Fundamental Math and Physics for Scientists and Engineers

David Yevick • Hannah Yevick

WILEY

FUNDAMENTAL MATH AND PHYSICS FOR SCIENTISTS AND ENGINEERS

FUNDAMENTAL MATH AND PHYSICS FOR SCIENTISTS AND ENGINEERS

DAVID YEVICK

HANNAH YEVICK

WILEY

Published by John Wiley & Sons, Inc., Hoboken, New Jersey
Published simultaneously in Canada

Library of Congress Cataloging-in-Publication Data:

Yevick, David, author.
Fundamental math and physics for scientists and engineers / David Yevick, Hannah Yevick.
1 online resource
"Published simultaneously in Canada"–Title page verso.
Includes index.
Description based on print version record and CIP data provided by publisher; resource not viewed.
ISBN 978-0-470-40784-4 (pbk.) – ISBN 978-1-118-98559-5 (Adobe PDF) – ISBN 978-0-470-40784-4 (pbk.)
1. Mathematics. 2. Physics. I. Yevick, Hannah, author. II. Title.
QA39.3
510–dc23

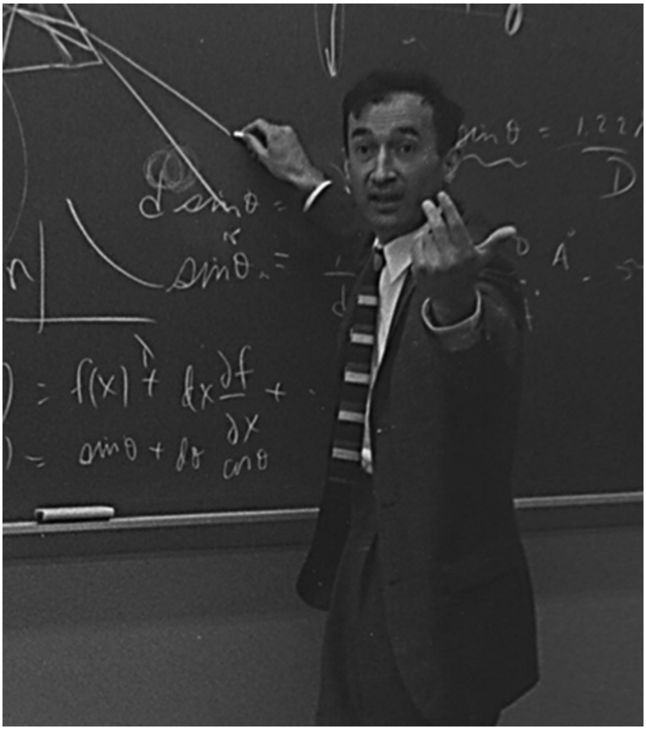
2014040697

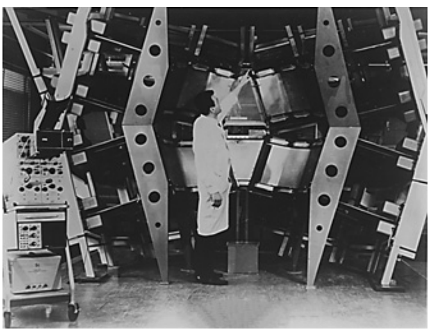
Cover Image: iStockphoto©nadla

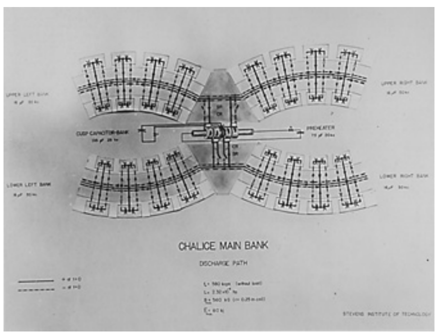
Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

To George






UPPER LEFT BANK
UPPER RIGHT BANK
LOWER LEFT BANK
LOWER RIGHT BANK
CUSP CAPACITOR BANK
PREHEATER
CHALICE MAIN BANK
DISCHARGE PATH
STEVENS INSTITUTE OF TECHNOLOGY

CONTENTS

1

INTRODUCTION

Unique among disciplines, physics condenses the limitlessly complex behavior of nature into a small set of underlying principles. Once these are clearly understood and supplemented with often superficial domain knowledge, any scientific or engineering problem can be succinctly analyzed and solved. Accordingly, the study of physics leads to unsurpassed satisfaction and fulfillment.

