A toolkit approach to solar radiation modelling

Mr. B. Davison[1]*, Mr. Y. Tham[2], Prof. T. Muneer[2]

[1]School of Computing, Edinburgh Napier University, Edinburgh, UK.

[2]School of Engineering, Edinburgh Napier University, Edinburgh, UK.

*Corresponding author: b.davison@napier.ac.uk

Abstract

Considerable effort is required to implement solar radiation models in software. Many existing implementations have efficiency as their main priority rather than re-usability, and this can adversely affect their further development since the relationships between the software and physical quantities may be obscured. The Solar Toolkit is an attempt to overcome such barriers by exploiting the current abundance of computing resource, and the availability of user-oriented tools such as Microsoft Excel®. The Solar Toolkit takes the form of a set of functions written in Visual Basic for Applications® (VBA) made available under the Academic Free Licence. Transparency is the overriding priority throughout the implementation so that the Toolkit can provide a platform for further modelling initiatives.

1 Background

Accurate estimates of solar radiation are important for many domains of activity including

- Photobiology and crop management

- Animal husbandry
- Energy generation, and electricity in particular
- Climatology and the mitigation of global warming
- Building design for thermal comfort and illumination

Over the last century, the mathematical understanding of the behaviour of solar radiation passing through the atmosphere has developed to a significant level of sophistication, and there is currently a great deal of activity surrounding the ongoing development, refinement and validation of competing models. It is important, therefore, that the effort going into such initiatives is invested in assets that can be re-used, adapted and further developed. Presently, this is not always the case and a number of problems with software implementations of solar radiation models can be identified.

1.1 Sources

When studying the background to a particular model with a view to implementation in software, a great deal of effort is often required to locate an authoritative version of the required equations. One reason for this is simply that the original papers are too old to be accessible through the usual electronic channels (eg. Woolf, 1968). On other occasions, published accounts may be abbreviated in order to focus on the novel contribution of the author, rather than covering background material in full detail.

1.2 Product orientation

Although many solar radiation modelling tools have been developed, the majority tend to aim towards a polished implementation of a particular model for a particular purpose. This approach leads to a usable product, but can on the other hand limit the re-usability of the code. The immediate purpose of the implementation takes precedence over general applicability in the mind of the developer, which can lead to a great deal of rework as future developers are obliged to re-invent wheels as they go along.

1.3 Third generation programming languages

Previous initiatives in solar modelling have made use of a wide range of computing languages, but perhaps the most effort has been put into the development of applications in FORTRAN. Like Pascal, C and Java, FORTRAN programs make no specific reference to the hardware on which they run. For this reason, they need to be processed by a compiler in order to produce an executable file that is appropriate for the user's specific computer. Such languages are known as "third generation" because of the greater usability they provide compared to lower-level binary and assembly code (Meehan, 1990). This usability is only apparent, however, at the program level. Obtaining and configuring a compiler can still add significantly to the difficulty and cost of providing an appropriate development environment. Although the source code for a FORTRAN application can be provided, such programs are often monolithic in structure and require a familiarity with the conventions of FORTRAN. Both of which can be significant barriers to understanding and further developing the code.

1.4 Programming culture

As the first language to provide a significant level of control, abstraction and portability to program development, FORTRAN became extremely popular in the scientific community after its invention in the late 1950s. At that time and for some decades afterwards, a high priority in program design was efficiency, so that an acceptable performance could be extracted from the available computing resources which were extremely weak when compared to those of today. Random access memory, for example, was a major bottleneck, and programmers had to be careful to manage the 10-20Kb at their disposal effectively. Today's desktop computers in contrast typically have several gigabytes of memory, a factor of about 10^6 more than those early systems. The effect of the resource restrictions was to engender a number of conventions in the way programs were written. Variable names, for example, were typically abbreviated to very short codes in order to conserve memory space. Similarly, programs were often structured in as succinct a way as possible for the sake of efficiency. The unintended effect of this preoccupation is that the correspondences between program variables and the physical quantities they represent are often obscured. Such practices can therefore act as a barrier to the effective interpretation by one programmer of code written by another.

