} \right\|} &< \epsilon_3 \end{aligned} \tag{7.31}$$

Typical termination is some multiple of eps, where eps is the machine precision. This may be something like:

$$4 \times N \times \text{eps}, \tag{7.32}$$

where N is the "size of the problem". Sometimes there may be physically meaningful values for the problem that define the desirable iteration cutoff. For example, for a voltage problem, it might be that precisions greater than a millivolt are not of interest.

7.6 AUTOMATIC ASSEMBLY OF EQUATIONS FOR NONLINEAR SYSTEM

Nonlinear circuits Start off by considering a nonlinear resistor, designated within a circuit as sketched in fig. 7.8.

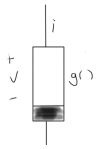


Figure 7.8: Non-linear resistor.

Example: diode, with $i = g(v)$, such as

$$i = I_0 \left(e^{v/\eta V_T} - 1 \right). \tag{7.33}$$

Consider the example circuit of fig. 7.9. KCL's at each of the nodes are

1. $I_A + I_B + I_D - I_s = 0$
2. $-I_B + I_C - I_D = 0$

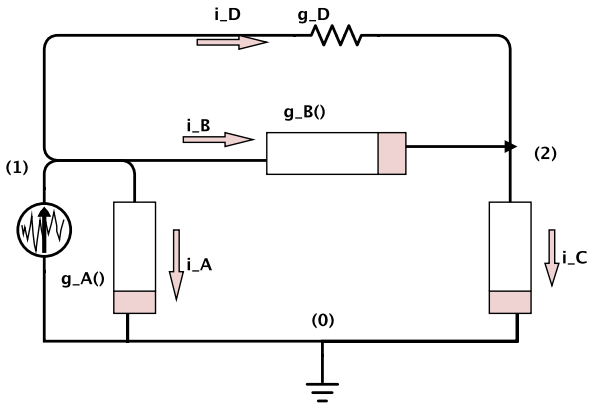


Figure 7.9: Example circuit.

Introducing the constitutive equations this is

1. $g_A(V_1) + g_B(V_1 - V_2) + g_D(V_1 - V_2) - I_s = 0$
2. $-g_B(V_1 - V_2) + g_C(V_2) - g_D(V_1 - V_2) = 0$

In matrix form this is

$$\begin{bmatrix} g_D & -g_D \\ -g_D & g_D \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} + \begin{bmatrix} g_A(V_1) & +g_B(V_1 - V_2) & -I_s \\ & -g_B(V_1 - V_2) \quad +g_C(V_2) & \end{bmatrix} = 0. \tag{7.34}$$

The entire system can be written in matrix form as

$$\boxed{F(\mathbf{x}) = G\mathbf{x} + F'(\mathbf{x}) = 0.} \tag{7.35}$$

The first term, a product of a nodal matrix G represents the linear subnetwork, and is filled with the stamps that are already familiar.

The second term encodes the relationships of the nonlinear subnetwork. This nonlinear component has been marked with a prime to distinguish it from the complete network function that includes both linear and nonlinear elements.

Observe the similarity with the previous stamp analysis. With $g_A()$ connected on one end to ground it occurs only in the resulting vector, whereas the nonlinear elements connected to two non-zero nodes in the network occur once with each sign.

Figure 7.10: Non-linear resistor circuit element.

Stamp for nonlinear resistor For the nonlinear circuit element of fig. 7.10.

$$F'(\mathbf{x}) = \begin{matrix} n_1 \rightarrow \\ n_2 \rightarrow \end{matrix} \begin{bmatrix} +g(V_{n_1} - V_{n_2}) \\ -g(V_{n_1} - V_{n_2}) \end{bmatrix} \tag{7.36}$$

Stamp for Jacobian

$$J_F(\mathbf{x}^k) = G + J_{F'}(\mathbf{x}^k). \tag{7.37}$$

Here the stamp for the Jacobian, an $N \times N$ matrix, is

$$J_{F'}(\mathbf{x}^k) = \begin{matrix} \\ 1 \\ \vdots \\ n_1 \\ n_2 \\ \vdots \\ N \end{matrix} \begin{matrix} \begin{matrix} V_1 & \cdots & V_{n_1} & & V_{n_2} & \cdots & V_N \end{matrix} \\ \begin{bmatrix} & & & & & & \\ & & \frac{\partial g(V_{n_1} - V_{n_2})}{\partial V_{n_1}} & & \frac{\partial g(V_{n_1} - V_{n_2})}{\partial V_{n_2}} & & \\ & & -\frac{\partial g(V_{n_1} - V_{n_2})}{\partial V_{n_1}} & & -\frac{\partial g(V_{n_1} - V_{n_2})}{\partial V_{n_2}} & & \\ & & & & & & \end{bmatrix} \end{matrix}. \tag{7.38}$$

7.7 DAMPED NEWTON'S METHOD

Figure 7.11 illustrates a system that may have troublesome oscillation, depending on the initial guess selection.

Large steps can be dangerous, and can be avoided by modifying Newton's method as follows

The algorithm is

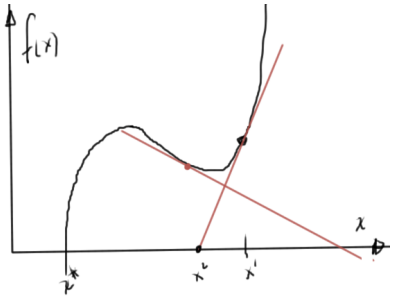


Figure 7.11: Oscillatory Newton's iteration.

Algorithm 7.2: Damped Newton's method.

Guess $\mathbf{x}^0, k = 0$.
repeat
  Compute F and J_F at $\mathbf{x}^k$
  Solve linear system $J_F(\mathbf{x}^k)\Delta\mathbf{x}^k = -F(\mathbf{x}^k)$
  $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha^k\Delta\mathbf{x}^k$
  $k = k + 1$
until converged

This is the standard Newton's method when $\alpha^k = 1$.

7.8 CONTINUATION PARAMETERS

Newton's method converges given a close initial guess. A sequence of problems can be generated where the previous problem generates a good initial guess for the next problem.

An example is a heat conducting bar, with a final heat distribution. The numeric iteration can be started with $T = 0$. The temperature can be gradually increased until the final desired heat distribution is increased.

To solve a general system of the form

$$F(\mathbf{x}) = 0. \tag{7.39}$$

modify this problem by introducing a parameter

$$\tilde{F}(\mathbf{x}(\lambda), \lambda) = 0, \tag{7.40}$$

where

- $\tilde{F}(\mathbf{x}(0), 0) = 0$ is easy to solve
- $\tilde{F}(\mathbf{x}(1), 1) = 0$ is equivalent to $F(\mathbf{x}) = 0$.
- (more on slides)

The source load stepping algorithm is

- Solve $\tilde{F}(\mathbf{x}(0), 0) = 0$, with $\mathbf{x}(\lambda_{\text{prev}} = \mathbf{x}(0)$
- (more on slides)

This can still have problems, for example, when the parameterization is multivalued as in fig. 7.12.

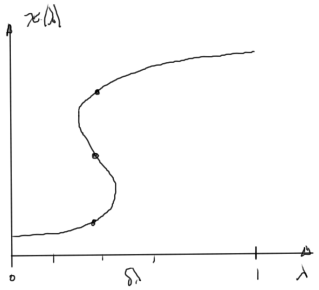


Figure 7.12: Multivalued parameterization.

It is possible to adjust λ so that the motion is along the parameterization curve.

7.9 SINGULAR JACOBIANS

(mostly on slides)

There is the possibility of singular Jacobians to consider. FIXME: not sure how this system represented that. Look on slides.

$$\tilde{f}(v(\lambda), \lambda) = i(v) - \frac{1}{R}(v - \lambda V_s) = 0. \tag{7.41}$$

An alternate continuation scheme uses

$$\tilde{F}(\mathbf{x}(\lambda), \lambda) = \lambda F(\mathbf{x}(\lambda)) + (1 - \lambda)\mathbf{x}(\lambda). \tag{7.42}$$

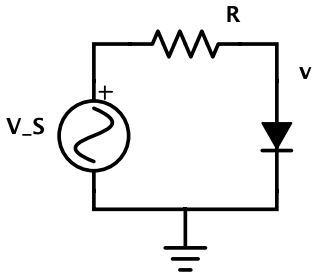


Figure 7.13: Diode system that results in singular Jacobian.

This scheme has

$$\tilde{F}(\mathbf{x}(0), 0) = 0 \tag{7.43a}$$

$$\tilde{F}(\mathbf{x}(1), 1) = F(\mathbf{x}(1)), \tag{7.43b}$$

and for one variable, easy to compute Jacobian at the origin, or the original Jacobian at $\lambda = 1$

$$\frac{\partial \tilde{F}}{\partial x}(x(0), 0) = I \tag{7.44a}$$

$$\frac{\partial \tilde{F}}{\partial x}(x(1), 1) = \frac{\partial F}{\partial x}(x(1)) \tag{7.44b}$$

7.10 STRUTS AND JOINTS, NODE BRANCH FORMULATION

Consider the simple strut system of fig. 7.14 again.

The unknowns are

1. Forces

   At each of the points there is a force with two components

   $$\mathbf{f}_A = \left(f_{A,x}, f_{A,y}\right) \tag{7.45}$$

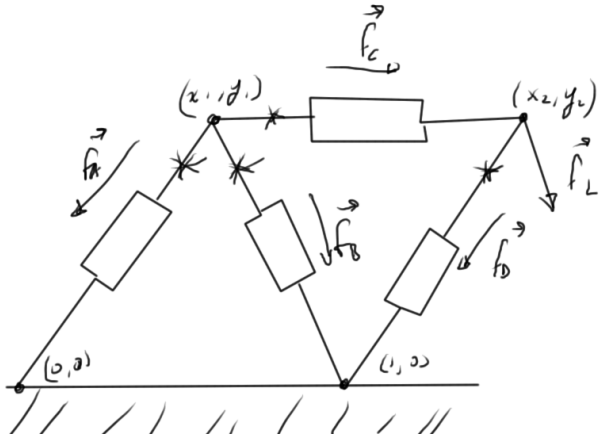


Figure 7.14: Simple strut system.

Construct a total force vector

$$\mathbf{f} = \begin{bmatrix} f_{\mathrm{A},x} \\ f_{\mathrm{A},y} \\ f_{\mathrm{B},x} \\ f_{\mathrm{B},y} \\ \vdots \end{bmatrix} \tag{7.46}$$

2. Positions of the joints

$$\mathbf{r} = \begin{bmatrix} x_1 \\ y_1 \\ y_1 \\ y_2 \\ \vdots \end{bmatrix} \tag{7.47}$$

The given variables are

1. The load force $\mathbf{f}_{\mathrm{L}}$.
2. The joint positions at rest.
3. parameter of struts.

Conservation laws The conservation laws are

$$\mathbf{f}_A + \mathbf{f}_B + \mathbf{f}_C = 0 \tag{7.48a}$$

$$-\mathbf{f}_C + \mathbf{f}_D + \mathbf{f}_L = 0 \tag{7.48b}$$

which can be put into matrix form

$$\begin{matrix} x_1 \\ y_1 \\ x_2 \\ y_2 \end{matrix} \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_{A,x} \\ f_{A,y} \\ f_{B,x} \\ f_{B,y} \\ f_{C,x} \\ f_{C,y} \\ f_{D,x} \\ f_{D,y} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -f_{L,x} \\ -f_{L,y} \end{bmatrix} \tag{7.49}$$

Here the block matrix is called the incidence matrix **A**. The system can be expressed as

$$\boxed{A\mathbf{f} = \mathbf{f}_L.} \tag{7.50}$$

Constitutive laws Given a pair of nodes as in fig. 7.15.

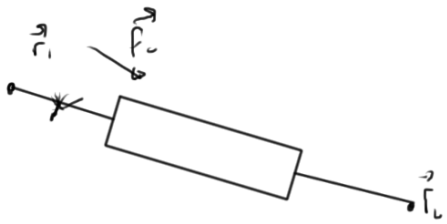


Figure 7.15: Strut spanning nodes.

each component has an equation relating the reaction forces of that strut based on the positions

$$f_{c,x} = S_x\left(x_1 - x_2, y_1 - y_2, p_c\right) \tag{7.51a}$$

$$f_{c,y} = S_y\left(x_1 - x_2, y_1 - y_2, p_c\right), \tag{7.51b}$$

where p_c represent the parameters of the system. Write

$$\mathbf{f} = \begin{bmatrix} f_{\mathrm{A},x} \\ f_{\mathrm{A},y} \\ f_{\mathrm{B},x} \\ f_{\mathrm{B},y} \\ \vdots \end{bmatrix} = \begin{bmatrix} S_x\left(x_1 - x_2, y_1 - y_2, p_c\right) \\ S_y\left(x_1 - x_2, y_1 - y_2, p_c\right) \\ \vdots \end{bmatrix}, \tag{7.52}$$

or

$$\boxed{\mathbf{f} = S(\mathbf{r})} \tag{7.53}$$

Putting the pieces together The node branch formulation is

$$\begin{aligned} A\mathbf{f} - \mathbf{f}_{\mathrm{L}} &= 0 \\ \mathbf{f} - S(\mathbf{r}) &= 0 \end{aligned} \tag{7.54}$$

Elimination of the forces (the equivalent of currents in this system) produces the nodal formulation

$$\boxed{AS(\mathbf{r}) - \mathbf{f}_{\mathrm{L}} = 0} \tag{7.55}$$

This nodal formulation cannot be used with struts that are so stiff that the positions of some of the nodes are fixed. As before this can be worked around by introducing an additional unknown for each component of such a strut.

