

# Newcomb

## A Solar System Ephemeris Program

Marc A. Murison  
murison@aa.usno.navy.mil

James L. Hilton  
jhilton@aa.usno.navy.mil

*Astronomical Applications Department  
United States Naval Observatory  
3450 Massachusetts Ave, NW  
Washington DC 20392*

*March 3, 1999*

# TABLE OF CONTENTS

<b>Chapter 1: Project Outline and Top Level Program Structure</b> .....	3
Introduction .....	3
Astronomical Context .....	3
Program Design and Language Considerations .....	3
Project Outline .....	4
Top-Level Structure .....	5
<b>Chapter 2: The Observations Module</b> .....	6
Introduction .....	6
Observing Platforms .....	6
Observation Types .....	6
<b>Chapter 3: The Parameter Adjustment Module</b> .....	8
Overview .....	8
Linear Parameter Estimation .....	8
Formal Parameter Errors .....	9
Nonlinear Parameter Estimation .....	10
<b>Chapter 4: The Integration Module</b> .....	11
Physical Model .....	11
<i>Newtonian Formulation</i> .....	11
<i>General Relativistic Formulation</i> .....	11
<i>The Lunar Orbit</i> .....	11
Numerical Integration Algorithms .....	11
An Object Oriented Approach .....	11
Analysis Submodules .....	11
<i>Automatic Detection of Mean-Motion Resonances</i> .....	11
<i>Close Approach Detection</i> .....	11
<i>Frequency Analysis</i> .....	11
Ephemeris Generation .....	11
<b>Chapter 5: The User Interface</b> .....	12
Introduction .....	12
Main Program Interface .....	12
Observation Module Interface .....	12
Integration Module Interface .....	12
<i>Solar System Settings</i> .....	12
<i>General Relativity Settings</i> .....	12
<i>Integrator Settings</i> .....	12
<i>I/O Settings</i> .....	12
<i>Initial Conditions</i> .....	12
<i>ICs Database Capabilities</i> .....	12
<i>Runtime Graphics</i> .....	12
Parameter Adjustment Module Interface .....	12
<b>Index</b> .....	13

# CHAPTER 1: PROJECT OUTLINE AND TOP LEVEL PROGRAM STRUCTURE

MARC A. MURISON  
 Astronomical Applications Department  
 U.S. Naval Observatory, Washington, DC

## 1. INTRODUCTION

This chapter describes the top-level structure of the planned Naval Observatory Solar System Ephemeris program, called Newcomb.<sup>1</sup> Newcomb is intended to be the successor of, and loosely derives its structural inheritance more or less from, PEP, the Planetary Ephemeris Program at the Smithsonian Astrophysical Observatory.<sup>2</sup> Computer program design and language capabilities have advanced far beyond the anticipations of three and a half decades ago when PEP and the JPL DE programs were originally developed. (DE and PEP are the only existing high-precision solar system ephemeris programs.) Program technology that is several generations out of date, combined with the practical inability to add further significant capabilities or modifications to PEP, has been deemed sufficient cause for development of a new ephemeris program. Additional motivations are that it is to the USNO's great advantage to have a comprehensive ephemeris capability in-house, and that Newcomb will provide a check against PEP and the JPL DE programs.

## 2. ASTRONOMICAL CONTEXT

### 3. PROGRAM DESIGN AND LANGUAGE CONSIDERATIONS

Chief among the advantages of writing a new program is the opportunity to make use of modern programming and design technologies, including object-oriented design (OOD) and object-oriented programming (OOP), as well as graphical user interfaces (GUIs) and the highly productive "components" programming associated with rapid application development (RAD) environments. Newcomb is written entirely in C++, and development and testing are done entirely within the best RAD environment currently available.<sup>3</sup> We take full advantage of standard OOP/OOD concepts and techniques, including data encapsulation, template classes, polymorphism, and, where necessary, multiple inheritance.

The benefits of a completely object-oriented approach are many, including faster prototyping and development, fewer and more easily locatable coding errors, vastly simpler and more intuitive design, more sophisticated functionality, easily extensible architecture, and (most importantly) drastically

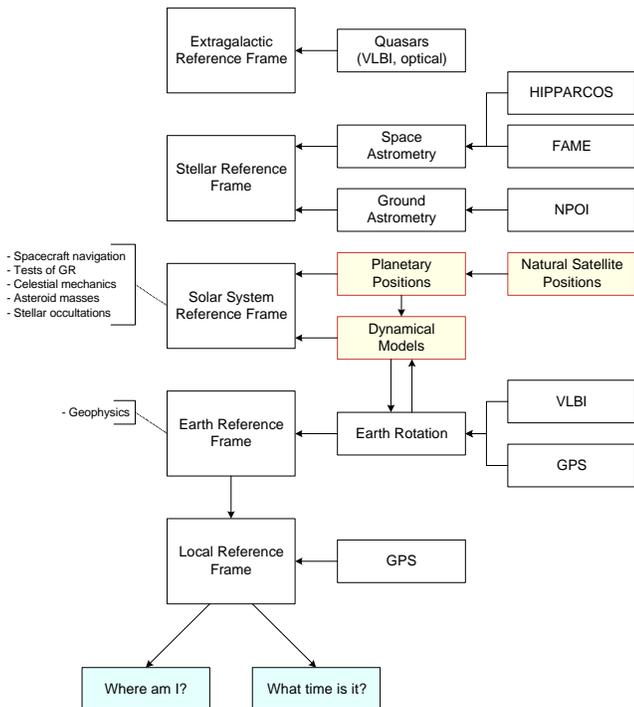


Figure 1 — Reference frame hierarchy.

