

WOODSTOCK

Modeling Reference

www.remsoft.com

Copyright© 1994-2006, Remsoft® Inc.

All rights reserved.

Woodstock Modeling Reference v 2006.8

Remsoft Incorporated, Suite 160 Frederick Square, 77 Westmorland Street, Fredericton, NB Canada E3B 6Z3

Tel.: (506) 450-1511

The information contained in this document is the exclusive property of Remsoft® Inc. This work is protected under the Canadian copyright law and other international copyright treaties and conventions. No part of this work may be reproduced or transmitted in any form or by any means, electronic, or mechanical except as expressly permitted in writing by Remsoft Inc.

Remsoft® is a registered trademark of Remsoft Inc. All other products, brands, etc. mentioned in this manual are the trademarks or registered trademarks of their respective companies and should be treated as such.

The information in this document is for information purposes only and is subject to change without notice. Remsoft assumes no responsibility for any errors which may appear in the manual.

Printed in Canada. Manual 2006.8, August 2006

REMSOFT INC.

SOFTWARE LICENSE AGREEMENT

IMPORTANT: THIS IS A LEGAL AGREEMENT BETWEEN THE INTENDED END-USER OF THE SOFTWARE DESCRIBED IN THIS AGREEMENT ("YOU") AND REMSOFT INC. ("REMSOFT"). BEFORE YOU CLICK "I AGREE" AND INSTALL THE SOFTWARE, PLEASE READ THIS SOFTWARE LICENSE AGREEMENT ("AGREEMENT") CAREFULLY.

BY CLICKING "I AGREE" AND INSTALLING THE SOFTWARE, YOU ARE AGREEING TO BE LEGALLY BOUND BY THE TERMS OF THIS AGREEMENT. IF YOU DO NOT AGREE TO THE TERMS OF THIS AGREEMENT, YOU ARE NOT AUTHORIZED TO INSTALL AND WILL BE UNABLE TO USE THE SOFTWARE.

Remsoft is the owner the Remsoft® computer software products known as the Remsoft Spatial Planning System, including Woodstock, Spatial Woodstock, Stanley and Allocation Optimizer, and is willing to license one or more of such products to you on the terms and conditions set out below.

Section 1 - Definitions

In this Agreement the following terms shall have the meanings set out below:

1.1 "License Fees" means the applicable license fees for the Software, as set out in the Remsoft Invoice.

1.2 "Related Materials" means user guides, operating manuals and specifications, whether in print or machine-readable media, in effect as of the date of shipment or download of the Software.

1.3 "Remsoft Invoice" means the invoice issued by Remsoft to you for software licensed to you under this Agreement.

1.4 "Software" means the Remsoft® software products identified in the Remsoft Invoice, and all future updated or upgraded versions of such products furnished to you by Remsoft.

Section 2 - Grant of Rights and Term

2.1 Subject to the terms and conditions of this Agreement, Remsoft hereby grants you, and you hereby accept from Remsoft, a perpetual, non-exclusive, non-transferable (except as set forth herein) license to load, install and use the Software (solely in object code format) and the Related Materials. You may load and install the Software on a stand-alone computer or on a network server, as specified in the Remsoft Invoice, and may permit use of the Software only by the number of users specified in the Remsoft Invoice.

2.2 You may not otherwise copy the Software and Related Materials except to make one copy of the Software and Related Materials for back up purposes only. The back up copy must include all copyright and trade-mark information contained on the original.

2.3 Except as expressly authorized in Section 2.1, no other rights are granted to you and you covenant not to make any other use of the Software and Related Materials whatsoever, including but not limited to decompiling, reverse engineering, disassembling, modifying, renting, leasing, loaning, distributing or creating derivative works from the Software or transmitting the Software over a network.

2.4 This term of this Agreement shall commence upon your receipt of the Software and Related Materials, and shall continue until terminated in accordance with Section 5.

Section 3 - License Fees

3.1 You shall pay Remsoft the License Fees as set out in the Remsoft Invoice. All License Fees are in Canadian dollars unless otherwise specified.

3.2 License Fees are exclusive of all taxes, including but not necessarily limited to sales, use, value-added or other taxes or levies on transactions contemplated under this Agreement. You shall pay Remsoft an amount equal to any tax Remsoft is required to collect and pay upon this transaction, unless you shall obtain and furnish evidence, in a form satisfactory to Remsoft, that you are exempt from the imposition thereof.

3.3 All License Fees pursuant to this Agreement and applicable taxes are due and payable 45 days from your receipt of the Remsoft Invoice.

Section 4 - Intellectual Property and Confidentiality

4.1 The Software and Related Materials and any trade-marks or trade names related thereto are owned or duly licensed by Remsoft and are protected by Canadian intellectual property laws and applicable international laws, treaties, and conventions. Remsoft (and its licensors) own and retain all right, title and interest in and to the Software and Related Materials. You agree to use your best efforts to protect the Software and Related Materials from unauthorized use, reproduction, distribution or publication. You shall not during or at any time after the expiry or termination of this Agreement in any way question or dispute the ownership thereof by Remsoft.

4.2 As part of this Agreement, Remsoft may disclose to you confidential and proprietary information relating to the Software or Related Materials. Unless otherwise indicated in writing, you agree not to disclose any confidential or proprietary information received from Remsoft, whether oral or written or in other tangible form including, without limitation, any past, present, or future research and development or business activities, corporate information, software development plans or any related documentation and materials to any third party without Remsoft's prior written consent.

4.3 You also acknowledge that such intellectual property and other rights belong to Remsoft and may only be used by you with the consent of Remsoft and during this Agreement. Upon expiry or termination hereof you shall forthwith discontinue such use, without any right of compensation for such discontinuation.

4.4 You acknowledge that any breach of this Section by you shall cause Remsoft irreparable harm. Accordingly, in addition to any legal remedies available to Remsoft for losses or damages resulting from such breach, Remsoft shall have the right to seek a restraining order, injunction, specific performance, or other equitable relief (or the equivalent of any such relief, though known or designated by some other name or term) from any court of competent jurisdiction.

Section 5 -Termination

5.1 This Agreement shall terminate:

(a) immediately upon your failure to make payment when due pursuant to Section 3.3;

(b) 30 days from the date of receipt from Remsoft of notice in writing to you of your default in respect of any of your other obligations pursuant to this Agreement, if you fail to remedy the default within such time;

(c) immediately if you make an assignment in bankruptcy or a receiver is appointed to undertake all or part of your property; or

(d) immediately upon receipt by Remsoft of notice in writing from you that you have discontinued use of the Software and Related Materials.

5.2 Upon termination, you shall discontinue use of the Software and Related Materials and deliver to Remsoft all Software and Related Materials provided by Remsoft and destroy all copies made thereof and shall warrant in writing to Remsoft within 30 days of termination that the foregoing conditions in this Section 5.2 have been complied with.

Section 6 - Warranty and Limited Liability and Indemnity

6.1 **THE SOFTWARE AND RELATED MATERIALS ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT. REMSOFT DOES NOT WARRANT THAT THE SOFTWARE WILL MEET YOUR REQUIREMENTS OR THAT ITS OPERATION WILL BE ERROR-FREE.**

6.2 THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE SOFTWARE AND RELATED MATERIALS IS WITH YOU. REMSOFT'S ENTIRE LIABILITY AND YOUR EXCLUSIVE REMEDY SHALL BE (A) THE REPLACEMENT OF ANY DEFECTIVE MEDIA AND RELATED MATERIALS WHICH ARE RETURNED TO REMSOFT WITHIN 30 DAYS OF PURCHASE WITH PROOF OF PURCHASE OR (B) REFUND OF THE LICENSE FEES AND TERMINATION OF THIS AGREEMENT.

6.3 UNDER NO CIRCUMSTANCES, INCLUDING NEGLIGENCE, SHALL REMSOFT BE LIABLE FOR ANY DIRECT, INCIDENTAL, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES ARISING OUT OF OR RELATING TO THIS AGREEMENT OR THE USE OF THE SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.

6.4 You acknowledge and agree that the Software is designed to provide information based on models created and data entered by you (collectively, "**Customer Content**") and that Remsoft has no responsibility for Customer Content and is not engaged in rendering models or predictions or in providing consulting services or business advice of any kind to you.

6.5 You agree to indemnify Remsoft against all liabilities, costs and expenses of any kind which Remsoft may incur as a result of any third party claim relating to the use of the Software, including any claim for negligence, or infringement of any copyright or other proprietary right relating to the use of Software or to any Customer Content.

Section 7 - General

7.1 This Agreement shall be binding on and enure to the benefit of Remsoft and its successors and assigns. This Agreement may not be transferred or assigned by you without the prior written consent of Remsoft. Any attempt by you to transfer or assign this Agreement or your rights hereunder shall constitute a material breach of this Agreement.

7.2 In the event any provision of this Agreement is held to be invalid or unenforceable, the remaining provisions of this Agreement will remain in full force and effect.

7.3 The waiver by either party of any default or breach of this Agreement shall not constitute a waiver of any other or subsequent default or breach.

7.4 This Agreement may not be modified or amended except by a subsequent written agreement signed by the parties.

7.5 This Agreement constitutes the entire agreement between the parties and supercedes all prior agreements, written or oral, concerning the subject matter of this Agreement.

7.6 This Agreement shall be governed by and construed in accordance with the laws of the Province of New Brunswick, Canada, excluding (i) its conflicts of laws principles, and (ii) the United Nations Convention on Contracts for the International Sale of Goods. Each party irrevocably and unconditionally submits to the non-exclusive jurisdiction of the courts of the Province of New Brunswick, Canada.

7.7 Any notice required or permitted under the terms of this Agreement or required by statute, law or regulation shall (unless otherwise provided) be in writing and shall be delivered in person, sent by registered mail or air mail as appropriate, properly posted and fully prepaid in an envelope properly addressed or sent by facsimile to the respective parties at the address set forth on the most recent invoice or receipt, or to such other address or facsimile number as may from time to time be designated by notice hereunder. Any such notice shall be in the English language and shall be considered to have been given at the time when actually delivered if delivered by hand, or upon the next working day following sending by facsimile or in any other event within 10 days after it was mailed in the manner provided.

Contents

Chapter 1
Introduction 11
- **About this manual** 11
- **Using this manual** 11
- **Conventions in this manual** 12
- **What's next?** 13

Chapter 2
Guiding Principles 15
- **What makes a forest model?** 15
- **Terminology** 16
- **Reports and graphics** 24
- **Simulation structures** 24
- **Linear programming structures** 27

Chapter 3
Inventory projection 31
- **Concept** 31
- **Example** 32
- **Binary Search** 33

Chapter 4
Monte Carlo simulation 35
- **Concept** 35
- **Probability and its implications** 36
- **Examples of stochastic processes** 37
- **Conclusion** 40

Chapter 5
An overview of linear programming 41
- **Concept** 41
- **Review of coordinate systems for graphics and plotting** 41
- **Assumptions of linear programming** 45
- **The steers and trees problem** 47
- **Computer based solution methods** 52

Conclusion 57

Chapter 6
A simple Woodstock model 59
The Tom Pike problem revisited 59
Modeling the Tom Pike problem 60
An alternative formulation 70

Chapter 7
Tutorials 73
About the structure of sample models 73
Tutorial One - MODEL1 73
Tutorial Two - MODEL2 74
Tutorial Three - MODEL3 75
Tutorial Four - CASE1 and CASE2 77
Tutorial Five - CASE3 78
Tutorial Six - CASE4 79
Tutorial Seven - CASE5 80
Tutorial Eight - CASE6 81
Tutorial Nine - CASE7 82
Tutorial Ten - CASE8 85
Summary 86

Appendix 1
Tutorial Answers 89
Tutorial One 89
Tutorial Two 89
Tutorial Three 90
Tutorial Four 91
Tutorial Five 91
Tutorial Six 92
Tutorial Seven 92
Tutorial Eight 92
Questions? 93

Appendix 2

Glossary of Terms 95

Index 103

x

Chapter 1: Introduction

About this manual

This manual, *Woodstock Modeling Reference,* accompanies the other materials in the Woodstock Forest Modeling Guides:

The Woodstock User Guide provides information on building and running Woodstock models with particular emphasis on the syntax of the Woodstock modeling language and considerations in choosing a model formulation.

This manual is designed primarily for new users, providing instructions on how to develop models using the Woodstock modeling language.

You'll also want to refer to the Spatial Woodstock manual for additional information on modeling with both Woodstock and Spatial Woodstock.

Using this manual

Woodstock Modeling Reference is geared to those people who are new to Woodstock or even to forest modeling itself. The manual starts off by reviewing different modeling approaches and discusses the advantages and disadvantages of each approach. Then, a discussion of conceptual modeling is given where you are encouraged to think about your management problem in terms of processes that can be represented in a computer model. Finally, you are presented with a hypothetical management problem that you are to address by creating a Woodstock model. Over the course of the tutorial, the scope of the problem is increased in complexity. You are encouraged to complete the exercises yourself and compare your approach to the one we selected.

This manual does not provide information about all features available in the Woodstock modeling system. If you want more information on optional syntax or features available, refer to the corresponding sections in the Woodstock User Guide. Once you've worked through the tutorial you should be ready to tackle more involved problems that may require advanced features.

What's inside

Woodstock Modeling Reference is divided into two major parts.

Part 1 of the manual deals with a brief review of deterministic and stochastic models and it discusses the different model formulations and structures that are possible using Woodstock:

- Chapter 2, “Guiding Principles”, describes some of the underlying principles of modeling and how these principles guide the use of Woodstock. The chapter begins by looking at real processes and how they might be represented in a model framework.
- Chapter 3, “Inventory projection”, describes some of the underlying principles of inventory projection and some advantages and disadvantages of these models.
- Chapter 4, “Monte Carlo simulation”, describes some of the underlying principles of Monte Carlo simulation and some advantages and disadvantages of these models.
- Chapter 5, “An overview of linear programming”, describes some of the underlying principles of linear programming and some advantages and disadvantages of these models.

Part 2 presents a framework to help you learn the Woodstock syntax and to guide you in the development of a Woodstock model. You will formulate a simple Woodstock model in Chapter 6. Beginning with a landscape classification scheme, you move on to actions definitions, transitions, outputs and yields. Finally, we cover the graphics and reporting features that can be tailored to almost any need. In Chapter 7, you work through the sample models provided on your distribution disk.

- Chapter 6, “A simple Woodstock model”, illustrates how to tackle a word problem by subdividing the problem into components and how to represent these components in Woodstock syntax. A functioning Woodstock model is dissected and each component is explained. Along with the syntax is some discussion of differences arising from formulation restrictions and some strategies for arriving at good solutions.
- Chapter 7, “Tutorials”, introduces you to the sample models provided on your distribution disk and suggests some exercises that will help you understand these models better and will provide practice as you learn the Woodstock syntax.
- Appendix 2, “Glossary of Terms.”, contains definitions for terms used throughout the Woodstock documentation.

Conventions in this manual

Emphasis *Italics* are used to introduce new terms that are defined fully in the Glossary.

Examples Examples of valid Woodstock syntax are presented in a fixed font:

```
SECTION
*SUBSECTION modifier _keyword
; this is a comment
```

Optional keywords and modifiers Optional keywords or modifiers will be enclosed in angle brackets, <keyword>.

What's next?

Woodstock is a very sophisticated and flexible modeling system designed to give you the power to model almost any aspect of forest dynamics and management. In order for you to gain the maximum benefit from Woodstock's capabilities, we recommend that you at least skim through this manual. The Woodstock Modeling Reference will guide you to a basic understanding of how Woodstock works, how to develop a conceptual model of forest dynamics, how to create Woodstock input files and where to find information about syntax as well as working examples. If you are completely new to Woodstock, we encourage you to complete the tutorials in Chapter 7.

Even if you have been using the system for some time, you may need to incorporate new features that you have not used before into your models. Skim through this manual to refresh your memory and refer to appropriate sections that are applicable to your situation in the Woodstock User Guide or in the context sensitive help. Terminology specific to Woodstock models is found in Appendix 2, "Glossary of Terms."

Chapter 2: Guiding Principles

In this chapter, we will go over some of the underlying principles of modeling and how these principles guide the use of Woodstock. We will begin by looking at real processes, noting what the principle components of the system are and then attempt to represent them in a model framework. The Woodstock model syntax was designed to make it relatively easy to represent conceptual forest models; thus, the difficulty in using Woodstock is not so much learning the syntax, but rather learning how to represent real processes in a modeling framework without oversimplification or undue complexity.

What makes a forest model?

Most forest planning models are designed for a specific purpose, and offer a range of options that the user may employ. For example, a program may offer two ways to simulate harvests (clear cuts, shelterwood cuts), three different regeneration options (natural regeneration, planting with ordinary growing stock, planting with genetically improved growing stock) and two different cultural treatments (precommercial thinning, fertilization). In this kind of planning model, a computer programmer has represented a particular conceptual model of the processes involved in each option within the program code. To use one of these programs, typically you evaluate the range of options available and, if suitable, you adapt your forest problem to fit the prescribed model structure.

Woodstock takes a different approach. Rather than provide a fixed range of model options, Woodstock provides you with the means to define any number of different forest classes, activities and outputs. Based on your conceptual model of forest processes, you define a model using your terminology, with actions that work exactly the way you want them to and any output you care to define. You do not have to worry about how the model implements an action because you must tell it how the action works. You do not have to generate long pages of program output to get a single number because you define what goes into reports. So, how do you come up with a conceptual framework for a Woodstock model in the first place? In this chapter, we will discuss methods for developing a conceptual model of the forest.

Terminology

In developing the Woodstock modeling system, we have standardized on terminology that we feel best explains concepts and functions within the modeling language. Some of these terms will probably be self-evident or at least consistent with common usage, but others likely have subtle differences in meaning, particularly if you have used a different modeling system extensively. Therefore to avoid potential misunderstanding, a brief explanation of the most important terms is given here. A complete glossary of terms is given in the Woodstock User Guide.

Conceptual versus computer models

Everyone knows that a model is an abstraction of reality, a representation of a real system in terms that exemplify form and function. For example, consider what we mean by a stand. One textbook definition of a *stand* is "a community particularly of trees possessing sufficient uniformity as regards composition, constitution, age, spatial arrangement or condition, to be distinguishable from adjacent communities, so forming a silvicultural or management entity." In practice, determining what is and is not a stand is not so easy. Stands rarely have sharp boundaries and there is often a continuous gradient from one stand into another. Furthermore, where one person acknowledges the existence of a single stand, another with a different point of view may see two or more stands. Thus, a stand is a conceptual model that has common elements among different people, but is highly dependent on one's point of view.

A *conceptual model* attempts to capture the essence of a real system by isolating and retaining the important elements of the system, and disregarding the rest. A conceptual model of a forest encompasses:

- a method of describing the forest (a classification scheme),
- a range of silvicultural activities and natural events, along with a specification of how affected areas develop following these activities and events,
- a means of measuring the impact of interventions and natural events (forest conditions, benefits and outputs),
- a method of accounting for the passage of time.

For consistency and to facilitate interpretation of results, the model processes should mimic those of the real system as much as possible.

A Woodstock model is a computer-compatible form of a conceptual model. Unlike many forest modeling systems, where you must adapt your problem to fit the structures provided by the system, the Woodstock modeling language was designed to adapt to your way of thinking about a problem. Thus, your Woodstock input files are like the instructions of a computer program - just as the compiler is not part of your final

program, the Woodstock interpreter used to execute your forest model is not part of the model itself.

Describing the land base

In a Woodstock model, your land classification scheme gives you the means to refer to specific parcels of the forest with varying degrees of generality. A *development type* represents all the stands that have the same yield and developmental characteristics regardless of age. A *development type class* represents a single age class of a development type. An *analysis area* represents a specific portion of a development type class and may represent a single stand or multiple stands that are identical in age and developmental characteristics. In contrast, a *common emphasis group* shares some common developmental characteristics, but it may be composed of several development types.

The following table includes examples to illustrate how these terms are commonly applied.

Scenario	Designation	Explanation
all loblolly pine plantations on high sites in region 3, regardless of age	development type	you are referring to a specific group of stands based on developmental characteristics but you are not differentiating them based on age
all 10 year old slash pine plantations on medium sites in region 2	development type-class	you are referring only to a single age class of the entire development type
all the high site plantations, regardless of species planted or age	common emphasis group	you are referring to a particular land category, which incorporates different species and age classes

In a Woodstock model, you associate forest conditions with actions and yields with varying degrees of specificity. For example, yield tables that detail harvest and inventory volumes are usually specified for development types, but eligibility for silvicultural actions is usually specified for management emphasis groups. Of course, nothing prevents you from using more specific associations in your models, but there are significant disadvantages in terms of conciseness and clarity to doing so.

Representing dynamic processes

In a Woodstock model, every event that changes the development of the forest is represented by actions and transitions. An *action* is simply any event that occurs, such as a silvicultural treatment, a natural catastrophic event, an administrative activity or an allocation choice. When an action occurs on a development type class, the class is assumed to undergo some sort of change, a *transition* to a new age class and/or one or more development types.

Many forest planning models classify harvesting activities differently from other silvicultural activities and if they permit them, natural events are represented differently again.

In a Woodstock model, there is no differentiation in activities and all actions are specified in a common section. Differentiation occurs when you specify the transition matrix associated with each action. Transition matrices may be one-to-one in that all of the area treated is assigned to the same development type class after treatment. Alternatively, in a one-to-many transition matrix the area treated is assigned proportionately to multiple development type classes.

Note that aging of stands is not considered an event that changes stand development. For example, it may be common for a stand to be dominated by an early successional intolerant species that gradually gives over to tolerant species. The successional changes that occur are thus assumed to be an integral part of the development type, and the dominance of individual species are tracked as yield components. Conversely, to model seral stages as different development types, you must define actions that effect the shift from one seral stage to the next.

For example, when mature jack pine stands are clearcut and site prepared this year, you may assume that the area harvested transitions to one-year old regenerating jack pine stands. Or, you may assume mixed-wood stands clearcut this year transition to 50% year old mixed-wood condition and 50% year old hardwood condition. For a spruce-fir stand that undergoes senescence, you may assume that the old stand transitions to a 30 year old spruce-fir stand due to advanced regeneration. Finally, you may model an administrative activity such as boundary marking in which the transition causes no changes in age or development type whatsoever.

Actions are an important part of a Woodstock model, because they are the triggering mechanism for reporting harvest volumes, acres treated, silvicultural costs incurred, inventory measures and any other measures of management success.

Measuring management success

In most forest models, the information used to evaluate management success is based on forest condition and benefits production. Forest condition usually incorporates things like standing inventory by species and volume, distributions by age class and forest cover type, wildlife habitat suitability, etc. Benefits production includes harvest volume, forage, recreational visitor days, net revenues and others. Unlike many forest models, a Woodstock model does not differentiate between these types of management indicators and they are all represented as *outputs*.

When you define an output in a Woodstock model, what you are really doing is telling the Woodstock interpreter how to calculate it. This “recipe” includes the development types that can produce output, the action that triggers production of the output and some per-unit-area estimate of yield. When an action occurs, the

area affected by the action is multiplied by the estimate of yield to determine the quantity of output produced.

In a Woodstock model, *yield components* are the basis for calculating outputs. Yield components may be estimates of volume, basal area or other stand attributes, estimates of silvicultural costs or revenues, or virtually any other characteristics of a development type. All of the yield information for a yield component is provided as a yield table, that may be either age-dependent, time-dependent or compound.

An *age-dependent yield table* represents a quantity that varies with the age of a development type rather than with the planning periods in the planning horizon. For example, standing volume per acre is a common age-dependent parameter of a development type. Conversely, a *time-dependent yield* table represents quantities that vary with planning periods rather than the age of a development type. Time-dependent yields are often associated with common emphasis groups - you could represent accessibility within a roadless area that is scheduled to be opened up in the future.

A *compound yield table* represents quantities that vary both with the age of a development type and the planning period. A good example of this type of yield would be revenues from harvesting. Since harvest volume per acre varies with the age of a stand and the stumpage price varies with time, revenues must be calculated by multiplying the age-dependent and time-dependent yield curves. In Woodstock models, a yield table that is the product of two other yield tables is *complex*. Therefore, a compound yield table is necessarily complex but a complex yield table is not necessarily compound since it could be the product of two age-dependent yield tables.