The cost of this Jacobian calculation required to approximate this system can still potentially be expensive.

7.11 PROBLEMS

Exercise 7.1 Newton's method.

Consider the circuit in fig. 7.16. By setting $R = 1$, $I_{s1} = 5$, $I_{s2} = 10^{-6}$, and using diode parameters $I_0 = 10^{-6}$ and $1/v_T = 80$, we can obtain the following nodal analysis equation.

$$x + 10^{-6} e^{80x} = 5 \tag{7.56}$$

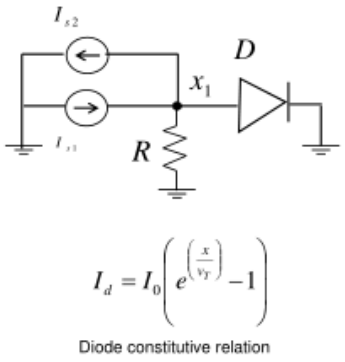


Figure 7.16: Circuit.

Write a program to use Newton's method to solve this equation for x, and use the program to answer the questions below:

a. Starting with an initial guess of $x = 0$, how many Newton iterations are required to compute an x that is within 10^{-6} of the exact solution?

b. How did you determine the accuracy of your solution?

c. Try using a simple source-stepping style continuation scheme to solve the system of equations, as in

$$x + 10^{-6} e^{80x} = \lambda * 5, \tag{7.57}$$

where λ is a continuation parameter that varies from zero to one. How many Newton iterations do you need to solve the original problem?

d. Compare the two approaches discussed in the slides for generating an initial guess for each step of your continuation scheme. That is, compare using just the previous step's converged solution

$$x^0(\lambda) = x(\lambda_{\text{prev}}), \tag{7.58}$$

to updating the converged solution using the derivative with respect to λ, that is:

$$x^0(\lambda) = x(\lambda_{\text{prev}}) + \frac{dx}{d\lambda}\left(\lambda_{\text{prev}}\right)\delta\lambda. \tag{7.59}$$

Does using the derivative help?

Answer for Exercise 7.1

PROBLEM SET RELATED MATERIAL REDACTED IN THIS DOCUMENT.PLEASE FEEL FREE TO EMAIL ME FOR THE FULL VERSION IF YOU AREN'T TAKING ECE1254.

[redacted] END-REDACTION

Exercise 7.2 Finite element methods.

When modeling distributions of charged particles governed by drift-diffusion equations, equilibrium analysis typically leads to the following nonlinear Poisson equation

$$-\frac{\partial^2 \psi}{\partial x^2} = -\left(e^{\psi(x)} - e^{-\psi(x)}\right), \tag{7.60}$$

where we will consider the interval $x \in [0,1]$ with the boundary conditions $\psi(0) = -V$ and $\psi(1) = V$. If a simple finite-difference scheme is used to solve the nonlinear Poisson equation on an N-node grid, the discrete equations are

$$2\psi_i - \psi_{i+1} - \psi_{i-1} + \Delta x^2 \left(e^{\psi_i} - e^{-\psi_i}\right) = 0, \tag{7.61}$$

for $i \in [2, \cdots, N-1]$,

$$2\psi_1 - \psi_2 - (-V) + \Delta x^2 \left(e^{\psi_1} - e^{-\psi_1}\right) = 0, \tag{7.62}$$

and

$$2\psi_N - \psi_{N-1} - V + \Delta x^2 \left(e^{\psi_N} - e^{-\psi_N}\right) = 0. \tag{7.63}$$

Note, $\Delta x = 1/(N+1)$ and not $1/(N-1)$. The nodes at $x = 0$ and $x = 1$ are not included in the discretization, but rather enter through the boundary conditions.

a. Prove that the Jacobian associated with the above discretized equations is non-singular regardless of the values for the ψ_i's.
b. Based on this observation, how do you expect damped Newton methods will perform, when applied to solving this problem?
c. Solve the above equations with a multidimensional Newton's method using a zero initial guess for the ψ_i's, and $N = 100$. Demonstrate that your program achieves quadratic convergence.

d. and determine the number of Newton iterations required to insure one part in 10^6 accuracy in the solution for two cases: when $V = 1$ and when $V = 20$.

e. What happens when $V = 100$?

Answer for Exercise 7.2

PROBLEM SET RELATED MATERIAL REDACTED IN THIS DOCUMENT.PLEASE FEEL FREE TO EMAIL ME FOR THE FULL VERSION IF YOU AREN'T TAKING ECE1254.[redacted]. END-REDACTION

8 TIME DEPENDENT SYSTEMS

8.1 ASSEMBLING EQUATIONS AUTOMATICALLY FOR DYNAMICAL SYSTEMS

Example 8.1: RC circuit.

The method can be demonstrated well by example. Consider the RC circuit fig. 8.1 for which the capacitors introduce a time dependence

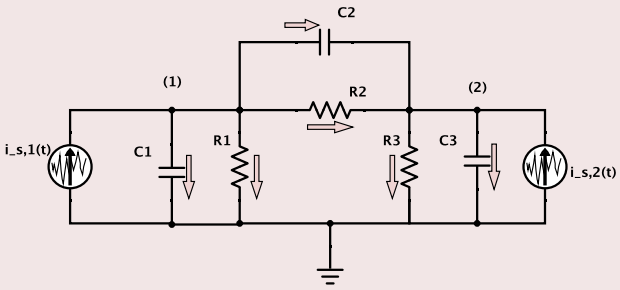


Figure 8.1: RC circuit.

The unknowns are $v_1(t), v_2(t)$.
The equations (KCLs) at each of the nodes are

1. $\frac{v_1(t)}{R_1} + C_1\frac{dv_1}{dt} + \frac{v_1(t)-v_2(t)}{R_2} + C_2\frac{d(v_1-v_2)}{dt} - i_{s,1}(t) = 0$
2. $\frac{v_2(t)-v_1(t)}{R_2} + C_2\frac{d(v_2-v_1)}{dt} + \frac{v_2(t)}{R_3} + C_3\frac{dv_2}{dt} - i_{s,2}(t) = 0$

This has the matrix form

$$\begin{bmatrix} Z_1+Z_2 & -Z_2 \\ -Z_2 & Z_2+Z_3 \end{bmatrix} \begin{bmatrix} v_1(t) \\ v_2(t) \end{bmatrix} + \begin{bmatrix} C_1+C_2 & -C_2 \\ -C_2 & C_2+C_3 \end{bmatrix} \begin{bmatrix} \frac{dv_1(t)}{dt} \\ \frac{dv_2(t}{dt}) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} i_{s,1}(t) \\ i_{s,2}(t) \end{bmatrix} . \quad (8.1)$$

Observe that the capacitor between node 2 and 1 is associated with a stamp of the form

$$\begin{bmatrix} C_2 & -C_2 \\ -C_2 & C_2 \end{bmatrix}, \tag{8.2}$$

very much like the impedance stamps of the resistor node elements.

The RC circuit problem has the abstract form

$$\mathbf{G}\mathbf{x}(t) + \mathbf{C}\frac{d\mathbf{x}(t)}{dt} = \mathbf{B}\mathbf{u}(t), \tag{8.3}$$

which is more general than a state space equation of the form

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t). \tag{8.4}$$

Such a system may be represented diagrammatically as in fig. 8.2.

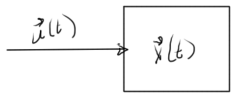


Figure 8.2: State space system.

The C factor in this capacitance system, is generally not invertible. For example, if consider a 10 node system with only one capacitor, for which C will be mostly zeros. In a state space system, in all equations have a derivative. All equations are dynamical.

The time dependent MNA system for the RC circuit above, contains a mix of dynamical and algebraic equations. This could, for example, be a pair of equations like

$$\frac{dx_1}{dt} + x_2 + 3 = 0 \tag{8.5a}$$

$$x_1 + x_2 + 3 = 0 \tag{8.5b}$$

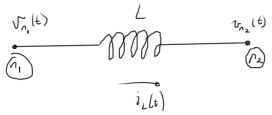


Figure 8.3: Inductor configuration.

How to handle inductors A pair of nodes that contains an inductor element, as in fig. 8.3, has to be handled specially.

The KCL at node 1 has the form

$$\cdots + i_{\mathrm{L}}(t) + \cdots = 0, \tag{8.6}$$

where

$$v_{n_1}(t) - v_{n_2}(t) = L\frac{di_{\mathrm{L}}}{dt}. \tag{8.7}$$

It is possible to express this in terms of i_{L}, the variable of interest

$$i_{\mathrm{L}}(t) = \frac{1}{L}\int_0^t \left(v_{n_1}(\tau) - v_{n_2}(\tau)\right) d\tau + i_{\mathrm{L}}(0). \tag{8.8}$$

Expressing the problem directly in terms of such integrals makes the problem harder to solve, since the usual differential equation toolbox cannot be used directly. An integro-differential toolbox would have to be developed. What can be done instead is to introduce an additional unknown for each inductor current derivative di_{L}/dt, for which an additional MNA row is introduced for that inductor scaled voltage difference.

8.2 NUMERICAL SOLUTION OF DIFFERENTIAL EQUATIONS

Consider the one variable system

$$Gx(t) + C\frac{dx}{dt} = Bu(t), \tag{8.9}$$

given an initial condition $x(0) = x_0$. Imagine that this system has the solution sketched in fig. 8.4.

Very roughly, the steps for solution are of the form

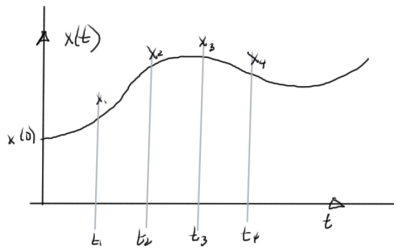


Figure 8.4: Discrete time sampling.

1. Discretize time
2. Aim to find the solution at $t_1, t_2, t_3, \cdots$
3. Use a finite difference formula to approximate the derivative.

There are various schemes that can be used to discretize, and compute the finite differences.

8.3 FORWARD EULER METHOD

One such scheme is to use the forward differences, as in fig. 8.5, to approximate the derivative

$$\dot{x}(t_n) \approx \frac{x_{n+1} - x_n}{\Delta t}. \tag{8.10}$$

Introducing this into eq. (8.9) gives

$$Gx_n + C\frac{x_{n+1} - x_n}{\Delta t} = Bu(t_n), \tag{8.11}$$

or

$$Cx_{n+1} = \Delta t Bu(t_n) - \Delta t Gx_n + Cx_n. \tag{8.12}$$

The coefficient C must be invertible, and the next point follows immediately

$$\boxed{x_{n+1} = \frac{\Delta t B}{C} u(t_n) + x_n \left(1 - \frac{\Delta t G}{C}\right)} \tag{8.13}$$

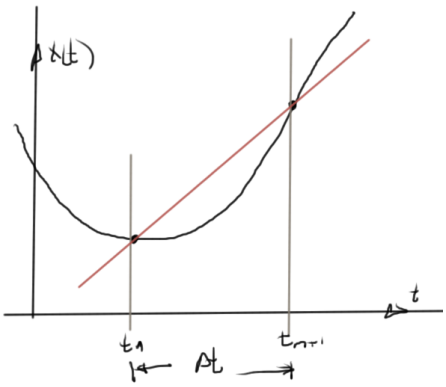


Figure 8.5: Forward difference derivative approximation.

8.4 BACKWARD EULER METHOD

Discretized time dependent partial differential equations were seen to have the form

$$\mathbf{G}\mathbf{x}(t) + \mathbf{C}\dot{\mathbf{x}}(t) = \mathbf{B}\mathbf{u}(t), \tag{8.14}$$

where $\mathbf{G}, \mathbf{C}, \mathbf{B}$ are matrices, and $\mathbf{u}(t)$ is a vector of sources.

The backward Euler method augments eq. (8.14) with an initial condition. For a one dimensional system such an initial condition could a zero time specification

$$Gx(t) + C\dot{x}(t) = Bu(t), \tag{8.15a}$$

$$x(0) = x_0 \tag{8.15b}$$

Discretizing time as in fig. 8.6.