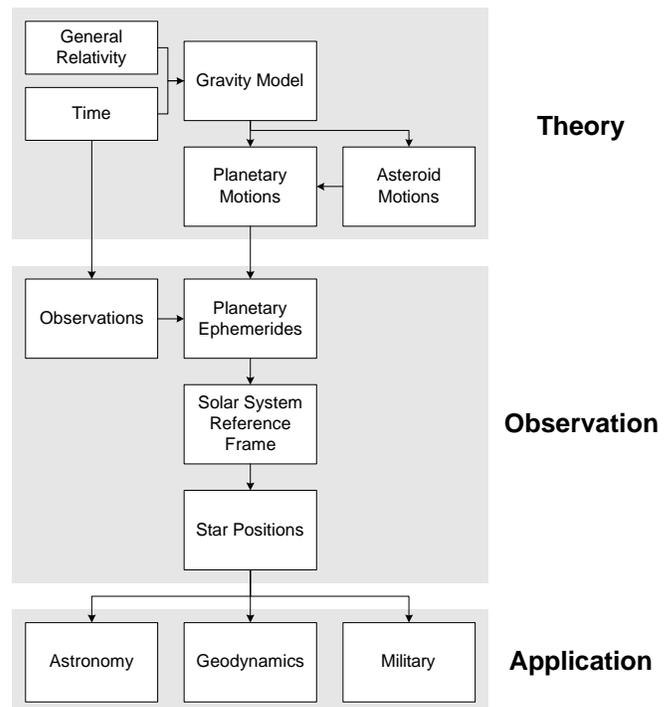


Figure 2 — Major dependencies.

<sup>1</sup> See <http://aa.usno.navy.mil/Newcomb/> for the official Newcomb web site.  
<sup>2</sup> See <http://cfa-www.harvard.edu/~reasen/ssd.html> for information about PEP.  
<sup>3</sup> <http://www.borland.com/bcppbuilder/>

reduced long-term maintenance costs. Another major benefit is that the program can be brought up and running with minimal functionality, allowing further capability to be easily and painlessly incorporated as need arises.

Ease of extensibility is largely a result of object-oriented design, but it is also directly related to how good that design is. Hence, considerable effort has gone and is still going into the design of Newcomb. Experience in the software industry over the last one to two decades abundantly shows that the payoff later on in terms of maintenance and extensibility is far out of proportion to the effort expended early on — in the design stages — of the program life cycle.

The benefits of a RAD environment for development and testing are also very attractive. Chief among the attractions is the ease by which it is possible to create highly sophisticated graphical user interfaces. During design, graphical interface components — such as buttons, edit fields, toolbars and so on — are “dropped” onto a window form or dialog box. Useful properties of the components are settable at design time, in addition to being available during runtime. It is easy to create custom components as well. For example, for Newcomb we designed a custom component that is in fact a fully functional and self-contained power spectral density (PSD) analysis package. All that is needed to add a PSD module to a program is to drop the PSD component onto a form or dialog. Hence, building, changing, and extending the graphical user interface of a program is astoundingly easy. This of course spills over and makes changing or extending major program structural elements correspondingly painless.

#### 4. PROJECT OUTLINE

In these beginning stages of the Newcomb project, tasks naturally fall into three main categories: program design, documentation, and science applications. A rough outline of the most obvious subjects that must be addressed is:

- I. Design Issues
  - A. numerical integration scheme
    - 1. object-oriented design
    - 2. Integrable objects have knowledge of dynamical environment as well as the ability to dynamically evolve in that environment.
  - B. exception handling
    - 1. all exceptions fully recoverable
    - 2. procedure stack traceback
  - C. robust parameter estimation
    - 1. Singular Value Decomposition (SVD)
    - 2. swipe a package from elsewhere
  - D. graphical user interface
    - 1. use GUI application frameworks package (such as ZAF from Zinc) to ensure platform portability
  - E. reduction of observations
  - F. individual class design and testing
- II. Science Issues and Projects to Consider
  - A. asteroids
    - 1. masses from orbital interactions
    - 2. provide ephemerides (services to the community)
    - 3. cumulative effects on planetary motions
  - B. lunar motion
    - 1. chaotic dynamics
      - a. predictions from numerical models
      - b. comparisons with LLR data
    - 2. radiation pressure [\[ref\]](#)
    - 3. resonant interaction between tidal and GR terms
    - 4. lunar librations
  - C. Nordtvedt  $h$  parameter (anomalous gravitational field energy effects — i.e., a difference between gravitational and inertial mass proportional to the gravitational binding energy of a body)
  - D. GR precession
    - 1. lunar orbit
    - 2. Earth’s spin
  - E. bounds on time variation of the gravitational constant
  - F. millisecond pulsars
    - 1. derive Earth orbit
  - G. bounds on dark matter in the solar system?
  - H. planetary satellites?
    - 1. centroiding vs. satellitederived center of mass
  - I. other science?
- III. Documentation
  - A. code
    - 1. source documentation model (see TM96-01)
    - 2. interface (“user’s manual”)
  - B. algorithms
  - C. physics
    - 1. GR and partial derivatives
  - D. parameter estimation and error and correlation analysis
  - E. numerical integration design
  - F. reduction of observations

## 5. TOP-LEVEL STRUCTURE

The top level process structure of Newcomb is shown in Figure 3. Basic operation is as follows.

The observations module is responsible for reading input astrometric observations and “massaging” them as necessary. Massaging operations are listed in [Chapter 2](#). The observations will be of various types (Figure 5), taken at various observing locations (Figure 4), including spacecraft.

The integration module is responsible for numerically integrating a sophisticated dynamical model of the solar system — including general relativistic terms, a detailed Earth-Moon system, planetary spin vectors including precession and nutation, and an unlimited number of asteroids — to produce an ephemeris.

The model ephemeris is then compared with the observations in the O-C section of the parameter adjustment module to produce a set of residuals. The parameter estimator uses the partial derivatives of the model equations with respect to the model parameters (including initial conditions) to solve the associated nonlinear least squares problem for the most probable set of model parameter values that minimizes the O-C residuals.

The adjusted model parameters are then fed back into both the ephemeris generator and the observation transformation methods. The data are rereduced as necessary, and a new ephemeris is generated by the integration module, using the updated parameter values. These are again combined to produce a new set of residuals. This process is iterated until the residuals satisfy predetermined success criteria.