Anatomy of a Woodstock model

Now let's look at the overall structure of a Woodstock input file. Woodstock models are arranged along functional lines, and each section performs a specific task. Thus, we will concentrate on the functions of each model section, and deal with the specifics of syntax in the subsequent chapters.

Common structures To begin, we will look at the model sections that are common to all Woodstock models. Most of the dynamics of a forest ecosystem are represented in the following sections, and regardless of the model formulation you plan to use, you will need to include these sections in your model. The method you use to model management control (inventory projection or linear programming) will determine what other sections will be required, and these sections will be covered later.

Time definition A Woodstock model keeps track of time in terms of *planning periods* only. Depending on your definition, a planning period may be individual years, decades, or some other time interval. The Woodstock model assumes all periods are the same length, and you must be careful to specify age-dependent

information consistently. The interpreter will not interpolate yearly increments from yield tables with 5-year entries for instance.

You also must specify the number of periods you want your model to simulate. The *planning horizon* is specified in the Control section, and the number of periods you specify is assigned to the global variable `_LENGTH`, that may be used when specifying time intervals in your model.

When planning periods represent a time interval spanning multiple years, you must consider that (in a Woodstock model) all actions are considered to occur at a single point in time regardless of the planning period length. In terms of yields, it is important to know whether quantity represents a stand at the lower limit, the middle or the upper limit of the age class. The possibility exists for significant bias in your yield estimates if your assumptions about when an action occurs within a period is not consistent with yield projections. A common assumption is to assume that actions and yields occur at the mid-point of a period but the Woodstock interpreter does not require any particular assumptions about time other than that they are consistent throughout your model.

Land organization A Woodstock model uses a sophisticated classification scheme to identify and keep track of individual pieces of land in the forest. You specify a number of *landscape themes* (analogous to fields in a GIS database) and the different *thematic attributes* that correspond to each theme. For example, you might want to stratify your forest on the basis of working group (species or seral stage), stocking level, site quality and condition class (thinned, insect damaged, etcetera). Each combination of valid attributes ordered in the correct theme sequence represents a unique *development type*. In order to differentiate two development types on the basis of eligibility for treatment or yield estimation, they must differ on at least one theme attribute. Otherwise the Woodstock interpreter will consider them identical development types.

Landscape themes do not have to represent biophysical parameters of the forest only. You may just as easily use administrative regions, restriction zones, aesthetic ratings, etcetera, as landscape themes in your Woodstock model. Given the fact that geographic identifiers may be used as easily as biophysical ones, a Woodstock model can be formulated in several different ways.

Example: *Consider a forest composed of 12 stands (8 loblolly pine, 4 mixed hardwood and 2 meadows), all divided into 2 zones. A stratum-based approach would recognize 3 strata for per-acre yield estimates. Using an area-based approach, we would recognize 14 units, one for each stand. Using a mixed approach, the 3 strata are further subdivided into either zone A or zone B, yielding a total of 6 analysis areas.*

The first approach is the *stratum-based* approach, where discontiguous but otherwise homogeneous lands are aggregated together to form a single analysis area. The advantage of this approach is that yield estimation is quite

straightforward, and large areas of the forest can use the same per-acre yield tables. The disadvantage of the approach is that spatial relationships are generally lost in the aggregation step.

The second approach is the *area-based* approach, where contiguous, heterogeneous lands are represented as distinct entities. Because the analysis areas are geographically defined, the spatial relationships are preserved but yield estimation is more complicated since each analysis area incorporates multiple forest types in varying proportions. Thus, there is no economy in using per-acre yield estimates since there are one-to-one correspondences between analysis areas and yield tables.

The third approach is a combination of the first two, allowing the model to recognize specific geographic locations for some portions of the forest, and broad strata for the remainder. This mixed approach is useful where spatial control is important for a few key areas but the majority of the forest can be modeled using strata.

Actions and transitions At its most basic level, a Woodstock model is simply a mechanism for growing stands to their maximum age, killing them and regenerating them. The Woodstock interpreter has no other intrinsic functions other than the ability to take an inventory of the forest condition. A real life analogue to such a model would be a forest that is essentially unmanaged, except that fire and insect controls prevent catastrophic losses. In modeling a wilderness area, we are less concerned about predicting output levels of commodity products and the actions that produce them, so it is possible that a Woodstock model would have no actions defined or only those that mimic nature (e.g. wildfire).

---

* *Woodstock provides intrinsic actions that allow for inventory outputs and death transitions in the absence of user-defined actions.*

On the other hand, forests that are used to produce timber, recreation or wildlife benefits undergo multiple silvicultural interventions, and it is important to be able to predict the aggregate outcome of these interventions. A Woodstock model of this type of forest uses actions to trigger changes in the structure of the forest and to trigger the production of outputs. In fact, outputs and transitions can never occur in a Woodstock model without a triggering action.

An action may occur and not trigger any outputs, but the development type that has had an action applied to it must undergo a transition. The transition may change the age of the type and/or one or more of the landscape thematic attributes. The rationale for defining an action in this way is to generalize the concept so that not only can conventional silvicultural activities be represented as actions, but so can administrative activities or allocation decisions.

Let us consider 2 management scenarios that can be represented in a Woodstock model. The first scenario is a clearcut and plant (`CC&P`) prescription to be used on pine plantations. Following the `CC&P` activity, the standing timber is removed, a new pine plantation is created in its place, and 50 cunits of harvest volume is produced. The second is an allocation decision (`ALLOCT`, `ALLOCW`) for a mature spruce-fir stand. If the stand is allocated to timber production, the stand is eligible for harvesting immediately. If the stand is allocated to wildlife management, it is not eligible for harvesting for 30 years because it is required for winter thermal cover.

Under `CC&P`, the acres that are harvested are transferred from the 25+ age classes to a new age class, but the attributes will not change. Under `ALLOCT` and `ALLOCW`, the acres that are allocated to timber or wildlife are shifted from the class with sf none attributes to either the classes with sf timber or sf wildlife attributes but the ages will not change.

Action	**Transition**	**Outputs**	**Explanation**
`CC&P` eligible on plantations that are 25 years of age or older	`pine(25) --> pine(0)`	wood volume, harvest cost, planting cost, acres planted, acres harvested	Acres harvested are shifted from 25+ aged pine plantations to 0 age pine plantations
`ALLOCT` eligible on unallocated sf types	`sf none --> sf timber`	acres of `sf` managed for timber	Acres allocated to timber are shifted from no designation to timber management
`ALLOCTW` eligible on unallocated `sf` types	`sf none --> sf wildlife`	acres of `sf` managed for wildlife	Acres allocated to wildlife are shifted from no designation to wildlife management

In a Woodstock model you must specify just three parameters to define an action: an action code to identify the action, whether the action changes the age of the type being treated and what the eligibility criteria are for applying the action. The eligibility criteria for an action are based on development type and one or more yield components. For example, you could define pruning as an action that requires pine plantation types (thematic attributes) to have more than 20% log volume (yield component).

Actions can be defined as *Monte Carlo* stochastic events where there is a probability of the action occurring on a given unit of area. For example, suppose you have 10 woodlot owners and there is a probability of 60% that any of them will harvest their woodlot. You could model this in a Woodstock model by having the interpreter randomly draw 10 numbers between 0 and 1, and applying the action only to those woodlots where the random number was less than or equal to 0.6.

Once an action has been defined, you must provide a transition specification for it, defining what will happen to each development type that is eligible for treatment. Optionally, you may specify that *transitions* be modeled as Monte Carlo events. An interesting thing to note is that in defining an action, you are telling the Woodstock interpreter how to model the action but you are not necessarily telling it to implement it – actions are only carried out if explicitly requested.

Yields and outputs

For most people, the terms yield and output are synonymous (i.e., some benefit or commodity extracted from the forest) and most forest modeling systems take a similar viewpoint. However, in a Woodstock model yields and outputs are very different entities. A set of *yield components* describes stand type development, indicating how the type changes through time. Outputs are measurable quantities that are of interest, whether they are harvest volumes, areas treated, costs incurred, revenues generated, forest conditions, or other items.

An output is produced from one or more development types when a triggering action occurs. This linkage to actions cannot be overstated, since the triggering action(s) must be provided as part of the output definition. Although you may give an output any name, such as “pine pulp production,” a more precise name for it would probably be “pine pulp produced through clearcut logging of pine types.” Outputs do not have to be tangible quantities like cubic metres of sawtimber or tonnes of biomass - they can also be acres harvested, recreational visitor days available, etc.

Yields and outputs are related because outputs are always derived from a yield component and the magnitude of an action. For example, to report pine volume harvested, the Woodstock interpreter would multiply the per acre yield component pine by the number of acres harvested. Similarly, planting cost would be reported by multiplying the planting cost per acre yield component and the number of acres planted. Since yield components can have any type of units, the number of different outputs that are possible is almost limitless.

For a partial cut *of a 2 ha stand with 80 m^3/ha of timber before the cut and 45 m^3/ha after the cut, the reported cut volume will be: (2·80)-(2·45)=70 m^3.*

Some actions are defined with the option `*PARTIAL`, and that requires more effort by the interpreter when generating outputs. First the Woodstock interpreter finds the yield components corresponding to the development type being treated and the development type(s) that result from the action. Then, to calculate the amount of the output produced, the Woodstock interpreter multiplies the area treated by the difference between the pre-treatment and the post-treatment yield components.

Rather like in the `ACTIONS` section, an output definition tells the Woodstock interpreter how to calculate an output. However, unless you

specifically request that an output be reported or include it in a controlling structure such as simulation target or a LP constraint, the Woodstock interpreter does not actually carry out the calculations.

Reports and graphics

In a Woodstock model, you may define the contents of a report file or graphics screen by specifying which outputs you wish to have reported. Outputs may be specified for a single period, multiple periods or one or more intervals in report files. Report files may be formatted as ASCII text files, DBF tables, in a common spreadsheet, or in a CSV format.

Graphics screens are defined in terms of graph windows and lines. First, you specify how many graph windows you want (and optionally the size and location of them) and then you specify which outputs to report as line graphs and the graph window to display them in. The interpreter displays the output level in each period of the planning horizon.

Simulation structures

The conventional operations research viewpoint holds that simulation models are inherently stochastic, since the problems they address are not amenable to solution by optimization methods that are largely deterministic in nature. However, heuristics can be used to solve problems deterministically and, for lack of a better term, we will call any Woodstock model that is formulated using inventory projection or Monte Carlo methods a simulation model.

Woodstock simulation models require that you specify how much of any action to apply in a given period, and the order in which actions are to be applied. You have a great deal of control over what goes on in simulation model - in a linear programming model, the LP solver determines the timing and extent of actions for you. Although LP is well suited to choosing among alternatives with trade-offs, it cannot be used to model stochastic actions or outcomes.

Output targets

Management control in a Woodstock simulation model is exerted through output targets, that are simply how much of each output you want produced each planning period. Although Woodstock knows how to calculate how much of an output is produced, you must tell the interpreter which actions to undertake and the scope of the action. Taken together, the output targets and the actions that are the sources of the outputs form the action *queue*.

Area control method *Harvest up to 1000 acres of land per period. Clearcuts make up 100% of harvests.*

Volume control method *Harvest up to 1 million m^3 of wood per period. Clearcuts make up 100% of harvests.*

The action queue is really just a list of actions you want the Woodstock interpreter to perform. Each period the interpreter starts at the top of the list implementing each action and evaluating whether it has succeeded in meeting the target you specified. You tell the Woodstock interpreter how much area to treat, in one of several ways. The most direct approach is to use area control - you specify an output target for the area you want treated using a specific action.

A less direct approach is to specify an output target for something produced by performing a single action (e.g., harvest volume arising from clearcutting). In this case, the area treated depends on the yield components associated with the output and will be greater for low yield stands and less for high yield stands.

Volume control method *Harvest up to 1 million m^3 of wood per period. First, use thinning to produce up to 250,000, then use partial cuts to produce another 250,000, and then use clearcutting to produce the final 500,000 m^3 of wood desired.*

The most general case is to specify an output target for something that is produced by multiple actions. In this scenario, you control both the proportion of the target assigned to each action and the order in which the actions are performed. If the forest were composed of a single, uniform stand, it would not matter which part of the forest were treated first. However, with multiple forest types and age classes, it is important that you be able to apply actions in a predictable way. This is accomplished in a Woodstock model through selection rules.

Selection rules are simply a ranking mechanism for deciding which portion of the forest to treat first. For example, any spruce-fir stand between the age of 50 years and break-up age may be eligible for harvest. However, stands that are 50 years of age are still growing vigorously whereas 150 year old stands are not; obviously you would like to harvest the slower growing stands first. To choose appropriately, you could instruct the Woodstock interpreter to estimate growth by comparing the per-acre volume of stands with the per-acre volume projected in the next period. Then, you would have the interpreter rank the stands according to growth, and harvest first those with the most negative values.

Now suppose you have two stands that have equal estimated growth rates, but one has more log volume than the other. If you were concerned with maximizing log volume, you might prefer to harvest the one with the higher log volume before the other. If you combine the first selection rule (slowest growing first) with the second (highest log volume first), you will have an overall ranking where older stands are harvested first but where a tie exists, the stands with higher log volumes are harvested before those with lower log volumes.

** In addition to any yield component in a yield set, you may use stand age in your selection rules (e.g., oldest-first rule).*

In a Woodstock model, selection rules are based on yield components. Although this might seem like a limitation, remember that a yield component is just a stand attribute, and can represent anything including standing volume, basal area, site class, etc. Because yield components project changes over time, they provide a better means of ranking stands for harvest than a static attribute alone would. As an alternative, you can specify a random selection rule that simply lists the eligible development types in a random order, but it cannot be used with any other selection criteria.

If the output target of a deterministic model is achieved, the Woodstock interpreter increments the planning period, ages the forest by one period and in the next period it starts at the top of the queue once again. If the target is not achieved, the interpreter ends the simulation and advises you that it failed to meet the target. There could be several reasons why an output target could not be attained:

- there may not be enough area available (according to your eligibility criteria) to treat the area you specified,
- the ordering of actions may have resulted in one action not being implemented because earlier actions exhausted the eligible acres,
- the output level in your target may simply be unattainable.

A failure to attain a target in a simulation does not mean that it is infeasible, but rather indicates that the current management scheme will not meet your goals. You may be able to determine a feasible solution by rearranging the queue, or by changing the proportions of the target associated with each action. In a Monte Carlo simulation, the situation changes slightly. The Woodstock interpreter still applies actions in the order specified in the queue and attempts to meet the output target. However, because the actions and/or the outcomes of actions are stochastic in nature, an output target that is attainable in a deterministic model may not be always attainable in a stochastic model.

Failure to attain an output target is a useful piece of information. In a deterministic model, the Woodstock interpreter can use a binary search algorithm to quickly converge to an optimal solution based on whether the current run was successful or not. Conversely, the Woodstock interpreter can rerun a stochastic simulation multiple times, keeping track of the number of unsuccessful runs and the periods in which the output targets were not attained.

Linear programming structures

In a Woodstock simulation model, you specify activity levels through output targets and then run the simulation to see if the target was achieved. By testing a number of different target levels and activity levels, you can determine the best solution among a number of management scenarios. In other words, your analysis consists of telling the model what to do and observing the outcome - in a linear programming (LP) formulation, the tactic is reversed.

With LP models, you specify the kind of solution you are looking for by specifying an *objective function* and *constraints*, and then you solve the LP to determine what activity levels are needed to achieve your goals. In other words, LP analysis consists of telling the model what you want and observing the optimal means of achieving it.

Constraints

There are three types of constraints that can be specified in any linear program (including those generated by Woodstock):

- *equality constraints* where some quantity must be exactly met (e.g., $X = 25$),
- *upper limit constraints* where some quantity must never exceed a given amount (e.g. $X < 25$),
- *lower limit constraints* where some quantity must always exceed a given amount (e.g. $X > 25$).

Note: *All the implicit constraints generated by the Woodstock interpreter are equality constraints rather than the upper limit or lower limit variety.*

In a Woodstock-generated linear program, there are two classes of constraints: *implicit constraints* that the interpreter sets up automatically, and *explicit constraints* that you must specify within the Optimize section of a Woodstock model.

The implicit constraints are used in 3 key areas:

- to assure that no more area is allocated to a management prescription than physically exists,
- to assure that areas harvested are properly assigned to new regeneration classes and
- to establish accounting variables that are used in other explicit constraints.

A wide variety of explicit constraints are available in Woodstock to give expression to various restrictions that control the management of your forest.

The explicit constraints that a user may specify will be in one of the following categories:

Category	Function	Example
Absolute	Controls the physical or financial amount of some output that can occur in one or more planning periods	Total silvicultural expenses must be less than $15 million per period
Flow	Controls the physical or financial rate at which some output can change from period to period	Total volume harvested in each period may change by at most 10%
General relational	Controls the proportional relationship between two outputs in one or more planning periods	The area planted to spruce must equal at least 50% of the area planted to pine

Absolute constraints are simple in that they do not reference other time periods or other outputs. If an absolute constraint on an output is applied in every period of a ten-period planning horizon, there will be 10 constraint rows generated that force the amount of output to be greater than, equal to or less than a fixed quantity. Examples of absolute constraints include:

- a requirement that the amount of revenue that must be received each period from the entire forest exceed $100,000,
- a requirement that the amount of sediment produced per period in management zone 4 be less than 5,000 tonnes,
- a requirement that the amount of small diameter timber that can be produced through clear-cut harvests be no more than 50,000 cords and no less than 20,000 cords.

Flow constraints by necessity reference other time periods. If an even flow constraint on an output is applied over a ten-period planning horizon, the Woodstock interpreter automatically generates accounting variables (O_1, O_2, O_3,... O_{10}) and accounting constraints to sum the output production in each period. In addition, nine constraint rows will be generated ($O_2 - O_1 = 0$, $O_3 - O_2 = 0$,... $O_{10} - O_9 = 0$) to force the output level to remain constant over the planning horizon. Examples of flow constraints include:

- a non-declining yield (NDY) of timber harvest volume over the planning horizon from the entire forest,
- a requirement that forage production in a given period must be within 10% of the production in the previous planning period,
- a requirement that the area precommercially thinned in a given period must not differ by more than 100 acres from the area thinned in the previous period,
- a requirement that the amount of old-growth timber inventory may decrease by at most 15% per period.

Woodstock provides the following constraint functions as a convenience to you. Note that all of these constraints can be formulated using the regular constraint syntax as well.

Category	Equality	Lower limit	Upper limit	Function
NDY(*x*)		÷		Requires that the output *x* not decline from one period to the next
EVEN(*x*)	÷			Requires that the output *x* not decline from one period to the next
SEQ(*x*,*y*,*z*)		÷	÷	Requires that the output *x* increase by no more than *y*% and decrease by no more than *z*% from one period to the next
STEP(*x*,*y*,*z*)		÷	÷	Requires that the output *x* increase by no more than *y* units and decrease by no more than *z* units from one period to the next

With the `STEP` function, if the output level in the first period is zero, due to no eligible types for example, it is still possible to increase the output level in the next period. If the `SEQ` function is used instead, and the initial period output level is zero, an increase in the next period is not possible because any percentage of zero is still zero.

General relational constraints are even more flexible than flow constraints because they can reference not only other time periods, but also other outputs. In addition, ten constraint rows will be generated ($O_1 - 0.3P_1 = 0$, $O_2 - 0.3P_2 = 0$, …, $\boldsymbol{O}_{10} - 0.3P_{10} = 0$) to force the output level *O* to be equal to 30% or output level *P*. Some examples of general relational constraints include:

- a requirement that for every 100 acres of land allocated to timber production, 5 acres must be allocated to recreation or wildlife habitat production,
- a requirement that the amount of winter forage produced each period must be within 20% of the amount of summer forage produced each period.

Objective functions

An *objective function* is very much like a constraint row but the right hand side of the equation is not a fixed quantity. Instead, the LP solver tries to make the right hand side quantity as large as possible (*maximization*) or as small as possible (*minimization*).

The Woodstock syntax allows you to define any number of regular objective function rows, but the LP solver can only optimize one objective function at a time. The Woodstock interpreter will generate row names for each of the objective functions you specify, but you must provide the specific row name to the LP solver.

Woodstock provides four types of objective function specifications, plus the goal programming formulation:

Category	Function
_MAX *x*	Simple maximization of the output *x*
_MIN *x*	Simple minimization of the output *x*
_MAXMIN *x*	Maximization of the minimum of output *x*
_MINMAX *x*	Minimization of the maximum of output *x*
_GOAL(g_1,g_2)	Minimize deviations from stated goals g_1 and g_2

`_MAXMIN` might be used when you know that an output is scarce and you want the shortfall to be as small as possible (maximize the minimum). Corresponding uses can be found for `_MINMAX`.

Goal programming is best used when maximizing a single objective function is less important than achieving a set of goals. You may even rank achievement of goals differently by applying different weights to the deviations. For example, it may be twice as important to meet goal 2 than goal 1, and so deviations from goal 2 will be penalized twice that of goal 1.

To properly account for outputs contributing to the objective function, you must tell the Woodstock interpreter which periods to include in the objective function. For example, suppose you have a nondeclining yield policy and you want to determine the optimal harvest level from the forest. Will an objective function that maximizes the harvest volume in the first period yield the same solution as an objective function that maximizes the harvest in all? Probably not. Although the nondeclining yield constraints will not let the volume decrease from period to period, it is possible that the second solution could start with a lower initial harvest than the first solution but end up with a higher overall harvest.

Chapter 3: Inventory projection

The purpose of this chapter is to provide a brief overview of inventory projection with a focus on those aspects that must be understood if you are going to use Woodstock effectively. Along with the basis of inventory projection and its advantages, we will also consider some of the limitations inherent to the technique.

Concept

Basic *inventory projection models* have simple structures: a forest is represented as an array of classes, each class associated with a particular set of yield curves and an age class. With each iteration, classes are ordered for harvest according to one or more selection rules. Classes are harvested sequentially, starting with those at the top of the list, until the target level of treatment is reached. Harvested classes are assigned to new classes with age set to zero before the age counter is incremented by one planning period. Then, the classes are resorted if necessary and the harvest procedure begins again. Figure 3.1 shows a simple "oldest first" harvest rule used in a uniform forest type.

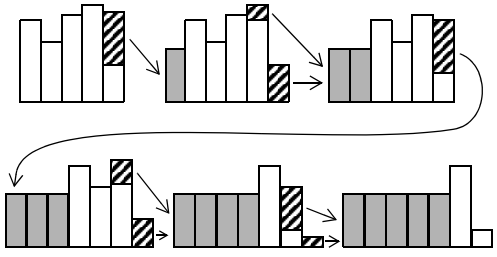
More complicated inventory projection models allow you to model silvicultural treatments such as planting and thinning, they allow forest class area to be assigned to more than one forest class following harvest, and they allow regeneration ages greater than zero.

Inventory projection models require three things:

- you must specify the sort criteria for choosing classes,
- you must specify the amount of area to harvest or treat silviculturally, and
- you must specify which actions to do first if there are multiple actions to be performed.

Inventory projection models typically employ *volume control* rather than *area control*, but you may use either. Both methods will eventually lead to the classical normal forest but volume control will usually take longer than the one rotation required for area control. Producing a normal forest is no longer the overriding

objective of forest management in most jurisdictions, but some agencies still adhere to area control as a means of regulation.



Graphical representation of an oldest-first, inventory projection model. The model has just one forest type, and uses an oldest-first harvest rule and area control. Blank bars indicate standing timber, dark bars indicate acres harvested and light bars indicate regenerated classes. Arrows indicate the transfer of acres from an existing class to a new regeneration class.

Example

Let us consider a simple 1,000 *ha* forest composed solely of lodgepole pine. A survey of the forest indicates 4 age classes, with 100 *ha* in the 81-85 year class, 300 *ha* in the 76-80 year class, 300 *ha* in 51-55 year class and 300 *ha* in the 16-20 year class. We will assume that lodgepole pine will regenerate fully following site preparation. The yield table for lodgepole pine (m^3/ha) is given in the following table.