The discrete derivative, using a backward difference, is

$$\dot{x}(t = t_n) \approx \frac{x_n - x_{n-1}}{\Delta t} \tag{8.16}$$

Evaluating eq. (8.15a) at $t = t_n$ is

$$Gx_n + C\dot{x}(t = t_n) = Bu(t_n), \tag{8.17}$$

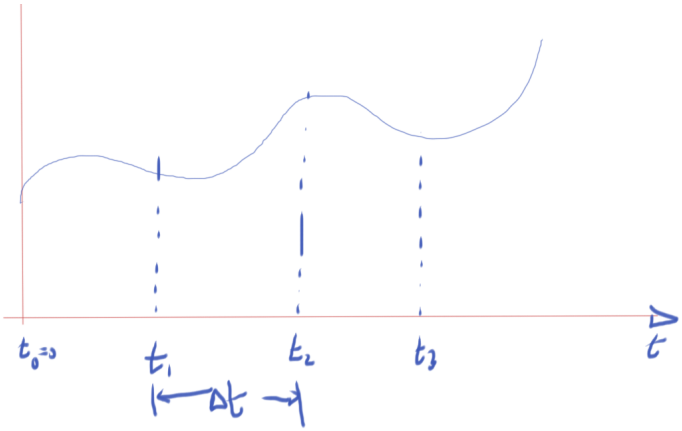


Figure 8.6: Discretized time.

or approximately

$$Gx_n + C\frac{x_n - x_{n-1}}{\Delta t} = Bu(t_n). \tag{8.18}$$

Rearranging

$$\left(G + \frac{C}{\Delta t}\right) x_n = \frac{C}{\Delta t} x_{n-1} + Bu(t_n). \tag{8.19}$$

Assuming that matrices G, C are constant, and Δt is fixed, a matrix inversion can be avoided, and a single LU decomposition can be used. For N sampling points (not counting $t_0 = 0$), N sets of backward and forward substitutions will be required to compute x_1 from x_0, and so forth.

Backwards Euler is an implicit method.

Recall that the forward Euler method gave

$$x_{n+1} = x_n \left(I - C^{-1}\Delta t G\right) + C^{-1}\Delta t B u(t_n) \tag{8.20}$$

This required

- C must be invertible.
- C must be cheap to invert, perhaps $C = I$, so that

$$x_{n+1} = (I - \Delta t G)\, x_n + \Delta t B u(t_n) \tag{8.21}$$

- This is an implicit method
- This can be cheap but unstable.

8.5 TRAPEZOIDAL RULE (TR)

The derivative can be approximated using an average of the pair of derivatives as illustrated in fig. 8.7.

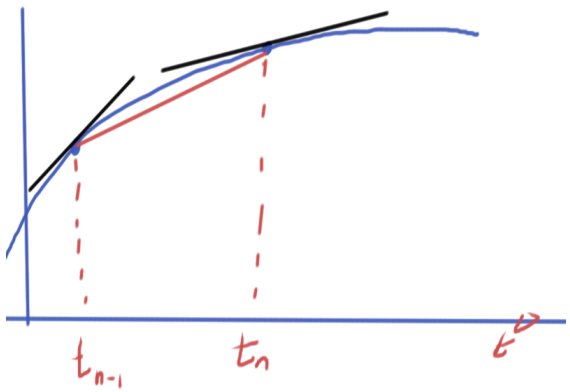


Figure 8.7: Trapezoidal derivative approximation.

$$\frac{x_n - x_{n-1}}{\Delta t} \approx \frac{\dot{x}(t_{n-1}) + \dot{x}(t_n)}{2}. \tag{8.22}$$

Application to eq. (8.15a) for t_{n-1}, t_n respectively gives

$$\begin{aligned} Gx_{n-1} + C\dot{x}(t_{n-1}) &= Bu(t_{n-1}) \\ Gx_n + C\dot{x}(t_n) &= Bu(t_n) \end{aligned} \tag{8.23}$$

Averaging these

$$G\frac{x_{n-1} + x_n}{2} + C\frac{\dot{x}(t_{n-1}) + \dot{x}(t_n)}{2} = B\frac{u(t_{n-1}) + u(t_n)}{2}, \tag{8.24}$$

and inserting the trapezoidal approximation

$$G\frac{x_{n-1} + x_n}{2} + C\frac{x_n - x_{n-1}}{\Delta t} = B\frac{u(t_{n-1}) + u(t_n)}{2}, \tag{8.25}$$

and a final rearrangement yields

$$\boxed{\left(G + \frac{2}{\Delta t}C\right) x_n = -\left(G - \frac{2}{\Delta t}C\right) x_{n_1} + B\left(u(t_{n-1}) + u(t_n)\right).} \tag{8.26}$$

This is

- also an implicit method.
- requires LU of $G + 2C/\Delta t$.
- more accurate than BE, for the same computational cost.

In all of these methods, accumulation of error is something to be very careful of, and in some cases such error accumulation can even be exponential.

This is effectively a way to introduce central differences. On the slides this is seen to be more effective at avoiding either artificial damping and error accumulation that can be seen in backwards and forwards Euler method respectively.

8.6 NONLINEAR DIFFERENTIAL EQUATIONS

Assume that the relationships between the zeroth and first order derivatives has the form

$$F\left(x(t), \dot{x}(t)\right) = 0 \tag{8.27a}$$

$$x(0) = x_0 \tag{8.27b}$$

The backward Euler method where the derivative approximation is

$$\dot{x}(t_n) \approx \frac{x_n - x_{n-1}}{\Delta t}, \tag{8.28}$$

can be used to solve this numerically, reducing the problem to

$$F\left(x_n, \frac{x_n - x_{n-1}}{\Delta t}\right) = 0. \tag{8.29}$$

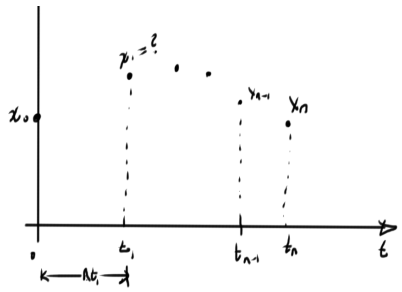


Figure 8.8: Possible solution points.

This can be solved with Newton's method. How do we find the initial guess for Newton's? Consider a possible system in fig. 8.8.

One strategy for starting each iteration of Newton's method is to base the initial guess for x_1 on the value x_0, and do so iteratively for each subsequent point. One can imagine that this may work up to some sample point x_n, but then break down (i.e. Newton's diverges when the previous value x_{n-1} is used to attempt to solve for x_n). At that point other possible strategies may work. One such strategy is to use an approximation of the derivative from the previous steps to attempt to get a better estimate of the next value. Another possibility is to reduce the time step, so the difference between successive points is reduced.

8.7 ANALYSIS, ACCURACY AND STABILITY ($\Delta t \to 0$)

Consider a differential equation

$$\dot{x}(t) = f(x(t), t) \tag{8.30a}$$

$$x(t_0) = x_0 \tag{8.30b}$$

A few methods of solution have been considered

(FE) $x_{n+1} - x_n = \Delta t f(x_n, t_n)$

(BE) $x_{n+1} - x_n = \Delta t f(x_{n+1}, t_{n+1})$

(TR) $x_{n+1} - x_n = \frac{\Delta t}{2} f(x_{n+1}, t_{n+1}) + \frac{\Delta t}{2} f(x_n, t_n)$

A common pattern can be observed, the generalization of which are called *linear multistep methods* (LMS) , which have the form

$$\sum_{j=-1}^{k-1} \alpha_j x_{n-j} = \Delta t \sum_{j=-1}^{k-1} \beta_j f(x_{n-j}, t_{n-j}) \tag{8.31}$$

The FE (explicit), BE (implicit), and TR methods are now special cases with

(FE) $\alpha_{-1} = 1, \alpha_0 = -1, \beta_{-1} = 0, \beta_0 = 1$

(BE) $\alpha_{-1} = 1, \alpha_0 = -1, \beta_{-1} = 1, \beta_0 = 0$

(TR) $\alpha_{-1} = 1, \alpha_0 = -1, \beta_{-1} = 1/2, \beta_0 = 1/2$

Here k is the number of timesteps used. The method is explicit if $\beta_{-1} = 0$.

Definition 8.1: Convergence.

With

$x(t)$: exact solution

x_n : computed solution

e_n : where $e_n = x_n - x(t_n)$, is the global error

The LMS method is convergent if

$$\max_{n, \Delta t \to 0} |x_n - t(t_n)| \to 0$$

Convergence: zero-stability and consistency (small local errors made at each iteration),
where zero-stability is "small sensitivity to changes in initial condition".

Definition 8.2: Consistency.

A local error R_{n+1} can be defined as

$$R_{n+1} = \sum_{j=-1}^{k-1} \alpha_j x(t_{n-j}) - \Delta t \sum_{j=-1}^{k-1} \beta_j f(x(t_{n-j}), t_{n-j}).$$

The method is consistent if

$$\lim_{\Delta t}\left(\max_{n}\left|\frac{1}{\Delta t}R_{n+1}\right|=0\right)$$

or $R_{n+1}\sim O(\Delta t^2)$

8.8 RESIDUAL FOR LMS METHODS

Mostly on slides: 12_ODS.pdf

Residual is illustrated in fig. 8.9, assuming that the iterative method was accurate until t_n

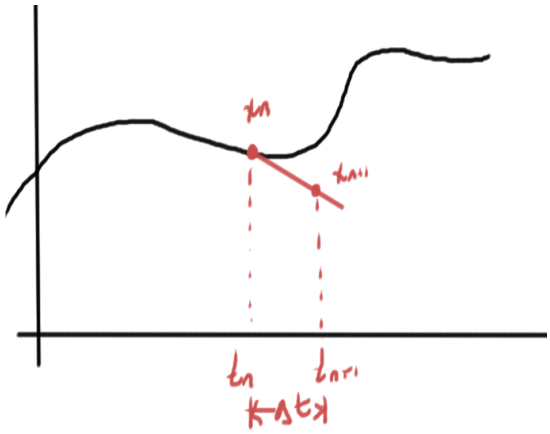


Figure 8.9: Residual illustrated.

Summary

FE : $R_{n+1}\sim(\Delta t)^2$. This is of order $p=1$.

BE : $R_{n+1}\sim(\Delta t)^2$. This is of order $p=1$.

TR : $R_{n+1}\sim(\Delta t)^3$. This is of order $p=2$.

BESTE : $R_{n+1}\sim(\Delta t)^4$. This is of order $p=3$.

8.9 GLOBAL ERROR ESTIMATE

Suppose $t \in [0,1]s$, with $N = 1/\Delta t$ intervals. For a method with local error of order $R_{n+1} \sim (\Delta t)^2$ the global error is approximately $NR_{n+1} \sim \Delta t$.

8.10 STABILITY

Recall that a linear multistep method (LMS) was a system of the form

$$\sum_{j=-1}^{k-1} \alpha_j x_{n-j} = \Delta t \sum_{j=-1}^{k-1} \beta_j f(x_{n-j}, t_{n-j}) \tag{8.32}$$

Consider a one dimensional test problem

$$\dot{x}(t) = \lambda x(t) \tag{8.33}$$

where as in fig. 8.10, $\mathrm{Re}(\lambda) < 0$ is assumed to ensure stability.

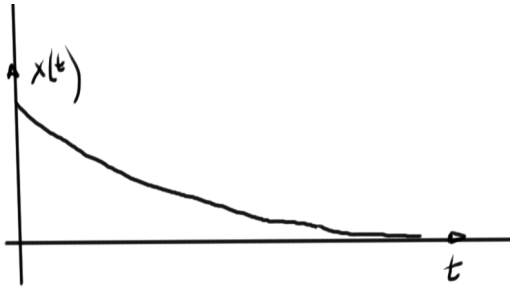


Figure 8.10: Stable system.

Linear stability theory can be thought of as asking the question: "Is the solution of eq. (8.33) computed by my LMS method also stable?"

Application of eq. (8.32) to eq. (8.33) gives

$$\sum_{j=-1}^{k-1} \alpha_j x_{n-j} = \Delta t \sum_{j=-1}^{k-1} \beta_j \lambda x_{n-j}, \tag{8.34}$$

or

$$\sum_{j=-1}^{k-1} \left(\alpha_j - \Delta \beta_j \lambda\right) x_{n-j} = 0. \tag{8.35}$$

With

$$\gamma_j = \alpha_j - \Delta\beta_j\lambda, \tag{8.36}$$

this expands to

$$\gamma_{-1}x_{n+1} + \gamma_0 x_n + \gamma_1 x_{n-1} + \cdots + \gamma_{k-1}x_{n-k}. \tag{8.37}$$

This can be seen as a

- discrete time system
- FIR filter

The numerical solution x_n will be stable if eq. (8.37) is stable.
A characteristic equation associated with eq. (8.37) can be defined as

$$\gamma_{-1}z^k + \gamma_0 z^{k-1} + \gamma_1 z^{k-2} + \cdots + \gamma_{k-1} = 0. \tag{8.38}$$

This is a polynomial with roots z_n (poles). This is stable if the poles satisfy $|z_n| < 1$, as illustrated in fig. 8.11

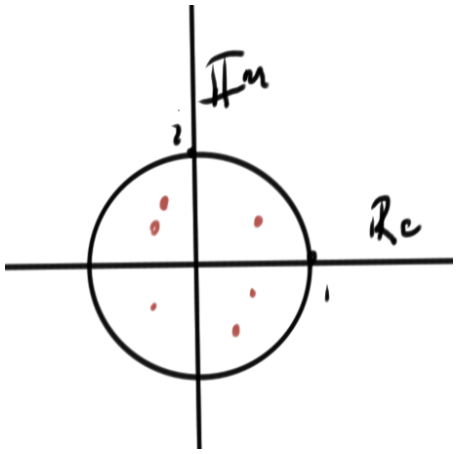


Figure 8.11: Stability.