At the end of the iterative process, we will have produced an ephemeris that best fits the observations, given the model used, as well as the best-fit model parameters, formal error estimates of those parameters, and the parameter cross correlations. The

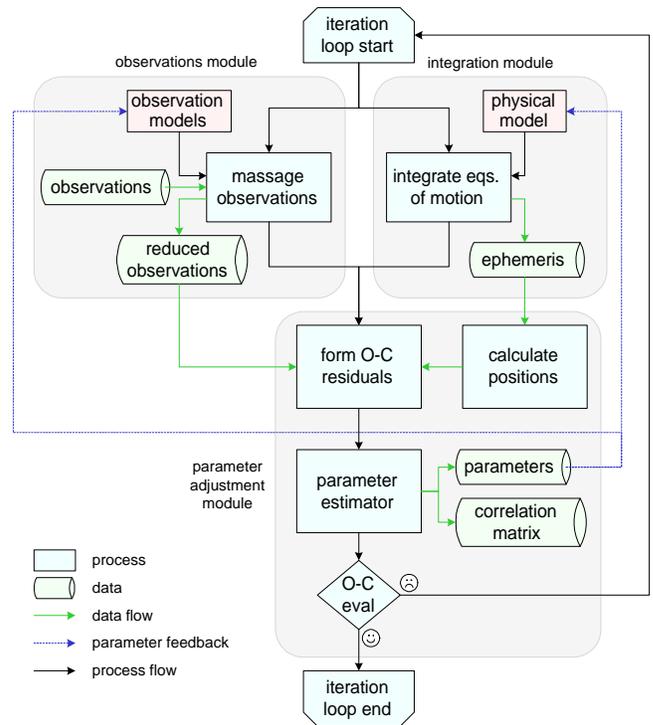


Figure 3 — Major program processes.

parameter error estimates and parameter correlations are derived from the partial derivatives and the correlation matrix from the least squares analysis. Experience with PEP has shown that, normally, at most only a couple or a few iterations are needed.

## CHAPTER 2: THE OBSERVATIONS MODULE

MARC A. MURISON  
Astronomical Applications Department  
U.S. Naval Observatory, Washington, DC

### 1. INTRODUCTION

Perhaps the most difficult section of the program will be the module that processes input observations and reduces them to a form suitable for passage to the O-C section of the parameter adjustment module (see Figure 5). Essentially, the observations will be sent to the O-C section in the form of apparent positions, corrected for various biases, including (but not limited to):

- ▶ catalog corrections
- ▶ delay/doppler bias corrections
- ▶ coordinate frame fiducialization
- ▶ aberration corrections
- ▶ nutation and precession

Integral to this section are the specific types of observational datasets and the specific types of observational platforms. The data and platform types vary widely.

### 2. OBSERVING PLATFORMS

One must consider the various observing platforms presently available in the solar system. They are

- I. Planet
  - A. Earth
    1. Earth-based observatories
    2. Earth orbiters
  - B. Planetary landers
  - C. Planetary orbiters

- II. Deep space probes (i.e., gravitationally unbound from all planets and satellites)

Figure 4 shows the object hierarchy of observing platforms.<sup>4</sup> The C++ code classes reflect this hierarchy. Each input data-stream will contain relevant observing platform information. An appropriate observing platform object will encapsulate this information. Each type of platform object also encapsulates the necessary functionality (referred to as *methods*) to provide information needed to manipulate or transform data of the corresponding type (see Figure 6).

For example, planetary observing platform objects know how to process and nutate coordinates to a specified epoch. Each base class contains parameters and functionality common to all subclasses derived from it. The derived classes contain only the additional or specialized parameters and functionality required to handle platforms of a specific kind. For example, since all planetary platforms have a basic precession and nutation capability, these methods reside in the base class `PlanetPlatform`. An `EarthPlatform` object automatically inherits all the functionality and data of `PlanetPlatform`. The `EarthPlatform` object therefore contains only additional abilities, data, or refinements, for example precession parameters specific to the Earth. Proper use of inheritance eliminates code duplication for common tasks in a natural and intuitive way. The inheritance mechanism is built into the C++ language and therefore requires no enforcement by or special discipline from the programmer.

Figure 4 intentionally shows only the major class types, in accord with the introductory nature appropriate to this Chapter. It is a simple matter to derive further specialized classes from the base classes shown. For example, one would derive a `VikingOrbiter` from `OrbiterPlatform`.

### 3. OBSERVATION TYPES

The various observation data types fall naturally into the two broad categories, timing (in a sense, the radial coordinate from the observer) and positions (on the sky, i.e. transverse to the radial direction). The complete breakdown is as follows:

- I. Transverse (position)
  - A. Optical observations
    1. Global positions
      - a. Transit circle
    2. Differential positions
    3. **Occultations**
      - a. **Satellite-planet**
      - b. **Star-planet**
      - c. **Spacecraft-planet**
    4. **Transits**
      - a. **Solar**
      - b. **Planetary**

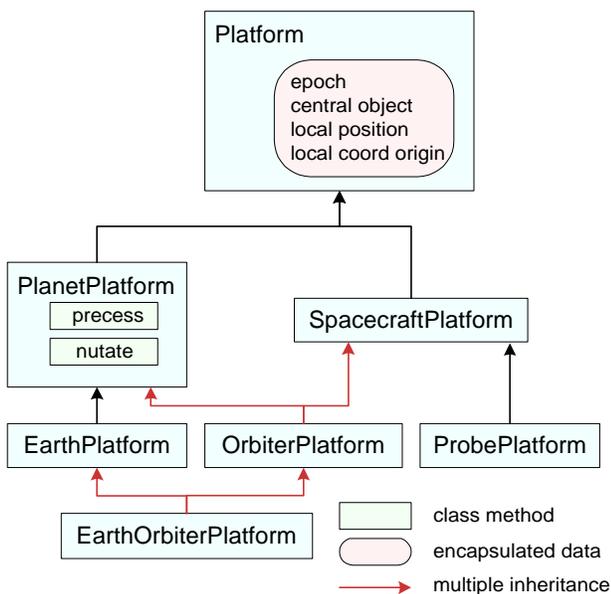


Figure 4 — Observing platform class hierarchy.