Age (*periods*)	1	2	3	4	5	6	7
Volume (m^3/ha)	0	0	2	6	10	25	45
Age (*periods*)	8	9	10	11	12	13	14
Volume (m^3/ha)	60	90	125	170	200	225	230
Age (*periods*)	15	16	17	18	19	20	21
Volume (m^3/ha)	225	220	215	205	195	180	155

To estimate the allowable cut for this forest, we will implement volume control using an oldest first rule with a 20 period planning horizon. First, we queue up the forest classes in descending order of age:

Age (*periods*)	**Area (*ha*)**		**Volume (m^3)**
17	100	100 · 215 =	21,500
16	300	300 · 220 =	66,000
11	300	300 · 170 =	51,000
4	300	300 · 6 =	1,800
			140,300

To begin, let us attempt a periodic harvest of 25,000 m^3. In the first iteration, we will liquidate the oldest age class and almost 16 *ha* from the second oldest age class; the forest will look as follows five years later:

Age (*periods*)	**Area (*ha*)**		**Volume (m^3)**
17	284.1	284.1 · 215 =	61,079.35
12	300	300.0 · 200 =	60,000.00
5	300	300.0 · 10 =	3,000.00
1	115.9	115.9 · 0 =	0.00
			124,079.35

In the second and third iterations, we can obtain the entire 25,000 m^3 from the current oldest age class. At the end of the third iteration, the forest looks like this:

Age (*periods*)	**Area (*ha*)**		**Volume (m^3)**
19	45.9	45.9 · 195 =	8,950.50
14	300	300 · 230 =	69,000.00
7	300	300 · 45 =	13,500.00
3	115.91	115.91 · 2 =	231.82
2	116.28	116.28 · 0 =	0.00
1	121.95	121.95 · 0 =	0.00
			91,682.32

We can continue to harvest 25,000 m^3 from the oldest age classes until after the eighth iteration the forest structure is as follows:

Age (*periods*)	**Area (*ha*)**		**Volume (m^3)**
7	68.53	68.53 · 45 =	3,083.85
6	121.95	121.95 · 25 =	3,048.75
5	115.67	115.67 · 10 =	1,156.70
4	111.11	111.11 · 6 =	666.66
3	113.64	113.64 · 2 =	227.28
2	196.08	196.08 · 0 =	0.00
1	273.02	273.02 · 0 =	0.00
			8,183.24

At this point there is insufficient standing inventory to meet the 25,000 m^3 target for the ninth iteration, even if the entire forest is liquidated. Therefore the sustainable harvest level is less than 25,000 m^3 per period.

Binary Search

Formulating the previous example as a Woodstock model takes just a few minutes and processing the model takes little more than a second or two. Therefore, we could easily try lower harvest levels until we found one that was sustainable for 20 periods. By systematically raising or lowering the target harvest level, we could narrow in on the maximum rather quickly.

For example, if we run the simulation again with a target harvest level of 12,500 m^3 we find that the harvest is sustainable for 20 periods. Since we know that 25,000 is too high and 12,500 might be too low, we can try the average of the two (18,750). If the run succeeds, we know that 18,750 is too low, and we try the average of 25,000 and 18,750. If the run fails however, we know that 18,750 is too high, and we try the average of 12,500 and 18,750. By repetitively testing the average of two endpoints, we quickly converge on the optimum solution in what is called a *binary search* algorithm.

In a maximization binary search, the decision to be made is to raise the output level from the last successful attempt or lower the output level

from the last unsuccessful attempt. In a minimization problem, the roles are reversed: unsuccessful attempts cause the output level to increase and successful attempts cause the output level to decrease. In the example presented earlier, a binary search revealed that the maximum sustainable harvest for 20 periods was 18,218.2 m^3 per period ($\pm 1 m^3$).

To illustrate the usefulness of binary search, suppose that planting lodgepole pine could reduce the regeneration lag by 1 period. However, the amount of planting stock available is limited and only 50 *ha* can be planted in any one period. What impact does this have on the allowable cut?

First, the plantation yield curve will need to be developed. Reducing the regeneration lag by one period is equivalent to shifting the yield curves left by one period as shown:

Age (periods)	1	2	3	4	5	6	7
Volume (m^3/*ha*)	0	2	6	10	25	45	60
Age (Periods)	8	9	10	11	12	13	14
Volume (m^3/*ha*)	90	125	170	200	225	230	225
Age (Periods)	15	16	17	18	19	20	21
Volume	220	215	205	195	180	155	125

Using binary search, we can determine that the maximum sustainable harvest has increased to 18,989.9 m^3, a difference of 4.2%. Simply by implementing a change in the silvicultural regime (planting), we have immediately increased the allowable cut. This immediate increase arising from silvicultural inputs is commonly called the *allowable cut effect*.

Advantages of inventory projection

- Inventory projection models are able to portray inventory in greater detail than LP models. Whereas a linear programming model can become extremely large when the planning horizon becomes long and/or the number of classes becomes large, a binary search model is relatively unaffected by them.
- It is usually easier to find feasible solutions using inventory projection (particularly with binary search) than with linear programming.

Disadvantages of inventory projection

- Although it is relatively easy to determine a feasible solution to a straightforward management problem with binary search, it may be far from optimal. Moreover, if there are numerous constraints or trade-offs to be made, a satisfactory feasible solution may be very difficult to find.
- Selection rules for harvesting and silviculture can be difficult to choose. Deciding whether to implement one treatment before another and deciding how much of each treatment to implement can be difficult, yet inventory projection models require the analyst to make these decisions before running the simulation.

Chapter 4: Monte Carlo simulation

The purpose of this chapter is to provide a brief overview of stochastic simulation, specifically the Monte Carlo simulation models that can be constructed using Woodstock.

Concept

Most forest models used for harvest scheduling and wood supply analyses are deterministic – the underlying assumption is that all relationships are understood perfectly and that the outcomes of various actions are known with certainty.

For example:

Based on a random sample of regeneration plots across the forest, we may find that a particular forest type regenerates to one of three types following harvest:

- 40% of the clearcuts regenerated to stand type *A*,
- 50% of the clearcuts regenerated to stand type *B*,
- the remaining 10% regenerated to stand type *C*.

But what does that information tell you?

Without going into a discussion of sampling statistics, one could view the sample information in one of two ways. We could assume the proportions are essentially forest-wide constants and that for every 100 acres clearcut, the regeneration response will be 40 acres to type *A*, 50 acres to type *B* and 10 acres to type *C*. This view works well in a deterministic model because the same proportion is assigned to every harvest of that forest type in every period. But what if there is some variation in the proportions, either across the forest or over time?

An alternative way to view the sample information is to look at them as probabilities: over the entire forest, individual clearcuts have a 40% probability of regenerating to type *A*, a 50% probability of regenerating to type *B* and a 10% probability of regenerating to type *C*. Given a large number of observations, the expected number of acres of type *A* following clearcut using the probabilistic approach will be the same as that calculated deterministically. Although the expected outcome calculated for a large number of harvest events is the same as the deterministic model, the outcome for any particular event is stochastic (i.e. random) and may be very different.

Probability and its implications

Let us consider the roll of a die. Assuming we are not using loaded dice, what is the probability of rolling a 2? Since there are six sides, and the probability of any side turning up is the same, the probability of rolling a 2 is 1/6. Does this mean that we will roll a 2 once every sixth roll of the die? No, of course not. On average, a 2 will turn up 1 time in 6, but for any given six rolls of the die, a 2 may not turn up at all or it may turn up several times.

Series	Numbers rolled	Response rate
(running average)		
1	1 1 6 2 2 5	2/6 (1/3)
2	2 1 3 3 1 3	1/6 (1/4)
3	1 6 1 2 6 3	1/6 (2/9)
4	1 2 3 2 5 2	3/6 (7/24)
5	3 1 6 2 5 6	1/6 (4/15)
6	3 6 5 1 1 1	0/6 (2/9)
7	4 1 4 1 5 4	0/6 (4/21)
8	5 5 4 2 5 4	1/6 (9/48)
9	6 4 6 3 5 2	1/6 (5/27)
10	4 6 6 1 6 3	0/6 (1/6)

In the table, notice that the number of times a 2 is rolled in a series varies from a low of zero to as high as three, but one in six is the most common. By the tenth series, the response rate equals the expected value (1/6) for the first time.

In a similar fashion, even though our sampling efforts indicated a 40% probability that stand type *A* would regenerate following harvest, the actual regeneration outcome for a particular harvest may in fact be type *B* or *C*. However, on average we would expect 40% of the harvested areas to regenerate to type *A*.

Suppose that stand type *A* represented a valuable species of tree and stand type *B* represented a less valuable species. In a deterministic model, the regeneration response is constant and the value of timber regenerating would be a weighted average of the two types. However, in a stochastic model, we may find that in one period 60% of harvested stands regenerated to type *A* but only 20% of harvested stands regenerated to type *A* in the following period. Although the average regeneration response was still 40% type *A*, the age class distribution is different from that predicted by the deterministic model. Depending on the particular circumstances, the harvest schedule may be very sensitive to variations in regeneration response but a deterministic model would be unable to demonstrate this sensitivity.

Examples of stochastic processes

To illustrate how stochastic elements can be incorporated into harvest scheduling models, we will examine two scenarios where probabilistic outcomes are explicitly recognized.

Stochastic actions

Suppose you are the manager of a forest products marketing board, and you wish to know how much wood is available from the non-industrial private forest (NIPF) land in your regulation area. A recently completed landowner survey has indicated that woodlot owners with properties under 100 acres in size have a 20% probability of harvesting, whereas owners of woodlots 100 - 250 acres in size have a 60% probability of harvesting and owners of woodlots larger than 250 acres have a 90% probability of harvesting. The Department of Natural Resources has provided you with inventory information gathered from aerial photography and ground surveys, so you know that 30% of your land base is comprised of small woodlots averaging 60 acres in size, 40% of your land base is comprised of medium-sized woodlots averaging 130 acres in size and the remaining land base is composed of large woodlots averaging 300 acres in size. How do you represent the landowner decision to harvest as a stochastic process?

After talking to some harvest contractors, you determine that most small landowners liquidate their holdings, whereas the larger landowners sell only a portion of their holdings, no more than 50%. Considering this information, you decide to model the harvest decision as a binary choice Monte Carlo process.

First, you merge all the separate stands into classes composed of the same cover type and age, for each of the small, medium and large holdings. Each class is then subdivided into 60 acre portions for the small holdings, 65 acre portions for the medium-sized holdings and 150 acre portions for the large holdings. You decide to use inventory projection as the method for estimating the sustainable cut level, using a "most valuable first" harvest rule and nondeclining yield.

Age	Area	Probability of MC event	Volume harvested
16	60	0.2 < (0.475)	0.00
17	60	0.2 < (0.362)	0.00
17	60	0.2 > (0.108)	12,900.00
17	65	0.6 > (0.274)	13,975.00
17	65	0.6 < (0.887)	0.00
16	150	0.9 > (0.550)	30,000.00

The entries in the table represent 3 small, 2 medium, and 1 large woodlot portions. The first Monte Carlo event draws a random number that

is larger than the probability value, so no harvest occurs. When the random number is less than the probability, the area is harvested.

For each portion of a class, you draw a random number between 0 and 1 to determine if the portion is harvested or not, hence the binary choice. The number of random numbers to be drawn depends on the size of a class and the portion size. For example, if there are 1,000 acres and you use a 50 acre portion size, you will need to draw 20 random numbers (1,000 / 50). With a large number of Monte Carlo events, the total area harvested should approach the expected value (total area in class times its probability), whereas for few events the area harvested may be significantly different from the expected value. Note also that you will not get the exactly the same results if you were to start over (unless your random number generator uses the same seed number as a starting point each time it runs).

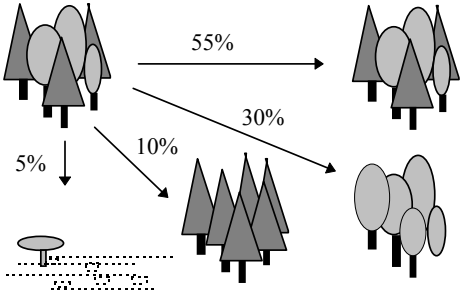
By running the inventory projection many times, you could determine what the highest sustainable harvest level is by increasing the cut until you cannot complete 100 simulations without failing to meet your target harvest level in one period. If you are less risk averse, you may want to increase the harvest level until you no longer meet the target harvest level at least 90% of the time.

You could, of course, use a similar approach to model natural events such as forest fire, windthrow and insect attack. In these cases, the action may destroy the stand, shift its composition or reduce the quality of standing wood. Furthermore, you may prefer not to rank stands according to some criterion as you did with the harvest rule, but instead have the selection of portions done randomly.

Stochastic transitions

You have developed an inventory projection model that has quite detailed inventory information and a rather sophisticated transition matrix for stands following clearcut harvest. You developed the transition matrix based on information gathered from regeneration surveys 10 years after harvest. Every forest type regenerates primarily to the cover type that was harvested, but some stands regenerate to one or more other cover types as well. The proportions vary of course, with the most variability observed in mixed wood classes. Since your inventory projection model is deterministic, these proportions are fixed for all planning periods and all acres harvested. However, because state regulations require you to meet

biodiversity targets for major cover types, you want to know the potential impact the variability in regeneration response might have.



Transition responses of mixedwood types following clearcut.

Since the average harvest block size your company uses is 150 acres, you decide to set up a Monte Carlo event for each 150 acre portion of a class that is harvested. The regeneration outcome for each portion depends on the proportions taken from your deterministic inventory projection model and a random number between 0 and 1. Note that this structure is different from the previous example since the choice is not binary – instead, one of several outcomes is possible. The random number drawn will determine the regeneration response as follows:

Class	Regeneration proportion	Range for random number
mixedwood	55%	0.000 - 0.550
hardwood	30%	0.551 - 0.850
softwood	10%	0.851 - 0.950
unstocked	5%	0.951 - 1.000

Just like the binary choice example, the number of random numbers to be drawn depends on the size of the class being harvested and the portion size chosen. If there are a large number of Monte Carlo events, the overall regeneration response should be close to the expected value predicted using the proportions but with few events the outcomes might be quite different.

By varying the portion size, you increase or decrease the noise factor associated with regeneration outcome. For example, suppose you have 500 acres of a class to be harvested. If you use a 500 acre portion size, all the area will be assigned to a single forest type following harvest. With a 50 acre portion size, you will more closely approximate the expected distribution, but it will still not be exact (5% of 500 acres equals 25 acres, but the minimum area allocated is 50 acres). With a 5 acre portion size, it is possible to fit the expected distribution exactly, but because of random variations you may not actually do so.

Conclusion

Advantages of Monte Carlo simulation

- Unlike deterministic models that ignore uncertainty, Monte Carlo simulations allow you to explicitly recognize uncertainty through random events.

1 Catastrophic natural events can be more realistically modeled than is possible using deterministic approaches.

2 Since each run is not the same, a series of simulations can be used to assess the risks associated with uncertainty in actions or outcomes.

Disadvantages of Monte Carlo simulation

- Monte Carlo simulations can be slow to execute. Because of the large number of calculations to be performed, these models take significantly longer to run than a deterministic counterpart. Furthermore, the execution time is sensitive to both the number of stochastic actions and the number of Monte Carlo events to be simulated. A complex model can require millions of calculations in a single iteration.
- Complex Monte Carlo simulations may be difficult to debug. For example, you may have defined a series of stochastic actions that follow on one another. Although each action appears to adequately reflect real processes, in combination they may yield incomprehensible results. It can be difficult to determine which of the linkages is causing the problem.
- Monte Carlo simulations may very well require more accurate data than a comparable deterministic model. This is simply a programming truism: the more a calculation is repeated, the more potential there is for round-off errors. Thus, a small error in a probability distribution can be compounded into a large error.

In general, Monte Carlo simulation models are best suited to risk analyses and forest planning problems where the uncertainty associated with important actions or outcomes must be explicitly recognized.

Chapter 5: An overview of linear programming

The purpose of this chapter is to provide a brief summary of linear programming (LP) and integer programming (IP) with a focus on those aspects that must be understood if you are going to formulate Woodstock LP models efficiently. We will not be dealing with concepts such as duality or specially refined solution techniques. If you are interested in learning more about these topics, consult one of the many texts available on mathematical programming.

Concept

LP and IP are members of a group of techniques known as mathematical programming, that also includes goal programming, mixed-integer programming, quadratic programming and other nonlinear programming methods. All of these techniques are considered optimization models in that they are designed to select an optimal alternative from the set of possible or feasible alternatives. An optimal solution is one that yields the minimum or maximum value of some numerically measurable performance criterion.

Each of these models is composed of a set of mathematical relationships that are functions of activities that comprise the alternatives. These relationships describe the criterion of optimality (the objective function) and the set of feasible alternatives (constraints due to limitations of the system being modeled). In a linear programming model, all mathematical relationships are linear functions. In an integer model, the variables are required to take on integer values only. If some of the variables are allowed to be continuous variables and others are integer, the problem is considered to be a mixed-integer programming (MIP) problem. However, the underlying principles for solving IP and MIP problems are the same, and many authors use the terms interchangeably.

Review of coordinate systems for graphics and plotting

To understand even a simple LP problem, certain fundamental concepts about coordinate systems and plotting of equations must be understood. If you are comfortable with this

material, feel free to just skim through this part quickly or just skip ahead to Chapter 6, "A simple Woodstock model".

Coordinate systems

You are probably familiar with an *x*-*y* coordinate system, but to review the concept, the purpose of this system is to assign an address to each point in a plane containing both an *x*-axis and *y*-axis. By convention, the *x*-axis is usually drawn as a horizontal line and the *y*-axis is drawn vertically. Furthermore, since it does not matter what the axes are named, we will change the names of the *x*-axis and *y*-axis to the x_1-axis and the x_2-axis (see Figure 5.1). Each point on the plane will be located by defining the distance from the intersection of the x_1 and x_2 axes (the origin) that by convention has the values $x_1 = 0$ and $x_2 = 0$.

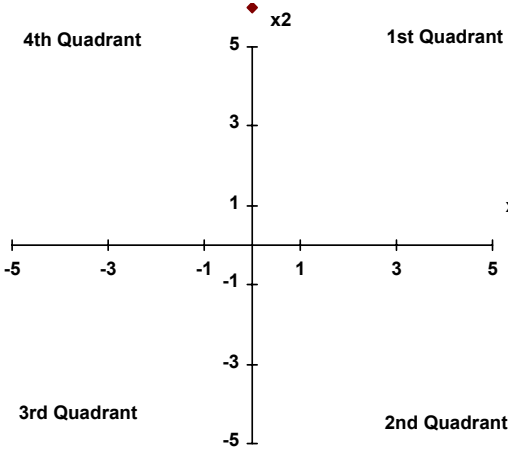


Figure 5.1. An x_1 - x_2 coordinate system.

Plotting single points or points corresponding to constant values for x_1 or x_2.

To locate a point on the plane, simply specify where along the x_1 axis and the x_2 axis it is located. For example, a point 3 units to the right of the origin along the x_1 axis and 2 points below the origin along the x_2 axis would have coordinates (3,-2). See Figure 5.2.

Now consider locating several points at once. For example, how would we locate all the points with an x_2 value equal to 4? Given that any two points can define a line, the simplest way to represent the solution graphically is to locate the points (2,4) and (-2,4) and using a straight-edge, draw a line that intersects both points.

This line is represented algebraically as $x_2 = 4$ and is illustrated in Figure 5.2 as well.

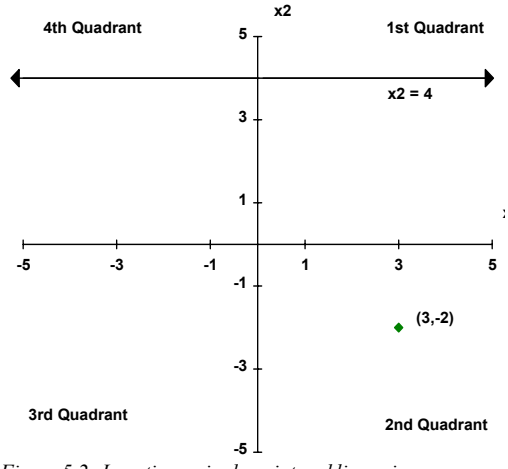


Figure 5.2. Locating a single point and line using x_1 - x_2 coordinates.

The equation $x_2 = 4$ can also be written as $0x_1 + 1x_2 = 4$ which is a linear equation. What if the coefficients for x_1 and x_2 are not 0 or 1? Consider the equation $2x_1 + 3x_2 = 6$. Again, we can plot a linear equation by drawing a line through two coordinates, but which two? To find two coordinates, simply set x_1 to the value of zero and solve for x_2 and then do the same for x_1 by setting x_2 to zero: (0,2) and (3,0). The plotted line is shown in Figure 5.3.

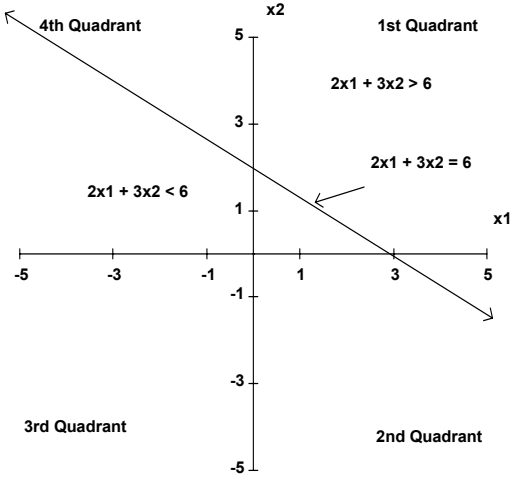


Figure 5.3. Plotting a linear equation on the x_1 - x_2 coordinate system using continuous variables.

Figure 5.3 illustrates an important point: you can represent an inequality graphically by plotting the line representing the corresponding equation and shading one side of the line. To determine which side of the line represents the region defined by the inequality, simply choose any point and substitute the values into the inequality. If the inequality holds, you have selected the correct side of the line, otherwise, choose the opposite side of the line. For instance, converting the equation from Figure 5.3 to an inequality $2x_1 + 3x_2 < 6$, we can determine that the region below the line $2x_1 +$

$3x_2 = 6$ is the correct region by testing the origin (0,0). Since $2 \cdot 0 + 3 \cdot 0 = 0$ and $0 < 6$, we know we are correct.

Suppose that we have defined the decision variable x_1 to be an integer variable (i.e., it may take on only integer values). What would the feasible region corresponding to $2x_1 + 3x_2 < 6$ look like now? Because x_1 cannot take on fractional values, we no longer have a continuous feasible region that lies below the line $2x_1 + 3x_2 = 6$. Instead, we have discrete areas of feasibility that correspond to each of the integer values of x_1.

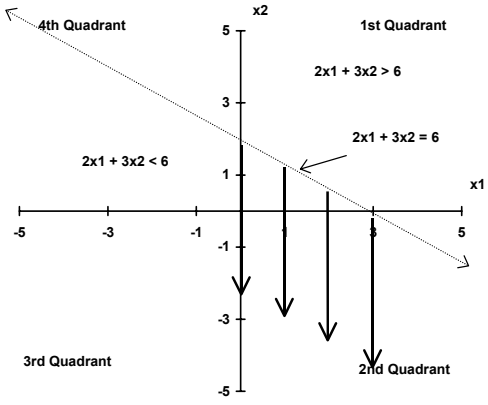


Figure 5.4. Plotting a linear equation on the x_1 - x_2 coordinate system using integer variables.

One of the basic assumptions of linear programming is that all decision variables must be non-negative. In other words, all variables must be greater than or equal to zero. Graphically, this restriction means that we only need to deal with the first quadrant, since the other quadrants by definition require that either x_1 or x_2 take on negative values.

In this next example, we will respect the non-negativity restrictions of LP and plot the set of points that satisfy 2 inequalities: $3x_1 + 2x_2 < 18$ and $x_1 + 6x_2 < 9$. First, we plot the corresponding equalities by finding intercepts as before. For the first equation, the intercepts are (6,0) and (0,9) and for the second equation, the intercepts are (9,0) and (0,1.5). To determine which side of the line is the region defined by the inequality, we test the point (0,0); both solve to zero and since $0 < 18$ and $0 < 9$, the regions defined by both inequalities lie below the lines as illustrated in Figure 5.5. If x1 is an integer variable, the feasible region is defined by the dotted lines corresponding to integer values of x_1: 1, 2, 3, 4, 5 and 6. Note that the integer feasible region is bounded by the feasible region of the LP formulation.