Observe that the $\gamma's$ are dependent on Δt.
Some of the details associated with this switch to discrete time systems have been assumed. A refresher, by example, of those ideas can be found in appendix D.

Example 8.2: Forward Euler stability.

For $k = 1$ step.

$$x_{n+1} - x_n = \Delta t f(x_n, t_n), \tag{8.39}$$

the coefficients are $\alpha_{-1} = 1, \alpha_0 = -1, \beta_{-1} = 0, \beta_0 = 1$. For the simple function above

$$\gamma_{-1} = \alpha_{-1} - \Delta t \lambda \beta_{-1} = 1 \tag{8.40a}$$

$$\gamma_0 = \alpha_0 - \Delta t \lambda \beta_0 = -1 - \Delta t \lambda. \tag{8.40b}$$

The stability polynomial is

$$1z + (-1 - \Delta t \lambda) = 0, \tag{8.41}$$

or

$$\boxed{z = 1 + \delta t \lambda.} \tag{8.42}$$

This is the root, or pole.
For stability we must have

$$|1 + \Delta t \lambda| < 1, \tag{8.43}$$

or

$$\left| \lambda - \left(-\frac{1}{\Delta t} \right) \right| < \frac{1}{\Delta t}, \tag{8.44}$$

This inequality is illustrated roughly in fig. 8.12.

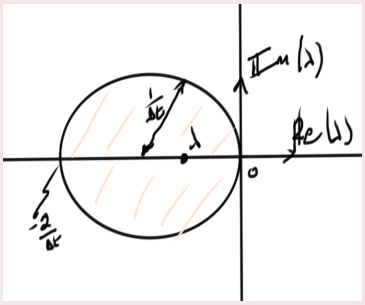


Figure 8.12: Stability region of FE.

All poles of my system must be inside the stability region in order to get stable γ.

8.11 STABILITY (CONTINUED)

Continuing with the simple continuous time test system

$$\dot{x}(t) = \lambda x(t) \tag{8.45}$$

With the application of a trial solution $x(t) = e^{st}$, the resulting characteristic equation is

$$s = \lambda, \tag{8.46}$$

and stability follows, provided that $\mathrm{Re}(\lambda) < 0$ as sketched in fig. 8.13.

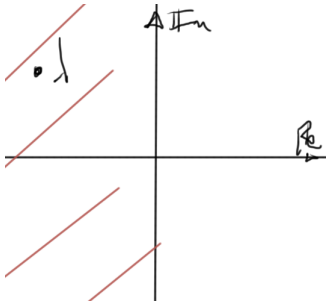


Figure 8.13: Stability region.

Utilizing LMS methods, for example TR, the discrete time system results, such as

$$\gamma_{-1}x_{n+1} + \gamma_0 x_n + \gamma_1 x_{n-1} + \cdots = 0. \tag{8.47}$$

With a substitution $x_{n+j} = z^{k-1+j}$, a characteristic polynomial results

$$\gamma_{-1}z^k + \gamma_0 z^{k-1} + \gamma_1 z^{k-2} + \cdots = 0. \tag{8.48}$$

This is stable provided $|z_l| < 1$, as illustrated in fig. 8.14.

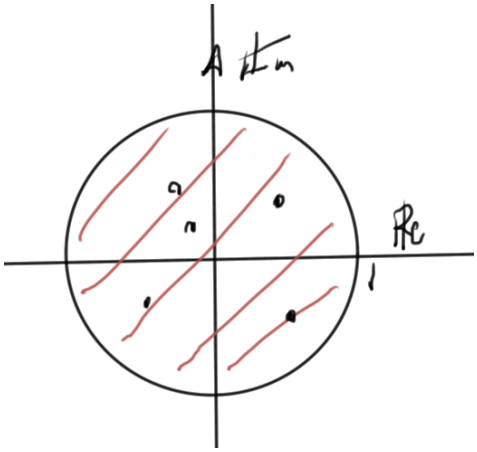


Figure 8.14: Stability region in z-domain.

... SWITCHED TO SLIDES.

Definition 8.3: Stiff.

A stiff system is one that has multiple timescales, as characterized by a significant range of eigenvalues.

Definition 8.4: Lossless system.

Lossless poles are those that reside strictly on the imaginary axis.

Definition 8.5: Lossly system.

Lossly poles are those that reside strictly to the left of the imaginary axis.

Definition 8.6: Active system.

Active poles are those that reside strictly to the right of the imaginary axis.

Definition 8.7: A-stable.

See slides.

Theorem 8.1: Dahlquist's theorem.

There are no LMS methods of order greater than 2 that are A-stable. Also known as the Dahlquist barrier. The TR method has the lowest error of the A-stable LMS methods.

Some recent developments The backward differentiation formulas (BDF) are of order $>$ 2. They are not A-stable, but can be sufficient with stability sketched in fig. 8.15.

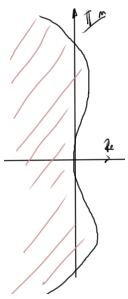


Figure 8.15: BDF stability region.

Also available are the Obreshkov formulas [1], which are both A-stable and of order > 2. There is a cost to both of these methods: computation of the derivatives at each step.

8.12 PROBLEMS

Exercise 8.1 Modeling inductors and capacitors and time dependence.

In this problem you will first extend the circuit simulator that you developed in the previous problem sets to include capacitors and inductors . Then, you will use it to simulate a clock-distribution network.

a. Modify the circuit simulator you developed for the previous assignments to handle capacitors and inductors. The program should read a file with the list of: resistors, currents sources, voltage sources, capacitors, and inductors. The syntax for specifying a capacitor is:

Clabel node1 node2 val

where label is an arbitrary label, node1 and node2 are integer circuit node numbers, and val is the capacitance (a floating point number). The syntax for specifying an inductor is:

Llabel node1 node2 val

where label is an arbitrary label, node1 and node2 are integer circuit node numbers, and val is the inductance (a floating point number). Explain how you handle inductors, and which stamp can be proposed to include them into the modified nodal analysis equations written in the form

$$\mathbf{G}\mathbf{x}(t) + \mathbf{C}\dot{\mathbf{x}}(t) = \mathbf{B}\mathbf{u}(t) \tag{8.49}$$

where the column vector $\mathbf{u}(t)$ contains all sources.

b. As test system, we consider the network in fig. 8.16 which distributes the clock signal to 8 blocks of an integrated circuit. The network is in the form of a binary tree with four levels. Each segment is a transmission line with characteristics (length, per-unit-length parameters) given in table 8.1. Divide each transmission line into segments of length $\Delta z = 0.05$mm, and model each segment with an RLC circuit as the one shown in the figure.

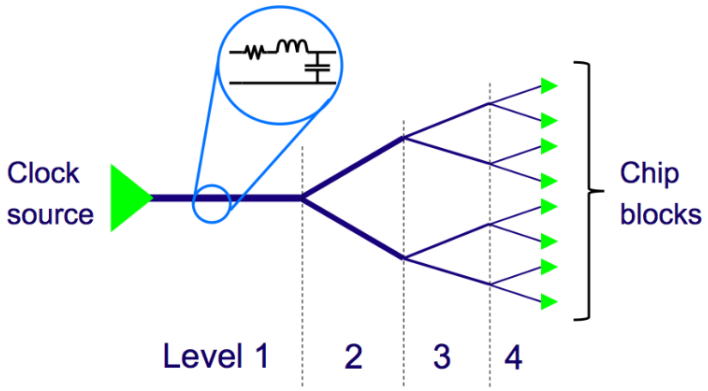


Figure 8.16: Network for the distribution of the clock to 8 IC blocks.

Table 8.1: Characteristics of the clock tree network. Note that the resistance, inductance and capacitance values are per-unit-length (p.u.l.)

Level	Length $[mm]$	Resistance p.u.l $R[\Omega/cm]$	Inductance p.u.l. $L[nH/cm]$	Capacitance p.u.l. $C[pF/cm]$
1	6	25	5.00	2.00
2	4	35.7	7.14	1.40
3	3	51.0	10.2	0.98
4	2	51.0	10.2	0.98

Model the clock source with a periodic voltage source with the following characteristics: amplitude 1 V, rise/fall time 100 ps, period 2 ns, duty cycle 50%, initial delay 100 ps. The clock source voltage is depicted in fig. 8.17.

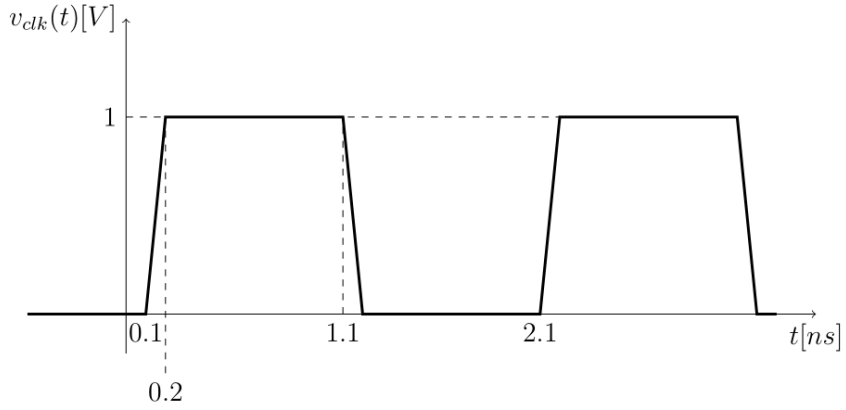


Figure 8.17: Clock signal.

Model each chip block with a 5 k Ω resistor in parallel with a 5 f F capacitor. Write a Matlab function that generates a spice-compatible netlist of the clock distribution network. Report in a table the values of the resistors, inductors and capacitors that you used in each section.

c. Simulate the circuit with backward Euler (BE). Use a constant time-step. Plot the voltage generated by the clock source and the voltage received by one of the blocks for $t \in [0, 5]$ ns. Suggestion: since the clock starts a few timesteps after $t = 0$, you can assume zero initial conditions for your simulation.

d. Explain how did you choose the timestep to be used in the simulation.

e. Implement and compare two methods for solving differential equations: backward Euler (BE), and trapezoidal rule (TR). Simulate the circuit with the two methods for different timestep values, ranging from a coarse timestep to a fine time step. Report in a table the error and the CPU time for each method. Since we don't have an exact solution for this system, use as reference solution the output voltage computed with TR with a very fine time step. In the table, report as error the maximum absolute error between the computed output voltage and the reference one.

f. Plot the error versus the time step on a log-log scale for the two methods, and comment the obtained results.

Answer for Exercise 8.1

PROBLEM SET RELATED MATERIAL REDACTED IN THIS DOCUMENT.PLEASE FEEL FREE TO EMAIL ME FOR THE FULL VERSION IF YOU AREN'T TAKING ECE1254.[redacted]

[redacted]. END-REDACTION

9 MODEL ORDER REDUCTION

Many examples on slides.

Definition 9.1: Admittance.

Given an impedance Z, the admittance Y is defined as

$$Y = \frac{1}{Z}.$$

9.1 MODEL ORDER REDUCTION

The system equations of interest for model order reduction (MOR) are

$$\mathbf{G}\mathbf{x}(t) + \mathbf{C}\dot{\mathbf{x}}(t) = \mathbf{B}\mathbf{u}(t) \tag{9.1a}$$

$$\mathbf{y}(t) = \mathbf{L}^{\mathsf{T}}\mathbf{x}(t). \tag{9.1b}$$

Here

- $\mathbf{u}(t)$ is a vector of inputs.
- $\mathbf{y}(t)$ is a vector of interesting outputs, as filtered by the matrix $\mathbf{L}$, and
- $\mathbf{x}(t)$ is the state vector, of size N.

The aim is illustrated in fig. 9.1.

Review: transfer function

$$\mathbf{G}\mathbf{X}(s) + \mathbf{C}\left(s\mathbf{X}(s) - \cancel{\mathbf{x}(0)}\right) = \mathbf{B}\mathbf{U}(s) \tag{9.2a}$$

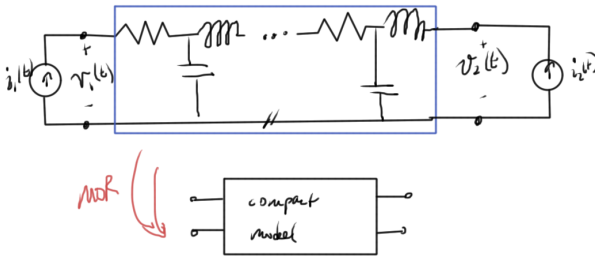


Figure 9.1: Find a simplified model that has the same input-output characteristics.

$$\mathbf{Y}(s) = \mathbf{L}^{\mathrm{T}}\mathbf{X}(s). \tag{9.2b}$$

Grouping terms

$$(\mathbf{G} + s\mathbf{C})\,\mathbf{X}(s) = \mathbf{B}\mathbf{U}(s), \tag{9.3}$$

and inverting allows for solution for the state vector output in the frequency domain

$$\mathbf{X}(s) = (\mathbf{G} + s\mathbf{C})^{-1}\,\mathbf{B}\mathbf{U}(s) \tag{9.4}$$

The vector of outputs of interest are

$$\mathbf{Y}(s) = L^{\mathrm{T}}\,(\mathbf{G} + s\mathbf{C})^{-1}\,\mathbf{B}\mathbf{U}(s) \tag{9.5}$$

The boxed portion is the transfer function

$$\mathbf{H}(s) = L^{\mathrm{T}}\,(\mathbf{G} + s\mathbf{C})^{-1}\,\mathbf{B}. \tag{9.6}$$

... Switched to slides.