1.5 Alternative technologies

In addition to FORTRAN, solar position and solar radiation modelling initiatives have made use of a range of technologies including for example FORTRAN with a Microsoft Excel interface (Gueymard, 1994), QBASIC (Burnett, 1997), Javascript and VRML (Noble *et al.*, 1998), C (Neteler & Mitasova, 2002) and Flash (Marsh, 2008).

As noted earlier, the abundance of computing resource now liberates programmers from many of the restrictions mentioned above. In addition to more powerful hardware, developers now have access to tools such as Microsoft Excel which provides an extremely intuitive interface for handling large sets of data and performing simple calculations. As such, Excel is a very convenient basis on which to implement predictive models. The ITACA photovoltaic system sizing calculator (ITACA, 2005) is an example which relies on the standard features of Excel. While this approach neatly packages the tool's main purpose by providing a series of fields which streamline and simplify user input, it obscures the more complicated calculations by spreading them across a series of cells in the spreadsheet.

Other Excel-based tools such as SolRad, maintained by the Department of Ecology of the State of Washington (Pelletier, n.d.) implement a set of functions using Visual Basic for Applications (VBA). Although this requires a level of programming skill beyond a familiarity with the standard Excel interface, it is possible in this environment to format the VBA code such that it parallels very closely the equations that can be found in research articles and textbooks, thus making a VBA implementation essentially self-documenting. SolRad implements the functions required for Bird and Hulstrom's solar radiation model (Bird & Hulstrom, 1981), and provides input and output worksheets to simplify the use of that model.

In an example of good programming practice, SolRad uses a set of naming conventions for functions and variables which facilitate their interpretation. Variable names are carefully chosen to parallel the conventional names for physical quantities familiar from published material, thus enhancing the transparency of the code. Apart from making the

VBA code itself easier to interpret, this same discipline applied to function names also facilitates the use of the new functions in the spreadsheet itself where they can be placed in cells just like any other. This effectively creates a non-procedural fourth generation tool for solar radiation modelling (Meehan, 1990).

Making even greater use of the features of the Microsoft environment, Sunlit Design provides a set of solar calculation functions packaged as a Windows dynamic link library (dll) file entitled the SUN API (application programmer interface) (Sunlit Design, 2001). These functions can be accessed from within a number of different programming environments including VBA by placing some simple reference code in an appropriate place. Although this provides a very simple way to use the functions and is not restricted to a single environment such as Excel, the approach means that the implementation details are not accessible, and cannot therefore be further developed by other programmers. The implementation is therefore a fixed quantity which serves a limited purpose. A further disadvantage is that it requires maintenance – the current version (02.02.04) dating from 2001 performs as expected using Excel 2003, but cannot be accessed from Excel 2007.

2 The Solar Toolkit

2.1 Guiding principles

The main purpose of the Solar Toolkit is to facilitate the implementation of solar radiation models in software. The three main priorities, therefore, are that the Toolkit should be easy to use, that it should support a generic modelling process, and that its individual elements should be re-usable from one implementation to the next. In this way,

the Toolkit becomes a growing repository of code that is constantly extended by the solar radiation research community as further additions are made. The next few sections discuss some of the design decisions that were made in order to support these principles.

2.2 Programming model

A common standard in software development is the object-oriented approach, in which software "objects" are defined in terms of their characteristics ("properties") and the operations that they support ("methods"). Object-orientation is intended to make it easier to re-use software components because an object's internal implementation is independent of its interface with other objects (Ambler, 2001). Although the VBA environment can support the object-oriented programming model, this has not been used for the Solar Toolkit in order to maintain accessibility for developers whose main specialisation is not programming. Object-orientation brings with it a set of concepts and technical terms which need to be understood in order to use the model to its full advantage. It was felt in this case that a functional approach was a more natural fit with the subject matter. Functions written in VBA parallel closely the scientific descriptions of physical relationships which facilitates their understanding, and functions are also re-usable in different contexts provided that care is taken in their definition.