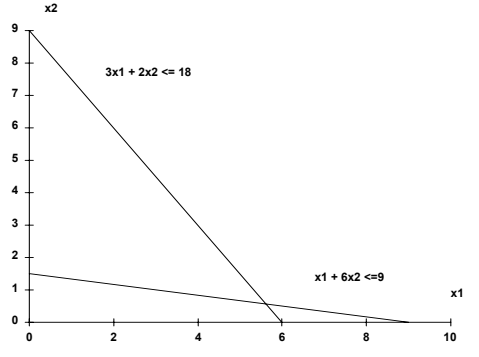


Figure 5.5. Locating the feasible region for two inequalities.

Assumptions of linear programming

All LP are implicitly assumed to be linear, divisible, non negative, independent and deterministic.

The *linearity* assumption means that all objective functions and constraints must be strictly linear over the entire range of each activity (i.e., doubling the area harvested on a development type yields double the output level and double the harvest cost).

The *divisibility* assumption means that actions or outputs can assume any real value including whole numbers and fractions. Therefore a solution calling for harvesting 1.283746 *ha* of a development type is just as acceptable as harvesting 100 ha of another type.

The *non negativity* assumption requires that actions or outputs assume positive values only. For the most part, this assumption causes few problems because negative harvest areas or negative volumes have no real world application anyway. However, costs are often treated as negative values and they need to be handled in a special manner so as not to violate the non negativity requirement.

The *independency* assumption is related to the linearity assumption, in that LP assumes that decision variables are completely independent of one another. In other words, choosing to harvest a particular forest type does not in any way affect the benefits of harvesting a different forest type. Of course, these kinds of effects occur in real forest planning all the time (edge-effects of habitat, visual quality).

The *deterministic* assumption means that outcomes are assumed to be known with certainty, and thus the same result will occur each and every time a particular action is carried out. Although you rarely understand forest dynamics to predict outcomes with absolute certainty, you can overcome some of this liability by testing various changes to your model and see what impact they have on the solution. This process is called *sensitivity analysis*.

Fundamentals of a LP model

In a Woodstock LP model, all of the actions you have defined represent alternatives that can be selected to achieve a particular objective. For example, if you have a development type consisting of spruce that is 7 decades old in the first planning period, and you have defined a clearcut action, the Woodstock interpreter can generate many alternatives for clearcut harvesting that spruce type: clearcut harvest spruce in period 1 at age 7, clearcut harvest spruce in period 2 at age 8 and so forth. Each combination of development type, age class, action and timing choice (period of implementation) is represented in a LP model as a *decision variable* defined as “the area allocated to prescription *x* in period *y*”.

If a decision variable is assigned a value greater than zero in an optimal solution, the prescription represented by the variable is assumed to take place and outputs will usually be generated. If you were interested in maximizing the amount of wood harvested from your forest, you would define an objective function that sums the outputs of interest. The Woodstock interpreter uses your output declarations to formulate the objective function as the sum of all decision variables that generate harvest volume, each multiplied by its appropriate yield coefficient.

Although the Woodstock interpreter can enumerate all possible prescriptions and generate decision variables for them, there have to be limits on the values the decision variables can assume. The Woodstock interpreter uses the information from the `AREAS` section to set bounds on the area assigned to prescriptions. For every age class in a development type, the associated decision variables must sum to the area of the class.

For example, if five decision variables are associated with a 5 ha class, the Woodstock interpreter would bound the area with a constraint equation similar to this: $x_1 + x_2 + x_3 + x_4 + x_5 = 5$. Each of the decision variables can assume values between 0 and 5, but the sum of all decision variables must be 5 because the constraint is an *equality constraint* and the right hand side (RHS) value equals 5. The Woodstock interpreter automatically generates all the constraints needed to ensure decision variables cannot take on impossible values.

Once all the decision variables have been enumerated and all the objective function and constraints have been generated, the forest problem has been formulated as a linear program.

To represent the problem in a concise format, LP problems are structured as a *matrix*:

Decision Variables as Columns		
Objective Function row	objective function coefficients	objective function value
Constraints as rows	constraint coefficients	equation RHS

Because each constraint is represented as a row in the matrix, and each decision variable as a column, you will often hear constraints referred to as rows and decision variables as columns. The total number of elements in a matrix is equal to the number of rows times the number of columns. However, many of the coefficients in the matrix will be zero because not every decision variable is included in every constraint. Those coefficients that are included in constraints represent the non-zero elements that number substantially fewer than the total number of elements but may still be numerous. The ratio of non-zero elements to the total number of elements is often reported as the matrix density. A sparse matrix is one where most elements are zero.

The steers and trees problem[1]

In the following example, we look at a simple land allocation problem that can be addressed using linear programming.

A part-time farmer has 24 acres of fallow land available and wants to use it to increase his income. He can either plant fast-gro hybrid Christmas tree transplants that mature in one year, or he can fatten steers by putting part of his acreage in pasture. The trees are planted and sold in lots of 1,000. It takes 1.5 acres to grow a lot of trees and 4.0 acres to fatten one steer. The farmer is busy and only has 200 hours per year to spend on this enterprise. Experience shows it takes 20 hours to cultivate, prune, harvest and package one lot of trees and 20 hours per steer are also required. He has $1,200 of operating budget available for the year and annual expenses are $30 per lot of trees and $240 per steer. He has already contracted to his neighbor for 2 steers. At current prices, Christmas trees will return a net revenue of $0.50 each and steers will return a net revenue of $1,000 each.

To formulate this problem as an LP problem, we need to determine decision variables, constraints and an objective function. First of all, what are the decisions to be made?

1. Davis, L.S. and Johnson, K.N. 1987. Forest Management. McGraw-Hill Book Company. New York. 790pp.

The farmer has two choices:

- fatten steers: x_1 = number of steers fattened per year.
- grow trees: x_2 = number of 1,000-tree lots grown per year.

The farmer's objective is to increase his income from the net revenues and an algebraic expression of this objective is:

$$\text{maximize } Z = 1000x_1 + 500x_2$$

where $1,000 is the net revenue per steer and $500 is the net revenue per tree lot (1,000 trees · 50¢ per tree). By convention, the objective function value is defined as *Z*.

The constraints on this problem are due to limited land, budget, labor, contracts and the standard non-negativity restrictions. The farmer has 24 acres available but 4 acres are required per steer and 1.5 acres are required per lot of trees. So, the land constraint is represented as follows:

$$4x_1 + 1.5x_2 \leq 24.$$

The total budget available for the year is $1,200 but the farmer will incur costs of $240 per steer and $30 per lot of trees; therefore, the budget constraints is formulated as:

$$240x_1 + 30x_2 \leq 1200.$$

The amount of labor the farmer is willing to invest is 200 hours annually, with each steer or lot of trees requiring 20 hours each. The labor constraint is:

$$20x_1 + 20x_2 \leq 200.$$

Because the farmer has already contracted to fatten 2 steers, this constraint is also required:

$x_1 \geq 2$.

The non-negativity constraints simply require $x_1 \geq 0$ and $x_2 \geq 0$.

To summarize, we can represent the steers and trees problem as a set of inequalities as follows:

Maximize

$$Z = 1000x_1 + 500x_2$$

subject to

$$4x_1 + 1.5x_2 \leq 24$$

$$240x_1 + 30x_2 \leq 1200$$

$$20x_1 + 20x_2 \leq 200$$

$$x_1 \geq 2$$

$$x_2 \geq 0$$

where

x_1 represents the number of steers, and

x_2 represents the number of tree lots.

The problem is represented graphically in Figure 5.6.

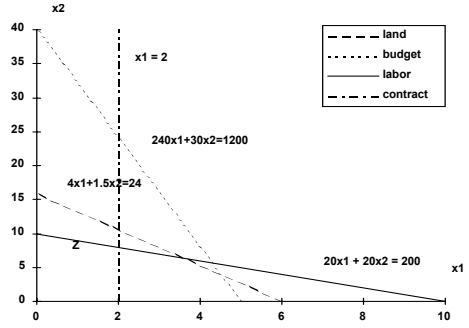


Figure 5.6. The steers and trees problem represented graphically.

To determine the optimal solution, we must find the point within the feasible region that yields the maximum value of *Z*. On the graph are two parallel lines drawn with dashes; these lines represent the objective function and each line represents a different combination of steers and trees. Any point along the line that is within the feasible region represents a possible combination of steers and trees and yields the same objective function value. The maximum objective function value is obtained by finding the point furthest from the origin but still within the feasible region. On the graph, this point is (3.6,6.4) which yields an objective function value of $6,800.

In reality, the farmer cannot produce 3.6 steers, but in many forestry problems the numbers are larger and you can round off to the nearest unit without too much loss in accuracy. For

example, if we had been dealing with a problem with quantities 10 times as large, the farmer would then produce 36 steers, 64 lots of trees and yield a net revenue of $68,000 with no fractional quantities at all. However, you cannot always just round off to the nearest whole number. This is where integer (or mixed-integer) programming comes to the fore.

Consider x_1 (the number of steers raised) and x_2 (the number of lots of trees) as integer variables. The feasible region then becomes a series of points bounded by the constraint lines (see Figure 5.7).

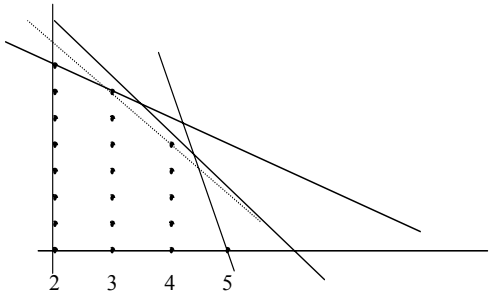


Figure 5.7. The integer steers and trees integer problem represented graphically.

If we solve the original LP, we derive an optimal solution of (3.6, 6.4). One might consider simply rounding the solution up or down to the nearest integer values (4,6), (4,7), (3,7), (3,6). Of these, only two are feasible, and the optimal solution is (3,7). The only way to be sure of the answer is to test a value and determine its feasibility, retaining that answer if it is better than the previous solution. For example, suppose, we start with x_1 equal to 4. If we round 6.4 up to 7, we see that the land constraint is violated:

$$4(4) + 1.5(7) = 26.5 .$$

Moreover, we can eliminate any further consideration of increasing x_2 because the land constraint will remain violated. If we round x_2 down to 6, the land constraint is still violated, and we need to round down again to 5. At (4,5), the solution is feasible with an objective function value of $6,500. Further reductions in x_2 will only yield lower objective function values and so we can eliminate further considerations of x_2 with $x_1 = 4$.

Now we can try setting $x_1 = 3$, and repeat the process. This time, rounding x_2 down to 6 yields a feasible answer with an objective function value of $6,000. Since this is lower than our current best feasible solution, we don't need to consider lowering x_2 any further. However, we need to test rounding x_2 up to 7 and this results in a feasible solution of (3,7) with an objective function value of $6,500. Testing (3,8) shows that further increases in x_2 are infeasible. At this point, we can stop because we know that none of the other combinations of (x_1, x_2) will yield better solutions, either because they are infeasible or they are guaranteed to have lower objective function values.

The process we used to find an integer solution is the basis for an algorithm called *branch and bound*. For each integer variable assigned a non-integer value, we need to test the effects of decreasing or increasing its value: this the

branching part of the algorithm. If in our testing we can show that further increases or decreases cannot be beneficial (the bounding part of the problem), we eliminate further consideration of that branch. Note that we only tested a small subset of the possible integer combinations of (x_1, x_2) and arrived at a solution very quickly. In general, the branch and bound algorithm is efficient and is used in virtually every commercial IP solver. Unfortunately, the number of branch and bound operations to be done grows exponentially with the size of the problem, and in the worst case, the branch and bound algorithm may reduce down to complete enumeration of all combinations. Hence, IP solvers are far more limited in the size of problems they can handle than a corresponding LP solver.

Infeasible solutions

The steers and trees problem has a single optimal solution that we found to be 3.6 steers and 6.4 lots of trees. Suppose the farmer had contracted to his neighbor to produce 6 steers instead of two. Now, there would be no feasible region as illustrated in Figure 8. We can verify this by substituting the value of 6 steers into the various constraints: 4(6) = 24, which does not exceed the land constraint; 20(6) = 120, which is within the labor allowance; 240(6) = 1,440, which exceeds the budget constraint. When one or more constraints cannot be met, the problem is deemed *infeasible*.

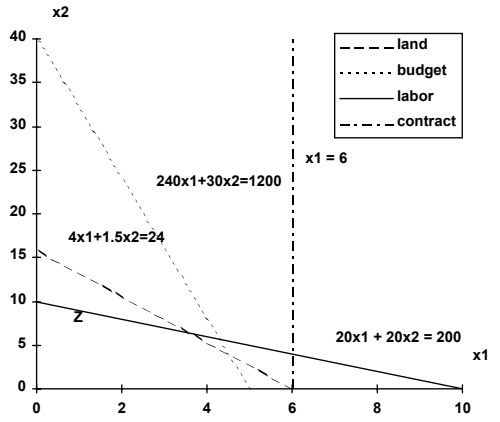


Figure 5.8. An infeasible LP with no defined feasible region.

Goal programming

With linear programming, one can only optimize a single objective function at a time; all other goals must be handled as constraints. However, the multiple goals may all be (nearly) equal in importance, and it would be ideal to simultaneously optimize all of them.

In goal programming, every goal is specified as an absolute constraint on an output. For example, you may have goals such as,

- harvest equal to 100,000 cunits per period,

- wildlife habitat at least equal to 5,000 acres per period,
- production of at least 500 recreational visitor days period,
- silviculture expenditures no greater than $250,000 per period.

Each of these goals can be specified algebraically as a constraint:

```
harv = 100000 1.._LENGTH
hab >= 5000 1.._LENGTH
rvd >= 500 1.._LENGTH
exp <= 250000 1.._LENGTH
```

To make these regular constraints into a goal programming formulation, we have to incorporate additional variables that measure deviations from the amount on the right hand side of the constraint expression. For the harvest goal, we need to add variables for over- and underachievement of the goal:

harv + under + over = 100 000

If the harvest is less than 100,000 the under variable will have to take on a positive value to take up the slack; if the harvest is over 100,000, then the over variable will take on a positive value to balance the equation. If the harvest is exactly 100,000 both over and under will equal zero.

Unlike the harvest goal, the other goals do not require a fixed value. If the solution produces more habitat or recreational visitor days than required, there is no penalty; similarly, there is no penalty for spending less than the allowable silvicultural budget. Thus, the algebraic formulation of these inequality constraints involves only under- or overachievement variables, but not both. To drive solutions to meet all the goals as closely as possible, we simply make the objective function the sum of these over and under variables and minimize the sum.

A goal programming formulation of steers and trees

Suppose that the farmer reviews our analysis and tells us that we did not quite capture the essence of his problem. For example, he had hoped to fully use the budget allocation because of certain tax advantages. Moreover, he would have preferred an even mix of steers and tree lots so that he could assign each of his children to take care of one steer and one lot of trees. He says that these goals were actually of more importance to him than simply maximizing profit. After some reflection, he tells us that the budget goal is twice as important to him as the goals for steer and tree production.

To formulate this problem as a GP problem, we need to determine decision variables, constraints and an objective function. All of the previously-defined decision variables are retained, but we need to add decision variables to represent under- and overachievement of the goals.

The first of farmer's goals is to produce 5 lots of trees:

$$x_2 - dto - dtu = 5$$

He also wishes to fatten five steers. Since his goal of 5 steers includes the two he has already contracted to produce, we can replace the contract constraint with the following goal for fattening steers:

$$x_1 - dso - dsu = 5$$

Finally, he would like to use the entire budget each year:

$$240x_1 + 3x_2 - dbo + dbu = 1200$$

Because the budget goals are twice as important to the farmer as the other goals, the objective function is as follows:

$$\text{MIN } dto + dtu + dso + dsu + 2dbo + 2dbu$$

The complete set of constraints for the GP steers and trees problem is:

$$x_1 - dso + dsu = 5 \text{ (steers goal)}$$

$$x_2 - dto + dtu = 5 \text{ (trees goal)}$$

$$240x_1 + 3x_2 - dbo + dbu = 1200 \text{ (budget goal)}$$

$$4x_1 + 1.5x_2 \leq 24 \text{ (land constraint)}$$

$$20x_1 + 20x_2 \leq 200 \text{ (labor constraint)}$$

When we solve this goal programming problem, the optimal solution is to produce 4.5 steers (x1) and 4 lots of trees, with an objective function value of 1.5. The solution under-achieves the steer goal by half of a steer and underachieves the tree goal by 1 lot of trees. However, the all of the budget and land are used; since the budget goal was weighted heavier than the other goals, it is not surprising that all the budget was used.

This example points out the two demanding requirements of goal programming on a decision maker:

- all the important goals must be known in advance and
- appropriate weights can be assigned the goal deviations that reflect the relative importance of goal achievement.

This is quite different from the usual case where the problem is to determine what the output levels should be; it may be quite difficult for the analyst to ascertain what the goals and the weights should be. While the idea of an optimal solution to the conflicting goals problem is appealing, goal programming may end up being more divisive than a well-executed bench-marking exercise.

Computer based solution methods

Obviously, the graphical method only works easily with two decision variables. Although problems with 3 decision variables could be solved using 3 dimensional graphing methods, problems with more than 3 decision variables must be solved through computational means. One algorithmic method of solving LP problems is called simplex method. Simplex is an efficient algorithm and it works for any size problem providing the problem is correctly formulated. Commercial computer programs for solving large LP problems use other

methods such as revised simplex method and interior point methods that are more amenable to computer computation and yield rapid solution times. However, the underlying principles for solving linear programs are the same.

We have seen that the feasible regions of IP and MIP problems are completely contained within the feasible region of their corresponding LP solution (where the integrality restrictions have been relaxed). Because of this property, a good starting point to finding an optimal solution to an IP or MIP problem is to solve the corresponding LP in the regular way. If the solution happens to be integral, then stop. Otherwise, each of the integer variables that have been assigned non-integral values need to be tested with integral values using an algorithm such as branch-and-bound. In cases where the number of integer variables is small, branch-and-bound can find an optimal solution very quickly. As the problem size increases, the number of branch-and-bound operations increases dramatically, and in the worst case the algorithm degenerates to complete enumeration. This is why large-scale IP (and MIP) problems are difficult to solve: enumerating even a fraction of all the possible combinations of integer values requires excessive amounts of time.

The Tom Pike problem

The steers and trees problem is a good example for illustrating the structure of a linear programming problem, but it represents a static allocation scenario: there is only one time period to consider and there are no changes in conditions over time. In a large forest, the system is continually undergoing change (growth and mortality, harvesting, fire and disease). To model this kind of problem, we need to extend the definition of decision variables.

Tom Pike owns a small sawmill and 500 ha of forest land. Tom employs a contractor who supplies part of the mill's stumpage requirements from harvests on Tom's 500 hectares. The 500 hectares consists of two different stand types, 100 ha of spruce-fir that is currently 35 years old with the remainder of the forest in hardwood that is 25 years old. Tom has used stand table projection methods to estimate the per hectare volumes that could be harvested from these stands, under alternative management regimes, over the next twenty years and has summarized this information in the following tables.

Projected yield streams for existing hardwood stands under two alternative management regimes. Under regime 1, the stand can be clearcut at age 25 or 30. Under regime 2, the stand is thinned at age 30 yielding 8 m^3/ha and is clearcut at age 40.

Age	**Volume (m^3/ha) under regime 1**	**Volume (m^3/ha) under regime 2**
25	19	19
30	28	20(8)
35	37	31

Age	Volume (m^3/ha) under regime 1	Volume (m^3/ha) under regime 2
40	42	49
45	45	64

Projected yield streams for existing spruce-fir stands under two alternative management regimes. Under regime 1, the stand can be clearcut at age 45 or 50. Under regime 2, the stand is thinned at age 35 yielding 28 m^3/ha and is clearcut at age 45.

Age	Volume (m^3/ha) under regime 1	Volume (m^3/ha) under regime 2
35	98	70(28)
40	106	87
45	122	108
50	135	131
55	145	152

Both stand types are expected to naturally regenerate to a mixed wood condition. However, cutovers can be planted to miracle spruce. In either case, regenerated stands are to be harvested at age 30.

Age	Volume (m^3/ha) natural regeneration	Volume (m^3/ha) plantation
5	2	4
10	12	18
15	26	34
20	38	50
25	52	68
30	74	84
35	98	116

Tom is interested in developing a harvest schedule for his 500 *ha* forest for the next 20 years using 5-year planning periods. He wishes to harvest as much wood as possible in the first planning period, and since his sawmill requires a relatively steady flow of wood, he wants to impose a non-declining yield constraint for the planning horizon. Tom also wishes to ensure that the ending inventory of the forest will be at least 60% of the initial inventory before harvesting commences.

Model I vs. Model II

In the Tom Pike problem, we are scheduling the harvest for four periods rather than just one. Furthermore, because the forest continues to grow with each planning period, the volume harvested from an acre of land is not the same in each planning period. To properly represent the choice to harvest in each period, we need to keep track not only what type of forest it is, but also the period in which it is harvested and the regeneration methods used following harvest.

There are two commonly used methods of representing decision variables in LP models, called Model I and Model II. In a *Model I*

formulation, we let x_{ijk} represent the number of acres harvested in period *i* using harvest method *j* and assigned to regeneration prescription *k*. Regeneration prescription *k* has a specific rotation length and a set of treatments that are repeated for the duration of the planning horizon. For example, suppose a stand is harvested in period 1 and assigned to a regeneration prescription of clearcut and natural regeneration with a rotation of 7 periods. In a 20 period planning horizon, the same acres harvested in the first period will be subsequently harvested and naturally regenerated in periods 8 and 15, and will be five periods old at the end of the planning horizon.

In a *Model II formulation*, we let x_{ijk} represent the number of acres born in period *i*, harvested in period *j* under prescription *k*. The prescription *k* may be the same type of treatment used in the Model I formulation, but it only holds for a single rotation. For example, using the same stand we used in the Model I example, say the stand is 10 periods of age when it is harvested in period 1. We might use a decision variable $x_{_911}$ that tells us that the stand was born 9 periods before the start of the planning horizon, and it was harvested in period one under prescription one. We will need a new decision variable to represent the regenerated stand that might be called x_{181} and that tells us the stand was born in period 1, harvested in period 8 (at age 7 periods) under prescription 1. To properly account for all the acres from the beginning of the planning horizon to the end, we need to subtract the acres in the regeneration stand from the initial stand, effectively transferring the acres from one class to another.

Both Model I and Model II have their advantages and disadvantages. Model I formulations necessarily have a larger number of decision variables to represent the range of possible treatment types and timing choices than does a Model II formulation. Conversely, because the Model II formulations need to transfer acres from one class to another, there are more constraint rows than in a Model I formulation. The Woodstock interpreter generates LP matrices using a Model II formulation to be consistent with its simulation modeling structures: the inventory projection and Monte Carlo simulation models also collapse forest classes after harvest.

Model II formulation: Tom Pike problem

Let us begin by defining the planning horizon. Since the yield tables are provided in 5-year increments, we can subdivide the 20-year planning horizon into 4 planning periods. We will also assume that harvests occur at the endpoint of each 5-year period.

Since we are using a Model II formulation, we need to define decision variables representing each of the harvest choices available to Tom. Let x_{ijk} be the number of hectares of stand type *x*, born in period *i*, harvested in period *j* under regime *k*. For example, if some of the spruce-fir was harvested under regime 2 in period 1, we

could represent that area with a variable called SF_{-612}. We will also define some *accounting variables* that will sum the quantities harvested plus the ending inventory: H_1 - H_4 will represent the harvests in periods 1 through 4 and *INV* will represent the inventory at the end of the planning horizon (which begins in period 5).