This book summarizes intermediate-, college-, and university-level physics and its associated mathematics, identifying basic formulas and concepts that should be understood and memorized. It can be employed to supplement courses, as a reference text or as review material for the GRE and graduate comprehensive exams.

Since physics incorporates broad areas of science and engineering, many treatments overemphasize technical details and problems that require time-consuming mathematical manipulations. The reader then often loses sight of fundamental issues, leading to gaps in comprehension that widen as more advanced material is introduced. This book accordingly focuses exclusively on core material relevant to practical problem solving. Fine details of the subject can later be assimilated rapidly, effectively placing leaves on the branches formed by the underlying concepts.

Mathematics and physics constitute the language of science. Hence, as with any spoken language, they must be learned through repetition and memorization. The central results and equations indicated in this book are therefore indicated by shaded text. These should be rederived, transcribed into a notebook or review cards with a summary of their derivation and memorized. Problems from any source should be solved in conjunction with this book; however, undertaking time-consuming problems

Fundamental Math and Physics for Scientists and Engineers, First Edition.
David Yevick and Hannah Yevick.

without recourse to worked solutions that indicate optimal calculational procedures is not recommended.

Finally, we wish to thank our many inspiring teachers, whose numerous insights guided our approach, in particular Paul Bamberg, Alan Blair, and Sam Treiman, and, above all, our father and grandfather, George Yevick, whose boundless love of physics inspired generations of students.

2

PROBLEM SOLVING

Problem solving, especially on examinations, should habitually follow the procedures below.

2.1 ANALYSIS

1. Problems are very often misread or answered incompletely. Accordingly, circle the words in the problem that describe the required results and underline the specified input data. After completing the calculation, insure that the quantities evaluated in fact correspond to those circled.
2. Write down a summary of the problem in your own words as concisely as possible.
3. Draw a diagram of the physical situation that suggests the general properties of the solution. Annotate the diagram as the solution progresses. Always draw diagrams that accentuate the difference between variables, e.g., when drawing triangles, be sure that its angles are markedly unequal.
4. Briefly contrast different solution methods and summarize on the examination paper the simplest of these (especially if partial credit is given).
5. Solve the problem, proceeding in small steps. Do not perform two mathematical manipulations in a single line. Align equal signs on subsequent lines and check

Fundamental Math and Physics for Scientists and Engineers, First Edition.
David Yevick and Hannah Yevick.

each line of the calculation against the previous line immediately after writing it down. Being careful and organized inevitably saves time.

6. Reconsider periodically if you are employing the simplest solution method. If mathematics becomes involved, backtrack and search for an error or a different approach.
7. Verify the dimensions of your answer and that its magnitude is physically reasonable.
8. Insert your answer into the initial equations that define the problem and check that it yields the correct solution.
9. If necessary and time permits, solve the problem a second time with a different method.

2.2 TEST-TAKING TECHNIQUES

Strategies for improving examination performance include:

1. For morning examinations, 1–3 weeks before the examination, start the day two or more hours before the examination time.
2. Devise a plan of studying well before the examination that includes several review cycles.
3. Outline on paper and review cards in your own words the required material. Carry the cards with you and read them throughout the day when unoccupied.
4. To become aware of optimal solution procedures, solve a variety of problems in books that provide worked solutions and rederive the examples in this or another textbook. Limit the time spent on each problem in accordance with the importance of the topic.
5. Obtain or design your own practice exams and take these under simulated test conditions.
6. In the day preceding a major examination, at most, briefly review notes—studies have demonstrated that last-minute studying does not on average improve grades.
7. Be aware of the examination rules in advance. On multiple choice exams, determining how many answers must be eliminated before selecting one of the remaining choices is statistically beneficial.
8. If allowed, take high-energy food to the exam.
9. Arrive early at the examination location to familiarize yourself with the test environment.
10. First, read the entire examination and then solve the problems in order of difficulty.