<sup>4</sup> Arrows in Figures 2 and 3 point *from* derived classes *to* parent (also called *base*) classes. This is the standard notation.

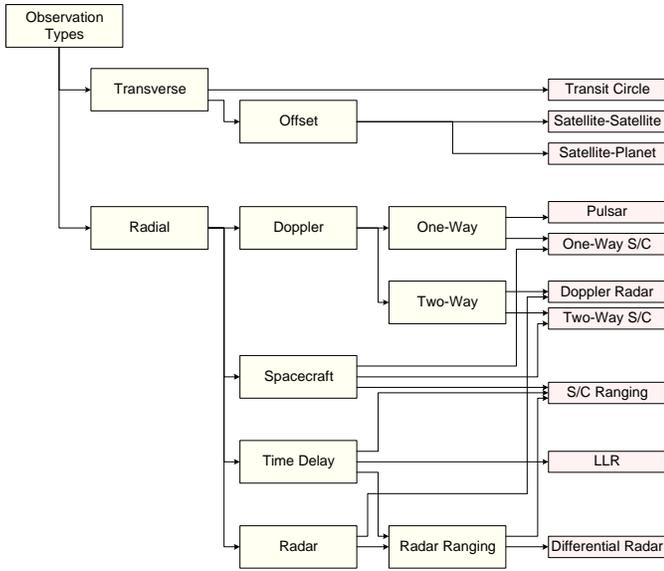


Figure 5 — Observation types.

- B. **VLBI**
- II. Radial (timing)
  - A. Doppler observations
    - 1. Oneway
      - a. Pulsars
      - b. Spacecraft
    - 2. Twoway
      - a. Radar
      - b. Spacecraft
  - B. Time delay observations
    - 1. LLR
    - 1. Radar
      - a. Differential radar
      - b. **Radar closure**
    - 2. Spacecraft
      - a. Single
      - b. **Multi**

For reasons having mainly to do with datasets that are currently insufficiently large or insufficiently accurate to have a

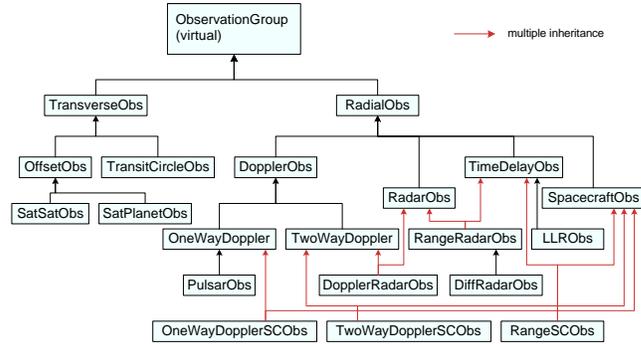


Figure 6 — Observational class hierarchy.

substantial effect on ephemeris accuracy, early versions of Newcomb will not include the observation types shown in **light red**. Because extensibility is built into the design of Newcomb, adding further capabilities as they become necessary will involve minimal effort — there is no need, from a maintenance standpoint, to include capabilities that are anticipated to go unused for a long time. That is, with a good object-oriented design we do not have to worry so much about “making room” for anticipated future capabilities. Figure 5 shows the observation types hierarchy. Figure 6 shows the proposed corresponding object class hierarchy.

Each type of input data stream will contain embedded type information, and instantiations of the appropriate data objects will handle the data. The specific objects shown in Figure 3 encapsulate not only the corresponding observational data but also the functionality required to reduce that data type. For example, notice that all datatype objects have, via inheritance from the base class `Observation`, platform information and the ability to handle (say) aberration.

As with Figure 4, Figure 6 is intentionally not complete, especially regarding encapsulated data and method details. However, all the important base classes, and their inheritance dependencies, are shown.

# CHAPTER 3: THE PARAMETER ADJUSTMENT MODULE

MARC A. MURISON  
Astronomical Applications Department  
U.S. Naval Observatory, Washington, DC

## 1. OVERVIEW

The parameter adjustment module is relatively straightforward. The processed observations from the Observations Module and the calculated ephemeris data from the Integration Module are compared, thus forming the O-C residuals. First, coordinate frame compatibility between the observations and the synthetic ephemeris is reconciled. The calculated ephemeris must be transformed to apparent positions in order to match the observations. The residuals are characterized, with statistical and descriptive output going to disk as well as to an output window on-screen. At this point, outlying data points can be automatically — or manually — detected and removed.

The core of the module follows with the determination of parameters via a maximum likelihood estimator. The normal equations are formed and solved, and the parameters and associated formal error estimates are saved. Finally, the residuals are evaluated, and the module exits with a solution “acceptability” code. Figure 7 illustrates the process.

Matrix inversion is accomplished via singular value decomposition (SVD), which is very robust and offers useful diagnostics for ill-conditioned matrices. Singularities are automatically detected and corrected, and the problem parameters are identified. In essence, if the algorithm encounters an ill-conditioned matrix, it safely steps around the problem point(s) and proceeds in such a way as to mine the matrix for the maximum amount of information. When a singularity (rare in practice) or degenerate column (not rare!) is encountered, the combination of parameters that led to the fault is easily extracted. Thus, not only are singularities safely handled, but — more importantly — parameter combinations to which the data are insensitive are automatically identified.

It is unusual to encounter a computational method that is this reliable and blowup-proof. I have already developed and tested matrix inversion using SVD and incorporated it into the Matrix utility class (Chapter ??). With regard to Newcomb, SVD is a “plug’n’play” capability.