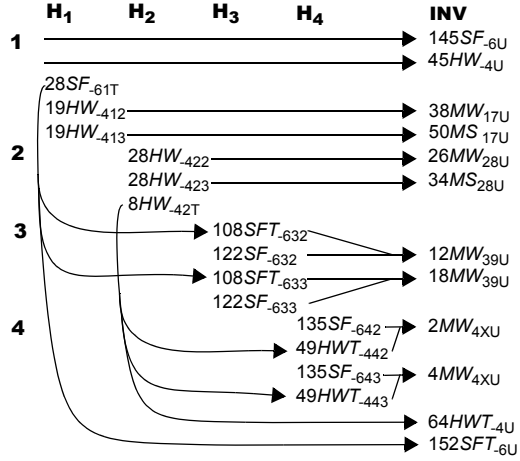


Figure 5.9. You can see that there are 14 decision variables representing the harvest decisions. There are an additional 12 variables (representing regeneration alternatives) that count toward the ending inventory. The final four decision variables in the INV column, bottom row, represent the choice to leave some hectares unharvested

With the information given in the table, it is now straightforward to construct the linear program. To keep things organized, we will deal with the acreage and transfer row constraints first, followed by the constraints explicitly laid out by Tom Pike. The first order of business, the objective function is trivial – because we defined a variable to sum the harvests of the first period, we simply maximize H_1.

MAX H_1

subject to

Nondeclining yield constraints ensure that the harvest in the current period is no less than the previous period.

$H_2 - H_1 \geq 0$

$H_3 - H_2 \geq 0$

$H_4 - H_3 \geq 0$

Ending inventory constraint ensures that ending inventory is a least 60% of initial inventory.

$INV >= 10440$

Initial area constraints add up the acres at the beginning of the planning horizon.

$SF_{-61T} + SF_{-632} + SF_{-633} + SF_{-642} + SF_{-643} + SF_{-6U} = 100$

$HW_{-42T} + HW_{-412} + HW_{-413} + HW_{-422} + HW_{-423} - HW_{-4U} = 400$

Acreage transfer constraints ensure that hectares thinned of final harvested are transferred from the original class to the new class properly. Obviously area regenerated must equal the area final harvested.

$SF_{\text{-}61T} - SFT_{\text{-}631} - SFT_{\text{-}633} - SFT_{\text{-}6U} = 0$

$HW_{\text{-}42T} - HWT_{\text{-}442} - HWT_{\text{-}443} - HWT_{\text{-}4U} = 0$

$HW_{\text{-}412} - MW_{17U} = 0$

$HW_{\text{-}413} - MS_{17U} = 0$

$HW_{\text{-}422} - MW_{28U} = 0$

$HW_{\text{-}423} - MS_{28U} = 0$

$SF_{\text{-}632} + SFT_{\text{-}632} - MW_{39U} = 0$

$SF_{\text{-}633} + SFT_{\text{-}633} - MS_{39U} = 0$

$SF_{\text{-}642} + HWT_{\text{-}442} - MW_{\text{-}4XU} = 0$

$SF_{\text{-}643} + SFT_{\text{-}443} - MS_{\text{-}4XU} = 0$

Harvest accounting constraints sum the volume harvested in each planning period.

$28SF_{\text{-}61T} + 19HW_{\text{-}412} + 19HW_{\text{-}413} - H_1 = 0$

$28HW_{\text{-}422} + 28HW_{\text{-}423} + 8HW_{\text{-}42T} - H_2 = 0$

$108SFT_{\text{-}632} + 122SF_{\text{-}631} + 108SFT_{\text{-}633} + 122SF_{\text{-}633} - H_3 = 0$

$135SF_{\text{-}642} + 49HWT_{\text{-}442} + 135SF_{\text{-}643} + 49HW_{\text{-}443} - H_4 = 0$

Inventory accounting constraints sum the volume left standing at the end of the planning horizon.

$38MW_{17U} + 50MS_{17U} + 152SFT_{\text{-}6U} + 64HWT_{\text{-}4U} + 26MW_{28U} + 34MS_{28U} + 12MW_{39U} + 18MS_{39U} + 145SF_{\text{-}6U} + 45HW_{\text{-}4U} + 2MW_{4XU} + 4MS_{4XU} - INV = 0$

Non-negativity constraint (implied) prevents decision variables from having negative values.

$X_{ijk} \geq 0$

Although the Tom Pike problem is admittedly small, it illustrates many of the same types of constraints that are present in a real-world problem. However, a real world problem can have tens of thousands of variables and hundreds or even thousands of constraints. To write a problem of that magnitude out in algebraic form as we have just done would require a sheet of paper the size of a football field or more. Luckily, Woodstock does most of this "grunt work" for you. However, even though you do not have to know exactly how Woodstock generates a LP matrix, you should have a basic understanding of where the information in a matrix file comes from so that you can find errors in logic when they occur.

Conclusion

Advantages of linear (integer) programming

- Linear programming models can simultaneously consider alternative prescriptions for the same parcel of land. Because alternatives are evaluated at the same time, it is possible for the model to make trade-offs among actions, outputs and planning periods.
- LP models can incorporate constraints on inventory, output and activity levels and product flows.

- LP solvers either produce an optimal solution (if the problem is feasible) or report that no solution is possible (infeasible).
- Integer programming problems can be used to make allocation choices among different alternatives optimally: for example, assigning an area of forest to either timber production OR dispersed recreation. Regular LP cannot guarantee such all-or-nothing solutions.

Disadvantages of linear (integer) programming

- Linear programming models grow rapidly in size when there are many inventory classes, many activities and timing choices and/or your model has a long planning horizon. It is quite possible to generate an LP matrix that is too large for your solver.
- Constraints and objective functions in a LP model must be linear functions of decision variables. However, many relationships found in natural systems are inherently nonlinear. It is possible to circumvent the linearity assumptions through linear transformations and piece-wise approximations of the nonlinear relationship, but there are certain nonlinear functions that simply cannot be addressed in a LP model yet they cause no problems at all in a simulation model.
- LP solutions can produce fractional quantities that are not practical. In some cases, you can round off such quantities to the nearest integer value, but in many cases, such rounded values are not feasible. LP models can be modified to include integer variables and such models are called mixed-integer programming (MIP) models. However, MIP models with large numbers of integer variables are exceedingly difficult to solve. Woodstock now supports limited use of integer variables in forest models. For details about the use of the integer variables, see the Optimize Section chapter in the Woodstock User Guide.
- Some users of LP models have an unfortunate tendency to accept optimal solutions as given, without examining the solution for potential errors or inconsistencies. A simulation model is less likely to suffer this problem because user-intervention is required throughout the analysis.

  In general, linear programming models are best suited to forest planning problems where there are relatively few inventory categories but there are unusual yield functions, many alternative harvest and silvicultural actions to consider and there are numerous constraints on outputs.

Chapter 6: A simple Woodstock model

The purpose of this chapter is to introduce you to the Woodstock modeling language and how to convert a forest management problem into a functioning Woodstock model.

The Tom Pike problem revisited

In Chapter 5 we were introduced to the Tom Pike problem, that we represented using an algebraic form of linear programming. What we will do now is build a Woodstock model that is functionally equivalent to the algebraic LP. Furthermore, our Woodstock model will also be amenable to inventory projection solution methods as well. First, let's restate the Tom Pike problem before we get down to the business of writing our Woodstock model.

Tom Pike owns a small sawmill and 500 *ha* of forest land. Tom employs a contractor who supplies part of the mill's stumpage requirements from harvests on Tom's 500 hectares. The 500 hectares consists of two different stand types, 100 *ha* of spruce-fir that is currently 35 years old with the remainder of the forest in hardwood that is 25 years old. Tom has used stand table projection methods to estimate the per hectare volumes that could be harvested from these stands, under alternative management regimes, over the next twenty years and has summarized this information in the following tables.

Projected yield streams for existing hardwood stands under two alternative management regimes. Under regime 1, the stand can be clearcut at age 25 or 30. Under regime 2, the stand is thinned at age 30 yielding 8 m^3/ha and is clearcut at age 40.

Age	**Volume (m^3/ha) under regime 1**	**Volume (m^3/ha) under regime 2**
25	19	19
30	28	20(8)
35	37	31
40	42	49
45	45	64

Projected yield streams for existing spruce-fir stands under two alternative management regimes. Under regime 1, the stand can be

clearcut at age 45 or 50. Under regime 2, the stand is thinned at age 35 yielding 28 *m³/ha* and is clearcut at age 45.

Age	Volume (m^3/ha) under regime 1	Volume (m^3/ha) under regime 2
35	98	70(28)
40	106	87
45	122	108
50	135	131
55	145	152

Both stand types are expected to naturally regenerate to a mixed wood condition. However, cutovers can be planted to miracle spruce. In either case, regenerated stands are to be harvested at age 30.

Age	Volume (m^3/ha) natural regeneration	Volume (m^3/ha) plantation
5	2	4
10	12	18
15	26	34
20	38	50
25	52	68
30	74	84
35	98	116

Tom is interested in developing a harvest schedule for his 500 ha forest for the next 20 years using 5-year planning periods. He wishes to harvest as much wood as possible in the first planning period, and since his sawmill requires a relatively steady flow of wood, he wants to impose a non-declining yield constraint for the planning horizon. Tom also wishes to ensure that the ending inventory of the forest will be at least 60% of the initial inventory before harvesting commences.

Modeling the Tom Pike problem

The Tom Pike problem can easily be modeled using Woodstock.

To model the Tom Pike problem using Woodstock:

1 Create a new model. See “Creating a new model” on page 61

2 Define landscape themes and attributes. See “Defining landscape themes and attributes” on page 61

3 Define planning horizon, lifespan and areas. See “Defining planning horizon, life span and areas” on page 62

4 Define actions and transitions. See “Defining actions and transitions” on page 64

5 Define yields and outputs. See “Defining yields and outputs” on page 65

6 Formulate a model. See “Formulating a model” on page 67

7 Specify reports. See “Specifying reports” on page 69

8 Run the model. See "Running the model" on page 69

Creating a new model

Start the Woodstock editor, and create a new forest model by selecting *File>New>New model*. When the dialog box opens, name your new model *tompike* and, in addition to the model sections already checked by default, select the *Queue* section by checking the appropriate box. The Woodstock editor will then create the files for editing.

Normally, we would use the standard multi-file structure for our forest models, but we want to see how all the parts of Woodstock model come together and interact. Therefore we will use a single-file model structure. To convert our model to the single-file format, select *Make a single file model* from the *Tools* menu.

Defining landscape themes and attributes

Usually, the first thing to do when building a Woodstock model is to come up with a land classification scheme that adequately represents both the current condition on the forest, as well as any possible future conditions.

To begin, we know there are only two kinds of forest present right now: spruce-fir and hardwood. However, both of these types could be thinned before final harvest and since there are differences in yield between the two, we need to differentiate them. Furthermore, we know that if any of the existing types is harvested and naturally regenerated, then a mixed-wood condition will result. On the other hand, if the harvested areas are planted, then the stand will be a miracle spruce plantation. Since the regeneration types are distinctly different from the existing types, a second theme to differentiate existing and regeneration types provides no additional information. Therefore, a single landscape theme and six thematic attributes will suffice. Click on the *Landscape* icon in the tree view to go to the Landscape section for editing. Type in the following information:

---

* *You will notice that the editor has already provided the *THEME keyword. This is because all Woodstock models must have at least one theme.*

```
LANDSCAPE
; Landscape
*THEME forest type
sf spruce-fir unthinned
hw hardwood unthinned
sft spruce-fir thinned
hwt hardwood thinned
mw mixed-wood
ms miracle spruce
```

You will see two upper-case keywords, `LANDSCAPE` and `*THEME`. The `LANDSCAPE` keyword tells the Woodstock interpreter that the Landscape section is beginning. The `*THEME` keyword marks the beginning of the first landscape theme. The label "forest type" that follows `*THEME` is there to remind us what the first theme represents, but it is optional and the Woodstock interpreter simply ignores this information. On each of the next six lines, you

will find a two or three-letter code followed by a longer description. The codes are used throughout our Woodstock model to differentiate the different forest types, but the longer descriptive names are again for our benefit, explaining what the codes mean.

Had we decided to use two landscape themes, we would have followed the last line of our example with another `*THEME` keyword and any thematic attributes belonging to that theme would be given on subsequent lines. The Woodstock interpreter keeps reading the input file sequentially until it encounters either another `*THEME` keyword, a new section keyword, or a syntax error.

So, what does the Landscape section do for us? How does the Woodstock interpreter know that there is some significance in the thematic attributes we have chosen? Actually, the interpreter does not associate any significance to our thematic attributes, but after reading the Landscape section it does know that a development type has only one thematic attribute, and it knows that a code like "`sp`" is not valid. In fact, much of the error checking capability of the Woodstock interpreter is based on the landscape theme specification.

Defining planning horizon, life span and areas

Now that we have a way to identify portions of the forest, we need to be able to keep track of time. Tom Pike has already decided to use 5-year periods and so we shall use them in our Woodstock model too. If we assume that the planning horizon starts immediately, then the first potential actions will occur immediately, since the stands are 25 or 35 years of age.

Obviously, this example problem is designed to work out easily, but what do you do if you have stand ages measured in years, but you are using 5 or 10 year periods? Obviously, you will have to average stand ages to fit within your classes. Furthermore, some stands may be harvested in your model at younger ages than predicted and others at ages older than predicted. The significance of this error is highly dependent on the distribution of ages within a class and the assumption of when actions occur in the period: the longer your period length, the more significant biases will be. To minimize potential biases, we are assuming that activities occur at the end of a period.

Now that we know the period length, we can specify both the planning horizon length and the area specification for the forest. Since the planning horizon is 20 years, we can represent this problem with a four-period model. Our existing stands of spruce-fir and hardwood will be 7 and 5 periods of age in the first period of the planning horizon (5 years hence). Click the *Control* icon in the tree view to edit the Control section.

The `CONTROL` keyword marks the beginning of the Control section that is used to tell the Woodstock interpreter not only how long the planning horizon is, but also whether to process

the model as a simulation or as a LP matrix generator. Delete all the lines within the Control section except the `*LENGTH` keyword and change it to 4 periods. We will be adding more keywords to the Control section later but for now, we will just specify the planning horizon length. The section should resemble the following:

```
CONTROL
; Control
*LENGTH 4 ; periods are 5 years
```

Woodstock also requires that you specify how old a development type may become before senescence or break-up is assumed to occur. In this model, the planning horizon is very short and the forest-types present will not grow old enough during the planning horizon for senescence to occur. However, a model with a long planning horizon that includes very old or short-lived species may very well encounter stand break-up in one or more of the planning periods. By forcing you to specify a maximum lifespan for each development type, the Woodstock interpreter is able to alert you that a development type has exceeded its expected life span. On the other hand, if you do not expect to have problems with senescence, a dummy life span specification such as the one presented here can be used. Click on the *Lifespan* icon in the tree view and enter the following information:

```
LIFESPAN
; Lifespan
? 20; maximum lifespan for all types = 100yrs
```

The `LIFESPAN` keyword marks the beginning of the Lifespan section and the global thematic attribute "`?`" indicates that all development types will be assigned the same maximum life span of 20 periods. Since the oldest development type will only be 11 periods of age at the end of the planning horizon, none of the development types will reach senescence during the planning horizon.

Next we must initialize the forest area and development type structure. Click on the *Areas* icon in the tree view to go to the Areas section and add the following text.

✻ *Notice that we do not have to specify areas for thinned or regenerated types. Woodstock will create these types for us later on.*

```
AREAS
; Areas
*A sf 7 100 ; 100 ha of 7-period spruce-fir
*A hw 5 400 ; 400 ha of 5 period hardwood
```

The `AREAS` keyword marks the beginning of the Areas section that tells the Woodstock interpreter the ages associated with specific development types. The `*A` keyword indicates that a new area declaration is beginning for the development type indicated by the thematic attribute(s) that follows. The first number indicates the age (in periods) of the analysis area and the second digit represents the number of hectares in the development type.

Defining actions and transitions

Now that we have described the existing forest, it is time to declare the activities that will occur during the planning horizon. Reading back over the problem, we can identify two harvest activities: a commercial thinning and a final clearcut harvest. In addition, there is an option to naturally regenerate clearcut stands or to plant them to miracle spruce. Although it would be possible to recognize the regeneration activities as separate actions in our model, we can construct a simpler model by combining them with the harvest activities. Click on the *Actions* icon in the tree view to go to the Actions section.

* *The indenting shown here is not required by the interpreter. We use it to make the input file more readable.*

```
ACTIONS
; Actions
*ACTION ccnat Y clearcut natural regenerate
 *OPERABLE ccnat
   sf  _AGE 9 _AGE 10 {age is in periods}
   sft _AGE = 9
   hw  _AGE 5 _AGE 6
   hwt _AGE = 8
 *ACTION ccplt Y clearcut & plant miracle
spruce
 *OPERABLE ccplt
   sf  _AGE 9 _AGE 10
   sft _AGE = 9
   hw  _AGE 5 _AGE 6
   hwt _AGE = 8
*ACTION thin N commercial thin
 *OPERABLE thin
   sf  _AGE = 7
   hw  _AGE = 6
 *PARTIAL thin
   wood
```

In our declaration of the thin action we specified an `N` following the action code to indicate that the action does not change the age of the type in the transition by default. In other words, the residual trees after the thin are considered to be the same age as the trees were before the thin. However, in the definitions of `ccnat` and `ccplt` we specified a `Y` indicating that the action does change the age of the type in the transition.

We also specified the `*PARTIAL` keyword for the thin action and specified a code `wood` on the following line. When the Woodstock interpreter calculates any outputs associated with the thin action and the yield component `wood`, it will estimate the yield as the difference between the standing volume before the thin and the standing volume afterward.

The definitions of `ccnat` and `ccplt` are identical except for the action codes themselves. This makes sense because the harvest action is determined by the standing timber, not the regeneration species to follow, and therefore the operability specification should be identical. So how does the Woodstock interpreter know to plant miracle spruce following `ccplt` and to naturally regenerate following `ccnat`? It uses the transition specification to determine the answer.

Click on the *Transitions* icon in the tree view and enter the following in the *Transitions* section:

Note: *Transitions must account for ALL possible outcomes in your model. Most errors arising from a new user's model typically are errors of omission. They forget to specify what happens after a particular action or what happens to a particular type after the action.*

```
TRANSITIONS

 *CASE thin
 *SOURCE  sf
   *TARGET sft 100
 *SOURCE  hw
   *TARGET hwt 100

 *CASE ccnat
 *SOURCE  sf
   *TARGET mw  100
 *SOURCE  sft
   *TARGET mw  100
 *SOURCE  hw
   *TARGET mw  100
 *SOURCE  hwt
   *TARGET mw  100

 *CASE ccplt
 *SOURCE  sf
   *TARGET ms  100
 *SOURCE  sft
   *TARGET ms  100
 *SOURCE  hw
   *TARGET ms  100
 *SOURCE  hwt
   *TARGET ms  100
```

For each action declared in the model, we must specify the transition matrix for each development type that is eligible. The `*CASE` keyword indicates to the Woodstock interpreter that the lines following are transition specifications associated with the action code that follows the keyword. The `*SOURCE` keyword and the thematic attribute that follows it indicate what development type is being treated by the specified action. The `*TARGET` keyword and the thematic attribute that follows it indicate what development type arises after the action is performed. Following the thematic attribute is the proportion of the source type that is assigned to the target type after the action is performed. In our model, all of the source area (100%) is transferred to the target type.

Defining yields and outputs

So far, we have defined all the dynamic processes that go on in the forest, except growth and yield. Tom has developed a set of yield tables for the various prescriptions available, and all we need to do is associate these with the development types we have declared. Click the *Yields* icon to go to the Yields section.

Note: *Observe the lack of units in the yields section. All units in a model are assumed by the modeler. The Woodstock interpreter does not assume any particular units (English, metric, cubic measure, board feet) at all.*

```
YIELDS
; Yields
; volume in cubic metres
*Y hw
wood   5   19   28   37   42   45
*Y hwt
wood   5   19   20   31   49   64
*Y sf
wood   7   98  106  122  135  145
*Y sft
wood   7   70   87  108  131  152
*Y mw
wood   1    2   12   26   38   52   74   98
*Y ms
wood   1    4   18   34   50   68   84  116
```

The `YIELDS` keyword indicates to the Woodstock interpreter the beginning of the Yields section. All of our yield tables are age-

dependent and so each yield set begins with the `*Y` keyword. Time- dependent yields are prefaced with the `*YT` keyword, and complex yields are prefaced with `*YC`. Following the `*Y` keyword is the thematic attribute for the associated development type. On the subsequent line, we find the yield component code `wood` and a series of numbers. The first number indicates the age of the first entry in the table, so for the `hw` type the first yield entry begins at age 5 periods.

The numbers that follow are the yields for the initial period and all subsequent periods. The Woodstock interpreter will check that yields are not specified for ages greater than the lifespan specified earlier. If the yield tables end before the lifespan age, the Woodstock interpreter assumes that the last entry in the table is common for all subsequent periods up to the lifespan age.

You are probably wondering how the Woodstock interpreter knows that the yields given have units of m^3/ha? The short answer is, it does not know. In the same way that period lengths are decided by you, the units used to declare areas and yield table entries are your responsibility and the Woodstock interpreter makes no assumptions about units whatsoever.

This may seem like an oversight on our part, but even if the Woodstock interpreter was told what the units were it would not be able to tell if the numbers you provided had errors or not. Since you can declare any yield components you need, we assume that you will use appropriate units; requiring you to specify units that are ignored by the interpreter is a waste of effort. However, you probably should document your yield tables with comments to remind readers what the assumed units are!

In the algebraic LP model we developed, we had to define accounting variables and construct accounting rows to report the harvest volumes in each period and the ending inventory. In a Woodstock model, we simply declare outputs and the interpreter does all the work associated with keeping track of them. Outputs are declared by specifying the action(s) that triggers the output, and the yield component(s) that provides estimates of production. Click on the *Outputs* icon in the tree view to edit the *Outputs* section.

Note: *For our problem, we need only declare two outputs, but we have declared additional outputs to make it easier later to understand the solution.*

```
OUTPUTS
; Outputs
; all volume units in cubic metres
*OUTPUT harvest Harvest volume
*SOURCE ? thin wood + ? ccnat wood + ? ccplt
wood
*OUTPUT inv     Standing inventory
  *SOURCE ? _INVENT wood
*OUTPUT sfplt   Spruce-fir area planted
  *SOURCE sf ccplt _AREA + sft ccplt _AREA
*OUTPUT sfthin  Spruce-fir area thinned
  *SOURCE sf thin  _AREA
*OUTPUT hwplt   Hardwood area planted
  *SOURCE hw ccplt _AREA + hwt ccplt _AREA
*OUTPUT hwthin  Hardwood area thinned
  *SOURCE hw thin  _AREA
*OUTPUT natarea Natural regeneration area
  *SOURCE ? ccnat  _AREA
```

The `OUTPUTS` keyword indicates the beginning of the Outputs section. The `*OUTPUT` keyword indicates that a new output is being declared

and the code that immediately follows it is used in reports and to set output targets or constraints. Following the output code is a descriptive name that is used in reports and graphics. The *SOURCE keyword tells the Woodstock interpreter to use the information that follows it in calculating the output code just declared.

For example, the harvest output is calculated by summing three sources. First, we multiply the area associated with all development types being treated with the thin action (using the ? wildcard to represent all types) by the appropriate yield table entry for the yield component wood. Then we add to that amount the area associated with the ccnat action times the yield entry for wood. Finally, we add in the area associated with the ccplt action times the yield entry for wood. Because we used the wildcard ? to represent any development type, any time one of the actions thin, ccnat or ccplt are performed, a calculation of harvest can be carried out.

To calculate the inv output, the Woodstock interpreter uses the inherent action _INVENT to inventory all development types by multiplying the area in each development type class by the yield component amount corresponding to that combination of development type and age. The remaining outputs are used to control how much area is thinned and planted for each type and they rely on the inherent yield component _AREA that simply adds up the area treated by the specified action(s).