2.3 Efficiency vs. re-usability

In the Solar Toolkit, transparency takes precedence over efficiency. This can lead to a certain degree of repetition and redundancy in the code. Because of the easy availability of computing resource, however, the need for optimum efficiency is no longer as pressing as it was in the past, and a trade-off in favour of transparency is much more acceptable.

Likewise, a preoccupation with delivering an optimised interface may help the user in the specific case, but the result may not be applicable in another implementation. The Solar Toolkit therefore provides some very generic features for data cleaning, for example, which paradoxically do not provide an optimum user experience in any one particular case, but which provide consistency from one case to the next.

2.4 Naming conventions

The Solar Toolkit sets out to make the use of its component functions as easy as possible, whether they are used directly in a spreadsheet or as part of a new VBA implementation. Consistent conventions for the naming of functions and their arguments have been adopted for this reason. Names of functions and subroutines are in uppercase, and where the name has several parts, they are joined with an underscore character "_". The names of function arguments are mainly in lowercase with no underscores. Where an argument name has several parts, the right-hand part begins with a capital letter. The following example illustrates the use of these conventions:

```
Function DAY_ANGLE(dayNumber As Long)
```

The immediate effect is that the purpose of the function and the meaning of each argument are immediately obvious when looking at the function definition. Again, in contrast to earlier times, there is no longer any advantage to abbreviating names: memory is abundant and even very long names can be used in full.

This approach also exploits the default behaviour of Microsoft Excel. User-defined functions can be pasted into a cell in the spreadsheet just like standard functions, and Excel provides a series of user dialogs to help the user complete the operation. Fig. 1 illustrates the use of transparent names to make clear to the user the quantities that are required.

2.5 Process support

In a typical modelling scenario, a mathematical algorithm is applied to a set of data, iterating over each record in turn in order to produce a predicted result. In the simplest case where the model is already considered reliable, the direct relationship between input data and predicted result can be represented by the diagram shown in Fig. 2. An example of this would be the use of the Yang (2001, 2006) model to predict solar irradiation using meteorological data as input.

In certain cases, a particular calculation within a model might require additional input data. In such as case, the process would essentially have the same structure as shown in Fig. 2. The calculation of the Ångström turbidity coefficient in the Yang (2001) model provides an example of this, where the input values are provided in the form of a lookup table from which appropriate values are selected. The use of this calculation could be represented by the diagram in Fig. 3.

In practice, the dependent calculation may be called many times during the application of the model – typically once for every record in the model input data. Fig. 3 is laid out to emphasise the similarity in program structure rather than the flow of control.

A more complex situation occurs where the purpose of the implementation is to compare the results of a model with actual measured data. This is a typical requirement in research studies that set out to evaluate the operation of a model, such as the one described in Tham *et al.* (2009). In this case, there may be a further set of input data as shown in Fig. 4.

A further extension of the structure can be envisaged for a comparison between two different models as shown in Fig. 5 in which the comparison step involves three sets of data. Because the same process structure is repeated, it is possible to provide generic components within the Solar Toolkit for handling the requirements at all levels.

In its current form, the Solar Toolkit provides a generic mechanism for managing input data sets via the default "Data" worksheet. This sheet, which can be duplicated and renamed as many times as required using the standard Excel interface, provides a basic data cleaning function to check for anomalous values. The user specifies parameters for each column of data, and code checks that each entered value

- is of the correct datatype
- is greater than the specified minimum
- is less than the specified maximum

The current support for process structure in the VBA code is less developed and relies mainly on the adaptation of the examples supplied. The main subroutine included in the Solar Toolkit template illustrates the iteration through two sets of data whose records are matched and combined in the results. Future versions of the Toolkit could be developed to include generic structures to support some of the patterns described in Figs. 2-5. The

use of design patterns in software development is currently a growth area (Lasater, 2006), and could be used to further simplify the creation of new model implementations. Table 1 summarises the purpose of the worksheets in the current version of the Toolkit.