In the sections that follow, we present the formalism used for our least squares estimation of the parameters from the reduced observational data. Section 2 introduces linear least squares, mainly in order to develop the formalism, since in practice we must perform a nonlinear least squares analysis. Section 3 derives the equations that give the *formal* parameter errors from a linear analysis. Section 4 then discusses the nonlinear least squares method that Newcomb uses.

## 2. LINEAR PARAMETER ESTIMATION

If the observational errors are uncorrelated, then maximum likelihood estimation becomes a simple linear least squares estimation. We cannot, of course, get away with linear least squares in actual practice, but it is useful to develop the formalism first before presenting the nonlinear least squares formalism in section 4.

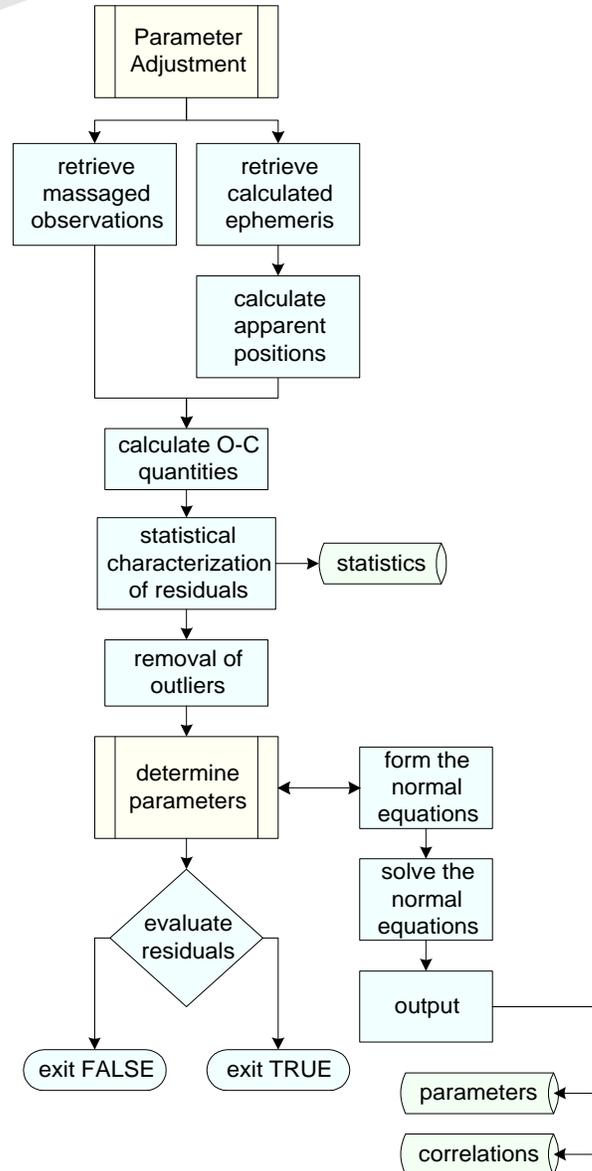


Figure 7 — Parameter adjustment module.

We will assume that the observational errors are uncorrelated and normally distributed. More precisely, the data errors are assumed to have two components: systematic errors and random errors. We assume the systematic components are modeled with bias parameters which will be estimated. It is the random components which we assume to be uncorrelated and normally distributed. Hence, the probability of a specific data-set occurring, given a model with  $n$  physical and bias parameters (a vector of length  $n$ ), is the product of the probabilities of the  $N$  individual data points:

$$P \sim \prod_{i=1}^N \exp \left[ -\frac{1}{2} \left( \frac{y_i - y(t_i, \mathbf{a})}{\sigma_i} \right)^2 \right] \quad (1)$$

where the  $y_i$  are the data and  $y(t_i, \mathbf{a})$  is the model function. Maximizing the probability means minimizing the negative of the logarithm of this expression, which within a factor of two becomes simply

$$\chi^2 \equiv \sum_{i=1}^N \frac{[y_i - y(t_i, \mathbf{a})]^2}{\sigma_i^2} \quad (2)$$

We recognize minimization of (2) as being equivalent to minimizing chi-squared, the usual linear least squares method. For notational convenience, define the summation operator

$$\{\cdot\}_i \equiv \sum_{i=1}^N (\cdot) \quad (3)$$

Thus, (2) can be written

$$\chi^2 = \left\{ \frac{[y_i - y(t_i, \mathbf{a})]^2}{\sigma_i^2} \right\}_i \quad (4)$$

When this operator is used, summation is always over the  $N$  observational data points.

Minimization of (4) requires the set of  $n$  equations

$$-\frac{1}{2} \frac{\partial \chi^2}{\partial \mathbf{a}} = \left\{ \frac{y_i - y(t_i, \mathbf{a})}{\sigma_i} \frac{\partial y(t_i, \mathbf{a})}{\sigma_i \partial \mathbf{a}} \right\}_i = 0 \quad (5)$$

to be satisfied simultaneously. Let the model function have the generalized representation

$$y(t, \mathbf{a}) = \sum_{k=1}^n a_k Y_k(t) \equiv \mathbf{a} \cdot \mathbf{Y}(t) \quad (6)$$

where the  $Y_k(t)$  are functions of time and the vector  $\mathbf{Y}(t) \equiv [Y_1(t), Y_2(t), \dots, Y_n(t)]$ . The basis functions  $Y_k(t)$  are not restricted to linearity and may have any form (polynomials, transcendental functions, etc.). The linearity that is important is in the dependence of the model function on the parameters  $\mathbf{a}$ . Equation (5) becomes

$$\left\{ \left[ \frac{y_i}{\sigma_i} - \sum_{k=1}^n a_k \frac{Y_k(t_i)}{\sigma_i} \right] \frac{\mathbf{Y}(t_i)}{\sigma_i} \right\}_i = 0 \quad (7)$$