Formulating a model

Now, we have all the pieces in place to describe forest dynamics and management success in our model, but we still need to figure out how to set up the control structures so we can run it. We will formulate an inventory projection first. Besides being conceptually easy to understand, an inventory projection model provides a good opportunity to test whether we have made any errors in our definitions affecting forest dynamics. First click the *Control* icon to return to the *Control* section and edit the text to match the following:

```
CONTROL
; Control
*LENGTH 4 ; periods are 5 years
*QUEUE ON ; using inventory projection
```

This line tells the Woodstock interpreter to run the model as an inventory projection and to base management control on selection rules. To specify these selection rules, click the *Queue* icon in the tree view to go to the *Queue* section.

```
QUEUE
; Queue
 *SELECT thin
   _MAX _AGE
 *SELECT ccplt
   _MAXDELTA wood 1
 *SELECT ccnat
   _MAXDELTA wood 1
 *TARGET harvest = 4000 1.._LENGTH
    *SOURCE thin 20
    *SOURCE ccplt 50
    *SOURCE ccnat 100
 *TARGET inv >= 0.6 * inv[0] _LENGTH
```

The Queue section begins with the QUEUE keyword. For each of the actions we have declared, we must provide selection rules that will tell the Woodstock interpreter how to order development type classes for treatments. Selection rules are specified using the

`*SELECT` keyword followed by the action code to which the rule applies. Given that thinning can only occur at a single age, there will be no confusion when a type may be thinned. Since we must specify a selection rule we will choose a simple oldest-first method (`_MAX _AGE`).

For the final harvests, development types may be harvested at different ages, and since we are trying to maximize the harvest it seems reasonable to use a "slowest-growing first" harvest rule. To implement this, we instruct the Woodstock interpreter to look at the `wood` yield component for each age class and each subsequent age class and calculate the difference using $yield_t - yield_{t+1}$. If the yield in the current period is higher than in the next period, then the quantity ($yield_t - yield_{t+1}$) is positive and nearer the top of the sorted list but if the yield in the current period is lower than in the next period, then ($yield_t - yield_{t+1}$) will be negative and nearer the bottom of the sorted list.

Finally, we must tell the Woodstock interpreter how much of each action to perform each period, and in what order to perform the actions. We have opted to use volume control with an even flow harvest policy and we represent this in the Queue section with an output target statement. We declared the output harvest to sum up volume produced by thinning and final harvests in each period. In the target statement, we want total harvest to equal 4,000 m^3 in each period of the planning horizon.

The source statements that follow the output target tell the Woodstock interpreter to obtain up to 20% of the 4,000 m^3 from thinning, followed by up to 50% of the 4,000 m^3 from clearcut and planting, with the remainder coming from clearcut and natural regeneration. Although it appears that we have requested 170% of our volume target, this is not the case. The Woodstock interpreter sees the percentages as upper limits, and only if it could perform neither of the first two actions would it obtain all the volume target from clearcut and natural regeneration actions. On the other hand, at least 30% of the volume target must come from clearcut and natural regeneration activities since at most 70% (20% + 50%) of the volume target can come from the other two activities.

The second output target statement is the ending inventory requirement. Since the output `inv` is calculated at the very beginning of the planning horizon (period 0) and at the end of every planning period, we can have the Woodstock interpreter compare the final inventory (period 4) to the initial inventory (period 0) and require that the final inventory be larger than 60% of the initial inventory. The square brackets following the `inv` code allow us to reference specific planning periods in the target statement.

There are no source statements for the second output target since inventory is an inherent forest condition. Although inventories are calculated after all other actions have been implemented, if the ending inventory output

target is not met then the Woodstock interpreter will report that the run failed even if the harvest volume target is met in all four periods.

Specifying reports

It may seem absurd, but the model we have developed so far will run without errors, but it will not report anything. To make the model useful, we need to know what the model did during execution and so we must declare a report specification. Click the *Reports* icon in the tree view and edit the *Reports* section as follows.

```
REPORTS
lkjasdf
*TARGET solution.txt
   inv     0.._LENGTH
   harvest 1.._LENGTH
   sfthin  1.._LENGTH
   hwthin  1.._LENGTH
   sfplt   1.._LENGTH
   hwplt   1.._LENGTH
   natarea 1.._LENGTH
```

The report declaration begins with the keyword `REPORTS`, followed by the `*TARGET` statement. The file name *solution.txt* indicates that we want an ASCII text file generated that includes the outputs that follow on subsequent lines. Although we can report on initial inventory conditions by specifying period 0 for `inv`, all other outputs that are not inventory measures are reported for periods 1 through the end of the planning horizon (indicated by the global variable `_LENGTH`, that equals 4 in our model).

We need to turn reporting on in the *Control* section by clicking on the *Control* icon in the tree view and adding the appropriate lines.

```
CONTROL
; Control
*LENGTH 4 ; periods are 5 years
*QUEUE ON ; using inventory projection
*REPORTS ON ; reports will be generated
*GRAPHICS OFF; no runtime graphics displayed
```

Running the model

At this point, our Woodstock model of the Tom Pike problem is complete. You can run it by pressing *F10*, or by selecting *Run Woodstock* from the *Run* menu. The Woodstock interpreter reads the entire input first, and then finds the appropriate sections within it by using the section keywords. You can see how Woodstock processes input file sections in the interpreter output.

Note: *If you encountered errors, the Woodstock interpreter will generate a log file that will indicate where the error is. Correct the errors and run the model again.*

After you run this model through the Woodstock interpreter, it should report a successful run. If you typed your model in correctly and did not encounter errors, an ASCII text file called *solution.txt* should have been created. To view it, go to the *Reports* section in your Woodstock model (by clicking the *Reports* icon in the tree view) and double-

click the filename *solution.txt* following the `*TARGET` keyword. The editor will open the file and it should look like this:

```
Run = 1    Period = 0
Standing inventory              17,400.00

Run = 1    Period = 1
Standing inventory              17,088.72
Harvest volume per period        4,000.00
Spruce-fir area thinned             28.57
Hardwood area thinned                0.00
Spruce-fir area planted              0.00
Hardwood area planted              105.26
Natural regeneration area           63.16


Run = 1    Period = 2
Standing inventory              18,563.91
Harvest volume per period        4,000.00
Spruce-fir area thinned              0.00
Hardwood area thinned              100.00
Spruce-fir area planted              0.00
Hardwood area planted               71.43
Natural regeneration area           42.86

Run = 1    Period = 3
Standing inventory              21,387.68
Harvest volume per period        4,000.00
Spruce-fir area thinned              0.00
Hardwood area thinned                0.00
Spruce-fir area planted             18.52
Hardwood area planted                0.00
Natural regeneration area           17.55

Run = 1    Period = 4
Standing inventory              23,991.19
Harvest volume per period        4,000.00
Spruce-fir area thinned              0.00
Hardwood area thinned                0.00
Spruce-fir area planted             14.81
Hardwood area planted                0.00
Natural regeneration area           14.81
```

You may not think this solution is the best possible, so feel free to make changes in the queue section, like changing the proportions of the harvest target volume or rearranging the actions in the queue. If you change the orders, be sure that you adjust the proportions accordingly because the Woodstock interpreter implements actions in the order they are listed. If you simply swap the positions of the `ccplt` and `ccnat` actions in the queue the Woodstock interpreter will never do any planting, since up to 100% of the volume target can come from `ccnat`. There would be no need to do any of the `ccplt` action since the volume target would already have been met from `ccnat`, if it were possible to do so.

An alternative formulation

Since the Tom Pike problem was originally formulated as a linear programming problem, it is instructive to see how this is accomplished in a Woodstock model. First we need to turn off the inventory projection mode and switch to optimization. Go to the *Control* section and make the following changes:

Note: *Notice that we do not have to delete the QUEUE section to change this Woodstock model to a LP formulation. The QUEUE section is ignored when optimization is used.*

```
CONTROL
; Control
*LENGTH 4 ; periods are 5 years
*QUEUE OFF ; disable inventory projection
*SCHEDULE ON; report on LP optimal solution
*REPORTS ON ; reports will be generated
*GRAPHICS OFF; no runtime graphics displayed
```

Next we have to edit the *Optimize* section and add an objective function and flow constraints. Go there by clicking on the *Optimize* icon in the tree view.

```
OPTIMIZE
 *OBJECTIVE
    _MAX harvest 1
 *CONSTRAINTS
    _NDY(harvest)        1.._LENGTH
   inv - 0.6 * inv[0] >= 0 _LENGTH
 *FORMAT LINDO
```

The *Optimize* section is headed by the OPTIMIZE keyword. Because we are interested in maximizing the first period harvest, our objective function is declared with the _MAX keyword, the harvest output code and the periods contributing to the objective (period 1). Although it is somewhat difficult to implement nondeclining flow in an inventory projection model, it is trivial in a LP model using the _NDY function. The ending inventory constraint is implemented using array syntax and at first appears a little confusing. What we have done is simply rearranged the algebraic statement inv >= 0.6 · inv[0], to conform with the LP convention of having variables on the left hand side of the inequality, and constants on the right hand side. Since the ending inventory is what we are interested in, we specify _LENGTH as the period in which the constraint applies.

Last, the *FORMAT keyword lets us specify the format of the LP matrix and the file name to store the matrix in. For illustration purposes, we chose to generate the LP matrix in Lindo format. C-Whiz, CPLEX, XA and LPSolve LP solvers are also supported.

After making the changes that we just outlined, you can generate an LP matrix by pressing *F9*, or by selecting *Generate matrix* from the *Run* menu.

A different run-time report appears in the interpreter screen, giving statistics about the LP matrix that was just generated. Click *OK* to close the Woodstock interpreter and then select *File* | *Open* | *Open file* to open the LP matrix file called *TOMPIKE.LDO*. It should look like the following:

```
!Objective
MAX
OB1MAX)  HARVEST1
ST
C1R1)   HARVEST2 -HARVEST1 > 0
C1R2)   HARVEST3 -HARVEST2 > 0
C1R3)   HARVEST4 -HARVEST3 > 0
C2R1)   INV4 >= 10440
! Existing Stand Acreage Constraints
C3)      A1+A8+A10+A11+A13+AU15 = 100
C4)      A2+A3+A4+A5+A6+AU16 = 400
! Future Stand Acreage Transfer Rows
! Period  1 Area to SFT (3) @ Age 7  C7)
-A1+R7+R9 + RU17 = 0
! Period  1 Area to MW (4) @ Age 0  C8)
 -A2 + RU18 = 0
! Period  1 Area to MS (5) @ Age 0  C9)
 -A3 + RU19 = 0
! Period  2 Area to MW (4) @ Age 0  C10)
 -A5 + RU20 = 0
! Period  2 Area to MS (5) @ Age 0  C11)
 -A6 + RU21 = 0
! Period  2 Area to HWT (6) @ Age 6  C12)
-A4+R12+R14 + RU22 = 0
! Period  3 Area to MW (4) @ Age 0  C13)
 -R7 -A8 + RU23 = 0
! Period  3 Area to MS (5) @ Age 0  C14)
 -R9 -A10 + RU24 = 0
! Period  4 Area to MW (4) @ Age 0  C15)
 -A11 -R12 + RU25 = 0
! Period  4 Area to MS (5) @ Age 0  C16)
 -A13 -R14 + RU26 = 0
! Accounting variables
 +28A1 +19A2 +19A3 -HARVEST1 = 0
 +8A4  +28A5 +28A6 -HARVEST2 = 0
 +108R7 +122A8 +108R9 +122A10 -HARVEST3 = 0
 +135A11 +49R12 +135A13 +49R14 -HARVEST4 = 0
 +135AU15 +42AU16 +131RU17 +38RU18 +50RU19
  +26RU20  +34RU21 +49RU22  +12RU23 +18RU24
  +2RU25   +4RU26  -INV4 = 0
```

If you compare this output to the algebraic model that was presented in Chapter 4, you should see that they are equivalent. Obviously the variable names are different, but you will find that either model yields the same answer.

---

* *Be sure you change the *FORMAT statement to match your solver (MPS=CWHIZ, CPLEX = CPLEX, XA = XA, LINDOMPS = LINDO, LPSOLVE = LPSolve).*

If you try to run this model by pressing *F10*, you will find that no actions are performed because we have not yet solved the LP. During matrix generation, the Woodstock interpreter also prepares batch files to call your LP solver software for you. For the moment, we will not go into details about how Woodstock works with solver software. You must also specify the file storing the schedule of activities from the LP solution. Edit the *Schedule* section in the primary input file and make it look like the following:

```
SCHEDULE [tompike.seq]
```

If you have a supported LP solver installed, you can generate a matrix, solve it and report on the solution by pressing *F12*, or by clicking on the triple-arrow button in the button bar. When processing is complete, you should find that the report file *solution.txt* now contains the following:

```
Run = 1    Period = 0
Standing inventory                17,400.00

Run = 1    Period = 1
Standing inventory                15,802.71
Harvest volume per period          5,514.38
Spruce-fir area thinned               59.15
Hardwood area thinned                  0.00
Spruce-fir area planted                0.00
Hardwood area planted                203.06
Natural regeneration area              0.00

Run = 1    Period = 2
Standing inventory                15,420.79
Harvest volume per period          5,514.38
Spruce-fir area thinned                0.00
Hardwood area thinned                  0.00
Spruce-fir area planted                0.00
Hardwood area planted                  0.00
Natural regeneration area            196.94

Run = 1    Period = 3
Standing inventory                16,046.16
Harvest volume per period          5,514.38
Spruce-fir area thinned                0.00
Hardwood area thinned                  0.00
Spruce-fir area planted               51.06
Hardwood area planted                  0.00
Natural regeneration area              0.00

Run = 1    Period = 4
Standing inventory                17,504.39
Harvest volume per period          5,514.37
Spruce-fir area thinned                0.00
Hardwood area thinned                  0.00
Spruce-fir area planted                0.00
Hardwood area planted                  0.00
Natural regeneration area             40.85
```

It is obvious that the LP solution is quite different from the inventory projection solution we ran earlier. For example, only 28 *ha* of spruce-fir was thinned in the first period in the inventory projection but 59 *ha* were thinned in the LP solution. Furthermore, less than half as much hardwood area was planted in period 1 of the inventory projection model than in the LP model. The LP also yielded a higher harvest level; to be fair, we were not trying to come up with an optimal answer when we set up the inventory projection and we could have done better by adjusting our selection rules.

Could you have come up with a set of selection rules and a harvest queue that would have been equivalent to the LP solution without prior knowledge of the optimal solution? Try modifying the *Queue* section to see how close you can come!

Chapter 7: Tutorials

The tutorials are directed at showing you what you can do with the Woodstock Forest Modeling System. We will be demonstrating features of the modeling language and how to implement some interesting modeling techniques rather than exploring conceptual modeling as we did in the last chapter.

As a novice, you will probably be most interested in seeing the structure of the models and how the parts work together. Later, as you become more proficient with the software, you may wish to revisit this tutorial for ideas that can be applied to your own management situation and purposes.

About the structure of sample models

The sample models that are provided on the distribution disks were designed to illustrate a wide range of options available in the Woodstock Forest Modeling System. We tried to use common data where possible to show that very different solutions can be found through alternative model formulations, goals and constraints. The reason for doing this was to draw your attention more to the features illustrated in the model rather than the underlying assumptions of forest dynamics that are common to the models. That said, there are a few structural principles we adhere to throughout the examples:

- unless otherwise indicated, primary input files consist solely of section keywords with external file references;
- within the Actions and Output sections, related items are grouped together;
- areas, yield components and outputs are in metric units (m^3/ha, m^3, *ha*, and so forth).

The input files for all the models are on your distribution CD in the *Samples* directory.

Tutorial One - MODEL1

MODEL1 is a very simple inventory projection model with only 2 forest types, softwood and hardwood (`SW` and `HW`). The volume target for `tot_pulp` is just an initial guess; it is unknown if this is too high or too low. Stands are assumed to regenerate naturally after harvest and are also assumed to regenerate 50% to hardwood and 50% softwood to the same forest type.

```
QUEUE
*SELECT harvest
_MAXDELTA pulp 2 ;first rule: slowest
growing 1st,2 period ahead
_MAX  _AGE ;second rule: otherwise select
oldest first
*TARGET tot_pulp = 100000 1.._LENGTH
 *SOURCE harvest 100
```

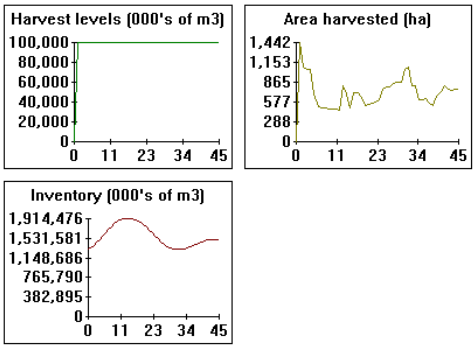
Exercise 1

Run *MODEL1* and determine if the 100,000 volume target is feasible. Rerun the model a few times, changing the volume target in 20,000 m^3 increments to make an initial estimate of allowable cut.

Since this model is so simple, we did not exert a great deal of effort to set up run-time graphics. Instead, we took advantage of the auto-scaling features of the Woodstock interpreter to generate graphs. The interpreter will size and place windows automatically to use the entire graphics screen and will apply colors and markers to line graphs to differentiate outputs.

```
GRAPHICS
*WINDOW
*WINDOW
*WINDOW
*LINES
 tot_pulp 1
 areacut  2
 growing  3
```

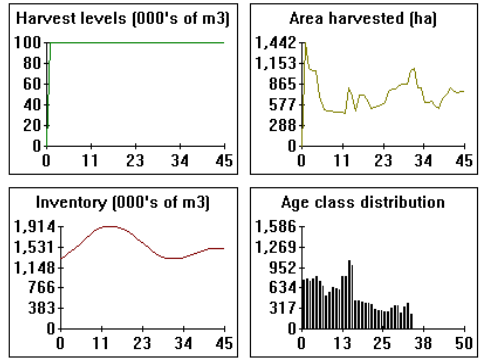
The resulting graph window from the Graphics section looks like this:



As you can see, the graphs are a bit cramped because we didn't scale the numbers to fit the titles (i.e., thousands of m^3 of m^3).

Exercise 2

Modify *MODEL1* so that the *y*-axes of the harvest levels and Inventory graphs are in the proper units. (*Hint:* use the scaling option). Also, add a new graph window to display age class information. (*Hint:* use the built-in output `_AGECLASS`). Your resulting screen should look as follows.



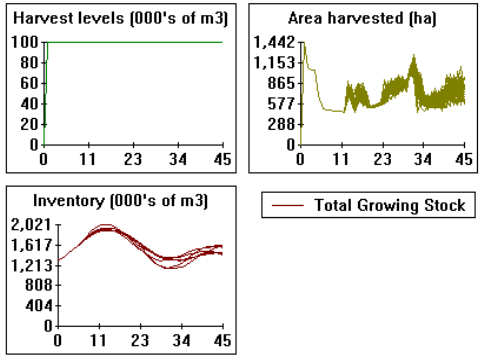
Tutorial Two - MODEL2

In *MODEL2*, we explore the aspect of uncertainty in transitions. Instead of 50% of the area harvested going into each of the 2 development

types (as in *MODEL1*), we now associate a probability of 50% for a 100 *ha* parcel of harvested land to transition to either forest type.

```
TRANSITIONS
*CASE _DEATH
 *SOURCE  ?
 *TARGET  ? 100
*CASE harvest _RANDOM 100 ;MC event = 100 ha
 *SOURCE  ?
 *TARGET sw  50 ;50% probability cutovers
                ;regenerate to sw
 *TARGET hw  50 ;50% probability cutovers
                ;regenerate to hw
```

If the Woodstock interpreter harvests 1,000 *ha* of a given development type, it will subdivide the area into 10 parcels of 100 *ha* each. Then, for each parcel, it will draw a random number between 0 and 1. If the random number is less than 0.5, the parcel is assumed to regenerate to sw and the 100 ha is transferred to the `sw` development type; otherwise, the parcel is transferred to the `hw` type. For a large number of runs, the expected outcome is for half the area regenerate to `sw` and half to `hw`, but for any given run, the proportions may be quite different. By examining the outcomes of numerous runs, you can assess the range of possible values for given outputs in each period, rather than just the expected value. For example, notice how much variation there is in the inventory and area cut amounts in this 50-run Monte Carlo simulation using a harvest level of 100,000.



Exercise 3

For a harvest level of 160,000, what is the success rate for 50 runs (what proportion of the runs permit a harvest level of 160,000 in all periods)? What is the highest harvest level you can set before the success rate falls below 100%.

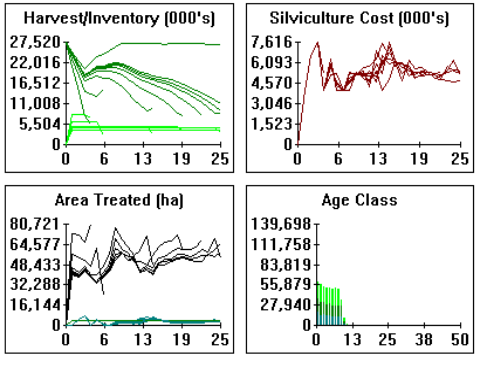
Tutorial Three - MODEL3

MODEL3 is a more complex model than *MODEL1*. Instead of just one landscape theme, there are now three: 4 in the first theme and 3 in the second and third themes. This means that up to 36 development types can be defined (4 x 3 x 3) within this model. Instead of just a generic harvest action, this model offers clearcut harvesting, planting and precommercial thinning actions. The forest dynamics in this

model are more realistic in that harvested development types do not all return to the same forest type, and some areas do not adequately regenerate at all.

Instead of a simple inventory projection where the user specifies the harvest level, *MODEL3* employs a binary search algorithm to quickly converge to a near-optimal solution.

The binary search algorithm converges to a near optimal solution in 24 iterations with a tolerance level of 1. Normally, this level of accuracy is not required and a stopping tolerance of 1,000 m^3 or 1% is sufficient.



Exercise 4

Modify the Queue section for *MODEL3* (*MODEL3.QUE*) and test the relative efficiency of tolerances of 1,000 and 1%. How different are the solutions, and how many iterations are required to converge to solution in each case?

Does the starting value in the `*TARGET` statement make a difference? Try different values and see.

Obviously, the harvest level determined in an inventory projection is going to be affected not only by what silvicultural activities that are carried out in each period, but by how much of each activity is carried out as well. It is a basic fact that if you increase the growth rate on your forest, you can expect to harvest more wood. If your forest is managed under some form of volume control, you can usually increase the rate of harvest immediately by increasing silvicultural activities like planting or spacing. This immediate increase in the harvest due to changes in planned activities is known as the *allowable cut effect* (ACE). By increasing the rate of growth in future periods, the existing growing stock does not need to last as long, allowing the harvest rate to increase immediately.

Exercise 5

What is the ACE from increasing planting by 1,000 *ha* per period in the first 15 planning periods? Compare this to the ACE from increasing precommercial thinning by 1,000 ha per period for the same interval. Is the ACE from increasing both planting and precommercial thinning simultaneously equal to the sum of the individual allowable cut effects? Why or why not?

Be aware that the economics of the allowable cut effect are highly debatable because you are capturing benefits now (more volume) and putting off the expenses that make that extra volume possible to the future. Unless the future activities are actually carried out, the allowable cut effect is meaningless. Moreover, blind pursuit of ACE without considering the costs associated with your silvicultural regimes can result in a management strategy far beyond the point of diminishing returns.

Tutorial Four - CASE1 and CASE2

All of the *CASEx* series models use the same underlying forest dynamics and are designed to illustrate the flexibility of the Woodstock modeling language to implement a variety of model formulations with minimal editing of the base model. *CASE1* and *CASE2* are designed to illustrate how a slightly different conceptual model of forest dynamics can yield different solutions, even when the development types, yields and outputs are exactly the same!