11. Maintain awareness of the problem objective; sometimes, a solution can be worked backward from this knowledge.
12. If a calculation proves more complex than expected, either outline your solution method or address a different problem and return to the calculation later, possibly with a different perspective.
13. For multiple choice questions, insure that the solutions are placed correctly on the answer sheet. Write the number of the problem and the answer on a piece of paper and transfer this information onto the answer sheet only at the end of the exam. Retain the paper in case of grading error.
14. On multiple choice tests, examine the possible choices before solving the problem. Eliminate choices with incorrect dimensions and those that lack physical meaning. Those remaining often indicate the important features of the solution and possibly may even reveal the correct answer.
15. Maintain an even composure, possibly through short stretching or controlled breathing exercises.

2.2.1 Dimensional Analysis

Results can be partially verified through dimensional analysis. Dimensions such as those of force, $[MD/T^2]$, are here distinguished by square brackets, where, e.g., D indicates length, T time, M mass, and Q charge. Quantities that are added, subtracted, or equated must possess identical dimensions. For example, $a = v/t$ is potentially valid since the right-hand side dimension of this expression is the product $[D/T][1/T]$, which agrees with that of the left-hand side. Similarly, the argument of a transcendental function (a function that can be expressed as an infinite power series), such as an exponential or harmonic function or of polynomials such as $f(x) = x + x^2$, must be dimensionless; otherwise, different powers would possess different dimensions and could therefore not be summed.

While the dimensions of important physical quantities should be memorized, the dimensions of any quantity can be deduced from an equation expressing this quantity in terms of variables with known dimensions. Thus, e.g., $F = ma$ implies that $[F] = [M][D/T^2] = [MD/T^2]$. Quantities with involved dimensions are often expressed in terms of other standard variables such as voltage.

Example

From $Q = CV$, the units of capacitance can be expressed as $[Q/V]$, with V representing volts. Subsequently, from $V = IR$ with $I = dQ/dt$, the dimensions of, e.g., $t = 1/RC$ can be verified.

3

SCIENTIFIC PROGRAMMING

This text contains basic physics programs written in the Octave scientific programming language that is freely available from http://www.gnu.org/software/octave/index.html with documentation at www.octave.org. Default selections can be chosen during setup. Octave incorporates many features of the commercial MATLAB® language and facilitates rapid and compact coding (for a more extensive introduction, refer to A Short Course in Computational Science and Engineering: C++, Java and Octave Numerical Programming with Free Software Tools, by David Yevick Copyright © 2012 David Yevick). Some of the material in the following text is reprinted with permission from Cambridge University Press.

3.1 LANGUAGE FUNDAMENTALS

A few important general programming concepts as applied to Octave are first summarized below:

1. A program consists primarily of statements that result from terminating a valid expression not followed by the continuation character … (three lower dots), a carriage return, or a semicolon.
2. An expression can be formed from one or more subexpressions linked by operators such as + or *.

Fundamental Math and Physics for Scientists and Engineers, First Edition.
David Yevick and Hannah Yevick.