Geometrical interpretation to the change of variables

$$
\begin{aligned}
x(t) &= \underbrace{\mathbf{V}}_{N \times N}\mathbf{w}(t) \\
&= \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_N \end{bmatrix} \begin{bmatrix} w_1(t) \\ w_2(t) \\ \vdots \\ w_N(t) \end{bmatrix} \\
&= \mathbf{v}_1 w_1(t) + \mathbf{v}_2 w_2(t) + \cdots \mathbf{v}_N w_N(t).
\end{aligned} \tag{9.7}
$$

The key to MOR is to find a way to select the important portions of these $\mathbf{v}_i, w_i(t)$ vectors-function pairs. This projects the system equations onto an approximation that retains the most interesting characteristics.

9.2 MOMENT MATCHING

Referring back to eq. (9.1) and it's Laplace transform representation, an input-to-state transfer function $\mathbf{F}(s)$ is defined by

$$
\mathbf{X}(s) = (\mathbf{G} + s\mathbf{C})^{-1}\,\mathbf{B}\mathbf{U}(s) = \mathbf{F}(s)\mathbf{U}(s) \tag{9.8}
$$

This can be expanded around the DC value ($s = 0$)

$$
\mathbf{F}(s) = M_0 + M_1 s + M_2 s^2 + \cdots M_{q-1} s^{q-1} + \underbrace{M_q}_{\text{moments}} s^q + \cdots \tag{9.9}
$$

A reduced model of order q is defined as

$$
\mathbf{F}(s) = M_0 + M_1 s + M_2 s^2 + \cdots M_{q-1} s^{q-1} + \tilde{M}_q s^q. \tag{9.10}
$$

9.3 MODEL ORDER REDUCTION (CONT).

An approximation of the following system is sought

$$
\mathbf{G}\mathbf{x}(t) + C\dot{\mathbf{x}}(t) = B\mathbf{u}(t) \tag{9.11a}
$$

$$\mathbf{y}(t) = \mathbf{L}^{\mathrm{T}}\mathbf{x}(t). \tag{9.11b}$$

The strategy is to attempt to find a $N \times q$ projector $\mathbf{V}$ of the form

$$\mathbf{V} = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_q \end{bmatrix} \tag{9.12}$$

so that the solution of the constrained q-variable state vector $\mathbf{x}_q$ is sought after letting

$$\mathbf{x}(t) = \mathbf{V}\mathbf{x}_q(t). \tag{9.13}$$

9.4 MOMENT MATCHING

$$\mathbf{F}(s) = (\mathbf{G} + s\mathbf{C})^{-1}\,\mathbf{B} = \mathbf{M}_0 + \mathbf{M}_1 s + \mathbf{M}_2 s^2 + \cdots + \mathbf{M}_{q-1}s^{q-1} + M_q s^q + \cdots \tag{9.14}$$

The reduced model is created such that

$$\mathbf{F}_q(s) = \mathbf{M}_0 + \mathbf{M}_1 s + \mathbf{M}_2 s^2 + \cdots + \mathbf{M}_{q-1}s^{q-1} + \tilde{\mathbf{M}}_q s^q. \tag{9.15}$$

Form an $N \times q$ projection matrix

$$\mathbf{V}_q \equiv \begin{bmatrix} \mathbf{M}_0 & \mathbf{M}_1 & \cdots & \mathbf{M}_{q-1} \end{bmatrix} \tag{9.16}$$

With the substitution of eq. (9.13) the system equations in the time domain, illustrated graphically in fig. 9.2, becomes

This is a system of N equations, in q unknowns. A set of moments from the frequency domain have been used to project the time domain system. This relies on the following unproved theorem (references to come)

thm:multiphysicsL19:280

Theorem 9.1: Reduced model moments.

If $\mathrm{span}\{\mathbf{v}_q\} = \mathrm{span}\{\mathbf{M}_0, \mathbf{M}_1, \cdots, \mathbf{M}_{q-1}\}$, then the reduced model will match the first q moments.

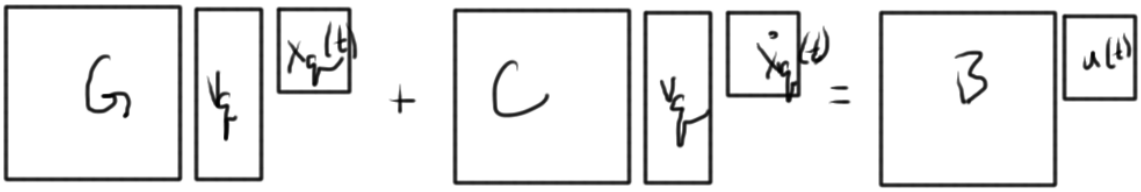


Figure 9.2: Projected system.

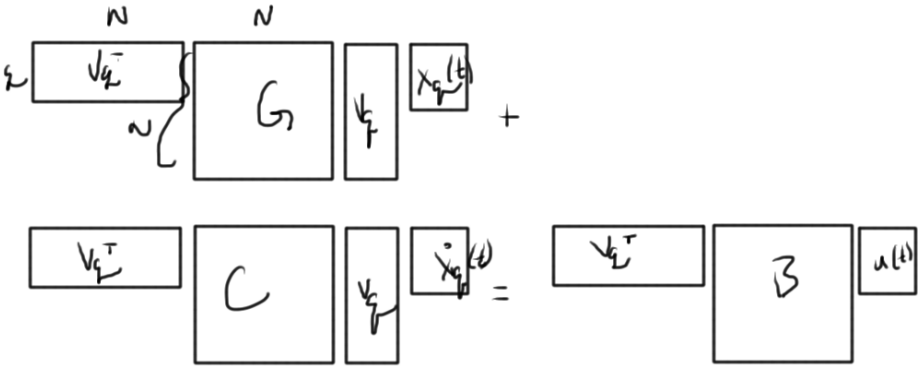


Figure 9.3: Reduced projected system.

Left multiplication by $\mathbf{V}_q^T$ yields fig. 9.3.
This is now a system of q equations in q unknowns.
With

$$\mathbf{G}_q = \mathbf{V}_q^T \mathbf{G} \mathbf{V}_q \tag{9.17a}$$

$$\mathbf{C}_q = \mathbf{V}_q^T \mathbf{C} \mathbf{V}_q \tag{9.17b}$$

$$\mathbf{B}_q = \mathbf{V}_q^T \mathbf{B} \tag{9.17c}$$

$$\mathbf{L}_q^T = \mathbf{L}^T \mathbf{V}_q \tag{9.17d}$$

the system is reduced to

$$\mathbf{G}_q \mathbf{x}_q(t) + \mathbf{C}_q \dot{\mathbf{x}}_q(t) = \mathbf{B}_q \mathbf{u}(t). \tag{9.18a}$$

$$\mathbf{y}(t) \approx \mathbf{L}_q^T \mathbf{x}_q(t) \tag{9.18b}$$

Moments calculation Using

$$\left(\mathbf{G} + s\mathbf{C}\right)^{-1} \mathbf{B} = \mathbf{M}_0 + \mathbf{M}_1 s + \mathbf{M}_2 s^2 + \cdots \tag{9.19}$$

thus

$$\begin{aligned} \mathbf{B} &= \left(\mathbf{G} + s\mathbf{C}\right) \mathbf{M}_0 + \left(\mathbf{G} + s\mathbf{C}\right) \mathbf{M}_1 s + \left(\mathbf{G} + s\mathbf{C}\right) \mathbf{M}_2 s^2 + \cdots \\ &= \mathbf{G}\mathbf{M}_0 + s\left(C\mathbf{M}_0 + \mathbf{G}\mathbf{M}_1\right) + s^2\left(C\mathbf{M}_1 + \mathbf{G}\mathbf{M}_2\right) + \cdots \end{aligned} \tag{9.20}$$

Since $\mathbf{B}$ is a zeroth order matrix, setting all the coefficients of s equal to zero provides a method to solve for the moments

$$\begin{aligned} \mathbf{B} &= \mathbf{G}\mathbf{M}_0 \\ -\mathbf{C}\mathbf{M}_0 &= \mathbf{G}\mathbf{M}_1 \\ -\mathbf{C}\mathbf{M}_1 &= \mathbf{G}\mathbf{M}_2 \end{aligned} \tag{9.21}$$

The moment $\mathbf{M}_0$ can be found with LU of $\mathbf{G}$, plus the forward and backward substitutions. Proceeding recursively, using the already computed LU factorization, each subsequent moment calculation requires only one pair of forward and backward substitutions.

Numerically, each moment has the exact value

$$\mathbf{M}_q = \left(-\mathbf{G}^{-1}\mathbf{C}\right)^q \mathbf{M}_0. \tag{9.22}$$

As $q \to \infty$, this goes to some limit, say $\mathbf{w}$. The value $\mathbf{w}$ is related to the largest eigenvalue of $-\mathbf{G}^{-1}\mathbf{C}$. Incidentally, this can be used to find the largest eigenvalue of $-\mathbf{G}^{-1}\mathbf{C}$.

The largest eigenvalue of this matrix will dominate these factors, and can cause some numerical trouble. For this reason it is desirable to avoid such explicit moment determination, instead using implicit methods.

The key is to utilize theorem 9.1, and look instead for an alternate basis $\{\mathbf{v}_q\}$ that also spans $\{\mathbf{M}_0, \mathbf{M}_1, \cdots, \mathbf{M}_q\}$.

Space generated by the moments Write

$$\mathbf{M}_q = \mathbf{A}^q \mathbf{R}, \tag{9.23}$$

where in this case

$$\begin{aligned} \mathbf{A} &= -\mathbf{G}^{-1}\mathbf{C} \\ \mathbf{R} &= \mathbf{M}_0 = -\mathbf{G}\mathbf{B} \end{aligned} \tag{9.24}$$

The span of interest is

$$\text{span}\{\mathbf{R}, \mathbf{A}\mathbf{R}, \mathbf{A}^2\mathbf{R}, \cdots, \mathbf{A}^{q-1}\mathbf{R}\}. \tag{9.25}$$

Such a sequence is called a Krylov subspace. One method to compute such a basis, the Arnoldi process, relies on Gram-Schmidt orthonormalization methods:

Algorithm 9.1: Arnoldi process.

$\mathbf{V}_0 = \mathbf{R}/\|\mathbf{R}\|$
for all $i \in [0, q-2]$ **do**
  $\mathbf{V}_{i+1} = \mathbf{A}\mathbf{V}_i$
  for all $j \in [0, i]$ **do**

$$\mathbf{V}_{i+1} = \mathbf{V}_{i+1} - \left(\mathbf{V}_{i+1}^{\mathrm{T}}\mathbf{V}_j\right)\mathbf{V}_j$$
end for
$$\mathbf{V}_{i+1} = \mathbf{V}_{i+1}/\|\mathbf{V}_{i+1}\|$$
end for

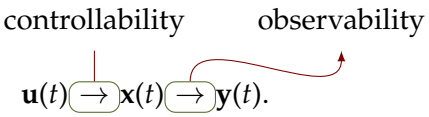
Some numerical examples and plots on the class slides.

9.5 TRUNCATED BALANCED REALIZATION (1000 FT OVERVIEW)

Consider a model in state space form

$$\begin{aligned}\dot{\mathbf{x}} &= \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t)\\ \mathbf{y} &= \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t).\end{aligned} \tag{9.26}$$

The system has the form



(9.27)

Definition 9.2: Observability.

The system is observable if $\forall t$, the initial state $\mathbf{x}(t_0)$ can be determined knowing $\mathbf{u}(t), \mathbf{y}(t), \quad t \in [t_0, t_1]$.

Equivalent condition (if A stable) If A stable, the observability Gramian $\mathbf{W}_0 > 0$

$$\mathbf{W}_0 = \int_0^\infty e^{\mathbf{A}^{\mathrm{T}}\tau}\mathbf{C}^{\mathrm{T}}\mathbf{C}e^{\mathbf{A}\tau}d\tau \tag{9.28}$$

This Gramian expression results from a calculation of the "energy" of the homogeneous system (no input $\mathbf{u}(t)$), for which the system is in an initial state $\mathbf{x}_0$ at $t = 0$, with output

$$\mathbf{y}(t) = \mathbf{C}e^{\mathbf{A}t}\mathbf{x}_0 \tag{9.29}$$

The output energy is

$$\begin{aligned}\int_0^\infty \mathbf{y}^\mathrm{T}\mathbf{y}dt &= \int_0^\infty \mathbf{x}_0^\mathrm{T} e^{\mathbf{A}^\mathrm{T}t}\mathbf{C}^\mathrm{T}\mathbf{C}e^{\mathbf{A}t}\mathbf{x}_0 dt \\ &= \mathbf{x}_0^\mathrm{T}\mathbf{W}_0\mathbf{x}_0 \\ &= \mathcal{E}_0\end{aligned} \tag{9.30}$$

Consider the eigenvalues and vectors of $\mathbf{W}_0$

$$\mathbf{W}_0\mathbf{x}_{0,i} = \lambda_{0,i}\mathbf{x}_{0,i}. \tag{9.31}$$

If $\mathbf{x}_0 = \mathbf{x}_{0,i}$ this implies

$$\mathcal{E}_0 = \mathbf{x}_{0,i}^\mathrm{T}\lambda_{0,i}\mathbf{x}_{0,i} \tag{9.32}$$

The eigenvalues of the observability Gramian can provide some useful information about what states are observable.