In *CASE1*, we represent harvesting and regeneration as a single action. To represent the alternatives of natural and artificial regeneration, we need to define two different actions, `clna` and `clpl`. The first action, `clna`, represents clearcut harvest followed by natural regeneration. The second action, `clpl`, represents clearcut harvest followed by planting.

```
ACTIONS
*ACTION   clna Y Clearcut & nat regen
 *OPERABLE clna
  jpwp ? ? ok _AGE 10 _AGE _MAXAGE
     ? ? ? ok totv 60 totv 40
*ACTION   clpl Y Clearcut & plant
 *OPERABLE clpl
  jpwp ? ? ok _AGE 10 _AGE _MAXAGE
       ? ? ? ok  totv 60  totv 40
TRANSITIONS
 *CASE clna
   *SOURCE   bfsp  ?  ?  ?
   *TARGET   bfsp  ? us na    80
   *TARGET   bfsp  ? tt na    20
   *SOURCE      ?  ?  ?  ?
   *TARGET      ?  ? us na   100


*CASE clpl
   *SOURCE   jpwp  ?  ?  ?
   *TARGET   jpwp  ? fs pl   100
   *SOURCE   bfih  ?  ?  ?
   *TARGET   wsrs  ? fs pl   100
   *SOURCE   spih  ?  ?  ?
   *TARGET   wsrs  ? fs pl   100
   *SOURCE   ihsf  ?  ?  ?
   *TARGET   wsrs  ? fs pl   100
   *SOURCE      ?  ?  ?  ?
   *TARGET   bsbs  ? fs pl   100
```

In *CASE2*, we represent harvesting and regeneration as separate actions. To represent the alternatives of natural and artificial regeneration, we need to define three different actions, `clct`, `regn` and `plnt`. The first action, `clct`, represents a clearcut harvest that results in an unstocked condition. The second action, `regn`, represents natural regeneration that results in a

stocked condition. The third action, `plnt`, represents planting that also results in a stocked condition.

```
ACTIONS
*ACTION    clct Y Clearcut harvest
 *OPERABLE  clct
  jpwp  ?  ? ok  _AGE 10   _AGE _MAXAGE
       ?  ?  ? ok  totv 60  totv 40
*ACTION regn N natural regeneration cutover
 *OPERABLE  regn
      ?  ? uu uu  _AGE 1  _AGE 1
*ACTION    plnt N Site prepare and plant
 *OPERABLE plnt
      ?  ? uu uu  _AGE 0  _AGE 1

TRANSITIONS
*CASE clct
  *SOURCE    ?  ?  ?  ?
  *TARGET    ?  ? uu uu   100
*CASE regn
  *SOURCE    bfsp  ? uu uu
  *TARGET    bfsp  ? us na    80
  *TARGET    bfsp  ? tt na    20
  *SOURCE       ?  ? uu uu
  *TARGET       ?  ? us na   100
*CASE plnt
  *SOURCE    jpwp  ? uu uu
  *TARGET    jpwp  ? fs pl   100
  *SOURCE    bfih  ? uu uu
  *TARGET    wsrs  ? fs pl   100
  *SOURCE    spih  ? uu uu
  *TARGET    wsrs  ? fs pl   100
  *SOURCE    ihsf  ? uu uu
  *TARGET    wsrs  ? fs pl   100
```

When we run these two models with binary search turned on, the projected even flow harvest from *CASE1* is 1,781,250 *m^3/period*; for *CASE2*, the even flow harvest is 5.6% higher at 1,881,043.

Exercise 6

Why is the target statement for `rgnarea` necessary? (*Hint:* try commenting it out and see what happens). Given that the same yields and outputs are used in *CASE1* and *CASE2*, why are the final harvest levels different? (*Hint:* Consider how sites for planting are chosen in *CASE1* versus *CASE2*). Can you modify *CASE2* such that it gives the same solution as *CASE1*, without changing the *Actions* or *Transitions* sections?

Tutorial Five - CASE3

CASE3 builds on the basic forest model of *CASE2* to make the model more realistic. Instead of directly limiting planting and thinning to fixed area amounts each period, we control these activities indirectly by limiting the total expenditure on silviculture.

In *CASE2*, thinning and planting are set at fixed amounts each period and therefore, there is no inherent trade-off between one and the other. Each activity is carried out until the target level is reached.

```
*TARGET plarea <= #plantarea 1.._LENGTH
 *SOURCE plnt  100
*TARGET tharea <= #thinarea  1.._LENGTH
 *SOURCE thin  100
*SOURCE       ?  ? uu uu
```

In *CASE3*, planting and thinning both contribute to an output called budget and it is this output that we control in the target statement.

```
*TARGET budget <= #dollars 1.._LENGTH
 *SOURCE plnt   50
 *SOURCE thin   50
 *SOURCE plnt  100
 *SOURCE thin  100
```

The interpreter reads this target statement and performs the following actions:

- with a budget of $1.5 million, up to $750,000 can be spent on planting before the interpreter stops (at $100 / *ha*, 7,500 *ha*);

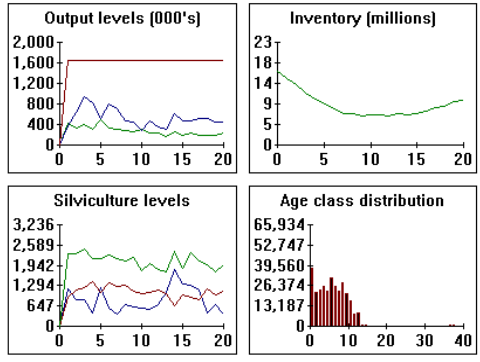
- then up to $750,000 can be spent on thinning before the interpreter stops (at $75 / *ha*, 10,000 *ha*);
- if insufficient area is available for thinning, then the remaining money (up to 100% of the budget) can now be spent on planting;
- otherwise the remaining budget can be spent on more thinning.

The other change in *CASE3* is the addition of an ending inventory restriction. In the earlier models, complete liquidation of the forest during the planning horizon is a possibility. Now, we are requiring the ending inventory to be at least 60% of what we started with.

```
*TARGET gstock >= #endinv _LENGTH
```

Exercise 7

Does it matter whether planting or thinning comes first in the queue? Is there a better mix of planting and thinning for the same total expenditure? Explain why the target statement for `gstock` has no corresponding source statement. How much impact does the ending inventory restriction have on harvest level? Reduce the inventory requirement to 50% of the initial inventory and compare.



Tutorial Six - CASE4

CASE4 is a minor modification to *CASE3* in that we changed the *Transitions* section to allow regeneration failures. Rather than have 100% of the unstocked areas become stocked following the *`plant`* and *`regn`* actions, a small percentage simply reverts to the unstocked condition.

```
TRANSITIONS
*CASE regn
 *SOURCE    bfsp  ? uu uu
  *TARGET    bfsp  ? fs na  76
  *TARGET    bfsp  ? tt na  19
  *TARGET    bfsp  ? uu uu 5 {regen failure}
 *SOURCE       ?  ? uu uu
  *TARGET      ?  ? fs na  90
  *TARGET     ?  ? uu uu   10 {regen failure}
```

```
*CASE plnt
 *SOURCE   jpwp  ? uu uu
  *TARGET   jpwp  ? fs pl 98
  *TARGET   bfsp  ? uu uu 2  {regen failure}
 *SOURCE   bfih  ? uu uu
  *TARGET   wsrs  ? fs pl 95
  *TARGET   bfih  ? uu uu 5  {regen failure}
 *SOURCE   spih  ? uu uu
  *TARGET   wsrs  ? fs pl 95
  *TARGET   spih  ? uu uu 5  {regen failure}
 *SOURCE   ihsf  ? uu uu
  *TARGET   wsrs  ? fs pl 95
  *TARGET   ihsf  ? uu uu 5  {regen failure}
 *SOURCE   wsrs  ? uu uu
  *TARGET   wsrs  ? fs pl 98
  *TARGET   wsrs  ? uu uu 2  {regen failure}
 *SOURCE     ?  ? uu uu
  *TARGET   bfsp  ? uu uu 2  {regen failure}
  *TARGET   bsbs  ? fs pl 98
```

When we run this model, the binary search algorithm converges to a solution of only 1,627,625 instead of the 1,637,848 found in CASE3.

Exercise 8

Having regeneration failures obviously reduces the effectiveness of the silviculture program and the allowable cut effect is reduced. How much more would you have to spend on silviculture to yield the same harvest level found in *CASE3*? Can you accomplish this by modifying the allocation of dollars in the *Queue* section rather than increasing the budget?

Tutorial Seven - CASE5

In *CASE5*, we reintroduce stochastic events. This time, we will model regeneration failures as Monte Carlo events. The Transitions section of *CASE4* is modified as follows:

```
TRANSITIONS
*CASE regn  _RANDOM 100
 *SOURCE bfsp  ? uu uu
  *TARGET bfsp  ? fs na 76
  *TARGET bfsp  ? tt na 19
  *TARGET bfsp  ? uu uu 5 {5% prob of fail}
 *SOURCE ? ? uu uu
  *TARGET ? ? fs na 90
  *TARGET ?  ? uu uu 10 {10% prob of fail}
*CASE plnt _RANDOM 100
 *SOURCE jpwp ? uu uu
  *TARGET jpwp ? fs pl 98
  *TARGET bfsp ? uu uu 2 {2% prob of fail}
 *SOURCE bfih  ? uu uu
  *TARGET wsrs  ? fs pl 95
  *TARGET bfih  ? uu uu 5 {5% prob of fail}
 *SOURCE spih  ? uu uu
  *TARGET wsrs  ? fs pl 95
  *TARGET spih  ? uu uu 5 {5% prob of fail}
 *SOURCE ihsf  ? uu uu
  *TARGET wsrs  ? fs pl 95
  *TARGET ihsf  ? uu uu 5 {5% prob of fail}
 *SOURCE wsrs  ? uu uu
  *TARGET wsrs  ? fs pl 98
  *TARGET wsrs  ? uu uu 2 {2% prob of fail}
 *SOURCE ?  ? uu uu
  *TARGET bfsp ? uu uu 2 {2% prob of fail}
  *TARGET bsbs ? fs pl 98
```

The `_RANDOM 100` in the *CASE* statements cause the Woodstock interpreter to use 100 *ha* allocation units when applying the transition outcome. For example, in the first transition specification, a random number of 0.843 would result in 100 ha being regenerated as fully stocked fir, a random number of 0.12 would result in 100 ha regenerated as a first thicket, and a random number of less than 0.05 would produce a regeneration failure.

To test the viability of the sustainable harvest level found in *CASE4*, we ran the model for 20 iterations and found that 55% of the time the

forest is capable of sustaining the harvest for the entire 20 planning periods. Depending on how risk-averse we are as managers, we may be willing to accept the harvest level from *CASE4*, given that it is sustainable more often than not. On the other hand, we may want to lower the harvest slightly to improve the success rate.

Exercise 9

Plot the relationship between success rate and harvest level, ranging from harvests of 150,000 up to 165,000 per period, in 1000 m^3 increments. What do you think you would select as the appropriate harvest level? How much more money would you spend on silviculture to lessen your risk?

Tutorial Eight - CASE6

All of the sample problems so far have been simulation models, either inventory projections or Monte Carlo simulations. In *CASE6*, we take the basic model presented in *CASE3*, but we jettison the *Queue* section and replace it with an *Optimize* section.

The key difference between an inventory projection and a linear programming formulation is how you structure the question to be answered. In a simulation model, you address the management problem in terms of "If I do the following actions in the order specified and at the levels indicated, what will be the result?" To get a good solution from this approach, you have to have a sense of how the system responds to different activities so that you can anticipate outcomes and choose activities appropriately.

In a LP analysis, you address the problem in terms of "I would like the outcome of management to have the following characteristics… what activities are required in what order and at what level to achieve this outcome?" You still need to understand how the system works, but rather than worry about how to choose among alternatives, you concentrate on the desired outcome.

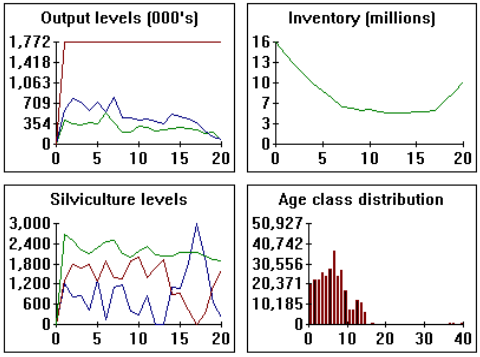
```
OPTIMIZE
*OBJECTIVE
  _MAX spfvol 1 ;maximize first period
harvest of spfvol
*CONSTRAINTS
 spfvol - spfvol[-1] >= 0 2.._LENGTH
;nondeclining yield
 budget <= #dollars       1.._LENGTH
 gstock >= #endinv _LENGTH ;ending inventory
constraint
*FORMAT MPS
SCHEDULE [case6.seq]
```

The *Optimize* section of *CASE6* incorporates an objective function and constraints that are similar in effect to the target specifications in the *Queue* section of *CASE3*. In both cases, we want to maximize the even flow harvest of timber, leave a sufficient ending inventory at the end of the planning horizon, and we do not want to exceed our silvicultural budget.

** Be sure you change the *FORMAT statement to match your solver (MPS = CWHIZ, CPLEX = CPLEX, XA = XA, LINDOMPS = LINDO, LPSOLVE = LPSolve).*

Because we have to solve the LP with a special solver program, we need to write the matrix in a format that the program can read. In this example, we used a solver called C-Whiz that reads MPS formats. If you have a supported solver installed, you can generate a matrix, solve it and report on the solution by pressing the F12 key or by selecting this option from the Run menu.

After the solver has found an optimal solution, the LP solution is converted into a Schedule section that the Woodstock interpreter follows to implement the optimal harvest schedule. This file is called *CASE6.SEQ* and is referenced in the *Schedule* section of the *CASE6* primary file. The results should look similar to the following:



The optimal solution yielded a harvest level of 1,771,975 m^3/*period* which is 8.2% higher than the 1,637,848 m^3/*period* found using binary search in *CASE3*.

Exercise 10

Given that the constraints and outputs from *CASE3* and *CASE6* were essentially identical, why is the LP solution better? Use the graphs from *CASE3* to help you decide. In the silviculture levels window, the green line is harvest area, the red is planting area and blue is thinning.

Exercise 11

Suppose we wanted to obtain the same harvest level, but because of limited planting stock, we need to minimize the area planted each period. Edit *CASE6* to reflect this new objective and modify the Optimize section appropriately. (*Hint:* If you don't need to harvest any wood, you don't need to plant anything!). Is there any difference in the solution? Why would it be possible to obtain the same harvest level with less planting?

Tutorial Nine - CASE7

Up until now, we have only really been concerned with timber production. However, virtually all public lands agencies, and many private and industrial landowners must also

address in their forest plans non-commodity outputs such as wildlife, recreation, water and other benefits. In many cases, these non-commodity benefits are conditions of the forest that are derived not from harvesting, but rather the presence or absence of certain species and age combinations on the forest.

To illustrate how such outputs can be represented in Woodstock, a fictitious 0-1 habitat suitability index was developed for mature conifer furbearers. This index depends on species and age of stands: only mature conifer species in older age-classes provide suitable thermal cover and forage. The index appears in the Yields section as an age dependent yield component (*mcfndx*), and the amount of suitable habitat (*habitat*).

Exercise 12

What do you think happens to MCF habitat under the optimal harvest schedule developed in *CASE6*? Now, open the *Schedule* section of *CASE7.PRI* and make sure that it refers to *CASE6.SEQ*, the optimal harvest schedule from the last tutorial. Press *F10* to play back the schedule and observe the results to test your prediction. Were you correct?

When you are faced with conflicting goals, it may be difficult to decide how best to proceed. With an optimization model, only a single objective function is allowed to be maximized or minimized; all other goals must be handled by constraints. Setting constraint levels can be difficult because people tend to value one set of outputs much more highly than others, depending on their individual perspectives. Compromise is often required, but in general, this is NOT the analyst's job. Instead, you should present the alternatives so that an informed decision can be made.

This is where benchmarking comes into play. In a benchmarking exercise, you identify the major outputs of interest and optimize each one in turn. In doing so, you identify not only the ceilings on output levels but also the floor values. Many people are often surprised to see that production of one output does not necessarily preclude production of another generally-competing output. By establishing floor and ceiling values for outputs of interest you have bounded the decision-space; in other words, demands for output levels outside the decision-space are not feasible and can be easily dispensed with. If you do not fully explore the decision-space, negotiations can be hampered by unrealistic demands, or more seriously, a viable alternative may be overlooked completely.

Question: *Why is the objective function _MAX distrev - discost rather than _MAX dnr?*

CASE7 is a simple example illustrating a trade-off between present net value, timber production and wildlife habitat. In the optimize section you will find three different objective functions that maximize discounted net revenue, timber volume and habitat over the entire planning horizon.

If you have installed C-Whiz, you can press *F11* (or select *Generate and Solve*) to solve all three of these objective functions in succession. C-Whiz will be called three times and the *Schedule* sections for the three optimizations will be stored in *CASE7.SQ1*, *CASE7.SQ2* and *CASE7.SQ3* respectively. To see the results of each optimization, edit the *Schedule* section and replace the file name with one of these sequence files.

If you have another solver, you will need to comment out 2 of the objective function statements and solve each one individually. If you have not yet installed a LP solver or do not wish to run one, the optimal Schedule sections are included on the distribution CD with the tutorial files and you may run these sequence files to see the results.

What do we learn from this benchmarking exercise? First, there is always some timber produced, whether we maximize discounted net revenue (DNR), timber volume or wildlife habitat.

Exercise 13

Clearly there must be timber harvested in order to have a positive DNR so why would the habitat maximization include harvesting at all? What is the most timber that could be harvested in a period during the planning horizon? What is the minimum amount of MCF habitat that will exist during the planning horizon? How much standing inventory is left at the end of the planning horizon?

Without additional constraints, none of these harvest schedules are particularly appealing. Flow constraints are probably necessary in order to meet mill demands, and the liquidation of inventory in the final planning period leaves no source of timber for future periods. Mature conifer furbearer habitat drops precipitously from the more than 50,000 ha available initially and never recovers under the timber and DNR maximization scenarios. Therefore, constraints that maintain some minimum level of MCF habitat are necessary.

We usually recommend that analysts incorporate multiple constraints one-at-a-time rather than all-at-once. In doing so, you are able to determine the individual costs of these constraints on your objective function. Moreover, there are often combined effects of multiple constraints that result in radically different solutions from those generated when the constraints are applied singly. By making one-at-a-time changes, the analyst is able to monitor how constraints work in concert. In some cases, constraints represent the only viable choice possible, and so the cost of the constraint is somewhat irrelevant since alternatives cannot be selected anyway. However, many goals are somewhat fuzzy, being tempered by other needs. In these cases, initial constraints may be relaxed or tightened up to generate a more acceptable solution. Each time a constraint is applied or modified, the scenario

changes. Woodstock's new scenario management feature lets you test a variety of constraint and objective combinations in a systematic manner, and lets you save and recall these scenarios directly.

Exercise 14

CASE7 includes a number of scenarios for you to try out. For example, click on *File | Scenarios | Restore* and select the *MaxDNR* scenario. The Optimize section will now include an objective that maximizes discounted net revenue and constraints on ending inventory and MCF habitat. Since this scenario was already solved, you can simply hit *F10* to see the results. Try out the other scenarios provided, and create a few of your own. Remember that you only need to save the model sections that change from scenario to scenario. If you click the *Default* button, the *Optimize*, *Schedule* and choices files are marked for inclusion; simply add any other desired files before saving the scenario.

Having performed the benchmark exercise, we know that none of the unconstrained "optimal" solutions are viable due to excessive fluctuations in output levels, insufficient outputs in one or periods and insufficient ending inventory for sustainability. Suppose that mill demand for timber is 2.4 million m^3*/period* and that wildlife biologists suggest that 50,000 *ha* of habitat is desirable to maintain a viable population of furbearers each period. Since timber and wildlife production are equally important goals, is it possible to jointly optimize these two? The answer is yes, through a technique called *goal programming*.

Tutorial Ten - CASE8

Goal programming is really just a minimization problem where we penalize over and under-achievement of output goals. For example, suppose we wanted to achieve 10,000 widgets and 5,000 grommets produced per period in a fictitious harvest schedule. We know that either the widget or the grommet targets are achievable alone but not together; we wish to jointly produce the maximum number of widgets and grommets with the smallest deviation from the stated goals.

In the previous tutorial, we performed a benchmark exercise to establish bounds on the decision space where we learned that maximizing timber production resulted in a loss of most MCF habitat, and maximizing habitat area yielded unacceptably low harvest volumes. CASE8 is essentially the same as CASE7, but employs a goal programming formulation. For sustainability reasons, the ending inventory is constrained to be at least equal to the average inventory over the planning horizon. Finally, budgeting requirements limit the annual silvicultural budget to be $300,000.

Exercise 15

Generate, solve and run *CASE8*. Does this solution strike you as a fair compromise between timber production and wildlife habitat? What is the implicit relationship between a unit of timber production and a unit of wildlife habitat? Suppose that timber interests place 5 times as much value on a cubic metre of timber than on 1 hectare of MCF habitat. How would you represent this perspective in the goal programming formulation? Is this a better or worse solution than the original *CASE8*? Could you devise a similar harvest schedule without using a goal programming formulation?

At first blush, goal programming seems like an ideal method for addressing equal but competing goals. However, goals are rarely exactly equal, and the goals can easily differ by an order of magnitude or more. Clearly, weights are needed to prevent one goal from dominating the solution, but in doing so, you are implicitly valuing one output in terms of another. It can be very difficult indeed for a group of managers to settle on a set of weights for goal programming; in fact, it is the same type of problem as determining what objectives to use and what constraints. In reality, when trade-offs among competing interests must be made, analysis can provide useful information but rarely forms the sole basis for a decision. Alternative solutions can usually provide as much useful insight into a problem as a goal programming formulation, without the headaches of trying to determine appropriate weights for different goals.

Although it has its pitfalls, goal programming can be useful when you are modeling highly constrained forest problems. Such models can be difficult to solve, requiring a large amount of processing time even to determine infeasibility. By employing a goal programming formulation, you can ensure that a feasible solution will be found that at least comes close to meeting your specified constraints.

Summary

These tutorials were designed to introduce the sample models provided on your distribution disk. All these examples are based on timber management objectives, but you are not limited to just looking at timber issues with Woodstock. For instance, Woodstock models have been used to look at wildlife habitat availability, vegetative management requirements in conservation areas, fire ecology issues, potential impacts of global warming, and many others. One adventurous modeler even used Woodstock to look at blueberry production!

We encourage you to experiment with the sample models. The best way to become proficient at modeling is through practice. However, don't just make changes and run the model again. Try to anticipate what the new outcome will be, and if you are wrong, try to figure out why the model responded differently than you

expected. Being wrong in a modeling exercise is not a bad thing! Observing a non-intuitive result and then figuring out why it happened means that you have learned something about the way your system works. And that's the whole point of this modeling business in the first place!

Appendix 1: Tutorial Answers

This appendix contains the answers to the tutorial exercises provided in Chapter 7 of the Woodstock Modeling Reference.

The numbers you arrive at may not correspond exactly with those reported here. This is not a problem so long as your numbers are relatively close—the tutorial answers are meant to illustrate some basic concepts and to act as a guide when working through the tutorials. It is the relative differences that are important, not the precise numbers.

Tutorial One

Exercise 1

The 100,000 m^3 volume target is feasible (i.e., the target was met in every planning period of the planning horizon).

The initial estimate of allowable cut made by changing the volume target in 20,000 m^3 increments is 140,000 m^3 */period*. Using a binary search, the estimated allowable cut was 157,734 m^3*/period*.

Exercise 2

The text below shows the modifications to the *MODEL1 GRAPHICS* section necessary to produce the screen shown (see “Exercise 2” on page 74).

```
GRAPHICS
*WINDOW "Harvest levels (k m3)"
*WINDOW "Area harvested (ha)"
*WINDOW "Inventory (M m3)"
*WINDOW "Age class distn"
*LINES
tot_pulp 1 1000 ;scale by 1,000
areacut 2
growing 3 1E6 ;scale 1,000,000
_AGECLASS 4 ;generate an age class
            ;distribution and
            ;place in the 4th
            ;graphics window
```

Tutorial Two

Exercise 3

Using a harvest target of 160,000 m^3, the success rate was 0% after 50 iterations. See the Transitions chapter of the User Guide for a discussion of random transitions in Woodstock.

The highest harvest target that may be set before the success rate falls below 100% is 154,000 m^3. Chapter 4, “Monte Carlo simulation” on page 35 of the Modeling Reference and Woodstock User Guide have

detailed discussions of Monte Carlo simulation and conducting a Monte Carlo simulation analysis.

Tutorial Three

Exercise 4

The table below shows the number of iterations required to converge to solution using either an absolute (1,000 m^3) or a percent (1%) tolerance. In all cases, a 1% tolerance yielded a lower harvest level than a 1,000 m^3 tolerance (4,718,750 m^3/period versus 4,732,910 m^3/period respectively).