Define the vector

$$\mathbf{S} \equiv \left\{ \frac{y_i}{\sigma_i} \mathbf{Z}(t_i) \right\}_i \quad (8)$$

and the symmetric matrix

$$A_{jk} \equiv \{Z_j(t_i) Z_k(t_i)\}_i \quad (9)$$

where, for convenience, we have set

$$\mathbf{Z}(t_i) \equiv \frac{\mathbf{Y}(t_i)}{\sigma_i} \quad (10)$$

Then (7) becomes

$$\sum_{k=1}^n A_{jk} a_k = S_j \quad (11)$$

or

$$\mathbf{A} \cdot \mathbf{a} = \mathbf{S} \quad (12)$$

Hence, the parameters are determined from the solution vector

$$\mathbf{a} = \mathbf{A}^{-1} \cdot \mathbf{S} \quad (13)$$

Equations (12) are called the *normal equations*. The matrix  $\mathbf{C} \equiv \mathbf{A}^{-1}$  is the *covariance matrix*. The covariance matrix is the key to *formal* knowledge of the errors in the parameter estimates, as well as the correlations between the various parameters. Indeed, as we shall see in the next section, the parameter errors are the diagonal elements of  $\mathbf{C}$ ,

$$\sigma_k \equiv \sqrt{C_{kk}} \quad (14)$$

The off-diagonal elements are the cross correlations,

$$\sigma_{jk} \equiv \sqrt{C_{jk}} \quad (15)$$

It must be stressed that the formal errors represent at best a lower bound on the “actual” errors, since unmodeled systematic errors pollute the process. Indeed, systematic error reduction is the entire game in this business. *Formal errors should always be viewed in this light.*

### 3. FORMAL PARAMETER ERRORS

This section contains a derivation of (14). We begin with a statement of propagation of errors, which we then use to define the formal errors of the parameters. Using the definition (9), we then arrive at (14).

Suppose we have a scalar function of  $M$  parameters,  $f(p_1, \dots, p_M)$ . Then the variation of  $f$  is

$$\Delta f = \sum_{k=1}^M \frac{\partial f}{\partial p_k} \Delta p_k \equiv \frac{\partial f}{\partial \mathbf{p}} \cdot \Delta \mathbf{p} \quad (16)$$

Now suppose  $N$  measurements of the parameters  $\mathbf{p}$  are taken and then the function  $f$  computed from these observationally determined parameters. The variance of  $f$  is then

$$\sigma_f^2 = \frac{1}{N} \{ (f_i - \langle f \rangle)^2 \}_i \quad (17)$$

Approximate  $f_i - \langle f \rangle$  by  $\Delta f$  from (16). Then

$$\begin{aligned} \sigma_f^2 &= \frac{1}{N} \{ (\Delta f)^2 \}_i = \frac{1}{N} \left\{ \left( \frac{\partial f}{\partial \mathbf{p}} \cdot \Delta \mathbf{p} \right)^2 \right\}_i \\ &= \frac{1}{N} \left\{ \sum_{k=1}^M \left( \frac{\partial f}{\partial p_k} \right)^2 \Delta p_k^2 \right. \\ &\quad \left. + \sum_{\substack{j,k \\ j \neq k}} \frac{\partial f}{\partial p_j} \frac{\partial f}{\partial p_k} \Delta p_j \Delta p_k \right\}_i \\ &= \sum_k \left( \frac{\partial f}{\partial p_k} \right)^2 \frac{1}{N} \{ (\Delta p_k)^2 \}_i \\ &\quad + \sum_{\substack{j,k \\ j \neq k}} \frac{\partial f}{\partial p_j} \frac{\partial f}{\partial p_k} \frac{1}{N} \{ \Delta p_j \Delta p_k \}_i \\ &\equiv \sum_k \left( \frac{\partial f}{\partial p_k} \right)^2 \sigma_k^2 + \sum_{\substack{j,k \\ j \neq k}} \frac{\partial f}{\partial p_j} \frac{\partial f}{\partial p_k} \sigma_{jk}^2 \end{aligned} \quad (18)$$

If the observations are uncorrelated, the cross terms (double sum) tend to cancel. Equation (18) describes the propagation of errors.

Now consider our least squares parameter estimation, eq. (13). The parameters  $\mathbf{a}$  are quantities determined from  $N$  measurements,  $[y_i]$ , with measurement errors  $\sigma_i$ . In light of

(18), and with a slight abuse of vector notation, we may write the parameter variance as

$$\sigma_a^2 = \left\{ \left( \frac{\partial \mathbf{a}}{\partial y_i} \right)^2 \sigma_i^2 \right\}_i + \left\{ \left\{ \frac{\partial \mathbf{a}}{\partial y_i} \frac{\partial \mathbf{a}}{\partial y_j} \sigma_{ij}^2 \right\}_j \right\}_i \quad (19)$$

We will assume that the observational errors are individually uncorrelated,  $\sigma_{ij} = 0 \forall i \neq j$ . Then we are left with

$$\sigma_a^2 = \left\{ \left( \frac{\partial \mathbf{a}}{\partial y_i} \right)^2 \sigma_i^2 \right\}_i \quad (20)$$

Now, from (13) and (8) we have

$$\mathbf{a} = \mathbf{C} \cdot \mathbf{S} = \mathbf{C} \cdot \left\{ \frac{y_i}{\sigma_i} \mathbf{Z}(t_i) \right\}_i \quad (21)$$

Thus,

$$\frac{\partial \mathbf{a}}{\partial y_i} = \frac{1}{\sigma_i} \mathbf{C} \cdot \mathbf{Z}(t_i) \quad (22)$$

Hence, (20) becomes

$$\sigma_a^2 = \{ (\mathbf{C} \cdot \mathbf{Z}(t_i))^2 \}_i \quad (23)$$

The  $k^{\text{th}}$  component of (23) may be written

$$\sigma_{a_k}^2 = \{ [\mathbf{C} \cdot \mathbf{Z}(t_i)]_k [\mathbf{C} \cdot \mathbf{Z}(t_i)]_k \}_i \quad (24)$$

Since  $\mathbf{C}$  is symmetric,

$$\sigma_{a_k}^2 = \{ [\mathbf{C} \cdot \mathbf{Z}(t_i)]_k [\mathbf{Z}(t_i) \cdot \mathbf{C}]_k \}_i \quad (25)$$

Interchange the order of summations:

$$\sigma_{a_k}^2 = [\mathbf{C} \cdot \{ \mathbf{Z}(t_i) \mathbf{Z}(t_i) \}_i \cdot \mathbf{C}]_{kk} \quad (26)$$

Using (9), we have

$$\begin{aligned} \sigma_{a_k}^2 &= [\mathbf{C} \cdot \mathbf{A} \cdot \mathbf{C}]_{kk} \\ &= [\mathbf{C} \cdot \mathbf{C}^{-1} \cdot \mathbf{C}]_{kk} \\ &= [\mathbf{C}]_{kk} \end{aligned} \quad (27)$$

which completes the derivation of (14).