Definition 9.3: Controllability.

If for any initial state $\mathbf{x}(t_0)$, there is a final time $t_1 > t_0$, and a final state $\mathbf{x}_1$ such that there exists an input $\mathbf{u}^*(t)$ such that $\mathbf{x}(t_1) = \mathbf{x}_1$.

Theorem 9.2: Controllability Gramian.

The system is controllable if and only if the controllability Gramian $\mathbf{W}_c > 0$.

What is the energy of the input $\mathbf{u}^*(t)$ that is needed to set up $\mathbf{x}_1$? On the slides it is shown that this is

$$\mathcal{E}_1 = \mathbf{x}_1^\mathrm{T}\mathbf{W}_c^{-1}\mathbf{x}_1. \tag{9.33}$$

if $\mathbf{x}_1$ is an eigenvector of $\mathbf{W}_c$ with eigenvalue λ_1, then this reduces to

$$\begin{aligned}\mathcal{E}_1 &= \frac{1}{\lambda_1}\mathbf{x}_1^\mathrm{T}\mathbf{x}_1 \\ &= \frac{1}{\lambda_1}.\end{aligned} \tag{9.34}$$

These eigenvalues provide information about how much energy is required to achieve any given state.

The purpose of considering observability and controllability is to help determine what modes can be safely discarded to reduce the model order. A state that cannot be achieved without excessive energy cost is not likely to be of interest.

While the explicit form of the controllability Gramian has not been given, it becomes of less interest since it is possible to make a change of variables, such that the controllability and observability Gramians become equal and diagonal.

$$\begin{aligned}\widetilde{\mathbf{W}_c} &= T\mathbf{W}_c T^{\mathrm{T}} \\ \widetilde{\mathbf{W}_o} &= T^{-\mathrm{T}}\mathbf{W}_o T^{-1}\end{aligned} \tag{9.35}$$

Using these provides a balanced method to determine the most important modes to retain.

9.6 PROBLEMS

Exercise 9.1 Model order reduction methods.

In this problem you will apply different model order reduction methods to the heat conducting bar. Use Backward Euler for time integration.

a. In this problem we shall introduce a heat source $u(t)$ and its spatial distribution in the form of $h(x)$.

$$\frac{\partial T(x,t)}{\partial t} = \frac{\partial^2 T(x,t)}{\partial x^2} - \alpha T(x,t) + h(x)u(t) \qquad x \in [0,1] \tag{9.36}$$

where $T(x,t)$ is the temperature at location x at time t. The boundary condition is that no heat flows away from the two ends of the bar, i.e.

$$\left.\frac{\partial T}{\partial x}\right|_{x=0} = \left.\frac{\partial T}{\partial x}\right|_{x=1} = 0. \tag{9.37}$$

The term with α models the heat dissipation from the bar to the surrounding environment. Let $\alpha = 0.01$. You can assume zero initial conditions. Explain how this heat equation can be formulated into an equivalent dynamical system in the form

$$\mathbf{G}\mathbf{x}(t) + \mathbf{C}\dot{\mathbf{x}}(t) = \mathbf{B}\mathbf{u}(t) \tag{9.38}$$

Explain the choice of your matrices $\mathbf{G}, \mathbf{C}, \mathbf{B}$. What do the states $\mathbf{x}(t)$ of your dynamical system correspond to, physically? We define the output $\mathbf{y}(t)$ as $\mathbf{y}(t) = \mathbf{L}^{\mathrm{T}}\mathbf{x}(t)$, where $^{\mathrm{T}}$ denotes the matrix transpose.

b. We are interested in the following scenarios:

Case 1 Input: heat flow at the left end of the bar
Output: temperature at the right end

Case 2 Input: heat flow at the left end of the bar
Output: average temperature along the bar

Case 3 Input: uniform heating
Output: temperature at the right end

Case 4 Input: uniform heating
Output: average temperature along the bar

Explain how you will pick the matrices $\mathbf{B}, \mathbf{L}$ in each case. **In all remaining questions, consider only case 1.**

c. Write a Matlab routine `PlotFreqResp(`ω`,G,C,B,L)` which takes in ω, G, C, B, L as input and plots the system frequency response. Here ω is a vector of frequencies in rad/s. For $N = 500$ (500 nodes) plot the real and imaginary part of the frequency response for case 1.

Hints: To plot the frequency response, use the command `semilogx(x,y)`. You can generate the frequency vector with the command: ω `= logspace(-8,4,500)`.

d. Reduce the dynamical system for $N = 500$ using the modal truncation method. Retain the modes that are associated with the "slowest" eigenvalues. Note that you will have to convert your system from the modified nodal analysis representation to the state-space representation. Use $q = 1, 2, 4, 10, 50$ (q = number of states in the reduced system). Plot the frequency response of the original system and frequency response of the reduced system in the same plot. Select a "reasonably small" order q for the reduced model, that ensures a reasonable approximation of the original system. Clearly explain how did you pick that q, which factors did you look at.

e. Using your time domain solver, compute the output of the original system ($N = 500$) and of the reduced model (with the order q you selected) for each of the following two inputs:

1. Take the first input $u(t) = u_1(t)$ as

$$u_1(t) = \begin{cases} 1 & \text{if } t \geq 0 \\ 0 & \text{if } t < 0 \end{cases} \tag{9.39}$$

   For $u_1(t)$ pick t_{stop} such that the system reaches steady state.

2. Take the second input $u(t) = u_2(t)$ as

$$u_2(t) = \sin(0.01t) \quad \text{for } t_{\text{stop}} = 10000 \tag{9.40}$$

   Be sure to pick an appropriate time step and explain your choice. How much speed-up did you get for your reduced models for time domain simulations? Explain your results.

f. Repeat part d using the PRIMA model order reduction algorithm (see provided routine **prima.m**). In order to maximize the efficiency of the reduced model, after you generate it with `prima()`, bring it to state-space form (by multiplying by $\mathbf{C}^{-1}$). Then, make the $\mathbf{A}$ matrix of the state space model diagonal. In this way, the reduced model becomes sparse rather than full, and can be solved more quickly.

g. Repeat part e using the PRIMA model order reduction algorithm.

h. Compare the results obtained with the two methods. A table with a few columns will suffice.

Answer for Exercise 9.1

PROBLEM SET RELATED MATERIAL REDACTED IN THIS DOCUMENT.PLEASE FEEL FREE TO EMAIL ME FOR THE FULL VERSION IF YOU AREN'T TAKING ECE1254.

[redacted]. END-REDACTION

10

HARMONIC BALANCE

PROBLEM SET RELATED MATERIAL REDACTED IN THIS DOCUMENT.PLEASE FEEL FREE TO EMAIL ME FOR THE FULL VERSION IF YOU AREN'T TAKING ECE1254.

[redacted]. END-REDACTION

Part II

APPENDICES

A SINGULAR VALUE DECOMPOSITION

In definition 4.1 singular value decomposition (SVD) was presented without any mention of how to compute it.

Here I'll review some of the key ideas from the MIT OCW SVD lecture by Prof. Gilbert Strang [12]. This is largely to avoid having to rewatch this again in a few years as I just did.

The idea behind the SVD is to find an orthogonal basis that relates the image space of the transformation, as well as the basis for the vectors that the transformation applies to. That is a relation of the form

$$\mathbf{u}_i = M\mathbf{v}_j \tag{A.1}$$

Where $\mathbf{v}_j$ are orthonormal basis vectors, and $\mathbf{u}_i$ are orthonormal basis vectors for the image space.

Suppose that such a set of vectors can be computed and that M has a representation of the desired form

$$M = U\Sigma V^* \tag{A.2}$$

where

$$U = \begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_m \end{bmatrix}, \tag{A.3}$$

and

$$V = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_n \end{bmatrix}. \tag{A.4}$$

By left or right multiplication of M with its (conjugate) transpose, the decomposed form of these products can be observed to have a very simple form

$$M^*M = V\Sigma^*U^*U\Sigma V^* = V\Sigma^*\Sigma V^*, \tag{A.5a}$$

$$MM^* = U\Sigma V^*V\Sigma^*U^* = U\Sigma\Sigma^*U^*. \tag{A.5b}$$

Because Σ is diagonal, the products $\Sigma^*\Sigma$ and $\Sigma\Sigma^*$ are also both diagonal, and populated with the absolute squares of the singular values that have been presumed to

exist. Because $\Sigma\Sigma^*$ is an $m \times m$ matrix, whereas $\Sigma^*\Sigma$ is an $n \times n$ matrix, so the numbers of zeros in each of these will differ, but each will have the structure

$$\Sigma^*\Sigma \sim \Sigma\Sigma^* \sim \begin{bmatrix} |\sigma_1|^2 & & & & & & \\ & |\sigma_2|^2 & & & & & \\ & & \ddots & & & & \\ & & & |\sigma_r|^2 & & & \\ & & & & 0 & & \\ & & & & & \ddots & \\ & & & & & & 0 \end{bmatrix} \tag{A.6}$$

This shows one method of computing the singular value decomposition (for full rank systems). Solution of the eigensystem of either MM^* or M^*M is required to find both the singular values, and one of U or V. U can be found from the r orthonormal eigenvectors of MM^* supplemented with a mutually orthonormal set of vectors from the Null space of M. Σ can be found by taking the square roots of the eigenvalues of MM^*. With both Σ and U computed, V can be found by inversion. Alternatively, it is possible to solve for Σ and V by computing the eigensystem of M^*M, also supplementing those eigenvectors with vectors from the null space, and then compute U by inversion.

Working the examples from the lecture is instructive to gain some insight about how this works.

Example A.1: Full rank 2×2 matrix.

In the lecture the SVD decomposition is computed for

$$M = \begin{bmatrix} 4 & 4 \\ 3 & -3 \end{bmatrix} \tag{A.7}$$

The matrix product with its conjugate is

$$MM^* = \begin{bmatrix} 32 & 0 \\ 0 & 18 \end{bmatrix} \tag{A.8a}$$

$$M^*M = \begin{bmatrix} 25 & 7 \\ 7 & 25 \end{bmatrix} \tag{A.8b}$$

The first is already diagonalized so $U = I$, and the singular values are found by inspection $\{\sqrt{32}, \sqrt{18}\}$, or

$$\Sigma = \begin{bmatrix} \sqrt{32} & 0 \\ 0 & \sqrt{18} \end{bmatrix} \tag{A.9}$$

Because the system is full rank, V follows by inversion

$$\Sigma^{-1}U^*M = V^*, \tag{A.10}$$

or

$$V = M^*U\left(\Sigma^{-1}\right)^*. \tag{A.11}$$

In this case that is

$$V = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{A.12}$$

V could also be computed directly by diagonalizing M^*M. The eigenvectors are $(1, \pm 1)/\sqrt{2}$, with respective eigenvalues $\{32, 18\}$.

This gives eq. (A.9) and eq. (A.12). Again, because the system is full rank, U can be computed by inversion

$$U = MV\Sigma^{-1}. \tag{A.13}$$

Carrying out this calculation recovers $U = I$ as expected. Looks like I used a different matrix than Prof. Strang used in his lecture (alternate signs on the 3's). He had some trouble that arrived from independent calculation of the respective eigenspaces. Calculating one from the other avoids that trouble since there

are different signed eigenvalues that can be chosen. Because specific mappings between the eigenspaces that satisfy the $\mathbf{u}_i = M\mathbf{v}_j$ constraints encoded by the relationship $M = U\Sigma V^*$ are desired, these U and V matrices cannot be selected independently.

A non-full rank example, as in the lecture, is also useful

Example A.2: A 2×2 matrix without full rank.

How about

$$M = \begin{bmatrix} 1 & 1 \\ 2 & 2 \end{bmatrix}. \tag{A.14}$$

The matrix and conjugate product is

$$M^*M = \begin{bmatrix} 5 & 5 \\ 5 & 5 \end{bmatrix} \tag{A.15a}$$

$$MM^* = \begin{bmatrix} 2 & 4 \\ 4 & 8 \end{bmatrix} \tag{A.15b}$$

For which the non-zero eigenvalue is 10 and the corresponding eigenvalue is

$$\mathbf{v} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}. \tag{A.16}$$

This gives

$$\Sigma = \begin{bmatrix} \sqrt{10} & 0 \\ 0 & 0 \end{bmatrix} \tag{A.17}$$

Since V must be orthonormal, there is only one choice (up to a sign) for the vector from the null space.