2.6 Alternative implementations

Several typical usage scenarios have already been mentioned above, and as a growing repository of code, the Solar Toolkit also offers additional advantages for education and for flexibility in particular implementations. In several cases, the Toolkit contains several functions for the same quantity based on different published algorithms, such as declination and the equation of time. It is therefore a simple matter to compare the results from these alternative implementations. Where alternatives exist, a "preferred" implementation is identified and named according to the quantity that it returns. The alternatives are given the same base name, but also include the author's name to distinguish them. For example, the preferred DECLINATION function is based on Yallop's (1992) algorithm, while DECLINATION_SPENCER provides and alternative based on Spencer's (1971) algorithm.

The availability of alternatives also allows components to be substituted for each other in certain contexts. In the development of the Yang model spreadsheet which is based on the implementation used in Tham *et al.* (2009), it was found that substituting the Gueymard function for precipitable water (Gueymard, 1994) gave a small improvement over that used by Yang (2006).

3 Future development

The Solar Toolkit is made available under the terms of the Academic Free Licence, which is an example of many licences approved by the Open Source Initiative (OSI, n.d.). The Open Source movement started as a way to break free from the restrictions on software development and use imposed by large companies, and instead to take advantage of the large pool of skills that exists in the community. The model of community development appears to be a good fit with the nature of solar modelling efforts, and developers are invited to make their own contributions to future versions of the Solar Toolkit. The current version is available on the ETRESH Web site (http://www.etresh.eu/downloads.htm).

The implementation of the Yang (2001, 2006) model using the Solar Toolkit is also available from the ETRESH Web site. It provides a ready-made means to estimate solar irradiation for any location with predominantly clear-sky conditions using commonly-available meteorological data as input (Tham *et al.*, 2009).

References

Ambler, S.W. (2001) The Object Primer: The Application Developer's Guide to Object-Orientation and the UML (2nd Ed.) Cambridge University Press

Bird, R.E. and Hulstrom, R.L. (1981) A Simplified Clear Sky model for Direct and Diffuse Insolation on Horizontal Surfaces. SERI Technical Report SERI/TR-642-761. Solar Energy Research Institute, Golden, CO.

Burnett, K. (1997) Approximate astronomical positions. Accessed 2 August 2009 at http://www.stargazing.net/kepler/

Gueymard, C., 1994. Analysis of monthly average atmospheric precipitable water and turbidity in Canada and northern United States. Solar Energy 53(1), pp. 57–71.

ITACA (2005) Advanced Solar Power Design Spreadsheet. Available online at http://www.itacanet.org/eng/elec.htm

Lasater, C.G. (2006) Design Patterns. Wordware Publishing Inc.

Marsh, A.J. (2008) Solar position calculator. Accessed 2 August at http://squ1.org/wiki/Solar_Position_Calculator

Meehan, D. (1990) An introduction to fourth generation languages. Stanley Thornes.

Neteler, M. and Mitasova, H. (2002) Open Source GIS: A GRASS GIS Approach. Kluwer International

Noble, D., Jain, A. and Kensek, K.M. (1998) VRSolar: A Web-Based Tool for Understanding Solar Movement. In Barton, C. (Ed.) 86th ACSA Annual Meeting Proceedings

OSI (n.d.) Open Source Initiative home page. Accessed 8 August 2009 at http://www.opensource.org/

Pelletier, G. (n.d.) SolRad, A solar position and solar radiation calculator for Excel/VBA. Available online at http://www.ecy.wa.gov/programs/eap/models.html

Spencer J. W. (1971). Fourier series representation of the position of the Sun. Search 2(5).