	Cutvol			
	3.0E6	**4.0E6**	**4.5E6**	**5.0E6**
Tolerance 1,000	14	14	15	16
Tolerance 1%	9	9	9	10

Fewer iterations are required to converge to solution using a 1% tolerance. The reason is obvious: a 1% change in the volume target (40,000 m^3) is greater than an absolute tolerance of 1,000 m^3. Woodstock stops iterating once the change in the harvest has exceeded the tolerance, therefore higher tolerances result in faster convergence. For the same reason, higher tolerances yield lower target levels; smaller tolerances force Woodstock to perform more iterations and continue to increase or decrease the target level beyond the level determined using higher tolerances. The starting value in the **TARGET* statement had virtually no impact on the number of iterations. For a further discussion of conducting a binary search analysis, see the Queue section chapter of the User Guide and "Binary Search" on page 33 of the Modeling Reference.

Exercise 5

The following table summarizes the ACE resulting from increased silviculture (planting and thinning) in the first planning period. Increased thinning alone did not generate an ACE, so the ACE produced by increasing thinning and planting together was the same as that from increasing planting alone.

	Periodic Harvest Level (m^3)	**ACE (m^3)**
No increase	4,523,437	
Increase plant only	4,570,312	46,875
Increase thin only	4,523,437	0
Increase plant and thin	4,570,312	46,875

The ACE shown in the table below resulted from increases of 1,000 hectares in each of the first 15 planning periods.

Note: *you will have to add additional target statements to the Queue section for each of the first 15 planning periods. As expected, the ACE*

was greater than before, but the trend was identical. The Woodstock User Guide discusses the Queue section and defining target levels.

	Periodic Harvest Level (m^3)	ACE (m^3)
No increase	4,523,437	
Increase plant only	4,710,937	187,500
Increase thin only	4,523,437	0
Increase plant and thin	4,710,937	187,500

Tutorial Four

Exercise 6

Since an action is used to trigger natural regeneration, Woodstock needs a target and action source statement in the harvest queue. Otherwise, Woodstock will not schedule the action. Sure enough, removal of the target statement for `rgnarea` caused the area naturally regenerated to drop to zero hectares in all periods. The harvest level also fell from 1,926,773 m^3/period to 1,409,484 m^3/period.

Differences in the final harvest level for *CASE1* and *CASE2* are attributable to differences in site selection for planting. In *CASE1*, the planting decision occurred at the time of the clearcut; the model chose either to clear-cut and plant or to clear-cut and naturally regenerate a site. In *CASE2*, the planting decision occurred after the decision to clear-cut (i.e., each action had its own selection rule). In this case, the model planted the best growing sites first, which permitted an increase in the harvest level.

You can modify *CASE2* to achieve the same answer as *CASE1* by making the selection rules

for planting the same as those for clear-cut (i.e., *_MAXDELTA soft 1*; *_MAXDELTA soft 2*).

Tutorial Five

Exercise 7

It does not matter in what order planting and thinning occur in the harvest queue. We tried several different combinations of planting and thinning, all to no avail. The harvest level remained the same in spite of the ordering.

Gstock is an inventory output (defined using *_INVENT*) and therefore does not require that an action occur for it to be calculated. Woodstock, therefore, does not require an action source statement for the *Gstock* target statement (or for any other inventory target statement for that matter).

Relaxing the ending inventory restriction from 60 to 50 percent of the initial growing stock level increased the harvest level by slightly more than 2 percent (1,637,847 m^3/period versus 1,673,889 m^3/period respectively). This indicated that the ending inventory target was

not overly restricting, but by relaxing it, Woodstock would be able to increase the harvest level.

Tutorial Six

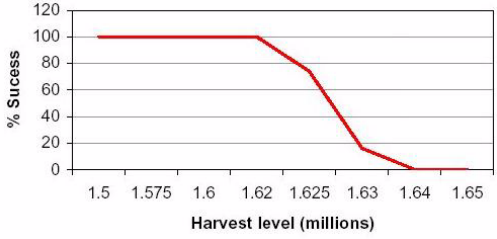
Exercise 8

For the *CASE4* model to yield the same harvest level as *CASE3*, you would have to increase silvicultural spending by 1.2 million dollars. You cannot increase the harvest level by modifying the allocation of dollars spent on planting and thinning in the Queue section.

Tutorial Seven

Exercise 9

The appropriate harvest level depends on your aversion to risk, as does the amount spent on silviculture to lessen risk. For example, a risk-averse decision-maker may choose a lower harvest level or spend more on silviculture to reduce risk than would a risk-indifferent decision maker.



Tutorial Eight

Exercise 10

The linear programming (LP) solution is better than the binary search simulation because the LP found a combination of treatments (in terms of the development types treated and the timing of treatments) that generated a higher harvest level. To read about some of the advantages and disadvantages of LP, see Chapter 5, "An overview of linear programming" in the Modeling Reference.

Exercise 11

To minimize the area planted, in the *OPTIMIZE* section, (1) change the LP objective to one that minimizes the area planted and (2) set as a constraint the harvest level determined in the initial run (1,771,974 m^3/*period*). What we are

doing is asking the model to plant as little as possible in each planning period while maintaining the same harvest level as before. In the worst case, the model will plant exactly the same amount of area as it did before, but you won't know for certain unless you ask.

```
OPTIMIZE
*OBJECTIVE
_MIN plarea 1.._LENGTH                (1)
*CONSTRAINTS
spfvol >= 1,771,974 1.._LENGTH        (2)
budget <= #dollars  1.._LENGTH
gstock >= #endinv       _LENGTH ;ending
                                ;inventory
                                ;constraint
```

In the first run (in which we maximized the harvest level), the total area planted over the planning horizon was 26,499 hectares compared to 23,591 hectares in this run. Concomitant reductions in silvicultural spending amounted to over 2.7 million dollars! For an overview of LP, see Chapter 5 of the Modeling Reference; for instructions on conducting an LP analysis see the sectionon LP analysis in the Woodstock User Guide.

Questions?

We would be happy to answer any questions you may have about Remsoft, Woodstock, or any of our other software packages.

You can contact us at the following address. Email is the preferred support format.

Remsoft Inc.
Suite 160, Frederick Square
77 Westmorland Street
Fredericton, New Brunswick, Canada E3B 6Z3

Tel: 1-800-792-9468 or +1-506-450-1511
Fax: +1-506-459-7290
Email: support@remsoft.com
www.remsoft.com

Appendix 2: Glossary of Terms

Absolute constraint An absolute constraint is used to control the quantity of some output that can occur in a period or interval. For example, an upper limit on silviculture expenditures or a lower limit on the amount of mature conifer habitat available. The output to be controlled must be greater than or equal to, equal to or less than or equal to a fixed quantity.

Accounting variable Accounting variables are used in linear programming models to simplify the construction of complex constraints. Rather than reference a large number of decision variables, a constraint is formulated forcing the accounting variable to sum the value of the decision variables.

Action An action is any silvicultural treatment, harvest, administrative activity or naturally occurring event that causes a development type to undergo a transition and may cause outputs to be generated.

Age class An age class represents all parcels of land that are similar in age. In Woodstock models, an age class covers the same time interval as planning periods. For example, if your model uses 5 year planning periods you must also use 5 year age classes.

Age class distribution An age class distribution is used to assign area to individual age-classes within a development type. The total area of the development type is equal to the sum of the areas in the individual age classes.

Age-dependent yield A yield stream (physical quantity, economic indicator or other parameter of interest) that varies with the age of a particular development type rather than with the planning periods in the planning horizon.

Aggregate attribute An aggregate landscape theme attribute represents an exclusive subset of a landscape theme. An aggregate attribute represents only a portion of the categories within a landscape theme. Aggregate attributes may not include basic attributes from multiple themes.

Allowable cut The quantity of timber that may be harvested from a forest for a specific time period. Usually this quantity is expressed as an annual volume estimate.

Allowable cut effect (ACE) An immediate change in the amount of harvest that is due to changed assumptions about the future. For example, by increasing the planned level of fertilization in the future from current levels, your model may indicate that you can increase your harvest immediately rather than wait until the fertilized stands are mature. The assump-

tions that make ACE occur are 1) constraints on harvest flows, and 2) the future activities are actually carried out. If either of these conditions is not true, there cannot be an allowable cut effect.

Allowable Sale Quantity (ASQ) U.S. Forest Service terminology. The quantity of timber that may be sold from the area of suitable land covered by the Forest Plan for a time period specified by the Plan. This quantity is usually expressed on an annual basis as the "average annual, allowable sale quantity" and usually applies for 10 years or longer. This level is considered the "ceiling" of timber quantity that can be harvested from suitable lands.

Analysis area An analysis area represents a specific portion of the forest and may be composed of a single stand or multiple stands. By definition, an analysis area represents a particular age class of a specific development type, but it does not have to represent the entire development type class.

Animal Unit Month (AUM) The tenure of 1 animal-unit (considered to be one mature 1,000-pound cow or the equivalent) based on the average daily forage consumption of 26 pounds of dry matter per day for a period of one month (Society of Range Management 1974).

Area control A classical forest management concept related to regulation and the normal forest. Full regulation can be obtained (in theory) in one rotation by harvesting a fraction of the total forest each year equal to $^1/_R$ where R = rotation length. Although the area harvested each year is the same, the volume flows from harvesting are not guaranteed to be even.

Basic attribute A basic landscape theme attribute is one that is defined as part of a *THEME specification. Basic attributes represent only a single category within a landscape theme.

Basic variable Decision variables that are considered part of an LP solution, usually with an assigned value greater than zero.

Basis A basis is the collection of basic decision variables. If you modify an LP very slightly, the new optimal solution may be very similar to the current one. Substantial decreases in solution time may be had by starting to solve the new LP from where the previous optimal solution terminated. Not all solvers allow insertion of a basis as an advanced starting point.

Best Management Practices (BMPs) Land management methods, measures or practices intended to minimize or reduce water pollution. Usually, BMPs are applied as a system of practices rather than a single practice. BMPs are selected on the basis of site-specific conditions that reflect natural background conditions and political, social, economic and technical feasibility.

Benchmarking In developing a preferred management strategy, it may be necessary to explore a variety of different objectives to satisfy conflicting goals. For example, you may

wish to maximize in turn present net value, timber production, wildlife habitat, recreational visitor days and other goals. By doing so, you can establish bounds on the decision space for the various goals that will aid in determining appropriate constraints on outputs.

Coefficient In a LP model, outputs are calculated as a sum of decision variables multiplied by corresponding yield entries. For example, if 1 unit of harvested forest land yields 100 units of timber, then 100 is the coefficient for the decision variable associated with that prescription.

Column A column is a decision variable in a linear programming matrix.

Compound yield A yield stream (physical quantity, economic indicator or other parameter of interest) that varies with both the age of a particular development type and the planning periods in the planning horizon. Compound yields are defined by multiplying an age-dependent yield table and a time-dependent yield table.

Complex yield table Complex yield tables are defined by referencing previously defined age-dependent or time-dependent yield tables. Compound yields are necessarily defined as complex yield tables but complex yield tables may be entirely age-dependent or time-dependent as well.

Constraint Constraints are a linear programming construct and represent limits to the problem, usually due to physical limitations of resources. For example, constraints are used to limit the area of forest in development type classes, but they may also be used to limit how much of an output is produced.

Decision variable In a linear programming model, every prescription (combination of a development type class, action and timing choice) represents a decision variable. For example, clearcut harvesting a 10-period old spruce type in period 1 would represent a decision variable; harvesting a 10-period old spruce type in period 2 would represent another. The constraints and objective function are linear functions of the set of decision variables in a model.

Development type A development type represents a parcel of forest land that is represented by a particular sequence of landscape theme attributes. Development types are assumed to follow identical developmental patterns regardless of age. In a Woodstock model, a development type is defined by specifying the appropriate landscape theme attributes and an associated age class distribution.

Development type class A development type class is a single age class within a development type.

Error message Error messages are generated by the Woodstock interpreter whenever a syntax or logic error is encountered, when an ambiguous or redundant construct has been encountered requiring the interpreter to assume a default condition or to warn the user about potential problems with a particular feature.

Error messages are designed to help you correct mistakes in your input files and are of two types: fatal error messages indicate an error that must be corrected and warning messages indicate a potential problem but can be ignored.

Fatal error A fatal error is one that the Woodstock interpreter cannot resolve on its own, forcing it to generate an error message and terminate the run. You must correct the error before your model will execute properly.

Flow constraint A flow constraint allows you to establish relationships between periods for output levels. Flow constraints are used to regulate the rate at which an output may change from one period to the next. For example, flow constraints may be used to guarantee an even flow of outputs or to bound the output levels between a lower and an upper limit.

General relational constraint General relational constraints allow you to require a proportional relationship between two outputs. For example, general relational constraints may be used to ensure that for every hectare of jack pine that is planted, two hectares of black spruce are planted.

Generalized Model II Model II is a form of linear programming harvest scheduling models where decision variables represent a stand only over the course of a single rotation. In traditional Model II, once the stand is harvested, the area in the original decision variable is transferred to a new decision variable at age 0. Generalized Model II not only allows the transfer of area when a stand is harvested, but also allows transfers due to mortality or silvicultural treatment and allows transfers to ages other than 0.

Global attribute A global landscape theme attribute represents all the categories within a landscape theme.

Goal programming A LP minimization problem where the objective function sums deviations from stated output goals; if all goals can be met exactly, then the objective function value is zero. Goals may be weighted equally or they may have different weights; goals with higher weights are favored over those with lower weightings.

Implicit constraint Implicit constraints do not have to be specified by you in the constraints specification but are generated by the Woodstock interpreter automatically. Implicit constraints include initial area constraints, transfer rows and accounting rows.

Infeasibility A problem is deemed infeasible if there is no solution to the problem without violating one or more constraints.

Infeasible solution If your LP solver software returns an infeasible solution, it indicates that one or more of your constraints cannot be met. For example, if you specify a constraint requiring 50,000 m^3 of wood be harvested in each planning period, but only 45,000 m^3 is available in one of the periods, your problem is infeasible and you will have to change the constraint.

Interval An interval is used to represent a range of one or more planning periods. In a Woodstock model you represent an interval by:

specifying an individual period, specifying the starting period followed by two dots followed by the ending period (i.e., 1…5), or specifying individual periods separated by commas.

Landscape theme Landscape themes are like map layers in a GIS system or fields in a forest inventory database in that they describe various aspects of the forest. Themes may describe physical features of the forest such as slope class, vegetative composition or road density, or they may be administrative features such as a ranger district or county.

Landscape theme attribute An attribute represents a particular category within a landscape theme. For example, a landscape theme called slope class may include 5 attributes such as inoperable (>45%), steep (>30%), moderate (>15%), gentle (>5%) and flat ($\leq$ 5%).

Lifespan The maximum age in periods that a development type may reach before senescence is assumed to occur.

Linear programming An optimization technique that will choose the best alternative from a set of possible or feasible alternatives. It is a method of allocating scarce or limited resources among competing activities in an optimal manner. LP models are composed of a linear objective function subject to linear constraints.

Mask Whenever you want to reference one or more development types, you must specify a sequence of landscape theme attributes arranged in the proper order. This sequence of attributes is a mask, and it may contain basic, global or aggregate attributes.

Matrix A rectangular array of numbers having a fixed number of rows (constraints and objective function) and columns (decision variables).

Mixed-mode Woodstock models may incorporate elements of simulation and optimization in a single model run. For example, you may wish to run a Monte Carlo simulation for a few periods to model stochastic events such as fire or insect attack, and then invoke linear programming to derive an optimal response solution to the random events. The Woodstock interpreter must then act both as a simulator and a matrix generator in a single model run.

Monte Carlo simulation A modeling technique that allows some activities to occur based on a probabilistic basis. Although the activity may occur with probability (P) in any given run, there is no guarantee that it will occur. Therefore, each simulation run can yield different results; in a regular inventory projection model where activities occur with certainty, the answer is always the same no matter how many times you run the model.

Nonbasic variable Decision variables that are not considered part of an LP solution and they always have an assigned value of zero.

Nondeclining yield Nondeclining yield is a form of flow control where the output level in a period may be the same as the previous period or higher, but the output level may not decrease from the level of the previous period.

Normal forest A classical forest regulation concept that describes a forest structured such that an equal amount of wood may be harvested periodically from the same area of land in perpetuity. A normal forest assumes no catastrophic losses such as wildfire and insect/disease attack.

Objective function The objective function quantifies the output or outputs that are to be maximized or minimized.

Operable inventory An operable inventory is the maximum amount of an output that could be produced in a period from development type classes eligible for a particular action.

Optimal A mathematical programming concept for a class of solutions to a problem definition in which the objective function value is at a global maximum or minimum.

Optimal Solution An optimal solution is a feasible combination of decision variables that yields the highest objective function value for a maximization problem and the lowest objective function value for a minimization problem. Although no other solution exists for a given problem that will yield a higher objective function value, other solutions may exist that yield the same objective function value. These are termed alternative optimal solutions.

Optimization A mathematical programming technique to find optimal solutions to mathematical programs. Optimization includes linear programming, integer programming, dynamic programming and others.

Output An output may be a physical quantity (i.e., wood volume), an economic quantity (i.e., total cost or revenue), or other indicator that is produced when a triggering action occurs.

Output target An output target is a desired output level that is used to control the magnitude of actions occurring in a simulation model.

Period A period is a time interval composed of one or more years. Periods are the basis for age classes as well as planning periods and they must be consistent throughout your models.

Planning horizon The planning horizon is the total number of planning periods or years that your model covers. For example, a model with a 20-period planning horizon and 5-year planning periods would cover 100 years.

Prescription A prescription is a set of management actions that are applied to a development type class or analysis area to produce one or more desired outputs. For example, a prescription might be described as precommercial thin a jack pine type at age 3 periods, thin from below at age 6 periods extracting 20 m^3/ha and final harvest at age 10 periods.

Queue In an inventory projection model, you must state the rules for applying activities. Once the rules are established, the order in which these activities can be performed forms a queue that is processed sequentially from beginning to end.

Right hand side In a constraint formulated as an algebraic expression, by convention decision variables are located on the left hand side of the relational operator and the right hand side is reserved for constants.

Row A row is an objective function or a constraint in a LP matrix.

Scenario A modified Woodstock model that incorporates one or more changes from the standard run. For example, an analyst may wish to determine the effects of different growth and yield estimates, constraints and/or objectives, or perform a risk analysis by incorporating random outcomes. Each of these changes would constitute a different scenario, and the Woodstock editor supports the saving and restoring of them easily.

Selection rules The rules that determine 1) the development types that are eligible for an activity and 2) the order in which activities are applied to eligible types.

Simulation In the context of Woodstock modeling, simulation models are those that do not rely on optimization techniques for their solution. In general, simulations refer to models that mimic the workings of a complex system, and usually these models are stochastic in nature.

Solution A solution is the result of processing a linear programming matrix with a LP solver. It may be optimal, infeasible or unbounded.

Source In a Woodstock model, a source indicates the development type(s) that will undergo transition due to an action; the action and development type(s) generating outputs contributing to an output target specification; and the action, development type(s) and yield components contributing to an output.

Standing inventory A standing inventory is the maximum amount of an output that could be produced in a period from development type classes if the entire class were liquidated completely. Of course, many classes are not eligible for actions and therefore standing inventories are more useful as condition indicators rather than output available for extraction.

Target In a Woodstock model, a target indicates the development type(s) that will arise following a transition due to an action, the output device to which a report will be written or the level of output desired in an action queue specification.

Time-dependent yield A yield stream (physical quantity, economic indicator or other parameter of interest) that varies with the planning periods in the model rather than with the age of a particular development type.

Transfer row A transfer row is an implicit constraint generated by the Woodstock interpreter. Transfer rows are the mechanism through which area is transferred from one decision variable to another following a harvest or other action.

Transition Transitions specify the outcomes of actions. If an action is applied to a development type, you must specify what happens to the area treated. For example, clear cut harvesting a softwood plantation may result in 100% of the area being transferred to a new softwood type, but clear cut harvesting a mixed wood stand may result in 50% of the area being transferred to a new softwood type and 50% being transferred to a new hardwood type.

Unbounded An unbounded solution indicates that a decision variable may become infinitely large even though the problem is feasible. Unbounded solutions mean that a constraint is incorrectly formulated, either through an error by the Woodstock interpreter or through a corrupted LP matrix. If you encounter an unbounded solution, try generating the LP matrix again. If the new matrix still results in an unbounded solution, contact Remsoft support.

Volume control A classical forest management concept related to regulation and the normal forest. Full regulation, can be obtained (in theory) over one or more rotations by harvesting an equal volume of wood each period. Although the volume harvested each period is the same, the area harvested is not guaranteed to be the same, and so it may take several rotations to achieve regulation.

Warning A type of error message generated by the Woodstock interpreter when it encounters a non-fatal error or when there is the possibility that an option was used incorrectly. In some cases the warning may be flagging a construction that is correct and fully intended; in these cases you may instruct the interpreter not to echo that particular warning message in further runs.

Yield component A yield component is the basis for calculating outputs and identifies particular yield tables within a yield set. Yield components are most often unit area estimates of volume, basal area or other stand attributes, but in fact they may be anything.

Yield set A group of one or more yield tables associated with one or more development types identified by a specific attribute mask that may contain basic, global and aggregate landscape theme attributes.

Yield table A yield table represents the yield stream for a particular yield component associated with a specific development type attribute mask.

Index

A

absolute constraints
  definition 28
accounting variables
  definition 56
action
  definition 17
age-dependent yield
  definition 19
allowable cut effect
  definition 34
analysis area
  definition 17
area control
  definition 31
area-based
  definition 21

B

binary search
  definition 33
branch and bound
  definition 49

C

common emphasis group
  definition 17
complex yield table
  definition 19
compound yield table
  definition 19
conceptual model
  definition 16
constraints
  definition 27
coordinate systems
  review of 41

D

decision variable
  definition 45
deterministic
  assumption of LP 45
development type
  definition 17, 20
development type class
  definition 17
divisibility
  assumption of LP 45

E

equality constraint
  definition 46
equality constraints
  definition 27
explicit constraints
  definition 27

F

flow constraints
  definition 28
  even flow 29
  nondeclining yield 29
  sequential flow 29
  stepwise flow 29

G

general relational constraints
  definition 29

I

implicit constraints
  definition 27
independency
  assumption of LP 45
infeasible
  definition 50
inventory projection
  advantages 34
  concept 31
  disadvantages 34
  example 32
inventory projection models
  definition 31

L

landscape themes
  definition 20
Linear programming 27
linear programming
  advantages 57
  computer solution methods 50
  concept 41
  disadvantages 58
linearity
  assumption of LP 45
lower limit constraints
  definition 27

M

matrix (LP)
  definition 46
maximization
  definition 29
minimization
  definition 29
Model I formulation 54
Model II formulation
  definition 55
Monte Carlo actions
  definition 22
Monte Carlo simulation

advantages 40
concept 35
disadvantages 40
Monte Carlo transitions
definition 23

N

non negativity
assumption of LP 45

O

objective function
definition 27, 29
outputs
definition 18

P

partial cut
definition 23
planning horizon
definition 20
planning periods
definition 19
probability
implications of 36

Q

queue
definition 24

S

selection rules
definition 25
sensitivity analysis
definition 45
stand
definition 16
steers and trees problem
definition 47
graphical representation 48
integer form 49
stochastic actions
example 37
stochastic transitions
example 38
stratum-based
definition 20

T

thematic attributes
definition 20
time-dependent yield
definition 19
Tom Pike problem
Actions, Transitions 64
complete Woodstock model 69
Control, Lifespan, Areas 62
definition 53
Landscape 61
Model II LP 55
new model creation 61
Optimize 70
Queue 67
Woodstock formulation 61
Yields, Outputs 65
transition
definition 17
Tutorials
Exercise 1 74
Exercise 10 82, 83, 84, 85, 86
Exercise 11 82
Exercise 2 74
Exercise 3 75
Exercise 4 76
Exercise 5 76
Exercise 6 78
Exercise 7 79
Exercise 8 80
Exercise 9 81

U

upper limit constraints
definition 27

V

volume control
definition 31

Y

yield components
definition 19, 23