Try

$$V = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{A.18}$$

MM^* has eigenvalues $\{10, 0\}$ as expected. The eigenvector for the non-zero eigenvalue is found to be

$$\mathbf{u} = \frac{1}{\sqrt{5}} \begin{bmatrix} 1 \\ 2 \end{bmatrix}. \tag{A.19}$$

It is easy to expand this to an orthonormal basis. Is it required to pick a specific sign relative to the choice made for V?

Try

$$U = \frac{1}{\sqrt{5}} \begin{bmatrix} 1 & 2 \\ 2 & -1 \end{bmatrix}. \tag{A.20}$$

Multiplying out $U\Sigma V^*$ gives

$$\begin{aligned} \frac{1}{\sqrt{5}} \begin{bmatrix} 1 & 2 \\ 2 & -1 \end{bmatrix} \begin{bmatrix} \sqrt{10} & 0 \\ 0 & 0 \end{bmatrix} \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} &= \begin{bmatrix} 1 & 2 \\ 2 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 \\ 2 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \\ &= \begin{bmatrix} 1 & 1 \\ 2 & 2 \end{bmatrix}. \end{aligned} \tag{A.21}$$

It appears that this works. This has not demonstrated why that should be, and could be due to luck with signs. To fully understand how to do this type of computation, more theoretical work would be required. That is probably a good reason to leave such a task to computational software.

B

BASIC THEOREMS AND DEFINITIONS

Definition B.1: Positive (negative) definite.

A matrix M is positive (negative) definite, denoted $M > 0 (< 0)$ if $\mathbf{y}^T M \mathbf{y} > 0 (< 0)$, $\quad \forall \mathbf{y}$.

If a matrix is neither positive, nor negative definite, it is called indefinite.

When zero equality is possible $\mathbf{y}^T M \mathbf{y} \geq 0 (\leq 0)$, the matrix is positive (negative) semi-definite.

Theorem B.1: Positive (negative) definite.

A symmetric matrix $M > 0 (< 0)$ iff $\lambda_i > 0 (< 0)$ for all eigenvalues λ_i, or is indefinite iff its eigenvalues λ_i are of mixed sign.

Theorem B.2: Mean value theorem.

For a continuous and differentiable function $f(x)$, the difference can be expressed in terms of the derivative at an intermediate point

$$f(x_2) - f(x_1) = \left.\frac{\partial f}{\partial x}\right|_{\tilde{x}} (x_2 - x_1)$$

where $\tilde{x} \in [x_1, x_2]$.

This is illustrated (roughly) in fig. B.1.

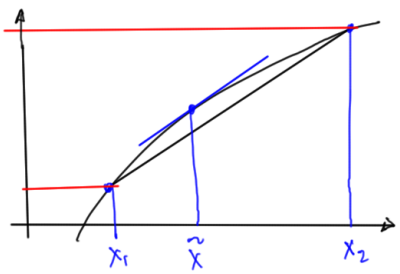


Figure B.1: Mean value theorem illustrated.

C

NORTON EQUIVALENTS

Exercise 6.1 contains a circuit with constant voltage source that makes the associated MNA matrix non-symmetric. Part a looks like it is there to provide a hint that this source V_s and its internal resistance R_s can likely be replaced by a constant current source.

Here two voltage source configurations will be compared to a current source configuration, with the assumption that equivalent circuit configurations can be found.

First voltage source configuration First consider the source and internal series resistance configuration sketched in fig. C.1, with a purely resistive load.

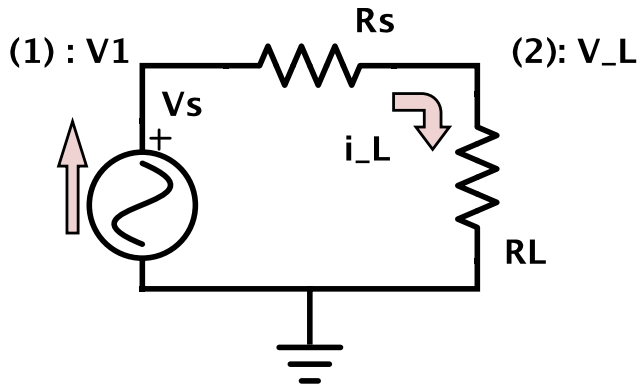


Figure C.1: First voltage source configuration.

The nodal equations for this system are

1. $-i_L + (V_1 - V_L)Z_s = 0$
2. $V_L Z_L + (V_L - V_1)Z_s = 0$
3. $V_1 = V_s$

In matrix form these are

$$\begin{bmatrix} Z_s & -Z_s & -1 \\ -Z_s & Z_s + Z_L & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} V_1 \\ V_L \\ i_L \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ V_s \end{bmatrix} \tag{C.1}$$

This has solution

$$V_L = V_s \frac{R_L}{R_L + R_s} \tag{C.2a}$$

$$i_L = \frac{V_s}{R_L + R_s} \tag{C.2b}$$

$$V_1 = V_s. \tag{C.2c}$$

Second voltage source configuration Now consider the same voltage source, but with the series resistance location flipped as sketched in fig. C.2.

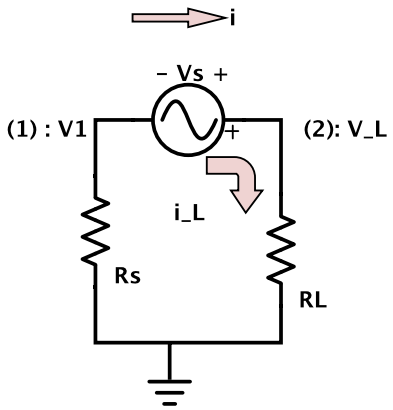


Figure C.2: Second voltage source configuration.

The nodal equations are

1. $V_1 Z_s + i_L = 0$

2. $-i_L + V_L Z_L = 0$
3. $V_L - V_1 = V_s$

These have matrix form

$$\begin{bmatrix} Z_s & 0 & 1 \\ 0 & Z_L & -1 \\ -1 & 1 & 0 \end{bmatrix} \begin{bmatrix} V_1 \\ V_L \\ i_L \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ V_s \end{bmatrix} \tag{C.3}$$

This configuration has solution

$$V_L = V_s \frac{R_L}{R_L + R_s} \tag{C.4a}$$

$$i_L = \frac{V_s}{R_L + R_s} \tag{C.4b}$$

$$V_1 = -V_s \frac{R_s}{R_L + R_s} \tag{C.4c}$$

Observe that the voltage at the load node and the current through this impedance is the same in both circuit configurations. The internal node voltage is different in each case, but that has no measurable effect on the external load.

Current configuration Now consider a current source and internal parallel resistance as sketched in fig. C.3.

There is only one nodal equation for this circuit

1. $-I_s + V_L Z_s + V_L Z_L = 0$

The load node voltage and current follows immediately

$$V_L = \frac{I_s}{Z_L + Z_s} \tag{C.5a}$$

$$i_L = V_L Z_L = \frac{Z_L I_s}{Z_L + Z_s} \tag{C.5b}$$

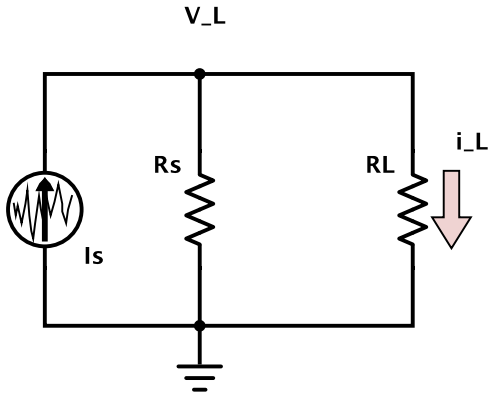


Figure C.3: Current source configuration.

The goal is to find a value for I_L so that the voltage and currents at the load node match either of the first two voltage source configurations. It has been assumed that the desired parallel source resistance is the same as the series resistance in the voltage configurations. That was just a guess, but it ends up working out.

From eq. (C.5a) and eq. (C.2a) that equivalent current source can be found from

$$V_L = V_s \frac{R_L}{R_L + R_s} = \frac{I_s}{Z_L + Z_s}, \tag{C.6}$$

or

$$\begin{aligned} I_s &= V_s \frac{R_L(Z_L + Z_s)}{R_L + R_s} \\ &= \frac{V_s}{R_S} \frac{R_s R_L(Z_L + Z_s)}{R_L + R_s} \end{aligned} \tag{C.7}$$

$$\boxed{I_s = \frac{V_s}{R_S}.} \tag{C.8}$$

The load is expected to be the same through the load, and is

$$i_L = V_L Z_L == V_s \frac{R_L Z_L}{R_L + R_s} = \frac{V_s}{R_L + R_s}, \tag{C.9}$$

which matches eq. (C.2b).

Remarks The equivalence of the series voltage source configurations with the parallel current source configuration has been demonstrated with a resistive load. This is a special case of the more general Norton's theorem, as detailed in [14] and [3] §5.1. Neither of those references prove the theorem. Norton's theorem allows the equivalent current and resistance to be calculated without actually solving the system. Using that method, the parallel resistance equivalent follows by summing all the resistances in the source circuit with all the voltage sources shorted. Shorting the voltage sources in this source circuit results in the same configuration. It was seen directly in the two voltage source configurations that it did not matter, from the point of view of the external load, which sequence the internal series resistance and the voltage source were placed in did not matter. That becomes obvious with knowledge of Norton's theorem, since shorting the voltage sources leaves just the single resistor in both cases.

D

STABILITY OF DISCRETIZED LINEAR DIFFERENTIAL EQUATIONS

To motivate the methods used to discuss stability of linear multistep methods, it is helpful to take a step back and review stability concepts for LDE systems.

By way of example, consider a second order LDE homogeneous system defined by

$$\frac{d^2x}{dt^2} + 3\frac{dx}{dt} + 2 = 0. \tag{D.1}$$

Such a system can be solved by assuming an exponential solution, say

$$x(t) = e^{st}. \tag{D.2}$$

Substitution gives

$$e^{st}\left(s^2 + 3s + 2\right) = 0, \tag{D.3}$$

The polynomial part of this equation, the characteristic equation has roots $s = -2, -1$.

The general solution of eq. (D.1) is formed by a superposition of solutions for each value of s

$$x(t) = ae^{-2t} + be^{-t}. \tag{D.4}$$

Independent of any selection of the superposition constants a, b, this function will not blow up as $t \to \infty$.

This "stability" is due to the fact that both of the characteristic equation roots lie in the left hand Argand plane.

Now consider a discretized form of this LDE. This will have the form

$$\begin{aligned} 0 &= \frac{1}{(\Delta t)^2}\left(x_{n+2} - 2x_{n-1} + x_n\right) + \frac{3}{\Delta t}\left(x_{n+1} - x_n\right) + 2x_n \\ &= x_{n+2}\left(\frac{1}{(\Delta t)^2}\right) + x_{n+1}\left(\frac{3}{\Delta t} - \frac{2}{(\Delta t)^2}\right) + x_n\left(\frac{1}{(\Delta t)^2} - \frac{3}{\Delta t} + 2\right), \end{aligned} \tag{D.5}$$

or

$$0 = x_{n+2} + x_{n+1}\left(3\Delta t - 2\right) + x_n\left(1 - 3\Delta t + 2\left(\Delta t\right)^2\right). \tag{D.6}$$

Note that after discretization, each subsequent index corresponds to a time shift. Also observe that the coefficients of this discretized equation are dependent on the discretization interval size Δt. If the specifics of those coefficients are ignored, a general form with the following structure can be observed

$$0 = x_{n+2}\gamma_0 + x_{n+1}\gamma_1 + x_n\gamma_2. \tag{D.7}$$

It turns out that, much like the LDE solution by characteristic polynomial, it is possible to attack this problem by assuming a solution of the form

$$x_n = Cz^n. \tag{D.8}$$

A time shift index change $x_n \to x_{n+1}$ results in a power adjustment in this assumed solution. This substitution applied to eq. (D.7) yields

$$0 = Cz^n\left(z^2\gamma_0 + z\gamma_1 + 1\gamma_2\right), \tag{D.9}$$

Suppose that this polynomial has roots $z \in \{z_1, z_2\}$. A superposition, such as

$$x_n = az_1^n + bz_2^n, \tag{D.10}$$

will also be a solution since insertion of this into the RHS of eq. (D.7) yields

$$az_1^n\left(z_1^2\gamma_0 + z_1\gamma_1 + \gamma_2\right) + bz_2^n\left(z_2^2\gamma_0 + z_2\gamma_1 + \gamma_2\right) = az_1^n \times 0 + bz_2^n \times 0. \tag{D.11}$$

The zero equality follows since z_1, z_2 are both roots of the characteristic equation for this discretized LDE. In the discrete z domain stability requires that the roots satisfy the bound $|z| < 1$, a different stability criteria than in the continuous domain. In fact, there is no a-priori guarantee that stability in the continuous domain will imply stability in the discretized domain.