3. Operators possess different levels of precedence, e.g., in 2/4 + 3, the division operation possesses a higher precedence and is therefore evaluated before addition. In expressions involving two or more operators with the same precedence level, such as division and multiplication, the operations are typically evaluated from left to right, e.g., 2/4 * 3 equals (2/4) * 3.
4. The parenthesis operator, which evaluates the expression that it encloses, is assigned to the highest precedence level. This eliminates errors generated by incorrect use of precedence or associativity.
5. Certain style conventions, while not required, enhance clarity and readability:
   a. Variables and function names should be composed of one or more descriptive words. The initial letter should be uncapitalized, while the first letter of each subsequent word should be capitalized as in outputVelocity.
   b. Spaces should be placed to the right and left of binary operators, which act on the expressions (operands) to their left and right, as in `3 + 4`, but no space should be employed in unary operator such as the negative sign in `-3 + 4`. Spaces are preferentially be inserted after commas as in `computeVelocity( 3, 4 )` and within parentheses except where these indicate indices.
   c. Indentation should be employed to indicate when a group of inner statements is under the logical control of an outer statement such as in

```
if ( firstVariable == 0 )
   secondVariable = 5;
end
```

   d. Any part of a line located to the right of the symbol % constitutes a comment that typically documents the program. Statements that form a logical unit should be preceded by one or more comment lines and surrounded by blank lines. Statement lines that introduce input variables should end with a comment describing the variables.

3.1.1 Octave Programming

Running Octave: Starting Octave opens a command window into which statements can be entered interactively. Alternatively, a program in the directory `programs` in partition `C:` is created by first entering `cd C:\programs` into the command window, pressing the enter key, and then entering the command `edit`. Statements are then typed into the program editor, the file is saved by selecting Save from the button or menu bar as a MATrix LABoratory file such as `myFile.m` (the .m extension is appended automatically by the editor), and the program is then run by typing `myFile` into the command window. The program can also be activated by including the statement `myFile;` within another program. To list the files in the current directory, enter `dir` into the Octave command window.

Help Commands: Typing `help commandName` yields a description of the command `commandName`. To find all commands related to a word `subject`, type `lookfor subject`. Entering `doc` or `doc topic` brings up, respectively, a complete help document and a description of the language feature `topic`.

Input and Output: A value of a variable `G` can be entered into a program (`.m` file) from the keyboard by including the line `G = input( 'user prompt' )`. The statement `format long e` sets the output style to display all 15 floating-point number significant digits, after which `format short e` reverts to the default 5 output digits.

Constants and Complex Numbers: Some important constants are `i` and `j`, which both equal $\sqrt{-1}$, `e`, and pi. However, if a variable assignment such as `i = 3`; is encountered in an Octave program, `i` ceases to be identified with the imaginary unit until the command `clear i` is issued. Imaginary numbers can be manipulated with the functions `real( )`, `imag( )`, `conj( )`, and `norm( )`, and imaginary values are automatically returned by standard functions such as `exp( )`, `sin( )`, and `sinh( )` for imaginary arguments.

Arrays and Matrices: A symbol `A` can represent a scalar, row, or column vector or matrix of any dimension. Row vectors are constructed either by

```
vR = [ 1 2 3 4 ];
```

or

```
vR = [ 1, 2, 3, 4 ];
```

The corresponding column vector can similarly be entered in any of the following three ways:

```
vC = [ 1
2
3
4 ];

vC = [ 1; 2; 3; 4 ];
vC = [ 1 2 3 4 ].';
```

Here `.'` indicates transpose, while `'` instead implements the Hermitian (complex conjugate) transpose.

A 2×2 matrix

$$\mathtt{mRC} = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

can be constructed by, e.g., `mRC = [ 1 2; 3 4 ];` after which `mRC(1, 2)` returns $(\mathrm{MRC})_{12}$, here the value 2. Subsequently, `size(mRC)` yields a vector containing the row and column dimensions of `mRC`, while `length( mRC )` returns the maximum of these values. Here, we introduce the convention of appending `R`, `C`, or `RC` to the variable name to respectively identify row vectors, column vectors, and matrices.

Basic Manipulations: A value `n` is raised to the power `m` by `n^m`. The remainder of n/m is denoted `rem( n, m )` and is positive or zero for $n > 0$ and negative or zero for $n < 0$. The function `mod( n, m )` returns `n` modulus `m`, which is always positive, while `ceil( )`, `floor( )`, and `fix( )` round floating-point numbers to the next larger integer, smaller integer, and nearest integer closer to zero, respectively.