Sunlit Design (2001) The SUN API. Available online at http://www.sunlit-design.com/products/thesunapi/index.php

Tham, Y, Muneer, T and Davison, B (2009) A generalized procedure to generate clear-sky radiation data for any location. International Journal of Low-Carbon Technologies. doi:10.1093/ijlct/ctp022

Woolf, H.M. (1968) On the Computation of Solar Evaluation Angles and the Determination of Sunrise and Sunset Times. National Aeronautics and Space Administration Report NASA TM-X -164, USA.

Yallop, B.D. (1992) Technical Note, Royal Greenwich Observatory, Cambridge.

Yang K., Huang G.W., Tamai N. (2001) A hybrid model for estimating global solar radiation. Solar Energy, 70(1), pp. 13-22(10)

Yang K, Koike T, Ye B. (2006) Improving estimation of hourly, daily and monthly solar radiation by importing global data sets. Agricultural and Forest Meteorology, 137, pp.43–55.

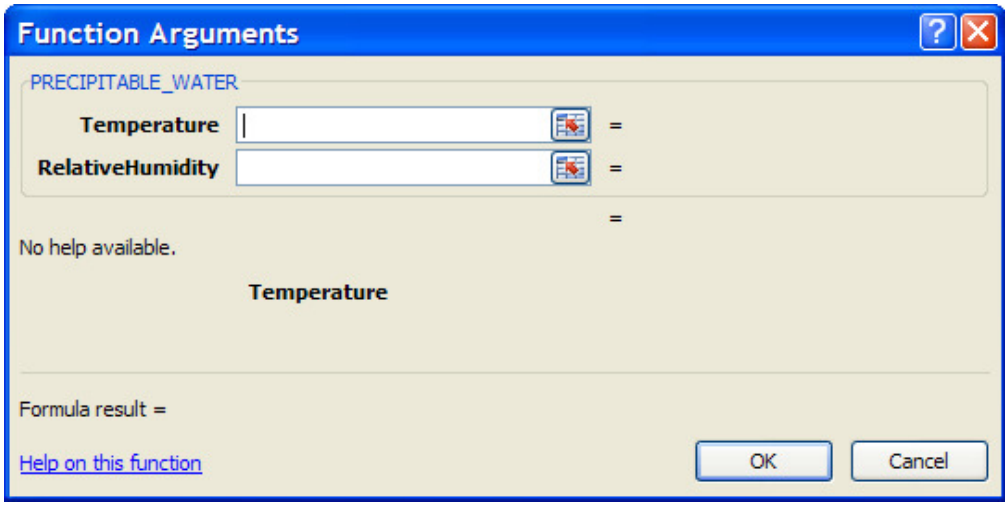

Figure 1: Example 'Insert function' dialog in Excel

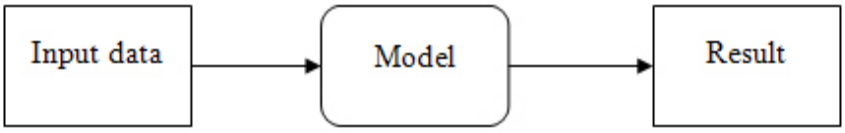

Figure 2: Simple modelling process

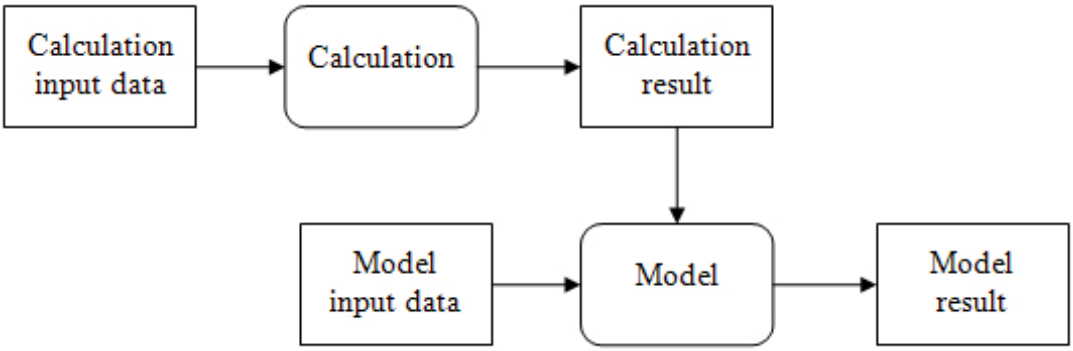