#### 4. NONLINEAR PARAMETER ESTIMATION

Consider the condition equations (5). If the model function is nonlinear in the parameters  $y(t, \mathbf{a})$  then the simple separation, eq. (6), which led to the easily-solved normal equations, (12), is no longer possible. Our goal is to minimize (4) with respect to variation of the model parameters, despite the nonlinear dependence of the model function on the parameters. To do this, we will adopt an iterative approach.

Let  $\mathbf{a} \equiv \bar{\mathbf{a}} + \Delta \mathbf{a}$ , where  $\bar{\mathbf{a}}$  are the true (or, more accurately, best) values of the parameters. Assume we start with parameter values that are reasonably close to the best values. We can then approximate (4) with a truncated Taylor series:

$$\chi^2(\mathbf{a}) \approx \chi^2(\bar{\mathbf{a}}) - \Delta \mathbf{a} \cdot \mathbf{B} + \frac{1}{2} \Delta \mathbf{a} \cdot \mathbf{M} \cdot \Delta \mathbf{a} \quad (28)$$

where

$$\mathbf{B} \equiv - \left. \frac{\partial \chi^2}{\partial \mathbf{a}} \right|_{\mathbf{a}=\bar{\mathbf{a}}} \quad \text{and} \quad [M]_{ij} \equiv \left. \frac{\partial^2 \chi^2}{\partial a_i \partial a_j} \right|_{\mathbf{a}=\bar{\mathbf{a}}} \quad (29)$$

Within the paraboloidal approximation (28), the correction vector  $\Delta \mathbf{a}$  which minimizes  $\chi^2(\mathbf{a})$  is given by the value for which the parameter correction gradient vanishes:

$$\left. \frac{\partial \chi^2(\mathbf{a})}{\partial \Delta \mathbf{a}} \right|_{\mathbf{a}=\bar{\mathbf{a}}} \approx -\mathbf{B} + \Delta \mathbf{a} \cdot \mathbf{M} \quad (30)$$

Hence,

$$\Delta \mathbf{a} = \mathbf{M}^{-1} \cdot \mathbf{B} \quad (31)$$

where we have used the fact that  $\mathbf{M}$  is symmetric. For  $\mathbf{B}$  and  $\mathbf{M}$  we calculate

$$\mathbf{B} = \left\{ \frac{y_i - y(t_i, \mathbf{a})}{\sigma_i^2} \frac{\partial y(t_i, \mathbf{a})}{\partial \mathbf{a}} \right\}_{\mathbf{a}=\bar{\mathbf{a}}} \Big|_i \quad (32)$$

and

$$[M]_{jk} = \left\{ \frac{y_i - y(t_i, \mathbf{a})}{\sigma_i^2} \frac{\partial^2 y(t_i, \mathbf{a})}{\partial a_j \partial a_k} \right\}_{\mathbf{a}=\bar{\mathbf{a}}} \Big|_{a=j} \Big|_{a=k} - \left. \frac{1}{\sigma_i^2} \frac{\partial y(t_i, \mathbf{a})}{\partial a_j} \right|_{\mathbf{a}=\bar{\mathbf{a}}} \left. \frac{\partial y(t_i, \mathbf{a})}{\partial a_k} \right|_{\mathbf{a}=\bar{\mathbf{a}}} \Big|_i \quad (33)$$

Unfortunately, although the merit function is unitless, the elements of eqs. (32) and (33) are not unitless (in general). This can lead too easily to an ill-conditioned matrix when the parameters exhibit numerically widely disparate units. Therefore, following Hessler et al.<sup>5</sup>, let us redefine  $\mathbf{B}$  and  $\mathbf{M}$  in a unitless fashion:

$$\begin{aligned} B_k &\equiv - \tilde{a}_k \left. \frac{\partial \chi^2}{\partial a_k} \right|_{\mathbf{a}=\bar{\mathbf{a}}} \\ &= \tilde{a}_k \left\{ \frac{y_i - y(t_i, \mathbf{a})}{\sigma_i^2} \frac{\partial y(t_i, \mathbf{a})}{\partial a_k} \right\}_{\mathbf{a}=\bar{\mathbf{a}}} \Big|_i \end{aligned} \quad (34)$$

and

$$\begin{aligned} [M]_{jk} &\equiv \tilde{a}_j \tilde{a}_k \left. \frac{\partial^2 \chi^2}{\partial a_j \partial a_k} \right|_{\mathbf{a}=\bar{\mathbf{a}}} \\ &= \tilde{a}_j \tilde{a}_k \left\{ \frac{y_i - y(t_i, \mathbf{a})}{\sigma_i^2} \frac{\partial^2 y(t_i, \mathbf{a})}{\partial a_j \partial a_k} \right\}_{\mathbf{a}=\bar{\mathbf{a}}} \Big|_{a=j} \Big|_{a=k} - \left. \frac{1}{\sigma_i^2} \frac{\partial y(t_i, \mathbf{a})}{\partial a_j} \right|_{\mathbf{a}=\bar{\mathbf{a}}} \left. \frac{\partial y(t_i, \mathbf{a})}{\partial a_k} \right|_{\mathbf{a}=\bar{\mathbf{a}}} \Big|_i \end{aligned} \quad (35)$$