Vector and Matrix Operations: Two vectors or matrices of the same dimension can be added or subtracted. Multiplying a matrix or vector by a scalar, c, multiplies each element by c. Additionally, `eye( n, n )` is the $n \times n$ unit or identity matrix with ones along the main diagonal and zeros elsewhere, while `ones( n, m )` and `zeros( n, m )` are $n \times m$ matrices with all elements one or zeros so that

$$\mathtt{2 + mRC = 2*ones(2,2) + mRC} = \begin{pmatrix} 3 & 4 \\ 5 & 6 \end{pmatrix}$$

and

$$\mathtt{2*eye(2,2) + mRC} = \begin{pmatrix} 3 & 2 \\ 3 & 6 \end{pmatrix}$$

Further, `mRC * mRC`, or equivalently `mRC^2`, multiplies `mRC` by itself, while

$$\mathtt{mRC.*mRC = mRC.\hat{}2} = \begin{pmatrix} 1 & 4 \\ 9 & 16 \end{pmatrix}$$

implements component-by-component multiplication. Other arithmetic operations function analogously so that the (i, j) element of `M ./ N` is $\mathtt{M}_{ij}/\mathtt{N}_{ij}$. Functions such as `cos( M )` return a matrix composed of the cosines of each element in `M`.

Solving Linear Equation Systems: The solution of the linear equation system `xR * mRC = yR` is `xR = yR / mRC`, while `mRC * xC = yC` is solved by `xC = mRC \ yC`. The inverse of a matrix `mRC` is represented by `inv( mRC )`. The eigenvalues of a matrix are obtained through `eigenValues = eig( mRC )`, while both the eigenvalues and eigenvectors are returned through `[ eigenValues, eigenVectors ] = eig( mRC )`.

Random Number Generation: A single random number between 0 and 1 is generated by `rand`, while `rand( m, n )` returns a $m \times n$ matrix with random entries. The same random sequence can be generated each time a program is run by including `rand( 'state', 0 )` before the first call to `rand`.

Control Logic and Iteration: The *logical operators* in octave are ==, <, <=, >, >=, ~= (not equal) and the and, or, and not operators—&, |, and ~, respectively. Any nonzero value is taken to represent a logical "true" value, while a zero value corresponds to a logical "false" as can be seen by evaluating, e.g., `3 & 4`, which produces the output `1`. Thus,

```
if ( S == 2 )
   xxx
elseif ( S == 3 )
   yyy
```

```
else
   zzz
end
```

executes the statements denoted by `xxx` if the logical statement `S == 2` is true, `yyy` if `S == 3`, and `zzz` otherwise. The `for` loop

```
for loop = 10 : -1 : 0;
       vR(loop) = sin(loop * pi / 10 );
end;
```

yields the array `vR = [ sin( π ) sin( 9π / 10 ) ... sin( π/10 ) 0 ]`, while `1 : 10` yields an array with elements from `1` to `10` in unit increments. *Mistakenly replacing colons by commas or semicolons results in severe and often difficult to detect errors*. If a `break` statement is encountered within a `for` loop, control is passed to the statement immediately following the `end` statement. An alternative to the `for` loop is the `while (logical condition) ... statements ... end` construct.

Vectorized Iterators: A *vectorized iterator* such as `vR = sin( pi: -pi/10: -1.e-4 )`, which yields, generates, or manipulates a vector far more rapidly than the corresponding `for` loop. `linspace( s1, s2, n )` and `logspace( s1, s2, n )` produce `n` equally/logarithmically spaced points from `s1` to `s2`. An isolated colon employed as an index iterates through the elements associated with the index so that `MRC(:, 1) = V(:);` places the elements of the row or column vector `V` into the first column of `MRC`.