Figure 3: Use of a calculation which requires input data

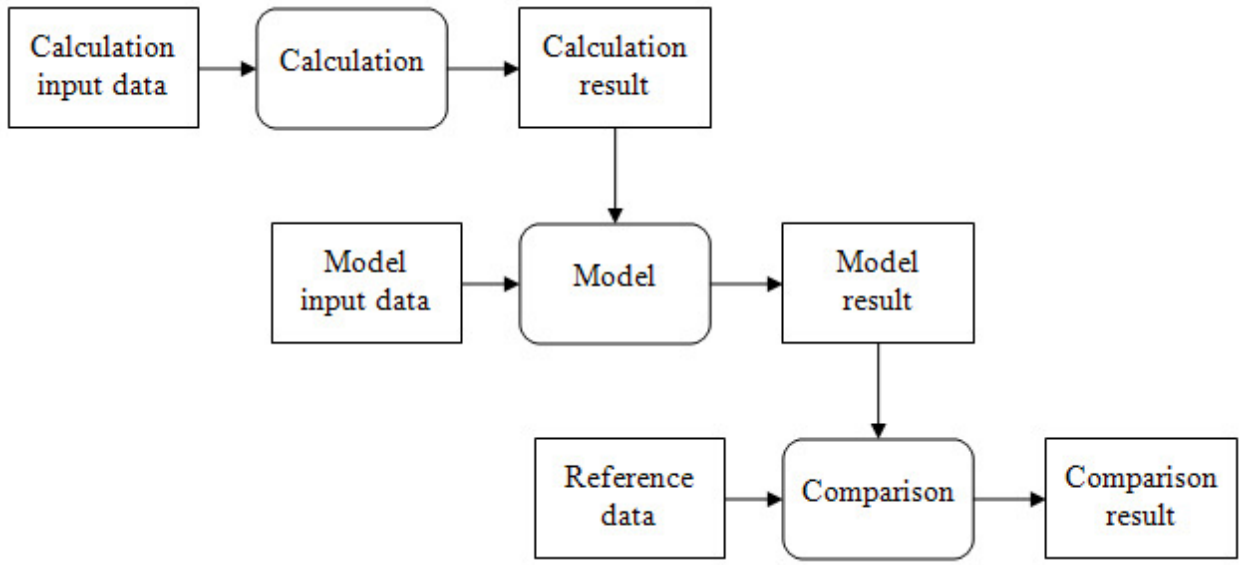

Figure 4: Process structure for model evaluation

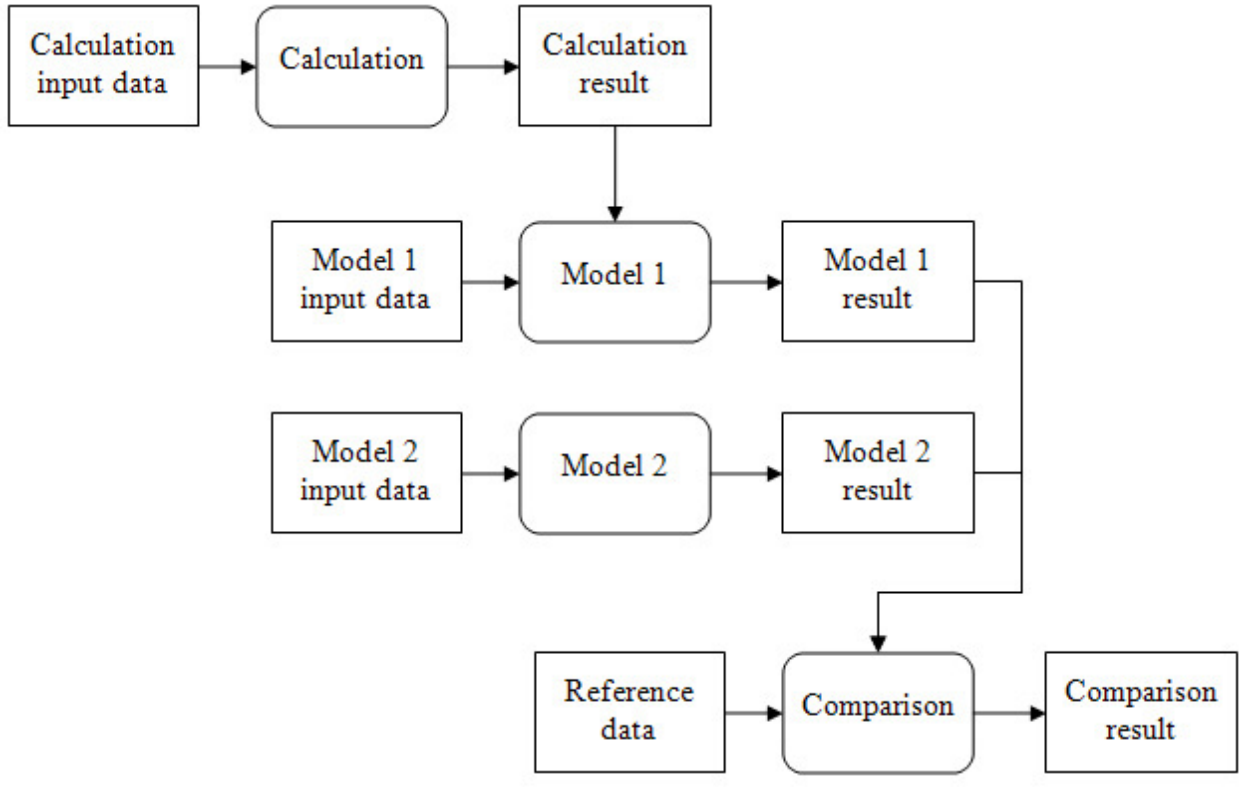

Figure 5: Process structure for model comparison

Tab name	Description
Start	Provides a location for welcome messages and basic instructions for using the particular configuration of the Solar Toolkit
Parameters	Allows the developer to provide the eventual user of the spreadsheet with a way to change fundamental values that are used in the calculations. Typical values include site-specific details such as latitude, elevation, etc. Once set, the parameter values can be accessed easily from within VBA.
Data	Holds an input data set and provides a basic cleaning function. It can be duplicated and renamed as required without losing this functionality.
Results	Accumulates the results of any calculations or comparisons. This approach keeps input data separate from the results of any calculations to maintain its integrity in case the user wishes to run the model several times with different parameters, for example.
Angstrom	Although the purpose of this worksheet is to hold input data to the BETA function, it is a special case of a data worksheet because it is not intended to change. It has therefore been protected using standard Excel security.
Log	Records messages about any errors encountered either during

	processing or related to data cleaning. Unforeseen errors are very likely when dealing with data from physical sources.
Index	Lists the functions in the Solar Toolkit with a brief description.
References	One of the purposes of the Toolkit is to make it easier to trace the origin of a particular calculation back to its source. Sources are acknowledged within the VBA code itself, and sometimes even in function names. This worksheet, however, provides a summary list of references.
Troubleshooting	Because the Toolkit takes advantage of some advanced features of Excel, there may be certain configuration parameters that need to be set in order for it to work as intended. As such issues come to light, useful information will be added to this worksheet in successive versions.

Table 1: Default worksheets included in the Solar Toolkit