To keep a unitless form for the merit function, define

$$\delta a_k \equiv \frac{a_k - \tilde{a}_k}{\tilde{a}_k} \quad (36)$$

Using (34) and (35), we can rewrite (28) as

$$\chi^2(\mathbf{a}) \approx \chi^2(\bar{\mathbf{a}}) - \delta \mathbf{a} \cdot \mathbf{B} + \frac{1}{2} \delta \mathbf{a} \cdot \mathbf{M} \cdot \delta \mathbf{a} \quad (37)$$

Requiring

$$\left. \frac{\partial \chi^2(\mathbf{a})}{\partial \delta \mathbf{a}} \right|_{\mathbf{a}=\bar{\mathbf{a}}} \approx -\mathbf{B} + \delta \mathbf{a} \cdot \mathbf{M} = 0 \quad (38)$$

we have, finally,

$$\delta \mathbf{a} = \mathbf{M}^{-1} \cdot \mathbf{B} \quad (39)$$

where  $\mathbf{B}$  and  $\mathbf{M}$  are given by (34) and (35). Notice that the form of eqs. (39) is identical to that of eqs. (13). The procedure, then, is:

- ▶ Start with an initial set of values for the parameters.
- ▶ Calculate the correction via eqs. (39).
- ▶ Correct the parameter values.
- ▶ Repeat until the values stop changing significantly.

<sup>5</sup> J.P. Hessler, D.H. Current, and P.J. Ogren (1996), "A new scheme for calculating weights and describing correlations in nonlinear leastsquares fits", Computers in Physics 10, 186.

# CHAPTER 4: THE INTEGRATION MODULE

MARC A. MURISON  
 Astronomical Applications Department  
 U.S. Naval Observatory, Washington, DC

## 1. PHYSICAL MODEL

### 1.1. Newtonian Formulation

### 1.2. General Relativistic Formulation

### 1.3. The Lunar Orbit

## 2. NUMERICAL INTEGRATION ALGORITHMS

## 3. AN OBJECT-ORIENTED APPROACH

## 4. ANALYSIS SUBMODULES

### 4.1. Automatic Detection of Mean-Motion Resonances

### 4.2. Close Approach Detection

### 4.3. Frequency Analysis

## 5. EPHEMERIS GENERATION

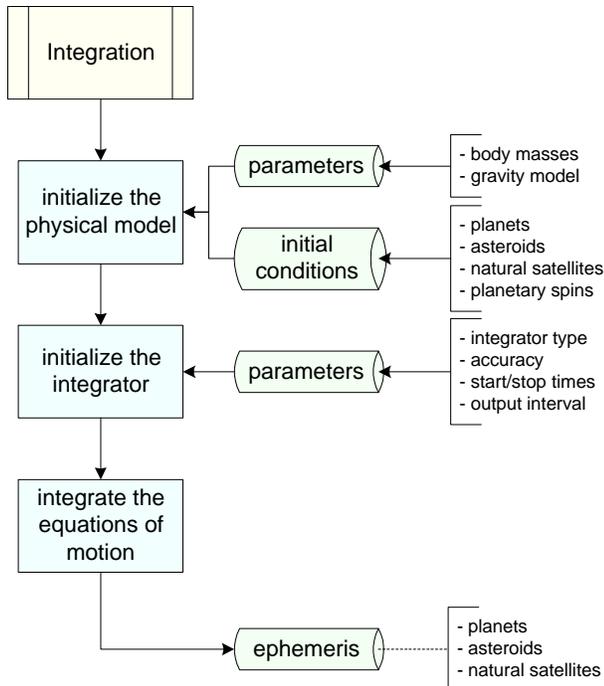


Figure 8 — The integration module.

## CHAPTER 5: THE USER INTERFACE

MARC A. MURISON  
Astronomical Applications Department  
U.S. Naval Observatory, Washington, DC

### 1. INTRODUCTION

### 2. MAIN PROGRAM INTERFACE

### 3. OBSERVATION MODULE INTERFACE

### 4. INTEGRATION MODULE INTERFACE

#### *4.1. Solar System Settings*

#### *4.2. General Relativity Settings*

#### *4.3. Integrator Settings*

#### *4.4. I/O Settings*

#### *4.5. Initial Conditions*

#### *4.6. ICs Database Capabilities*

#### *4.7. Runtime Graphics*

### 5. PARAMETER ADJUSTMENT MODULE INTERFACE

## INDEX

- C**
- condition equations, 10
  - covariance matrix, 9
- D**
- Design Issues, 4
  - Doppler observations, 7
- E**
- extensibility, 7
- F**
- Figure 1, 5
  - Figure 2, 6
  - Figure 3, 7
  - Figure 4, 8
  - Formal Parameter Errors, 9
- H**
- Hessler, 10
- I**
- Integration Module, 11
- J**
- JPL, 3
- L**
- Linear Parameter Estimation, 8
  - LLR, 7
  - Lunar Orbit, 11
- M**
- Matrix utility class, 8
  - maximum likelihood estimator, 8
  - merit function, 10
  - model function, 10
- N**
- Newtonian Formulation, 11
  - Nonlinear Parameter Estimation, 10
  - normal equations, 8, 9
  - Numerical Integration Algorithms, 11
- O**
- object-oriented, 3
  - Observation Types, 6
  - Observations, 6
  - Observations Module, 6
  - Observing Platforms, 6
  - O-C, 6, 8
  - Optical observations, 6
- P**
- Parameter Adjustment Module, 8
  - PEP, 3. *See Also* Planetary Ephemeris Program
  - Physical Model, 11
  - Planetary Ephemeris Program, 3
  - Project Outline, 4
  - propagation of errors, 9
- R**
- Radar closure, 7
- S**
- Science Issues, 4
  - singular value decomposition, 8
  - SVD, 8. *See Also* singular value decomposition
- T**
- Time delay observations, 7
  - Transits, 6
- U**
- User Interface, 12