Let's plot those z-domain roots for this example LDE, using $\Delta t \in \{1/2, 1, 2\}$. The respective characteristic polynomials are

$$0 = z^2 - \frac{1}{2}z = z\left(z - \frac{1}{2}\right) \tag{D.12a}$$

$$0 = z^2 + z = z\,(z+1) \tag{D.12b}$$

$$0 = z^2 + 4z + 3 = (z+3)(z+1). \tag{D.12c}$$

These have respective roots

$$z = 0, \frac{1}{2} \tag{D.13a}$$

$$z = 0, -1 \tag{D.13b}$$

$$z = -1, -3 \tag{D.13c}$$

Only the first discretization of these three yields stable solutions in the z domain, although it appears that $\Delta t = 1$ is right on the boundary.

E

LAPLACE TRANSFORM REFRESHER

Laplace transforms were used to solve the MNA equations for time dependent systems, and to find the moments used to in MOR.

For the record, the Laplace transform is defined as:

$$\boxed{\mathcal{L}(f(t)) = \int_0^\infty e^{-st} f(t) dt.} \tag{E.1}$$

The only Laplace transform pair used in the lectures is that of the first derivative

$$\begin{aligned} \mathcal{L}(f'(t)) &= \int_0^\infty e^{-st} \frac{df(t)}{dt} dt \\ &= e^{-st} f(t)\Big|_0^\infty - (-s) \int_0^\infty e^{-st} f(t) dt \\ &= -f(0) + s\mathcal{L}(f(t)). \end{aligned} \tag{E.2}$$

Here it is loosely assumed that the real part of s is positive, and that $f(t)$ is "well defined" enough that $e^{-s\infty} f(\infty) \to 0$.

Where used in the lectures, the Laplace transforms were of vectors such as the matrix vector product $\mathcal{L}(\mathbf{G}\mathbf{x}(t))$. Because such a product is linear, observe that it can be expressed as the original matrix times a vector of Laplace transforms

$$\begin{aligned} \mathcal{L}(\mathbf{G}\mathbf{x}(t)) &= \mathcal{L}\Big[G_{ik} x_k(t)\Big]_i \\ &= \Big[G_{ik} \mathcal{L} x_k(t)\Big]_i \\ &= \mathbf{G}\Big[\mathcal{L} x_i(t)\Big]_i. \end{aligned} \tag{E.3}$$

The following notation was used in the lectures for such a vector of Laplace transforms

$$\mathbf{X}(s) = \mathcal{L}\mathbf{x}(t) = \Big[\mathcal{L} x_i(t)\Big]_i. \tag{E.4}$$

F

DISCRETE FOURIER TRANSFORM

Transform pair In [4] a verification of the discrete Fourier transform pairs was performed. A much different looking discrete Fourier transform pair is given in [2] §A.4. This transform pair samples the points at what are called the Nykvist time instants given by

$$t_k = \frac{Tk}{2N+1}, \qquad k \in [-N, \cdots N] \tag{F.1}$$

Note that the endpoints of these sampling points are not $\pm T/2$, but are instead at

$$\pm \frac{T}{2} \frac{1}{1+1/N}, \tag{F.2}$$

which are slightly within the interior of the $[-T/2, T/2]$ range of interest. The reason for this slightly odd seeming selection of sampling times becomes clear if one calculate the inversion relations.

Given a periodic ($\omega_0 T = 2\pi$) bandwidth limited signal evaluated only at the Nykvist times t_k,

$$\boxed{x(t_k) = \sum_{n=-N}^{N} X_n e^{jn\omega_0 t_k},} \tag{F.3}$$

assume that an inversion relation can be found. To find X_n evaluate the sum

$$\begin{aligned} \sum_{k=-N}^{N} x(t_k) e^{-jm\omega_0 t_k} &= \sum_{k=-N}^{N} \left(\sum_{n=-N}^{N} X_n e^{jn\omega_0 t_k} \right) e^{-jm\omega_0 t_k} \\ &= \sum_{n=-N}^{N} X_n \sum_{k=-N}^{N} e^{j(n-m)\omega_0 t_k} \end{aligned} \tag{F.4}$$

This interior sum has the value $2N+1$ when $n = m$. For $n \neq m$, and $a = e^{j(n-m)\frac{2\pi}{2N+1}}$, this is

$$\begin{aligned}\sum_{k=-N}^{N} e^{j(n-m)\omega_0 t_k} &= \sum_{k=-N}^{N} e^{j(n-m)\omega_0 \frac{Tk}{2N+1}} \\ &= \sum_{k=-N}^{N} a^k \\ &= a^{-N} \sum_{k=-N}^{N} a^{k+N} \\ &= a^{-N} \sum_{r=0}^{2N} a^r \\ &= a^{-N} \frac{a^{2N+1}-1}{a-1}.\end{aligned} \tag{F.5}$$

Since $a^{2N+1} = e^{2\pi j(n-m)} = 1$, this sum is zero when $n \neq m$. This means that

$$\sum_{k=-N}^{N} e^{j(n-m)\omega_0 t_k} = (2N+1)\delta_{n,m}, \tag{F.6}$$

which provides the desired Fourier inversion relation

$$\boxed{X_m = \frac{1}{2N+1} \sum_{k=-N}^{N} x(t_k) e^{-jm\omega_0 t_k}.} \tag{F.7}$$

Matrix form The discrete time Fourier transform has been seen to have the form

$$x_k = \sum_{n=-N}^{N} X_n e^{2\pi jnk/(2N+1)} \tag{F.8a}$$

$$X_n = \frac{1}{2N+1} \sum_{k=-N}^{N} x_k e^{-2\pi jnk/(2N+1)}. \tag{F.8b}$$

A matrix representation of this form is desired. Let

$$\mathbf{x} = \begin{bmatrix} x_{-N} \\ \vdots \\ x_0 \\ \vdots \\ x_N \end{bmatrix} \tag{F.9a}$$

$$\mathbf{X} = \begin{bmatrix} X_{-N} \\ \vdots \\ X_0 \\ \vdots \\ X_N \end{bmatrix} \tag{F.9b}$$

Equation (F.8a) written out in full is

$$\begin{aligned} x_k = {} & X_{-N} e^{-2\pi j N k/(2N+1)} \\ & + X_{1-N} e^{-2\pi j (N-1) k/(2N+1)} \\ & + \cdots \\ & + X_0 \\ & + \cdots \\ & + X_{N-1} e^{2\pi j (N-1) k/(2N+1)} \\ & + X_N e^{2\pi j N k/(2N+1)} \end{aligned} \tag{F.10}$$

Following the FFT discussion in [9], let $W = e^{2\pi j/(2N+1)}$. The matrix relation is

$$\mathbf{x} = \begin{bmatrix} W^{NN} & W^{(N-1)N} & \cdots & 1 & \cdots & W^{-(N-1)N} & W^{-NN} \\ W^{N(N-1)} & W^{(N-1)(N-1)} & \cdots & 1 & \cdots & W^{-(N-1)(N-1)} & W^{-N(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ W^{-N(N-1)} & W^{-(N-1)(N-1)} & \cdots & 1 & \cdots & W^{N-1(N-1)} & W^{N(N-1)} \\ W^{-NN} & W^{(1-N)N} & \cdots & 1 & \cdots & W^{(N-1)N} & W^{NN} \end{bmatrix} \mathbf{X} \tag{F.11}$$

Similarly, from eq. (F.8b), the inverse relation expands out to

$$\begin{aligned}(2N+1)X_n = {} & x_{-N}e^{2\pi jnN/(2N+1)} \\ & + x_{1-N}e^{2\pi jn(N-1)/(2N+1)} \\ & \cdots \\ & + x_0 \\ & \cdots \\ & + x_{N-1}e^{-2\pi jn(N-1)/(2N+1)} \\ & + x_N e^{-2\pi jnN/(2N+1)},\end{aligned} \tag{F.12}$$

with a matrix form of

$$(2N+1)\mathbf{X} = \begin{bmatrix} W^{-NN} & W^{-N(N-1)} & \dots & 1 & \dots & W^{N(N-1)} & W^{NN} \\ W^{-(N-1)N} & W^{-(N-1)(N-1)} & \dots & 1 & \dots & W^{(N-1)(N-1)} & W^{(N-1)N} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ W^{(N-1)N} & W^{(N-1)(N-1)} & \dots & 1 & \dots & W^{-(N-1)(N-1)} & W^{-(N-1)N} \\ W^{NN} & W^{N(N-1)} & \dots & 1 & \dots & W^{-N(N-1)} & W^{-NN} \end{bmatrix} \tag{F.13}$$

Letting

$$\mathbf{F} = \begin{bmatrix} W^{NN} & W^{(N-1)N} & \dots & 1 & \dots & W^{-(N-1)N} & W^{-NN} \\ W^{N(N-1)} & W^{(N-1)(N-1)} & \dots & 1 & \dots & W^{-(N-1)(N-1)} & W^{-N(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ W^{-N(N-1)} & W^{-(N-1)(N-1)} & \dots & 1 & \dots & W^{N-1(N-1)} & W^{N(N-1)} \\ W^{-NN} & W^{(1-N)N} & \dots & 1 & \dots & W^{(N-1)N} & W^{NN} \end{bmatrix}, \tag{F.14}$$

the discrete transform pair has the following compactly matrix representation

$$\mathbf{x} = \mathbf{F}\mathbf{X} \tag{F.15a}$$

$$\mathbf{X} = \frac{1}{2N+1}\bar{\mathbf{F}}\mathbf{x}, \tag{F.15b}$$

where $\bar{\mathbf{F}}$ is the complex conjugate of $\mathbf{F}$.

G

HARMONIC BALANCE, ROUGH NOTES

PROBLEM SET RELATED MATERIAL REDACTED IN THIS DOCUMENT.PLEASE FEEL FREE TO EMAIL ME FOR THE FULL VERSION IF YOU AREN'T TAKING ECE1254.

[redacted]. END-REDACTION

H

MATLAB NOTEBOOKS

PROBLEM SET RELATED MATERIAL REDACTED IN THIS DOCUMENT.PLEASE FEEL FREE TO EMAIL ME FOR THE FULL VERSION IF YOU AREN'T TAKING ECE1254.

[redacted]. END-REDACTION

I

MATHEMATICA NOTEBOOKS

These Mathematica notebooks, some just trivial ones used to generate figures, others more elaborate, and perhaps some even polished, can be found in
https://github.com/peeterjoot/mathematica/tree/master/.

The free Wolfram CDF player, is capable of read-only viewing these notebooks to some extent.

- Nov 3, 2014 ece1254/plotDiode.nb

  Plot of somewhat ill behaved sum of linear and exponential function

Part III

INDEX

INDEX

Part IV

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] John Butcher. *Order and stability for single and multivalue methods for differential equations*, 2014. URL https://www.math.auckland.ac.nz/~butcher/CONFERENCES/SouthAfrica/WITS/wits-slides.pdf. [Online; accessed 19-Nov-2014].

[2] Franco Giannini and Giorgio Leuzzi. *Nonlinear Microwave Circuit Design*. Wiley Online Library, 2004.

[3] J.D. Irwin. *Basic Engineering Circuit Analysis*. MacMillian, 1993.

[4] Peeter Joot. *Condensed matter physics.*, chapter Discrete Fourier transform. peeterjoot.com, 2013. URL http://peeterjoot.com/archives/math2013/phy487.pdf. [Online; accessed 14-February-2019].

[5] Peeter Joot. *Simple C++ double representation explorer*, 2014. URL https://github.com/peeterjoot/physicsplay/blob/master/notes/ece1254/samples/rounding.cpp. [Online; accessed 29-Sept-2014].

[6] Chris Maes. *Singular Value Decomposition, from the Wolfram Demonstrations Project*, 2014. URL http://demonstrations.wolfram.com/SingularValueDecomposition/. [Online; accessed 2-October-2014].

[7] Harry M Markowitz. The elimination form of the inverse and its application to linear programming. *Management Science*, 3(3):255–269, 1957.

[8] Farid N Najm. *Circuit simulation*. John Wiley & Sons, 2010.

[9] William H Press, Saul A Teukolsky, William T Vetterling, and Brian P Flannery. *Numerical recipes in C*. Cambridge University Press, 1996.

[10] Alfio Quarteroni, Riccardo Sacco, and Fausto Saleri. *Numerical mathematics*, volume 37. Springer, 2007.

[11] Jonathan Richard Shewchuk. *An introduction to the conjugate gradient method without the agonizing pain*, 1994. URL http://www.cs.cmu.edu/~quake-papers/painless-conjugate-gradient.pdf. [Online; accessed 30-Oct-2014].

[12] Gilbert Strang. *Singular Value Decomposition*, 2014. URL http://ocw.mit.edu/courses/mathematics/18-06-linear-algebra-spring-2010/video-lectures/lecture-29-singular-value-decomposition/. [Online; accessed 30-Sept-2014].

[13] Timothy Vismor. *Pivoting To Preserve Sparsity*, 2012. URL https://vismor.com/documents/network_analysis/matrix_algorithms/S8.SS3.php. [Online; accessed 15-Oct-2014].

[14] Wikipedia. Norton's theorem — wikipedia, the free encyclopedia, 2014. URL http://en.wikipedia.org/w/index.php?title=Norton%27s_theorem&oldid=629143825. [Online; accessed 1-November-2014].

[15] Wikipedia. Singular value decomposition — wikipedia, the free encyclopedia, 2014. URL http://en.wikipedia.org/w/index.php?title=Singular_value_decomposition&oldid=626808859. [Online; accessed 2-October-2014].