Files and Function Files: A function that returns variables `output1, output2` ... is called `[ output1, output2, ... ] = myFunction( input1, input2, ... )` and normally resides in a separate file `myFunction.m` in the current directory, the first line of which must read `function [ aOutput1, aOutput2, ... ] = myFunction( aInput1, aInput2, ... )`. Variables defined (created) inside a function are inaccessible in the remainder of the program once the function terminates (unless `global` statements are present), while only the argument variables and variables local to the function are visible from within the function. A function can accept other functions as an arguments either (for Octave functions) with the syntax `fmin( 'functionname', a, b )` or through a function handle (pointer) as `fmin( @functionname, a, b )`.

Built-In Functions: Some common functions are the discrete forward and inverse Fourier transforms, `fft( )` and `ifft( )` and `mean( )`, `sum( )`, `min( )`, `max( )`, and `sort( )`. Data is interpolated by `y1 = interp1( x, y, x1, 'method' )`, where 'method' is `'linear'` (the default), `'spline'`, or `'cubic'`; `x` and `y` are the input *x*- and *y*-coordinate vectors; and `x1` contains the *x*-coordinate(s) of the point(s) at which interpolated values are desired. The function `roots( [ 1 3 5 ] )` returns the roots of the polynomial $x^2 + 3x + 5$.

Graphic Operations: `plot( vY1 )` generates a simple line plot of the values in the row or column vector `vY1`, while `plot( vX1, vY1, vX2, vY2, ... )` creates a single plot with lines given by the multiple (x, y) data sets. Hence, `plot( C, 'g.' )`, where `C` is a complex vector, graphs the real against the imaginary part of `C` in green with point marker style. Logarithmic graphs are plotted with `semilogy( )`, `semilogx( )`, or `loglog( )` in place of `plot( )`. Three-dimensional grid and contour plots with `nContours` contour levels are created with `mesh( mRC )` or `mesh( vX, vY, mRC )` and `contour( mRC )` or `contour( vX, vY, mRC, nContours )` where `vX` and `vY` are row or column vectors that contain the *x* and *y* positions of the grid points along the axes. The commands `hold on` and `hold off` retain graphs so that additional curves can be overlaid. Subsequently, axis defaults can be overridden with `axis( [ xmin xmax ymin ymax ] )`, while axis labels are implemented with `xlabel( 'xtext' )` and `ylabel( 'ytext' )` and the plot title is specified by `title( 'title text' )`. The command `print ( 'outputFile.eps', '-deps' )` or, e.g., `print( 'outputFile.pdf', '-dpdf' )` yields, respectively, encapsulated postscript or .pdf files of the current plot window in the file `outputFile.dat` or `outputFile.pdf` (`help print` displays all options).

Memory Management: User-defined variable or function names hide preexisting or built-in variable and function names, e.g., if the program defines a variable or function `length` or `length( )`, the Octave function `length( )` becomes inaccessible. Additionally, if the second time a program is executed a smaller array is assigned to an variable, the larger memory space will still be reserved by the variable causing errors when, e.g., its length or magnitude is computed. Accordingly, each program should begin with `clear all` to remove all preexisting assignments (a single construct `M` is destroyed through `clear M`).

Structures: To associate different variables with a single entity (structure) name, a dot is placed after the name as in

```
Spring1.position = 0;
Spring1.velocity = 1;
Spring1.position = Spring1.position + deltaTime * k/m *
                   Spring1.velocity
```

Variables pertaining to one entity can then be segregated from those, such as `Spring2.position`, describing a different object. The names of structures are conventionally capitalized.

4

ELEMENTARY MATHEMATICS

The following treatment of algebra and geometry focuses on often neglected aspects.

4.1 ALGEBRA

While arithmetic concerns direct problems such as evaluating $y = 2x + 5$ for $x = 3$, algebra addresses arithmetical inverse problems, such as the determination of x given $y = 11$ above. Such generalizations of division can be highly involved depending on the complexity of the direct equation.

4.1.1 Equation Manipulation

Since both sides of an equation evaluate to the same quantity, they can be added to, subtracted from, or multiplied or divided by any number or expression. Therefore,

$$\frac{a}{b} = \frac{c}{d} \tag{4.1.1}$$

can be simplified through *cross multiplication*, e.g., multiplication of both sides by bd to yield

$$ad = bc \tag{4.1.2}$$

Fundamental Math and Physics for Scientists and Engineers, First Edition.
David Yevick and Hannah Yevick.

Similarly, the left hand of one equation can be multiplied or divided by the left-hand side of a second equation if the right-hand sides of the two equations are similarly manipulated (as the right and left sides of each equation by definition represent the same value).

Example

Equating the quotients of the right- and left-hand sides of the following two equations

$$\begin{aligned} 3y(x+1) &= 4 \\ 4(x+1) &= 2 \end{aligned} \tag{4.1.3}$$

results in $3y/4 = 2$.

4.1.2 Linear Equation Systems

An algebraic equation is *linear* if all variables in the equation only enter to first order (e.g., as x and y but not xy). At least N linear equations are required to uniquely determine the values of N variables. The standard procedure for solving such a system first reduces the system to a "tridiagonal form" through repeated implementation of a small number of basic operations.

Example

To solve,

$$\begin{aligned} x + y &= 3 \\ 2x + 3y &= 7 \end{aligned} \tag{4.1.4}$$

for x and y, the first equation can be recast as $x = 3 - y$, which yields a single equation for y after substitution into the second equation. Alternatively, multiplying the first equation by two results in

$$\begin{aligned} 2x + 2y &= 6 \\ 2x + 3y &= 7 \end{aligned} \tag{4.1.5}$$

Subtracting the first equation from the second equation then gives

$$\begin{aligned} 2x + 2y &= 6 \\ y &= 1 \end{aligned} \tag{4.1.6}$$

The inverted pyramidal form is termed an *upper triangular linear equation system* and can be solved by *back-substituting* the solution for y from the second equation into the first equation, which then solved for x.

A set of equations can be redundant in that one or more equations of the set can be generated by summing the remaining equations with appropriate coefficients. If the number of independent equations is less or greater than N, infinitely many or zero solutions exist, respectively. Nonlinear equation systems can sometimes be linearized through *substitution* of new variables formed from nonlinear combinations of the original variables. Thus, defining $w = x^2,\ z = y^3$ recasts

$$\begin{aligned} x^2 + 3y^3 &= 4 \\ 2x^2 + y^3 &= 3 \end{aligned} \tag{4.1.7}$$

into the linear equations $w + 3z = 4, 2w + z = 3$.

4.1.3 Factoring

The inverse problem to polynomial multiplication is termed *factoring*. That is, multiplication and addition yield

$$(ax + b)(cx + d) = acx^2 + (bc + ad)x + bd \tag{4.1.8}$$

which is reversed by factoring the right-hand side into the left-hand product of two lesser degree polynomials. For quadratic (second-order) equations, the quadratic formula states that the roots (solutions) of $ax^2 + bx + c = 0$ are

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{4.1.9}$$

implying that the polynomial $ax^2 + bx + c$ can be factored as $(x - x_1)(x - x_2)$. Equation (4.1.9) is derived by first *completing the square* according to

$$\begin{aligned} ax^2 + bx + c &= a\left(x^2 + \frac{bx}{a}\right) + c \\ &= a\left(x^2 + \frac{bx}{a} + \frac{b^2}{4a^2}\right) - \frac{b^2}{4a} + c \\ &= a\left(x + \frac{b}{2a}\right)^2 - \frac{b^2 - 4ac}{4a} \end{aligned} \tag{4.1.10}$$

Multiplying N terms of the form $(x - \lambda_i)$ yields

$$(x - \lambda_1)(x - \lambda_2)\ldots(x - \lambda_N) = x^N + x^{N-1}\sum_i \lambda_i + x^{N-2}\sum_{\substack{i,j \\ i \neq j}} \lambda_i\lambda_j + \ldots + \prod_i \lambda_i \tag{4.1.11}$$